Isolate and Classify Wildlife Images
Final Year Project Report

DT228
BSc in Computer Science

Emmet Rowe
Jack O'Neill

School of Computing
Dublin Institute of Technology

12th April 2019

Abstract

The goal of this project is to build and deploy a web application that can identify the subject and remove the background from uploaded wildlife images. In addition a web view based android application will be developed as another way to interface with the web application.

Image processing implemented using TODO: Finish A convolutional neural network was created using TensorFlow and the Keras API to identify wildlife species. Uploaded images are collected and can potentially be used for expanding the dataset or creating a new one.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

Emmet Rowe

12th April 2019

Acknowledgements

Body text

Contents

1 Introduction

1.1 Project Overview

TODO: All Overview

1.2 Project Objectives

The objective of this project is to make a full stack web application with features that analyses and modifies user photos. This requires a number of smaller objectives to be completed which are listed below:

- Develop an algorithm for separating the subject of a photo from the background using image processing.
- Source a suitable dataset that contains a large number of wildlife images separated into different species
- Build and train a machine learning model that is capable of classifying the subject of a photo into a specific animal category.
- Create a website that allows users to upload photos for classification and background removal. A user feedback form will be made to enable collection of user evaluations and comments.
- Create an android application interface for interacting with the web application.

1.3 Project Challenges

1.3.1 Introduction

This is a large-scale project with challenges and issues that were encountered throughout. It was important to identify risks at the beginning, so it would be possible to prepare and possibly avoid some problems entirely. Risks that are not accounted for can cost significant time which is the most valuable resource for this project. This section covers the identified risks and challenges. These include sourcing a dataset, time management and technology experience.

1.3.2 Dataset Acquisition

A large challenge is finding a suitable dataset. The classification of input images will solely rely on a model trained with a specific dataset as machine learning is the only technique used in the classification process. It's valuable to compare the results from different splits from several datasets to observe how they influence the model. The aim is to have high accuracy classification and even if the training process is well thought out the training data is where it learns available features and their relationship with the categories.

Finding a relevant dataset is essential for the classification process. This meant a lot of time was invested in sourcing a dataset and a number of promising ones were found early on. In addition to these a number of viable methods have been thought of for sourcing suitable datasets.

A method that technically would work is scraping google image search. This would obviously be a very poor-quality dataset due to the amount of unsuitable or irrelevant images returned. To have any hope of being making the dataset viable all returned images would need to be extensively reviewed and filtered. While this could give a good resulting dataset, it is far too time intensive and therefore not feasible.

Datasets can often be sourced from educational institutions or from already completed studies. This ensures high quality images and possibly a large degree of relevancy if it is from a similar project. The last option considered for acquiring a dataset is the google dataset search tool which, despite still being in beta, looks to be a powerful tool and will likely be useful.

1.3.3 Time Management

This project requires a large amount of work on researching, designing, implementing, evaluating and testing. Each of these stages requires a time investment and all must be completed within the timeframe of 8 months which is enough as long as there is preparation so that it's allocated well. This means planning out when to start on each section and the extent of time it is to be worked on. If targets are missed catching up quickly will be essential to avoid being overwhelmed when approaching the deadline.

The most important thing is to invest enough time into the project. Having time allocated throughout for project work while leaving time for all other work such as

college modules. Following a plan consistently will ensure the project is kept on track. When encountering obstacles that prevent further development of the application, they need to be resolved swiftly. The best approach in these circumstances is to seek help. Several lecturers have said they are willing to give advice if needed. The supervisor of this project can also be asked for help in certain areas.

1.3.4 Technologies

Researching technologies is useful but it is an entirely different matter to learn them. It will take time to become competent with each technology and put them into practice. There is also the possibility that any of them could have limited functionality in the desired area, be incompatible with other required technologies or too difficult to work with to justify the time put in.

These problems can only be avoided by thoroughly researching all these aspects before choosing a technology. If a problem is still encountered despite best efforts, research should have highlighted a comparable technology that can then be considered as a replacement.

1.4 Structure of Document

Research
This chapter explores the background research into image classification and foreground extraction. It examines alternative existing solutions and technologies that can be used to achieve this projects goal. Finally it discusses the technologies and methods that will be implemented throughout this project.

Design and Architecture
This chapter explores the methodology used in the development the development of the project. It compiles the features that will be available in the web application and illustrates the features that are available to users and administrators. The technical architecture is examined and the components of each tier will be discussed.

Development
This chapter will explain the development of the web and android application components. It will discuss the building and training of a convolutional neural network and breakdown the image processing technique used for background removal. The chapter shows how the user interface and client application were implemented. The android application's creation is explained.

Testing and Evaluation
This chapter depicts the process used to test the project and ensure its quality, both manual and automated. The chapter covers the different methods used to evaluate components of the project and reviews the results.

Project Plan
This chapter covers the initial plan that was proposed for the project and the differences in how it was actually carried out.

Conclusion and Future Work
Interesting observations and results of the project are described in this chapter. Ideas for future work to improve the web application are highlighted in this section.

2 Research

2.1 Introduction

This chapter deals with key areas of research that were considered or incorporated in this project, the most important areas of research are object recognition, background subtractionand web design. Systems that have capabilities similar to this projects goals will be reviewed and technologies that can help in the creation of the project are examined for feasibility. Exploring theses topics will allow a well informed design to be created,a suite of technologies selected and effective approaches planned.

TODO: FIX:add intro to object recognition,.. web design and background subtraction

Object recognition also referred to as object classification is mainly done using machine learning nowadays. This meant exploring a number of well developed methods that are available for object recogniton. I decided to explore the general topic and see what methods were available. I also checked to see if any of the datasets they used were publicly available.

TODO: FIX headings

2.2 Background Research

2.2.1 Object Recognition

2.2.2 Edge Detection

2.2.3 Convolutional Neural Networks

2.2.4 Background Subtraction

2.3 Background Research

2.3.1 Object Recognition

Baxter et al. [1] attempted object recognition based upon segmenting an image into regions and attempting to match supplied keywords to different region types. It used a dataset that was not named or made publicly available. The approach they proposed was invariant with regard to feature type but had the issue that unwanted bias tended to be present as the dataset provided had a lack of variety.

Belongie et al. used shape matching and shape contexts to recognise objects/shapes. [2] They used a variety of datasets. The result of the study demonstrated that their simple approach that relied on estimation of shape similarity and shape context provides great shape recognition with invariance to image transformations.

Barla et al. [3] used what is known as a Hausdorff Kernel for 3D Object Acquisition and Detection. Their approach uses a kernel that examines one object type at a time. It successfully identifies the smallest sphere in the feature space using the training data and this sphere can be used as a decision surface to classify new examples. It tended to have an impressive success rate for detecting 3D models.

Lowe [4] focused on "Object Recognition from Local Scale-Invariant Features". Although he outlined an approach for finding a Scale Invariant Feature Transformation (SIFT) that can then be used for identifying additional features which then allows for robust recognition even with partial occlusion in cluttered images Lowe did not report on what data was used for the experiments or provide detailed experimental results.

2.3.2 Edge Detection

The initial idea for isolating the subject in this project was to first focus on finding the edges in a photo. Edge detection typically works by identifying points at which

image brightness changes drastically and organised into a set of line segments which are labelled edges. There are two main abstract methods, search based(5) and zero-crossing based.

Search based calculates a scale for edge strength and highlights accordingly. It seems to be more a proof of concept rather than a well-developed functional process for identifying edges. There are almost no examples of it being used to develop an algorithm or approach for recording defined edges.

Zero-crossing is more popular with multiple well-known edge detection operators using this approach. Two important ones to examine are the Sobel Edge Detector[5] and the Canny Edge Detector.[6]

Sobel Edge Detection uses kernels for estimating the gradient along the x-axis and another for the y-axis and these can be used in a number of ways to approximate the magnitude of the gradient. It can even use simplified calculations to reduce accuracy but greatly increase processing speed.

Sobel is a straightforward method for detecting edges along with their orientation and can be optimised (by sacrificing a degree of accuracy) to give almost instant results. However, even without speed optimisation it has accuracy issues and is thrown off by noise in an image.[7]

Canny Edge Detection uses differentiation to mathematically calculate edges, requires pre-processing, typically smoothing using a Gaussian filter. Gradient magnitude is calculated in a similar fashion as was used in Sobel. Finally, thresholding is used along these edges.

Canny has in built functionality for removing noise. It has good localisation and has an impressively high signal to noise ratio. Getting the algorithm to run in real time is extremely difficult as the processing of this approach is extremely time consuming.

2.3.3 Visual Recognition

Visual recognition research led to a paper about a yearly challenge that invited competitors to try get the best possible accuracy for a specific dataset with any approach they wanted.

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC)[8] has been run annually since 2010. It is a benchmark with millions of images broken down into hundreds of categories that aims to provide the necessary base for object category

classification and recognition. The premise was adopted from the Pascal Visual Object Classes Challenge[9].

The group that got the highest accuracy used an object detection system that has a focus on discouraging the algorithm from 'bad behaviour' such as recognising duplicate instances in the same photo, false positives and failing to recognise objects. It provides possible detections along with confidence scores.

They found for categorisation that from their test with humans that even the best automated system fell below the human's average by 1.7% in terms of success rate. However, an important note is that this was for humans that had undergone training, so it is likely that the top automated system would be fantastic for users as it would be more accurate than your typical person.

2.4 Alternative Existing Solutions

2.4.1 Subject Detection

Subject detection can be completed by identifying the background or the foreground and then it is simply removing the background from the original image.

It uses image segmentation which groups areas that share certain characteristics. A simple approach to this is thresholding where the image is split into two parts that are in a specific range of RGB values and those that are not. Two other methods for this are the GrabCut[10] algorithm and the Watershed[11] algorithm.

Thresholding is the most basic subject detection method. It is difficult to define a generic approach, it might be possible through thorough analysis of the image statistics but still it would probably need further processing.

Thresholding gives a well-defined separation between foreground and background but even with great image analysis will not be dynamic enough. The solution provided would most likely have a high failure rate if additional features for pre-processing the image were not used.

The GrabCut algorithm on its own doesn't suit the project well but could be useful for testing purposes. A user specifies a rectangular region that contains all the foreground. Everything outside of this is marked as definitely background and is used to identify everything else that is background. A result is shown, and the user can make small marks on regions that are incorrectly labelled and the algorithm does another run-through taking this into account.

TODO:Add grabcut diagram

GrabCut is extremely accurate given the right prompting but requiring user input slows down the experience, makes it less user friendly and adds a lot of difficulty in mass testing. Could potentially be used for giving feedback on a failed result for a different run of an algorithm.

The Watershed algorithm is an automatic process. It requires some pre-processing, so it begins by thresholding the image and denoting definite-foreground and definite-background. A distance transformation is applied to help split touching objects and the separated object areas are saved as an array. OpenCV then has an inbuilt watershed method that can be called.

Watershed is efficient, gives great results and has lots of fine tuning available for the pre-processing but can falter if there is little differentiation between foreground and background.

2.4.2 Object Classification

A research paper called Cats and Dogs[12] is a project based around identifying whether a picture contains a cat or a dog and then what breed the animal is. It uses a lot of techniques that will be beneficial to look at as they could pertain to my project. The project finds the face using a deformable part model as it can't be too strict and try focus on rigid shapes due to the countless factors that can distort an animals face as even a slight change in viewpoint would invalidate that kind of approach. There are then two different classification approaches that are compared against each other. Firstly, a two-tier approach where the animal is decided to be either a cat or a dog and then assigned a breed. Secondly, a simpler approach where the program only attempts to match a breed to the image without classifying it to a type of animal.

The separation between foreground and background is made by using the segmentation technique known as GrabCut, which usually relies on user input to get a rough bounding box around the subject of the photo but is made automatic in this case by getting the estimate of the bounding box programmatically.

The goal of this project was making a system with the ability to accurately identify the breed and family of two specific animal groups. There were three different approaches, the last of which is a combination of the first two. Shape only gave an accuracy of 94.21%, appearance only was 87.78% and combining them resulted

in an improvement over both with an accuracy of 95.37%. As expected, the more information provided about the image the better the result, this can also be noted for the appearance-only approach where accuracy improved as more localisation was done to the pet body.

2.5 Technologies Researched

2.5.1 OpenCV

OpenCV[13] was originally developed by the Intel Research Labs with the hope to improve the ease of access to hopefully advance vision research by providing open, well optimised code for computer vision basics. This would save a lot of time on the necessary setup for tackling image processing. They wanted their code base to be used as a standard so that knowledge and techniques would be more transferable. The OpenCV library was made available on as many platforms as possible so that it would be extremely portable. The license was made to be non-restrictive so that it could be used for more than just public open projects.

There are many benefits of using OpenCV, as mentioned above it is quite portable so it's easy to incorporate it into different code bases. It is free to use which is always a positive and this paired with its great accessibility results in high usage in the industry and a lot of experienced users writing.

It has low ram usage and is very fast for its C/C++ implementations. There is also a python library that while slower, can be preferable due to having a simpler code base and more readable scripts. OpenCV-Python works with Numpy which is an extremely efficient library that has a wide range of mathematical computations. Numpy is used by a large number of libraries so its implementation can improve compatibility with other technologies.

