Independent project (degree project), 15 credits, for the degree of Degree of Bachelor of Science (180 credits) with a major in Computer Science
Spring Semester 2020
Faculty of Natural Sciences

# Evaluating Methods for Optical Character Recognition on a Mobile Platform

## Comparing standard computer vision techniques with deep learning in the context of scanning prescription medicine labels

Jonathon Bisiach
Matej Zabkar

**Authors**
Jonathon Bisiach & Matej Zabkar

**Title**
Evaluating Methods for Optical Character Recognition on a Mobile Platform: Comparing standard computer vision techniques with deep learning in the context of scanning prescription medicine labels

**Supervisor**
Kamilla Klonowska

**Examiner**
Ola Johansson

**Abstract**
Deep learning has become ubiquitous as part of Optical Character Recognition (OCR), but there are few examples of research into whether the two technologies are feasible for deployment on a mobile platform. This study examines which particular method of OCR would be best suited for a mobile platform in the specific context of a prescription medication label scanner. A case study using three different methods of OCR – classic computer vision techniques, standard deep learning and specialised deep learning – tested against 100 prescription medicine label images shows that the method that provides the best combination of accuracy, speed and resource using has proven to be standard seep learning, or Tesseract 4.1.1 in this particular case. Tesseract 4.1.1 tested with 76% accuracy with a further 10% of results being one character away from being accurate. Additionally, 9% of images were processed in less than one second and 41% were processed in less than 10 seconds. Tesseract 4.1.1 also had very reasonable resource costs, comparable to methods that did not utilise deep learning.

**Keywords**
Optical Character Recognition, Deep Learning, Tesseract, EAST, Testing, Performance, Android

# Contents

# Abbreviations & Acronyms

| Abbreviation or acronym | Meaning | First used |
|---|---|---|
| OCR | Optical Character Recognition | 1 |
| NLM | National Library of Medicine | 1.1 |
| CCVTs | Classic Computer Vision Techniques | 1.2.1 |
| StanDL | Standard Deep Learning | 1.2.1 |
| SpecDL | Specialised Deep Leaning | 1.2.1 |
| EAST | Efficient Accurate Scene Text detector | 1.2.1 |
| CPU | Central Processing Unit | 2.2.2 |
| CSV | Comma Separated Values file | 2.2.3 |
| OSD | Orientation and script detection | 2.2.4 |
| LSTM | Long Short-Term Memory | 2.2.4 |
| DCNN | Deep Convolutional Neural Networks | 3.1 |
| CNN | Convolutional Neural Networks | 3.1 |
| RNN | Recurrent Neural networks | 3.2.2 |
| FCN | Fully-Convolutional Network | 3.2.3 |
| NMS | Non-Maximum Suppression | 3.2.3 |
| MLP | Multi-Layer Perceptron | 8.1.2 |
| RGB | Red-Green-Blue | 8.1.2 |
| ReLU | Rectified Linear Unit | 8.1.2 |

# 1 Introduction

*Optical Character Recognition (OCR)* is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text. *Deep learning* is part of a broader family of machine learning methods based on artificial neural networks that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning has become ubiquitous as part of OCR, but there are few examples of research into whether the two technologies are feasible for deployment on a mobile platform.

The purpose of this thesis is to research and analyse which approach to OCR performs the best in the context of a mobile application using image recognition. Specifically, in an application developed to accurately identify prescription drugs and to help patients in avoiding situations where prescribed medication may be harmful when taken with certain other prescription medication.

## 1.1 Problem and motivation

In 2017 more than 6.6 million Swedes took at least one prescription medication, corresponding to approximately 66% of the population. The largest proportion of these users is found in age groups 65 years and above [1]. In the US, preventable medical errors are the third leading cause of death after heart disease and cancer, with the largest subset of medical errors being medication error [2] [3]. On top of this, Sweden has an aging population – the percentage of Swedes above the age of 65 has grown from under 12% in 1960 to over 20% in 2018 [4].

In 2016 the National Library of Medicine (NLM) hosted the Pill Image Recognition Challenge as part of its research and development in Computational Photography Project for Pill Identification (C3PI). The Challenge asked for submissions from teams that would contribute to the creation of a software system that can match photos taken by a smartphone to the NLM database of high-resolution prescription pill images. The winner of this competition, Zeng et al [5] produced an application that recognised the correct pill within the top-5 results at 83% accuracy. Delgado et al [6] later demonstrated it was possible to produce results within the top-5 at 94% accuracy under comparable, though not identical configurations.

## 1.2 Background

### 1.2.1 Optical Character Recognition

One facet of image recognition technology is OCR, or the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo or from subtitle text superimposed on an image [7]. After text is detected at a line/word level there are many methods that can be used to convert the text which generally come from three main approaches [8]:

1. Classic Computer Vision Techniques (CCVTs), typically involving applying filters to make characters stand out from the background, contour detection to recognise the characters individually and image classification to identify the characters.
2. Standard Deep Learning (StanDL), using an artificial neural network that combines multiple nonlinear processing layers that uses simple elements operating in parallel.
3. Specialised Deep Learning (SpecDL) that uses such technologies as convolutional-recurrent neural networks, such as or Semi-Supervised End-to-End Scene Text Recognition (SEE) and Efficient Accurate Scene Text detector (EAST).

The first recorded uses of this technology can be traced as far back as the 19$^{th}$ century in reading devices for the visually impaired. In 1904 Emanuel Goldberg (1881 - 1970) developed a machine that read characters and converted them into standard telegraph code (and would later develop an OCR device for searching microfilm archives that would be acquired by IBM) [9], while around the same time Edmund Fournier d'Albe (1868 - 1933) developed a handheld scanner (see Figure 1) that, when moved across a printed page, produced an audio tone that corresponded to specific letters or characters [10].

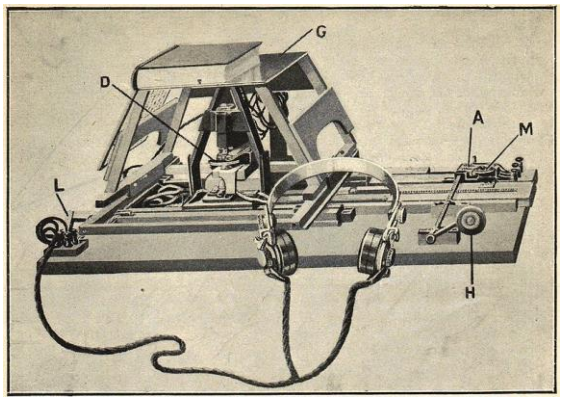


*Figure 1- A detail view of d'Albe's 'Octophone' scanning device, invented in 1913. [11]*

### 1.2.2 Deep Learning

Deep learning, also known as deep structured learning or differential programming, is a far more recent development in technology, the concept of which was first presented as a part of a broader family of machine learning methods based on artificial neural networks that imitates the workings of the human brain in processing data and creating patterns for use in decision making in 1986 [12]. Machine learning can be loosely described as a self-adaptive algorithm that gets increasingly improved analysis and patterns with experience or with newly added data, and deep learning utilises a hierarchical level of artificial neural networks to carry out the process of machine learning. The artificial neural networks are built like the human brain, with neuron nodes connected together like a web, as shown in Figure 2. While traditional programs build analysis with data in a linear way, the hierarchical function of deep learning systems enables machines to process data with a nonlinear approach. Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction [13].

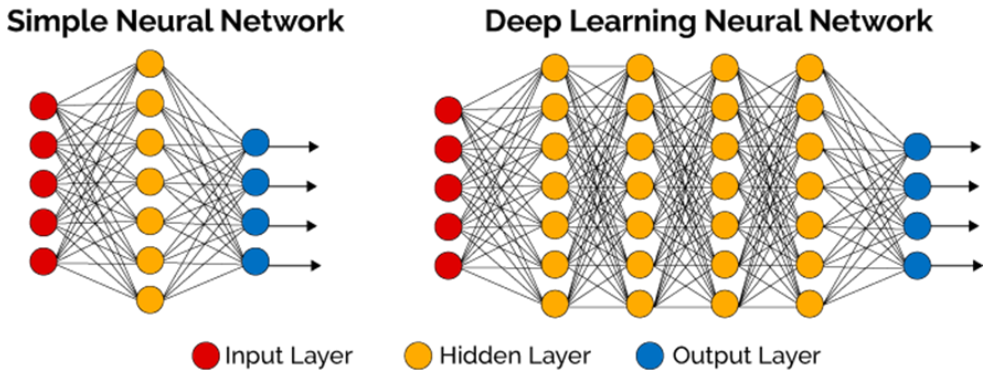


*Figure 2- Illustration of two traditional neural networks: a Single-Layer Perceptron (SLP) and a Multi-Layer Perceptron (MLP).*

One of the biggest advantages of using deep learning over standard machine learning is its ability to execute feature engineering without additional resources. In this approach, an algorithm scans the data to identify features which correlate and then combine them to promote faster learning without being told to do so explicitly, as illustrated in Figure 3. Another advantage is the elimination for the need of expensive and time-consuming data labelling as the algorithms can learn unsupervised or semi-supervised [13].

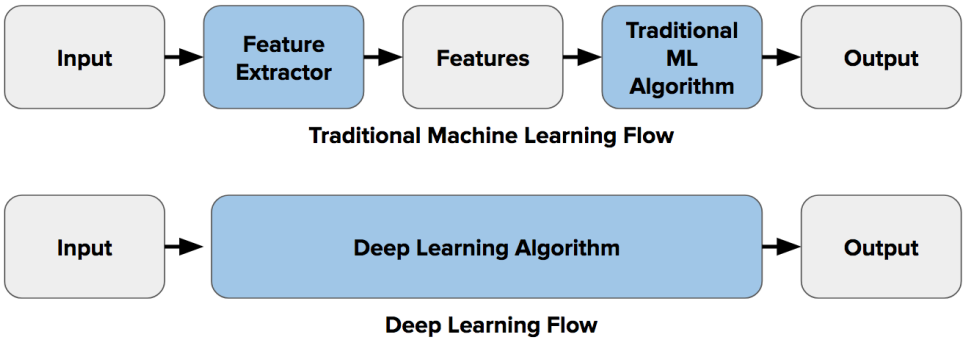


*Figure 3- Traditional machine learning flow vs Deep learning flow.*

These methods have dramatically improved the state-of-the-art in image classification, object detection, object tracking, pose recognition, video analytics, synthetic picture generation and many other domains such as drug discovery and genomics. Deep learning approaches like neural networks can be used to combine the tasks of localising text, or text detection, in an image along with understanding and converting the text into machine language, or text recognition.

## 1.3 Research questions

Given that there are several methods to accomplish OCR, our research question is:

*In the context of a mobile application using image recognition to scan prescription medication labels, which approach to OCR provides the best performance?*

In seeking to answer this question, additional arising issues will need to be addressed, such as:

- What is meant by ‘performance’ in this setting and what criteria can be used to measure it
- To what extent is it possible to design a fair comparison of approaches to reading text that are radically different, and
- What potential extraneous factors could skew the results of performance tests.

## 1.4 Aim and purpose

Incorporating OCR into a mobile application that simply and easily can scan a prescription medication container, accurately read the label and provide the user with information regarding the medication could prove to be a valuable tool in reducing the number of preventable medical errors. To that aim this research seeks to compare three methods of OCR deployed in a mobile device in order to determine not only which method provides the best performance in terms of accuracy but also in regarding memory consumption, power drain and speed.

## 1.5 Limitations

### 1.5.1 Prescription medicine labels

The dataset used in the practical case study (see 2.2.1) consists of scanned prescription medicine labels of varying size, resolution and quality. These are all flat scans in two dimensions. As such, how the three methods of OCR process non-planar or non-paper objects cannot be considered.

### 1.5.2 Hardware limitations

Testing and analysis of the three methods of OCR is to be conduction exclusively on Android mobile platforms. No consideration could be given to other platforms such as iOS. The primary mobile device used to conduct the case study is a Huawei P30 (for hardware specifications see Appendix 2: Case study details). No other Android devices could be considered due high cost and restrictions stemming from COVID-19.

### 1.5.3 Android version limitations

Testing of the three methods of OCR on the Android platform will be limited to Android version 10.0. Both in Sweden and worldwide Android versions 9.0 and 10.0 are the most commonly used, with 29.92% and 41.97% of Swedish, and 32.43% and 22.06% of worldwide Android using version 9.0 and 10.0 respectively [14] (see Figure 4). Rather than test the three methods of OCR on every Android version, the decision was made to conduct tests on only the most used version of platform in Sweden.

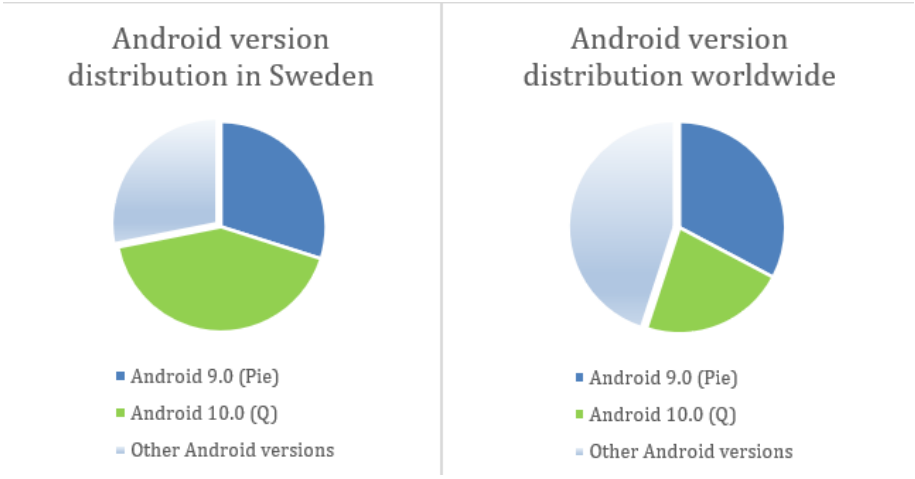


*Figure 4- Distribution of Android versions in Sweden and worldwide.*

### 1.5.4 Deep learning models

The choice was made to use pre-trained models Tesseract 4.1.1 and EAST instead of training these neural networks. This decision was made after reading Tesseract's documentation that advised that training the neural networks could potentially take weeks alone [15], and initial testing that showed the pre-trained neural networks were abundantly fit for purpose.

# 2 Methodology

This study has been conducted both through theoretical research and practical implementation of OCR and deep learning, following the acquisition of prescription medication labels forming the test dataset.

## 2.1 Literature search

Deep learning and especially OCR are by no means new technologies and thus have been the subject of a substantial amount of academic research in a wide variety of applications, both individually and in collaboration. However, as shown below, the alliance of these two technologies on mobile platforms is a more recent development, with the majority of academic research being focused more upon the feasibility and the accuracy of the results rather than performance.

