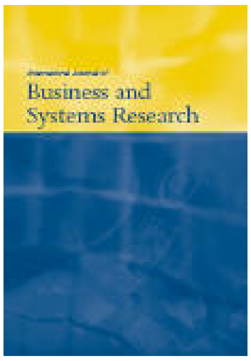


International Journal of Business and Systems Research

ISSN online: 1751-2018 - ISSN print: 1751-200X
https://www.inderscience.com/ijbsr

E-commerce assistant application incorporating machine learning image classification

Victor Chang, Robert Marshall, Qianwen Ariel Xu, Anastasija Nikiforova

DOI: 10.1504/IJBSR.2022.10036200

Article History:

Received:	18 August 2020
Accepted:	11 October 2020
Published online:	15 December 2022

E-commerce assistant application incorporating machine learning image classification

Victor Chang* and Robert Marshall

Artificial Intelligence and Information Systems Research Group,
School of Computing, Engineering and Digital Technologies,
Teesside University,
Middlesbrough, UK
Email: victorchang.research@gmail.com
Email: rm133094@gmail.com
*Corresponding author

Qianwen Ariel Xu

Artificial Intelligence and Information Systems Research Group,
School of Computing, Engineering and Digital Technologies,
Teesside University,
Middlesbrough, UK
and
IBSS,
Xi'an Jiaotong-Liverpool University,
Suzhou, China
Email: iamarielxu@163.com

Anastasija Nikiforova

Faculty of Computing,
University of Latvia,
Raiņa Boulevard 19, Riga, LV-1586, Latvia
Email: nikiforova.anastasija@gmail.com

Abstract: The rise of mobile applications has helped to provide information in a broader network of products remotely. They simplify the identification of products by using their barcode or even an image of the item. This paper, therefore, aims to create an e-commerce assistant Android application that incorporates machine learning, more precisely, image classification, to assist potentially disadvantaged people on a tight budget in looking to save money. This is achieved by collecting an image dataset of essential products, training a machine learning model, and applying it to the developed application. Although the proposed solution appears to be useful and consistent with all the defined goals, the available datasets are currently insufficient, taking into account both their size and quality of individual images, which negatively affect the machine learning model and limit the potential of the solution being developed. We concluded that our final product succeeds in serving the basic functionality of the app's requirements. In the future, we will reach a wider network of users and investigate their needs, then develop these functions into the application.

Keywords: e-commerce applications; e-commerce; electronic commerce; machine learning; image classification; disadvantaged people.

Reference to this paper should be made as follows: Chang, V., Marshall, R., Xu, Q.A. and Nikiforova, A. (2023) 'E-commerce assistant application incorporating machine learning image classification', *Int. J. Business and Systems Research*, Vol. 17, No. 1, pp.1–26.

Biographical notes: Victor Chang is a Full Professor of Data Science and Information Systems at the SCEDT, Teesside University, Middlesbrough, UK since September 2019. He has 22 years of experience in IT and academia. Within four years, he completed his PhD (CS, Southampton) and PGCert (Higher Education, Fellow, Greenwich) while working full time. He achieved 97% on average in 27 IT certifications. He won many awards and best papers including Outstanding Young Scientist 2017. He is an editor of several top journals. He founded four international conferences and gave 18 international keynotes. He is regarded as a world-leading young scientist in data science/IoT/AI/security/IS.

Robert Marshall graduated with a first class honours BSc in Computing. He was under Prof. Chang's supervision for one year. He is a good programmer, particularly proficient in Python.

Qianwen Ariel Xu completed her MSc in Business Analytics with Distinction at the University of Liverpool and Xi'an Jiaotong-Liverpool University, in 2020. She has worked with Prof. Victor Chang in the last 1.5 years part time. She is a capable, efficient and hardworking researcher under Prof Chang's guidance. She will work towards her PhD under Prof Chang's supervision.

Anastasija Nikiforova is a researcher and Lecturer at the Faculty of Computing, University of Latvia, where she has received her PhD in Computer Science and Informatics. Her current researches include data quality, IoT, open government data, MBT as well as concurrency in business processes. She is currently involved in two ERDF co-funded projects. She is an associate member of the Latvian Open Technology Association. She also serves as a program and organising committee for several conferences. In 2019, she was nominated for the International WDS Data Stewardship Award as one of the most exceptional contributors of early career researchers.

1 Introduction

Nowadays, the e-commerce term is becoming more and more popular and widespread for many purposes. Users have a wide variety of products that have many alternatives and analogues. As a result of this trend, customers are no longer able to choose products and services effectively. Even more, this is the case for those socially disadvantaged groups of individuals affected by poverty or with a tight budget aimed to save the money. The presented study deals with this issue. Compared with a more popular cluster of studies focusing mainly on recommending systems that provide more suitable item selection, which focuses on analysis derived from user behaviour or cross-users' analysis (Cho and Kim, 2004; Schafer et al., 2001). Therefore, we focus on the topic of more social importance by presenting an e-commerce assistant application that allows the user to find a cheaper product taking into account the location of the user by capturing an item image or a barcode.

This study aims to create an e-commerce assistant application that incorporates our own image classification model. After completion, the application should allow a user to capture the image or barcode of an item and then access collated pricing data related to the given item and the information on where to purchase the product locally. Although our application shares some similar functions with the existing applications, such as identifying items with barcodes, the uniqueness of our application is that it is planned to directly identify items through the image. This function is more convenient for users when they do not know what the item is or without barcodes.

The study presents two main components of the study conducted:

1 the machine learning model on the image classification

2 the application using the developed machine learning model forming user interface.

The first section addresses the machine learning model on the image classification, which encompasses all the artificial intelligence (AI) elements of the project and mainly involves creating a large image dataset and a machine learning model. The model is then trained using the image dataset and imported into the predeveloped Android application. The selection of each component is based on a comprehensive literature review aimed at facilitating selection and improving the final output of the study. The final image dataset used was a combination of two existing datasets [Caltech 101 (Vision.caltech.edu, 2020) and Fruits 360 (Kaggle.com, 2020) datasets]. The data contains images of three kinds of fruit, which are apples, oranges and pineapples. Additionally, the dataset will be divided into the train set and the validation set.

The second section of the project is devoted to the Android mobile application and its interface. The application will harness the results from the image classification. It will show a user the lowest local prices of the selected product and the location of the stores in question through a pipeline.

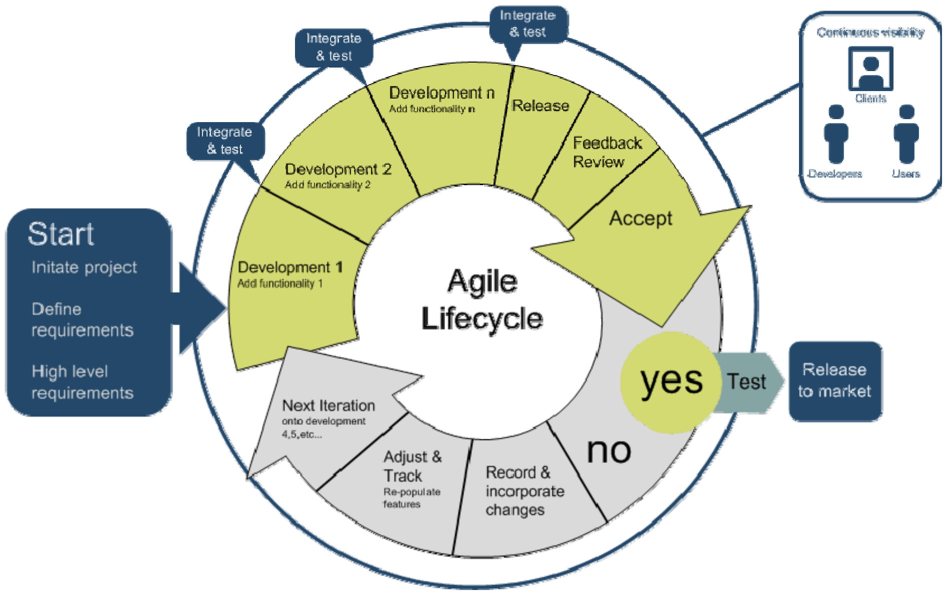
The final product serves the app's basic functionality while there is plenty of room for expansion and improvement in all functions. The machine learning model is not as accurate as initially planned. This is mainly due to the dataset used to train the model, which consisted of images that were not informative enough to pick up trends and patterns within the data. Additionally, the datasets used were not fully balanced, which led to the model becoming biased towards certain classes.