2.5.2 TensorFlow

TensorFlow was developed by the Google Brain Team and is heavily used in the realm of machine learning. It has in built functionality for running computation on CPUs, multiple GPUs and TPUs and on devices such as mobile, desktop and server clusters. This allows users to minimise deep learning training time by sharing the load.

It works with Python 3 and has some image processing abilities, however most of these are designed for formatting images to be used as training data. It has many useful APIs, for example keras. Keras is well suited for fast prototyping as it is designed based on user interaction rather than to fully describe machine learning logic. This user-friendly design paired with its modular design make it a great tool for making an implementation quickly that has the ability to be expanded upon later.

It is well suited for deep learning and is perfect for neural networks with lots of layers and strange topology. TensorFlow has great tools for visualisation that assist in debugging and optimising applications. It is a widely used and freely accessible framework that is extremely well documented with hundreds of tutorials.

2.5.3 Amazon Web Services

Amazon Web Services (AWS) is a cloud services platform that offers functionality such as content delivery, computing power and database storage[14]. AWS had a bare-bones launch in 2002. Its focus is on providing businesses solutions that increase flexibility, scalability and reliability.

The amount of technologies it is able to support is astounding. It has ways of incorporating almost every technology researched in this section apart from rival cloud services of course. It is easy to use and deploying basic applications to the cloud can be achieved quickly by following a range of their well laid out tutorials. The scalability solutions mean there are no limits on capacity and the current capacity can be upgraded as needed. Other services force businesses to select specific tiers meaning money is wasted if future use is over estimated, while AWS lets businesses pay for the storage used.

Flexibility means you can get access to needed services quickly. Business in need of technical expansion typically have to plan acquisition and setup of hardware that could take days before forward progress can be made. AWS allows access to these services almost immediately and with tools for elastic load balancing business can match their exact demand.

AWS has good security that doesn't require any extra effort on businesses end to maintain. All data is kept on their private secure servers and protection is size invariant, so vulnerabilities don't arise for even the largest databases.

A big disadvantage of the platform is their technical support fees. The service is meant to be almost completely automated for the user so it should be rare that

support is needed but for moderately sized businesses, monthly support fees can be hundreds or even thousands.

2.5.4 PythonAnywhere

PythonAnywhere[15] is a python based development and hosting environment that allows enables users to get web applications up and running quickly. It's developer focused with many tools such as an in built integrated development environment and automated task scheduler that allow easy management of websites.

PythonAnywhere has great tutorials and application templates for starting a project, especially applications using the Flask or Django web framework. It also has simple configuration so apps already under development can be pulled from GitHub and setup to run publicly in a matter of minutes.

It has effective load balancing features, however it could use a finer level of control for this area. It has a batteries included feature that makes many libraries available straight away without the need of a virtual machine.

The pricing is great for smaller projects as a free tier is available indefinitely while if more resources are needed all paid plans can be highly customised so users don't pay for features they don't need.

2.5.5 Digital Ocean

DigitalOcean define themselves as "a values-driven organization" that believe they should, "Start by defining the problem. Have the courage to approach a problem from a different perspective. Plan ahead, but act decisively. Consider that the simplest approach is often the best." [16] They have a large focus on developer experience and this influences their user interface design. This can be seen in the control panel for their service, where a lot of time consuming tasks are made simpler with fast compute server creation, reliable object storage and management tools for infrastructure.

The DigitalOcean cloud is SSD-only. This is rare in the cloud services market as it's cheaper to use slower mechanical hard drives in the creation of a cloud platform. The performance benefits of the system are available to all users allowing server launch in less than a minute. This paired with the control panel makes server configuration quick and easy.

The quality of the hosting is incredible and has a low barrier of entry as it is less than €10 a month. This means it is a tool that is feasible for both a lone developer or a giant business. Learning to use digital oceans service is easy as there are many tutorials available as well as a well-developed Q & A Section.

DigitalOcean have many Linux distributions available but not even a single kind of windows distribution is available which can be limiting depending on developers experience and preferences. Also they don't use centralised storage, meaning backups need to be handled by the user if the data loss risk is to be mitigated.

2.5.6 Flask

Surprisingly, Flask was created as an April fools joke[17] and had its initial release in 2011. Despite this odd start, it has grown to be one of the most popular web frameworks for non-enterprise developers. This is most likely thanks to its versatility and lack of boilerplate meaning it is very pythonic and not too abstract.

Flask is a microframework that supports hundreds of extensions so the features that can be implemented suit the majority of users needs. This means you're not locked down to a specific database or templating engine, it gives developers a lot of freedom.

It is extremely easy to get started with flask, an application can be made in less than 10 lines. These basic apps can then be expanded upon to improve understanding and start to develop into useful web apps. This is something to be valued as some web frameworks can take many hours of learning to even get a proof of concept app running reliably. Flask is a great tool for any developers that enjoy the hands-on approach for learning a new technology.

A downside to the versatility means developers can find themselves in uncharted territory if they are using a rare combination of extensions. Even with an active community there is going to be many blind spots in the documentation for a framework such as Flask.

2.5.7 Django

"Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development"[18] They also make the important note that the framework is free and open source.

Django is designed to speed up the creation of web applications. It is referred to as a batteries included framework meaning that developers have what the tools they need to power their web application out of the box. Even though many of the features may go unused they are wide ranging enough that most elements of a project can be created just using the default packages. This means after successive applications have been created with Django a developer can quickly deploy their arsenal of tools on a new project and reach completion faster.

This also highlights how Django developers can be transferred far more seamlessly than other framework developers. The tools being ubiquitous among Django projects ensures that developers can jump right into a project at any point. Also, with it being a scalable framework that means there are many approaches for handling larger projects such as running separate servers for each element of the application or load balancing.

The main negative point for Django is how it can be quite obtuse for smaller projects. It can be annoying having such a large project structure for a single page web personal website. Of course, this doesn't make Django any less effective, just a bit overkill when it comes to simple applications.

2.5.8 MySQL

"MySQL is the world's most popular open source database. With its proven performance, reliability and ease-of-use, MySQL has become the leading database choice for web-based applications, used by high profile web properties including Facebook, Twitter, YouTube, Yahoo! and many more."[19] It is a high performance database management system that is compatible with a lot of different setups.

It is quite proficient at handling multi-user interaction unlike the previous technology. It is a fast and stable server that is well suited for programs that handle large numbers of concurrent additions and edits such as order or booking systems.

It is incorporated in the LAMP stack used in many web-based applications. LAMP (Linux, Apache, MySQL, PHP/Perl/Python) is widely used throughout the industry and is one of the leading open source web platforms.

The only disadvantage is the requirement of a running server for access, however this is the case for all but a few database implementations.

2.5.9 SQLite

SQLite is one of the most basic database implementations possible. It is entirely self-contained, does not require a server or even any configuration. It is different to the SQL databases that require a server process as it reads and writes directly to disk.

The main advantages and disadvantages for SQLite are directly related to scale. For a simple database with only a few tables it is perfect for the task, easy setup, simple operations and no requirement to run a server process. For a large database accessible by many users simultaneously there will be a large drop in speed. This is due to the database being locked during access meaning users need to queue for access. That means there is a point at which an SQLite database base will become unusable with no way of improving performance above a specific scale.

2.6 Resultant Findings and Requirements

2.6.1 Image Processing

The above research covered the image processing library OpenCV. Its Python library is well documented, and has been shown to be quite intuitive. The main downside to OpenCV-Python is its lower speed compared to the C++ version. The use case for image processing in this project can manage without the second or two speed increase of the C++ version as the benefits that the python implementation provides are higher priority.

The python version has many more compatible libraries that allow it to be easily implemented alongside other technologies into a web framework. A larger benefit is the simplicity of development in python which allows functionality to be built faster and will reduce overall development time.

2.6.2 Image Classification

Taslk about tensorflow wwith or without keras, keras is nicely abstracted, allows focus on the model building and dataset instead of the training process

TODO: Web framework and hosting service/cloud platform

3 Design and Architecture

3.1 Introduction

Following on from the previous chapter, where some of the key background research was presented, these themes will be continued in this chapter, where the design of the system will be presented. The first section will look at the software methodology employed in this project. The next section outlines the technical architecture of the system and gives a detailed breakdown of the three tiers.

3.2 Design Methodology

3.2.1 Feature Driven Development

Feature driven development (FDD) is defined as an iterative and incremental software development process. It is a variation of agile methodology that has a big focus on end user experience. It has five main iterative activities that repeat during the development process.

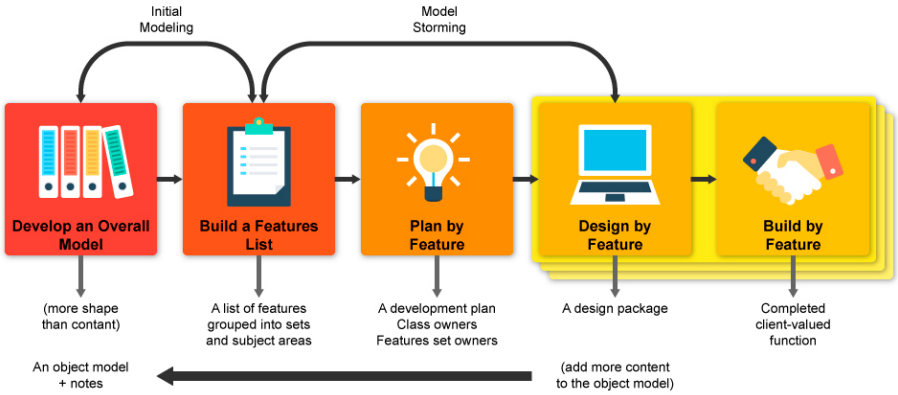

Figure 1: The steps of feature driven development

Develop an Overall Model: First of all there is a high level overview of the system to understand components priority and how they will fit together. Next, models are created for different domain areas and are reviewed. These models will slowly merge into a single comprehensive model.

Build a Features List: Domain areas are broken down into their possible interactions and this allows features to be identified. Each feature can be expressed in a the format "¡action¿ ¡result¿ ¡object¿", for example 'Filter images in gallery'. It is important that no feature takes more than two weeks to complete. If an activity will most likely take longer then that time frame it should be broken down into smaller more manageable steps.

Plan by Feature: Priority is decided for each features based on factors such how mandatory are they for system operation, are they a prerequisite for other features and do they improve system usability. These are taken into account and a development plan is made.

Design by Feature: The necessary functions for each feature are defined and a sequence diagram is made which will be implemented in the next step after a design inspection.

Build by Feature: The feature is developed and implemented and unit tests are made for it.

FDD is typically performed by development teams so it will be used on a much smaller more direct scale, mainly so there will be clear direction and progress while having considered the overall structure of the project and not losing sight of the goal.

Reasons that FDD is suitable for this project:

- The straight forward five step process makes it easy to get started quickly.
- Risk of not having a finished product is reduced as a base application is built early to slot features into.
- Easy to adapt and add new features as new needs arise.

Reasons that FDD is unsuitable for this project:

- On smaller projects only one person creates models.
- No written documentation.

3.3 Feature List

This section lists and describes the features that were decided on for the application

- Image classification - Any user can upload an image that the system will attempt to classify and return that label to the user.
- Background removal - Users can upload an image which the system will analyse and attempt to remove the background. The resulting image with a transparent background will be provided back to the user.
- Submit feedback - After having an image classified and isolated the user can leave feedback details such as label accuracy, isolation quality and general comments.
- View gallery - Users can navigate to the gallery page to see all images uploaded by users that have been given mod approval.
- Filter gallery - Users can filter the gallery by animal species or type of image e.g. original or isolated image.
- Theme selection - By default the website has a dark theme but users can change it at any time to a different colour scheme.
- Account registration - Users can create an account for the application.
- Account login - Once a user is registered the they are able to login to their account.
- View own uploads - Registered users have access to a personal gallery where they can see the results of all the images they uploaded regardless of whether they have mod approval or not.
- Mod approval - Admins have access to an admin approval page where they can approve, deny or delete any user uploads.