A search of the following keywords on Summon@HKR yielded a substantial number of results even when filtered for peer-reviewed content. The number of results when conjoining search terms reduces significantly, as shown in Figure 5:

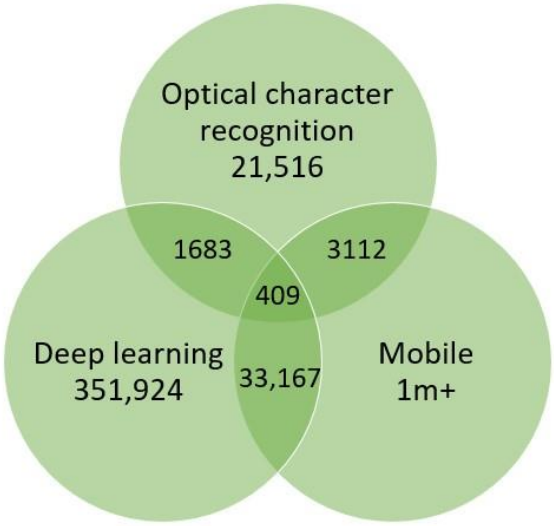


*Figure 5- The number of search results using the Summon@HKR platform showing the crossover when search using joint search terms.*

When using the conjoined search term 'optical character recognition AND deep learning AND mobile', there was a total yield of 409 articles at the time of writing. Using alternate search terms such as 'OCR', 'optical character reader', 'DL' or 'machine learning' in various combinations produced no more results than the 409 as shown in Figure 5. When the search terms used were then abutted with 'performance', this narrows the pool to 340 peer-reviewed results at the time of writing. However, of these 340, only a handful – less than five – could be considered to explore subjects related to different approaches to OCR including deep learning on mobile platforms, and only one of these considered performance as a primary concern as well as accuracy.

The Summon research was integrated with an investigation on Kristianstad University's Databases: the ACM Digital Library, ScienceDirect, SpringerLink, the Institute of Electrical and Electronics Engineers digital library and Wiley Online Library.

## 2.2 Case study design and implementation

This section describes the practical component of the thesis: a case study for testing three different methods of OCR on the Android platform to measure the accuracy and performance of each method.

One of the challenges of detecting and reading text from prescription medication labels is the same as detecting and reading text from random places in a natural scene:

- Text density: on a standard printed/written page, text is dense, while on a medicine label the text can be sparse.
- Structure of text: text on a page is structured, mostly in strict rows, while text on a medicine label be scattered and may be horizontal or vertical.
- Fonts: printed fonts in standards texts are uniform, while a medicine label may have many different, sometimes stylised fonts and text sizes.
- Artifacts: Depending on the quality of the image and the design of the label, the image can be noisier and contain more artifacts than a standard text image.
- Location: medicine labels can include cropped/centred text and text that may be located in random locations in the image [16].

This means that deep learning approaches specialised for use in text detection in natural scenes, such as OpenCV's EAST, can also be used to great effect in detecting text in images of medicine labels.

### 2.2.1 Dataset

In order to test the three methods of OCR, a large sampling of prescription medicine labels was required. Acquiring this proved to be more challenging than anticipated. Initially the Swedish organisations Apoteket and Farmaceutiska Specialiteter i Sverige (FASS) were contacted for assistance, but this proved fruitless. A substantial repository of prescription medicine labels was found online at the NLM [17], but this repository was structured in such a way that each label was contained in the form of a JPEG file in a folder along with several other image JPEGs, such as chemical compound structures, various dosage charts, manufacturer logos, and other information, as per Figure 6. Accordingly, there were 39,603 folders containing a total of 216,062 images. There was no uniform naming convention to these files to allow for simple extraction of the label images exclusively, with each folder was named in hexadecimal code according to the NLM's prescription medicine sorting Application Programming Interface (API).

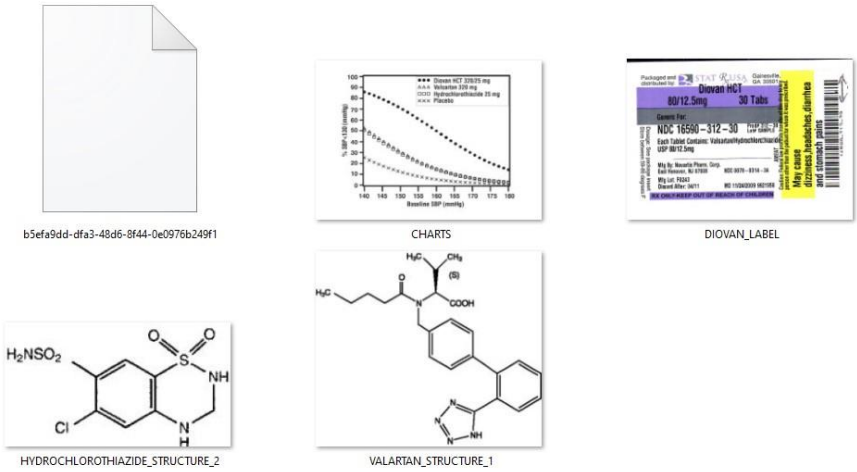


*Figure 6- An example of the content of one of 39,603 folders in the NLM repository.*

From these 216,062 images, the goal was to obtain a minimum of 10,000 viable images of prescription medication labels.

The observation was made that of each of the prescription medication labels contained a barcode. An algorithm that scans through images searching for vertical and horizontal lines, indicative of a barcode, was created in Java using Aspose.BarCode. At each pass, if no barcode has been found, the image was rotated by 45 degrees and scanned again up to a maximum of four times.

Using this method, in approximately six hours 12,455 images were filtered. Out of these, 10 were unusable corrupted JPEGs, and after a manual sorting, a further 256 images were false positives (2.06%), leaving a dataset of 12,189 label images. From this data a random selection of 100 images to use in the case study was obtained using a Java program that used a modulo method in order have the widest variety of images possible.

### 2.2.2 Testing parameters

In terms of measuring the performance of the three methods of OCR, the parameters that are being assessed as are follows:

- Time: the taken for each individual image to be scanned and the total time taken for all images in the dataset to be scanned.
- Accuracy: a sample of 100 images for each of the three methods programmatically checked against a list containing the medicine names in each image for accuracy of the text recognition, and the findings are extrapolated as a percentage. A result

was deemed positively accurate if the correct name of the medication appears in the scanned text (see Figure 7), whereupon the outputted text can be cross-referenced with a database of prescription medications in a real-world scenario. A result was deemed a 'near miss' if the output is one character away from being correct, which was checked manually.

- Central Processing Unit (CPU) usage: Average percentage of and CPU usage over the time taken to scan 100 images as well as a range of CPU use.
- Space: the minimum and maximum amounts of native memory heap used over the time taken to scan all images in the dataset, and as an average over time. Native memory was selected as a parameter over other memories as the Tesseract and EAST libraries are written in C and C++ code, and use only native memory.
- Power consumption: the amount of power consumed as a percentage of total battery capacity over 100 images scanned.


```
Denver: 15 mL
NDC01214794|5
THEOPHYLLINE
ORAL SOLUTION
80m 15 mL
Alcohoi 20°
Fox INsnYUnoNAL Usr on“
|1|||||||||||||
I||| I
(DI)
I]
030121.“
RXONLY
Subudun ol Emu ubmmm m
Mtg,wrmmlummunms in:
5: mm M0 Sam
M A» Phalmxcuhcastatmu 1M
(m‘enufle 5L mm
```

*Figure 7- An example of a prescription medicine label (left) and the OCR output (right) using Tesseract 3.05.*

The decision was made to conduct tests using 100 images rather than the full dataset of 12,189 was made due to the consistency of the results irrespective of the size of the testing sample and the time it took to process the results (see Table 7).

### 2.2.3 Android application

An application built in Android Studio serves to 'host' the three methods of optical character recognition, give an output to the scanned images in the form of plaintext and to provide performance data on each method as per a list of selected parameters (see 2.2.2). As such, the application must be capable of:

- accessing sample images stored in devices' external storage as per typical mobile photo storage
- recording the duration of processing of each individual image and the total processing duration of the sample data
- writing results to a log folder
- logging memory information, and
- recording power consumption.

To that end, the Android application designed to house the methods for OCR also measures the performance of the working application using the tools Android Profiler and Battery Historian. The tests are conducted in two phases:

1. The first test phase involves profiling performance parameters and accuracy (see 2.2.2) with the phone connected to a PC via a USB cable. This involves:
    a. starting the testing application, an Android Package (APK) file
    b. starting the Android Profiler tool on the PC
    c. commencing scans of the 100 dataset images
    d. exporting the test results to the PC
2. The second test phase is solely for measuring power consumption where the phone is not connected to a power source. This involves:
    a. resetting battery statistics
    b. commencing scans of the 100 dataset images
    c. dumping battery data onto the PC
    d. creating a bug report from the raw data
    e. importing data into the Battery Historian tool

In addition to the fact that the power consumption test could only be done with the phone not being connected to a power source, tests are done in two parts address two of the sub-questions raised in 1.3: testing all performance parameters at once significantly affected the time it took to process images during initial testing phases.

Each test outputs a log folder containing:

- scan times for each individual image are presented as a list with the corresponding image number contained in one Comma Separated Values (CSV) file
- CPU and memory usage are presented graphically using Android Profiler
- power consumption is contained in a Docker file, and
- for measuring accuracy, the resulting text output from a scanned image is checked against an array list in code imported from CSV file containing the names of medicines.

### 2.2.4 Methods for OCR: Tesseract settings

Tesseract is the OCR engine used in various applications in testing the three approaches to OCR for the case study (see 3.2). As the purpose of the experiment is to compare the performance of a CCVT with two methods of OCR that incorporate deep learning, the version that is used for the purposes of the experiment is the latest 3.0x version, 3.05 (released December 2015). Tesseract gives developers a number of options for scanning, including the Tesseract engine, segmentation and dictionary options (see Appendix 2: Case study details). In order to evaluate which options would produce the accurate results, subset of 10 images were scanned and assessed. As a result, Tesseract 3.05 was deployed using the default engine, optimised for sparse text with orientation and script detection (OSD), and the dictionary disabled.

The same segmentation developer options are available in Tesseract 4.1.1 as are in version 3.05 but with some novel engine settings introduced in version 4.1.1 (see Appendix 2: Case study details), and the same subset tests were conducted in order to evaluate which options gave the best accuracy. As a result, for testing StanDL, Tesseract 4.1.1 was deployed using the Long Short-Term Memory (LSTM) engine, optimised assuming the scanned image was a single uniform block of text, and the dictionary disabled.

In testing SpecDL, Tesseract 4.1.1 was deployed using the LSTM engine, optimised assuming the scanned image was a single word of text, as per the bound boxes provided by EAST, and the dictionary disabled.

# 3 Literature review

The literature review aims at investigating previous work in the field as well as establishing the foundation for the proposed case study in consideration of the scarcity on academic research given to this specific area.

## 3.1 Related work

In 2019 Benaddy et al. employed Deep Convolutional Neural Networks (DCNN) (see Appendix 1: Neural network terminologies) in order to improve the accuracy in identifying Tifinagh characters [18]. This technique involves using a deep learning algorithm which can take in an input image, assign importance to various aspects/objects in the image and be able to differentiate one image from the other. This proposed system was tested on data set of approximately 25,000 and achieved the best recognition accuracy (99.10%) when compared to other established OCR methods such as the horizontal and vertical centerline or baseline of characters. However, this was an improvement of only 0.07% on the previously most accurate method of a combination of multiple classifiers with statistical features.

Roy et al. had already adopted and improved on this principle in 2017 when they employed a layer-wise technique of DCNN in order to improve the accuracy in identifying Bangla characters [19]. This technique DCNN incorporated with the layer-wise training model, which is a multi-stage process that involves adding layers of convolutional and pooling processes followed by fully connected layers which contain multiple neurons, and applies the back-propagation algorithm to find the weights of importance. This layer layer-wise-trained DCNN produced results that improved upon standard DCNN nearly 10%.

In 2018 Jangid & Srivastava performed a similar study using Devanagari [20]. This layer-wise-trained DCNN produced results that improved upon standard DCNN by up to 1.5% across sample sizes of up to almost 57,000 characters.

However, as noted in 3.2.2, the additional resources required for the use of neural networks in conjunction with OCR can be considerable. Yet, there seems to be scarcity of research into the application of deep learning in OCR in regards to the performance costs of doing so, and whether this is even feasible on a mobile platform integrated into a mobile application that expects real-time results.

Yin et al. expanded on Jangid & Srivastava's researched further in 2019 using deep learning techniques to improve existing OCR approaches for recognising Chinese uppercase character by considering network weight, and test time [21]. Generally, the deeper the number of layers of the neural network and the more parameters, the more accurate the conclusions are, but accurate results mean more computing resources are consumed. In their study Yin et al. reduced overhead through the practice of 'pruning', or the removal of parameters that do not contribute significantly to the output. This is achieved by identifying the most redundant neurons using an Average Percentage of Zeroes (ApoZ) algorithm and removing some of the unnecessary network neurons and retaining the weight parameters which are important to the network and reducing the parameters in order to reduce the computational complexity of the model. Using this method, accuracy decreased by 1.26% when compared with a standard Convolutional Neural Network (CNN) but achieved a network weight reduction of 96.5%.

Likewise, Valueva et al. researched two techniques in combination to reduce hardware costs in the implementation of CNN architecture [22]. In their application of a CNN, the neural network is divided into distinct hardware and software partitions to increase performance and reduce the cost of implementation resources. They combined this with novel residue number system in the hardware part of the convolutional layer of the CNN to implement the convolutional layer of the neural network. A residue numeral system is a numeral system representing integers by their values modulo several pairwise coprime integers called the moduli. This 'multi-modular arithmetic' system is widely used for computation with large integers, typically in linear algebra, because it provides faster computation than with the usual numeral systems, even when the time for converting between numeral systems is taken into account [23]. The implementation of these two methods in combination showed a reduction in hardware costs of 7.86% to 37.78% and reduced the average time of image recognition by 41.17%.

## 3.2 Three approaches to OCR

Of all the peer-reviewed articles found using the joint search term combining 'OCR' and 'deep learning', a large majority dealt with using OCR in conjunction with deep learning in reading hand-written non-Latin script languages. In each of these cases the studies follow the typical pattern of:

1. Identify the problem language.
2. Propose a deep learning-aided OCR technique for improving recognition accuracy.
3. Conduct tests on an existing or generated database of words and characters.
4. Report on results.

These studies proved extremely useful in helping selection what type of standard deep learning and custom deep learning platform to apply when conducting our own tests.

### 3.2.1 Classic computer vision techniques: Tesseract 3.05

Tesseract is an open source OCR released under the Apache License, originally developed by Hewlett-Packard as proprietary software between 1985 and 1995. It was released as open source in 2005 and development has been supported by Google since 2006 [24]. Before machine learning and deep learning became ubiquitous in OCR around the 2010s, Tesseract was considered one of the most accurate open-source OCR engines available at that time [25].

Text image processing in Tesseract 3.05 follows a traditional step-by-step pipeline, as per Figure 8:

1. A connected component analysis in which outlines of the components are stored. Outlines are gathered together by nesting into Blobs.
2. Blobs are organized into text lines, and the lines and regions are analysed for fixed-pitch or proportional text.
3. Text lines are broken into words differently according to the kind of character spacing:
    a. Fixed-pitch text is chopped immediately by character cells.
    b. Proportional text is broken into words using definite spaces and fuzzy spaces.
    c. Recognition proceeds as a two-pass process:
    d. An attempt is made to recognise each word in turn. Each word that is satisfactory is passed to an adaptive classifier as training data. The adaptive classifier can then, more accurately, recognize text lower down the page.

e. A second pass is run in which words that were not sufficiently recognised are recognised again to compensate for the adaptive classifier still adapting on its first pass.

