IdentiFlora

Houseplant Identification and Care Application

Shape

Description automatically generated with medium confidence

Icon

Description automatically generated

Joseph Jarvis

18018718

UXCFXK-30-3

Digital Systems Project

Text

Description automatically generated

Abstraction:

Table Of Contents

1 Introduction

1.1 Project Summary

The point of this project is to build an AI system that can identify a large variety of the most common species of indoor/ houseplants. This system will then provide information on how to care for the specific species of plant. This system must be lightweight and easily accessible to the average user, so must take the form of a mobile application, that when fed an image of a houseplant will attempt to identify it, returning what the AI system thinks the houseplant is, along with any corresponding information that is relevant to the effective care of that identified houseplant.

1.2 The Real-World Problem

Humans have been using indoor plants for decoration for a large proportion of our history, as stated by Bringslimark, Hartig, and Patil, “Written evidence shows that the Egyptians brought plants indoors in the 3rd century BC, and the ruins of Pompeii revealed that interior plants were used there more than 2000 years ago” (Bringslimark, Hartig, and Patil, 2009). With this being the case combined with the information that there are a large variety of different plant species with “350,000 accepted species, of which 325,000 are flowering plants” (Royal Botanic Gardens, 2020) and the ever-increasing demand for houseplants, ensuring the proper care is vital to allow not only for these plants to thrive but to allow for the people who own these plants to continue enjoying them for years to come. To ensure their proper care these plants must first be identified, due to the large variety of species, this will be difficult to achieve for non-experts in the field, so a way of simulating expert knowledge in an easily accessible format is needed.

1.3 Aims & Objectives

* To create an artificial intelligence system that can identify a large variety of different houseplants
* Create a database of plant species containing plant care information
* Create a mobile user interface that will allow the user to upload an image of their plant for the AI system to identify, displaying relevant care information for the identified species of plant
* Identify the most effective artificial intelligence method for plant identification
* Identify the most effective ways to optimise an artificial intelligence for use on a mobile device
* Identify how to effectively integrate an artificial intelligence into a mobile/IoT platform

2 Research and Literature Review

2.1 Plant Identification

As concluded by Cham et al. (2015) plants can be identified through their flowers, fruits, stem, and bark, however plants can be most effectively identified through their leaves as a leaf’s features are more universal and persistent over each example of a given plant species. The primary features of the leaf that are useful for identification consist of leaf shape and how the leaves are structured together on the plant, known as the plants canopy structure, as stated by Jones et al. the “canopy structure and leaf shape have been key features for plant species identification” (Jones et al., 2006). Furthermore, it has been concluded by Awang et al. (2013) that along with shape, the colour and texture features of a plants leaf were also effective indicators on what type of species the plant is.

2.2 Using AI to Identify Plants

As concluded by the above research, any AI system implemented must be able to effectively identify a plant based on leaves and overall canopy structure.

Image recognition and classification with AI can be done in a few ways, as concluded by Al-Murad, Islam, and Raj (2017) in their analysis of current AI methods for image recognition and classification. One stated method is to use an artificial neural network (ANN), another would be to use a convolutional neural network (CNN). Finally, Al-Murad, Islam, and Raj (2017) concluded that a support vector machine (SVM) is also a suitable AI method for image recognition and classification.

To move forward, each method must be evaluated to conclude which would be most suitable for plant recognition through the evaluation of previous work done in the area.

2.2.1 Evaluation of ANNs

As concluded by Aakif and Khan (2015) ANNs can be used to great effect, with them achieving results of over 96% accuracy with their implementation of an ANN using backpropagation, stating that “we have tested it on three different sets and achieved accuracy greater than 96%.” (Aakif and Khan, 2015). Furthermore, Macario, Oliveira and Pacifico (2018) also achieved a similar result using a multi-layered perceptron, a type of feedforward ANN, also using back propagation, with their implementation achieving a similar accuracy in real world tests of the MLP “the algorithm was able to achieve an average accuracy of 97.16%”( Macario, Oliveira, and Pacifico, 2018).

However as stated by Choo, Huang, Liu and Wang (2017) as well as Lu and Wang (2005) ANNs have two main disadvantages, the first drawback being that “ANN-based classifications are slow as these are black box models with a gradient descend optimization and too many parameters.” (Choo et al, 2017) and the second being that ANNs have a prevalent issue with overfitting, more so on average than other methods, this results in ANNs if not carefully implemented becoming far less effective when handling real world data outside their initial training and validation datasets.

2.2.2 Evaluation of CNNs

As concluded by Arfin, Hossain, Islam, and Rabby (2019) CNNs can be used to great effect, with their implementation of a CNN with the addition of ADAM optimization, which, as concluded by Zhang (2018), is an adaptive optimisation algorithm which adaptively adjusts the learning rate of a deep neural network (like a CNN) to determine and set the most optimal learning rate for each parameter of the deep neural network (DNN) , with them stating that “The model ran for 50 epoch resulted training accuracy 96.54% and validation accuracy 95.86%” (Afrin et al, 2019), with similar high results being achieved by Aptoula, Ghazi, and Yanikoglu (2017) who conducted comparative research, where they used a CNN with 3 different deep learning architectures, these being GoogLeNet, AlexNet, and VGGNet, in which their best case achieved an “overall accuracy of 80% on the validation set” (Aptoula, Ghazi, and Yanikoglu , 2017).

In addition to this Gajjar et al (2021), also achieved a high accuracy when using a CNN to identify different plants to determine not only their identity but also to conclude whether the plant was healthy or diseased, and if so what plant disease that might be, “that the proposed CNN architecture performs well in classification of diseases from leaves, giving an accuracy around 96.88%” (Gajjar et al, 2021).

An interesting point to note is the fact that there are other ways to identify plant leaves other than through the shape and colour of leaves, which can be effectively picked out by a CNN, as noted by Chan, Lee, Remagnino and Wilkin (2015) who implemented a CNN to identify plants not only based on leaf shape but also based on the venation structure, referring to the vein structure inside the leaf, of the leaf itself, achieving a high accuracy as a result . “Moreover, we demonstrated that venation structure is an important feature to identify different plant species with performance of 99.5%, outperforming conventional solutions.” (Chan, Lee, Remagnino and Wilkin, 2015). This is an important thing to note as not only does this provide a new area of exploration for plant identification, but also provides a potential method to achieve high levels of accuracy with a CNN, as with this method, Chan, Lee, Remagnino and Wilkin CNN has outperformed every AI system discussed previously.

However, it has been noted that using a full CNN for both feature extraction and identification might be unnecessary and somewhat excessive as concluded by Chao, Li, and Nie (2020) ,who proposed a potential implementation where a shallow CNN consisting of four convolutional layers and two pooling layers, for feature extraction and then utilised a SMV to achieve effective identification, they proposed this due to one of the primary weakness of CNNs, which is the fact they require a large amount of parameters and layers to function effectively , as they state here ” The popular deep learning models require lots of parameters and layers to enhance their learning ability, consuming amount of computing resources.” (Chao, Li, and Nie, 2020). This means any way of effectively scaling a CNN down whilst still gaining the benefits of using it would be beneficial, as shown by their shallow implementation, combined with a lighter weight SVM to do the plant identification after the CNN handles the feature extraction.

2.2.3 Evaluation of SVMs

As concluded by the comparative experiments analysis undertaken by Balasaravanan, Priya and Thanamani (2012) comparing the effectiveness of k-NN and SVM for image classification, SVM managed to achieve a high level of accuracy when it came to classifying different plants, with them stating that they achieved a very high accuracy when classifying plants leaves, “The accuracy obtained by SVM in flavia dataset is 94.5%” (Balasaravanan, Priya and Thanamani ,2012), this result was improved upon with their real world data set “In case of real dataset, the accuracy of k-NN is 81.3% and the accuracy of proposed SVM classification approach is 96.8%” (Balasaravanan, Priya and Thanamani, 2012). This high level of accuracy was also concluded by Arora and Kour (2019) who using a SMV that was optimised with particle swarm optimisation, managed to achieve an average result of “classification accuracy = 95.23” (Arora and Kour, 2019) in comparison to other algorithms that were implemented including an ANN optimised with a Genetic algorithm which only achieved an average accuracy of “85.42”(Arora and Kour, 2019).

However, this does come with a few disadvantages, as concluded by Patle and Prajapati (2010) the effectiveness of a SVM is highly dependent on the kernel used, “The choice of kernel is an important issue in the SVM algorithm, and the performance of SVM largely depends on the kernel. As per our knowledge, no general rule is available as to which kernel should be used.” (Patle and Prajapati, 2010). This is an issue as there is no set method of determining the most effective kernel for a given problem without testing each potential kernel, making the process of implementing a SVM time consuming, most of which will be wasted on testing kernels that are potentially worse. Furthermore, SVM does not scale well with larger data sets, as concluded by Akata, Harchaoui, Perronnin, and Schmid (2015) who stated that “one of the limitations of nonlinear SVM classifiers is that they do not scale well with the number of training samples” (Akata, Harchaoui, Perronnin, and Schmid, 2014). Furthermore, SVMs have an issue with the lack of overall transparency in its results, as stated by Abdullah et al (2014), who stated that “SVMs have also some disadvantages. A common one is the lack of transparency in results.”(Abdullah et al, 2014) making calibrating and fine tuning a SVM difficult.

2.2.4 Evaluation of AI Methods Conclusion

As established by the above evaluation, it can be concluded that for plant classification a CNN is the most suited, with the ability to achieve the highest level of accuracy based on the research conducted, in comparison to a standard ANN and an SVM. Furthermore, the benefit of the CNN being able to do both feature extraction and image classification would allow for a quicker, more concise implementation in comparison to if the other methods were used which would require a separate algorithm to do the feature extraction.

2.3 Optimization Methods for a CNN

To reduce the error in the CNN an optimisation algorithm is needed. It was determined by Rao and Vani (2019) in their study where they implemented and compared 7 identical CNNs with different optimisation algorithms, trained on a dataset consisting of different types of Indian Pines, in order to determine the most effective, these algorithms include, “Stochastic Gradient Descent (SGD), RMSProp, Adam, Adamax, Adagrad, Adadelta, and Nadam” (Rao and Vani, 2019). From their experiments it was concluded that for a CNN, the Adamax optimisation algorithm was the most effective, outperforming all other implemented optimisation algorithms, with them stating that the “Adamax optimizer has outperformed the remaining with an accuracy of 99.58%. (Rao and Vani, 2019). However, in another comparative study of the best optimisation algorithm for a CNN, in this case for the image processing of brain tumours, undertaken by Arshid et al (2020) it was concluded that ADAM was the more effective optimisation algorithm, with it achieving an accuracy rate of “0.99” (Arshid et al, 2020) after 300 epochs outperforming Adamax that only achieved “0.96” (Arshid et al, 2020).

From this it can be concluded that a comparative implementation should be made, testing both the Adam and Adamax optimisation algorithm to determine which will be most suitable for the final version of the house plant CNN.

2.4 Integrating Neural Networks with IoT/Mobile Devices

Due to how the system is required to function and to ensure the user can quickly and easily identify a given plant, having the CNN effectively work on small scale, less powerful devices, such as a mobile is a must. An effective tool for this must be chosen. As concluded by Alsing (2018) in their thesis, one potential method of integrating a CNN such as this is through the use of TensorFlow, a software library developed by Google to give the user extensive access to machine learning algorithms, and the use of its lighter weight version, TensorFlow Lite, as stated by Alsing “TFL is the evolution of TFM, which already supports deployment on mobile and embedded devices” (Alsing, 2018) making it ideal for the mobile application integration of the required CNN model. This was also concluded by Karthikeyan (2018), in their book, where they stated that “TensorFlow Lite is a lightweight, energy-and memory- efficient framework that will run on embedded smaller-form factor devices.” (Karthikeyan, 2018) again making it an ideal framework tool for use of CNN integration on mobile phones.