The paper is presented as follows: the methodology (Section 2), a brief overview of the existing applications (Section 3), design of the application (Section 4), implementation through the machine learning (Section 5), constraints (Section 6), testing and evaluation of the presented solution (Section 7), discussion (Section 8), future work (Section 9) and conclusions (Section 10).

2 Methodology

The agile development methodology shown in Figure 1 has been strictly adhered to project development throughout the development process. As part of this weeklong sprint, the methodology has been used to keep track of progress and keep consistency. The sprints in question usually involve working on a certain new feature by looking at the requirements for the project, designing a feature, then implementing it. In line with this methodology, the feature is added to the main project only when the test results are met. Otherwise, the process is iterated again.

Figure 1 Agile development process (see online version for colours)



Source: Saigon Technology (2020)

3 Research

Before any design or implementation, as well as during the project process, the research of this project was undertaken to gain knowledge in the domain of this project. The research involved studying various websites for more specific practical research and academic papers to improve understanding of existing research and well-renowned processes.

3.1 Existing applications

As previously mentioned, nowadays, e-commerce solutions and recommendation systems are becoming more popular. Therefore, we have addressed a few already available solutions to gain an overview of what has been achieved in this subject area. Our attention was mainly drawn to such applications as:

1 RedLaser

2 ShopSavvy

3 The Find.

The above apps were tested to see the range of functionality found that these apps are mainly aimed at reading the barcode of an item and bringing back-related data about the product. While the functionality provided by them compared to the application under development seems to be similar, they do not cover the entire range of functionality of the app developed in the scope of this study, thus, demonstrating the uniqueness of this app, where perhaps the most significant difference of the developed app is an ability to gain information about the product and possible cheaper alternatives by means of not only a barcode but also an image of the product.

Also, another research was done into the use of existing image classification and the API (Imagga Visual search) it employed seems to be based on a similar idea to this project. This aspect seems to be one of the crucial for this study because by making product discovery easier, API enables users to find the closest possible or exact representation of the item they are searching for. It is also known that although there are many different solutions for image identification and classification, general and universal solutions are still lacking. Most of the existing solutions are domain-specific, i.e., those used for bill changing machines, optical character readers, blood cell analysers, robotic welders, electronic circuit inspectors, etc. (Sternberg et al., 2004). And while one of these tasks may be done perfectly, others will not be successful because, according to Sternberg et al. (2004), the only universal recognition mechanism by now is the human brain. Therefore, it is crucial to identify the most suitable task-specific technology that is further used for the proposed solution.

Many websites were viewed throughout the research (Semantics3.com, 2020; Barcodelookup.com, 2020; Google Cloud, 2020; Google Developers, 2020a, 2020b; Imagga Blog, 2020a, 2020b), which helped provide insight into the current technologies and the primary uses of image recognition seemed to be:

1 automated image organisation (Charania and Berger, 2016)
2 stock photographs
3 a visual search for improved product recognition.

3.2 Machine learning and image classification

Machine learning is a sub-category of AI, allowing a system to learn from given data to perform decision-making independently, i.e., without any human interaction (Abadi et al., 2016). This means that a system looks for reoccurring patterns in data, gaining a sense of knowledge and experience, thus, transforming data into knowledge. Generally, the machine learning process follows such steps as:

- *Data collection* – Collecting the target data to train the model.
- *Data preparation* – Preparing the data for training by removing duplicates or errors.
- *Data augmentation* – Using slight variations in existing data to increase the diversity of the dataset without collecting any new data.
- *Select a model* – Choosing a model which uses the correct algorithm for the task working on.
- *Train the model* – Giving the model a target, i.e., correct answer. The algorithm will then find any patterns/attributes in the training data that link to the target.

- *Evaluate the model* – Using some form of metrics to measure the model's performance.

Image classification is a type of machine learning in which a system is trained to detect an image representing its classes of images. The image classification model training gives images with their associated class name, which the model will learn to identify.

Kamavisdar et al. (2013) explained the various methods and approaches used for image classification. Their paper also helped acquire knowledge about the steps taken for image classification, meaning the project could follow a tried and tested process.

TensorFlow (2020a, 2020c) core is an open-source platform for machine learning, allowing users to create machine learning models on any platform flexibly. This platform introduces image classification and specifically provided information on the recommended workflow and the steps to use this framework.

Perez and Wang (2017) introduced the concept of data augmentation (the third step in the list mentioned above), which involves increasing the diversity of data given to the training model without collecting any new data. Methods used include padding, flipping, or cropping existing images within the dataset to capture a slightly altered version of existing data.

Considering that authors found out that data augmentation produces 'promising ways to increase the accuracy of classification tasks', this paper, therefore, incorporates data augmentation into the workflow.

Besides, the *feature extraction*, which is a type of image pre-processing, involves transforming input data into a set of features used to reduce the time needed to classify the images and examined in detail (Sharmila, 2014), was found to be appropriate for the study.

4 Design

4.1 Application requirements

The main functions available for the user in the developed applications are:

- scan and identify products by image or a barcode
- view the average and lowest prices of a chosen product
- view local shops that stock a chosen product
- view similar/related products.

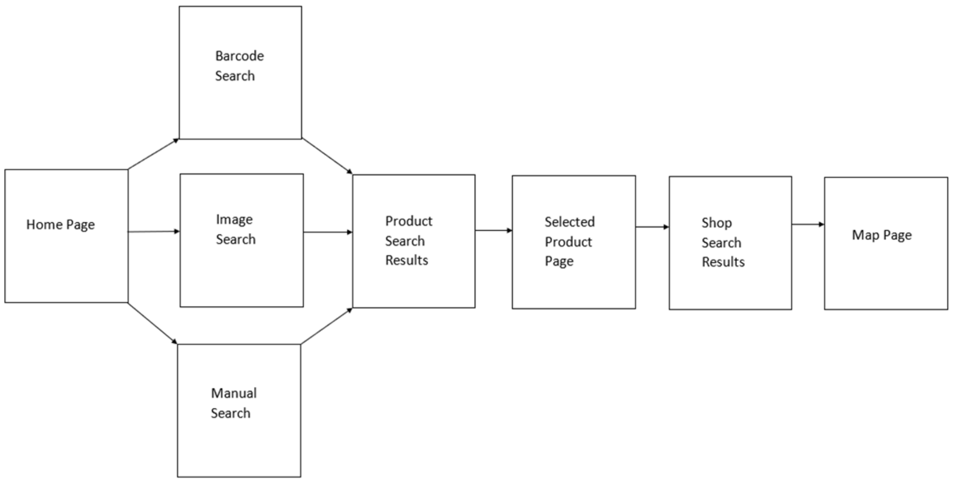
This list of requirements to be fulfilled by the Android application was used at the designing stage facilitating the decision on the content of each page of the app to contain at a high-level and how to allow a user to browse these pages. The main flow of the application to be implemented is shown in Figure 2. According to this flow, a user can choose one of three search options:

1. a barcode search that appears as one of the most popular features for this type of applications
2. manual search – perhaps, the most classic type of searching since the beginning of developing such systems

3 an image search that, according to our analysis, seems to be one of the most unique search types.

After the search type is selected, a page consisting of search results is provided to the user, which, when a specific option is selected, displays a list of shops that the user can also choose, and returns the map page, which helps the user to find out the location of the store that offers the selected item at the lowest price.

Figure 2 The flow of the Android application



4.2 Interfaces

After defining the application requirements and flow, the next step was to develop an interface. In the interface development scope, the colour scheme and the design of the logo were also addressed. The preference was given to lighter colours for the colour scheme, as it makes shopping applications like Amazon, eBay, Gearbest, AliExpress, etc. In addition, our own experiment with a darker two-colour scheme proved to be unsuccessful, more precisely by making the text harder to read, but because of the nature of the application, it is necessary to achieve straightforward and easy-to-read content. The logo of this application was then developed, following the main colour scheme according to principles such as simplicity, memorabilia, and easy recognisability (see Figure 3).

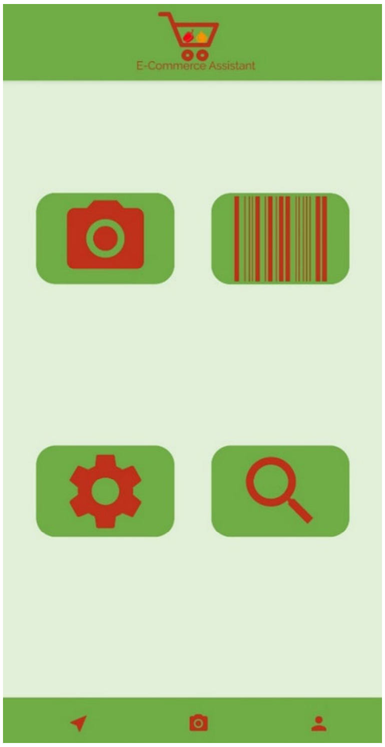
4.2.1 Homepage

The homepage allows users to navigate every type of input or search, whether that be a camera, manual search, or barcode, as well as a setting option (Figure 3). This page also has a straightforward and open layout, making the app easy to understand and use. Besides, as part of this simple design, the emblems on the buttons – glyphicons are large and self-explanatory, allowing the app for:

1 people in different countries or one country but speaking different languages

2 elderly people since the design is very intuitive.

