

**School of Computer Science
University of Lincoln**

CMP2803M Team Software Engineering

Assessment 3

Project Summary Report



**UNIVERSITY OF
LINCOLN**

**Oliver Carlton, Daniel Dolby, Thomas Hardcastle, Jack
Marshall, James Peachey, Oakleigh Weekes**

**Submitted as part fulfilment of the requirement for
BSc/MComp Computer Science
May 2021**

Contents

Introduction	4
Objective	4
Aims	4
Datasets	4
Models	4
Team Members & Contribution.....	4
GitHub	4
Full Pipeline code	4
Website Code only	4
Website	4
Demonstration Video.....	4
Software Engineering.....	5
Implementation	7
Testing Strategy	14
Release	15
How was the article released?	15
Precautions for release	16
Evaluation	17
Product.....	17
Process	19
Performance	20
Group Work Conclusion	21
Oliver Carlton	21
Daniel Dolby.....	21
Thomas Hardcastle	21
Jack Marshall.....	22
James Peachey	22
Oakleigh Weekes.....	22
References	23
Appendix	25
Appendix A -Minutes	25
Appendix B- B Repos & Meeting Organisation	30
Appendix C- Report writing Allocations.....	31
Appendix D- Flower Dataset testing results	31
Appendix E- Training and testing model accuracy.....	32

Appendix F- Website tests 34

Appendix G – Product Review sample 35

Appendix H – Media Materials 36

Appendix I – Contribution Document 36

Introduction

Objective

To create a website which can import an uploaded greyscale image and return a coloured variant, using a deep learning model.

Aims

1. To locate a suitable image dataset and pre-process the images for machine learning.
2. To train an established TensorFlow neural network with the team's supplied dataset.
3. To test the dataset with new unseen images and examine the model.
4. To create a general model which will colourise any greyscale image to a suitable accuracy and standard.
5. To design a suitable website for users to query the model.
6. To integrate the trained model into a website using Flask API.
7. To iteratively improve the model by further training and testing.

Datasets

- 102 Category Flower Dataset (Nilsback, Zisserman 2008).
- Natural Images Dataset (Roy 2018).

Models

1. "Coloring-greyscale-images" (Wallner 2019).
2. "Auto Colorization of Gray Scale Image using CNN" (Poudel 2019).
3. "Python for microscopists – autoencoder colorize" (Bhattiprolu 2020).

Team Members & Contribution

Group Member Name	Contribution percentage
Oliver Carlton	32
Daniel Dolby	0
Thomas Hardcastle	11
Jack Marshall	25
James Peachey	0
Oakleigh Weekes	32

GitHub

Full Pipeline code

<https://github.com/oakleighw/TSEG33ColourisationProject>.

Website Code only

[MilkyCarton/colourisationapp \(github.com\)](https://github.com/MilkyCarton/colourisationapp)

Website

<http://colourisation.herokuapp.com/>

[Note: Due to the free version limitations, please reload the page as first-time loading gives an error]

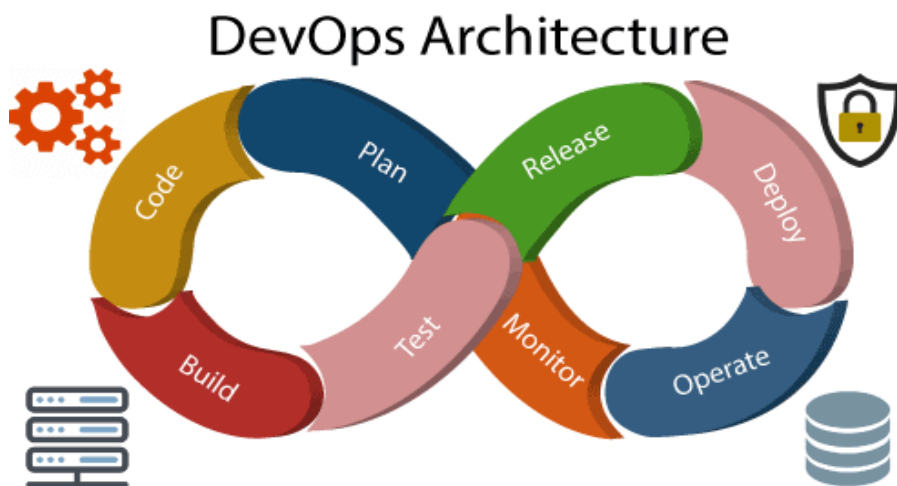
Demonstration Video

https://youtu.be/y_89wNxr0N8

Software Engineering

Meetings were structured biweekly, with a supervisor meeting occurring fortnightly. Supervisor meetings took place on Microsoft Teams, with more informal meetings happening on Discord, where mob programming took place. Minutes were taken regularly to summarise our progress ([Appendix A](#)).

We had planned to use DevOps as the software development framework for the project, while using the integrated Kanban boards with Azure DevOps. The board maintenance proved difficult, as our knowledge of machine learning was in its infancy. It was hard to plan out the required short-term tasks. DevOps proved to be a useful framework however, as the testing, design, development, and coordinator members had insight into the running code. As the team learned the machine learning pipeline and requirements, the solution was developed and tested as it went on, like keeping in touch with a client. Mob programming was heavily relied on, as we held a session every meeting to program the different models, combining our growing knowledge of machine learning to get a running solution. We used Azure DevOps for the team repository. This was because of the initial



Javapoint - devops-architecture

integration with Visual Studio Code, however we switched to GitHub for the release of the artefact. ([Appendix B](#))

Static analysis was used as we wanted to make sure that we could have a clear understanding of the way that the code worked, but most importantly so that we did not have any violations of coding standards. The static analysis was completed using the “Inspect Code” prompt in PyCharm (for those that used it instead of visual studio code) and selecting the scopes of the files that we wanted to analyse.

We did three types of analysis; these were control analysis, fault analysis and data analysis. The control analysis was to focus on the control flow within the calls to specific functions. The fault analysis was to be sure that there would be no active faults or failures in the model components. The data analysis was so that we could make sure that the defined data, i.e., JPEG, PNG is properly used whilst also making sure that the data objects were operating correctly.

We decided to use a modern testing technique, agile testing, allowing everyone to test the code alongside application development. Originally, Daniel was an assigned member for testing, but as he did not contribute, agile testing was a good contingency strategy. In doing so the development process was sped up as it allowed many people to work, test, and learn as they go, as well as fix bugs or issues during development.

Isolation methods included keeping separate models in different python notebooks and moving erroneous code into new notebook cells to verify previous code was working satisfactorily. The team used step-through debuggers, primarily using function breakpoints and inline breakpoints to see whether the program would respond correctly when integrating the model with the website in visual studio code. Breakpoints allowed the team to inspect broken code and prevent it from affecting the working program, allowing them to break the execution and debug.



Libraries & Dependencies

“TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.” (TensorFlow 2021, Website). It integrated well with Google Colab

as the library is under the same Google umbrella of open-source software services. It works compatibly with the Python language, which helps abstract complex machine learning algorithms to allow for higher-level machine learning model configuration.

TensorFlow notably includes Keras (2021), a deep learning library. Keras includes useful preprocessing techniques on training images, such as ImageDataGenerator, array_to_img, img_to_array and load_img. ImageDataGenerator was useful in creating new training data for the model through image contortion. Through Kera’s models and layers libraries, creation of the model structure was simplified to adding one-line sequential layers.

Scikit-Image (2020) is an image processing library. It was used to convert training images to grayscale, for them to be trained against their coloured counterparts.

Pillow (2021) Imaging Library was used to contort the images to a uniform shape as inputs into the defined model. OpenCV (2021) was attempted for the alpha, beta and full version 1 however there were conflicts with NumPy arrays and the converted images.

Bootstrap was used for the design of the webpage. “jQuery is a fast, small, and feature-rich JavaScript library” (jQuery 2021, Website). It is a more efficiently written JavaScript format. It allows JavaScript to be more accessible with a simpler API scheme.

Flask (2021) is known as a “micro web framework”, it is used for API creation. When an image is uploaded to the website, Flask queries the saved model algorithm and returns the output.



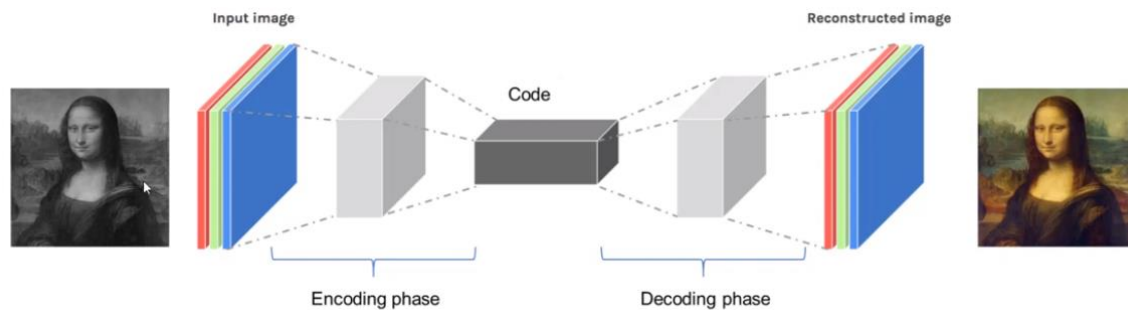
NumPy (2021) is a python package for scientific computing. It was used for the ability to create multi-dimensional arrays that could hold each image's pixel values.

Implementation

We have created a colourising solution that converts greyscale images into a colour variant. We have trained a model using our pre-processed dataset and hosted the model on a website to be queried using python flask API.