2. The final phase resolves fuzzy spaces and checks alternative hypotheses for the x-height to locate small-cap text. [8] [26]

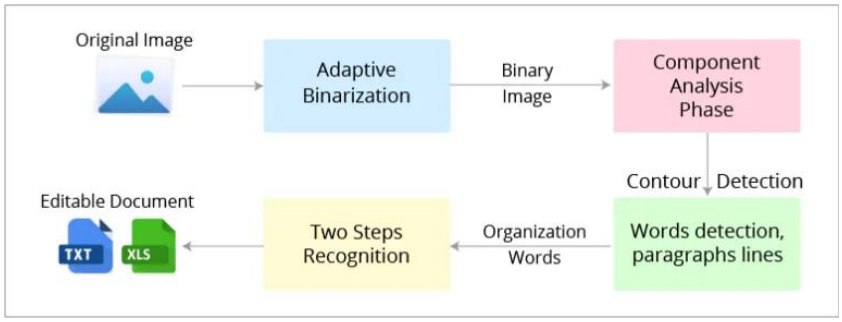


*Figure 8- An illustration of the Tesseract OCR architecture.*

### 3.2.2 Standard deep learning: Tesseract 4.1.1

As of version 4 (released October 2018) Tesseract incorporated Long Short-Term Memory (LSTM) within a Recurrent Neural Network (RNN) architecture (see Appendix 1: Neural network terminologies), using a text line recognizer in its new neural network subsystem. In modern OCR, it is ubiquitous to use a CNN to recognise an image that contains a single character, but in Tesseract 4 text that has arbitrary length and a sequence of characters is solved using RNNs and LSTM. The Tesseract input image in LSTM is processed in bound boxes line by line that inserts into the LSTM model and gives the output [27].

According to Tesseract's own documentation, the Tesseract 4 neural network subsystem is heavily compute-intensive, using the order of ten times the CPU resources of the base Tesseract unless adequate mitigation in the form parallel processing is not undertaken [15].

### 3.2.3 Specialised deep learning: OpenCV's EAST and Tesseract 4.1.1

OpenCV's EAST text detector is a deep learning model based on a novel architecture and training pattern. Its sole function is text detection in an image, not actual recognition or reading of the text. EAST is used to detect text in an image and bind text in horizontal and rotated bounding boxes which are divided into individual images (see Figure 9), which are then fed into a text recognition method [28], in this case Tesseract 4.1.1.

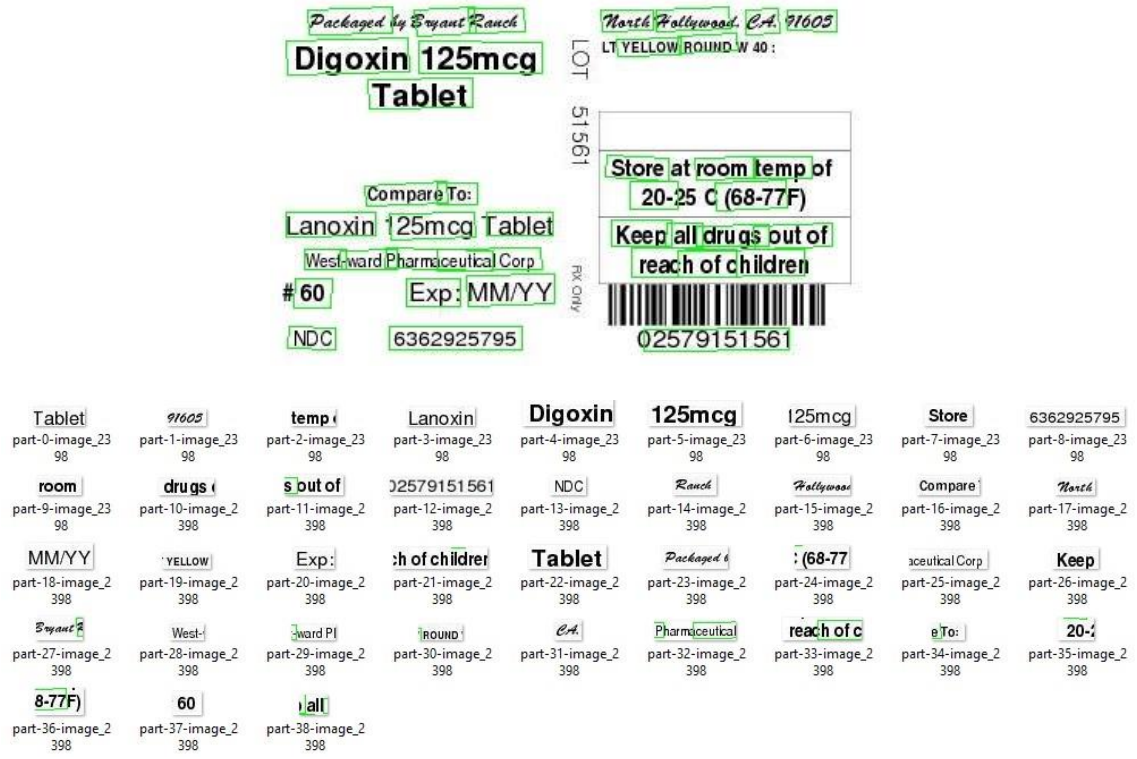


*Figure 9- An example of a prescription medicine label with bounding applied by OpenCV's EAST (above) and the resulting separation of the image into bound boxes for text recognition (below).*

EAST uses a single pipeline that directly predicts words or text lines of arbitrary orientations and quadrilateral shapes in full images, eliminating unnecessary intermediate steps, such as candidate aggregation and word partitioning, with a single neural network. The text detection pipeline has two stages, as per Figure 10:

3. A Fully-Convolutional Network (FCN) (see Appendix 1: Neural network terminologies) to directly produce word or text-line level prediction, which could be rotated rectangles or quadrangles.
4. A Non-Maximum Suppression (NMS) merging state to yield the final output.

The neural network model is trained to directly predict the existence of text instances and their geometries from full images. The model is an FCN adapted for text detection that outputs dense per-pixel predictions of words or text lines: an image is fed into the FCN and multiple channels of pixel-level text score map and geometry are generated. One of the predicted channels is a score map whose pixel values are in the range of [0; 1]. The remaining channels represent geometries that encloses the word from the view of each pixel. The score stands for the confidence of the geometry shape predicted at the same location. Thresholding is then applied to each predicted region, where the geometries whose scores are over the predefined threshold is considered valid and saved for later

NMS. The output after NMS, a post-processing algorithm responsible for merging all detections that belong to the same object [29], is considered the final output of the pipeline [28].

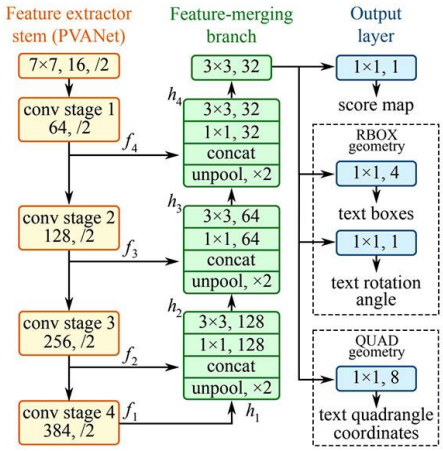


*Figure 10- The structure of the EAST text detection FCN.*

EAST is considered to be wholly robust, capable of localizing text even when it is blurred, reflective, or partially obscured. Regarding performance, since the EAST's deep learning model is end-to-end, it is possible to avoid computationally expensive sub-algorithms that other text detectors typically apply, including candidate aggregation and word partitioning.

# 4 Results & analysis

Table 1 is a general overview of the results from scanning the dataset of 100 images using each of the three OCR methods on the Android platform. A more in-depth review is of these is presented in this chapter. The results are broken down into categories of accuracy, time and resource use.

*Table 1- Overview of the results from testing 100 images using the three methods of OCR on the Android platform*

| | **CCVT - Tesseract 3.05** | **StanDL - Tesseract 4.1.1** | **SpecDL - EAST + Tesseract 4.1.1** |
|---|---|---|---|
| **Accuracy** | 50% | 76% | 74% |
| **Total time for 100 images (m, s)** | 23m 44s | 28m 16s | 60m 29s |
| **Average time per label (seconds)** | 14.24 | 16.97 | 36.29 |
| **Average CPU usage** | 12% | 12% | 20% |

| | | | |
|---|---|---|---|
| **Native memory heap avg. (MB)** | 150 | 93 | 515 |
| **Battery usage** | 1.82% | 5.23% | 8.25% |

## 4.1 Accuracy

Table 2 provides a more in-depth overview of the accuracy results from the test of the three methods. For a complete breakdown of the accuracy of all three methods, see Appendices 3: Test results.

*Table 2- A detailed presentation of the accuracy of each of the three method of OCR on the Android platform.*

| | **CCVT - Tesseract 3.05** | **StanDL - Tesseract 4.1.1** | **SpecDL - EAST + Tesseract 4.1.1** |
|---|---|---|---|
| **Complete match** | 50% | 76% | 74% |
| **Near miss** | 19% | 10% | 15% |

Table 2 shows that StanDL and SpecDL have the best accuracy of the three OCR methods, but the CCVT has the best potential for improvement, with 19% of scan results being only one character away from being a complete match. This does not count for much, however, as Tesseract 3.05 is the only method not to use any form of deep learning and therefore does not have the capability to be further trained without developer input and further iterations. Tesseract 4.1.1 and EAST, on the other hand, with their deep learning mechanisms, could be further trained and optimised by the user rather than by the developer. With more training, the StanDL and SpecDL methods could potentially achieve an accuracy rate approaching 86% and 89% respectively by correcting each ‘near miss’.

Accuracy could potentially be improved across all methods with further pre-processing of images before they are fed into the OCR methods, and by further testing with different combinations of segmentation and OCR engine parameters.

## 4.2 Time

Table 3 provides a more in-depth overview of the time taken for each image to be processed individually and as a whole dataset from the tests conducted. For a complete breakdown of the time taken for each image to be processed by each all three methods, see Appendices 3: Test results.

*Table 3- A detailed explanation of the time taken for scanning of each image and the dataset of 100 images.*

| | **CCVT - Tesseract 3.05** | **StanDL - Tesseract 4.1.1** | **SpecDL - EAST + Tesseract 4.1.1** |
|---|---|---|---|
| **Total time for 100 images (m, s)** | 23m 44s | 28m 16s | 60m 29s |
| **Time range (seconds)** | $0.02 \leq t \leq 86.11$ | $0.25 \leq t \leq 92.35$ | $8.53 \leq t \leq 167.11$ |
| **Average time per label (seconds)** | 14.24 | 16.97 | 36.29 |
| **# of images scanned in ≤ 1s** | 1% | 9% | - |
| **# of images scanned in ≤ 5s** | 21% | 19% | - |
| **# of images scanned in ≤ 10s** | 51% | 41% | 11% |

These results show that CCVTs and StanDL are certainly comparable in the fields of total time taken to process 100 images, the time range for individual images, and the average time taken to process an image. The standout, however, is StanDLs result in the category of images being processed in less than one second: 9% when compared to just 1% for CCVTs and zero for SpecDL. StandDL also shows promising results in the percentage of images processed in less than five and ten seconds respectively (19% and 41%). This demonstrates that for use in mobile scanner that aims to provide the user with near-instantaneous results, StanDL appears to be the best suited.

When viewing the time taken for each image to be processed when compared to the size of the image (see Figure 11, Figure 12 and Figure 13), there does not appear to be any substantive correlation between file size and time when using either CCVTs or SpecDL. However, there does appear to be some correlation when using only StanDL, especially with larger file sizes. This could be explained by Tesseract 4.1.1's use of LSTM, which is a form of RNN (see Appendix 1: Neural network terminologies): the larger the image, the more feedback in the RNN.

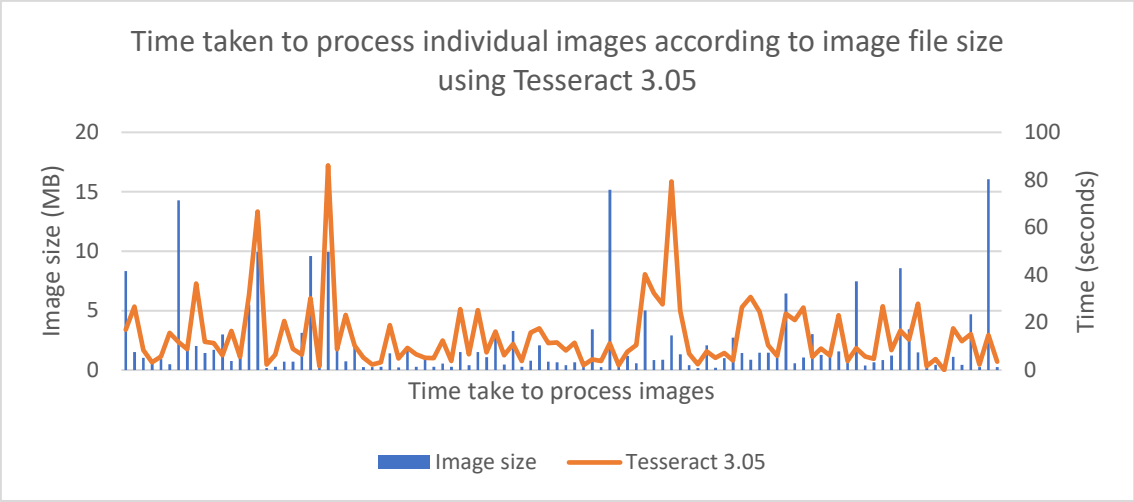


*Figure 11- A graphical depiction of the time taken to scan 100 individual images depending on the size of the image using Tesseract 3.05.*

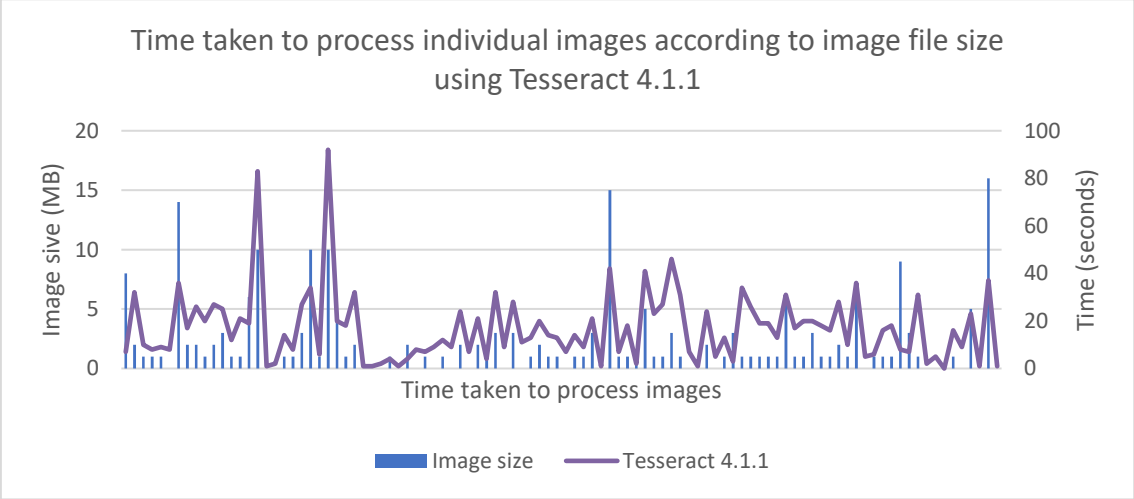


*Figure 12- A graphical depiction of the time taken to scan 100 individual images depending on the size of the image using Tesseract 4.1.1.*

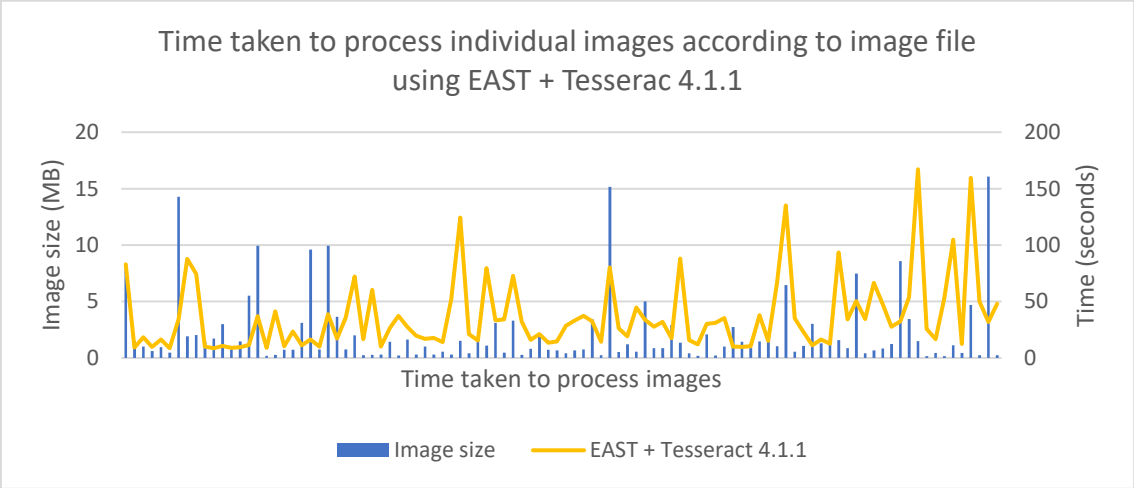


*Figure 13- A graphical depiction of the time taken to scan 100 individual images depending on the size of the image using EAST + Tesseract 4.1.1.*

When examining the time taken to process individual images according to the dimensional rather than file size, the does not appear to be any correlation between dimensional size of an image and the time taken for the image to be processed using this CCVT and StanDL, as per Figure 14 and Figure 15.