Figure 3 App homepage (see online version for colours)



4.2.2 Image result page

Once a user selects an image from the camera or gallery, it is classified, and the result page is displayed to clarify the meaning of the image (Figure 4). The aim is to allow the user to double-check the correctness of the results again before continuing to work with the app. A straightforward design and layout have been used because the purpose of this page is only to show the result of the image classification that will be addressed in Section 3.3.

4.2.3 Product price page

When a user selects to view product prices, the price page (Figure 5) is shown, giving users the best-priced instances of the desired product. The results are presented in the list ordered with the cheapest item at the top, which corresponds to the app's primary function of helping users save money on shopping.

Figure 4 Image result page (see online version for colours)

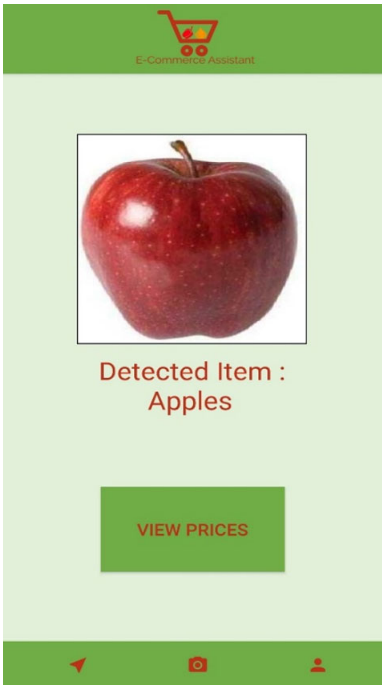
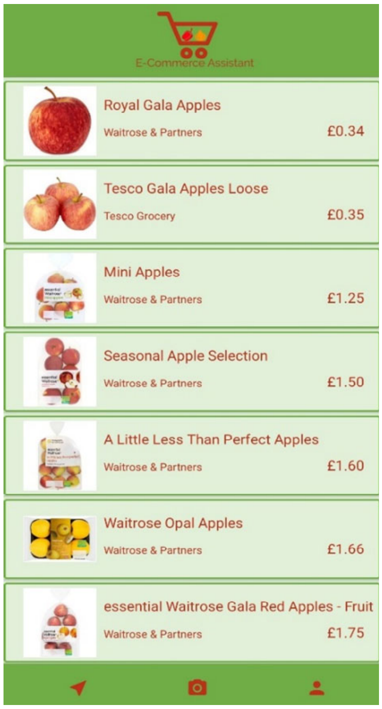
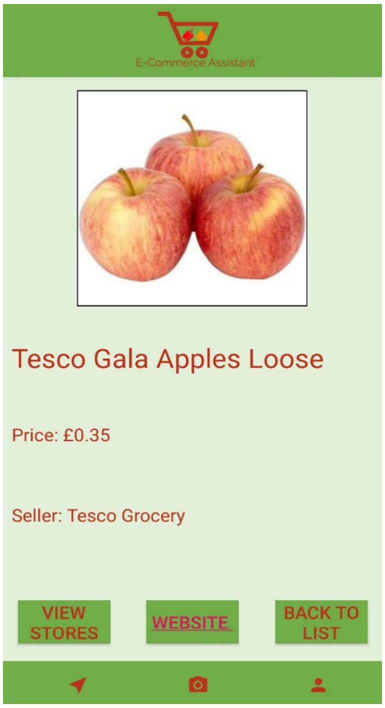


Figure 5 Product price page (see online version for colours)



Royal Gala Apples	Waitrose & Partners	£0.34
Tesco Gala Apples Loose	Tesco Grocery	£0.35
Mini Apples	Waitrose & Partners	£1.25
Seasonal Apple Selection	Waitrose & Partners	£1.50
A Little Less Than Perfect Apples	Waitrose & Partners	£1.60
Waitrose Opal Apples	Waitrose & Partners	£1.66
essential Waitrose Gala Red Apples - Fruit	Waitrose & Partners	£1.75

Figure 6 Selected product page (see online version for colours)

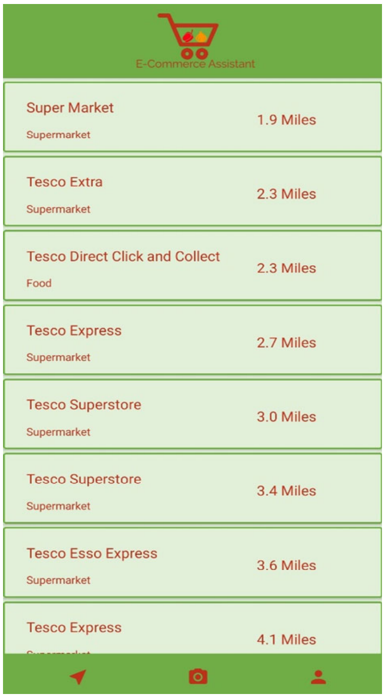


Tesco Gala Apples Loose

Price: £0.35

Seller: Tesco Grocery

Figure 7 Store page (see online version for colours)



Super Market Supermarket	1.9 Miles
Tesco Extra Supermarket	2.3 Miles
Tesco Direct Click and Collect Food	2.3 Miles
Tesco Express Supermarket	2.7 Miles
Tesco Superstore Supermarket	3.0 Miles
Tesco Superstore Supermarket	3.4 Miles
Tesco Esso Express Supermarket	3.6 Miles
Tesco Express	4.1 Miles

4.2.4 *Selected product page*

When an item is selected from the product price page, the selected product page is shown (Figure 6). This page is in keeping with the simple layout of the previous pages and seeks to provide the user with information on the product chosen.

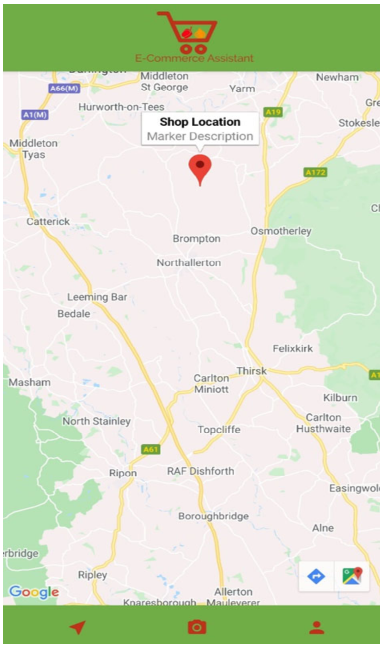
4.2.5 *Store page*

If a user selects 'view stores', the store page will then be shown. This lists the local branches of the chosen shop and sorts them in ascending order of distance to reduce the travel time for the user and to prioritise the use of local shops.

4.2.6 *Map page*

When a user selects a store, another screen will open, showing a map with two markers (the user's current location and the location of the desired shop). On clicking the destination marker, the user can then receive directions to the location via Google.

Figure 8 Map page (see online version for colours)



5 Implementation

5.1 *Machine learning and image classification*

The implementation of the proposed solution supposes the use of image classification. Therefore, this section addresses machine learning, which involves collecting training data or images and constructing an image classifier.

5.1.1 Machine learning model

The machine learning model was created in the development environment PyCharm and used the TensorFlow machine learning framework. The initial machine learning model was simple enough and was created following the online TensorFlow (2020a, 2020b) tutorials for image classification covering various practices aiming to achieve the highest accuracy, which is crucial for this study.

The process used in this project is as follows:

1 *Load data:* The dataset directory is declared, and both the train and validation directories are defined.

2 *Preparation and augmentation of data:* The images are loaded from the disk, converted to floating-point tensors, and then rescaled from 0 to 255 to a value between 0 and 1 because neural networks prefer smaller input values.