Initially the group started with limited machine learning knowledge. To build up a working model and artefact the group decided to create various implementations as alpha, beta, final models. The motive behind this approach was to give the group a working model as soon as possible, create more polished models and foundations to build upon in future versions.

Our model is an autoencoder which means we are training an image on another image and trying to reconstruct it.



90 – Application of Autoencoders – Image colorization (DigitalSreeni, 2020)

Alpha Development

Starting with nothing, we used references of other open-source models found on the internet as stated in the project proposal. Emil Wallner's model was used for the alpha and beta models. The group decided to use this model since it gave a basic understanding on the concepts of machine learning and colourisation. The code had to be altered to work on our local machines to fix certain errors we were acquiring through execution. The alpha implementation would train using 1 image and recolour it from black and white back to colour. This was a good starting point as we could develop further features such as mass training, image manipulation pre-processing and model saving.



Beta Development

Continuing from the success of the alpha model, the Beta model built upon the foundations of the Alpha model. A large dataset could be used in this version since an array of images would be trained rather than just one image. In addition to this, the dataset was artificially increased in size by using image manipulation. Images would be sheared, zoomed, rotated, and flipped. However, this implementation tested on the same images it trained. The success of this model was mixed since we incurred many errors.



Once errors were suppressed, the model began to train and our resultant images were unfortunately incorrect. The images would display as with a colour filter. The team decided to cut their losses and continue with the full implementation due to approaching deadlines.

Full Implementation

In this attempt, the team wanted to add an image classification neural network to our model. The classification neural network would detect what is in the image and colour it accordingly by sharing data with our model. Image classifiers work by factoring viewpoint variation, scale variation, illumination conditions, deformation of objects, background objects, occlusion, and intra-class variation. Instead of creating our own model for this to overwhelm our workforce, we decided to use a popular classification network called Inception ResNet v2.

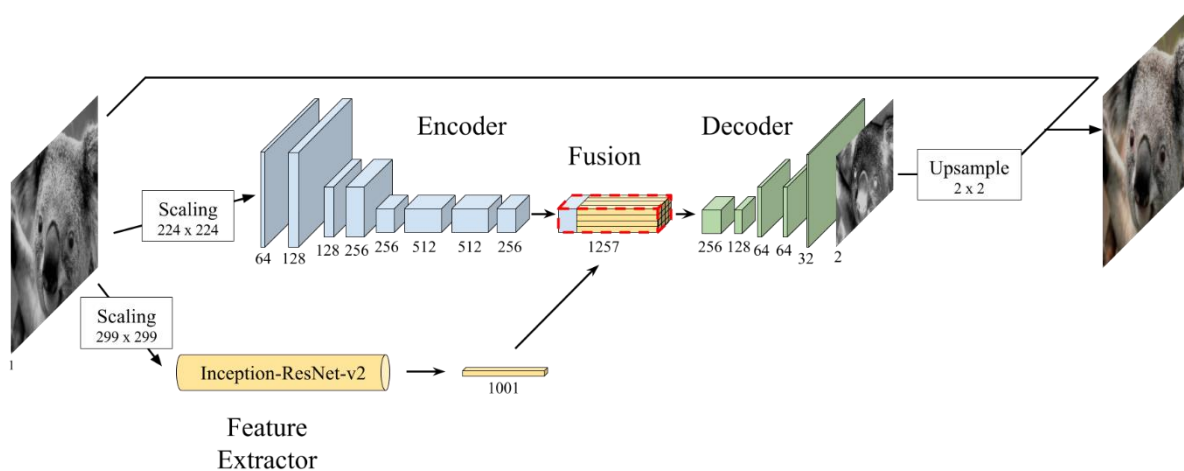


Figure 1: An overview of the fusion model layer architecture.

(Baldassarre et al, 2017, fig 1)

Through implementation of this method, we ran through many errors. After contacting a PhD student for guidance, he advised moving to a different model. This could have been caused through modules incorrectly working and code not pairing well due to deprecated libraries and different versions. We spent a couple of weeks trying to fix errors such as tensor errors, but for each one we fixed a new one would arise. As a result, we decided to go scale back on complexity and decide to return to these enhancing features in the future of the artefact.

AttributeError: in user code:

```
<ipython-input-43-f10fc778ed16>:5 create_inception_embedding *  
i = resize(i, (299, 299, 3), mode='constant')  
C:\Users\oakle\Anaconda3\lib\site-packages\skimage\transform\_warps.py:94 resize *  
if output_ndim > image.ndim:
```

AttributeError: 'Tensor' object has no attribute 'ndim'

```
-----  
AttributeError                                Traceback (most recent call last)  
<ipython-input-50-f10fc778ed16> in <module>  
    34 #Train model  
    35 model.compile(optimizer='rmsprop', loss='mse')  
--> 36 model.fit(image_a_b_gen(batch_size), epochs=1, steps_per_epoch=36)  
~\Anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py in fit(self, x, y, batch_size,  
epochs, verbose, callbacks, validation_split, validation_data, shuffle, class_weight, sample_weight,  
initial_epoch, steps_per_epoch, validation_steps, validation_batch_size, validation_freq, max_queue_size,  
workers, use_multiprocessing)  
    1048     training_utils.RespectCompiledTrainableState(self):  
    1049         # Creates a `tf.data.Dataset` and handles batch and epoch iteration.  
-> 1050         data_handler = data_adapter.DataHandler(  
    1051             x=x,  
    1052             y=y,
```

Implementation v1 error turning point.

Full Implementation v2

Full Implementation v2 is a reworked version of the Beta Development model. As the group decided to focus on a working artefact to build upon, the idea of a fusion layer was scrapped for potential future implementation. Full Implementation v2 worked on creating an efficient model that could be released as a 1.0 version. A resize algorithm was added to automatically resize images in each folder for training and colourisation, an example of the code is below.

```
#Resize Train  
count = 1  
for imagename in os.listdir('Dataset/TrainUncropped/'):   
    image = Image.open('Dataset/TrainUncropped/'+imagename)  
    image = image.resize((256,256))  
    image.save('Dataset/Train/image_'+str(count)+'.jpg')  
    count+=1
```

Once the model had been trained, on google colab or locally on the group's machines, the program would save the model as a .h5 and .json file for reuse later for the web release. In earlier versions of the product, the model would have to be consistently trained every time the program was run, resulting in slow progress of testing and training images.

Full Implementation v3 – Final model

Full Implementation v3 takes concepts from v2 but removes some image generator feature to speed up training since we had issues with machines being able to run the algorithms in timely fashions. The code was referenced from GitHub:

(Bhattiprolu S, (2021), 090a-autoencoder_colorize_V0.2.py,(2021), GitHub)

Since we focussed our training to be on the cloud, we did some adaptations to the code to make it more efficient for it. The team also moved to using the “natural images” dataset which had more of a variety of pictures including vehicles and animals as opposed to just flowers. A first step to this was

by downloading the dataset via Kaggle during the runtime of the Jupyter notebook. We achieved this using the code below, which would download the dataset by uploading our Kaggle API key. The dataset would then appear in the cloud machine's directory to be modified later.

```
from google.colab import files
files.upload()
!pip install -q kaggle
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!ls ~/.kaggle
!chmod 600 /root/.kaggle/kaggle.json
# Download natural dataset
# Prasun Roy (2018) Natural Images. Kaggle Dataset. Available at: https://www.kaggle.com/prasunroy/natural-images. Last Accessed: 03 May 2021
!kaggle datasets download -d prasunroy/natural-images -
p /content/gdrive/My\ Drive/kaggle/natural
```

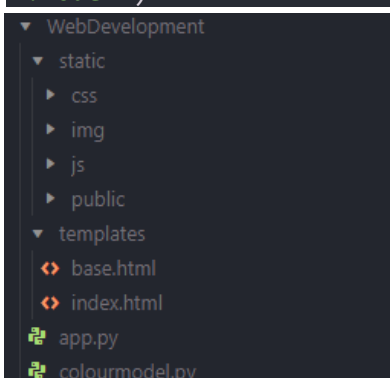
This code was the final version the group used to create a model. Improvements on the model and dataset could include hyper-parameter tuning as the project's life cycle continues, increasing the efficiency and accuracy of the conversions. Training with further large, varied datasets would also be of aid.

Web Development

The web development began early into the project's lifetime, on the creation of a website. The final model had to be adapted to be implemented into the final artefact. With the code from Full Implementation v2, we could export the model data from training into a .h5 and a .json file. A process like this allows the product to be very modular and easily updated with newer or more efficient model algorithm configurations throughout the lifetime of the product.

```
# Save model
model_json = model.to_json()
with open('/content/gdrive/My Drive/kaggle/natural/models/model.json', "w") as
    json_file:
        json_file.write(model_json)
model.save_weights("/content/gdrive/My Drive/kaggle/natural/models/model.h5")

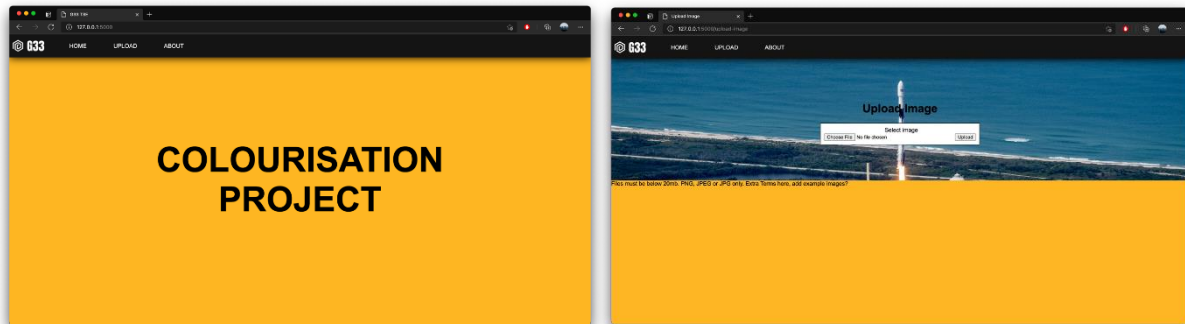
model.save('/content/gdrive/My Drive/kaggle/natural/models/colorize_autoencode
r.model')
```



This approach also allowed for a tidier backend for the website. The code was massively reduced since many modules and functions were not needed to be reimplemented; the model does not need to be re-trained every time an image is uploaded to be converted.