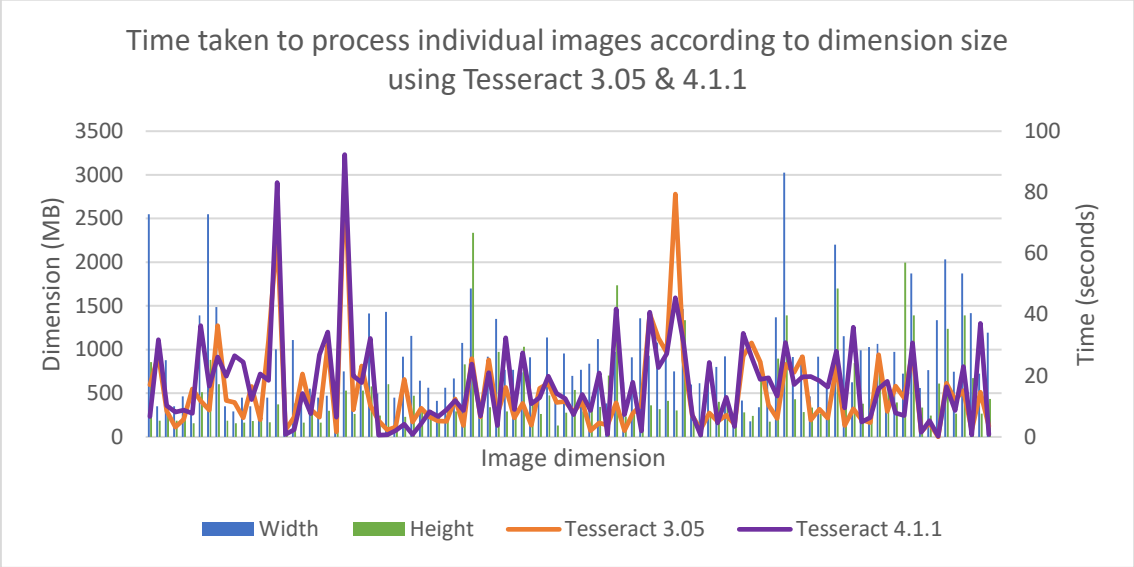


*Figure 14- A graphical depiction of the time taken to scan 100 individual images depending on the dimensions of the image using Tesseract 3.05& 4.1.1.*

Conversely, there appears to be a strong correlation between dimensional size of an image and the time taken for the image to be processed using SpecDL, as per Figure 15.

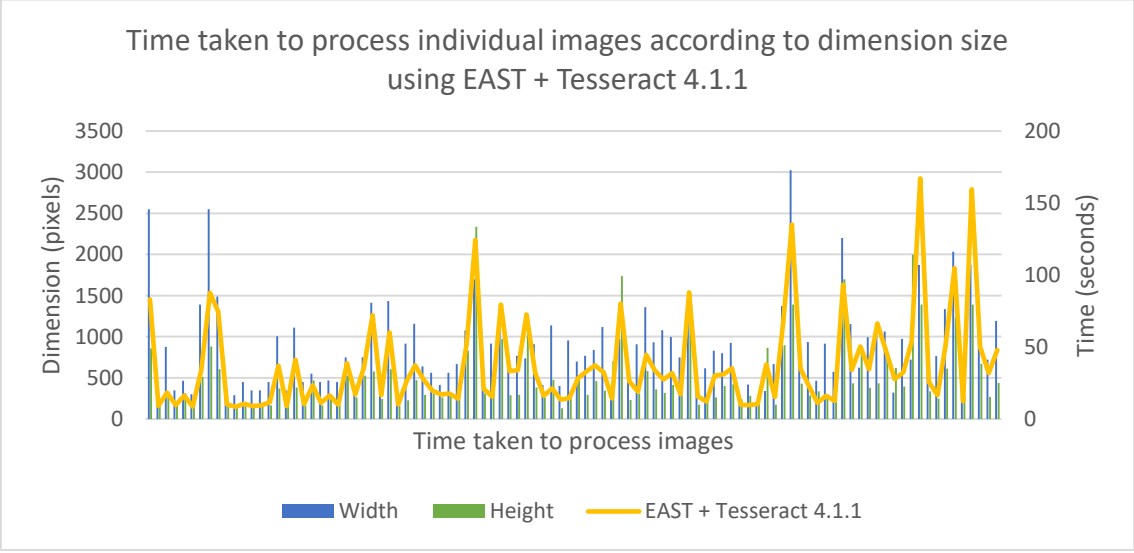


*Figure 15- A graphical depiction of the time taken to scan 100 individual images depending on the dimensions of the image using EAST + Tesseract 4.1.1.*

The correlation between dimensional size of an image and the time taken for the image to be processed using SpecDL can be explained with EAST's use of bound boxes as a

function of text detection (see 3.2.3): the larger the dimension of an image, the more likelihood there is for text to be detected. Demonstrating this further is evidence that the more bound boxes generated by EAST, the longer an image takes to generate an output, as shown by Figure 16. Overall, this would suggest that rather than the file size of the image dictating the scanning time using SpecDL, it would depend on the number of words in contained in the image.

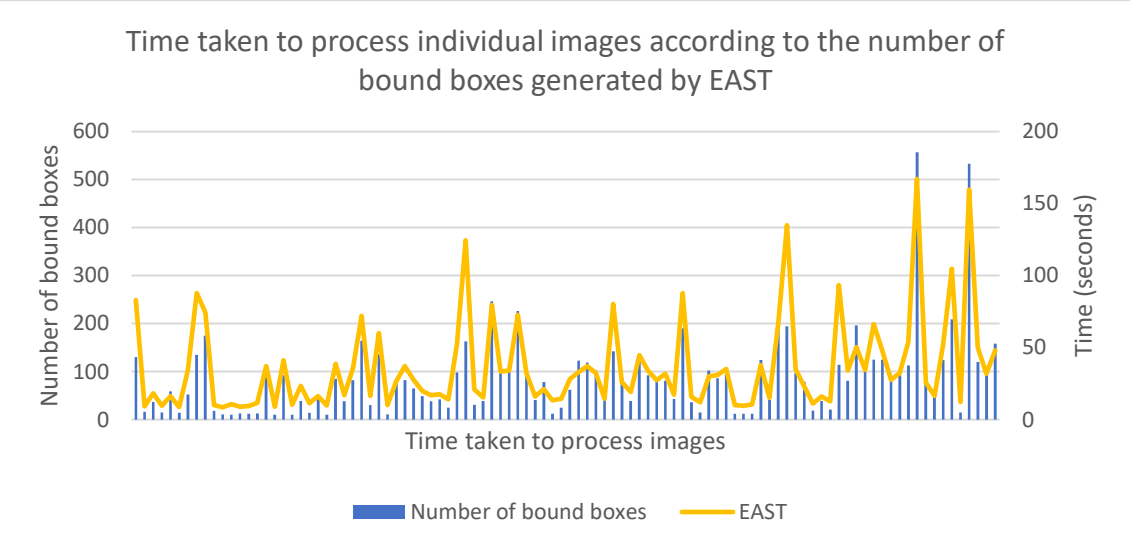


*Figure 16- A graphical depiction of the time taken to scan each individual images against the number of bound boxes generated in processing that image using EAST.*

## 4.3 Resource use

Table 4 provides a more in-depth overview of the performance results from the test of the three methods. For a complete breakdown of the performance of all three methods, see Appendices 3: Test results.

*Table 4- A detailed explanation of the performance aspect of processing 100 prescription label images using the three methods of OCR.*

| | **CCVT - Tesseract 3.05** | **StanDL - Tesseract 4.1.1** | **SpecDL - EAST + Tesseract 4.1.1** |
|---|---|---|---|
| **Average CPU usage** | 12% | 12% | 20% |
| **CPU usage range (% of capacity)** | $10 \leq c \leq 14$ | $10 \leq c \leq 14$ | $10 \leq c \leq 85$ |
| **Native heap memory avg (MB)** | 78 | 93 | 515 |

| | | | |
|---|---|---|---|
| **Native heap memory range (MB)** | $63 \leq m \leq 140$ | $80 \leq m \leq 152$ | $274 \leq m \leq 2050$ |
| **Battery usage** | 1.82% | 5.23% | 8.25% |

These results show that the performances of the methods for CCVTs and StanDL are comparable, while SpecDL is more resource hungry by several orders of magnitude. While slightly more resource intense than Tesserect 3.05, the multiple cores in the P30 appear to be able handle the extra processing that LSTM requires despite the developer's warnings [15]. Overall, this serves to suggests that CCTVs and StanDL are more suited to a wider range of mobile devices, while all but the higher end mobile devices are currently equipped to handle the resource costs of SpecDL.

As shown in Figure 17, Figure 18 and Figure 19, there appears only to be a slight correlation between the file size of an image and the amount of native heap memory used for all three methods of OCR. Though, it is apparent for all three methods of OCR is that the pattern of native heap memory is consistent over the 100 images, albeit at slightly different magnitudes for Tesseract 3.05 and 4.1.1, and at a substantially larger magnitude for EAST + 4.1.1.

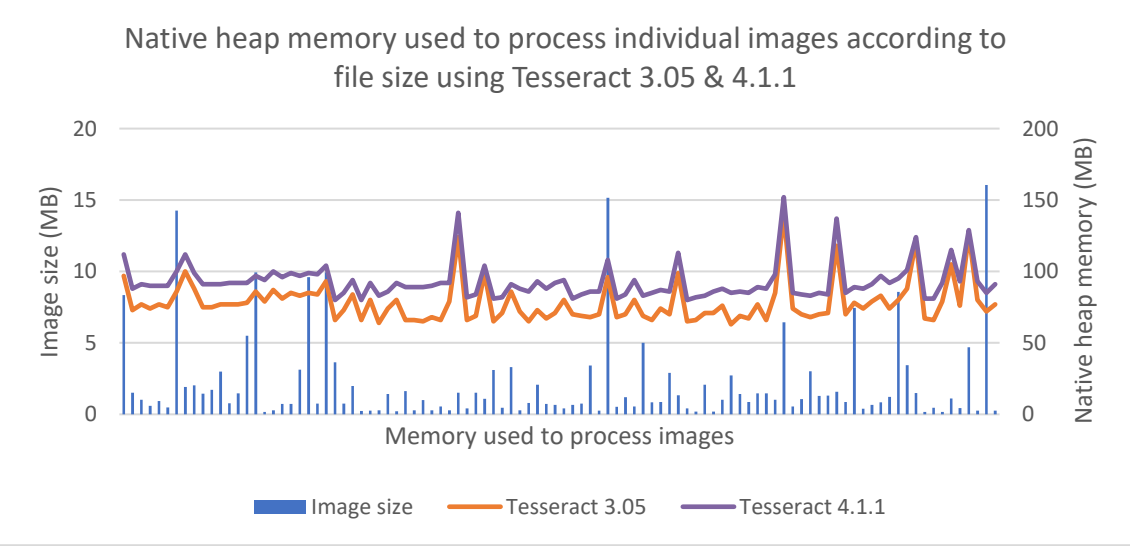


*Figure 17- A graphical representation of native heap memory used to process individual images according to file size using Tesseract 3.05 & 4.1.1*

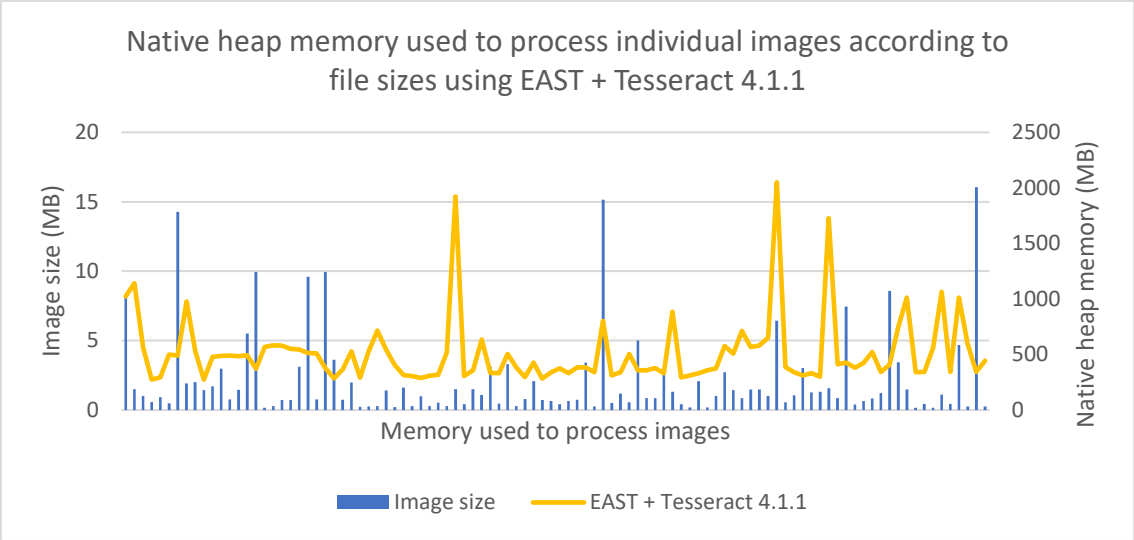


*Figure 18- A graphical representation of native heap memory used to process individual images according to file size using EAST + Tesseract 4.1.1*

There is a far more pronounced association between native heap memory used and dimension size in all three methods of OCR, as shown in Figure 19 and Figure 20. As with comparing file size and memory cost, it is apparent for all three methods of OCR is that the pattern of native heap memory use is consistent over the 100 images, albeit at slightly different magnitudes for Tesseract 3.05 and 4.1.1, and at a substantially larger magnitude for EAST + 4.1.1.