  One more point to cover here is *overfitting* we avoided successfully. Overfitting refers to a model that is modelled too well on the training data and will obtain irrelevant and random information from the dataset, which will impact the model's overall performance negatively. We avoided it by the augmenting training dataset to create a more substantial amount of training examples without collecting any brand-new data. This was achieved through several processes applied to the images under analysis, including:

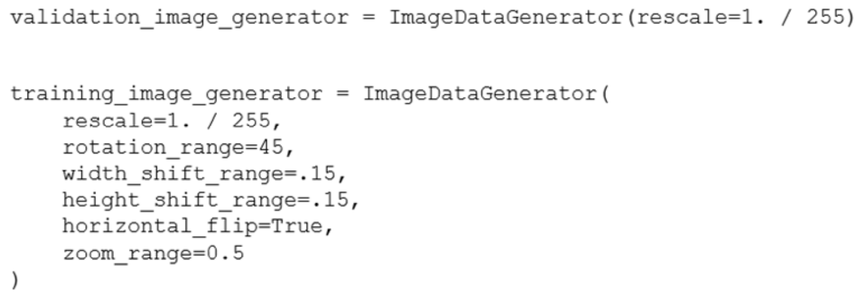
  a horizontal flip

  b random rotation

  c zoom augmentation.

  The example code of both processes showing the validation and training image generators is shown in Figure 9.

Figure 9 Data augmentation code



```
validation_image_generator = ImageDataGenerator(rescale=1. / 255)


training_image_generator = ImageDataGenerator(
    rescale=1. / 255,
    rotation_range=45,
    width_shift_range=.15,
    height_shift_range=.15,
    horizontal_flip=True,
    zoom_range=0.5
)
```

  Only the training data is augmented.

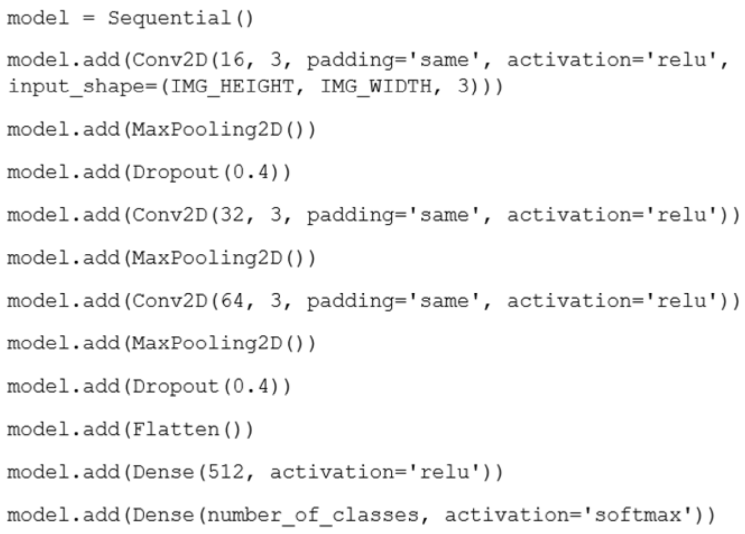
3. *Creation of model:* The model is created using three main types of layers:

  a *A convolutional layer* used to retrieve features from an image or form part of an image.

  b *A pooling layer* used to reduce the number of attributes of every single feature to highlight the most critical aspects (several iterations of convolutional and pooling layers are typically used).

c *A fully connected layer* is used to take the features found in the previous layers in a flattened form, which are then used to make a prediction. Additionally, to these three layers, a technique known as *dropout* was used, which is commonly used in machine learning to combat overfitting and involves randomly ignoring a selection of layer outputs. This helps to let the next layer be treated as a layer with a different number of nodes to the previous layer. The code used to construct the model is shown in Figure 10 illustrating how the various layers are stacked on top of each other to form the complete model.

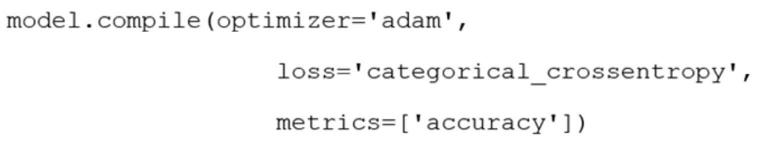
Figure 10 Model construction code



```
model = Sequential()

model.add(Conv2D(16, 3, padding='same', activation='relu',
input_shape=(IMG_HEIGHT, IMG_WIDTH, 3)))

model.add(MaxPooling2D())

model.add(Dropout(0.4))

model.add(Conv2D(32, 3, padding='same', activation='relu'))

model.add(MaxPooling2D())

model.add(Conv2D(64, 3, padding='same', activation='relu'))

model.add(MaxPooling2D())

model.add(Dropout(0.4))

model.add(Flatten())

model.add(Dense(512, activation='relu'))

model.add(Dense(number_of_classes, activation='softmax'))
```

4 *Compiling the model:* The model is then compiled. For this project, the ‘adam’ optimiser (Brownlee, 2020) has been chosen as it is straightforward to implement; it does not require large amounts of memory and is well suited for large datasets. Also, for this project, the ‘categorical cross-entropy’ loss function was used. This is because this function is used for ‘multi-class’ classification, i.e., a classification task with more than two exclusive classes in the dataset. See the code of how the model is compiled in Figure 11.

Figure 11 Compiling model code



```
model.compile(optimizer='adam',

                       loss='categorical_crossentropy',

                       metrics=['accuracy'])
```

5 *Train the model:* The training of the model starts using the training data provided and iterate for as many *epochs* – a complete pass over the data, as is deemed necessary. Usually, a substantial number of epochs were chosen as this yield’s better

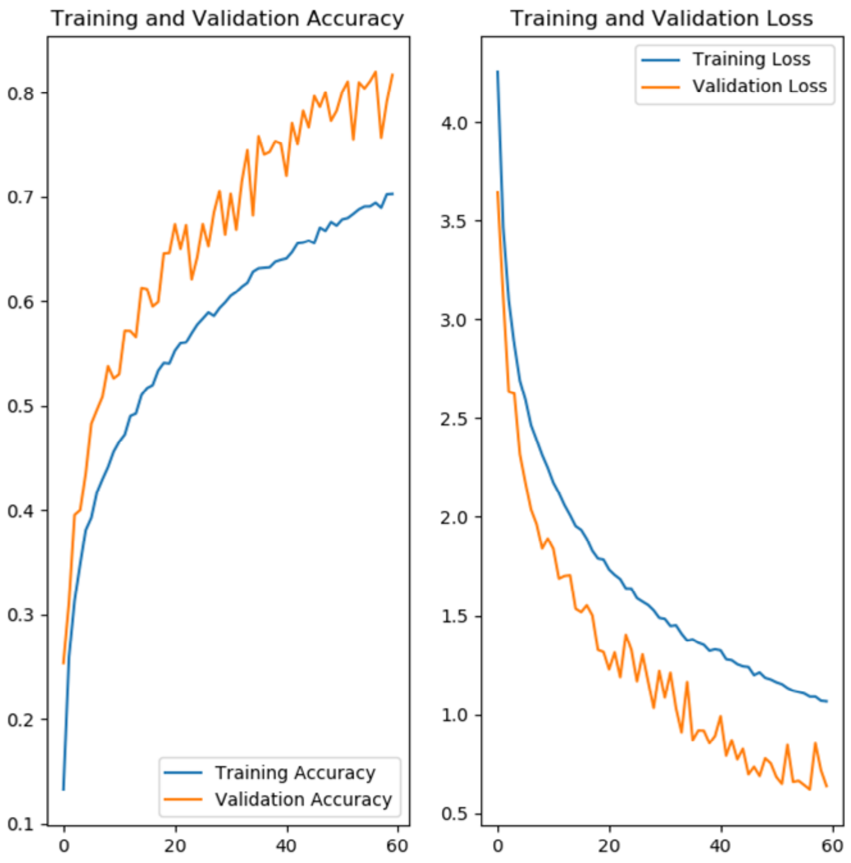
accuracy. The accuracy was closely checked at regular intervals to monitor the progress of the model.

6 *Test the model:* To test the model, a single image was passed to the model, and the predicted class was checked against the image. However, only a model with a substantial level of accuracy was considered for testing.

7 *Visualise training results:* The results of the training were then generated in the form of two graphs. One of them shows the training and validation accuracy, which should increase as the process continues. Another graph shows the training and validation loss, which, in contrast, should decrease over time.

Figure 12 provides a set of training results that accurately exhibit this behaviour, where *training accuracy* refers to "How the model is progressing in terms of the training?', *validation accuracy* – "How well the model performs when making predictions on data it has not yet seen (the validation dataset)?", *training loss* – "The summation of all the errors made with the training set", and *validation loss* – "The summation of all the errors made with the validation set."