The structure of our web development is shown on the left. App.py is the main driver of the website and colourmodel.py is the function to convert images from greyscale to colour.

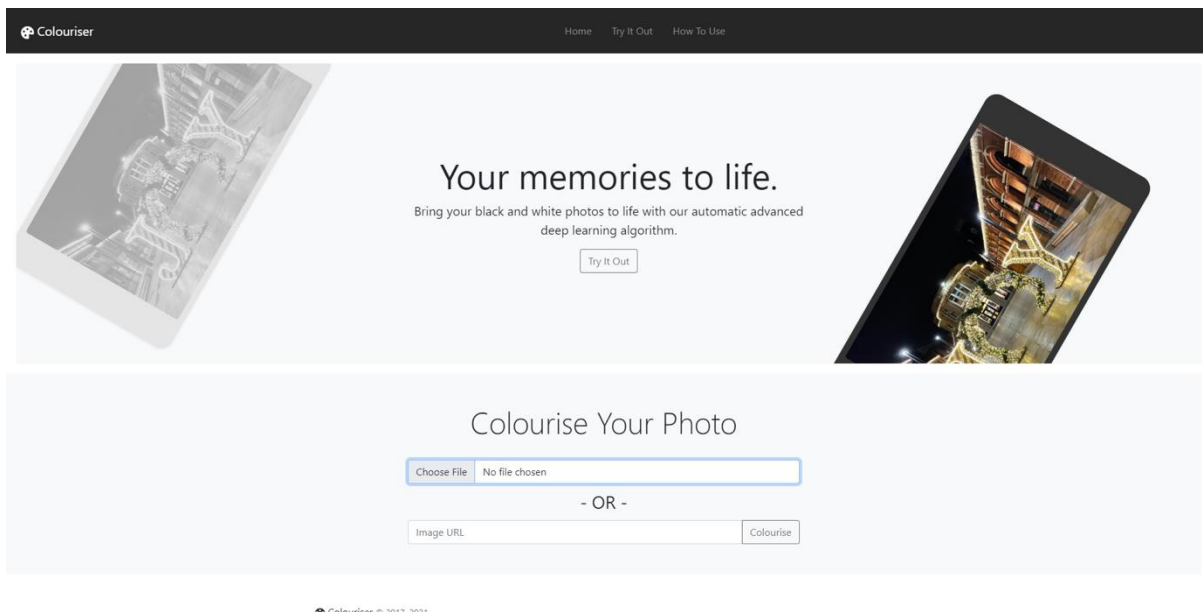
To initially link a front and back-end, a basic website prototype was created. Design elements of an upload button were already considered as a requirement as well as an image to display, converted after upload.



The website was created entirely from scratch through HTML and CSS and a small amount of JavaScript. As displayed in the images above, a navigation bar to specific pages on the site is present along with an upload button. We chose these features for the prototype as they are key design elements to a website format. These are features that must be tested early in development to ensure the core elements work before implementing more advanced features into future development. This will help avoid overcomplicating the artefact, making sure that basic features work before extending functionality.

Linking the front-end prototype website to the back end of the model was achieved using the python library Flask. Flask was the library of choice as it is a well-known library for creating web applications. This meant there would be lots of documentation to help the team learn syntax and application through videos and written tutorials. The app.py (the main flask app) imports colourmodel.py (the colouriser) to combine the front and back-end.

Through careful conversion and additions to the flask code, we transferred a working prototype into the final bootstrap website.



Challenges

There was a variation of challenges we encountered during the production of the artefact. Initially the main problem were the continuous conflicts with open-source code from GitHub and modules in Python. We overcame a lot of these by spending a few meetings finding suitable version numbers for the modules and libraries which worked. Anaconda was a potential viable solution to this, however inconsistencies between the group's python versions and general errors made this as a non-viable solution. Getting this to work for everyone was very inconsistent so in the end, especially for the web development, and so PyCharm was used. The reason for using PyCharm was that all group members had access to it, can create virtual environments and install the required modules which did not cause errors like Anaconda did.

Effectively making use of training the model was a challenge due to the compute power we had. For the first few models, where many batch images were being trained, the teams own local machines were used as consistent training could happen. However, as the group moved to a more rigid training method with more images and longer training times, the personal machines could not train the model in a fast-enough time without it timing out. The team moved to an online cloud platform, using Google Colab, referenced in the evaluation and testing section. In short Google Colab allowed the group to work on their machines whilst the notebooks online were being trained. Training still took a long time, which made it challenging to test a model effectively, especially if there were errors or the model was not working. There was not much to do to overcome this challenge but to spend a lot of time developing the models, attempt to reduce the error through careful consideration of hyper parameters and code before executing the training process. To help counter this, the team commenced development on the website whilst the models were training in the cloud.

Justification for a yearlong project

Taking upon the task of creating of a machine learning artefact without the group having much combined knowledge required a longer process to complete it. Much of the time of the project was dedicated to researching and solving problems that the team have not encountered before. Finding a way to effectively work together using online methods only was a challenge, especially when many team members did not know each other previously. Also, the team software engineering lecture material was all that was learned, with no practical experience in the field before.

As mentioned in the challenges section, training takes time and, in this case, the team created a vast range of models throughout the development of the product. This resulted in a lot of development time invested within training models which could have unpredictable outputs when testing hyper parameters.

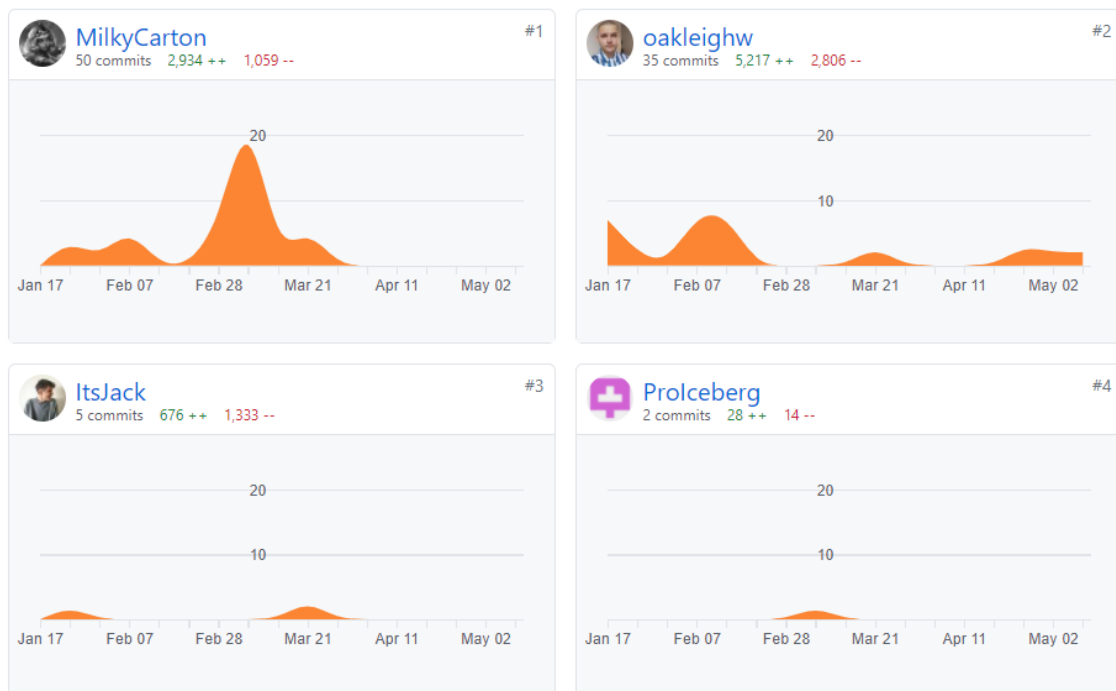
Worthwhile building?

As a research project, this artefact was very much worth building. The information learnt for machine learning, front-end and back-end development through being in a scenario to work on something that has not been done much before was invaluable. The team developed the skills to work together effectively with what was had which will improve the team software skillset for the future.

Media links

https://youtu.be/y_89wNxr0N8 -- Video no music

Skills Allocated in Development



These are the code contributions during the last main development phase. Oliver's (MilkyCarton's) versatility across coding languages and quick learning allowed him to contribute a majority in both sections of development, the front-end flask coding and the Machine Learning models. Jack's (ItsJack's) knowledge in web development has also assumed his role on developing and designing the front end and adding extra functionality. Oakleigh's (Oakleighbw's) previous knowledge in Machine Learning was used to help troubleshoot and produce effective models. As a team leader, Oakleigh also took a role of refactoring code for anyone who would wish to use our code in their products / other team members who may join in future development.