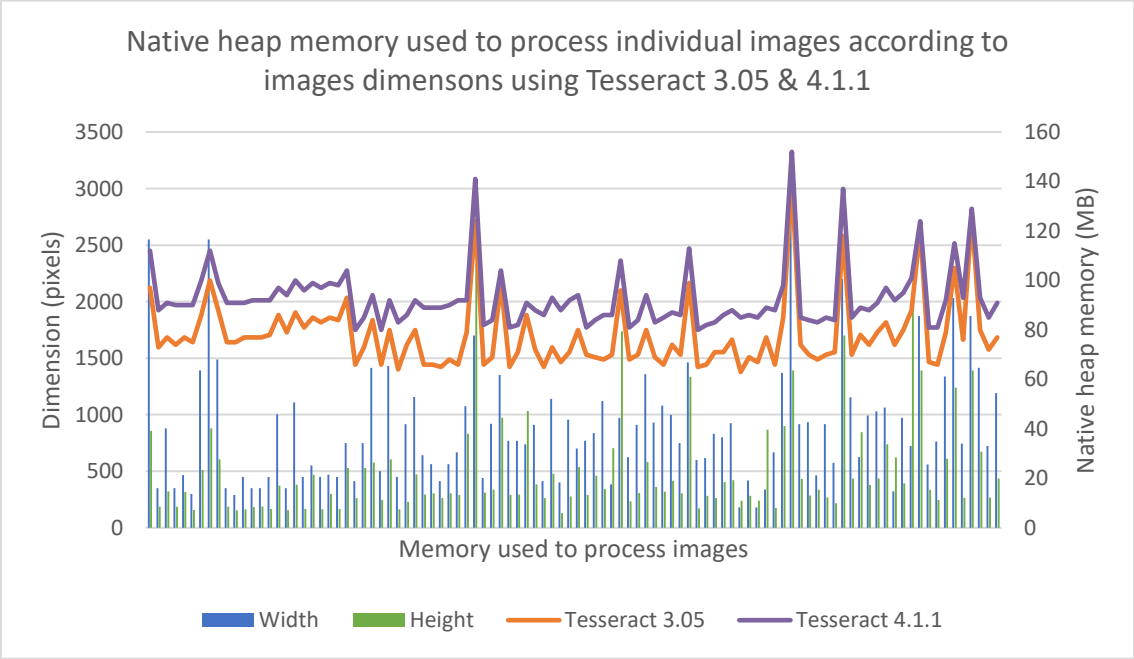


*Figure 19- A graphical representation of native heap memory used to process individual images according to images dimensions using Tesseract 3.05 & 4.1.1*

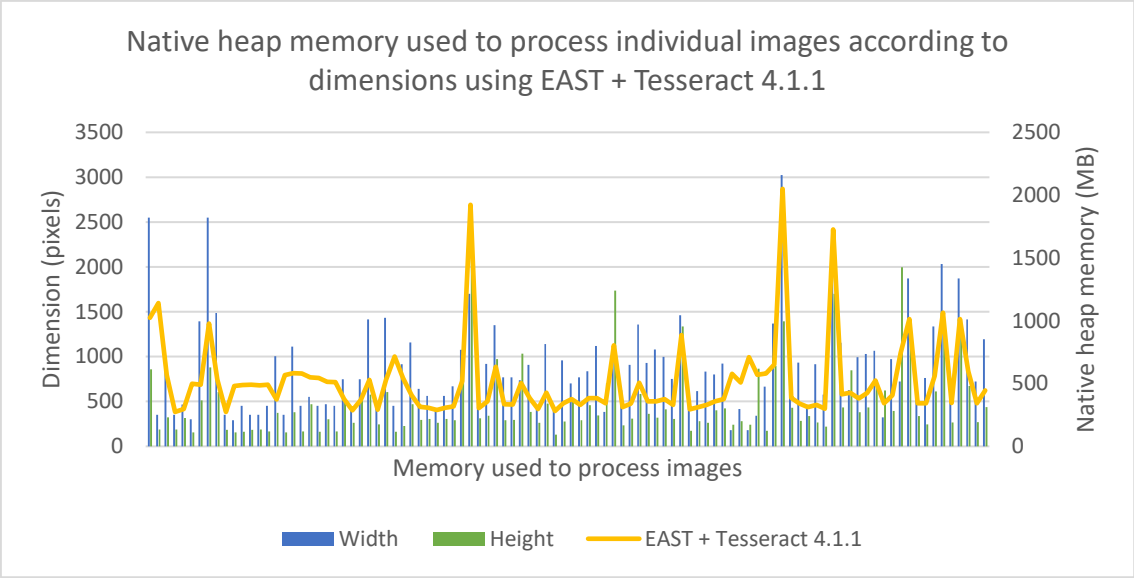


*Figure 20- A graphical representation of native memory used to process individual images according to dimensions using EAST + Tesseract 4.1.1.*

Unlike the time taken to process each images see (Figure 16), there does not appear to be any substantive association between the number of bound boxes generated by EAST in the SpecDL method, as shown in Figure 21.

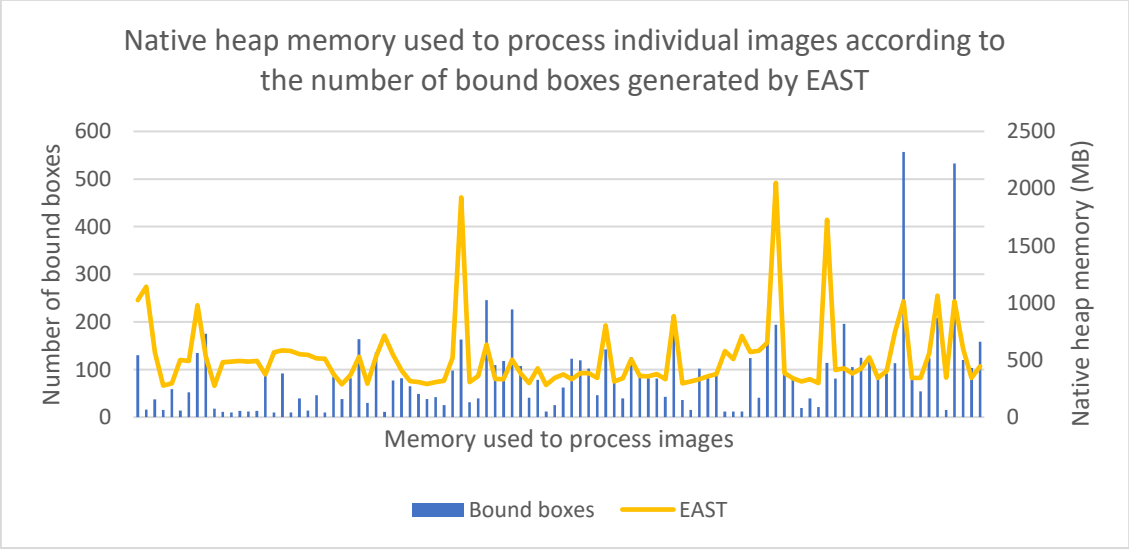


*Figure 21- A graphical representation of native heap memory used to process individual images according to the number of bound boxes generated by EAST.*

Regarding power consumption, it is perhaps not surprising that SpecDL consumes the most amount of power considering it takes approximately twice as long to scan the 100-image dataset and demands more resources by several orders of magnitude. However, while the CCTV and the StanDL methods are comparable for the range and average CPU and native memory heap usage, StanDL uses nearly three times the amount of power than

the CCTV method. Assuming these results are accurate, this could be explained in Tesseract's documentation [15] as mentioned in 3.2.2: the Tesseract 4 neural network subsystem is heavily compute-intensive, using the order of ten times the CPU resources of the base Tesseract unless adequate mitigation in the form parallel processing is not undertaken. The P30 does possess multiple cores (see Appendix 2: Case study details) and appears to have had no issue mitigating the LSTM network used by Tesseract 4.1.1 but this parallel processing may have not been detected by Android Profiler in measuring the CPU usage. This would also explain why Tesseract 3.05 and Tesseract 4.1.1 had practically identical CPU usage results.

# 5 Discussion

The results garnered in Chapter 4 show that, perhaps unsurprisingly, the two methods of OCR that utilise deep learning provide not only the greatest accuracy but also the potential for improvement. Regarding speed and performance, the StanDL method was the standout, providing by far the most near-instant results at a reasonable resource cost, while SpecDL was both slow and by far the most resource hungry, making it unfeasible for use in a scanner on a mobile device in a real-world application.

## 5.1 Label examples and analysis

When attempting to establish what determines how long an image takes to process, Chapter 4 establishes that there appears to be no strong correlation between image file dimension size and time for CCVTs and SpecDL, while the is a correlation using StanDL with is particularly pronounced is larger image sizes. There is a strong association between both file size and dimensional magnitude and time when using SpecDL due to the nature of the text detection algorithm.

Table 5 provides a selection label images, their file size and dimension, and processing times using the three method of OCR. The results of all processed images contained therein were either accurate or deemed a 'near miss' under manual review.

*Table 5- An example of selected label images, their file size and dimension and processing times using the three method of OCR.*

| Image number | File size (MB) | Width (px) | Height (px) | Tesseract 3.05 (seconds) | Tesseract 4.1.1 (seconds) | EAST + Tesseract 4.1.1 (seconds) |
|---|---|---|---|---|---|---|
| image_02300.jpg | 9.94 | 749 | 529 | 86.11 | 92.35 | 38.78 |
| image_02398.jpg | 3.63 | 413 | 263 | 8.85 | 19.99 | 17.04 |
| image_06616.jpg | 0.18 | 615 | 282 | 2.52 | 0.61 | 12.02 |

Figure 22 shows image_2300.jpg at a substantially reduced size, it being one of the largest file sizes in the dataset at 9.94MB. All three methods OCR took over 30 seconds to process this one image, with the CCVT and StanDL methods taking over well over 60 seconds, as per Table 5. Although the roughly 50% of the image is negative space, the text contained in the label presents both vertical and horizontal text in dense sections, making it especially difficult for OCR methods use text segmentation, as does Tesseract 3.05 and 4.1.1, rather than text detection as does EAST. Additionally, numbers of dense areas of smaller text have proven to be especially challenging to all methods of OCR, resulting in significantly higher processing times in these cases. In this case, EAST generated no less than 85 bound boxes for processing.


*Figure 22- image_2300.jpg*

Figure 23 shows image_02389.jpg at its actual size although it is still in the top 20% of the file sizes in the dataset. All methods of OCR used processed the image in under 20 seconds. As the majority of the text contained in the image is horizontal and the text is relative sparse when compared to Figure 22, processing the image takes less, but not insubstantial, time for all three methods. The correlation between image size and processing serves to explain why the StanDL method took the longest time of the three methods to process this image in particular.


*Figure 23- image_02398.jpg*

Figure 24 is the smallest file in Table 5 and one of the smallest in the dataset, although it is larger dimensionally than Figure 23. Despite this, two of the three methods of OCR processed this image in less than five seconds. As with Figure 22, a large portion of this image is negative space and presents text in dense blocks. Again, it is evident that the dense area of small text proved to be an issue for EAST.

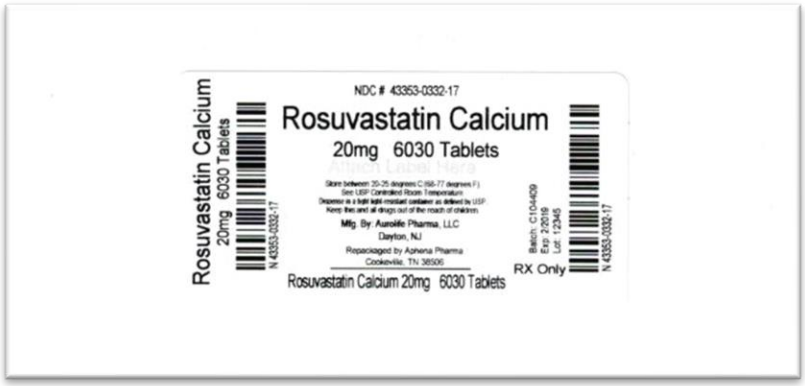


*Figure 24- image_06616.jpg*

## 5.2 Problem labels

Out of the dataset of 100 label images, 6% of these could not be processed accurately by any of the three methods of OCR (see Appendices 3: Test results). When examining these labels, there are two recurring factors that appear in all 6%:

- Low resolution, blocky font, as shown in Figure 25 and Figure 26, and
- Text on a coloured background, as shown in Figure 27.


*Figure 25- image_01147.jpg*


*Figure 26- image_01156.jpg*


*Figure 27- image_07343.jpg*

Should any prescription medication scanner that uses one of the three methods of OCR utilised here become popular, medicine manufacturers may want to take this into consideration in designing labels.

# 6 Conclusion

This study has explored the subjects of OCR and deep learning with the ultimate goal of examining which particular method of OCR would be best suited for a mobile platform in the specific context of a prescription medication label scanner. A review of peer-reviewed literature found that there was a definite scarcity of research on the subject of OCR that incorporates deep learning on specifically on mobile platforms, and practically none were performance was the main consideration. A case study conducted using three methods of OCR, two of which utilised deep learning to varying degrees, proved that one of these methods was categorically more suited than other for the particular context in which the study was conducted.

## 6.1 Research answers

The preceding research served to answer the question:

*In the context of a mobile application using image recognition to scan prescription medication labels, which approach to OCR provides the best performance?*

The method of OCR that provides the best combination of accuracy, speed and resource using has proven to be Standard Deep Learning, or Tesseract 4.1.1 in this particular case. StanDL proved to be the most accurate method, had the highest number of images processed in less than one second and demonstrated that it had very reasonable resource costs – comparable to methods that did not utilise deep learning.

## 6.2 Ethics, sustainable development and societal aspects

As stated in 1.1, patients misidentifying medication is a real concern in, and the problem is only growing in magnitude in this country as the population tends toward a higher median age. It is important to be equipped with the right tools to achieve the highest accuracy in the identification of prescription medication for the health and well-being of the population.

If an application for scanning prescription medication was to be further developed, consideration of how to securely store patient data will to be addressed. As people store sensitive information on their mobile phones, any application using OCR implementation needs to be secure because as it is accessing the phone's camera and external storage.

Additionally, an official government/public health institution would be required to acquire accurate data of each medication and any potentially conflicting medications.

Battery consumption on a single phone may be insignificant but when considering an application with multiple millions of users, battery consumption and temperature increase during app usage becomes an issue. Every step towards optimisation matters.

## 6.3 Future work

Future work on this study could resolve some of the following areas of improvements:

- Expanding testing hardware and software to include other models of mobile phone and other mobile software platforms, especially iOS
- Expanding the dataset to include images of prescription medicine labels in real world environments rather than digital scans
- Training neural networks for the specific purpose rather than using a pre-trained network
- Testing various software models of each OCR method
- Experimenting with parallelisation of OCR methods over multiple cores to achieve faster performance.

# 7 References


1. Socialstyrelsen. *Statistik om läkemedel*. [Online]. [cited 2020 Jan 22]. Available from: https://www.socialstyrelsen.se/statistik-och-data/statistik/statistikamnen/lakemedel/.

2. Institute of Medicine. *Preventing Medication Errors*. Washignton, DC: The National Academies Press; 2007. Report No.: ISBN 978-0-309-10147-9.

3. Institute of Medicine. *To Err Is Human: Building a Safer Health System*. Washington, DC: The National Academies Press; 2000. Report No.: ISBN 978-0-309-06837-6.