Figure 12 Example training results (see online version for colours)



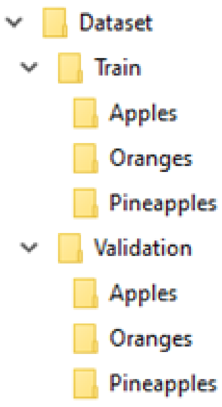
5.1.2 Datasets

In order to train the machine learning model, an image dataset was required. To this end, a number of pre-compiled online datasets were studied to find the most suitable for reuse in this project. One such dataset was found (opensource.google, 2020), which seemed to be the largest and most extensive available dataset. However, due to a lack of equipment with large enough memory to store such a large dataset and sufficient processing power, it was not used. However, hypothetically this dataset would have yielded the most accurate and varied results allowing the machine learning model to classify the largest number of objects.

Therefore, there was an inevitable compromise in the diversity of accuracy of the data that would be used. The final dataset used was a combination of Caltech 101 (Vision.caltech.edu, 2020) and Fruits 360 (Kaggle.com, 2020) datasets.

Figure 13 shows the final dataset structure obtained as combining two existing datasets used in this project. The 'train' directory contains the images used to train the model, whereas the 'validation' directory holds the images used to assess the performance of the model.

Figure 13 Example dataset folder structure (see online version for colours)



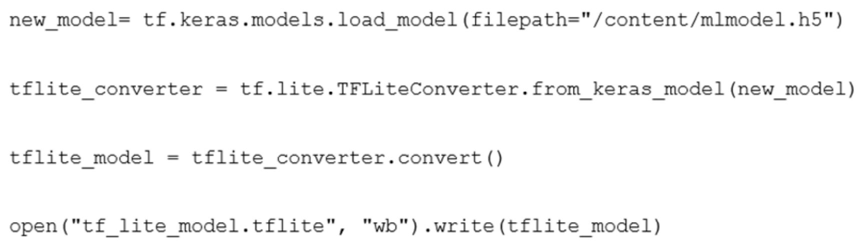
5.2 Android application

This section involves the construction of the Android application. The app is constructed using the main activity as the base, which contains a view pager. Each page of the app is added to this view pager as a fragment. All the fragments access the same main activity, which allows a user to quickly navigate through the app, passing information through the pages easily without the need for multiple activities.

5.2.1 Importing machine learning model

In order to import the machine learning model addressed in the previous section, the model file first had to be converted to a 'tflite' (TensorFlow, 2020d) format explicitly designed for use on mobile devices. The model could then be loaded into the Android application. The code used to convert the model is shown in Figure 14.

Figure 14 Tflite conversion code



```
new_model= tf.keras.models.load_model(filepath="/content/mlmodel.h5")

tflite_converter = tf.lite.TFLiteConverter.from_keras_model(new_model)

tflite_model = tflite_converter.convert()

open("tf_lite_model.tflite", "wb").write(tflite_model)
```

5.2.2 Data inputs

The home screen of the app contains all the buttons that take the user to a given type of data input, which could be in the form of:

a An image was taken with a camera or an image selected from the phone gallery, where a user is asked to choose from an image taken from the camera or previously saved in the gallery. Once the image is selected, the image will be passed to the image classifier as a bitmap. From here, the image is converted once again to a byte buffer to be classified. The class with the highest probability is then returned.

b A bar code image, where a user can scan a product's barcode using the Google mobile vision barcode detector (Google Developers, 2020a, 2020b, 2020c), which on sensing a barcode, returns the raw numerical value.

c A manual product search where a user enters the name of the product he/she wish to search for, bypassing the image classification stage and going straight to the search results page (see Figures 3 to 5).

5.2.3 Product search results

Once a product name was received from one of the inputs, a call to the 'ZenSerp Google Search Result API' (Zenserp.com, 2020) is made, simulating a search using Google shopping. This is then parsed and returned as JSON. A snipped of this output can be seen in Appendix showing one search result. This output is then parsed into a 'recyclerview' showing each item in a list and is sorted by price in ascending order.

5.2.4 Selected product

If a user wants more information about a product, the item in the list should be selected. A new page will then be opened and show more details about the product, including a link to the products website and a button to find local shops that sell the selected product.

5.2.5 Local shop results

When a user has requested to see local stores that sell a certain product, a call is made to the Google Maps 'place search' (Google Developers, 2020a, 2020c, 2020d) API. Then, the app returns a JSON of map results parsed into a list sorted in ascending order by their distance to the user. A snippet of this output can be seen in Appendix. A map is shown with one marker at the user's current location and the desired shop location when a shop is selected.

6 Constraints

During the study, we have met a list of constraints that we had to address, including:

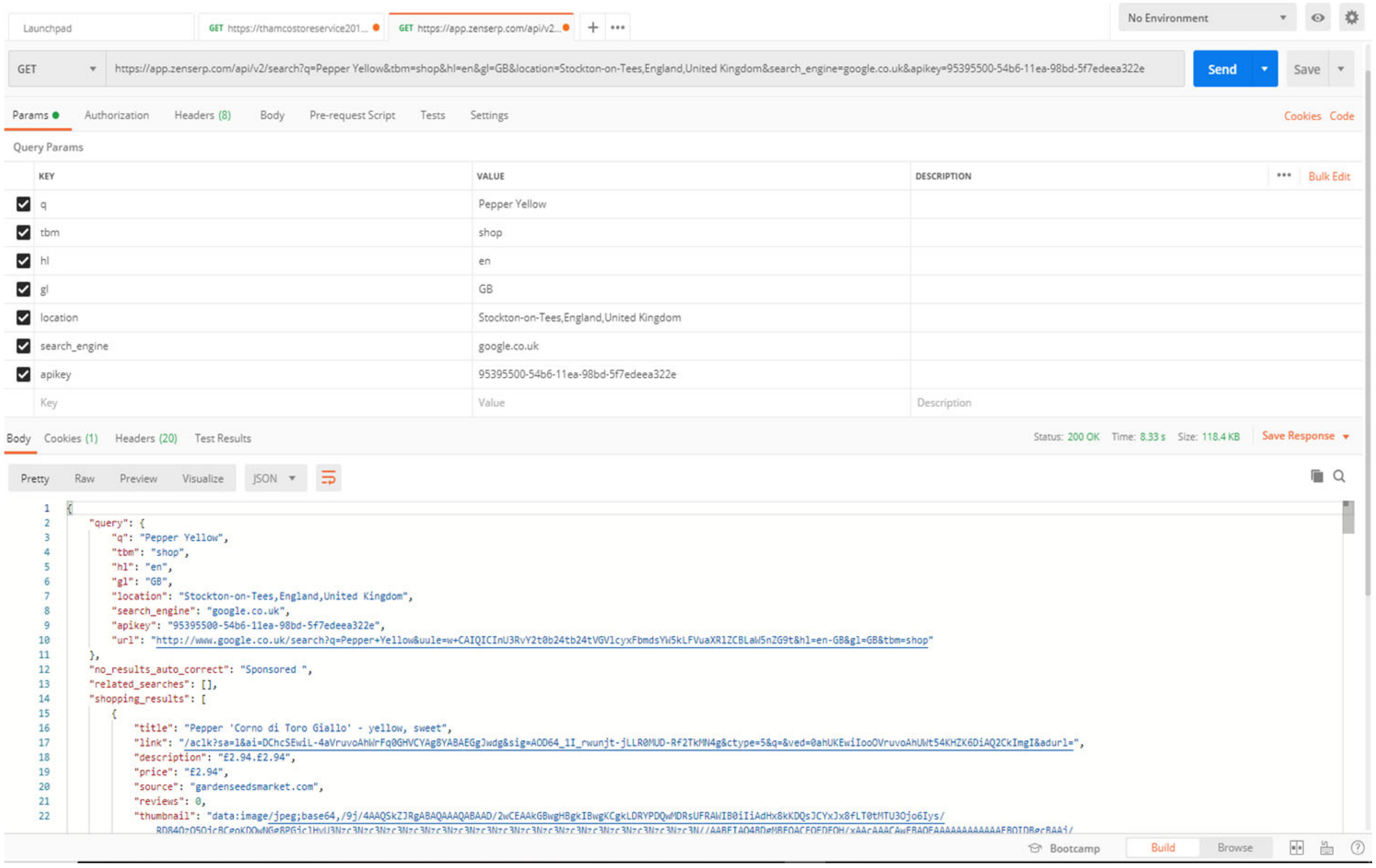
1. *Lack of API calls for shopping results:* According to the abovementioned, the main resource for obtaining shopping results came from a third-party API [Zenserp Google shopping API (Zenserp.com, 2020)], which after the study on possible alternatives, was found to be the best option to be used. However, there is a significant constraint related to the number of requests allocated to the API each month, which is only 50. In order to combat this, one search result was saved locally and used to populate the app for testing and then for demonstration purposes, the real endpoint was used. If this project were to be scaled up commercially, then a professional payment plan would be arranged with the API to allow more requests.
2. *Lack of suitable image datasets:* After extensive research into pre-compiled image datasets which suited the needs of the project, it was found that there was no definitive existing dataset that could be used for this project. In order to combat this, multiple datasets were used to create one image dataset. However, as we have already emphasised, the presence of more datasets would positively impact the application that has been developed, providing richer results in terms of different product types.
3. *Lack of memory and processing power available* on the device holding the dataset. If a much larger dataset could have been stored, it would have yielded much more accurate results. Therefore, there was a compromise of accuracy because of the limits of memory. Also, much larger pre-compiled datasets were found but could not be used. In addition to this, the device used had a lower amount of RAM (4 GB) than was advised by TensorFlow. Therefore, training models were a prolonged process at times.
4. Another major constraint on this project was the global outbreak of coronavirus in 2020, which slowed progress for various reasons, from the apparent slowdown of the Internet to the difficulty of obtaining feedback.