Furthermore, as concluded by Okamoto, Tanno, and Yanai (2016) in their paper where they examined different CNN architectures to determine which is most suitable for mobile implementation, for android devices a CNN consisting of a NEON SIMD instruction set was the most effective, having the quickest response time by far as well as concluding that for iOS devices the CNN architecture consisting of a BLAS library was the most effective, “As results, it has been revealed that BLAS is better for iOS, while NEON is better for Android” (Okamoto, Tanno, and Yanai, 2016). Furthermore, they also conclude that reducing the actual size of input image was an effective way of speeding up processing time, up to a point, as they stated “Until 180x180, reducing the size of an input image when using the CNN is effective and easy way to adjust the trade-off between accuracy and processing time”(Okamoto, Tanno, and Yanai, 2016) an important thing to note to ensure quick and effective response times.

2. 5 Ethical Considerations

This project never stores or uses any data about the user and simply identifies the given image of a house plant to the best of its ability. This results in this project from an ethical point of view being very low risk.

3 Requirement Analysis

3.1 Introduction

The following consists of the requirement analysis of the system that will be implemented, this includes user stories as well as the functional and non-functional requirements alongside acceptance testing.

3.1.1 Priority System

Throughout this section the MoSCoW priority system will be used to identify the necessity of each requirement. As stated by Garleanu, Mărzan, Paul, Spiru, and Velciu, “MoSCoW stands for must, should, could and have requirements to accomplish business needs.” (Garleanu et al, 2019) and is primarily used to conclude between designers and developers what should be developed, which must be prioritise as well as what features are not viable, either due to time restraints, lack of resources, or features that are simply not needed for a functional and effective final product. This method is ideal for a project such as this, as with a restrictive time limit and the necessity to balance work from other modules alongside this project, compromises on scope and the overall functionality of the project must be made.

3.1.2 Justification of Requirements

Throughout this section any design choices made, including system functionality, requirements and their priority will be justified by stating the source of these requirements.

* 1. User Stories

User stories for the System

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | As a… | I want… | So that… | MoSCoW |
| US.AI1 | User | A method of identifying what my house plant is | I can identify unknown plants that I have in my possession | M |
| US.AI2 | User | To know the name of the plant that I am having identified | I know what the unknown plant is called. | M |
| US.AI3 | User | To be able to identify lots of different house plants | I can identify many of different houseplants | M |
| US.AI4 | User | The plant identification to be accurate. | I can ensure that the plant in not misidentified | M |
| US.AI5 | User | This plant identification to be done quickly. | I can easily determine what my unknown plant is in a timely manner. | M |
| US.AI6 | User | To have my plant identified using a picture. | I can easily give the information needed for my plant to be identified | M |
| US.AI7 | User | To be able to have my plants identified with a portable device | I can identify my plant on an easily accessible device as well as on the go | M |
| US.AI8 | User | To be able to have the number of different houseplants identifiable easily expanded upon | Over time as more plants become available I can continue to use the system. | S |
| US.AI9 | User | The identification method to be easily explainable | I can understand how my plant is being identified | C |
| US.UI1 | User | To be able to upload images of my plant to an application for identification | I can have means of quick and portable plant identification | M |
| US.UI2 | User | To be able to upload pictures of my unknown plant from files stored on my device | I can identify plants I’ve found online or overwise do not have direct access to | S |
| US.UI3 | User | To be able to take pictures of my unknown plant, and upload them, through the system | I can identify any plant I have direct access to | M |
| US.UI4 | User | To have the uploaded image of my unknown plant displayed in the system | I can see my plant to allow for me to associate more effectively what it is with an example of the plant | M |
| US.UI5 | User | To have information displayed about how to care for my identified plant | I can effectively take care of my newly identified plant | M |
| US.UI6 | User | Plant information to be displayed in the form of a table | I can see information about my plant in a concise format | M |
| US.UI7 | User | To be able to upload multiple consecutive images of different plants | I can identify multiple different plants without restarting the system | M |
| US.UI8 | User | To be able to queue up photos of different plants to have them identified | I can identify multiple different plants all at the same time. | C |

3.3 Functional Requirements

The following are a sample of the functional requirements, to view the others, see appendix …

3.3.1 Functional Requirements of the Neural Network

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirements | Priority (MoSCow) | Source/ Justification |
| AI1 | The AI system will need to identify images of house plants | M | Core functionality |
| AI2 | The AI system will need to support different species of house plant | M | Core functionality |
| AI3 | The AI system must be able to be integrated into a user interface | M | Core functionality |
| AI4 | The AI system must return the name of the houseplant identified | M | Core functionality |
| AI5 | The AI system must be able to handle invalid images | M | Core functionality |
| AI6 | The AI system must be able to handle poor quality images of houseplants | S | Core functionality |
| AI7 | The AI system implemented must consist of a CNN | M | Literature review: Arfin et al (2019), Aptoula, Ghazi, and Yanikoglu (2017), Gajjar et al (2021), and Chan et al (2015) |
| AI8 | Two CNNs must be created, one using ADAM optimisation, the other using Adamax optimisation, to determine the best method | M | Literature review: Rao and Vani (2019), and Arshid et al (2020) |
| AI9 | The CNN will be able to support full colour images | S | Literature review: Awang *et al*. (2013) |
| AI10 | The CNN will be able to be portable and lightweight enough to function on a mobile platform | M | Core functionality |

3.3.2 Functional Requirements of the User Interface

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirements | Priority (MoSCow) | Source/ Justification |
| UI1 | The system will require a front-end user interface | M | Core functionality |
| UI2 | The user interface will need to interface with the neural network | M | Core functionality |
| UI3 | The user interface will need to interface with a data base | M | Core functionality |
| UI4 | The user interface will need to pull relevant plant information from the database, based on the results of the neural network | M | Core functionality |
| UI5 | The User interface must allow the user to upload an image to the CNN | M | Core functionality |
| UI6 | The user interface must allow the user to do repeat submissions of different images to the CNN | M | Core functionality |
| UI7 | The user interface must display the correct image of the given plant identified by the CNN | M | Core functionality |
| UI8 | The user interface must display the name of the identified plant | M | Core functionality |
| UI9 | The user interface will need to be able to clear old identification requests | M | Core functionality |
| UI10 | The user interface must scale input images down to 180x180 before passing it to the CNN | M | Literature review: Okamoto, Tanno, and Yanai (2016) |

3.3.4 Functional Requirements of the Database

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirements | Priority (MoSCoW) | Source/ Justification |
| DB1 | The database must be able to store data | M | Core functionality |
| DB2 | The database will need a table to store care information about plants | M | Core functionality |
| DB3 | The databases plant information table will need a column to store plant names | M | Core functionality |
| DB4 | The databases plant information table will need a column to store basic plant descriptions | M | Core functionality |
| DB5 | The databases plant information table will need a column to store a plants ideal light levels | M | Core functionality |
| DB6 | The databases plant information table will need a column to store the amount of water a plant needs. | M | Core functionality |
| DB7 | The databases plant information table will need a column to store the ideal temperature for a plant | M | Core functionality |
| DB8 | The databases plant information table will need a column to store the amount of potting space needed | S | Core functionality |
| DB9 | The databases plant information table will need a column to store the soil type needed | S | Core functionality |
| DB10 | The databases plant information table will need a column to store any plant nutritional requirements | S | Core functionality |

3.4 Non-Functional Requirements

The following are a sample of non-functional requirements, to view the others, see appendix …

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirements | Priority (MoSCoW) | Source/ Justification |
| NF.AI1 | The CNN must have a minimum accuracy of 95% on training data | M | Core functionality & Literature review : Arfin et al (2019), Aptoula, Ghazi, and Yanikoglu (2017), Gajjar et al (2021), and Chan et al (2015) |
| NF.AI2 | The CNN must have a minimum accuracy of 90% on validation data | M | Core functionality & Literature review : Arfin et al (2019), Aptoula, Ghazi, and Yanikoglu (2017), Gajjar et al (2021), and Chan et al (2015) |
| NF.AI3 | The CNN must have a minimum accuracy of 90% on test data | M | Core functionality & Literature review : Arfin et al (2019), Aptoula, Ghazi, and Yanikoglu (2017), Gajjar et al (2021), and Chan et al (2015) |
| NF. AI 4 | The CNN once integrated with the mobile application, must have a response time of less than 1 second to identify a plant | M | Core functionality & Literature review: Okamoto, Tanno, and Yanai (2016) |
| NF. AI 5 | The CNN must be able to identify at least 10 different species of common house plants | M | Core functionality |
| NF. AI 6 | The CNN must be implemented through TensorFlow Lite, to allow for effective mobile integration | M | Core functionality & Literature review: Alsing (2018) & Karthikeyan (2018) |
| NF.UI1 | All interactions with the user interface must respond within 1 second of interaction | M | Core functionality |
| NF.UI2 | All information displayed through the app must be in English | M | Core functionality |
| NF.UI3 | All information displayed through the user interface must be clear and readable | M | Core functionality |
| NF.UI4 | All interaction with the user interface must be clear and understandable | M | Core functionality |
| NF.DB1 | Once the plant is identified, the database must respond and return the relevant care information within 2 seconds | M | Core functionality |
| NF.DB2 | Any data modification made to the database must be updated to all users within 3 seconds of the update occurring | M | Core functionality |
| NF.DB3 | The database must be easily maintainable, allowing for new information to be easily added, altered, or deleted | M | Core functionality |
| NF.DB4 | All information delivered from the database will consist of standard English, all spelt correctly | M | Core functionality |
| NF.DB5 | The database must be implemented with SQLite, to allow for mobile integration | M | Core functionality |
| NF.All1 | From initial submission of the plant image, the system must return both the plants identity (if it can) and the relevant information from the database within 3 seconds | M | Core functionality |
| NF.All2 | All services, including the CNN, mobile application and the database must have an uptime of over 99% | M | Core functionality |
| NF.All3 | All software must be runnable on android devices | M | Core functionality |
| NF.All4 | The application must be able to run on at least 50% of android devices currently on the market | S | Core functionality |
| NF.All5 | All software must be runnable on iOS devices | C | Core functionality |
| NF.All6 | The application must be able to run on at least 50% of iOS mobile devices on the market | C | Core functionality |

* 1. Acceptance testing

The following consist of a sample of the acceptance tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Which requirements are being tested? | How they are to be tested | Pre-requisites | Expected output |
| AT1 | AI1, AI2, AI3, AI4, AI7, AI9, AI10, UI1, UI2, UI3, UI4, UI5, UI7, UI8, UI10, DB1, DB2, DB3, DB4, DB5, DB6, DB7, DB8, DB9, DB10 | Upload an image, through the camera, of a plant for the CNN to identify | System must be running | The system will return the label of the plant that it deems the image to be, as well as display the image of the houseplant. All relevant care information for the plant that matches the label returned by the CNN will also be displayed |
| AT2 | AI6, UI6, UI9 | Upload a poor image of a houseplant after a houseplant image has already been submitted | System is running and an image has already been uploaded to the system | The system will clear the previous plant photo and information and attempt to identify the new image to the best of its ability, displaying the new plant image and its information, determined by the label. |
| AT3 | AI5 | Upload a random image of something that is not a house plant to the system. | System must be running | The CNN will attempt to label the image the best it can, returning the label with the highest probability as determined by the CNN, at which point the image and the relevant plant data for the label are displayed |
| AT4 | AI8 | Two identical CNN models will be created with two different optimisation algorithms | N/A | The two CNNs will be trained, the method with the highest accuracy and validation accuracy will be chosen and integrated into the application |

1. Methodology

A choice of methodology for a project such as this needs to take a few key criteria in to consideration. These being,

* Firstly, the project is being undertaken by one person, resulting in all methodology roles being undertaken by myself
* Secondly, the development process must be flexible to allow for both the completion of this project as well as to ensure this project does not affect the other modules that I am taking.
* Thirdly, each component of the methodology, from research to development to testing, must be able to be taken in parallel, i.e., the research for the CNN must be able to synchronise closely with the development of the CNN as well as the testing of the CNN. This is to ensure that a project of this scale can run smoothly in the limited time frame provided.
* Finally, the methodology must allow for adaptive code implementation alongside rigours documentation. This is due to the nature of this project requiring both a report and an artefact to be implemented, so a methodology that results in large amounts of redundant documentation if changes must be made is not ideal.