4. The World Bank. *Population ages 65 and above (% of total population) - Sweden, United States, World*. [Online]. [cited 2020 January 22]. Available from: https://data.worldbank.org/indicator/SP.POP.65UP.TO.ZS?end=2018&locations=SE-US-1W&start=1960&view=chart.

5. Zeng X, Cao K, Zhang M. MobileDeepPill. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services - MobiSys 17*; 2017.

6. Delgado NL, Usuyama N, Hall AK, Hazen RJ, Ma M, Sahu S, et al. Fast and accurate medication identification. *npj Digital Medicine*. 2019; 2(1).

7. P A. Optical Character Recognition. *International Journal of Scientific Research and Management*. 2016.

8. Smith R. An Overview of the Tesseract OCR Engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*; 2007. p. 629-633.

9. Dhavale SV. In Advanced Image-Based Spam Detection and Filtering Techniques. Hershey, PA: Information Science Reference; 2017. p. 91.

10. d'Albe E. On a Type-Reading Optophone. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*. 1914 July; 90(619): p. 373–375.

11. *Vetenskapen och livet*. 1922: p. 105.

12. Dechter R. Learning While Searching in Constraint-Satisfaction-Problems. In *Proceedings of the 5th National Conference on Artificial Intelligence*; 1986; Philadelphia, PA. p. 178-185.

13. Schmidhuber J. Deep Learning in Neural Networks: An Overview. *Neural Networks*. 2015 January; 61: p. 85-117.

14. GlobalStats. *statcounter*. [Online]. [cited 2020 June]. Available from: https://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide.

15. tesseract-ocr. *Overview of the new neural network system in Tesseract 4.00*. [Online].; 2019 [cited 2020 May 2]. Available from: https://tesseract-ocr.github.io/tessdoc/NeuralNetsInTesseract4.00.

16. Mancas C, Gosselin T, Gosselin B. *Vision Systems: Segmentation and Pattern Recognition* Obinata G, Dutta A, editors.: IntechOpen; 2007.

17. US National Library of Medicine. *SPL RESOURCES: Download All Drug Labels*. [Online].; 2020 [cited 2020 May 1]. Available from: https://dailymed.nlm.nih.gov/dailymed/spl-resources-all-drug-labels.cfm.

18. Benaddy M, El Meslouhi O, Es-saady Y, Kardouchi M. Handwritten Tifinagh Characters Recognition Using Deep Convolutional Neural Networks. *Sensing and Imaging*. 2019;(20).

19. Roy S, Das N, Kundu M, Nasipuri M. Handwritten isolated Bangla compound character recognition: A new benchmark using a novel deep learning approach. *Pattern Recognition Letters*. 2017 April; 90.

20. Jangid M SS. Handwritten Devanagari Character Recognition Using Layer-Wise Training of Deep Convolutional Neural Networks and Adaptive Gradient Methods. *Journal of Imaging*. 2018; 4(41).

21. Yin Y, Zhang W, Hong S, Yang J, Xiong J, Gui G. Deep Learning-Aided OCR Techniques for Chinese Uppercase Characters in the Application of Internet of Things. *IEEE Access*. 2019; 7: p. 47043-47049.

22. Valuevaa MV, Nagornovb NN, Lyakhova PA, Valueva MV, Chervyakova NI. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation*. 2020 November; 177: p. 232–243.

23. Parhami B. *Computer Arithmetic: Algorithms and Hardware Design*. 2nd ed. New York: Oxford University Press; 2010.

24. Vincent L. *Google code*. [Online].; 2006 [cited 2020 May 15]. Available from: https://googlecode.blogspot.com/2006/08/announcing-tesseract-ocr.html.

25. Willis N. *Linux.com*. [Online].; 2006 [cited 2020 May 15]. Available from: https://www.linux.com/news/googles-tesseract-ocr-engine-quantum-leap-forward/.

26. Boiangiu CA, Ioanitescu R, Dragomir RC. Voting-Based OCR System. *Journal of Information Systems & Operations Management*. 2016; 10.

27. tesseract-ocr. *Tesseract OCR*. [Online].; 2020 [cited 2020 May 2]. Available from: https://github.com/tesseract-ocr/tesseract.

28. Zhou X, Yao C, Wen H, Wang Y, Zhou S, He W, et al. EAST: An Efficient and Accurate Scene Text Detector. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*; 2017; Honolulu. p. 2642-2651.

29. Hosang J, Benenson R, Schiele B. Learning Non-Maximum Suppression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*; 2017. p. 4507-4515.

30. Kumar Singh M, Baluja GS, Prasad Sahu D. Understanding the Convolutional Neural Network & its Research Aspects in Deep Learning. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*. 2017 June.

31. Zhang Y, Itoh K, Tanida J, Ichioka Y. Parallel distributed processing model with local space-invariant interconnections and its optical architecture. *Applied Optics*. 1990: p. 4790-4797.

32. Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*; 2015. p. 3431–3440.

33. Kulkarni TD, Whitney WF, Kohli P, Tenenbaum JB. Deep Convolutional Inverse Graphics Network. In *Advances in Neural Information Processing Systems 28*; 2015.

34. Di Pietro R, Hager GD. *Handbook of Medical Image Computing and Computer Assisted Intervention*. In SK Z, D R, G F, editors. Handbook of Medical Image Computing and Computer Assisted Intervention. Baltimore: Academic Presss; 2019. p. 503-519.

35. Hochreiter S, Schmidhuber J. Long Short-Term Memory. *Neural Computation*. 1997 November; 9(8).

# 8 Appendices

## 8.1 Appendix 1: Neural network terminologies

### 8.1.1 Feedforward neural networks

Feedforward neural networks are named after the way they channel information through a series of mathematical operations performed at the nodes of the network. Input examples are fed into the network and transformed into an output. Under supervised learning the output would be a label, or a name applied to the input and raw data is mapped to categories, recognising patterns that may signal. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. As information only moves in one direction, there are no cycles or loops in the network (see Figure 2) [13].

### 8.1.2 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) are a kind of feedforward artificial neural network that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required is much lower as compared to other classification algorithms; previous methods' filters are human-engineered, whereas with enough training, CNNs have the ability to learn these filters/characteristics. Convolutional networks are neural networks that use convolution in place of general matrix multiplication in at least one of their layers [30].

While there are many applications for traditional neural network architecture, there are several limitations, especially when it comes to image processing. Traditional neural network architecture, or Multi-Layer Perceptron (MLP) use one perceptron, or an algorithm for supervised learning of binary classifiers for each input. The amount of weights in an MLP rapidly becomes unmanageable for large images. Additionally, MLPs are usually fully connected, with each neuron in one layer is connected to all neurons in the next layer, which often causes overfitting of data. Another common problem is that MLPs react differently to an input and its shifted version in that they are not translation invariant [31].

In contrast, CNNs take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme, using relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

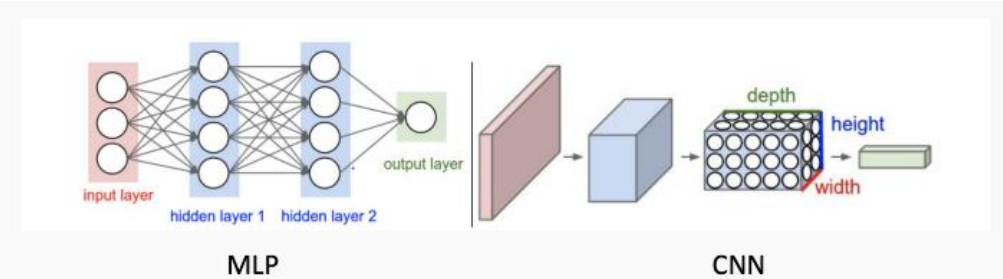


*Figure 28- Comparison of architecture for MLP and CNN.*

The structural design of CNN is analogous to that of the connectivity pattern of neurons in the human brain and was inspired by the arrangement of the visual cortex.

A CNN usually consists of the following components:

- Input layer: a single raw image is given as an input. For a Red-Green-Blue (RGB) image its dimension will be AxBx3, where 3 represents the three colours.
- A convolution layer: a convolution layer is a matrix of dimension smaller than the input matrix. It performs a convolution operation with a small part of the input matrix having same dimension. The sum of the products of the corresponding elements is the output of this layer.
- Rectified Linear Unit (ReLU): ReLU is mathematically expressed as max(0,x), where any number below 0 is converted to 0 while any positive number is allowed to pass as it is.
- Maxpool: this passes the maximum value from amongst a small collection of elements of the incoming matrix to the output. Usually it is a square matrix.
- Fully connected layer: the final output layer is a normal fully-connected neural network layer, which gives the output. [30].

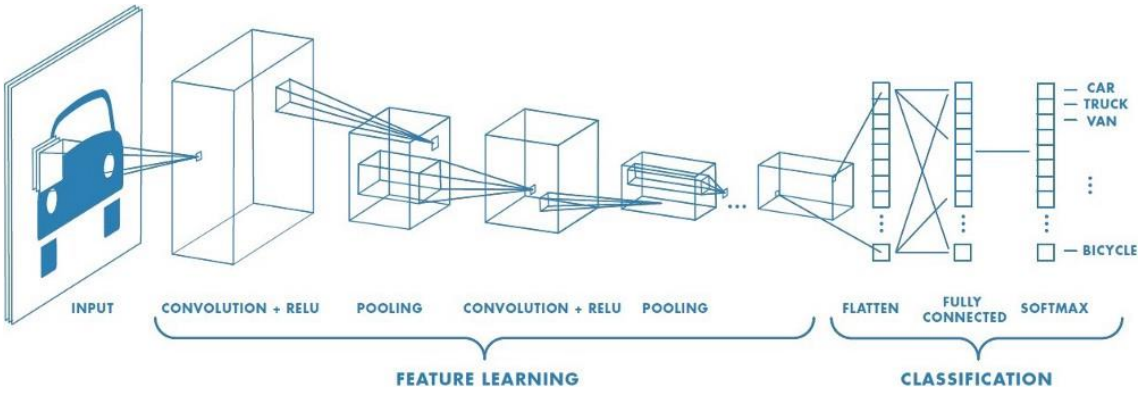


*Figure 29- An illustration of a typical CNN.*

### 8.1.3 Fully-Convolutional Network (FCN)

A Fully-Convolutional Network (FCN) uses a CNN to transform image pixels to pixel categories. Unlike CNNs, all the learnable layers in an FCN are convolutional so it does not have any fully-connected layer. An FCN transforms the height and width of the intermediate layer feature map back to the size of input image through the transposed convolution layer, so that the predictions have a one-to-one correspondence with input image in spatial dimension (height and width). Given a position on the spatial dimension, the output of the channel dimension will be a category prediction of the pixel corresponding to the location [32].

The main advantages of an FCN over a CNN include:

- Input image size: In a CNN, the fully-connected layer expects inputs of a certain size. Without this connected layer in the network, images of virtually any size can be processed.
- Spatial information: As all output neurons are connected to all input neurons in the fully-connected layer in a CNN, this can cause loss of spatial information, making segmentation impossible. [33]

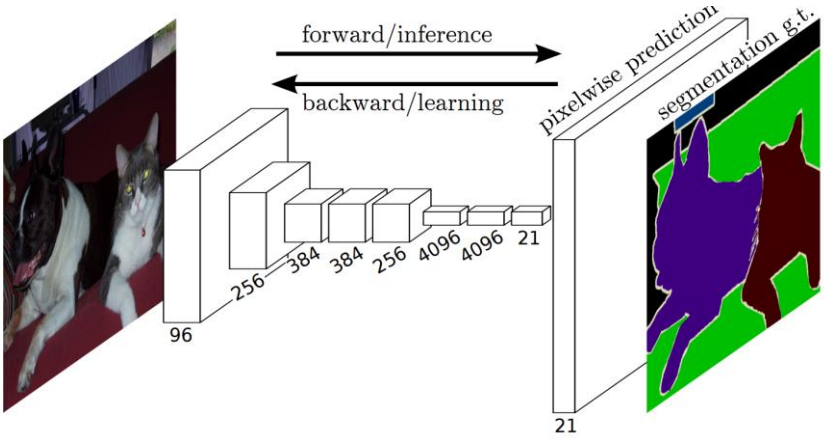


*Figure 30- An illustration of a traditional FCN.*

### 8.1.4 Recurrent Neural Network (RNN)

Recurrent neural networks (RNN) are another type artificial neural network derived from feedforward neural networks, but unlike feedforward neural networks, can process input not just the current input example received, but also what has been perceived previously. The decision a RNN at time step *t-1* affects the decision it will reach one moment later at time step *t*, meaning RNNs have two sources of input: the present and the recent past, which combine to determine how they respond to new data. Connections between nodes form a directed graph along a temporal sequence, allowing it to exhibit temporal dynamic behaviour.

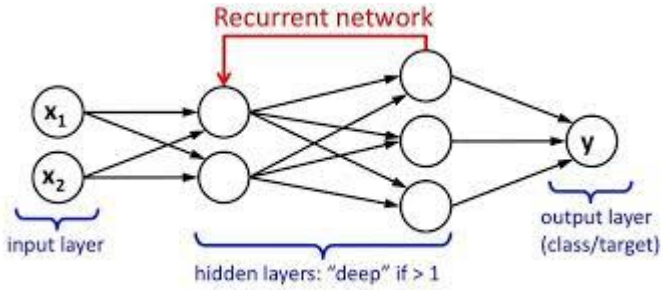


*Figure 31- An illustration a simple RNN.*

RNNs can use their internal state, or memory, to process variable length sequences of inputs. That sequential information is preserved in the recurrent network's hidden state, which manages to span many time steps as it cascades forward to affect the processing of each new example. It is finding correlations between events separated by many moments because an event downstream in time depends upon, and is a function of, one or more events that came before. In this way, RNNs share weights over time [34].

### 8.1.5 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are not so much a different variant of RNN architecture, but rather introduces changes to how outputs and hidden state using inputs are computed. Although traditional RNNs can keep track of arbitrary long-term dependencies in the input sequences, problems can arise when training a RNN using back-propagation, where the gradients which are back-propagated can tend towards zero, or 'vanish', or can tend to infinity, or 'explode'. This is due to the computations involved in the process, which use finite-precision numbers. RNNs using LSTM units partially solve the vanishing gradient problem as they are capable of learning long-term dependencies.

There are several architectures of LSTM units, and common architecture is composed of a cell (the memory part of the LSTM unit) and three 'regulators' or 'gates', of the flow of information inside the LSTM unit: an input gate, an output gate and a forget gate.

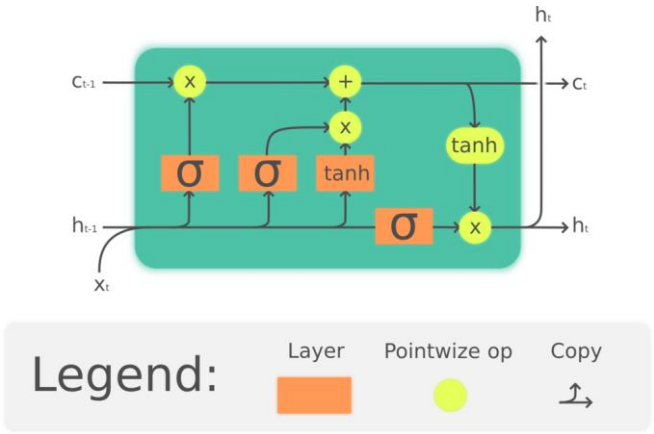


*Figure 32- A schematic of a LSTM unit as used in the hidden layers of an RNN.*

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series [35].