7 Testing and evaluation

Due to the functionality of the application relying on external API calls to retrieve results, it was essential to test the API calls to assure no errors were returned and to understand the format of the JSON returned. This way, the data could be correctly parsed for use in the app.

The method of testing involved using the Postman (2020) API testing tool, which allows calling an API endpoint, analysing the results and whether debugging is required. This method permitted API calls to be tested before incorporating such requests into the app itself.

Figure 15 Testing shopping results API (see online version for colours)



GET https://app.zenserp.com/api/v2/search?q=Pepper Yellow&tbm=shop&hl=en&gl=GB&location=Stockton-on-Tees,England,United Kingdom&search_engine=google.co.uk&apikey=95395500-54b6-11ea-98bd-5f7edeea322e

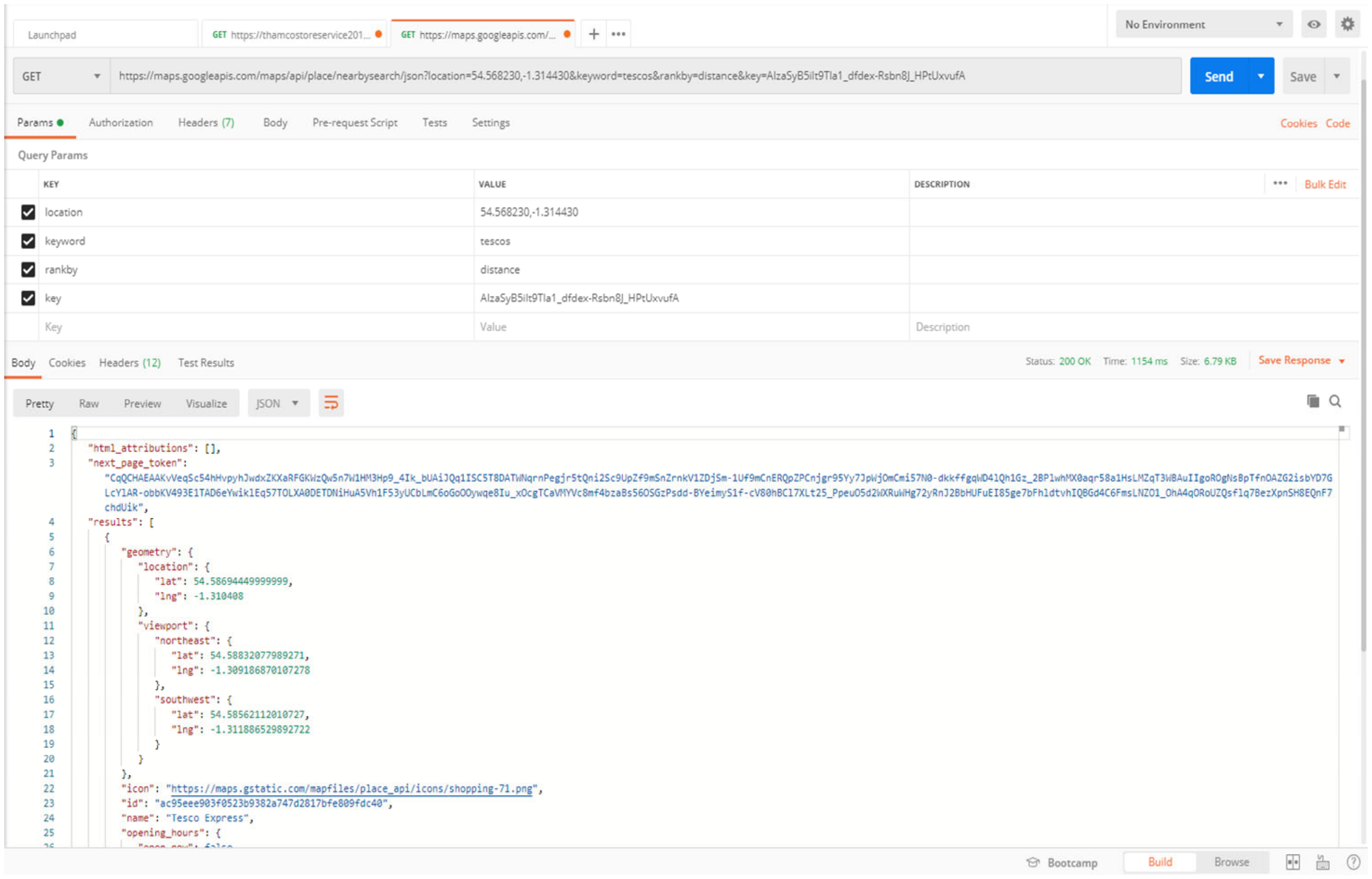
Query Params

	KEY	VALUE	DESCRIPTION
✓	q	Pepper Yellow	
✓	tbm	shop	
✓	hl	en	
✓	gl	GB	
✓	location	Stockton-on-Tees,England,United Kingdom	
✓	search_engine	google.co.uk	
✓	apikey	95395500-54b6-11ea-98bd-5f7edeea322e	
	Key	Value	Description

Status: 200 OK Time: 8.33 s Size: 118.4 KB

```
{
    "query": {
        "q": "Pepper Yellow",
        "tbm": "shop",
        "hl": "en",
        "gl": "GB",
        "location": "Stockton-on-Tees,England,United Kingdom",
        "search_engine": "google.co.uk",
        "apikey": "95395500-54b6-11ea-98bd-5f7edeea322e",
        "url": "http://www.google.co.uk/search?q=Pepper+Yellow&uule=w+CAIQICInU3RvY2t0b24tb24tVGVlcyxFbmdsYW5kLFVuaXRlZCBLaW5nZG9t&hl=en-GB&gl=GB&tbm=shop"
    },
    "no_results_auto_correct": "Sponsored ",
    "related_searches": [],
    "shopping_results": [
        {
            "title": "Pepper 'Corno di Toro Giallo' - yellow, sweet",
            "link": "/aclk?sa=l&ai=DChcSEwiL-4aVruvoAhWrFq0GHVCYAg8YABAEGgJwdg&sig=AOD64_1I_rwunjt-jLLR0MUD-Rf2TkMN4g&ctype=5&q=&ved=0ahUKEwiIooOVruvoAhUWt54KHZK6DiAQ2CkImgI&adurl=",
            "description": "£2.94.£2.94",
            "price": "£2.94",
            "source": "gardenseedsmarket.com",
            "reviews": 0,
            "thumbnail": "data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAAQABAAD/2wCEAAkGBwgHBgkIBwgKCgkLDRYPDQwMDRsUFRAWIB0iIiAdHx8kKDQsJCYxJx8fLT0tMTU3Ojo6Iys/
                RD84Oz05OjcBCgoKDQwNGg8PGjclHyU3Nzc3Nzc3Nzc3Nzc3Nzc3Nzc3Nzc3Nzc3Nzc3Nzc3Nzc3Nzc3Nzc3N//AABEIAO4BDgMBEQACEQEDEQH/xAAcAAACAwEBAQEAAAAAAAAAAAAEBQIDBgcBAAj/
```

Figure 16 Testing Google Map results API (see online version for colours)



GET https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=54.568230,-1.314430&keyword=tescos&rankby=distance&key=AlzaSyB5ilt9Tla1_dfdex-Rsbn8J_HPtUxvufA