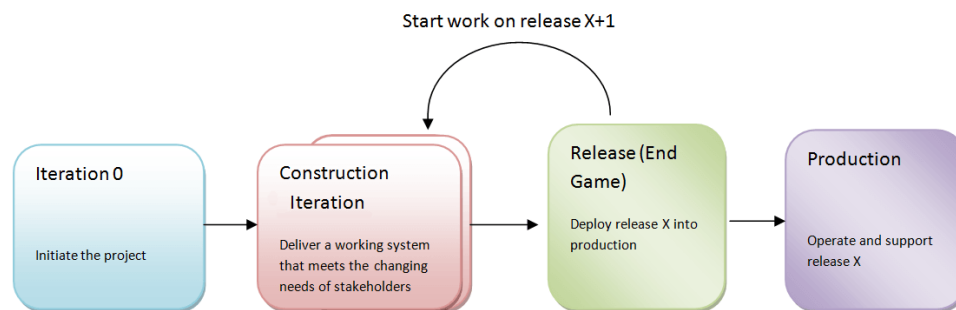
The flexibility of the group was very helpful to cover for sections that were not fully covered by members that were not present. Our group had a diverse skill range that allowed our objectives and aims to be met without much disruption to the project's creation.

Testing Strategy

Three key areas of our code were to be tested: code functionality, model behaviour and user interactivity.

An agile approach to testing was carried out through the modification of the models. Code was written in python notebooks, and so was automatically segmented into units that were easy to test for code functionality. White box testing was used for these individual blocks, through witnessing output of functions (e.g., `model.summary()` summarising model; `flow_from_directory` showing image number) or print statements (e.g., printing `X.shape` to view array size, or printing `randomData` to show a list of images chosen for training). White box testing worked well with these units as we “know how all the methods are written, what the data is supposed to look like, what the method signatures are, and what the return values and types should be” (Dooley 2017 p.253). Difficulty came initially due to misunderstanding of the code, as the model code was partially rewritten. The team had to break down each coding unit and abstract the meaning from it before they could white box test.

When Waller’s full implementation of the model was used, problems were faced with TensorFlow versioning. The team had previously augmented his alpha and beta model to work with the current version by using OpenCV to pre-process images. However, with his full implementation, this method had conflicts with the NumPy array structure. This delayed progress for a couple of weeks as there were little resources online to aid, and the way TensorFlow abstracted machine learning code made it easier for the team to modify, but more difficult to debug. After reaching out to a PhD student, they advised that we use a different model altogether, which led to using the second option, “Auto Colorization of Gray Scale Image using CNN” (Poudel 2019). This model code ran smoothly with the flower dataset without code errors.



Agile Testing Strategy

2 Guru99 (2021)

The model behaviour was iteratively tested during construction. Hyper parameters were tweaked to achieve the best visual results and accuracy metrics at every training run. These were documented and the model with the best results was brought forward for integration with the website.

Unfortunately, it was found that using a flower dataset overfitted to flower images, colourising most images with coloured centre and green surroundings ([Appendix D](#)). As the solution was generally working, solving this did not occur until production, where another model was trained with a different dataset.

Integration testing happened during web development and was mostly white box tested. A “dummy” website was created to test the input and output of the model after integration with the flask API. During construction we focused on both the confirmatory testing and the investigative

testing. We took the confirmatory testing on board as soon as the beginning of this phase of the project began, concentrating on the verification that the system would fulfil the needs and the intents of the metaphorical “client”. Our investigative testing consisted of issue resolution; these issues ranged from the stress testing of when we would implement something new, to integration testing where we would see how the main program would integrate with the API.

System Testing occurred after the website was complete, with front end, backend and model API working in unison. It could be seen as the final step in waterfall testing. This would usually be done by a separate testing organization, but due to lack of those resources, the team used written tests mimicking user interaction ([Appendix F](#)). After release, the model structure was changed, and a more general “natural images dataset” (Roy P, 2018) was used. As with the flower dataset, the model was trained and tested until the best accuracy metric with the best visual output was achieved (Please see [Appendix E](#)).

Testing Control:



3 car_108 original image (natural dataset)

4 car_108 greyscale image (natural dataset)

Release

How was the article released?

During the development process, the group had no choice but to self-host the release version of the project on their own devices. Whilst it was known to be a limitation for a public release at the time, it allowed for fast-paced progress and testing to be made. Hosting the project with no budget, especially due to the use of Python and a list of dependencies, is something that became apparent very quickly to be a challenge.

The method of finally releasing the artefact was using a Platform as a Service (PaaS) website called Heroku. Heroku is a way of hosting applications of different languages cheap, and free in our case due to a free trial.

Latest activity

[All Activity](#) ➔



oliver.carlton@live.co.uk: Deployed a6b5696e

Today at 11:03 AM · v6 · [Compare diff](#)



oliver.carlton@live.co.uk: Build succeeded

Today at 11:01 AM · [View build log](#)

To release this artefact, it was linked to a GitHub Repo, referenced at the start, [MilkyCarton/colourisationapp \(github.com\)](https://github.com/MilkyCarton/colourisationapp). Whenever a change is pushed to the repo, the website will rebuild, so careful consideration in the future when deploying changes will need to be thought about. Certain files were required to deploy the system onto Heroku, files such as requirement.txt, .slugignore, runtime.txt and a procfile. These files tell the systems on Heroku to download certain libraries/modules in addition to files to ignore and what version of python to run.

The product was released on 11/5/2021, short time was given however this was to reduce the amount of time the product is live for assessment to reduce costs. The product was ready to release 2 months prior on GitHub.

oakleighw changed to model test 6			d9762b9 2 days ago	History
..				
static	Added Web Development		2 months ago	
templates	Added Web Development		2 months ago	
app.py	Added certain directory creation if they don't already exist.		2 months ago	
colourmodel.py	Added Web Development		2 months ago	

MilkyCarton Added certain directory creation if they don't already exist.			Latest commit 4da1925 on 25 Mar	History
3 contributors				

Additions have been made since, as the website is modular and can be adapted as time goes on to fix bugs and enhance the experience. The 1.0 product was ready to release on 25th march, the date the web development folder was added to GitHub.

Why is the chosen platform suitable?

By hosting our project on a web server, it allows for exponential growth and usability of our work due to the accessibility of the internet. It also means that the end users are not required to download or install any software or dependencies themselves and allows the final product to be compatible with an extensive number of devices. The website is also completely responsive, so it is great to use on a variety of devices, no matter the screen size.

Heroku was chosen as the host specifically because of its free trial service to avoid additional costs. The process was relatively simple with some errors around versions of modules and python. Further research resolved the issues by adding the specific python versions and downgrading the TensorFlow module to TensorFlow CPU to reduce the slug size.

Precautions for release

As a group, it was decided that it was very important to have the website up and running as soon as possible to allow for testing during our alpha and beta testing stages of the project.

The appropriate licensing is important to consider. The group opted for an Apache 2.0 license as an open-source project. Assets in our final product used code from other projects under the same license.

[Apache License, Version 2.0](#)

An updated future model can be easily added by exchanging the model.h5 and model.json file to a new algorithm.

Evaluation

Product

The basic aims were met. The team have successfully completed the product requirements within the assignment timeframe. An artefact has been produced that colourises images using a website application.

The first TensorFlow model was outdated due to versioning in Emil Waller's solution. This was due to deprecated libraries and functions. Using different models and comparing them aided in forming a working solution, which ended with a third version. The model colourises models but does so relatively poorly. A model that would work with generally any image would need to be trained on many more images, with a high level of variety, which would need resources and time that the team did not have. "The larger variety of data points your data-set contains, the more complex a model you can use without overfitting" (Müller & Guido 2016, p.29). If the model colourised more accurately with a wider variety of images, further models for other image manipulation could be incorporated into the website, such as filter or saturation removal.



Documenting training runs aided the team in deciding the best hyperparameters for the model. This in turn led to a relatively more optimised colourisation algorithm. We learnt that changing the number of epochs, the percentage of validation images, the number of images and the batch size can affect model accuracy greatly. We found with 6890 images, a validation percentage of 30%, 350 epochs and a batch size of 48, we achieved the best accuracy of 88%. However, due to our resources the colourisation was not satisfactory due to the mentioned lack of resources. Hopefully if we were to carry out another machine learning project, we would have the required resources for training with more images.

Testing Car_0186

1. model.fit(x, y, validation_split=0.3, epochs=100, batch_size=32)
 \Rightarrow loss=0.0081 Accuracy: 0.6650, images=3500
2. split=0.3 epochs=500 batch_size=32 images=4500
 \Rightarrow loss: 3.3358e-04 accuracy: 0.9096
 \Rightarrow 0.57 validation acc
3. split=0.3 epochs=750 batch_size=32 Images=6000
 \Rightarrow loss=0.0172 accuracy=0.6280
 \Rightarrow 0.5919 val acc
4. split=0.3 epochs=650 batch_size=48 Images=6000
 \Rightarrow loss=0.0171 acc=0.6018 val acc=0.6309
5. Same Epoch=265
 \Rightarrow loss: 8.7571e-04 acc=0.8491 val acc=0.5683
6. Split=0.3 epochs=350 batch_size=48 Image=6890
 \Rightarrow loss: 9.4446e-04 acc=0.8773 val acc=0.5985

Google Colab was the best method for training models with a lack of on-premises resources. The purchased “Google Colab Pro” allowed for more RAM and a better GPU, meaning training could happen a lot faster and with more images. A longer runtime also meant that running code did not have to be supervised as much, meaning other project tasks could be completed in the meantime (Google 2021). Using Kaggle for the second dataset training was more efficient. With Kaggle API, the dataset can be loaded into and unzipped into a Google Colab notebook without having to download locally.