## 8.2 Appendix 2: Case study details

### 8.2.1 Mobile phone hardware specifications:

*Table 6- Hardware specifications for the Huawei P30 mobile phone*

| | |
|---|---|
| **CPU** | Huawei Kirin 980 (8 core):<br>2x Cortex-A76 @ 2.6GHz |

| | |
|---|---|
| | 2x Cortex-A76 @ 1.92GHz<br>4x Cortex-A55 @ 1.8GHz<br>(4MB shared L3 cache) |
| **RAM** | 6.0 GB, LPPD4X @ 2133 MHz |
| **Screen resolution** | 2330x1080 |
| **Battey** | 3650mAh |

### 8.2.2 Code repositories

100 image dataset: https://github.com/jbisiach/ImageDataset100.git

Java image filter: https://github.com/matzab/DataFilter_

Android OCR test application: https://github.com/matzab/OCRTest

### 8.2.3 Pre-testing results

*Table 7- Tests conducted using Tesserect 3.05 with different sized image sets.*

| | **10 images** | **100 images** |
|---|---|---|
| **Total time (seconds)** | 99.26 | 1424.41 |
| **Accuracy** | 60% | 66% |
| **Average CPU usage** | 12% | 12% |
| **Memory usage range (MB)** | $120 \leq m \leq 140$ | $120 \leq m \leq 250$ |
| **Power consumption** | 0.9% | 12.6% |

### 8.2.4 Tessreact & EAST libraries

Tesseract 3.05: https://github.com/adaptech-cz/Tesseract4Android

Tesseract 4.1.1: https://sourceforge.net/projects/opencvlibrary/files/4.1.1/

EAST: https://github.com/opencv

### 8.2.5 Tesseract options

Tesseract 3.05 engine options:

1. Tesseract engine only

2. Cube engine only
3. Tesseract and Cube combined
4. Default engine only

Tesseract 4.1.1 engine options:

1. Tesseract engine only
2. LSTM only
3. Tesseract and LSTM combined
4. Default OCR engine mode.

Tesseract orientation and segmentation options:

1. Orientation and script detection only
2. Automatic page segmentation with OSD
3. Fully automatic page segmentation, but no OSD, or OCR
4. Fully automatic page segmentation, but no OSD
5. Assume a single column of text of variable sizes
6. Assume a single uniform block of vertically aligned text
7. Assume a single uniform block of text (default)
8. Treat the image as a single text line
9. Treat the image as a single word
10. Treat the image as a single word in a circle
11. Treat the image as a single character
12. Find as much text as possible in no particular order
13. Sparse text with OSD
14. Treat the image as a single text line, bypassing ‘hacks’ that are Tesseract-specific.

Tesseract dictionary options:

1. Whitelist of characters to recognize
2. Blacklist of characters to not recognize
3. Save blob choices allowing acquisition of alternative results
4. String value used to assign a boolean variable to true
5. String value used to assign a boolean variable to false

## 8.3 Appendices 3: Test results

### 8.3.1 Test results

Raw test results repository: https://github.com/jbisiach/Test-Results.git

### 8.3.2 Accuracy

*Table 8- Comprehensive test results denoting which images were correctly or almost correctly identified using each method of OCR*

● denotes exact match

○ denotes near miss

| File Name | CCVT - Tesseract 3.05 | StanDL - Tesseract 4.1.1 | SpecDL - EAST + Tesseract 4.1.1 |
|---|---|---|---|
| image_00000.jpg | | ● | ● |
| image_00079.jpg | | ● | ○ |
| image_00258.jpg | ● | ● | ● |
| image_00377.jpg | | ○ | ○ |
| image_00414.jpg | | ● | ● |
| image_00478.jpg | | ○ | |
| image_00515.jpg | ● | ● | ● |
| image_00689.jpg | | ● | ● |
| image_00698.jpg | ● | ● | ○ |
| image_00771.jpg | | ○ | |
| image_01147.jpg | | | |
| image_01156.jpg | | | |
| image_01294.jpg | | ○ | ○ |
| image_01384.jpg | | ○ | ○ |
| image_01478.jpg | | ● | ○ |
| image_01569.jpg | ○ | ● | ● |
| image_01660.jpg | | ● | ○ |
| image_01751.jpg | ● | ● | ● |
| image_01842.jpg | | | |
| image_01936.jpg | ● | ● | ○ |
| image_02027.jpg | | | |
| image_02118.jpg | | ● | ● |
| image_02209.jpg | | ● | ● |
| image_02300.jpg | ● | ● | ● |
| image_02398.jpg | ○ | ● | ● |

| | | | |
|---|---|---|---|
| **image_02489.jpg** | ● | ● | ● |
| **image_02670.jpg** | ● | ● | ● |
| **image_02760.jpg** | | ○ | ○ |
| **image_02851.jpg** | ○ | ● | ● |
| **image_02941.jpg** | | | |
| **image_03032.jpg** | ○ | ● | ● |
| **image_03125.jpg** | | ● | ● |
| **image_03216.jpg** | | ○ | ○ |
| **image_03310.jpg** | ● | ● | ● |
| **image_03401.jpg** | ○ | ● | ● |
| **image_03495.jpg** | ● | ● | ● |
| **image_03588.jpg** | ○ | | ● |
| **image_03678.jpg** | ● | ● | ● |
| **image_03864.jpg** | ● | ● | ● |
| **image_03957.jpg** | ● | ● | ● |
| **image_04048.jpg** | ● | ● | ● |
| **image_04231.jpg** | ● | ● | ○ |
| **image_04322.jpg** | ● | ● | ● |
| **image_04414.jpg** | ● | ● | ● |
| **image_04504.jpg** | ● | | ● |
| **image_04596.jpg** | ● | ● | ● |
| **image_04689.jpg** | ○ | ● | ● |
| **image_04873.jpg** | ● | ○ | ● |
| **image_04967.jpg** | | | ● |
| **image_05058.jpg** | | ● | ● |
| **image_05240.jpg** | ○ | ○ | ● |
| **image_05332.jpg** | | ● | ● |
| **image_05423.jpg** | ● | ● | ● |
| **image_05513.jpg** | ● | ● | ● |
| **image_05605.jpg** | ○ | ● | ● |
| **image_05698.jpg** | ● | ● | ● |
| **image_05789.jpg** | ○ | | ● |
| **image_05879.jpg** | ● | ● | ● |
| **image_05975.jpg** | ● | ● | ● |
| **image_06064.jpg** | ● | ● | ● |
| **image_06157.jpg** | ● | ● | ● |
| **image_06248.jpg** | | ● | ● |

| | | | |
|---|---|---|---|
| **image_06341.jpg** | ● | ● | ● |
| **image_06433.jpg** | ● | ● | ● |
| **image_06525.jpg** | ● | ● | ● |
| **image_06616.jpg** | ● | ● | ● |
| **image_06708.jpg** | ● | ● | ● |
| **image_06799.jpg** | ● | | |
| **image_07070.jpg** | ○ | ● | ● |
| **image_07161.jpg** | | | ● |
| **image_07251.jpg** | ○ | ● | ● |
| **image_07343.jpg** | | | |
| **image_07435.jpg** | ○ | ● | ● |
| **image_07528.jpg** | ● | ● | ● |
| **image_07618.jpg** | ● | ● | ● |
| **image_07807.jpg** | ● | ● | ● |
| **image_10087.jpg** | | ○ | ● |
| **image_10180.jpg** | ○ | ● | ● |
| **image_10273.jpg** | | ● | ● |
| **image_10364.jpg** | ● | ● | ● |
| **image_10458.jpg** | ○ | ● | ○ |
| **image_10551.jpg** | ○ | | |
| **image_10646.jpg** | ○ | ● | ● |
| **image_10740.jpg** | ● | | |
| **image_10831.jpg** | ● | ● | ○ |
| **image_10922.jpg** | ● | ● | ● |
| **image_11013.jpg** | ● | ● | ● |
| **image_11104.jpg** | ● | ● | ● |
| **image_11196.jpg** | ● | ● | ● |
| **image_11288.jpg** | ● | ● | ● |
| **image_11379.jpg** | ● | ● | ● |
| **image_11650.jpg** | ○ | ● | ● |
| **image_11740.jpg** | ● | ● | ● |
| **image_11830.jpg** | ● | ● | ● |
| **image_11938.jpg** | | ● | ● |
| **image_12027.jpg** | ● | ● | ● |
| **image_12127.jpg** | ● | ● | ● |
| **image_12220.jpg** | ○ | ● | ● |
| **image_12310.jpg** | | ● | ○ |

| | | | |
|---|---|---|---|
| **image_12400.jpg** | ● | ● | ○ |

### 8.3.3 Time

*Table 9- The time taken in seconds for each image to be processed using each of the three methods of OCR.*

| File Name | CCVT - Tesseract 3.05 | StanDL - Tesseract 4.1.1 | SpecDL - EAST + Tesseract 4.1.1 |
|---|---|---|---|
| **image_00000.jpg** | 17.06 | 6.65 | 82.98 |
| **image_00079.jpg** | 26.66 | 31.83 | 9.26 |
| **image_00258.jpg** | 8.44 | 10.20 | 18.31 |
| **image_00377.jpg** | 3.21 | 8.14 | 9.71 |
| **image_00414.jpg** | 5.72 | 8.80 | 16.37 |
| **image_00478.jpg** | 15.63 | 7.76 | 8.63 |
| **image_00515.jpg** | 11.76 | 36.43 | 34.17 |
| **image_00689.jpg** | 8.76 | 16.66 | 87.76 |
| **image_00698.jpg** | 36.43 | 26.13 | 74.33 |
| **image_00771.jpg** | 11.93 | 19.85 | 10.13 |
| **image_01147.jpg** | 11.30 | 26.54 | 8.53 |
| **image_01156.jpg** | 6.23 | 24.51 | 10.63 |
| **image_01294.jpg** | 16.47 | 12.18 | 9.00 |
| **image_01384.jpg** | 5.60 | 20.60 | 9.46 |
| **image_01478.jpg** | 31.54 | 18.53 | 11.58 |
| **image_01569.jpg** | 66.62 | 83.22 | 37.1 |
| **image_01660.jpg** | 2.30 | 0.84 | 8.91 |
| **image_01751.jpg** | 6.50 | 2.37 | 41.26 |
| **image_01842.jpg** | 20.58 | 14.21 | 10.31 |
| **image_01936.jpg** | 8.95 | 7.61 | 23.42 |
| **image_02027.jpg** | 6.47 | 26.84 | 11.28 |
| **image_02118.jpg** | 30.14 | 34.35 | 16.25 |
| **image_02209.jpg** | 1.64 | 6.44 | 9.93 |
| **image_02300.jpg** | 86.11 | 92.35 | 38.78 |
| **image_02398.jpg** | 8.85 | 19.99 | 17.04 |
| **image_02489.jpg** | 23.18 | 17.74 | 35.69 |
| **image_02670.jpg** | 10.32 | 32.23 | 72.09 |
| **image_02760.jpg** | 5.31 | 0.65 | 16.61 |
| **image_02851.jpg** | 2.29 | 0.74 | 60.12 |
| **image_02941.jpg** | 3.25 | 2.07 | 10.08 |
| **image_03032.jpg** | 18.84 | 4.18 | 26.55 |
| **image_03125.jpg** | 4.92 | 0.85 | 37.26 |

| | | | |
|---|---|---|---|
| **image_03216.jpg** | 9.34 | 4.36 | 27.31 |
| **image_03310.jpg** | 6.60 | 8.20 | 19.77 |
| **image_03401.jpg** | 5.16 | 6.62 | 16.94 |
| **image_03495.jpg** | 5.05 | 8.84 | 17.61 |
| **image_03588.jpg** | 12.39 | 11.95 | 14.06 |
| **image_03678.jpg** | 3.76 | 8.70 | 52.42 |
| **image_03864.jpg** | 25.62 | 23.82 | 124.46 |
| **image_03957.jpg** | 6.58 | 6.90 | 21.01 |
| **image_04048.jpg** | 25.22 | 21.05 | 15.44 |
| **image_04231.jpg** | 7.39 | 3.72 | 79.55 |
| **image_04322.jpg** | 16.17 | 32.41 | 33.29 |
| **image_04414.jpg** | 6.24 | 8.94 | 34.00 |
| **image_04504.jpg** | 11.00 | 27.54 | 72.69 |
| **image_04596.jpg** | 3.87 | 10.90 | 32.2 |
| **image_04689.jpg** | 15.84 | 12.90 | 16.13 |
| **image_04873.jpg** | 17.51 | 19.88 | 21.15 |
| **image_04967.jpg** | 11.32 | 14.17 | 13.45 |
| **image_05058.jpg** | 11.63 | 12.52 | 14.56 |
| **image_05240.jpg** | 8.26 | 7.31 | 28.35 |
| **image_05332.jpg** | 11.47 | 13.88 | 33.04 |
| **image_05423.jpg** | 2.11 | 8.65 | 37.26 |
| **image_05513.jpg** | 4.53 | 20.86 | 32.34 |
| **image_05605.jpg** | 3.81 | 0.77 | 14.4 |
| **image_05698.jpg** | 11.20 | 41.85 | 80.19 |
| **image_05789.jpg** | 2.10 | 7.27 | 26.08 |
| **image_05879.jpg** | 7.67 | 17.91 | 19.06 |
| **image_05975.jpg** | 10.54 | 1.96 | 44.65 |
| **image_06064.jpg** | 40.28 | 40.83 | 33.77 |
| **image_06157.jpg** | 32.29 | 22.65 | 27.49 |
| **image_06248.jpg** | 27.61 | 27.22 | 31.86 |
| **image_06341.jpg** | 79.39 | 45.56 | 17.31 |
| **image_06433.jpg** | 25.11 | 30.59 | 87.93 |
| **image_06525.jpg** | 6.90 | 7.16 | 15.79 |
| **image_06616.jpg** | 2.52 | 0.61 | 12.02 |
| **image_06708.jpg** | 7.78 | 24.33 | 30.10 |
| **image_06799.jpg** | 5.11 | 4.54 | 30.94 |
| **image_07070.jpg** | 7.13 | 12.98 | 35.24 |

| | | | |
|---|---|---|---|
| **image_07161.jpg** | 4.13 | 3.46 | 9.97 |
| **image_07251.jpg** | 26.49 | 33.88 | 9.66 |
| **image_07343.jpg** | 30.79 | 26.09 | 10.42 |
| **image_07435.jpg** | 24.53 | 18.79 | 37.85 |
| **image_07528.jpg** | 10.40 | 19.30 | 15.15 |
| **image_07618.jpg** | 6.00 | 13.42 | 67.31 |
| **image_07807.jpg** | 23.78 | 30.82 | 135.07 |
| **image_10087.jpg** | 21.07 | 17.16 | 35.03 |
| **image_10180.jpg** | 26.29 | 19.65 | 23.04 |
| **image_10273.jpg** | 5.53 | 19.76 | 11.16 |
| **image_10364.jpg** | 9.08 | 18.38 | 16.15 |
| **image_10458.jpg** | 6.11 | 16.26 | 12.66 |
| **image_10551.jpg** | 23.09 | 27.97 | 93.36 |
| **image_10646.jpg** | 3.76 | 9.50 | 34.08 |
| **image_10740.jpg** | 9.21 | 35.91 | 50.26 |
| **image_10831.jpg** | 5.64 | 4.86 | 34.48 |
| **image_10922.jpg** | 4.72 | 6.33 | 66.4 |
| **image_11013.jpg** | 26.86 | 15.80 | 47.54 |
| **image_11104.jpg** | 8.35 | 18.15 | 27.63 |
| **image_11196.jpg** | 16.62 | 7.86 | 32.34 |
| **image_11288.jpg** | 12.78 | 6.95 | 53.74 |
| **image_11379.jpg** | 27.96 | 30.72 | 167.11 |
| **image_11650.jpg** | 1.69 | 1.54 | 25.62 |
| **image_11740.jpg** | 4.59 | 5.26 | 16.58 |
| **image_11830.jpg** | 0.02 | 0.25 | 52.46 |
| **image_11938.jpg** | 17.59 | 16.40 | 104.81 |
| **image_12027.jpg** | 12.16 | 8.63 | 12.29 |
| **image_12127.jpg** | 15.13 | 22.95 | 159.56 |
| **image_12220.jpg** | 2.26 | 0.74 | 50.1 |
| **image_12310.jpg** | 14.65 | 37.11 | 31.85 |
| **image_12400.jpg** | 3.56 | 0.82 | 47.99 |