	KEY	VALUE	DESCRIPTION
✓	location	54.568230,-1.314430	
✓	keyword	tescos	
✓	rankby	distance	
✓	key	AlzaSyB5ilt9Tla1_dfdex-Rsbn8J_HPtUxvufA	
	Key	Value	Description

Status: 200 OK Time: 1154 ms Size: 6.79 KB

```
{
  "html_attributions": [],
  "next_page_token":
    "CqQCHAEAAKvVeqSc54hHvpyhJwdxZKXaRFGKWzQw5n7W1HM3Hp9_4Ik_bUAiJQq1ISC5T8DATWNqrnPegjr5tQni2Sc9UpZf9mSnZrnkV1ZDjSm-1Uf9mCnERQpZPCnjgr95Yy7JpWjOmCmi57N0-dkkffgqWD41Qh1Gz_2BPlwhMX0aqr58a1HsLMZqT3WBAuIIgoROgNsBpTfnOAZG2isbYD7G
    LcY1AR-obbKV493E1TAD6eYwik1Eq57TOLXA0DETDNiHuA5Vh1F53yUCbLmC6oGoOOywqe8Iu_xOcgTCaVMYVc8mf4bzaBs56OSGzPsdd-BYeimyS1f-cV80hBCl7XLt25_PpeuO5d2WXRuWHg72yRnJ2BbHUFuEI85ge7bFhldtvhIQBGd4C6FmsLNZO1_OhA4qORoUZQsflq7BezXpnSH8EQnF7
    chdUik",
  "results": [
    {
      "geometry": {
        "location": {
          "lat": 54.58694449999999,
          "lng": -1.310408
        },
        "viewport": {
          "northeast": {
            "lat": 54.58832077989271,
            "lng": -1.309186870107278
          },
          "southwest": {
            "lat": 54.58562112010727,
            "lng": -1.311886529892722
          }
        }
      },
      "icon": "https://maps.gstatic.com/mapfiles/place_api/icons/shopping-71.png",
      "id": "ac95eee903f0523b9382a747d2817bfe809fdc40",
      "name": "Tesco Express",
      "opening_hours": {
```

Figures 15 and 16 show the testing of the external APIs used in this project. The test was mainly done to ensure the API endpoints are operational and many mock variables are passed through to make sure they gave the desired output. Both requirements were successfully tested, allowing us to believe the app works as expected and no significant improvements are needed at this point.

7.1 Testing new app functionality

Initially, two testing devices were considered, one of which was an Android phone emulator and the other being a real Android device Samsung Galaxy A50, where our preference got a real device given that as the emulator performance was found to be considerably lower.

When a new piece of functionality was added to the app, it was compiled and then loaded onto the testing device. The new feature was then tested by using the app and checking if it functions as expected. If not, any error logs were studied, and breakpoints were set at points in the newly added code to observe further what was happening, such as the state of variables. This testing method proved to be very effective at identifying countless problems by tracking the flow of the app.

Global variables were used to pass information between fragments in the app. A problem arose since all app fragments were loaded on start-up; therefore, some pages were trying to access global variables that have not yet been defined. This problem was identified with the testing methods discussed above.

7.2 Testing machine learning model

In order to test the machine learning model's overall performance each time the model was run, several graphs were generated, giving a very effective indicator of the model's progress. This method of testing proved to be a very helpful starting point. The model showed signs of overfitting from one of the graphs produced. Also, any anomalies within the model were clearly visualised.

Besides, the model could be manually tested by saving it after training. A single image could be passed through to the model and the class it belongs to could then be predicted. The code below shows this process of testing.

```
model = keras.models.load_model('mlmodel.h5')
image = "D:/Datasets/Dataset\Train\Oranges\oranges_1.png"
test = image.load_img(image, target_size=(150, 150))
image = np.array(test, dtype=np.float32)
image = np.reshape(test, (-1, 150, 150, 3))
pred_probability = model.predict(image)[0]
pred_class = list(pred_probability).index(max(pred_probability))
print(pred_class)
```

8 Reflective discussion

On completing this project, the base functionality to be achieved in the initial goals of the project has been successfully completed, that is, allowing a user to search for a product and provide them with information on where to buy the product. The solution proposed appears to address a very sensitive problem in terms of the choice of cheaper products, taking into account the location of a person affected by a tight budget.

However, due to the various constraints surrounding the project, there are limits to the app's performance. One major limitation is the number of items classified through image classification, which is not particularly large when items are under 50. However, the user still has access to a barcode or manual search, so this limitation does not stop the app from being useful entirely and is a feature that would be expanded further in the future. In addition to this, the range of search results is limited to Google shopping results, thus potentially missing other sources of information entirely. However, given that Google shopping seems to be one of the most complete at this point, this was the most appropriate choice, except for the combination of more than one.

Meanwhile, the machine learning model is not as accurate as initially planned, mainly due to the dataset used to train the model, which consisted of images that are not 'informative' enough to pick up on trends and patterns within the data. Additionally, the datasets used were not fully balanced, which led to the model becoming biased towards certain classes. The decision to create our dataset, therefore, was considered. However, it was not deemed feasible due to the time required to compile thousands of acceptable quality images.

9 Possible improvements or future works

One of the most obvious improvements regarding machine learning could be expanding the image classifier to incorporate *multi-label classification* would have increased the range of labels returned from each image classification. This would provide a wider variety of features retrieved from the passed image, allowing a user to find the closest possible representation of the item in question, rather than a more generic result. For example, instead of simply 'an apple', the model could return 'red Granny Smith apple'. However, this was not implemented due to a lack of suitable datasets containing relative images. This could be seen as the next step in improving the developed solution since the database obtained can be enriched by improving the possible coverage of products and their diversity as soon as appropriate datasets are available, which is also in line with the agile methodology used for this project, which supposes multiple iterations and product improvements during its life cycle.

As regards the application, the list of possible improvements to be considered is long and diverse, including:

- A page containing *settings* that could be changed to enhance or tailor the app for a specific could be added. This page remains unimplemented because there is no current need for such a page as the number of modifiable elements of the app was too low to dedicate a whole page. However, if the app was scaled up to a larger set of users, then the need for a settings page may be higher. Hypothetically, some of the settings that could be introduced could be:

  a changeable currency
  b location of the user
  c search radius
  d metric or imperial.

- *The collection of user preferences and search habits* aimed to enhance the app's experience further and tailor it uniquely for each user. Each time a user searched for an item, it would be added to a local file that would suggest/recommend similar items based on their search history or notify them of any price changes. However, the lack of data that we met at this point does not allow it to be implemented. Furthermore, the app had not yet been released to many users to collect such data, which goes to the list of future work to be considered when the app reaches a wider network of users. At that time, certain features would be in demand and identified through the feedback of the users or testers, during which our assumptions of the needs of the users or their potential interest in such features could be accepted or declined;
- Another section to be considered is a 'user dashboard' – A section on the main homepage that would have helped quickly communicate to a user about recommended items to purchase and price changes too frequently viewed items.
- *Filters on search results*.

10 Conclusions

In conclusion, the Android application presented in this article satisfies the initial goal of facilitating short-distance shopping to local stores and encouraging people to shop at the best price for the product.

Although the machine learning aspect, data size and model accuracy was not of such high quality as we have expected due to the constraints discussed, the usefulness of the application as a whole cannot be offset. In addition, identified constraints are the goals we want to solve in the future. Although the final product serves the app's basic functionality, there is plenty of room for expansion and improvement in all functions. When the app reaches a wider network of users, certain functions will be required to be updated or added, and we will develop these functions into the application in our future work. In our future work, we will construct a more varied and useful dataset and has a larger range of classes and better scalability. Additionally, the new functions related to filters on search results, the addition of user preferences and the user dashboard will be developed.

Acknowledgements

This research is partly supported by VC Research (VCR 000112).

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. and Kudlur, M. (2016) 'Tensorflow: a system for large-scale machine learning', in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp.265–283.

Barcodelookup.com (2020) *Barcode Lookup API | Search Our UPC, EAN and ISBN Database* [online] https://www.barcodelookup.com/api (accessed 19 January 2020).

Brownlee, J. (2020) *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*, Machine Learning Mastery [online] https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning (accessed 1 February 2020).

Charania, R. and Berger, J.E. (2016) *U.S. Patent No. 9,405,774*, U.S. Patent and Trademark Office, Washington, DC.

Cho, Y.H. and Kim, J.K. (2004) 'Application of web usage mining and product taxonomy to collaborative recommendations in e-commerce', *Expert systems with Applications*, Vol. 26, No. 2, pp.233–246.