3.4 Use Case Diagram

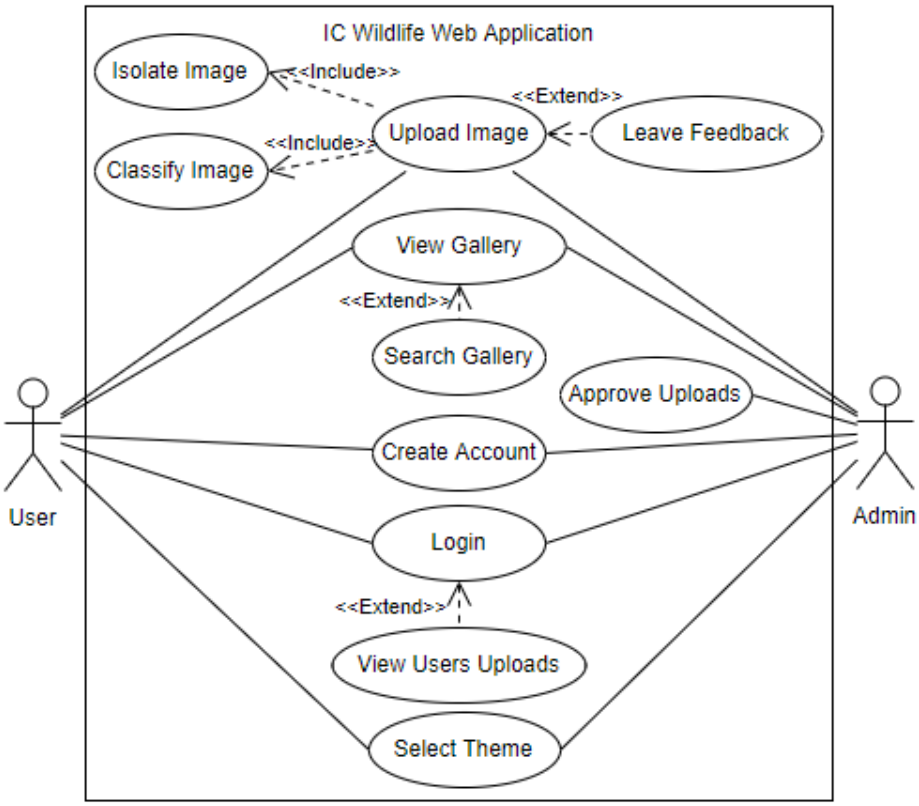

Figure 2: Use Case diagram for IC Wildlife web application

The use case diagram above shows the base functionality of the IC Wildlife web application. The main use case is Upload Image and it is key to the application as it includes the two most complex use cases and extends another. There are two different kinds of actors that will use the application, users and admins and available features to each are shown in the diagram. Admins have access to all the same use cases as base users except they also have access to approving uploads to appear in the public gallery.

3.5 Technical Architecture Overview

This section discusses the technical architecture of the system, it first discusses the front-end of the design known as the presentation tier. This is where the user interface

is located and client side validation occurs. It is the only layer visible to users so there is more to consider than just input and output. The next section is the middleware/logic tier which handles the main processing and handles requests from the presentation layer. The final element of the technical architecture is the back-end design known as the data tier. This tier handles data persistence and data access.

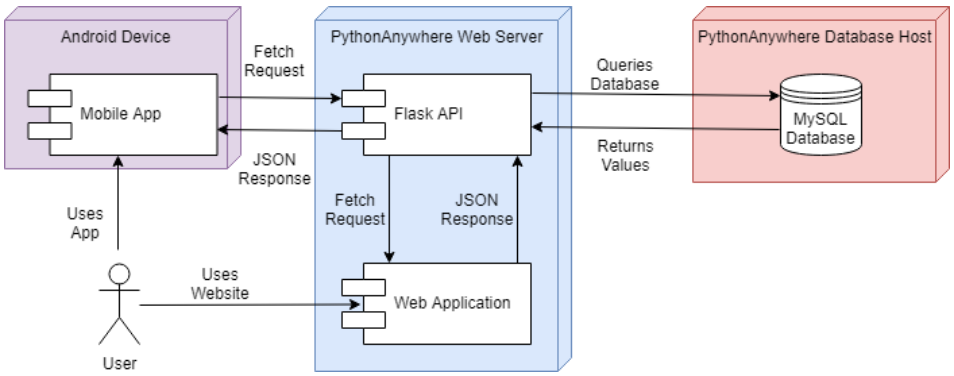

Figure 3: Technical Architecture Diagram

One thing to note is that technically the web application pages are provided by the Flask API. however these two elements operate completely differently in practice as they are logically different elements and the above diagram represents this separation. The web application interface could be separated from the API but this would increase complexity without any any practical improvements.

This diagram represents the structure of the application well but programmatically it is split into the previously mentioned presentation, logic and data tiers. The following diagram is an abstraction that shows the components organised into their tiers which accurately represents the flow of the application.

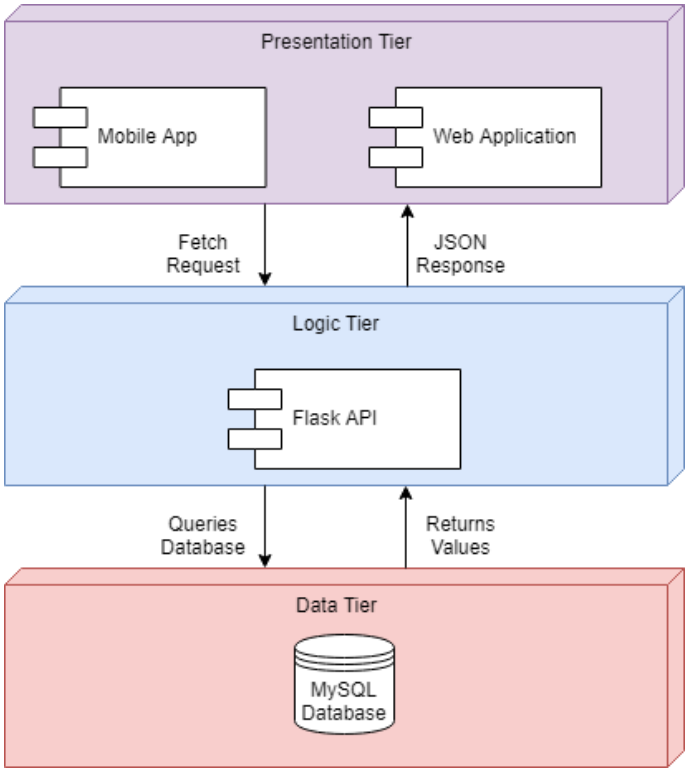

Figure 4: Three-Tier Architecture Diagram

In the above diagram the web application and mobile app are the components of the presentation tier, the Flask API is the logic tier while the MySQL database is the data tier.

3.6 Presentation Tier

The presentation tier has to provide users an avenue for interaction and needs to be able to structure data received from these interactions so that request for processing can be made to the logic tier. The user interface needs to present the user with all the information that is necessary to navigate and operate features of the web application. The presentation tier needs to be robust and forgiving of user mistakes so the application does not become irritating and/or difficult to use. A focus is put on error prevention over error handling where possible which increases the fluidity of the application. Actions that change the context of a page should affect the interface accordingly e.g. loading messages are displayed in the image frames upon upload and are replaced with the result once a response is received.

3.6.1 Initial User Interface Design

When making a user interface there are a number of best practices that should be followed.[20] These include keeping interfaces simple, using common UI elements, and being purposeful in page layout. These were all kept in mind while creating the initial screen wireframes.

The initial designs were made assuming they would be viewed on desktop screens, however they are placed in a loose grid layout so elements can easily be scaled or moved to adjust for varying screen sizes. This support allows the same UI design to work on both computer and mobile due to it's adaptability. Without this approach the plan to create a user friendly android application would not be feasible.

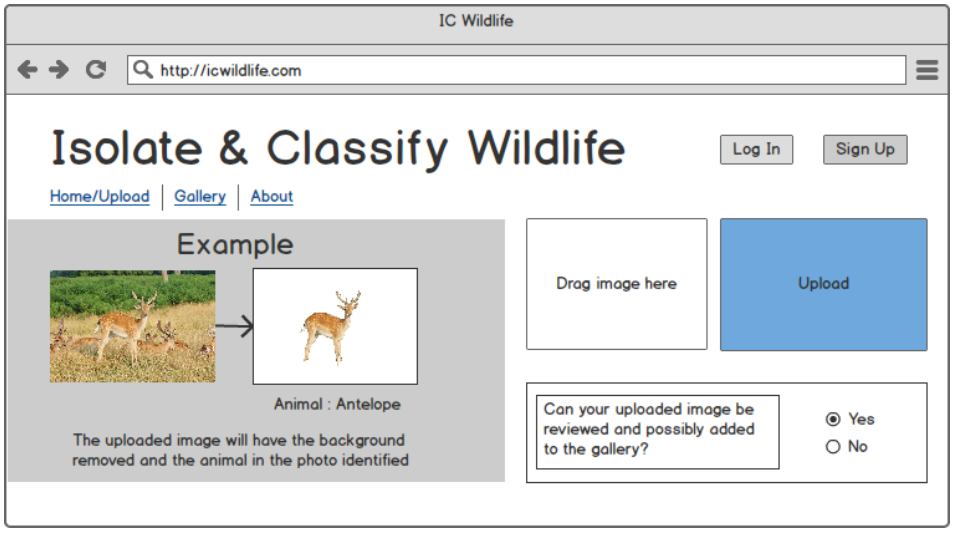

Figure 5: Home page wireframe

The navigation bar is at the top of the page and contrasts the background. Log in and sign up are easily visible but further away due to the fact they shouldn't need to be used more than once a visit. An example of what the site does is displayed prominently on the page. This visually shows the user that an image of an animal going in will result in a label being added and the background of the photo being removed.

The upload section has clearly labelled buttons with a more prominent colour on the upload button. The buttons along with a permission input are grouped together in

a square. An example of how display will be handled for smaller screens is that this square would be shifted to underneath the example images to fit the different scale.

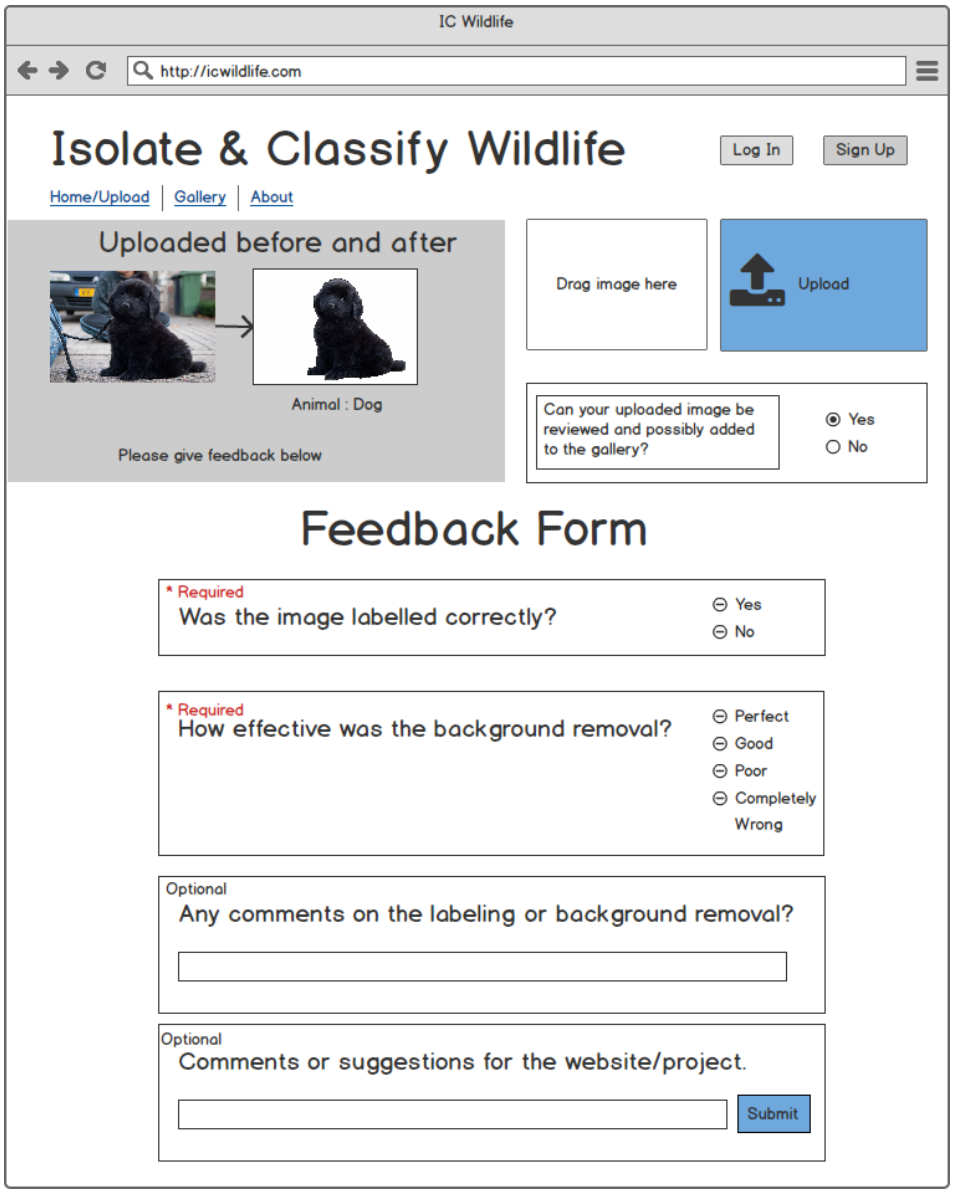

Figure 6: Home page after upload wireframe

After an image is uploaded the example is replaced with the before and after images. This prevents the screen from getting cluttered by having two similar elements

stacked on top of one another. The upload buttons remains so that additional uploads can be made.

An optional feedback form appears below. The most important feedback has simple radio button inputs as a too involved process might discourage users from leaving feedback. The ratings for background removal are a bit subjective so a button to show example images for each rating will be added. The comment input boxes are non mandatory as many users will simply not have any comments they wish to say. The submit button will have it's size and positioning adjusted to make it more prominent.