 Colab Pro

Get more from Colab

UPGRADE NOW

\$9.99/month

Recurring billing • Cancel anytime

[Restrictions apply, learn more here](#)

After some research it was found that nowadays over half of internet traffic comes from mobile (Clement 2021). The team realised it may be useful to have a phone app version of the product which makes the colourisation model more accessible. As an alternative the team could have made the website more mobile-friendly. A more responsive, flexible design would allow the application to reach more of the target audience through internet browsers (Schubert 2016).

It was difficult to obtain reviews as communication had to happen online due to COVID restrictions. Reviews consisted of family and friends of the team. The reviews ([Appendix G](#)) gave good insight into the product. They confirmed that the quality of image colourisation was not up to par for a commercial product, and so further training would be needed. It was found that no one would pay for such a service. If it were to be monetised, it would probably have to earn money through advertisement revenue. Originally, we believed that the product would be marketed at older people with black and white photos that wished to be brought to life in colour. However, reviews have

shown us that there is a target market in removing the modern black & white filters from images. This could be for aesthetics, with the original-coloured image being lost. Reviewers suggested the potential of the application for checking if a manipulated image that someone is using is an image of someone else i.e., they are “catfishing” using converted images (Cybersmile 2020). If this target-market were catered to in future, the conversion model could be integrated with reverse image search sites such as Google Reverse Image (Google 2021) for a full stolen-image inspection.

Latest Responses	Latest Responses
"colours b&w images"	"needs a better form of adding colour"
"it adds some colour to dull images"	"it needs a better colouring system"
"might help to spot if someone is using a picture of me but with a filter"	"better colouring, conversion from different filters"

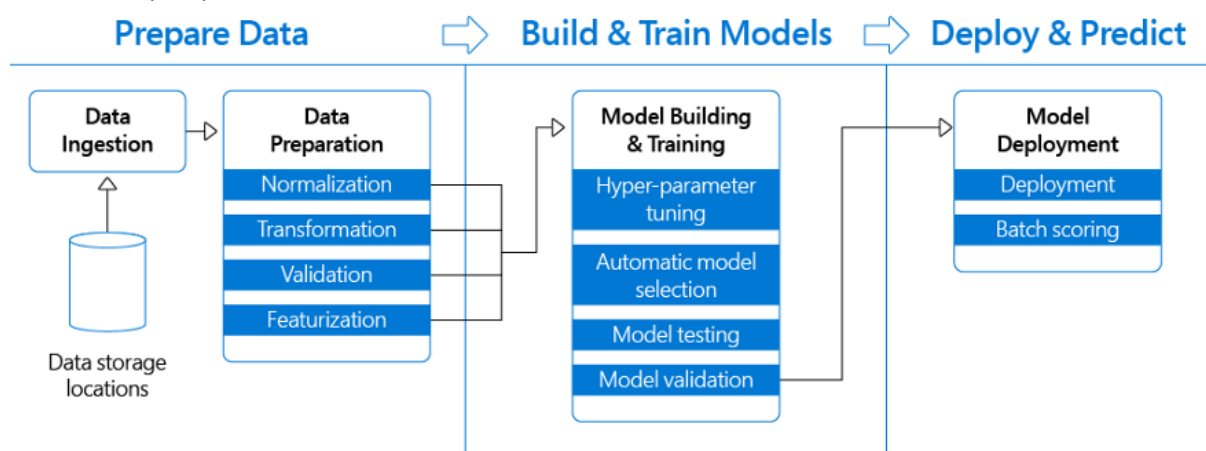
Process

There was a learning curve with machine learning as the module was not yet delivered to our year group. There was a delay with learning required materials to work on the solution, and it even prevented some members from wishing to contribute entirely. Time should be allocated to learning required materials for a project.

Meetings on discord were originally great for involvement. However, the meeting booking system “Seshbot” was very unreliable compared to the Microsoft teams meeting organisation system. In the future Microsoft Teams would need to be relied on more for communication.

Having a shared, branched version control system was difficult for this project domain. Mob programming was used for much of the project as it was difficult to divide work on coding individual models. In turn the team rotated work between each other. This led to each member having to download and configure the required dependencies when working from visual studio code. Google Colab became more practical.

Switching to a GitHub repository allowed all team members to contribute compared to the Azure DevOps limitations. Azure DevOps limits functionality for free users, and only five people maximum can work with the repository. University resources were not obtainable and so working in the cloud was a good option. Azure DevOps is better used for a development team that can pay for the full version. In the future, GitHub would have been a better version control system as it includes all the functionality required for free.



Lazzeri F (2019) DevOps with Machine Learning lifecycle

The team overall has learned more about the machine learning pipeline. The agile development lifecycle worked well for learning how to implement a machine learning model whilst simultaneously coding and integrating. They understand how a dataset needs to be pre-processed, trained, and tested iteratively, and how an API endpoint can be integrated with an application. This knowledge would make the team's project plans organised, such that better planning charts can be created, with more realistic task deadlines. PERT chart creation may benefit future projects, especially if the tools and resources used are not necessarily known. They provide a high-level overview of all the tasks involved, their dependencies, and their estimated completion dates, whilst Gantt charts are more useful when created whilst the project is underway (McAbee 2020).

Performance

The team has made sufficient progress with a lack of contribution from some members. The artefact was made within the timeframe of the assignment due to the diligence of working members. The artefact was released early and so was able to iteratively be improved. Keeping a contribution log ([Appendix I](#)) made the work ethic transparent to the team.

Contribution was difficult to measure. Attendance in earlier meetings was generally satisfactory, however around deadlines and nearing production attendance was poor. However, it was found that attendance was not a good measure of engagement with the work. Some members muted their microphones and did not contribute to discussion in meetings. Sharing work was difficult with a project that utilised one python notebook at a time. The lack of a team member who disappeared early into the project did not affect the team's overall performance due to concise contingency plans being established on project proposal.

Risk	Effect	Prevention	Contingency	Risk Level
Loss of data 1	Having to revert to earlier versions/saves in the project. Potentially having to restart the whole project. 3	Cloud storage, Frequent backups on each team members local machines.	Use data recovery methods.	3
Loss of team member 1	Losing a piece of workforce will result in slower progression and more stress on other workers. 3	N/A	Adjust the work schedule accordingly for the other team members.	3
Setback of team member 2	Losing a part of the workforce temporarily will result in slower progression and more stress on other workers for limited duration. 2	N/A	Resulting team members to take on extra work evenly, temporarily.	4

Training the model was difficult on our own systems. Before learning about Google Colab, visual studio code was used as an IDE to train the model on the python notebooks extension. However only one team member owned a computer that was able to be used in that capacity. The over reliance on the team member meant training was slower until we changed to using a cloud-based training approach.

Working from home was a huge challenge. Communication was difficult via the microphone, and directing conversation was particularly challenging. Enthusiasm was lacking due to personal circumstance caused by national lockdown. Encouragement was difficult as it was hard to reach members via online mediums. Team building casual exercises may help combat this and boost morale (Beauchamp et al 2017).

Creating short term deadlines may help motivating all team members. Purposely encouraging updates from individual team members at meetings may help participation. Keeping track of the Kanban boards with a dedicated member to maintain them would allow us to have a more visual representation of progress.

Group Work Conclusion

Our team was lucky in that the motivated members coped with taking over the work that some members did not participate in, with thanks to the contingency plan. The remaining members work ethic was satisfactory, and we had good lines of communication to work on the project in manageable pieces whilst working on other assignments. Team member roles ended up being flexible, to aid each other and to fill in for work that had not been completed. For those that usually complete assignments last minute it was a challenge to complete tasks on deadline, but they generally aided in the project due to positive encouragement. Our individual reflections surfaced our personal challenges.

Oliver Carlton

Beginning this project and knowing nothing technical about machine learning I knew it was going to be difficult taking upon a software development role. Researching through other people's similar methods was an effective way to grasp the concepts in addition to using a programming language already familiar to me. As a group member I feel I effectively communicated with the group coordinator Oakleigh and group members to plan certain objectives. The work I was allocated was handed in time consistently so I can be a reliable asset to the team. There were points where I would do more work than assigned, however that was due to a desire to research more and a lack of attendance and contribution from other members. I would have to do more to keep the project on track. A thing I could personally improve on is documenting my work more often and commenting on the work I have done. This would help the other members who work on it. I could ask other members to review it to get secondary opinions on the work to get more contribution from them.

Daniel Dolby

Daniel was assigned a testing role. He was asked to test functions every few meetings and to create unit tests and log results. Due to personal reasons, it was a challenge communicating with him and as such testing became a joint effort by the rest of the team.

Thomas Hardcastle

In the beginning of this project, I was given the role of head developer. I would like to think that I helped assume this role as best as I could, in assisting the software developers. I would hope that I helped the team ensure a delivery using Agile processes over the timespan, and worked effectively with Oliver, Oakleigh, and Jack to ensure we deliver on the agreed priorities. I had to take up the media creation section as there was a lack of contribution presence in that area. As a support role to the group, in the future I would like to be able to help the software developers out more with any questions or problems that they had, or perhaps I should have reached out to struggling group

members if they need it. If we had the chance for a second attempt, I would try to help provide a better structure so that all members could contribute at an even amount.