Google Cloud (2020) *Cloud Vision Documentation | Cloud Vision API | Google Cloud* [online] https://cloud.google.com/vision/docs/ (accessed 19 January 2020).

Google Developers (2020a) *Barcode API Overview | Mobile Vision | Google Developers* [online] https://developers.google.com/vision/android/barcodes-overview (accessed 19 January 2020).

Google Developers (2020b) *Barcode Detector | Google APIs for Android | Google Developers* [online] https://developers.google.com/android/reference/com/google/android/gms/vision/barcode/BarcodeDetector (accessed 4 April 2020).

Google Developers (2020c) *Place Search | Places API | Google Developers* [online] https://developers.google.com/places/web-service/search (accessed 6 March 2020).

Imagga Blog (2020a) *How Retailers Use Visual Search to Gain Competitive Advantage* [online] https://imagga.com/blog/how-retailers-use-visual-search-for-competitiveadvantage/ (accessed 19 January 2020).

Imagga Blog (2020b) *The Top 5 Uses of Image Recognition* [online] https://imagga.com/blog/the-top-5-uses-of-image-recognition/ (accessed 19 January 2020).

Kaggle.com (2020) *Fruits 360* [online] https://www.kaggle.com/moltean/fruits (accessed 18 February 2020).

Kamavisdar, P., Saluja, S. and Agrawal, S. (2013) 'A survey on image classification approaches and techniques', *International Journal of Advanced Research in Computer and Communication Engineering*, Vol. 2, No. 1, pp.1005–1009.

opensource.google (2020) *Projects* [online] https://opensource.google/projects/open-images-dataset (accessed 24 February 2020).

Perez, L. and Wang, J. (2017) *The Effectiveness of Data Augmentation in Image Classification Using Deep Learning*, arXiv preprint arXiv: 1712.04621.

Postman (2020) *Postman | The Collaboration Platform for API Development* [online] https://www.postman.com (accessed 1 February 2020).

Saigon Technology (2020) *Agile Software Development* [online] https://saigontechnology.com/assets/media/agile-development-process-in-software-outsourcing.png (accessed 9 October 2020).

Schafer, J.B., Konstan, J.A. and Riedl, J. (2001) 'E-commerce recommendation applications', *Data Mining and Knowledge Discovery*, Vol. 5, Nos. 1–2, pp.115–153.

Semantics3.com (2020) *Product Database APIs from the World's Largest UPC Database | Semantics3* [online] https://www.semantics3.com/products/product-data-apis (accessed 19 January 2020).

Sharmila, S.T. (2014) 'Efficient analysis of satellite image denoising and resolution enhancement for improving classification accuracy', *Archives of Oral Biology*, Vol. 59, No. 11, pp.1183–1191.

Sternberg, S.R., Dargel, W., Lennington, J.W. and Voiles, T. (2004) *U.S. Patent No. 6,763,148*, U.S. Patent and Trademark Office, Washington, DC.

TensorFlow (2020a) *Image Classification | TensorFlow Core* [online] https://www.tensorflow.org/tutorials/images/classification (accessed 16 January 2020).

TensorFlow (2020b) *Install TensorFlow for Java | TensorFlow* [online] https://www.tensorflow.org/install/lang_java (accessed 21 January 2020).

TensorFlow (2020c) *TensorFlow Core* [online] https://www.tensorflow.org/overview (accessed 16 January 2020).

TensorFlow (2020d) *Tensorflow Lite | ML for Mobile and Edge Devices* [online] https://www.tensorflow.org/lite (accessed 6 February 2020).

Vision.caltech.edu (2020) *Caltech101* [online] http://www.vision.caltech.edu/Image_Datasets/Caltech101/ (accessed 20 February 2020).

Zenserp.com (2020) *Playground* [online] https://zenserp.com/playground/?tbm=shop (accessed 6 January 2020).

Appendix

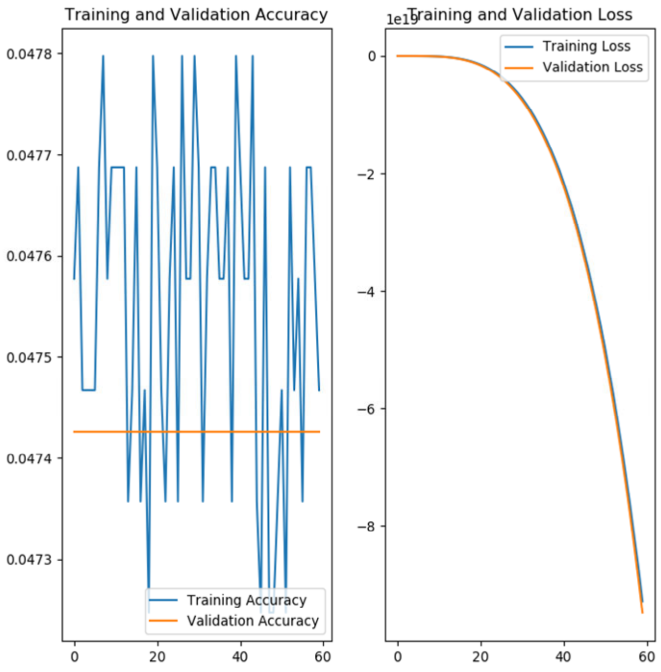
Shopping results JSON output

```
 "shopping_results": [
         {
             "title": "Sainsbury’s Apples 1kg",
             "link": "/aclk?sa=l&ai=DChcSEwjulsjH7-jnAhUS-
             VEKHRsGDY8YABAPGgJ3cw&sig=AOD64_1okoL1Rd_
             dO9mXj117rRSw5p65jQ&ctype=5&q=&ved=0ahUKEwij6cTH7-
             jnAhWE34UKHWA7ArAQ2CkI3AI&adurl=",
             "description": "£1.95.£1.95",
             "price": "£1.95",
             "source": "sainsburys.co.uk",
             "reviews": 0,
             "thumbnail": "https://encrypted-tbn3.gstatic.com/
             shopping? q=tbn:ANd9GcT2o5AzA2d4E4deIzkpaD3ug_Mu_
             RJzuw57U4OzjcXfujVpNCTZ&usqp=CAE",
             "extensions": [],
             "price_parsed": {
                 "currency": "GBP",
                 "value": 1.95
             }
         }
   ]
```

Map results JSON output (see online version for colours)

```
{
        "geometry" : {
            "location" : {
                "lat" : 54.58694449999999,
                "lng" : -1.310408
            },
            "viewport" : {
                "northeast" : {
                    "lat" : 54.58832077989271,
                    "lng" : -1.309186870107278
                },
                "southwest" : {
                    "lat" : 54.58562112010727,
                    "lng" : -1.311886529892722
                }
            }
        },
        "icon" : "https://maps.gstatic.com/mapfiles/place_api/
        icons/shopping-71.png",
        "id" : "ac95eee903f0523b9382a747d2817bfe809fdc40",
        "name" : "Tesco Express",
        "opening_hours" : {
            "open_now" : true
        },
        "photos" : [
            {
                "height" : 1192,
                "html_attributions" : [
                    "\u003ca href=\"https://maps.google.com/maps/
                    contrib/105308839422118761639\"\u003eNorton
                    Express\u003c/a\u003e"
                ],
                "photo_reference" : "CmRaAAAAEFmWEcW69tej2vFvshnFDx
                iNdzceVfazw8peZ0i2hZfOb3lxhWSC7fqjkT2TvjbwPruvp-
                xwEc7NR3oaak9RN6VUzD-WZVPSeim5yBqwKYuf_2KKB_
                8mzos7nolwjHvuEhBafnxvhtwNuRaeRydgMx6oGhTeOdCjoUbj
                JEUovxh4E_vjIj4OpQ",
                "width" : 2118
            }
        ],
        "place_id" : "ChIJfVEVcZGSfkgReSgfWwWBiyI",
        "plus_code" : {
           "compound_code" : "HMPQ+QR Stockton-on-Tees",
           "global_code"" : "9C6WHMPQ+QR"
        },
        "price_level" : 1,
        "rating" : 4.6,
        "reference" : "ChIJfVEVcZGSfkgReSgfWwWBiyI",
        "scope" : "GOOGLE",
        "types" : [
           "supermarket",
           "convenience_store",
           "grocery_or_supermarket",
           "food",
           "point_of_interest",
           "store",
           "establishment"
        ],
        "user_ratings_total" : 21,
        "vicinity" : "23-27 High St, Norton, Stockton-on-Tees"
}
```

Figure A1 Training results (see online version for colours)


Training and Validation Accuracy
Training Accuracy
Validation Accuracy
1e19
Training and Validation Loss
Training Loss
Validation Loss