As stated by Butler and Vijayasarathy (2016) there are three schools of thought in software development methodology, these include the traditional linear software development approach, known as waterfall, the more modern adaptive approach, known as agile, and finally hybrid versions of these approaches.

4.1 Waterfall

Due to the nature of the project, the primary issue that waterfall has, as identified by Agrawal and Chari (2018), that the model does not support evolving and changing requirements, can be easily mitigated with effective research and planning, since the primary course of change, that being the client wanting new or different functionality of the system, is not applicable in this scenario. However, this does mean that any changes that need to be made later in the project’s life cycle due to new information or potential misunderstandings made at the beginning of the project become very difficult and time consuming to implement. Since the project does involve the exploration and implementation of new technologies, changes will need to be made throughout so the project methodology must be flexible, something that the linear nature of waterfall does not support.

However, due to this project’s significant focus on documentation, elements of waterfall would be beneficial, due to, as stated by Balaji and Murugaiyan (2012), waterfall prioritising thorough documentation after each phase, which allows for more clarity in later phases, something that would benefit the project greatly.

4.2 Agile

Agile development, as stated in the Manifesto for Agile Software Development by Beck et al (2001), primarily focuses on the idea that software developments should be focused on creating software quickly that effectively adapts to an ever-changing list of requirements, often done in an iterative process. This allows for a more adaptive and overall faster software development process that more effectively reflects what the customer/user base wants from the software, however, considering the fact, as stated by the Manifesto for Agile Software Development, it prioritises a working product over documentation, which goes against the main criteria of this project which is to produce both a working product and comprehensive and thorough documentation, a hybrid version of agile would need to be used, combining both the effective and adaptive code development of an agile approach, with the rigorous documentation provided by traditional software development methods such as waterfall to allow for this project to be completed to a high standard.

As discussed by Dingsøyr and Dyba (2008) Agile software development incorporated multiple different software development methodologies including but are not limited to, Scrum, Extreme programming (XP), Dynamic software development method (DSDM), Lean software development and Feature-driven development.

4.3 Chosen Methodology

Based on the above research a hybrid approach will be taken, consisting of scrum with a primary focus on documentation as well as the implementation of core functionality in a sequential and prototype driven fashion, with each component of both the development and the documentation being broken up into core phases, with each phase being further broken down into sprints. These phases will consist of the following,

* Research
* Requirements gathering
* System Design
* Choosing a methodology
* Dataset procurement and creation
* Implementation of the AI component of the system
* The implementation of the mobile application interface
* The implementation of the database layer of the application
* The integration of the AI and database into the mobile application
* Overall testing of the artifact and its components
* Project evaluation and improvements

4.4 Project Plan and Timeline

The following Gantt chart and hierarchical breakdown of the system were produced to show the estimated time frame and general structure that the project is intended to be done in.

A picture containing bar chart

Description automatically generated4.4.1 Gantt chart

Figure 1

As shown above, due to the workload and the component like nature of the system, some phases of the project are intended to run in parallel to each other, not only to allow for effective time management, but to also allow for effective integration and testing of each subsystem of the artifact, as the parallel development allows for greater focus to be placed on the smooth implementation and integration of each component.

Diagram

Description automatically generated4.4.2 Hierarchical Breakdown

Figure 2

As shown in the hierarchical breakdown, the project is intended to be broken down into 5 core phases, this being the research stage which contained a heavy focus on documentation, the design phase, with the core focus on the design of each component of the system, including the CNN, the mobile application interface, and the database, as well as the overall structure of the system. The software development phase, consisting of 4 core components with each of the 3 main sub systems of the artifact having their own subcomponents that must be implemented before final integration of these subsystems into the final artifact can take place. The testing phase, primarily focused on ensuring the artifact functions as intended and meets all requirements that have been set out in the research phase, if it does not, this is where the project can safely loop back to any of the previous stages, ensuring the agile methodology is adhered to and appropriate changes to the project or the artifact can be made. Finally, the project closure phase will be focused on the final project evaluation, identifying what was achieved in the project, what could be improved in the future as well as where the artifact and the project can be taken from here, as well as any final documentation improvements.

5 Design

The following section consists of the design of the artifact, this has been split it four core components, the design of the overall system, the design of the CNN, the design of the mobile application and the design of the database, with justifications for each design choice made.

5.1 System Architecture

Diagram

Description automatically generated

Figure 3

The following shows an overview of the overall system, the system is intended to be structured to ensure low coupling between components, as shown above with by the TensorFlow lite model and the database having no direct interaction between each other, instead using the mobile application as the central point of communication and interaction, resulting in less redundant interactions between modules in the system. Furthermore, the system has been designed to have a high level of cohesion, as demonstrated by the strong centralisation of data and processing onto the mobile interface. This is to ensure reduced complexity in the system, whilst also making the development of the system and debugging the system far easier as this reduces potential points of failure as well as , when combined with the low coupling, makes isolating points of failure a more streamlined task.

Diagram

Description automatically generated5.1 Design of the Neural Network - Prototype CNN

The diagram to the right shows the intended structure of the CNN, this will consist of a sequential model that will consist of 5 convolutional layers using “rulu” activation, with the initial input layer reshaping input images into an 180x180 pixel format, with each following convolutional layer doing feature extraction of the relevant image. The final layer will consist of a flattening layer that will reduce the input data from the previous neurons into a single dimension, in comparison to its previous format which was 2 dimensional, this is to allow for more effective classification based on feature extraction from the previous 5 convolutional layers, this is then to be followed by a dense layer using “rulu” activation where all previous neurons will be linked to one dense neuron, this is to be followed by a dropout layer to assist in preventing overfitting, where the model becomes to finely tuned to the datasets resulting in it being unable to be effectively used on any real world data. This will then be followed by a final dense layer using “softmax” activation which allows for more effective classification of images where there are more than two labels (i.e., when the CNN needs to identify more than two different houseplants, which is a fundamental requirement of the system).

For the choice in optimisation algorithms, the comparison between “adam” and “adamax” will take place in the development phase, using the above CNN structure to conclude which method would result in an overall higher accuracy, validation accuracy and testing accuracy, with the optimisation algorithm with the highest results being chosen.

The model is then intended to be trained over a minimum of 20 epochs, with an 80/20 split of image data, with 80% becoming training data, and the remaining 20% being split again, with 10% of the overall data becoming validation data and the remaining overall 10% becoming testing data.

Note: due to the nature of any neural networks, the exact numbers, such as epochs, and activation algorithms are subject to change to allow for effective fine tuning of the CNN to assist in creating a more effective AI system.

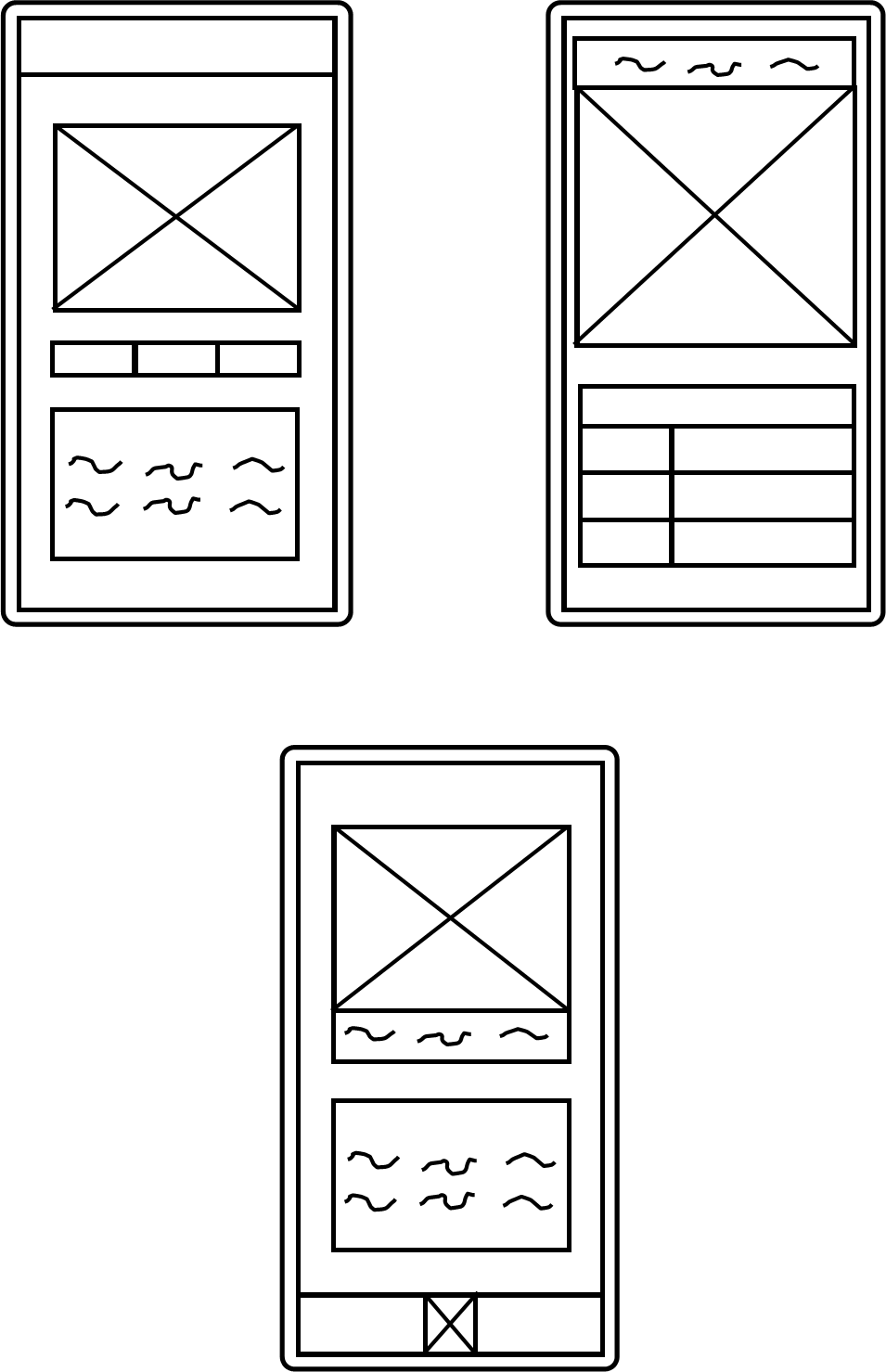
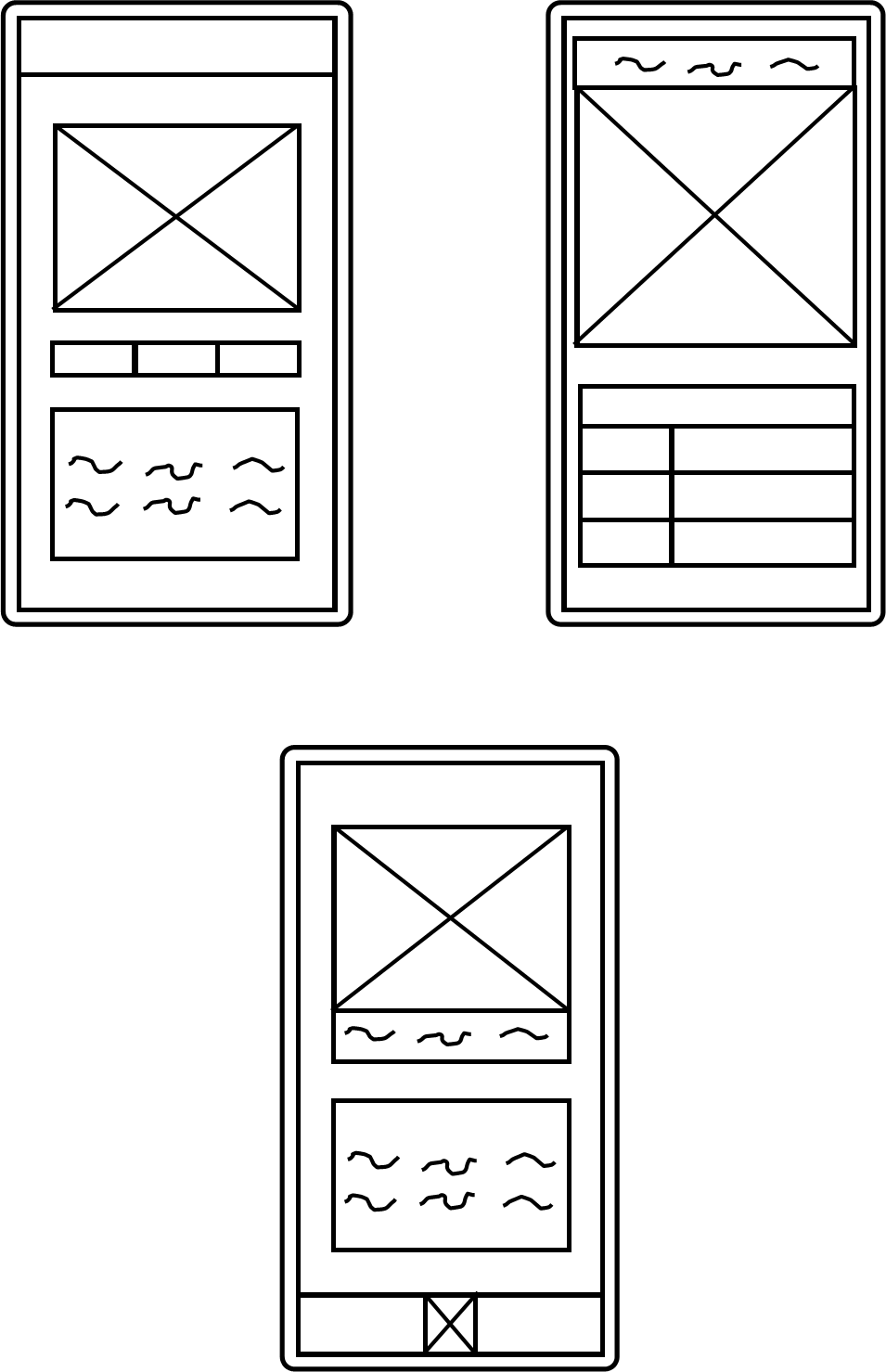
Figure 4 : Initial Protoype CNN strcture, designed ….