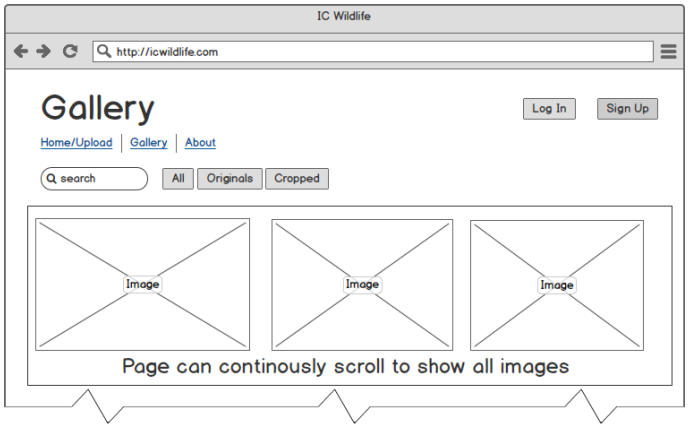

Figure 7: Gallery page wireframe

The gallery page by default displays all images and can be filtered by searching a species of animal and specifying original or cropped images. The images are displayed along with their labels and the page will continually scroll to show all images.

The my uploads page is not shown as it is identical to the gallery page except only images that the logged in user uploaded will be displayed.

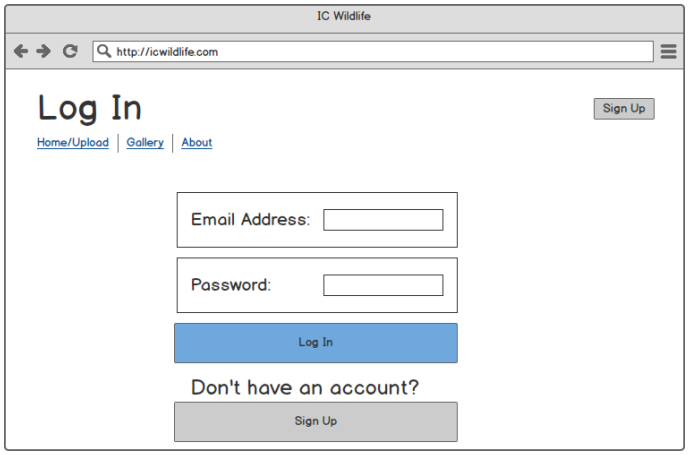

Figure 8: Login page wireframe

A basic form for logging in. It is done in a simplistic style as there is no need to over complicate straight forward pages. Client side error checking will be used to inform users if invalid input such as a blank password is used on a login attempt and server side validation will inform of incorrect login information. A link to the sign up page is included at the bottom for convenience and also to indicate to users who may have misclicked sign up that this is the wrong page and this button can get them to the correct one.

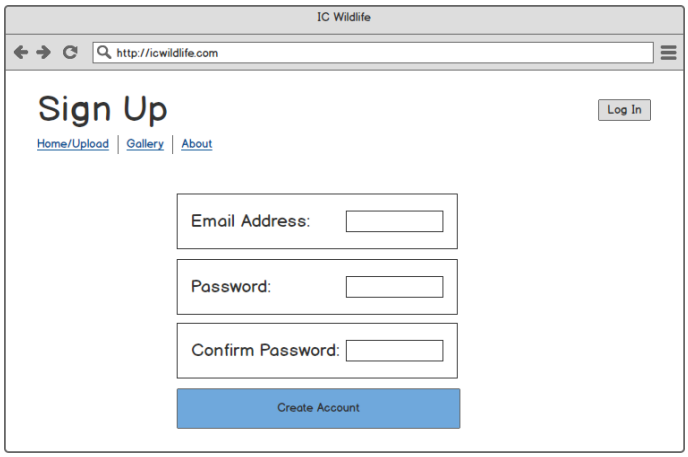

Figure 9: Sign up page wireframe

The sign up page follows the same simplistic styling as the login page. Client side validation will check that the password fields match while server side validation will

check if the requested username is available.

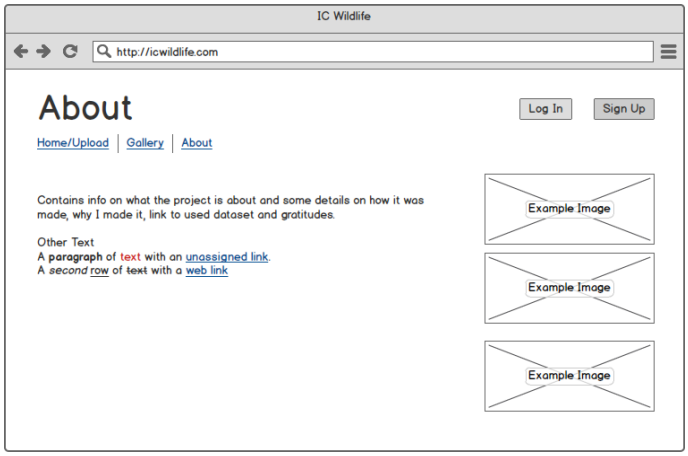

Figure 10: About page wireframe

The about page has basic information on it and is only a display page with no unique functionality. Images are added to the side of the page to make it more visually pleasant.

3.7 Logic Tier

The logic tier contains all server side processing. It needs a code base that can fulfil all advertised features. It communicates with both the presentation and data tier. Logic tiers are designed to be durable, reliable and predictable. This tier has the job of handling both image processing and image classification using a convolutional neural network while also being the middleman for getting data tier results to the front end of the application.

3.7.1 Background Removal

It was decided that the background removal would be completely separate to the machine learning process. Incorporating machine learning would improve accuracy for classes in the dataset but would not be applicable to any other animals. In theory the generic approach using pure image processing and analysis will work for any image with a clear subject.

Image analysis can be a time consuming and complex process, the proposed approach for this application will start by breaking images into regions known as superpixels.

Superpixels[21] are simply regions of pixels that are a similar colour. They are calculated by first partitioning an image into grid squares and then making multiple passes refining each superpixel by shifting individual or groups of pixels at boundary edges from one superpixel to another. Using these will allow fast analysis of the image and easier attribute gathering of regions of the image. The superpixels will be used to find a bounding box around the image for use with the GrabCut algorithm.

3.7.2 Animal Classification

The most important design decision for the animal classification part of the project was dataset collection. ImageNet is a dataset organised into different categories known as synonym sets (synset)[22]. The synsets were explored for use in this project but there were not enough specific species synsets to create a viable dataset. The CIFAR-100 dataset[23] is comprised of 32X32 pixel images and 35 of the categories in it were considered for use in this project. However, while the small images would be quick to train, there is a huge loss in identifiable features for images this size.

The dataset that was decided on to use in the project was the Animal with Attributes 2 dataset[24]. It contains categories for 50 different animals. The images are of good quality with high resolutions and an average of 746.6 images per category. The only issue is that some categories have a significantly smaller amount of images than others, however this can be resolved with oversampling.

A convolutional neural network model will be used with this training data to get a usable model. The building ofthis model will be expressed in more detail in the development chapter.

3.7.3 Security

There are a few points in communication with the logic tier where sensitive information such as usernames and passwords are transferred. The system will be designed to use HTTPS (HyperText Transfer Protocol Secure) instead of HTTP so that these points of transfer will not be vulnerable. Login state is handled using JSON Webtokens, the client is given one on successful login or signup and this token is passed along with any user dependant requests. To ensure tokens have not been interfered with they are signed upon creation using HMAC-SHA256, and this signature is validated when passed with client requests. Additional security is also incorporated by the cloud provider. The data passed along for each request is also validated to ensure it contains all expected values in the correct formats.

3.7.4 API Configuration

It was decided to make all functionality for the webserver operate through API calls. All requests are handled in a similar fashion where a GET or POST request is received by the server, the relevant validation and processing occurs and then a JSON response is returned. This allows the webserver to work independently without needing an understanding of the front end design. This style of webserver can support many program types other than web application, for example a console app that can format http requests could tag and remove the background from batches of images without needing any kind of user interface or interaction at all.

3.7.5 File Storage

The design decision was made to store images on the server rather than in the database. Storing images in a database is more expensive than file system storage, it is slower to access an image in a database, webservers have in built processes for accessing images that databases don't and it is far less complex to merely store file paths in database entries.

3.8 Data Tier

The data tier deals with persistent storage. A database design has been created to support the necessary elements of the project. For this application the primary elements being stored are user submissions. Each submission is comprised of a main submission entry and three associated images. Optionally feedback can be attached and the submission can record the user who uploaded it.

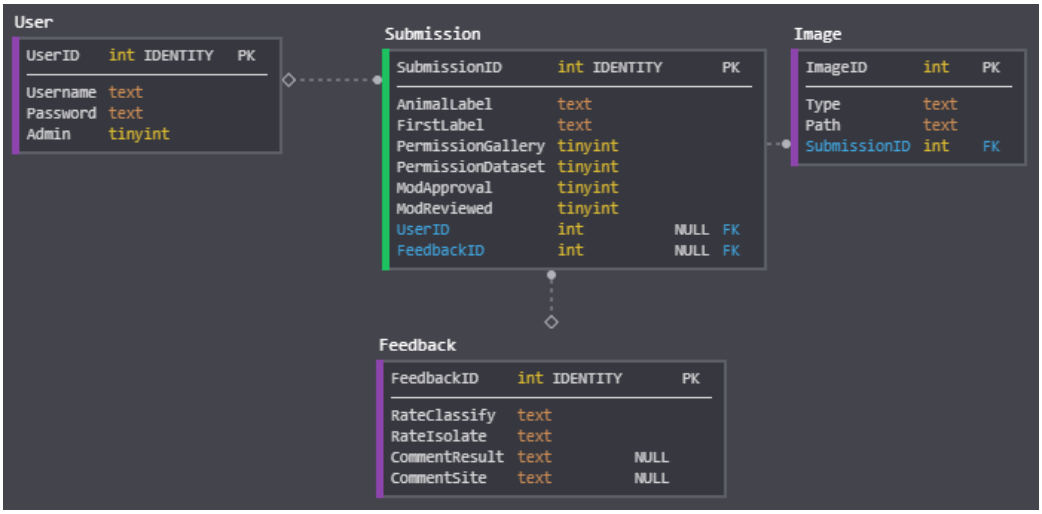

Figure 11: Entity Relationship Diagram

3.8.1 Tables

Submission

The submission table contains data relevant to user submissions. It has a unique primary key that is auto generated. It contains two labels, one for the initial animal label a submission was given and a second that can be updated to accurately reflect what animal is present in the image if the first label was incorrect. It contains two attributes for recording user permissions, Permission Gallery indicates whether the associated images are allowed to appear in the public gallery. PermissionDataset is whether the user will allow the image to be included in public datasets for machine learning. ModApproval indicates whether an admin will allow a submissions images to appear in the public gallery, this attribute should never be positive if PermissionGallery is negative. ModReviewed keeps track of if a submission has been considered for moderator approval. The last two attributes are optional foreign keys to associated user and feedback form.

User

The user table contains all data required for a user account on the web application. Username and Password are required for verification during login and username must be unique. Admin records the permissions of the user to access moderator functionality.

Feedback
The feedback table contains user responses to a number of questions. RateClassify states whether classification was successful or not. RateIsolate records the level of success the application had with background removal. CommentResult and CommentSite are optional fields that can store users thoughts on the general process and site or mobile app.

Image
The image table contains all the attributes that are relevant to an image that aren't already included in the submission table. Type records whether the image is an original, isolated or summary image. Path stores the filepath to the image.

3.8.2 Queries

Queries will be needed for features such as the gallery, login, signup and admin approval. Gallery queries will all require that images have mod approval and user permissions from their associated submission entries. These image queries will also have the option of being filtered by type and animal species.The admin approval queries will return all information related to a submission including images, user and feedback. The remaining queries will handle basic information like matching and comparing usernames to passwords and ensuring duplicate accounts aren't made.

3.8.3 Inserts and Updates

The majority of inserts for this application will be in relation to submissions. On upload if it is a valid image, three image entries will be made for storing the original, summary and isolated images which will then be attached to the submission object along with the permissions given by the uploader. If this occurs while a user is logged in their account will be associated with the submission. User entries will be inserted on sign up. Updates will occur for attaching user feedback to an existing solution and for marking the mod approval status on submissions.

4 Development

4.1 Introduction

This chapter continues the subjects explored in the previous chapter, and will outline the components created during the development process starting with the building

and training of a convolutional network. It then describes in detail the development of each level of the three tier architecture.

4.2 Convolutional Neural Network

The convolutional neural network (CNN) was built using the Keras API for Tensorflow. It was built and trained offline due to processing power limitations on the web server.

The dataset used for training is the Animal with attributes 2 dataset that was discussed in the design section which is comprised of 37,322 images of 50 animal classes. The dataset was a great asset to the project due to its variety, the only weakness was the inconsistent number of images per animal class, it was considered mixing it's contents with other datasets but other datasets that were considered didn't have the perfect mix of quality and variance to easily be integrated.

The model is a sequential model made up of a number of layers that use the previous layers output in the next layers input. However there are only five different types of layers used which are convolution, pooling, dropout, flatten and dense layers.