Jack Marshall

When choosing the topic of Machine Learning for this module, I was hoping to learn more about the functionality and methods behind the concept. After this group project, I believe I have done just that! Throughout this project, I helped in small amounts wherever I could. Whether it is suggesting ideas during a screen share session, working on the Python code, or contributing to the report, but my strongest contribution was undoubtedly the researching, designing, creating, and testing of the frontend website, and the backend connection to the website, using a variety of languages such as HTML, JS, CSS, Python and the Python module, Flask. I also came up with the idea to try and create a system to colourise photos, and I think we have done a brilliant job. I pride myself on being an enthusiastic, chatty person and whilst it does have its disadvantages, I believe I have helped our meetings to be productive but casual and kept morale high within the group. If we decided to continue this project for a real-world scenario, I think our project is something that could be expanded upon to satisfy the mass market.

James Peachey

James was not active in this project. He was reached out to by email and arrived late before the project proposal. He attended some meetings with little contribution. As he did not end up coding, he was encouraged to participate by being tasked media creation and website design. However, these were taken up by other members due to him not participating or attending many meetings.

Oakleigh Weekes

I had the group coordinator role and found it difficult in this project. I had partial knowledge of machine learning which did help guide some tasks and help those struggling. I think I should have been more authoritative near the start of the project, which may have given those who did not contribute a push to take part more. I found communicating with the non-participating members particularly difficult, especially all online. I did let them know when we needed them and tried to keep them in the loop by reaching out to them personally regularly. I tried to prevent micromanagement, but this led to participating members having to do the work that non-participating members did not do. Surprisingly, reaching out and delegating tasks did not help get some members to contribute, however attention could have been given to help members with personal COVID- related problems. The biggest lesson was learning when to ask for external help, as it steered our project when we hit a bug in our model that felt like a dead end.

References

Baldassarre F., González Morín D., Rodés-Guirao L. (2017) Deep Koalarization: Image Colorization using CNNs and Inception-Resnet-v2. p. 4.

Beauchamp MR, McEwan D, Waldhauser J (2017) Team building: conceptual, methodological, and applied considerations. *Current Opinion in Psychology*. Vol 16 p. 114-117.

Bhattiprolu S (2020) (Full Implementation V3 code base), *Python for Microscopists, and other image processing enthusiasts*, https://github.com/bnsreenu/python_for_microscopists/blob/master/090a-autoencoder_colorize_V0.2.py [Last Accessed 12 May 2021]

Bhattiprolu S, (2021), *python_for_microscopists/090a-autoencoder_colorize_V0.2.py*, (2021), GitHub Available at: https://github.com/bnsreenu/python_for_microscopists/blob/master/090a-autoencoder_colorize_V0.2.py [Last Accessed 06 May 2021].

Clement J (2021) Share of global mobile website traffic 2015-2021. Statista. Available at: <https://www.statista.com/statistics/277125/share-of-website-traffic-coming-from-mobile-devices/> . [Last Accessed 08 May 2021].

DigitalSreeni, (2020), 90 – Application of Autoencoders – Image colorization. Available at: <https://www.youtube.com/watch?v=EujccFRio7o> [Last Accessed 06 May 2021]

Dooley J.F. (2017) Unit Testing. In: *Software Development, Design and Coding: With Patterns, Debugging, Unit Testing, and Refactoring* (2nd ed). Apress Media, California. p. 253-269.

Google (2021) Colab Pro. Available at: <https://colab.research.google.com/signup>

Google (2021) Search with an Image on Google. Available at: <https://support.google.com/websearch/answer/1325808?co=GENIE.Platform%3DDesktop&hl=en>. [Last Accessed 08 May 2021].

Guru99 (2021) What is Agile Testing? Methodology, Process & Life Cycle. Available online: <https://www.guru99.com/agile-testing-a-beginner-s-guide.html> . [Last Accessed 06 May 2021].

JavaPoint (2021) DevOps Architecture. Available at: <https://www.javatpoint.com/devops-architecture>. [Last Accessed 11 May 2021].

Keras (2021) Available at: <https://keras.io/> [Last Accessed 19 April 2021].

Lazzari F (2019) How to accelerate DevOps with Machine Learning lifecycle management. Available at: <https://medium.com/microsoftazure/how-to-accelerate-devops-with-machine-learning-lifecycle-management-2ca4c86387a0> . [Last Accessed 11 May 2021].

McAbee (2020) PERT Charts vs. Gantt Charts — What Are the Differences? Available at: <https://www.wrike.com/blog/pert-vs-gantt-charts/> . [Last Accessed 08 May 2021].

Müller AC, Guido S (2016) Supervised Machine Learning. In: *Introduction to Machine Learning with Python: A Guide for Data Scientists* .(1st ed) . O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. p. 25-130

Nilsback ME, Zisserman A (2008) 102 Category Flower Dataset. Available at: <https://www.robots.ox.ac.uk/~vgg/data/flowers/102/index.html> . [Last Accessed 06 May 2021].

NumPy (2021) Available at <https://numpy.org/devdocs/user/whatisnumpy.html> [Last Accessed 22 April 2021].

OpenCV (2021) Available at: <https://opencv.org/> [Last Accessed 22 April 2021].

Flask (2021) Available at: <https://flask.palletsprojects.com/en/1.1.x/> [Last Accessed 22 April 2021].

Pillow (2021) Available at: <https://pillow.readthedocs.io/en/stable/> [Last Accessed 22 April 2021].

Poudel R (2019) Auto-Colorization-Of-GrayScale-Image. Available at: <https://github.com/rrupeshh/Auto-Colorization-Of-GrayScale-Image>. [Last Accessed 06 May 2021].

Roy P (2018) Natural Images dataset, Kaggle. Available at: <https://www.kaggle.com/prasunroy/natural-images/metadata> . [Last Accessed 06 May 2021].

Roy P (2018) *Natural Images*. Kaggle Dataset. Available at: <https://www.kaggle.com/prasunroy/natural-images> [Last Accessed 03 May 2021].

Schubert D (2016) Influence of Mobile-friendly Design to Search Results on Google Search. *Procedia - Social and Behavioral Sciences*. vol 220, p.424-433.

Scikit-image (2020) Available at: <https://scikit-image.org/> [Last Accessed 22 April 2021].

TensorFlow (2021) Available at: <https://www.tensorflow.org/> [Last Accessed 19 April 2021].

The Cybersmile Foundation (2020) Catfishing. Available at: <https://www.cybersmile.org/what-we-do/advice-help/catfishing> . [Last Accessed 08 May 2021].

Wallner E (2019) Alpha, Beta, Full Implementation V1 (Deprecated) *Colourisation code* <https://github.com/emilwallner/Coloring-greyscale-images>

Appendix

Appendix A -Minutes

Week 16/11/20 -Brainstorming project Ideas and software Tools

Minutes

- Assignment report to be carried out using a shared OneDrive word document.
- MS Teams to be used as a formal communication platform and discord for informal meetings.
- Azure DevOps to be used as a form of version control; implements SDF.
- Using Visual studio Code as common IDE amongst dev team.
- Project Idea: Recolouring B&W (Black and white) images.
- Use of CNN (Convolutional Neural Networks in Deep Learning).
- Keras /TensorFlow Python libraries.
- The need to cover lecture material in the project proposal.

Agenda

- Research Papers for picture formatting with DNNs (Deep Neural networks).
- Marketing Domain?
- Discussion on Ethics.
- Discussion on chosen SDF (Software development life cycle).
- James Peachey Group member update.
- Assignment Report/Team split?
- Discussion on all Python tools needed for project.

Week 1 (23/11/20) – Choosing dataset and working out greyscale conversion methods.

Minutes

- Title: Colourizing Black & White Images of locations.
- 'Greyscaling' a Coloured dataset for training using Python or Photoshop batch convert?
- Testing on 1950s black & white images?
- Google Docs Report set up.
- Looked at previous projects that use convolutional neural networks.
- Ethics – EA1 Ethics code – images of locations (not humans/animals)
- James Peachey has communicated with the group.
- Using Google colab (for GPU).
- SDF – Kanban boards and Gantt.
- QA – testing using smoke box.

Agenda

- Photoshop conversion/ Matplotlib greyscale conversion / Caffe pretrained model?
- Report writing

Week 18/01/21 – Using flower dataset, Working with Wallner's Alpha Model, deciding coding dynamic.

Minutes

- Using Oxford flower dataset to train the model.
- Alpha bot now works with our input; produces a recoloured image after training on the same picture from our dataset.
- OpenCV library was used to convert image into a 400x400 format to input into model.
- Visual Studio Code is now preferred over Google Colab as it can be integrated better with Azure DevOps.
- Installing Visual studio Code TensorFlow dependencies on different members' PCs proved difficult. Google Colab may be used by the members that cannot run the code on their native machines.
- Import code had to be changed for Alpha bot as the library names have changed since Emir's code implementation.
- Mob programming is preferred weekly as the project works on a small amount of code; thereby benefiting from the collaborative approach.
- Point allocation for end of project needs to be consistently monitored.

Agenda

- Configure beta model to work with the new dataset.

Week 01/02/21 – Working out the best way of training the model.

Minutes

- Storing the dataset locally will alleviate problems working on/off with it on the cloud.
- School resources may need to be used if our systems cannot feasibly train the model due to lack of resources e.g., RAM.
- We need to compare different team members' PC Specifications to see which would train and run the model efficiently.
- We need to actively follow our software development life cycle plan.
- We need to break up the code given to identify which parts do what – and comment on them for our final report.
- We can test the code with less images for debugging purposes.
- Contorting the image is a good idea to create additional for training or testing.
- Once the model is working, parameters can be tuned for a better result.
- The Gantt chart can be re-drafted to better reflect the time spent on different areas of the project.
- Training time will be a part of the report; the time training the model could take a day or more.