5.2 Design of the Mobile Interface

5.2.1 Wireframes

Initial Mock-ups

Figure 5



As shown in the above initial mock ups, the application interface will consist of 3 core components, these include the central image, which is intended to display the user’s plant that they are identifying, the table/text layout that will display the care information for the identified house plant, and finally the interactions layer, which will be how the user uploads an image of their house plant to the system. As shown in the wireframe mock ups, image up loading could be done in two possible ways, the required option (as stated by the requirements) is to be able to upload the image by letting the user take the image through the application using their phones camera, this would be done by tapping a button ( potentially consisting of the central image, as shown in the middle wireframe mock up). Furthermore, it would be beneficial to have the option to upload an image from the users locally stored files.

Final Version

A picture containing chart

Description automatically generated

Figure 6

As shown in the above wire frame, the application is intended to consist of two primary screens, a splash screen when the user launches the application and the main screen, where the user can take a photo of the plant they want identified and upload it to the system using the image button, in addition the care information of that plant then be displayed in the care information table.

5.2.2 Colour Schemes

Proposed Scheme 1:

Figure 7

Max Green

Hex Code : #4c9a2a

RGB (76, 154, 42)

Purpose : Banners and header section

Very Deep Spring Green

Hex Code : #011910

RGB (1, 25, 16)

Purpose: Background

White

Hex code: FFFFFFFF

RGB (255,255,255)

Purpose : Text

The above proposed colour scheme is designed around the idea of it being used as a possible dark mode for the application, allowing the user to customise the experience more effectively, whilst also assisting those with visual impairments to interact with the application more effectively with alternative views to suit their needs.

Proposed Scheme 2:

Figure 8

Max Green

Hex Code : #4c9a2a

RGB (76, 154, 42)

Purpose : Banners and header section

Black

Hex Code : #000000

RGB (0, 0, 0)

Purpose : Text

White

Hex code: FFFFFFFF

RGB (255,255,255)

Purpose: Background

The second proposed colour scheme would ideally be used for a light mode in the application, with bright contrasting colours, whilst keeping the plant aesthetic, makes this suitable for those with the most common forms of colour blindness assisting in making the application more accessible as well as assisting in making the user experience more pleasant and customisable, due the fact the user could choose which colour scheme they would prefer.

5.2.3 Composites/ Initial mock-up

Graphical user interface

Description automatically generatedThe following consists of the composites for the mobile application. Please note that this is not the final design of the application, these composites are to demonstrate both the layout and colour scheme of the application, meaning every other part of this composite is subject to change.

Figure 9

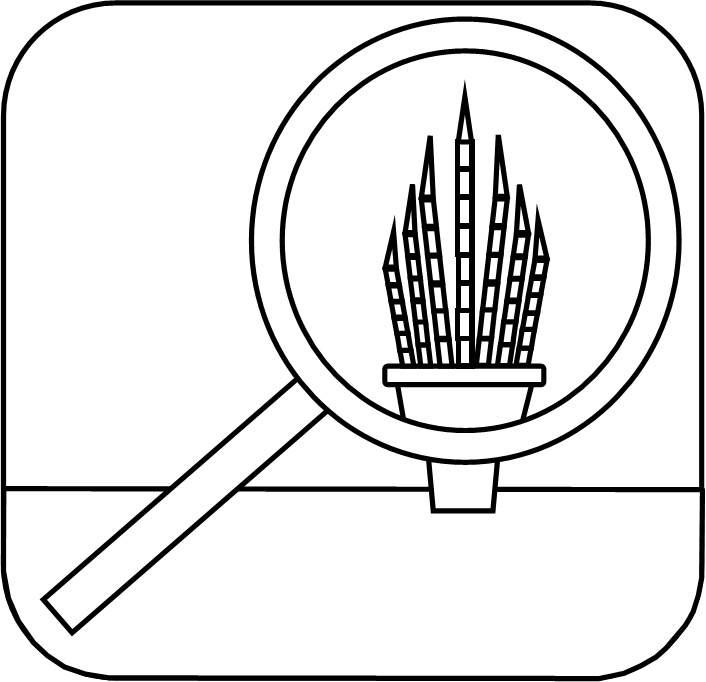
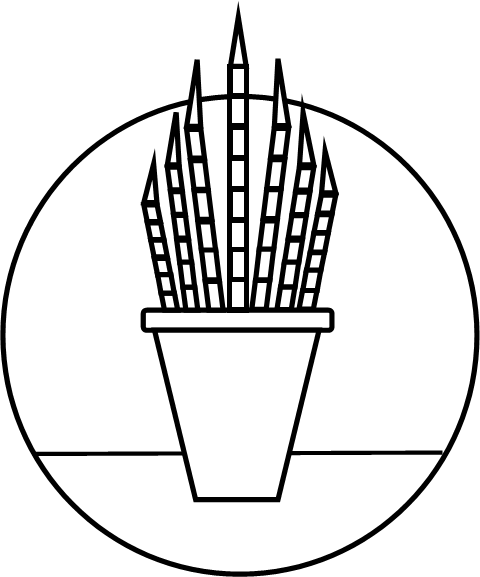
As shown above, the application is intended to work with minimal input from the user, once the photo is uploaded to the application, all the work done in the system is done by the CNN and the database, which is done automatically, resulting in only three interactions in the core interaction loop of the application, consisting of tapping the central icon, taking a photo and then confirming the submission of that photo, which can then be repeated for each plant they want identified. This minimal interaction loop is to ensure the user can obtain the information they want as quickly and with as little effort as possible, making the use of the application seamless and efficient.

5.2.4 Assets

The following section consists of design work done for each asset of the application interface.

5.2.4.1 Logo

Logo Initial digital mock-ups:



The initial logo designs consist of simplistic stylised representations of a cylinder snake plant, assisting in demonstrating the core premise of the application in a fun and stylised manner. The second design expands upon this idea with the addition of the magnifying glass, including a small zoom in effect that makes the premise of the application which is identifying plants more explicit to the user at a passing glance.

Logo Design 1:

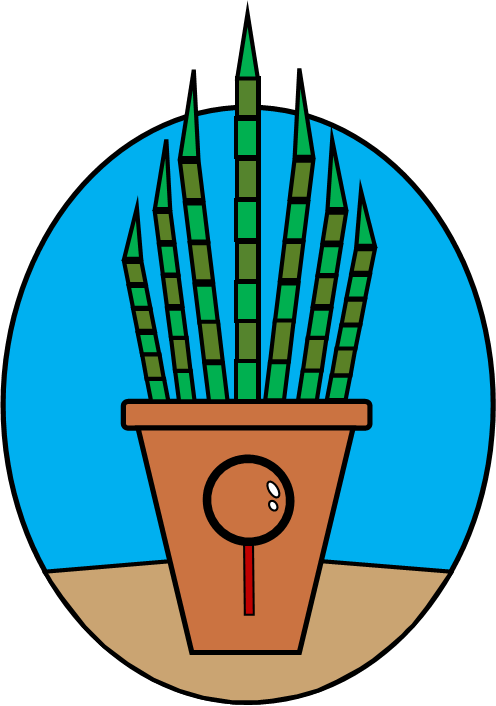


Figure 10

The diagram on the right has been further expanded upon from its original design by firstly the addition of the small magnify glass on the plant’s pot, to better signify the purpose of the application. Furthermore, the logos background has been made more oblong to guide the users focus to the core image of the cylinder snake plant. Finally, colour has been added to the logo assisting in making the logo more visually interesting as well as assisting the user in better identifying the applications purpose from a glance.

Logo Design 2:



Figure 11

The logo on the rights main improvement is the addition of more colour to the image, making it look more vibrant than its original counterpart. This logo benefits from a squarer design making it more suitable as a standard icon, due to the way both android and iOS devices display app icons as small squares with circular edges.

Note these assets are not finalised and are subject to minor changes where deemed needed.

5.3 Design of the Database

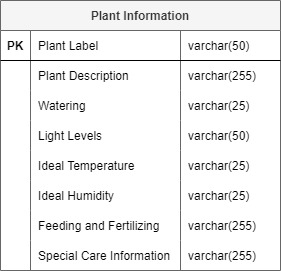


Figure 12

As shown above, due to the nature of the system, the database will consist of a small, one-table SQLite database, integrated internally into the mobile application. The rationale behind this decision is that firstly, the database is only intended to be read from and never written to by the users so a consistent internal database would be highly beneficial as storage becomes less of an issue and this allows for greater speeds of access in comparison to an eternally hosted server, as well as circumvents the need for the application to have an internet connection, allowing for plant identification no matter where the user is. Furthermore, for this database to require new entries, the CNN would need to be able to identify a new houseplant, so updating the database is only relevant when a major change to the AI model is made, which would require a full software update of the application, so the database would therefore need to be updated in tandem, resulting in remote updates becoming unnecessary.