Convolution
Convolution layers are used to to reduce the relevant size of the input while summarising pixels and their neighbours. It goes through an image and focuses on one element at a time where it adds the element to its local neighbours, weighted by a kernel. This process results in a feature map.

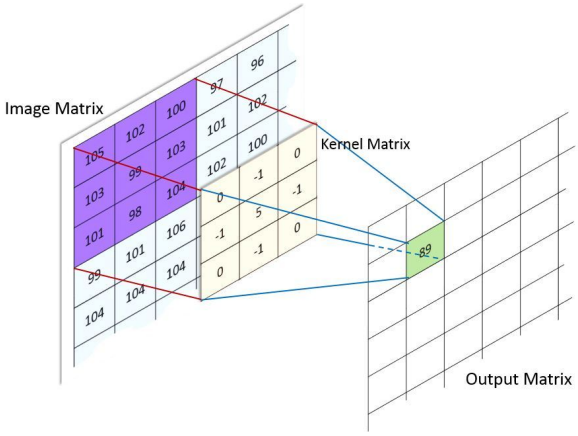

Figure 12: Basic illustration of convolution

Pooling

Pooling layers group elements together on a feature map and only pass on one element per array. Max pooling is the type used in this network and it simply takes the max value from each array.

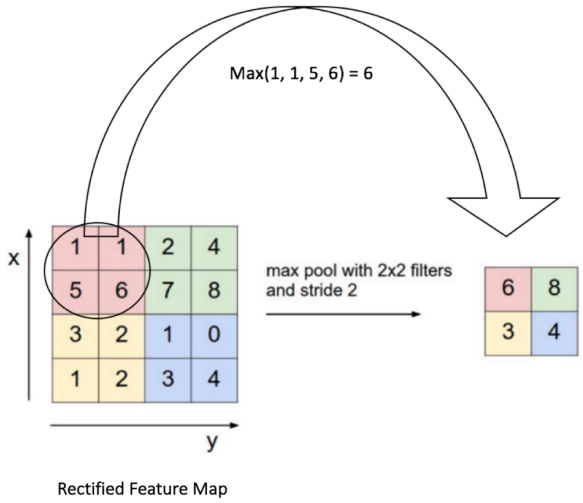

Figure 13: The pooling process on a feature map

Dropout

Dropout layers are used to help prevent overfitting. Some neurons in a neural network will become too specialised and not relevant to unseen input. Dropout is the process of randomly removing a specified percentage of neurons and can help trim away undesirable neurons.

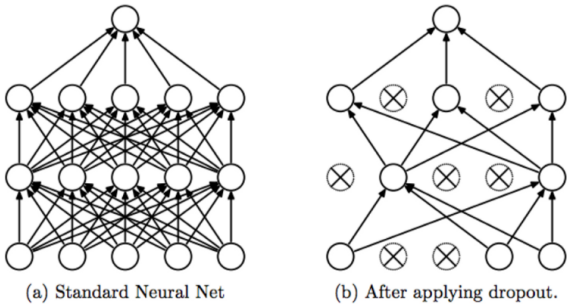

Figure 14: Dropouts affect on a neural network

Flatten
Flatten layers change the dimensions of an input. In this application it converts a 2D feature map into a 1D array.

Dense
Dense layers specify how many targets the neurons are for. In this application the most important dense layer is used to specify that there are 50 output classes. Softmax activation is used so that results are normalised into a probability distribution in the range 0 to 1 and can be considered as probability values.

The model alternates between 2D convolution and max pooling so that finer levels of detail can be found. It has the potential to build up basic elements like edges and corners and through a few iterations may start understanding shapes and features.

Images needed to be formatted going in so they would all be the same dimensions. Padding had to be added to many images as re-sizing images without accounting for its proportions left images distorted beyond use. The padding style uses blank pixels as filler, other styles such as extending the edge colour linearly outwards added the risk of misidentifying these lines as relevant features. The blank pixel method meant that the model took very little notice of padded areas.

Training initially took place on 50 by 50 resolution images due to memory constraints and the need to test quickly. Given time a dataset loader was created which allowed more effective batch training and removed the memory limit for training sets. This allowed 200 by 200 resolution images to be used, this was a nice balance between quality of the image and time feasible to train with. The training process still took multiple hours but all the issues had been worked out so the model only needed to be trained a couple times to save the weights and evaluate the model.

This setup resulted in a model with 62.2% accuracy. The weights from this model were able to be saved and transferred to the web server so the weights could be loaded into an identical model and used in the web application.

4.3 Presentation Tier Development

The web application interface is structured with HTML and styled using bootstrap and a small amount of custom CSS. Client-side functionality was implemented using JavaScript. The android application was made as a convenient way to access the website from a mobile device.

4.3.1 Application Interface

The grid like structure decided on during design is implemented using bootstraps grid system. This provides the ability to specify what fraction of the parent container an element should take up dependant on the current screen size. To be more precise, the bootstrap row element is broken down into 12 columns and child elements define the amount of columns they are going to take up.

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
# of columns	12				
Gutter width	30px (15px on each side of a column)				
Nestable	Yes				
Column ordering	Yes				

Figure 15: Bootsrap Grid System Attributes

Clever use of the grid system enables the implemented design to scale fluidly between screen sizes, for example the gallery page changes the amount of elements that it will display per row as the screen changes.

```
<div class=" col-lg-3 col-md-4 col-sm-6 col-12 theme-colour-c">
    <div>
        <img class="img-responsive border mx-auto d-block" src="${path}" width="100%" >
    </div>
</div>
```

Figure 16: Bootstrap column width classes

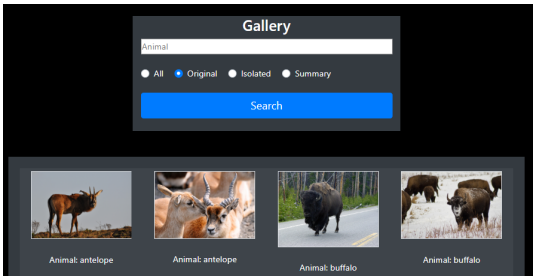

Figure 17: Column sizing on a large screen

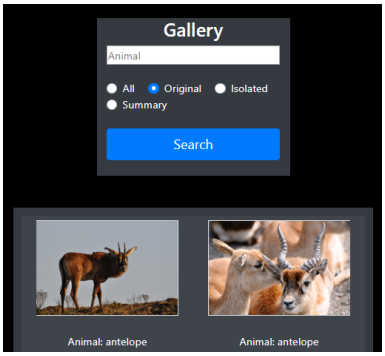

Figure 18: Column sizing on a small screen

A number of theme files were made which all contained the same classes, the only difference being the background and text colours for each class. Colours increase in brightness from background to foreground, following this style means the visual style remains pleasant and elements are still highlighted in the same tiers despite colour changes.

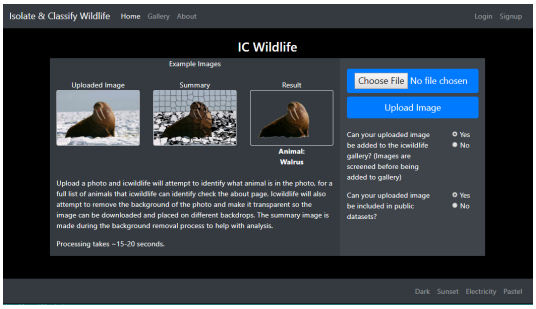

Figure 19: Dark theme home page

Figure 20: Sunset theme home page

Figure 21: Electricity theme home page

Figure 22: Pastel theme home page

The final page layouts were implemented almost exactly as planned in the design section of the report, there are only a few differences. An admin approval page was designed with features such as the ability to relabel incorrectly classed images and view user feedback alongside submission images, feedback is marked as null if the user did not leave any. File selection is handled using a FileUpload object, this functionality is widely supported which lets the operating system handle directory navigation. Text boxes are all basic text input fields while buttons are stylised using bootstrap. The gallery page has a filterable dropdown menu so animals can be easily searched.

All final page designs are shown below, excluding the my uploads page as it is identical to the gallery and the default home page as it was shown already along with it's possible themes.