Agenda

- Work with the beta model to get it to work on our local machine(s) if possible.

Week 08/02/21 – Working with Wallner's Beta model, Contacting the University about training resources.

Minutes

- Contacted the school for use of computing resources for training the model, awaiting response.
- Using the Spiral software development life cycle
- Repo directories have been reorganised.
- Beta Model code now runs.
- Beta Model trained using a small dataset of 260 images produces blue pictures.
- Full Model is now implemented within the repository.
- May need to use other resources to train model with more images; we run out of RAM even using Google Colab.
- Google Colab can be used for testing code by mounting the dataset from Google drive. This is good for those with lower end PCs to test code.
- The Marketing & design team has been established for media creation and to encourage participation – Jack, James and Daniel.
- Talks have been made about implementing a simple web app for consumer use of the future trained model.
- Meetings via Discord have been established for every Monday at 3pm and Wednesday at 10am. These may be subject to change for any personal reasons.

Agenda

- The design team will draft up a marketing model and web page design.
- The full model code needs to run with small amounts of data for debugging.
- The beta model needs to be trained on a larger dataset with tuned hyper-parameters for a better output.
- Gantt chart needs to be re-established to reflect different deadlines.
- Talks need to be had regarding possible further implementation of the solution.

Week 24/02/21 -Problems with Wallner's full implementation and TensorFlow library versions

Minutes

- There has been no response from IT resources at the University yet, we may contact them via MS Teams. This is for training resources.
- We have hit a bug in the full implementation of the project, and we have spent days trying to find the problem. We have had no luck yet -"AttributeError: 'Tensor' object has no attribute 'ndim'". This Error occurs at the model.fit training line of code.
- The bug has had an impact on our meetings and how much work gets done regarding the project.
- Oliver has found a similar model which could possibly be used instead, but more research into it is needed.
- We are presuming the error is due to a mismatch between NumPy arrays and tensors, but we are struggling to find the source.
- We have contacted our supervisor for aid, and they are providing us with resources to help.
- Jack has set up the template for the webpage which will act as an interface to using the model to colourize greyscale images.

- A contribution monitoring process is being worked on by Oakleigh to calculate marking points for members at the end of the project.

Agenda

- Once the bug has been fixed, we need to find a way of optimizing the beta and full version such that the model colourises to a certain standard.
- We then need to integrate the model into the web app using flask to create an api to query the model.
- More research is needed with regards to flask.
- The new design team needs to begin working on a design for media materials to accompany the solution.

Week 01/03/21 – Pre-processing image sizes, finding a code model, contribution document creation, Flask API introduction

Minutes

- IT resources are unavailable to train our model, however Oliver Carlton has had success with training the new model in the Google Collab environment.
- A Pillow solution has been created to pre-process images into a uniform shape before training. This replaces our OpenCV method, which may have caused the previous errors.
- Oakleigh has summarised and distributed a Flask tutorial to aid the web design team to format the model querying API.
- A contribution monitoring ongoing document has been created by Oakleigh and has been added to the Teams Group. This will hopefully address lack of member involvement.
- We have discovered that TensorFlow versions can affect code quite drastically. Functions are continuously added and deprecated due to the open-source nature of the library. This news will be taken into our future projects.
- A PhD student guided us where to look for other colourisation models that use auto-encoding.

Agenda

- The .h5 model needs to be saved after training to be added to the API code.
- The final model needs to be optimised to output the best resulting-coloured images.
- The Flask API solution needs to be developed and tested (using the loopback IP address).
- The web team need to integrate the API with the front-end webpage.
- Discussion needs to be made about media materials to accompany the colourisation project.

Week 08/03/21 – New training model working, Working flask back-back end.

Minutes

- Oliver has managed to integrate the .h5 model with the flask API framework.
- Website front end is complete, needs to be merged with the flask api and a successful photo upload system needs to be created.
- Thomas, Daniel and James are assigned to Promotional Media regarding the solution.
- Oakleigh had to leave the Monday meeting early due to Covid related personal problems.
- Jack did not attend the Wednesday meeting which led to the front and back end of the website so far not being merged, however he is working on website front end design.

Agenda

- The Media team need to start thinking about ideas for a promotional video, which will be created after the website is complete.
- The results of the colourisation algorithm can be improved, epochs and number of pictures will be changed to experiment for better output.
- The front end and back end for the website need to be merged.
- The code needs improvements with regards to comments.
- The code should be moved to GitHub in full such that other people can view the project. Azure DevOps has a hard limit to viewers in the free version.
- Test the model with a small number of images at a time.

Week 22/03/21 – successful integration of website front end with back end, version 1 colourisation program released.

Minutes

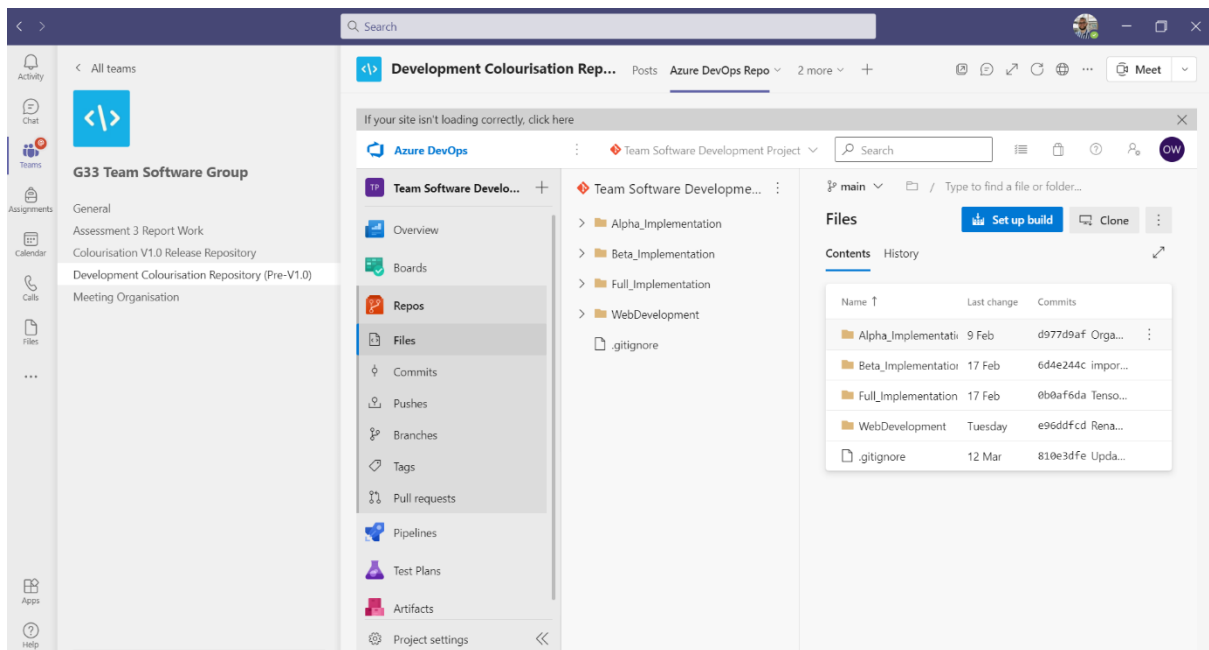
- The website front end has been successfully integrated with the back end, with the flask API model.h5 file being queried successfully.
- The model works well with converting greyscale flower images but not images of other themes. Further training of the model will need to be carried out.
- The model training code and web code has been moved to a public GitHub repository, for easy access. It also allows for more people to work on the project.
- The artefact is now released as version 1.0.

Agenda

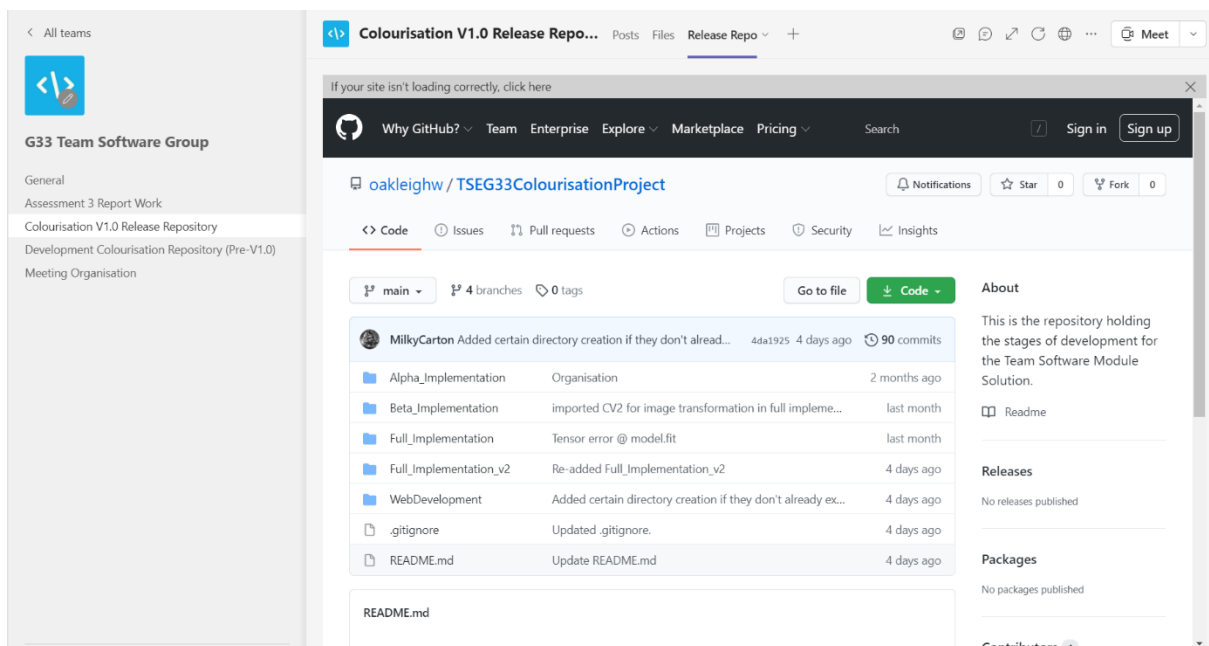
- The Media team need to create the promotional video on how to use the artefact; how it can colorize uploaded images.
- Daniel needs to create and document unit test cases for the model code and website.
- Team members are to start working on their assigned sections of the Assignment Report.
- A readme file needs to be created to explain to a future developer how to use the code in our repository.
- The code could use more comments for readability and ease to update in the future.
- A new dataset needs to be found and trained to improve the model file.
- It is possible to show the uploaded picture next to the colourised version after uploading. This can be worked on.