6. Implementation & Testing

6.1 CNN Implementation

6.1.1 Overview

In this stage of the project the CNN component of the system was created, this was planned to be broken down into 6 core sprints, obtaining the datasets needed to train the CNN, Image pre-processing, the initial creation of the CNN, the evaluation of optimisation algorithms , AI fine tuning and optimisation.

6.1.2 Datasets

6.1.2.1 Pre-existing datasets

The initial intend for this project was to use pre-existing datasets to train the CNN model however, upon further research it was concluded that this was not a viable solution, as after in depth searching only one dataset consisting of 4 different species of house plant were found. This lack of pre-existing specialised datasets specifically for houseplants meant that a choice between two design decisions has to be made, either expand the scope of the project to include other plants, such as different types of wild plants and non-indoor plants, which went against the core idea and purpose of the project which was to explore AI houseplant identification, something that has not been covered in much detail in the scientific field, as the primary focus has been on edible plant identification, wild plant identification and plant disease identification, or create the datasets needed from scratch, which came with its own set of challenges, those being the fact that firstly these images would need to be gathered, a time consuming task which to do effectively required a large amount of expert knowledge, and access to a large amount of different houseplants of the same species whilst ensuring that variant of that species are also covered e.g. nerve plants nerve like structure on the leaves can come in multiple different colour, and to ensure that these plants can be identified the datasets will need to take this into account .

To ensure the purpose of the project was adhered to as well as allow for experience to be gained, it was concluded that the best cause of action was to create the remaining houseplant datasets for the project.

6.1.2.2 Creating datasets

To obtain images to create these new datasets, the primary way of doing this was to take images of houseplants from the authors own private collection of houseplants, this laid the ground work of the different species of houseplants the CNN at the end of the project would be able to identify, however to use only these plants would limit the effectiveness the AI as due plant variations in the same species, as stated above with nerve plants, using the limited yet extensive collection of houseplant the author had to hand was no ideal, as using the same specimen of plant would significantly decrease the accuracy of the AI when it would attempt to identify other plants of the same species, due the lack of variety that would be in the training data. To supplement this, excursions to local plant nurseries and garden centres were made to obtain addition samples to ensure the variety of plants of the same species in the final dataset was extensive enough to be effective. Finally, to ensure that the was enough variety of specimens for each species of houseplant was sufficiently these datasets were bolstered with plant images obtained through free, open access online repositories which allowed an individual to use their images for non-commercial purposes.

At the end of this process there were 4 pre-exist plant species data sets all containing 150 images per species of plant and 7 handmade plant species data sets containing 100 images per species of plant, resulting in the final version of the CNN for this project, however this does leave plenty of room to expand upon, however due to time constrains and lack of processing power, datasets of a greater number with more images are not viable for this project.

6.1.3 Image pre processing

Firstly, to ensure effective training, the datasets had to be balanced with each species of plant have the exact same number of images. The reason behind this is to assist in making the CNN be more effective at identify each plant to an equal amount, giving it an overall higher accuracy as, as concluded by Lalithnarayan (2020) an imbalanced dataset results in lower overall accuracy and the CNN having a natural bias toward predicting the input classed (the different time of plants) with the higher number of images. The result was all plants classes (both handmade and prebuilt) being, initially, cut down to 100 images each, this was later cut down further as discussed in the problems and challenged section.

In addition, a python program called CNNImagePreprocessor.py was created, the purpose of this program was to reformate all training images to a 180 by 180-pixel format, label them accordingly and convert the image data into a 3d NumPy array to store coloured image data and a 2d array to store the corresponding labels of the plant. This was done to allow for both image data and colour data to be considered when this data is being used to train the CNN to allow for more effective feature extraction and to achieve better results when the AI undertake real word plant identification.

A further program was intended to be created called CNNImageNoise.py that would do minor image manipulation on training images. This was designed as minor image manipulation, including tilting images at different angles and the process of salt and pepper (randomly introduction noise in the form of black and white pixel throughout an image) firstly allows for the expansion of the datasets giving the AI more data to be trained on, with very little effort needed, but also better trains the AI system to handle noisy input data, better training the AI to handle re work input that doesn’t tend to be a high quality as the image provided from the datasets. However, this program was later shelved halfway through development due to issues with hardware and training time, meaning increasing the number of images in the dataset was not viable whilst ensuring the project ran on schedule.

6.1.4 Optimisation algorithm comparison

In this sprint, a rudimentary CNN, in accordance the CNN designed in the design section, was created in python using the TensorFlow library, in the PrototypeCNN.py, to allow for the testing on the two optimisation algorithms. Each CNN was trained on the 4 pre-existing datasets, 100 images each for 20 epochs each.

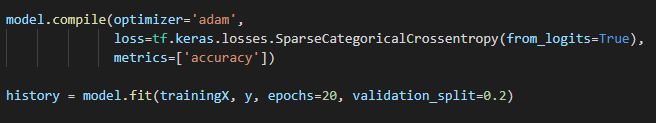
Text

Description automatically generated

Figure : Prototype CNN TensorFlow Model for optimisation algorithm testing

6.1.4.1 Adam optimisation :

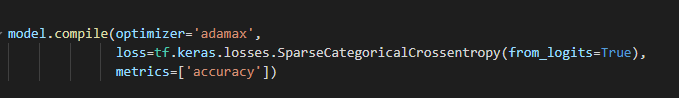
Figure 14 : Model compiler using Adam optimisation and results



After training, the prototype CNN using “adam” optimisation achieved an overall accuracy of 78.75% with a validation accuracy of 77.5%.

6.1.4.2 Adamax optimisation:

Figure 15: Model compiler using Adamax optimisation and results



After training, the prototype CNN using “adamax” optimisation achieved an overall accuracy of 80% with a validation accuracy of 82.5%.

6.1.4.3 Optimisation Algorithm Comparison Conclusion

As shown above, the “adamax” optimiser outperforms the “adam” optimiser for the specific use case, which is also reinforced by the literature, which states that whilst “adam” is an effective general optimisation algorithm, “adamax” is capable of outperforming it, especially with models that have a high levels of embedding, such as in this model, where complex and large data inputs consisting of high resolution images which also include pixel colour data, have been translated in to a low dimensional space for training purpose, in this case flattened from a 3d array to a 2d array. This means “adamax” would naturally have a slight advantage in the current use case, as shown in comparison between the two optimisation algorithms.

6.1.5 Initial CNN training

In this sprint the CNN architecture was improved upon and refined, as well as initial training took place.

The initial ideal was to train the CNN as a TensorFlow model, then design a python program that would then covert that model into a TensorFlow lite model, to achieve this a python program called tensorFlowToTensflowLiteConverter.py was created, this loaded in a .h5 TensorFlow model then convert it into a .tflite model using the TensorFlow tf.lite.TFLiteConverter.from\_keras\_model() method provided by the TensorFlow library. This method whilst successful came with some drawbacks, for example, it was found that the use of this model would occasionally cause the application to crash. The solution this problem was to, rather than build the model and then convert it, was to build the model natively for TensorFlow Lite, this was achieved through the creation of tensorFlowLiteModelGenerator.py, this program combined both the image pre-processing of the CNNImagePreprocessor.py, and the model generating capabilities of PrototypeCNN.py Into one single program, that output the trained TensorFlow Lite model alongside the relevant label of each species of plant. This was achieved using the TensorFlow and the TensorFlow Lite Model Maker which assisted in the simplification of model creation using automatic image pre-processing and the ability to use pre-existing as well as customer model architecture. From this, successful stable TensorFlow Lite models where successfully created, allowing for the CNN model development to move forward.

Attempt 1 : CNN trained on all plants.

Text

Description automatically generated

Figure 16: Results of training the first tensorflow lite implementation of the CNN

As shown above, after training the CNN for 10 epochs, trained on a 80/20 split between training data and validation and testing data, the CNN achieved an accuracy of 95.83%, however when compared to testing data, only achieved an accuracy of 80%, which whilst fair good did not meet the required minimum accuracy of 90% accuracy on test and validation date as stated in the requirements, however it is noted that it did achieve slightly above the required 95% accuracy on the training data.

Attempt 2: CNN TensorFlow lite model trained with more validation data and more epochs



Figure 17: Results of training the second tensorflow lite implementation of the CNN plant classification AI

As shown above, training the AI with 15 epochs, and adding more validation data, increasing the split slightly from 75% training data, 10% validation date and 5% validation data. The TensorFlow lite model achieved are far greater training accuracy of 98.56%, and validation accuracy of 96.36% and a testing accuracy of 92.73%, meeting all requirement for model accuracy set out in the requirement stage, as was chosen for later integration into the mobile application.

6.1.6 Final product

The final product consists of a TensorFlow lite CNN model called IdentiFloraCNN.tflite that can identify 11 different species of houseplant.

6.1.7 Problems and challenges with the development of the CNN

One problem with this section of the project was that training times for the CNN, even with a small number of epochs, were substantial. This is primarily due to the insufficient hardware available for training, as the CNN was being trained on a laptop with Intel i5-8250U CPU @ 1.60GHz, 1800 Mhz, 4 Core(s), 8 Logical Processor(s), a NVidia GTX 1050 with 2GB of Vram, and 8GB of 2400Mhz DDR4 ram. At the beginning of the project this was not the plan, as the AI component was intended to be trained on a desktop PC with an Intel i7 6700K CPU @ 4 GHz, 4 Core, 8 Threads, a NVidia GTX 1060ti with 6GB of Vram and 16GB of DDR4 3000 MHz, however due to time constraints and lack of access to the PC due to not having access to it outside of the holidays, this was not possible. There two potential solutions to this, the first one being would be to limit the amount of training data, this would speed up the training time significantly at the cost of making the AI system less effective. The second potential solution was to use Google Colab, this would allow for the training of the AI component to be done on external google servers, however, to get access to good hardware and to prevent the system from timing out (the free version having very limited hardware and timing out after 24 hours of training) it would require a monthly subscription.

It was decided that the due the finical cost, cutting down the number of images for each specimen in the system was the most viable solution giving the current the lack of hardware.

6.2 Mobile application Implementation

6.2.1 Overview

In this stage of the project the mobile application interface was created, this split up into 2 key sprints, the first being the initial creation of the base application the creation of the the layout and visual components of the application in accordance with the composites and wireframes, the second was implementing the camera functionality ensuring input images were displayed as intended.

6.2.2 Creating the mobile application

This sprint consisted of the creation of the user interface of the application, due to the extensive design work done in the design phase of the project, implementing the UI based on the composites and wireframes was a simple task. To achieve this, firstly the initial base application was created in android studio, and the primary activity screens was designed using the android pallets provided, with the central image consisting of an imagebutton pallet to allow for later integration of camera functionality, the use of the tablelayout pallet to create the care information table, as well as the initial implementation of the dark mode colour scheme, the importation of the previously designed images assess as well as the implementation of the IdentiFlora app icon, shown in Figure 18.