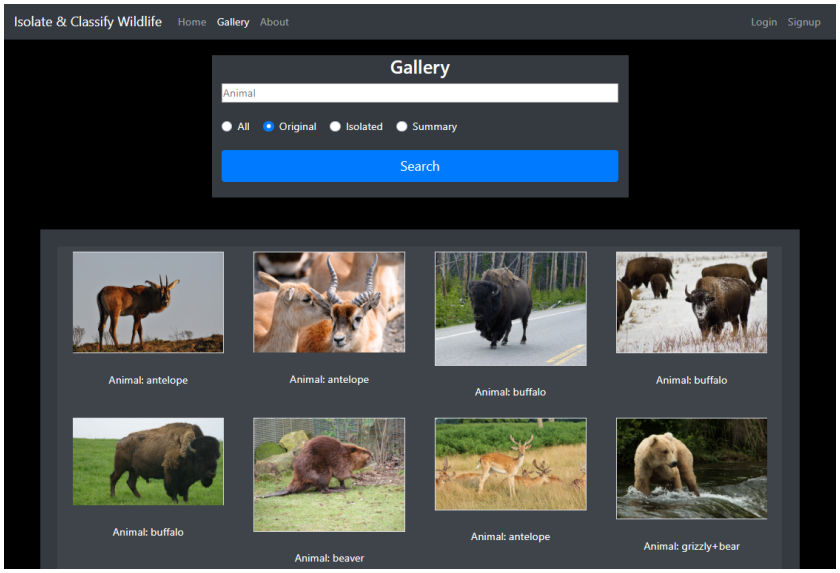

Figure 23: The gallery page

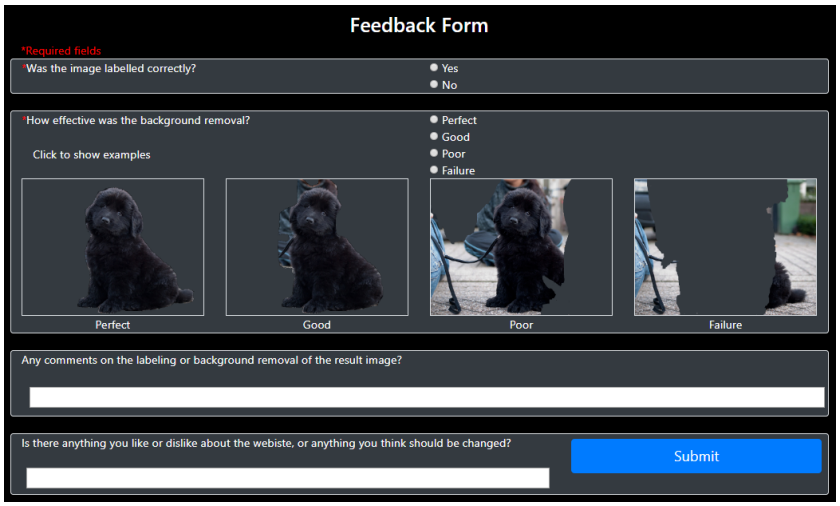

Figure 24: The submit feedback form

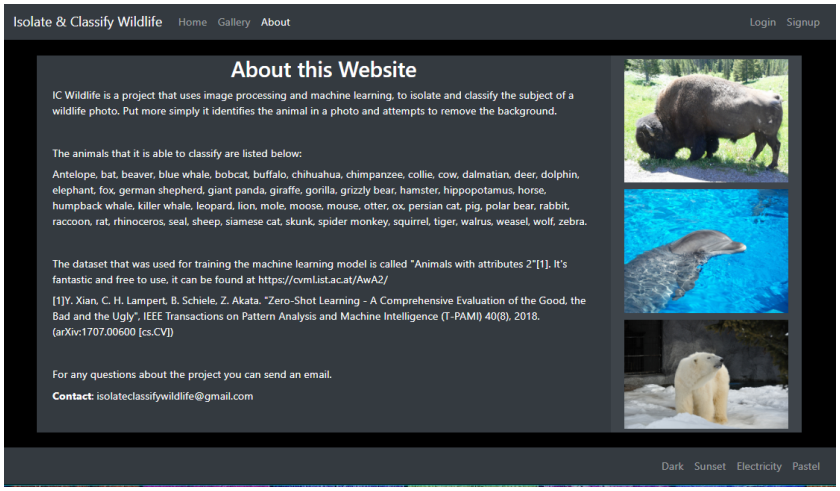

Figure 25: The about page

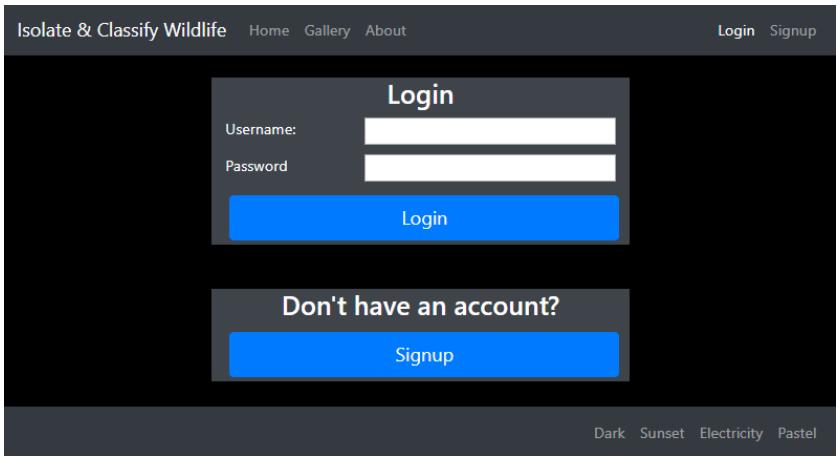

Figure 26: The login page

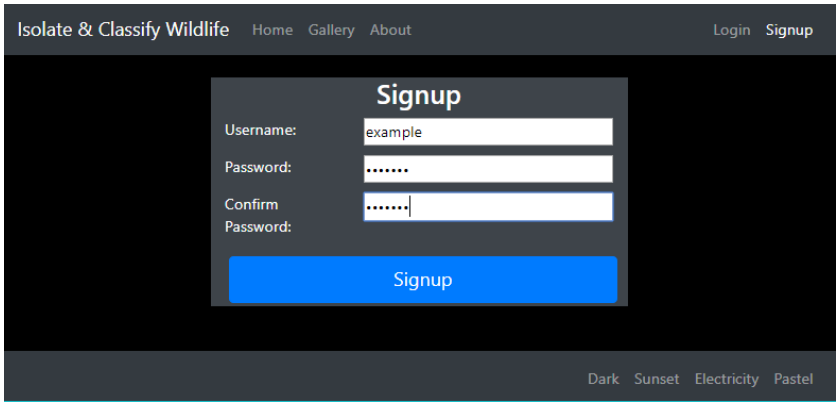

Figure 27: The sign up page

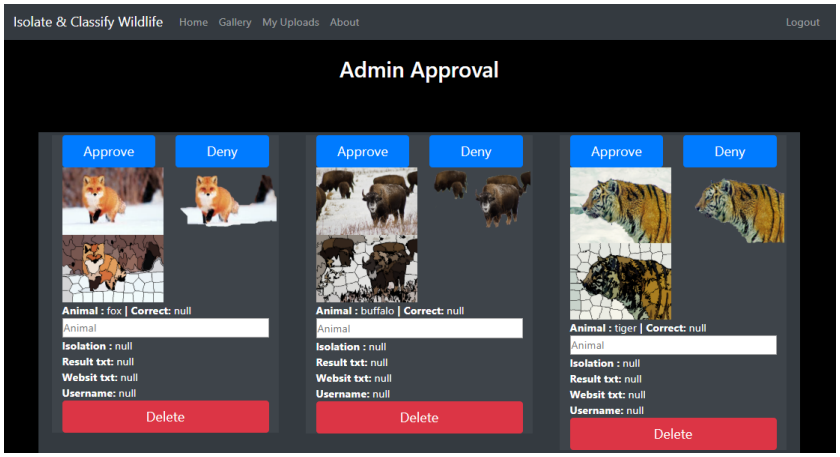

Figure 28: The admin approval page

4.3.2 Client Side Processing

All client side processing is performed with JavaScript. Each page in the site uses a JavaScript file unique to them for specific page processing while all of them also make use of two JavaScript files, assistMethods.js and themeMethods.js that contain shared common functionality,.

themeMethods.js
The theme methods file contains functionality for saving the currently active theme and reapplying it when the page is changed. Every time the page is changed there is

a check to see if a theme has been specified and if so the the pages theme css file is replaced with a reference to the active theme. Due to having identical class names all styling is neatly replaced.

assistMethods.js
The assist methods file holds some commonly used and absolutley vital functionality. The most important function is getJsonData as it sends out a formatted request to the web servers API and converts the response object into a javascript friendly dixtionary object. It also handles response errors and passes back info for error reporting.

The next two important functions are for handling JSON Web Token (JWT) objects. A JWT is recieved by the client upon successful signup or login however the client is unable to decode the token. The token is then included with all requests to the API that require user validation. The client being unable to read the token means there is no way of reverse engineering a token to create false credentials.(the server handles the encryption and description but that will be described later)

The remaining methods deal with page management such as hiding elements and displaying errors

clientHome.js
This file, as the name suggests, deals with the client side processing for the home page. This includes two API requests. All API requests follow a similar structure, collect neccesary data into a form object, adjust UI, send the form and authorisation header if required with a post request to the server, when response is recieved validate that all expected vlaues are present and if not display an error message and finish by again adjusting the user interface.

For example an upload request starts by putting the state of the user permissions for adding their image into the gallery and public dataset into a form object. The UI is then adjusted to indicate that it is waiting for response from the server. If a user is logged in a token is placed in the authorisation header of a post request and sent to the web server. Once a response is received the client ensures that paths for three images, an animal label and a submission id token are present. If they are not present or the API failed to respond an error message is displayed. On a valid response the UI is once again adjusted to show the returned images.

The feedback request collects the users feedback responses and alerts the user if the required fields have not been filled in. If valid the feedback along with the previous

submission token are sent with a post request. A thank you message is displayed to user if the feedback request is successful.

(write about the example images for background removal being displayed)

clientGallery.js
This file sends out requests to the API for returning gallery eligible images. The user can filter by animal label and image type. When a response with images is recieved the client clears the gallery of the previously present images and starts populating it again with the newly received images. This is achieved using a markup template where the label and source for the image is dynamically allocated.

clientLogin.js
The login JavaScript file contains the functionality for passing user name and password to the API and receiving a token in response. It checks that the username and password are valid non empty strings and informs the user of any issues. It displays an error if a username password combination is rejected by the server.

clientSignup.js
The signup file is extremely similar to the login, the main difference being the types of error messages that can be received. The client ensures that password and confirm password values match and will inform the user if the requested username already exists after a signup attempt.

clientAbout.js
The about file is an empty placeholder that is present to maintain the standard structure of the site and in case any future processing is added.

clientAdminapproval.js
The admin approval file has the functionality to request images that have not been reviewed by a moderator. It displays these in a similar fashion to the gallery by having a markup template except it include a few additional UI options. It has buttons for approving, denying and deleting a submission as well as a text box for replacing incorrect labels. All requests made to the server for admin functionality send a token with the user id of an admin. An error message is displayed if a request is rejected.

4.3.3 Android Application

The android application was created using the WebView object. This allows an app to natively load a web page. Since the web application was designed with scalability

in mind it easily adapts for the mobile screen. After adding the WebView as the main activity there were three functional additions that needed to be made.

The first two were minor changes to the android manifest file. A request for camera and file access was added so that users could upload images taken directly with the camera or images that are already stored on the phone. The second change was setting the configuration for when the screen changed orientation, so that the activity was not reloaded but the screen size was adapted for the new aspect ratio.

The final addition was file upload functionality that supported the majority of android versions. It is actually a very difficult and well known issue that the different methods for enabling file upload across different android versions are painful to implement. It is such a well known issue that a public GitHub project was made to assist developers with this functionality. Ghazi Khan[25] made the Os-FileUp project and described it as *"an Android WebView Project to help app developers understand how to upload and process image for any hybrid app"*. His sample code was used to enable file upload across different android versions for this projects android app.

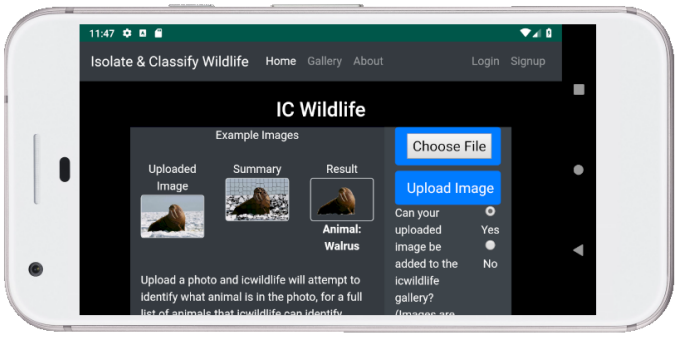

Figure 29: The home page on a horizontal phone

Figure 30: The home page on a vertical phone

Once the android app had full functionality the display name was changed to IC Wildlife and an app icon was made and imported into android studio.

Figure 31: IC Wildlife app icon variations

4.4 Logic Tier Development

The logic tier contains the code base for image classification and image isolation. These features are handled by the flask application that responds to GET and POST requests from the client application.

4.4.1 Flask Application

The below table shows the routes provided by the flask application.

Method	Route	Description
GET	/	Home page
GET	/gallery	Gallery page
GET	/myuploads	My uploads page
GET	/signup	Sign up page
GET	/login	Login page
GET	/adminapproval	Admin approval page
POST	/uploadrequest	Isolate and classify image
POST	/loginrequest	Login to account
POST	/signuprequest	Register for account
POST	/givefeedbackrequest	Give submission feedback
POST	/galleryrequest	Fetch gallery images
POST	/myuploadsRequest	Fetch my uploads
POST	/adminapprovalrequest	Fetch submissions for approval
POST	/setApprovalrequest	Set approval for submission
POST	/deleteSubmissionrequest	Delete a submission

The GET methods are all simple requests that return the HTML web page for each route. The post requests all follow a similar format. Request values are received and validation confirms that expected values are present. Inputs are formatted e.g. true/false string gets converted to Boolean. A default error message is prepared. If an authorisation token is in the header it gets decoded and the payload is validated. The remaining inputs are checked to ensure none are invalid. A database query or modification is attempted, if successful the appropriate response is prepared and converted to JSON before being returned.

The upload request is the central functionality to this project as it is where the isolation and classification occurs. The big difference with this request is that it needs to receive an image file, ensure it's not corrupted and create new images which includes constructing image paths.

OpenCv is used to validate that the file uploaded is a non-corrupt image file by seeing if it can successfully be converted to a NumPy array representation. A file name for the images is created by combining the current datetime with a random five character string. This file name is then appended to three paths, an original images folder, a summary images folder and an isolated images folder. These paths are passed into the appropriate functions so as images are created they can be saved to disk.

4.4.2 Image Classification

For the image classification it requires a similar setup as to what was used for training. Images need to be formatted into 200 by 200 resolutions adding padding where necessary, then converted from BGR to RGB and finally all values need to be normalised in the range 0 to 1. The same sequential model needs to be created and the checkpoint file is used to load the weights that were calculated during training.

Once the model and image are prepared a predictions array can be generated and to get a classification label the highest probability prediction is taken. In the web application the model can be kept as a global variable to save time rather than rebuilding the model and reloading the weights for every classification.

4.4.3 Image Isolation

The image isolation process scales images down so that its largest dimension is no more than 500 pixels. This does reduce accuracy a small amount however it reduces run time and file size significantly so for the sake of user experience the image is scaled down.

The initial image has a small amount of Gaussian blur applied and then superpixel segmentation occurs.

Figure 32: Example of superpixel segmentation

To make analysing the image easier these superpixels are used to create information contour objects (info contours) The superpixel boundary is stored as a contour, the size and centre are estimated using a bounding box around the contour.

The final attribute is the most relevant colour found within a superpixel. This is calculated by getting a list of all colours within a contour, then using k means clustering on this list which gives us the most relevant colour. This is used instead of finding the average colour because the average colour doesn't take into account that the RGB channels of a colour are related and for most images it results in a brown hue.

Figure 33: Example of contours with most relevant colour

These info contours are then filtered to remove contours that are below half the average size to improve relevancy of the remaining contours and reduce processing time for future steps. So the image can be easily traversed along the x and y axis a grid is overlaid on the image and contours are assigned to these grid squares based on the location of their centre points.

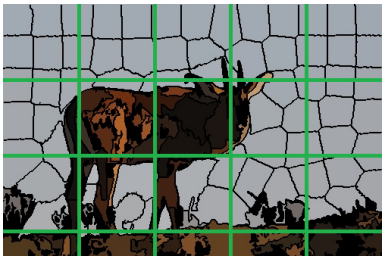

Figure 34: Image with overlaid grid

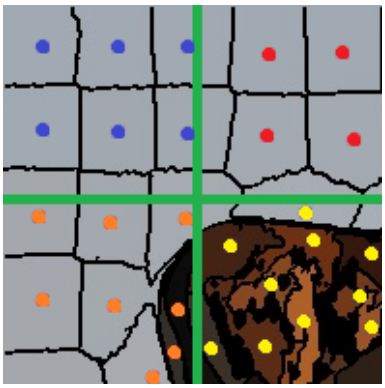

Figure 35: Contours being assigned to grid squares

The current goal is to get a bounding box around the subject of the photo. Using the grid made in the previous step the images contours can be traversed and analysed to find the first contour along a grid line that differs greatly from the contours by the edges. Here is an example of an image being traversed from left to right and finding the earliest definite edge of the images subject.

Figure 36: Contours with little difference from left to right

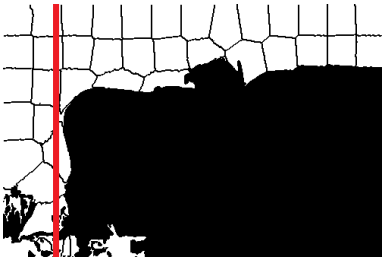

Figure 37: Found edge at earliest large difference

The above image demonstrates how it would look if it performed perfectly but for average use the path would rarely reach all the way to the opposite side of the image. Also to save on processing time, only the contours in a certain colour range with the largest aggregate size within the grid square are used.