Appendix B- B Repos & Meeting Organisation

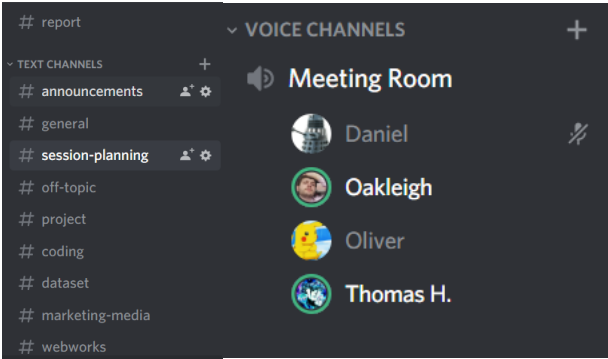
Teams environment with pre-release Azure DevOps repo integration.



Teams environment with post-release GitHub repo integration.



Discord Informal Meetings & Channels



Appendix C- Report writing Allocations.

HEAD: Responsible for section

Participant: Help with another section

These initially were the assigned parts of the report; however, the planning strategy did not work as some members did not participate.


Name	Software		Testing	Release	Evaluation	Appendix/Links/References	Media Works (Separate)
	Eng	Implementation					
Oakleigh	Participant	Participant				Head	
Oliver	Participant	Head					
Jack				Head	Participant		Participant
Daniel			Head				Participant
James			Participant	Participant			Head
Thomas	Participant				Head		Participant



Appendix D- Flower Dataset testing results







Appendix E- Training and testing model accuracy

Training Iteration & Visual Result	Dataset Size	Validation split	Batch Size	Epochs	Accuracy	Loss	Validation Accuracy
1. 	3500	0.3	32	100	0.6650	0.0081	-----

2.		4500	0.3	32	500	0.9096	3.3358e-04	0.57
3.		6000	0.3	32	750	0.6280	0.0172	0.5919
4.		6000	0.3	48	650	0.6018	0.0171	0.6309

5.		6000	0.3	48	265	0.8491	8.7571e-04	0.5683
6.		6890	0.3	48	350	0.8773	9.44446e-04	0.5983

Appendix F- Website tests

All elements of the webpage load successfully	PASS
"Choose File" button prompts image upload	PASS
Colourisation occurs after Image upload	PASS
Colourisation occurs on entering image URL	PASS
The "Home" button brings the user to the top of the page	PASS
"Try it out" button links to usage instructions	FAIL

Appendix G – Product Review sample

1. How often do you use image manipulation applications?

[More Details](#)

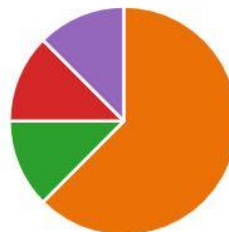
Weekly	1
Monthly	3
Yearly	1
Never	3



2. How useful do you find our product?

[More Details](#)

Extremely useful	0
Somewhat useful	5
Neutral	1
Somewhat not useful	1
Extremely not useful	1



3. What problem does this product solve?

[More Details](#)

8
Responses

Latest Responses
"colours b&w images"
"it adds some colour to dull images"
"might help to spot if someone is using a picture of me but with a filter"

4. What features would you include to help improve the product?

[More Details](#)

8
Responses

Latest Responses
"needs a better form of adding colour"
"it needs a better colouring system"
"better colouring, conversion from different filters"

5. How would you rate the quality of the greyscale to colour conversion?

[More Details](#)

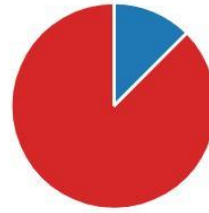
8
Responses

3.13
Average Number

6. In your opinion who is this product aimed for?

[More Details](#)

Younger people	1
Middle-aged people	0
Older People	0
General use	7
Other	0



7. How easy is it to use our product?

[More Details](#)

8

Responses

8.75

Average Number

8. How likely would you recommend this product to others?

[More Details](#)

8

Responses

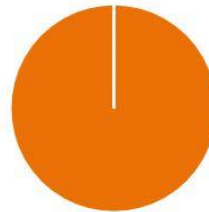
2.75

Average Number

9. Would you pay to use this service?

[More Details](#)

Yes	0
No	8
Maybe	0



Appendix H – Media Materials

https://youtu.be/y_89wNxr0N8

Appendix I – Contribution Document

Backlog

Oakleigh

- Created Teams Group and integrated Azure DevOps and the Repo
- Planned and scheduled weekly discord meetings and supervisor meetings.
- Took part in coding the alpha model.
- Took part in coding the beta model.
- Took part in debugging full implementation V1.
- Took part in discussing crucial steps for direction for the project.
- Emailed the University about resources
- Emailed a PhD student about acquiring help with TensorFlow debugging.
- Takes minutes.

CMP2803M Team Software Engineering Assessment 3

- Supplied examples of TensorFlow models.
- Oliver
- Records attendance for meetings.
 - Took part in discussing crucial steps for direction for the project.
 - Took part in coding the alpha model.
 - Took part in coding the beta model.
 - Took part in debugging and coding full implementation v1.

James

Daniel

Thomas

Jack

-
- Created Azure DevOps environment.
- Supplied machine learning educational resources.
- Provided support on use of Pillow library.
- Came up with recolouring idea.
- Marked up a webpage design for the application.
- Took part in coding the alpha model.
- Took part in coding the beta model.
- Took part in debugging full implementation v1.

Week 01/03/2021

Oakleigh

- Fixed meetings to correctly invite all members.
- Created this ongoing collaboration report to address team member involvement issues.
- Found Flask tutorials and summarised a code example which uses the flask API framework to return a “hello world” json.
- Communicated updates to supervisor and PhD student who has offered us Deep Learning advise.
- Created Minutes from this week and created the Agenda for next week.

Oliver

- Took part in debugging full implementation v2.
- Used his PC to train the model.
- Converted code to be used on Google Colab for additional online training.
- Generated a .h5 and .json model to use for front-end, back-end development.

James

Daniel

Thomas

Jack

-
-
- Researching API development for front end completion using flask and python correction

Week 08/03/2021

Oakleigh

- Wrote minutes for the weekly meetings.

Oliver

- Created a mock-up website to test implementation.

CMP2803M Team Software Engineering Assessment 3

- Created flask python API implementation to link the back-end model to the front-end.
- Tested front-end and back-end connection.

James

-

Daniel

-

Thomas

- Provided educational sources and tutorials for flask API.
- Promo video planning, creation and editing.

Jack

-

Week 15/03/2021

Oakleigh

- Assigned Report sections responsibility and created the template.
- Summarised Lecture materials to use as resources for the report.

Oliver

- Discussed progress and the future of the project with team supervisor during meeting.
- Further improved on the front, back-end development, fixing issues present in first version, such as random filenames to prevent filename clashes.
- Started work on implementation section for the report.

James

Daniel

Thomas

Jack

- Implementing the back-end with the official website.

Week 22/03/2021

Oakleigh

- Helped test implementation on other local machines.
- Moved Repository from Azure DevOps to Github.
- Took minutes for this week's meeting and the Agenda for next week.
- Created a new Teams Channel for Artefact V1.0, And included new github repo.

Oliver

- Tested final implementation.
- Added directory creation on version 1.0.

Thomas

- Researched a new padding method for the final implementation
- Started planning for the 'Software Development' section for the report and will start planning the 'Evaluation' section for the report.
- Will start the creation of the promo piece for the assessment.

Week 29/03/2021

Oakleigh

- Added and formatted minutes integration into the report Appendix.

CMP2803M Team Software Engineering Assessment 3

- Added Teams environment images & report allocations to report Appendix.
- Started to collect references.
- Created a Windows Forms to survey potential website users.
- Worked on software engineering and Group Work report sections.

Oliver

- Expanding on bullet points, approx. 500 words into the implementation section

Thomas

- Filling in on report sections, i.e. Testing

Jack

Daniel

James

Week 05/04/2021

Oakleigh

- Bought Google Colab pro
- Trained the model using a new “natural images” dataset from kaggle using kaggle API
- Got survey responses to a user review questionnaire

Thomas

- Filling in on report sections.

Week 26/04/2021

Oakleigh

- Found a public model code base to try to achieve a better working model.
- Tested model training with different hyperparameters; Epoch count, Number of images, percentage of validation dataset against accuracy, validation accuracy, loss and test image visual result.
- Achieved a more general working model.