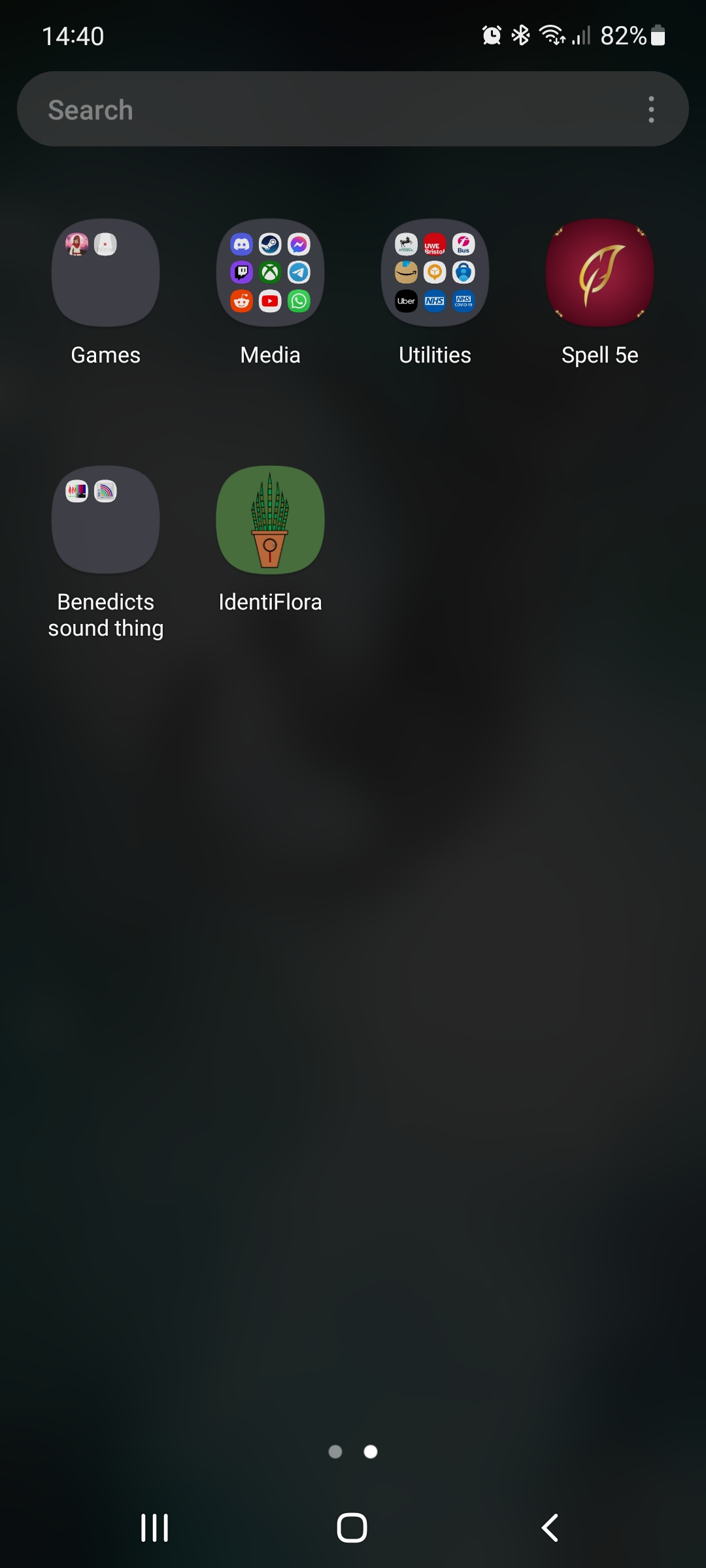


Figure 18 : IdentiFlora App Icon

The completion of this sprint concluded with successful implementation of the user interface, as shown in Figure 19.



Figure : Application interface as views in android studios .xml design editor

6.2.3 Camera utilisation and integration

This sprint involved the implementation of the camera functionality of the application. There were two possible implementation for this, one would be to call the camera activity handing activity control over to the devices camera application, then the camera application would directly pass the image data as in intent bundle from the camera and store it as a local variable in the program itself, this would be simplest option and one of the recommended ones in the android studio documentation, however this does come with the issue that is limits image files sizes to 1MB resulting in an overall poor quality looking image being displayed by the application as well as results in poor quality images then being given to the TensorFlow Lite CNN model to identify, resulting in an overall worse performance due to the noise in the image data. The second option would be to pass in a file location to the camera application when the camera activity is called, this would specify where the image should be stored locally on the device, once the user has taken their image and it has been stored, the activity control would be passed back to the application, which can then retrieve the relevant image from the local file storage. Whilst more complicated to implement, requiring android application permissions to store external files on the user device and requiring the implementation of a file provider, it allows for higher resolution images with no imposed size limit to be rendered in the application as well as provided to the TensorFlow Lite CNN model, resulting in an overall better user experience. **Figure 20** shows the code implemented to achieve this.



Figure 20 : Code used to open the default andriod camera application

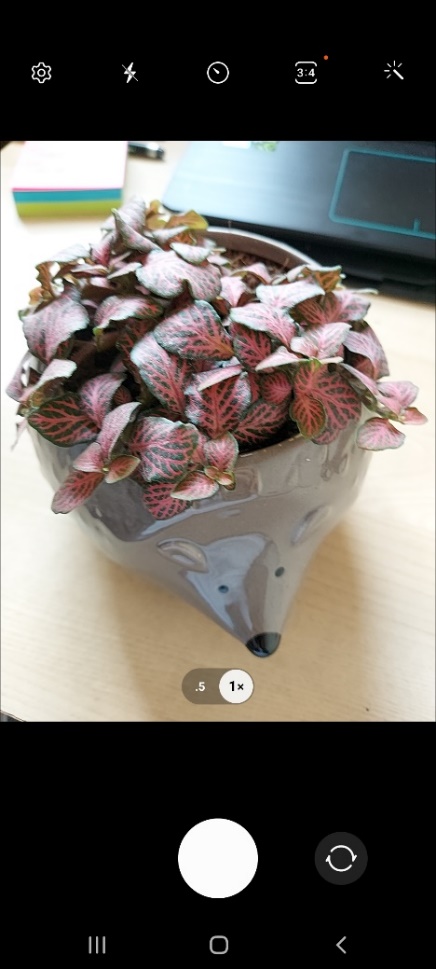


Figure 21: Camera open through the application

As shown, in **Figure 21, 23,** and **24**, this worked as intended with the successfully , however as discussed in the problems and challenges with the development of the mobile interface section, there were minor display issues with displaying the image taken by the camera correctly, however the issue was successfully determined and correct .

6.2.4 Final Product



Figure 22: Screenshot of the mobile application at the end of the mobile application development phase

As shown in **Figure 22** , the development of the application user interface has been successful, ready for the final phase of AI and database integration.

6.2.5 Problems and challenges with the development of the mobile interface

Issue 1:

Issues: Any image taken in portrait through the application is displayed incorrectly.

Evidence:

Graphical user interface, website

Description automatically generated

Figure 23: Screen shot of image taken with camera not displaying correctly

Solution: reformate all images inputted by 90 degrees

Evidence:

Graphical user interface, website

Description automatically generated

Figure 24:Screenshot of the outcome of the initial fix

The issue with this solution: This has resulted in any images taken in landscape now no longer displaying at all.

Solution: The actual issue was with how android restarts the applications when transitioning the phone from portrait to landscape and visa verse. There were two possible solutions for this, the first and most complicated solution was to save the state of the application before the transition from portrait to landscape, this would be done by taking the current state of the activity running on the android application and then saving it to be displayed once the rotation has taken place. However, this solution brought forward an issue with how the application was displayed, resulting in the landscape version of the application being difficult to read as well as unpleasant to look at, to fix this the application would need to have two separate layouts, one for portrait, and one for landscape, a time-consuming task that offered little benefit to the functionality of the application.

The second solution was to not allow the application to be able to display in any other manner other than portrait. This solution was implemented however resulting in the same issue of images not being displayed properly. This was solved by using the EXIF tags , "Exchangeable Image File Format”, of the taken image which included the angle of the phone when the image was taken, once this angle is known, the image can then be rotated accordingly and displayed in the intended manner, this was achieved with the code shown in ***Figure 24***.

Text

Description automatically generated

Figure 25: Code used to more effective rotate the image to matter the rotation of the device using EXIF image tag data

6.3 Database Implementation

6.3.1 Overview

This stage of the project was focus on the creation of the SQLite database. This was split into two core sprints, these being firstly the creation of the core database, the second sprint being the implementation of plant care data into the database.

6.3.2 Creating the SQLite database

In the sprint a simple python programme CreatePlantDatabase.py was created, this programme using the sqlite3 python library which allowed for the use of SQL commands in python, to create the initial table in the database and populate each header in each column with the appropriate fields, these being the plant name (which acts as the primary key for table), the plants description, how much water the plant requires, the required light levels, the ideal temperature, the ideal humidity, how often the plant requires to be fertilized and the final column that contains any unique care information about the plant.

6.3.3 Implementing plant data into the database

In this sprint the plant information for each species of plant the CNN could identify was added to the database. This was a simple process of adding each plants details manually to the database using the cursor.execute() command using the column structure implemented in the previous sprint.

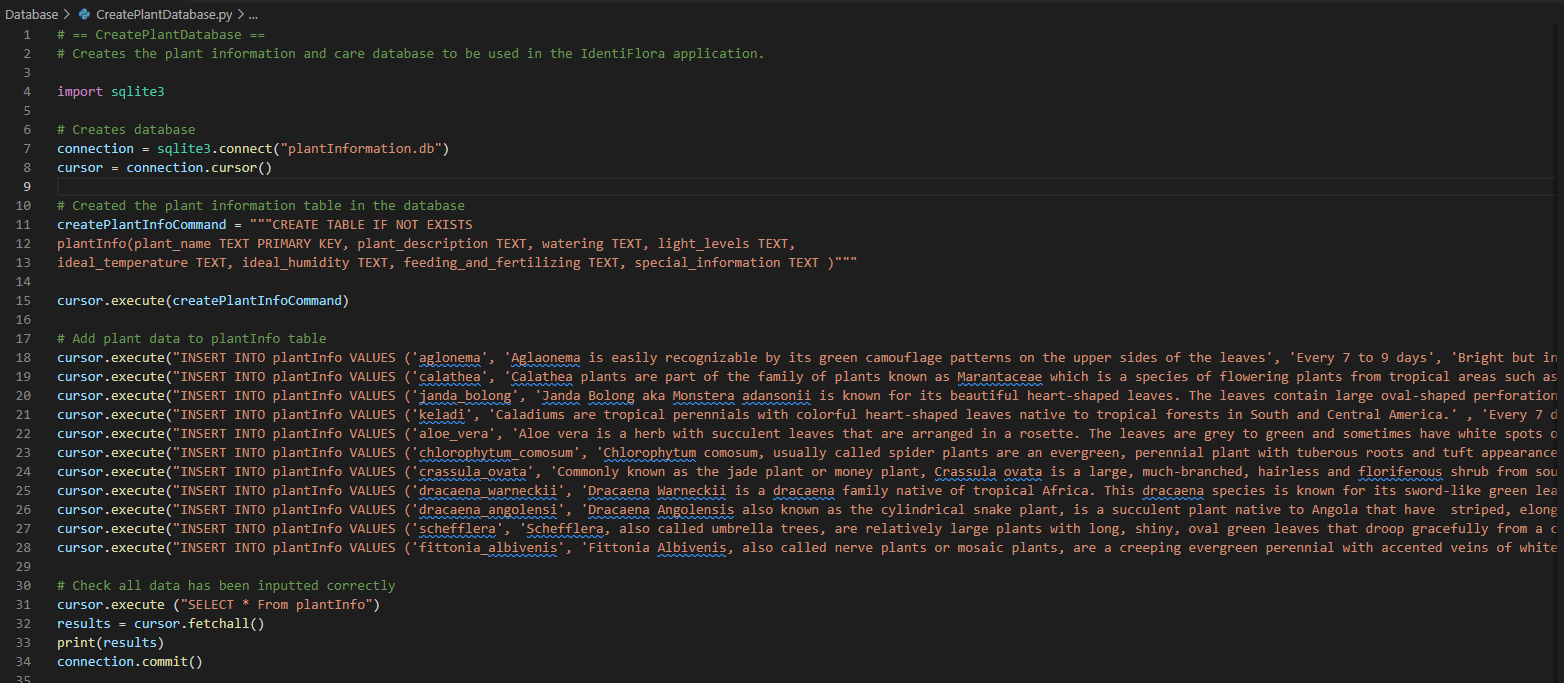


Figure : Database creation code

6.3.4 Final product

As shown in the diagram below, the database has been successfully created containing all the relevant plant data as shown below,