The bounding box is being found to use with the GrabCut algorithm. GrabCut works better with a more reserved approach compared to an aggressive one as everything outside the bounding box is marked as definite background so large sections of the subject could be lost if any part of it is outside the bounding box.

The traversal process is repeated for the other three directions to find the entire bounding box. The bounding box is then used to apply the GrabCut algorithm to the original image. There were plans to use the identified background contours to further refine the isolation of the subject but an issue with false positives resulted in worse isolation the majority of the time.

The result image has the background segment converted to a transparent layer so that results can be more easily incorporated into other images.

4.5 Data Tier Development

4.5.1 MySQL Database

The entity relationship diagram for the database was shown in the design section of this report, these tables were created in a MySQL database for the data tier of this project. SQLAlchemy is an object relational mapper and it is used to interface with the database. The most useful feature of SQLAlchemy for this project is it's scoped sessions feature. These sessions can be distributed to users so they all have access to the database without any possible interference from one another. The sessions allow an individuals changes to be committed or rolled back safely.

Tables are implemented with a back reference feature of SQLAlchemy so that any foreign key relationships can be traversed both ways, for example submission.feedback and feedback.submission are both valid referencing methods despite the feedback table not directly containing a foreign key for the submission table. SQLAlchemy support some non-MySQL data types like Boolean and Text which it automatically converts to the equivalent supported type for use in the database. Boolean variables are heavily used in these tables and the application so the data type support improves ease of use.

```
class Submission(Base):
    __tablename__ = 'submission'

    id = Column(Integer, primary_key=True)
    animalLabel = Column(Text,nullable=False)
    firstLabel = Column(Text,nullable=False)
    permissionGallery = Column(Boolean,nullable=False)
    permissionDataset = Column(Boolean,nullable=False)
    modApproval = Column(Boolean,nullable=False)
    modReviewed = Column(Boolean,nullable=False)

    feedbackId = Column(Integer,ForeignKey('feedback.id'),nullable=True)
    userId = Column(Integer,ForeignKey('user.id'),nullable=True)

    feedback = relationship(Feedback, backref=backref("submission", uselist=False))
    user = relationship(User, backref="submission")
```

Figure 38: Code for creating the Submission table with SQLAlchemy

A typical insert involves creating the element, using SQLAlchemy's object relations to assign foreign key elements, attempting to commit the new object into the database and finally verifying that the commit was successful.

```
def insertGuestSubmission(originalPath,isolatePath,summaryPath,animalLabel,
    permissionGallery,permissionDataset):

    newSubmission = table.Submission(animalLabel=animalLabel,firstLabel=animalLabel,
        permissionGallery=permissionGallery, permissionDataset=permissionDataset,
        modApproval=False, modReviewed=False)

    newSubmission.images.append(table.Image(type='original',path=originalPath))
    newSubmission.images.append(table.Image(type='isolate',path=isolatePath))
    newSubmission.images.append(table.Image(type='summary',path=summaryPath))

    Session.add(newSubmission)
    success = safeCommit()

    newSubmissionId = None
    if success:
        newSubmissionId = newSubmission.id

    return newSubmissionId
```

Figure 39: Code for creating a new submission

The majority of queries are for simple functionality like checking if a user exists, and checks a single table for the first or all instances of a database object. More complex queries can be built by adding filters and/or joins dynamically. An example of this is the query for the gallery search where the filters to ensure only images with the valid permissions are selected is not executed straight away. Conditions are evaluated to see if additional filters need to be applied and the query is then executed.

```
def idUserExist(id):
    user = Session.query(table.User)\
    .filter(table.User.id==id).first()

    return user != None
```

Figure 40: Basic query to check if a single user exists

```
def filterGalleryImages(category,label):
    partialQuery = Session.query(table.Image)\
    .join(table.Image,table.Submission.images)\
    .filter(table.Submission.permissionGallery==True)\
    .filter(table.Submission.modApproval==True)

    if label != None:
        partialQuery = partialQuery.filter(table.Submission.animalLabel==label)

    if category != None:
        partialQuery = partialQuery.filter(table.Image.type==category)

    images = partialQuery.all()

    return images
```

Figure 41: Dynamic query for image search and filter

5 Testing and Evaluation

5.1 Introduction

This chapter discusses the testing and evaluation of the system. In this chapter we will using the term Testing to refer to our own appraisal of the system, and Evaluation to refer to the appraisal of the system by other people and other pre-existing metrics. In this chapters the Testing will include unit testing, API testing, and user testing while the Evaluation will include image isolation evaluation, image classification evaluation and usability evaluation.

5.2 System Testing

5.2.1 Unit Testing

A small amount of unit testing was performed during development. The tests were constructed to ensure functions responded correctly to expected input, invalid input and edge cases like extremely large and small inputs. The main objectives of these tests were to ensure that as other components changed the isolate and classify processes remained stable. The tests checked that these functions could process average, tiny and giant images. The functions error checking was also tested by passing in invalid variables, unsupported file types and corrupted images. These unit tests were run after every major system change once they were initially created.

TODO: Add more unit tests and show a unit test result screenshot

5.2.2 API Testing

The API is used for all interactions between the client and server so testing it was a necessity. The client application prevents certain types of input being used in API calls so an alternative method was needed to to test that all endpoints were stable. Postman[26] is a tool commonly used by developers for API testing, it provides the ability to easily format HTTP requests and save request templates making it easy to repeat tests.

Each endpoint was tested by passing valid and invalid query parameters. For example login attempts were made with correct user credentials and then attempted again using non existent credentials. Secure endpoints such as the one for deleting submissions were posted to with and without having a valid authorisation token. All these tests covered security, consistency and stability for the API. Covering these areas meant when testing the client application the API was guaranteed not to be the cause of any issues. Below is a list of tested requests and some example requests.

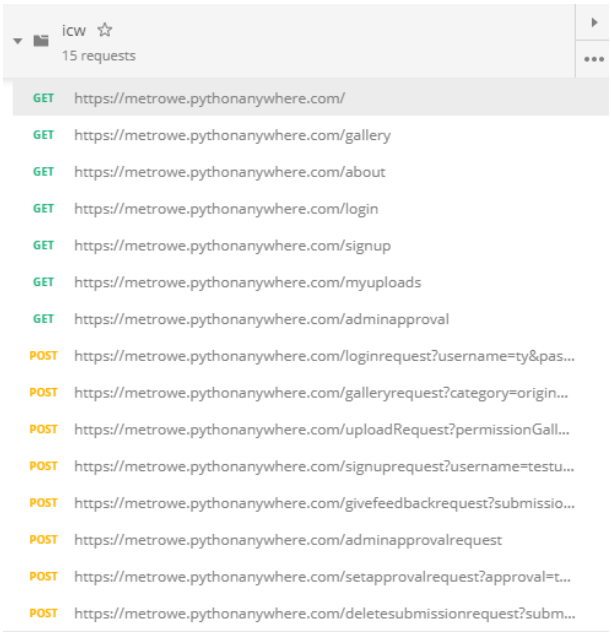

Figure 42: Postman list of requests

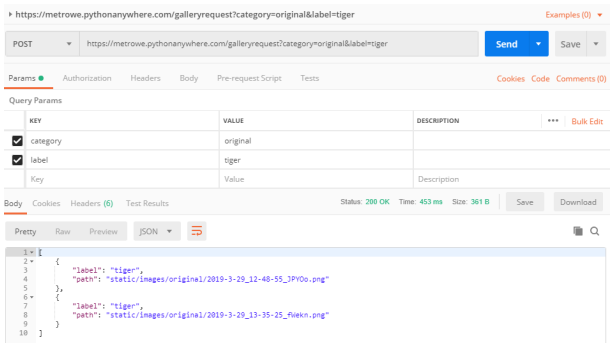

Figure 43: Postman gallery request

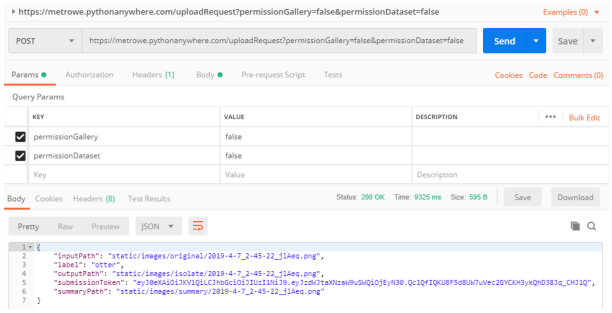

Figure 44: Postman upload request

5.2.3 User Testing

The project underwent two kinds of user testing, guided and unguided. Unguided testing occurred after the site was made public and posts were made on multiple social media platforms asking for people to visit the website. This resulted in over fifty submissions to the site and around twenty users filled out the user feedback form. The site does not request any user details so the demographic of the visitors is unknown. A small payment for additional system resources was made to the cloud provider before the site went public and the site did not experience any noticeable slowdown even when at its busiest.

The server logs and user responses were examined daily to see if users were experiencing any issues. This allowed some bugs to be identified. The most notable issue was when users attempted to access certain features of the application such as the gallery around five percent of the time the resources failed to load initially and the

page needed to be refreshed before the gallery contents were displayed. A user highlighted this error and it was found that in the server logs connection to the database was lost before the request could be fulfilled. It was soon identified that connections were going stale due to delays between requests. This issue was resolved quickly by increasing how often the database connection is refreshed.

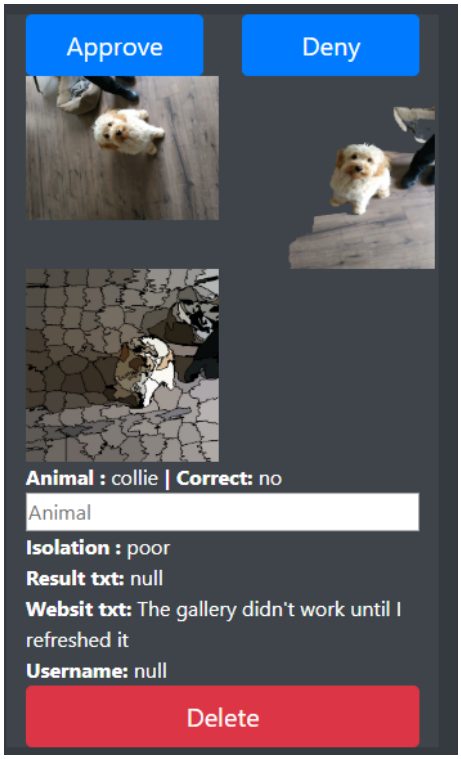

Figure 45: User feedback highlighting gallery issue

Users also gave feedback on aesthetic things such as text sizes and page layouts. Adjustments were made to accommodate many of these as a lot of them were nice visual improvements. TODO: Include section on adjusting input for files to include image links

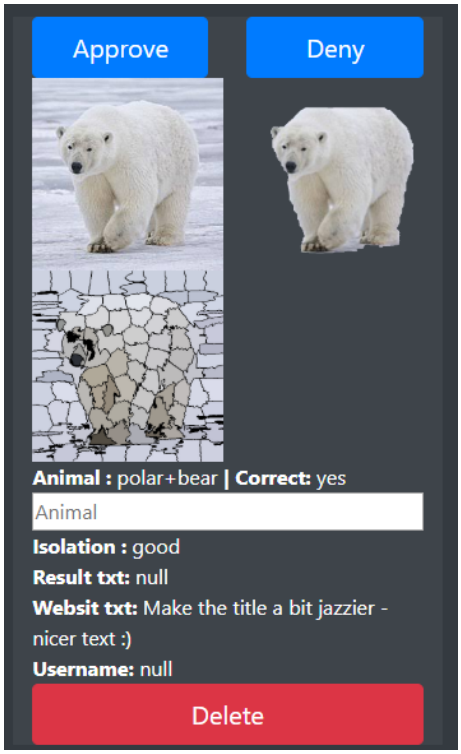

Figure 46: User feedback about aesthetics

Guided testing was performed by ten volunteers. These were mostly college aged men and woman but there was also three participants in the over forty age group. They were given access to the web application and a set of test cases to perform. This allowed features like registering an account to get a lot more use than it did during public testing. During this testing it was found that upload was prevented if no file was selected but no error message was being displayed. It was also found that the invalid or corrupt file error message did not disappear after a successful upload. These issues were promptly fixed.

Test #	Test Description	Result
01	Access website and it loads	Pass
02	All pages are available	Pass
03	User can upload image	Pass
04	Correct images are displayed after upload	Pass
05	User can leave feedback	Pass
06	Images are displayed in gallery	Pass
07	Gallery can be filtered by image type and animal species	Pass
08	Users can register an account	Pass
09	Users can login to an existing account	Pass
10	Logged in users can view and filter my uploads page	Pass
11	Logged in users can logout	Pass
12	Users can change current theme	Pass
13	Logged in admins can access the admin approval page	Pass
14	Admins can approve and deny submissions	Pass
15	Admins can delete submissions	Pass
16	Upload prevented without file selected	Pass
17	Upload prevented with invalid or corrupt file	Pass
18	Alert if user already exists in signup	Pass
19	Error message if blank username or password fields	Pass
20	Alert if incorrect username or password on login	Pass

5.3 System Evaluation

5.4 Image Isolation Evaluation

The image processing in this application in the simplest terms is based around deciding what pixels are part of the foreground. By using sample photos where the pixels of the foreground are already recorded, the effectiveness of the background removal by the application can be evaluated mathematically. The metrics that has been chosen is precision, recall and F-measure.