### 8.3.4 Native heap memory

| File Name | CCVT - Tesseract 3.05 | StanDL - Tesseract 4.1.1 | SpecDL - EAST + Tesseract 4.1.1 |
|---|---|---|---|
| **image_00000.jpg** | 97 | 112 | 1023 |
| **image_00079.jpg** | 73 | 88 | 1141 |
| **image_00258.jpg** | 77 | 91 | 566 |

| | | | |
|---|---|---|---|
| **image_00377.jpg** | 74 | 90 | 275 |
| **image_00414.jpg** | 77 | 90 | 296 |
| **image_00478.jpg** | 75 | 90 | 499 |
| **image_00515.jpg** | 86 | 100 | 491 |
| **image_00689.jpg** | 100 | 112 | 978 |
| **image_00698.jpg** | 88 | 99 | 531 |
| **image_00771.jpg** | 75 | 91 | 274 |
| **image_01147.jpg** | 75 | 91 | 480 |
| **image_01156.jpg** | 77 | 91 | 487 |
| **image_01294.jpg** | 77 | 92 | 491 |
| **image_01384.jpg** | 77 | 92 | 485 |
| **image_01478.jpg** | 78 | 92 | 492 |
| **image_01569.jpg** | 86 | 97 | 373 |
| **image_01660.jpg** | 79 | 94 | 567 |
| **image_01751.jpg** | 87 | 100 | 583 |
| **image_01842.jpg** | 81 | 96 | 579 |
| **image_01936.jpg** | 85 | 99 | 550 |
| **image_02027.jpg** | 83 | 97 | 545 |
| **image_02118.jpg** | 85 | 99 | 514 |
| **image_02209.jpg** | 84 | 98 | 511 |
| **image_02300.jpg** | 93 | 104 | 376 |
| **image_02398.jpg** | 66 | 80 | 287 |
| **image_02489.jpg** | 73 | 85 | 368 |
| **image_02670.jpg** | 84 | 94 | 528 |
| **image_02760.jpg** | 66 | 80 | 292 |
| **image_02851.jpg** | 80 | 92 | 530 |
| **image_02941.jpg** | 64 | 83 | 714 |
| **image_03032.jpg** | 74 | 86 | 544 |
| **image_03125.jpg** | 80 | 92 | 407 |
| **image_03216.jpg** | 66 | 89 | 315 |
| **image_03310.jpg** | 66 | 89 | 307 |
| **image_03401.jpg** | 65 | 89 | 290 |
| **image_03495.jpg** | 68 | 90 | 308 |
| **image_03588.jpg** | 66 | 92 | 317 |
| **image_03678.jpg** | 79 | 92 | 519 |
| **image_03864.jpg** | 124 | 141 | 1922 |
| **image_03957.jpg** | 66 | 82 | 306 |

| | | | |
|---|---|---|---|
| **image_04048.jpg** | 69 | 84 | 359 |
| **image_04231.jpg** | 99 | 104 | 634 |
| **image_04322.jpg** | 65 | 81 | 336 |
| **image_04414.jpg** | 71 | 82 | 332 |
| **image_04504.jpg** | 86 | 91 | 504 |
| **image_04596.jpg** | 72 | 88 | 383 |
| **image_04689.jpg** | 65 | 86 | 297 |
| **image_04873.jpg** | 73 | 93 | 427 |
| **image_04967.jpg** | 67 | 88 | 282 |
| **image_05058.jpg** | 71 | 92 | 342 |
| **image_05240.jpg** | 80 | 94 | 375 |
| **image_05332.jpg** | 70 | 81 | 331 |
| **image_05423.jpg** | 69 | 84 | 384 |
| **image_05513.jpg** | 68 | 86 | 384 |
| **image_05605.jpg** | 70 | 86 | 342 |
| **image_05698.jpg** | 96 | 108 | 803 |
| **image_05789.jpg** | 68 | 81 | 312 |
| **image_05879.jpg** | 70 | 84 | 341 |
| **image_05975.jpg** | 80 | 94 | 506 |
| **image_06064.jpg** | 69 | 83 | 359 |
| **image_06157.jpg** | 66 | 85 | 358 |
| **image_06248.jpg** | 74 | 87 | 377 |
| **image_06341.jpg** | 70 | 86 | 331 |
| **image_06433.jpg** | 99 | 113 | 885 |
| **image_06525.jpg** | 65 | 80 | 295 |
| **image_06616.jpg** | 66 | 82 | 313 |
| **image_06708.jpg** | 71 | 83 | 331 |
| **image_06799.jpg** | 71 | 86 | 358 |
| **image_07070.jpg** | 76 | 88 | 376 |
| **image_07161.jpg** | 63 | 85 | 577 |
| **image_07251.jpg** | 69 | 86 | 508 |
| **image_07343.jpg** | 67 | 85 | 711 |
| **image_07435.jpg** | 77 | 89 | 569 |
| **image_07528.jpg** | 66 | 88 | 581 |
| **image_07618.jpg** | 85 | 98 | 652 |
| **image_07807.jpg** | 140 | 152 | 2050 |
| **image_10087.jpg** | 74 | 85 | 386 |

| | | | |
|---|---|---|---|
| **image_10180.jpg** | 70 | 84 | 341 |
| **image_10273.jpg** | 68 | 83 | 312 |
| **image_10364.jpg** | 70 | 85 | 331 |
| **image_10458.jpg** | 71 | 84 | 299 |
| **image_10551.jpg** | 118 | 137 | 1728 |
| **image_10646.jpg** | 70 | 85 | 414 |
| **image_10740.jpg** | 78 | 89 | 426 |
| **image_10831.jpg** | 74 | 88 | 382 |
| **image_10922.jpg** | 79 | 91 | 424 |
| **image_11013.jpg** | 83 | 97 | 523 |
| **image_11104.jpg** | 74 | 92 | 344 |
| **image_11196.jpg** | 80 | 95 | 406 |
| **image_11288.jpg** | 88 | 101 | 745 |
| **image_11379.jpg** | 121 | 124 | 1013 |
| **image_11650.jpg** | 67 | 81 | 342 |
| **image_11740.jpg** | 66 | 81 | 343 |
| **image_11830.jpg** | 79 | 92 | 560 |
| **image_11938.jpg** | 105 | 115 | 1063 |
| **image_12027.jpg** | 76 | 93 | 345 |
| **image_12127.jpg** | 126 | 129 | 1012 |
| **image_12220.jpg** | 80 | 93 | 597 |
| **image_12310.jpg** | 72 | 85 | 344 |
| **image_12400.jpg** | 77 | 91 | 445 |

### 8.3.5 Image size & number of bound boxes created by EAST

*Table 10- The file size of each image and the number of bound boxes created by EAST for each image.*

| File Name | Image size (MB) | Width (px) | Height (px) | # of bound boxes created by EAST |
|---|---|---|---|---|
| **image_00000.jpg** | 8.33 | 2550 | 857 | 130 |
| **image_00079.jpg** | 1.51 | 350 | 187 | 16 |
| **image_00258.jpg** | 1.01 | 879 | 322 | 37 |
| **image_00377.jpg** | 0.59 | 350 | 187 | 15 |
| **image_00414.jpg** | 0.93 | 465 | 315 | 59 |
| **image_00478.jpg** | 0.48 | 300 | 157 | 14 |
| **image_00515.jpg** | 14.27 | 1392 | 513 | 52 |
| **image_00689.jpg** | 1.91 | 2550 | 881 | 135 |

| | | | | |
|---|---|---|---|---|
| **image_00698.jpg** | 2.02 | 1488 | 604 | 175 |
| **image_00771.jpg** | 1.43 | 350 | 185 | 18 |
| **image_01147.jpg** | 1.70 | 291 | 156 | 11 |
| **image_01156.jpg** | 2.99 | 450 | 162 | 10 |
| **image_01294.jpg** | 0.77 | 350 | 183 | 13 |
| **image_01384.jpg** | 1.46 | 350 | 189 | 12 |
| **image_01478.jpg** | 551 | 450 | 166 | 13 |
| **image_01569.jpg** | 9.94 | 1004 | 372 | 90 |
| **image_01660.jpg** | 0.17 | 350 | 155 | 10 |
| **image_01751.jpg** | 0.27 | 1110 | 382 | 92 |
| **image_01842.jpg** | 0.71 | 450 | 165 | 10 |
| **image_01936.jpg** | 0.72 | 552 | 470 | 39 |
| **image_02027.jpg** | 3.12 | 450 | 164 | 14 |
| **image_02118.jpg** | 9.60 | 470 | 300 | 46 |
| **image_02209.jpg** | 0.75 | 450 | 165 | 10 |
| **image_02300.jpg** | 9.94 | 749 | 529 | 85 |
| **image_02398.jpg** | 3.63 | 413 | 263 | 38 |
| **image_02489.jpg** | 0.73 | 749 | 529 | 82 |
| **image_02670.jpg** | 1.98 | 1414 | 576 | 164 |
| **image_02760.jpg** | 0.24 | 499 | 245 | 30 |
| **image_02851.jpg** | 0.25 | 1432 | 604 | 136 |
| **image_02941.jpg** | 0.28 | 450 | 163 | 11 |
| **image_03032.jpg** | 1.42 | 917 | 228 | 77 |
| **image_03125.jpg** | 0.21 | 1157 | 471 | 82 |
| **image_03216.jpg** | 1.61 | 642 | 293 | 65 |
| **image_03310.jpg** | 0.28 | 563 | 305 | 49 |
| **image_03401.jpg** | 0.99 | 413 | 263 | 38 |
| **image_03495.jpg** | 0.28 | 563 | 305 | 42 |
| **image_03588.jpg** | 0.53 | 668 | 292 | 25 |
| **image_03678.jpg** | 0.28 | 1076 | 832 | 98 |
| **image_03864.jpg** | 1.51 | 1700 | 2338 | 163 |
| **image_03957.jpg** | 0.41 | 441 | 311 | 31 |
| **image_04048.jpg** | 1.51 | 918 | 340 | 39 |
| **image_04231.jpg** | 1.08 | 1352 | 972 | 246 |
| **image_04322.jpg** | 3.11 | 768 | 291 | 109 |
| **image_04414.jpg** | 0.47 | 768 | 294 | 118 |
| **image_04504.jpg** | 3.30 | 738 | 1033 | 226 |

| | | | | |
|---|---|---|---|---|
| **image_04596.jpg** | 0.28 | 910 | 384 | 107 |
| **image_04689.jpg** | 0.78 | 413 | 263 | 41 |
| **image_04873.jpg** | 2.08 | 1139 | 477 | 78 |
| **image_04967.jpg** | 0.72 | 402 | 129 | 12 |
| **image_05058.jpg** | 0.66 | 957 | 276 | 25 |
| **image_05240.jpg** | 0.41 | 700 | 536 | 62 |
| **image_05332.jpg** | 0.66 | 768 | 292 | 123 |
| **image_05423.jpg** | 0.74 | 838 | 460 | 119 |
| **image_05513.jpg** | 3.42 | 1120 | 343 | 102 |
| **image_05605.jpg** | 0.25 | 383 | 703 | 46 |
| **image_05698.jpg** | 15.16 | 973 | 1736 | 142 |
| **image_05789.jpg** | 0.52 | 625 | 233 | 86 |
| **image_05879.jpg** | 1.19 | 910 | 308 | 39 |
| **image_05975.jpg** | 0.56 | 1359 | 582 | 126 |
| **image_06064.jpg** | 5.01 | 931 | 361 | 92 |
| **image_06157.jpg** | 0.85 | 1080 | 319 | 82 |
| **image_06248.jpg** | 0.86 | 998 | 414 | 81 |
| **image_06341.jpg** | 2.90 | 750 | 304 | 43 |
| **image_06433.jpg** | 1.33 | 1463 | 1336 | 190 |
| **image_06525.jpg** | 0.41 | 600 | 172 | 36 |
| **image_06616.jpg** | 0.18 | 615 | 282 | 15 |
| **image_06708.jpg** | 2.07 | 832 | 262 | 102 |
| **image_06799.jpg** | 0.20 | 800 | 403 | 86 |
| **image_07070.jpg** | 1.01 | 924 | 422 | 97 |
| **image_07161.jpg** | 2.72 | 179 | 240 | 12 |
| **image_07251.jpg** | 1.43 | 417 | 281 | 12 |
| **image_07343.jpg** | 0.86 | 179 | 240 | 12 |
| **image_07435.jpg** | 1.47 | 340 | 867 | 124 |
| **image_07528.jpg** | 1.47 | 667 | 174 | 41 |
| **image_07618.jpg** | 1.03 | 1370 | 898 | 193 |
| **image_07807.jpg** | 6.44 | 3024 | 1392 | 194 |
| **image_10087.jpg** | 0.56 | 916 | 431 | 97 |
| **image_10180.jpg** | 1.07 | 934 | 285 | 79 |
| **image_10273.jpg** | 3.02 | 464 | 336 | 19 |
| **image_10364.jpg** | 1.28 | 917 | 267 | 39 |
| **image_10458.jpg** | 1.31 | 575 | 218 | 21 |
| **image_10551.jpg** | 1.58 | 2200 | 1700 | 114 |

| | | | | |
|---|---|---|---|---|
| **image_10646.jpg** | 0.87 | 1153 | 434 | 81 |
| **image_10740.jpg** | 7.46 | 626 | 846 | 196 |
| **image_10831.jpg** | 0.39 | 993 | 379 | 105 |
| **image_10922.jpg** | 0.66 | 1030 | 435 | 125 |
| **image_11013.jpg** | 0.83 | 1064 | 737 | 125 |
| **image_11104.jpg** | 1.23 | 323 | 622 | 85 |
| **image_11196.jpg** | 8.57 | 973 | 394 | 92 |
| **image_11288.jpg** | 3.43 | 723 | 1996 | 113 |
| **image_11379.jpg** | 1.49 | 1872 | 1392 | 557 |
| **image_11650.jpg** | 0.16 | 560 | 336 | 85 |
| **image_11740.jpg** | 0.45 | 763 | 246 | 54 |
| **image_11830.jpg** | 0.16 | 1337 | 611 | 124 |
| **image_11938.jpg** | 1.12 | 2032 | 1238 | 209 |
| **image_12027.jpg** | 0.44 | 745 | 266 | 15 |
| **image_12127.jpg** | 4.69 | 1872 | 1392 | 533 |
| **image_12220.jpg** | 0.25 | 1416 | 672 | 120 |
| **image_12310.jpg** | 16.06 | 723 | 268 | 103 |
| **image_12400.jpg** | 0.25 | 1192 | 436 | 158 |