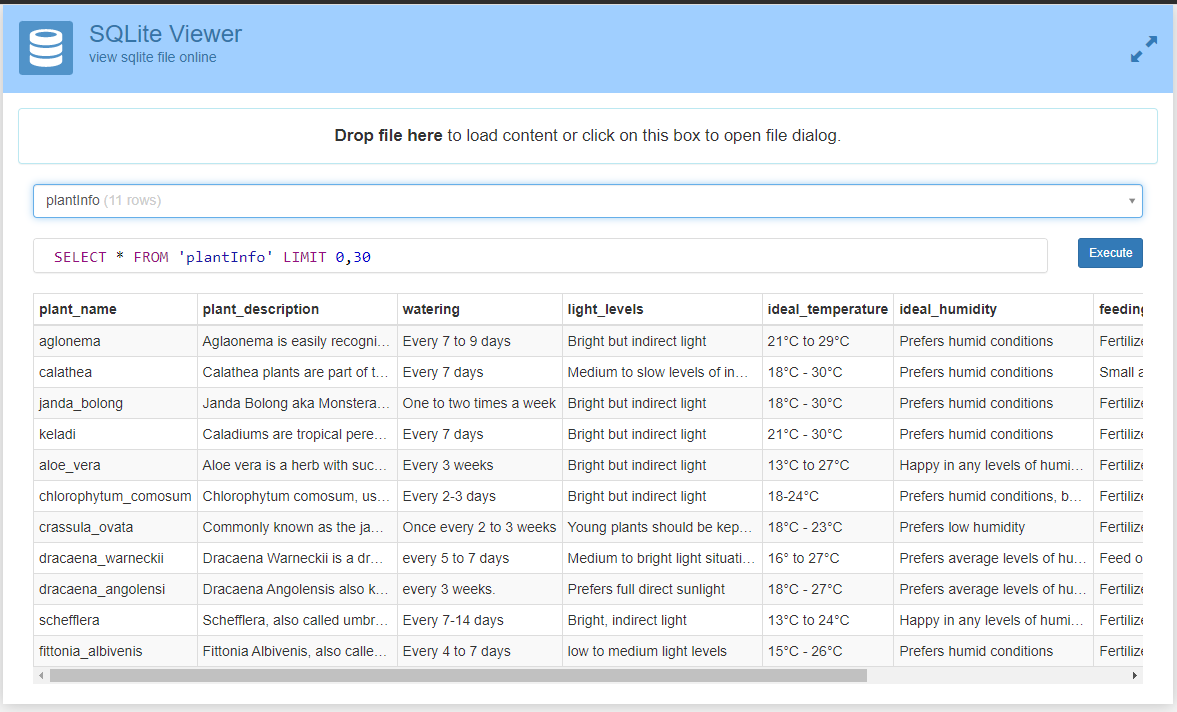


Figure : Plant information and care database visualised using SQLite Viewer

6.3.5 Problems and challenges with building the database

Due to the simplicity of the stage of the project, no issues were encountered.

6.4 Component integration

6.4.1 Overview

This stage of development was focus on integrating each core component of system into the mobile application. This was split up into 2 sprints, the first being the integration of the CNN into the application and making it provide predictions based on input images, the second was the integration of the database in the application, making the database provide the correct information based on the prediction made by the CNN.

6.4.2 Integrating the CNN into the mobile application

This sprint consisted of adding the CNN TensorFlow Lite model into the mobile application. Due to the fact that both android and TensorFlow are owned by Google, AI integration into android application is not only encourage but also is extensively supported by android studio and the Kotlin programming language make the integration of the TensorFlow Lite model into the application a simple process of importing the AI model into the application using the provided importation tool, then, as shown in **Figure 28**, the model could then be called and new instance of it could be created, from here an image can be provide to the model for identification, to achieve this the probabilistic values provided by the CNN model where used, with the species class with the highest probability of being present, as determined by the model, being selected as the plant species present in the image, the model could then be closed and the relevant plant label could be returned allows for the retrieval of care information for that plant to be done through the database as well as for the plants name to be displayed in the application. The model could then be closed to prevent any wasted resources and to ensure the stability of the application.

Text

Description automatically generated

Figure 28 : Code used to provide to obtain the probabilistic result of which plant is present in the provided image

6.4.3 Integrating the database into the mobile application

This sprint consisted of integrating the database into the mobile application, creating the corresponding database handler names PlantDatabaseHandler.kl to handle calls to and from the database. This sprint also involved taking the relevant data pulled from the data. As discussed in the problems and challenges section, getting the application to open the SQLite database provided slightly obtuse and not as simple at the Android Studio documentation would suggest, however, once the database was integrated into the application and readable, the matter of displaying the relevant information based on CNN outputs could be tackled. The following code, block of code in tandem with the database handler, shown in **Figure 29**.

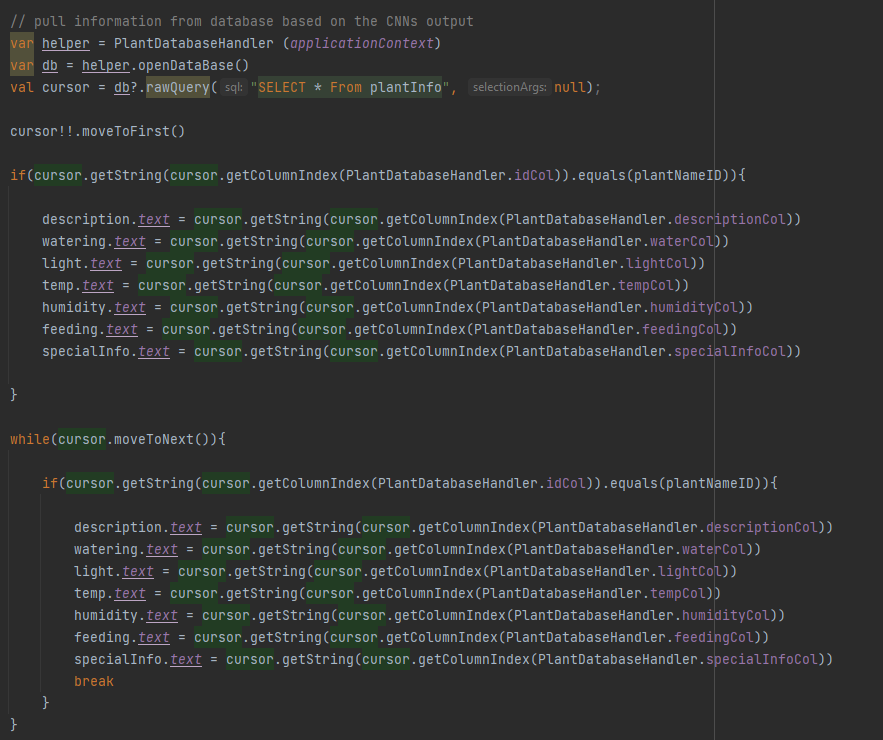
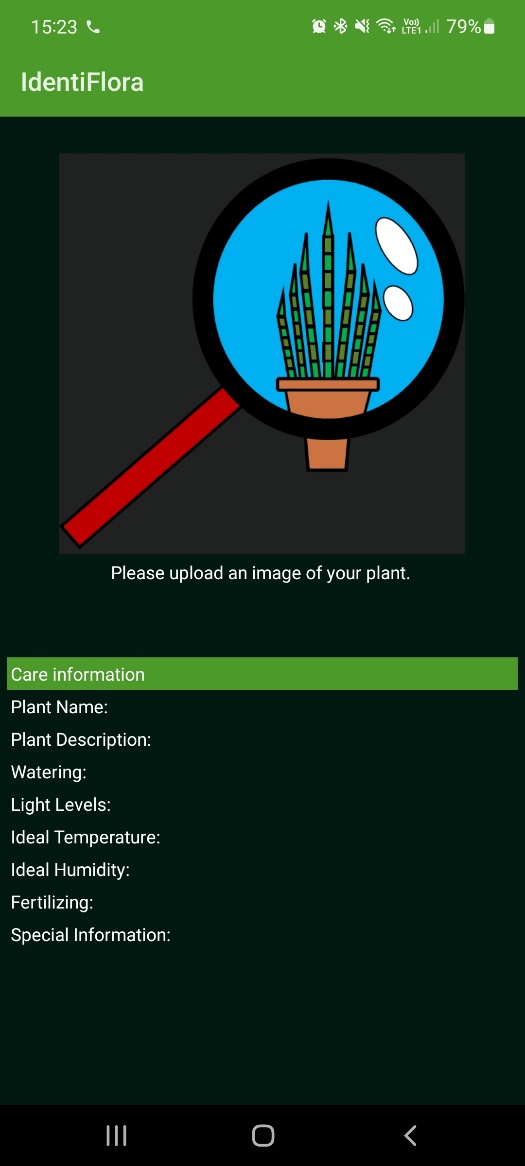
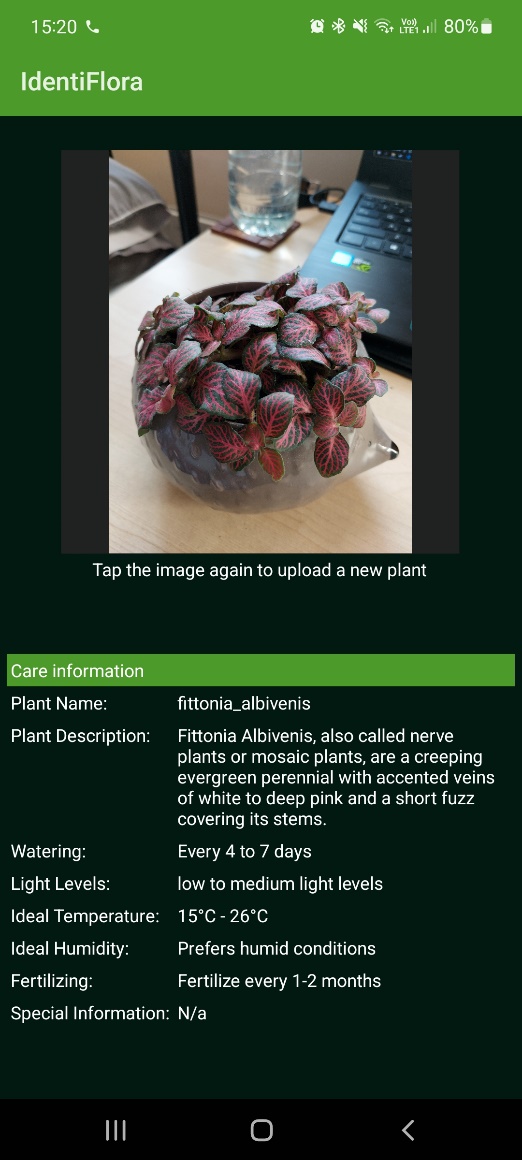


Figure : Code for displaying content in the table view from the database based on CNN output

6.4.4 Final product

The artifact at the end of this phase represents the end of the development phase of the project, with the final product, as shown in **Figure 30**, being feature complete and ready to be tested.

Figure : Screenshot of the final application, shown here identifying a nerve plant



6.3.4 Problems and challenges with component integration

Problem 1: The system cannot find the plant information table in data base

Text

Description automatically generated

Figure : Empty Database error message

Graphical user interface, application

Description automatically generated

Figure : Database made in python before porting over

A picture containing application

Description automatically generated

Figure : Exported database that was being open by the android application

As shown here, it was determined that in the process of the application opening the database from the assets folder the contents were being wiped.