Precision can be thought of as how accurate the contents of the result set are. In this case high precision means that the majority of the pixels in the calculated foreground mask are actually part of the foreground. This means false positives reduce accuracy. The calculation is the number of true positives divided by the number of marked positives (true + false positives). At one hundred percent precision all selected pixels are actual foreground pixels.

Recall can be thought of as how covered the relevant elements are by the selection. High recall means that the majority of foreground pixels are contained within the calculated foreground mask. The calculation is the number of true positives divided by the number of relevant elements(true positives + false negatives). At one hundred percent recall all actual foreground pixels are contained within the selected pixels.

F-measure is a metric that takes into account both precision and recall to get the harmonic mean, this is more valuable as it requires good performance in both areas to get a high accuracy. Twenty images were randomly selected for evaluation. These images had foreground specified manually and also processed by the application to estimate foreground. Below is a table containing the results from calculating the f-measure for these images.

F-measure	0.7173	0.9149	0.8544	0.9794	0.5690
	0.8641	0.9502	0.8163	0.7862	0.6725
	0.8307	0.9322	0.7626	0.7847	0.7053
	0.8377	0.7248	0.8474	0.6841	0.8053

Figure 47: F-Measure for 20 test images

These results were totalled and used to calculate the average f-measure which is 0.802. Perfect precision and recall results in an f-measure of 1 which is the highest possible value. Considering this 0.802 is a very positive result. Due to the cautious nature of the processing there was usually more background left in rather than foreground being removed mistakenly. This lead to high recall in almost all cases and not incorrectly parts of the subject as background meant low f-measure results weren't terrible visually.

Ten volunteers were shown images with background removal of varying F-measures and asked to rate them as perfect, good, poor or failure. All volunteers rated the 0.9794 F-Measure image as perfect. Seven volunteers rated a 0.8163 image as good while three said it was poor. All volunteers agreed that a 0.6841 image was poor. One user marked a 0.5690 image as poor and the remainder marked it as a failure.

5.5 Image Classification Evaluation

Image classification is evaluated using the total accuracy for a specific test set of images. There is more nuance to this such as not being able to include any images

from the training set in the test set. They need to be unseen images because it is extremely easy to fit a machine learning model closely to a specific training set so it has almost 100% accuracy for these images but falsely classifies almost any other input, This is known as overfitting a model. To ensure accuracy for unseen data the model for this project was evaluated using k-fold cross-validation.

K-fold cross-validation starts by randomising the order of the dataset and then splitting it into K subsets of equal size. The model is then trained in K iterations each time setting a different subset as the test set and the remaining subsets as the training set. The accuracy is recorded for each of these iterations and the average accuracy is calculated.

Figure 48: Illustration of K-Fold Cross-Validation

For this project the dataset was split into five subsets for k-fold cross-validation and the results are displayed in the table below.

Iteration #	1	2	3	4	5
Accuracy	61.76%	62.44%	61.98%	62.10%	62.58%

Figure 49: Results of K-Fold Cross-Validation

The average accuracy for the cross-validation is 62.17% Excluding submissions that contained animals that the model was not trained to classify, of those 26 up-

loads 19 were correctly classified, giving a success rate of 73.08 percent, however the majority of these uploads had good contrast and not very noisy backgrounds.

The three largest influences on the systems accuracy have been the size of the dataset, the quality of images in the dataset and the number of species it is trained to identify. Initially the model was trained using the animal categories of the CIFAR-10 and CIFAR-100 datasets. The model dropped in accuracy when going from identifying the six CIFAR-10 animal categories to the thirty five animal categories in CIFAR-100. It increased in accuracy when using low resolution images from the Animals with Attributes 2 dataset due to the larger number of images per category. Finally there was a huge increase in accuracy when the images resolutions were upgraded from 50X50 to 200X200.

5.6 Usability Evaluation

The think-aloud protocol is commonly used to get user insight on application interactions. Users are observed while they use an application and are asked to say their thought process out loud as the site is interacted with. This method was performed by the same volunteers that took part in the testing process.

Thinking allowed highlighted some good design features of the site. The use of imagery and following industry standards allowed a lot of features to be quickly understood and easily interacted with. The example images for uploaded photos and results let most users understand what would occur after an image was uploaded without needing to read the text explanation. The sign up and login process had comparable success due to their similar visual layout to the same types of pages on commonly used websites.

The most appreciated feature of the gallery page was the text input, dropdown menu combination. This gave users a list of animals available to search that auto filtered as the user typed an animal name. The users had a number of dislikes about the styling, mainly issues with odd spacing around buttons and text inputs. They found the website feedback form question to be poorly worded and asked for it to be made more specific. It now asks specifically for users likes or dislikes about the website. A really helpful suggestion was to add examples of ratings for the image isolation feedback question.

6 Project Plan

6.1 Original Plan and Changes

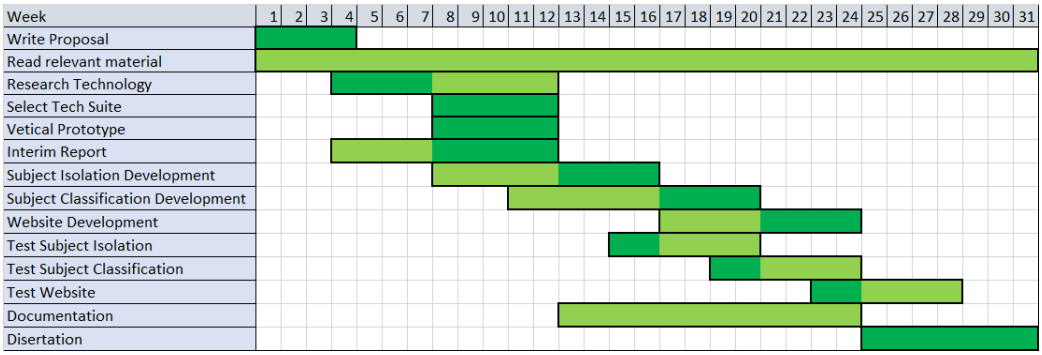

Figure 50: Original Gantt chart

The Gantt chart above shows the initial timeline for this project. Dark green indicates that for that week the associated task is high priority while light green is low priority. The three code deliverables are the subject isolation program, the subject classification program and the website.

TODO: REWORD section from interim, too much first person language

6.2 Key Differences

TODO: All Key

7 Conclusion and Future Work

7.1 Introduction

In this chapter key findings from this project will be presented. A number of conclusions will be reviewed starting with dataset collection and the application of superpixels. Following this, future prospects of the project will be discussed, including continuous training and user functionality for fixing isolation.

7.2 Conclusions

Datasets can be built or added to by offering functionality that requires the desired data. This project has three main features that attract user submissions, giving users a number of reasons to visit the site. Moderators can be given an interface to quickly review submissions which is far faster than seeking out images for inspect. The only limiting factor is the amount of user uploads, which could be increased if there was some kind of advertising budget.

The use of superpixels can greatly simplify the process of image analysis. Being able to breakdown an image consisting of hundreds of thousands of pixels into a few hundred areas of similar colour makes the elements of an image easier to understand and analyse. The superpixels can be evaluated on their colour content, shape, size, convexity and a number of other factors. Areas of interest can be found by examining superpixels with rough edges. Smaller than average superpixels can be screened to remove noise. In this project superpixels allowed images to be traversed and analysed swiftly as well as eliminating areas to be checked by filtering attributes.

Features with visual appeal are extremely useful when trying to attract users to a website. Social media posts that included the mosaic like summary images garnered far more attention and resulted in more visits to the website than those without images. The summary images stand out as they are distinctly different to normal wildlife photos. Attracting attention by advertising the more unique features of an application is a huge benefit as it gets more user testing and feedback for the website.

7.3 Future Work

A feature that was explored was allowing users to touch up and fix issues with background removal on images. Since the background removal is already implemented using the GrabCut algorithm, if the user was given an input option for drawing on the image to mark areas of incorrect foreground and background. This input would benefit from a more integrated process with the server which does not fit into the API call method of communication that is currently implemented. If given more time the framework could be adjusted to allow for this type of interaction as it would add greatly to the functionality of the project.

An interactive game had been considered as a feature for the web application however there was not enough time to implement it. The idea came from the "Who's that Pokemon" game from the Pokemon television show. Images where the background

removal was almost perfect would have their shape extracted so it could be used as a silhouette. A random selection of these silhouettes would be displayed to the user and they would need to guess what animal each silhouette was. It would have the option of a time limit for guessing so that it had some level of challenge. Having this feature would make users more likely to share the website with their friends and make the site appeal to a much wider audience.

Continuous training of the model seemed like a positive avenue when staring this project, however the use of a large dataset with relatively high resolution images meant continually training with the available resources was not feasible. However all that is required is a dedicated machine separate to the web server to retrain and transfer the model periodically. The current project has the infrastructure to integrate uploaded images with user permission into the current dataset. It even has the ability to relabel user uploads to efficiently add new categories to the dataset.

References

[1] Baxter R, Hastings N, Law A, Glass E. Object Recognition as Machine Translation Learning a Lexicon for a fixed Image Vocabulary; 2008.

[2] Belongie S, Malik J, Puzicha J. Shape matching and object recognition using shape contexts; 2008. Available from: `https://apps.dtic.mil/docs/citations/ADA640016`.

[3] Annalisa B, Francesca O, Alessandro V. Hausdorff Kernel for 3D Object Acquisition and Detection; 2002. Available from: `https://pdfs.semanticscholar.org/cf26/a593ceef822f666e1b472253614f946a255e.pdf`.

[4] Lowe D. Object Recognition from Local Scale-Invariant Features; 1979.

[5] Gao W, Yang L, Zhang X, Liu H. An improved Sobel edge detection; 2010.

[6] Canny J. A Computational Approach to Edge Detection; 1986.

[7] Kittler J. On the accuracy of the Sobel edge detector; 1983. Available from: `https://www.sciencedirect.com/science/article/pii/0262885683900069`.

[8] Russakovsky O, Deng J, Su H, Krause J, et al. ImageNet Large Scale Visual Recognition Challenge; 2015.

[9] Everingham M, L VG, Williams C, Winn J, Zisserman A. The pascal visual object classes (VOC) challenge; 2015.

[10] Rother C, Kolmogorov V, Blake A. "GrabCut": interactive foreground extraction using iterated graph cuts; 2004.

[11] Osma-Ruiz V, Godino-Llorente J, Sáenz-Lechón N, Gómez-Vilda P. An improved watershed algorithm based on efficient computation of shortest paths; 2007.

[12] Parkhi O, Vedaldi AV A Jawahar. Cats and Dogs; 2015.

[13] Pulli K, Baksheev A, Kornyakov K, Eruhimov V. Real-time computer vision with OpenCV; 2012. Available from: `http://dl.acm.org/citation.cfm?doid=2184319.2184337`.

[14] Amazon Web Services Inc. About AWS; 2018. Available from: `https://aws.amazon.com/about-aws/`.

[15] PythonAnywhere LLP. About PythonAnywhere; 2018. Available from: `https://blog.pythonanywhere.com/`.

[16] Digital Ocean LLC. About DigitalOcean; 2018. Available from: `https://www.digitalocean.com/about/`.

[17] Ronacher A. Opening the Flask; 2011. Available from: `http://mitsuhiko.pocoo.org/flask-pycon-2011.pdf`.

[18] Django Software Foundation. The Web framework for perfectionists with deadlines; 2005. Available from: `https://www.djangoproject.com/`.

[19] Oracle Corporation. About MySQL; 2018. Available from: `https://www.mysql.com/about/`.

[20] Jesse G. The Elements of User Experience; 2010. [Accessed 03 Jan. 2019]. Available from: `http://ptgmedia.pearsoncmg.com/images/9780321683687/samplepages/0321683684.pdf`.

[21] Michael VdB, Xavier B, Gemma R, Luc VG. SEEDS: Superpixels Extracted via Energy-Driven Sampling; 2012. Available from: `https://arxiv.org/pdf/1309.3848.pdf`.

[22] Stanford Vision Lab. About ImageNet; 2016. Available from: http://image-net.org/about-overview.

[23] Alex K, Vinod N, Geoffrey H. Learning Multiple Layers of Features from Tiny Images; 2009. Available from: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[24] Yongqin X, Christoph HL, Bernt S, Zeynep A. Zero-Shot Learning - A Comprehensive Evaluation of the Good, the Bad and the Ugly; 2018. Available from: https://arxiv.org/pdf/1707.00600.pdf.

[25] Ghazi K. OS-FileUp Project; 2015. Available from: https://github.com/mgks/Os-FileUp.

[26] Postman Inc. About Postman; 2018. Available from: https://www.getpostman.com/company.