The initial conclusion was the database in the assets files was broken, however it was later concluded that the database was fine and contained all the relevant data, further supporting the argument that something was going wrong when the application opens the database, with data not being translated over properly .

Solution: due to limitations with android you cannot read from a database in the assets folder, and therefore a temporary copy of that database inside the code must be made, which is done using the following code here,

Text

Description automatically generated

Figure : Code used to copy the database from the assets folder so it could then be used

6.5 System Testing

6.5.2 Acceptance Testing

|  |  |  |
| --- | --- | --- |
| Acceptance Test Being tested | Pass or fail | Comments |
| AT1 | Passed |  |
| AT2 | Passed |  |
| AT3 | Passed |  |
| AT4 | Passed |  |

6.5.2 AI Performance Testing on Real World data

In this section the performance of the TensorFlow Lite model will be tested to determine is effective on real world plants. Each class of plant that the AI should be able to identify will be tested on 10 samples of real-world plants of that type. Each test plant is unique and has not been used in training to assist in better accessing the real-world performance. Based on the requirements as well as the testing undertaken on the AI model in the development phase, the estimated minimum value should be 9 with a margin of error of ± 1.

|  |  |  |  |
| --- | --- | --- | --- |
| Plant Type being Tested | Score (out of 10) | Pass or fail | Comments |
| Aglaonema | 9/10 | Pass |  |
| Aloe vera | 7/10 | Fail | It was noted that when this plant was miss identified, it was determined to be Dracaena Angolensis (a cylinder snake plant). This is most likely due very similar leave shape (both consist of long spear like leaves) and canopy structure these plants share. |
| Calathea | 9/10 | Pass |  |
| Chlorophytum Comosum | 10/10 | Pass |  |
| Crassula Ovata | 6/10 | Fail | This was determined to be due to the dataset itself not including enough examples of adult jade plants (CrassulaOvata), as whilst undertaking further testing it was concluded that the AI model worked effectively with younger specimens with an accuracy of 9/10 on young plant specimens, but could rarely identify adult jade plants, only scoring 4/10 on a test set on only adult jade plant specimens. |
| Dracaena Angolensis | 10/10 | Pass |  |
| Dracaena Warneckii | 9/10 | Pass |  |
| Fittonia Albivenis | 10/10 | Pass |  |
| Janda Bolong | 9/10 | Pass |  |
| Keladi | 8/10 | Pass | Whilst lower than the ideal minimum, these results do fall in the margin of error. |
| Schefflera | 9/10 | Pass |  |

7. Project evaluation

8. References / Bibliography

* Aakif, A. and Khan, M.F. (2015) Automatic Classification of Plants Based on Their Leaves. Biosystems Engineering [online]. 139 (1), pp. 66-75. [Accessed 25 January 2022].
* Abdullah S. M., Halimi M., Karamizadeh S., Rajabi M. J. and Shayan J. (2014) Advantage and drawback of support vector machine functionality. In: (Unknow editor) ed. 2014 International Conference on Computer, Communications, and Control Technology (I4CT). Langkawi, Malaysia, 2-4 Sept. IEEE, pp. 63-65
* Agrawal, M. and Chari, K. (2018) Impact of Incorrect and New Requirements on Waterfall Software Project Outcomes. Empirical Software Engineering [online]. 23 (3), pp. 165-185. [Accessed 31 January 2022].
* Akata, Z., Harchaoui, Z., Perronnin, F. and Schmid, C. (2014) Good Practice in Large-scale Learning For Image Classification. Transactions on Pattern Analysis and Machine Intelligence [online]. 36 (3), pp. 507-520. [Accessed 28 January 2022].
* Al-Murad A. , Islam K. T., and Raj R. G. (2017) Performance of SVM, CNN, and ANN with BoW, HOG, and Image Pixels in Face Recognition. In : (Unknow editor) ed. 2017 2nd International Conference on Electrical Electronic Engineering (ICEEE). Rajshahi, Bangladesh, 27-29 Dec. IEEE, pp. 1-4
* Alsing, O. (2018) Mobile Object Detection using TensorFlow Lite and Transfer Learning, [online]. MSc, KTH Royal Institute of Technology. Available from: https://www.diva-portal.org/smash/get/diva2:1242627/FULLTEXT01.pdf [Accessed 30 January 2022].
* Aptoula, E., Ghazi, M.M. and Yanikoglu, B. (2017) Plant Identification Using Deep Neural Networks Via Optimization of Transfer learning Parameters. Neurocomputing [online]. 235 (1), pp. 228-235. [Accessed 23 October 2021].
* Arfin M. H. R., Hossain S. A., Islam M. M., and Rabby S. A. (2019) PataNET: A Convolutional Neural Networks to Identify Plant from Leaf Images, 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Daffodil International University, Kanpur, India, 6-8 July 2019, IEEE [online]. Available from: https://ieeexplore-ieee-org.ezproxy.uwe.ac.uk/document/8944667 [Accessed 24 October 2021].
* Arora, S. and Kour, V.P.(2019) Particle Swarm Optimization Based Support Vector Machine (P-svm) For the Segmentation and Classification of Plants. Ieee Access [online]. 7 (1), pp. 29374-29385. [Accessed 22 January 2022].
* Arshid, K., Feng, J., Jia, K., Mehmood, A., Rehman, Z.U., Yaqub, M. and Zia, M.S. (2020) State-of-the-art Cnn Optimizer For Brain Tumor Segmentation in Magnetic Resonance Images. Brain Sciences [online]. 10 (7), pp. 427-447. [Accessed 29 January 2022].‌
* Awang, K, Che Hussin, N.A., Jamil, N. and Nordin, S. (2013) Plant species identification by using Scale Invariant Feature Transform (SIFT) and Grid Based Colour Moment (GBCM). In: (Unknown), ed. 2013 IEEE Conference on Open Systems. Kuching, Malaysia, 2-4 Dec. IEEE, pp. 226-230
* Balaji, S. and Murugaiyan, M. (2012) Waterfall Vs V-model Vs Agile: A Comparative Study on SDLC. International Journal of Information Technology and Business Management [online]. 2 (1), pp. 26-30. [Accessed 31 January 2022].
* Balasaravanan T., Priya C.A, and Thanamani A.S. (2012) An efficient leaf recognition algorithm for plant classification using support vector machine . In: (Unknown Editor), ed. International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME-2012). Salem, India, 21-23 March. IEEE, pp. 428-432
* Beck, K., et al. (2001) Manifesto for Agile Software Development. Available from: https://agilemanifesto.org/ [Accessed 31 January 2022].
* Bringslimark, T., Hartig, T., and Patil, G.G., (2009) The Psychological Benefits of Indoor Plants: A Critical Review of the Experimental Literature. Journal of Environmental Psychology [online]. 29 (4), pp. 422-433. [Accessed 01 January 2022].
* Butler, C.W. and Vijayasarathy, L.R. (2016) Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter?. Ieee Software [online]. 33 (5), pp. 86-94. [Accessed 31 January 2022].
* Cham, W.K., Chan, S.S.F., Chu, L.M. and Zhao, C. (2015) Plant Identification Using Leaf Shapes—a Pattern Counting Approach. Pattern Recognition [online]. 48 (10), pp. 3203-3215. [Accessed 21 January 2022].
* Chan C. S., Lee H., Remagnino P., and Wilkin P. (2015) Deep-plant: Plant identification with convolutional neural networks. 2015 IEEE International Conference on Image Processing (ICIP), University of Malaya, Royal Botanic Gardens, & Kingston University, Quebec City, QC, Canada, 27-30 Sept. 2015: IEEE [online]. Available from: https://ieeexplore.ieee.org/abstract/document/7350839 [Accessed 23 October 2021].
* Chao, X., Li, Y. and Nie, J. (2020) Do We Really Need Deep Cnn For Plant Diseases Identification?. Computers and Electronics in Agriculture [online]. 178 (1), pp. 1-7. [Accessed 28 January 2022].
* Choo K.R., Huang F., Liu P. and Wang L. (2017) Svm Or Deep Learning? a Comparative Study on Remote Sensing Image Classification. Soft Computing [online]. 21 (1), pp. 7053-7065. [Accessed 25 January 2022].
* Dingsøyr, T. and Dyba, T. (2008) Empirical Studies of Agile Software Development: A Systematic Review. Information and Software Technology [online]. 50 (10), pp. 833-859. [Accessed 31 January 2022].
* Gajjar, R., Gajjar, N., Patel, N.P., Thakor, V.J. and Ruparelia, S. (2021) Real-time Detection and Identification of Plant Leaf Diseases Using Convolutional Neural Networks on an Embedded Platform. The Visual Computer [online]. 1 (1), pp. 1-16. [Accessed 26 October 2021].
* Garleanu, A., Mărzan, M., Paul, C., Spiru, L., and Velciu, M. (2019) THE REVERSED MOSCOW METHOD. A GENERAL FRAMEWORK FOR DEVELOPING AGE-FRIENDLY TECHNOLOGIES. In: (editor Unknown) ed. 11th International Conference e-Health 2019. Porto, Portugal, 17 – 19 July. Inderscience Publishers, pp. 75- 81.
* Jones, D.D., Meyer, G.E., Neto, J.C. and Samal, A.K. (2006) Plant Species Identification Using Elliptic Fourier Leaf Shape Analysis. Computers and Electronics in Agriculture [online]. 50 (2), pp. 121-134. [Accessed 21 January 2022].
* Karthikeyan, N.G (2018). Machine Learning Projects for Mobile Applications: Build Android and IOS Applications Using TensorFlow Lite and Core ML [online]. Birmingham: Packt Publishing Ltd. [Accessed 30 January 2022]
* Lalithnarayan, C. (2020) Handling Imbalanced Datasets in Machine Learning. Available from: https://www.section.io/engineering-education/imbalanced-data-in-ml/ [Accessed 13 February 2022].
* Lu, W.Z. and Wang, W.J. (2005) Potential Assessment of the “Support Vector Machine” Method in Forecasting Ambient Air Pollutant Trends. Chemosphere [online]. 59 (5), pp. 693-701. [Accessed 25 January 2022].
* Macario V., Oliveira J. F. L., and Pacifico L. D. S. (2018) Plant Classification Using Artificial Neural Networks. In: (unknown editor) ed. 2018 International Joint Conference on Neural Networks (IJCNN). Rio de Janeiro, Brazil, 8-13 July. IEEE, pp. 1-6
* Okamoto, K., Tanno, R. and Yanai, K. (2016) Efficient Mobile Implementation of A CNN-based Object Recognition System. In: (Unknow editor) ed. 24th ACM international conference on Multimedia. Amsterdam, The Netherlands , October 15 – 19. Association for Computing Machinery, pp. 362–366
* Patle, A. and Prajapati, G. L. (2010) On Performing Classification Using SVM with Radial Basis and Polynomial Kernel Functions In: (Unknow editor) ed. 2010 3rd International Conference on Emerging Trends in Engineering and Technology. Goa, India , 19-21 Nov. IEEE, pp. 512-515
* Rao, T. V. M. and Vani, S. (2019) An Experimental Approach towards the Performance Assessment of Various Optimizers on Convolutional Neural Network In: (Unknow editor) ed. 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI). Tirunelveli, India, 23-25 April. IEEE, pp. 331-336.
* Royal Botanic Gardens (2020) State of the World’s Plants and Fungi 2020 [online]. London, UK: Royal Botanic Gardens, Kew. [Accessed 21 January 2022].
* Zhang, Z. (2018) Improved Adam Optimizer for Deep Neural Networks. In: (unknown editor) ed. 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS). Banff, AB, Canada, 4-6 June. IEEE, pp. 1-2.

10 Appendix

Diagram

Description automatically generated

A picture containing graphical user interface

Description automatically generated