Autophotographer

CS39440 Major Project Report

Author: Oscar Pocock (osp1@aber.ac.uk)
Supervisor: Dr. Hannah Dee (osp1@aber.ac.uk)

6th May 2022
Version: 1.0 (Release)

This report was submitted as partial fulfilment of a BSc degree in Computer Science (with integrated year in industry) (G401)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, U.K.

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Registry (AR) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name: Oscar Pocock

Date: 6th May 2022

Consent to share this work

By including my name below, I hereby agree to this project's report and technical work being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name: Oscar Pocock

Date: 6th May 2022

Acknowledgements

I would like to thank the following people for their support and contributions towards the project:

- my major project supervisior, Dr Hannah Dee, for helping me keep calm and collected thoughout the project and for keeping me on track.
- Aberystwyth University and Dave Price for providing me access to the unviersity's GPU compute cluster and providing assistance with using it.

Abstract

The goal of the Autophotographer project is to explore the idea of combination of machine learning and convential image processing techniques to select images with a high-level of aesthetic quality.

Contents

List of Figures

List of Tables

Chapter 1

Background & Objectives

1.1 Background

1.1.1 Context

In a world where most people have a camera in their pocket, the number of pictures being taken every minute has exponentially increased in the last few decades. Smartphones cameras have reached the point where they can rival their fully-fledged counterparts. The necessity[1] of online accounts on *Android* and *iOS*, which both include cloud storage (*Google Drive / Google Photos* and *iCloud* respectively) have made automatic cloud backup for photographs the norm. Together, these aspects have lowered the entry requirements and have enabled an entire generation to produce large quantities of high-resolution photographs, which have been previously limited to professional and hobbyist photographers.

The upsurge of data produced and people's reliance on free cloud storage has started to create storage concerns for providers. To combat this, certain providers have taken measures to reduce the load by reducing or introducing service limits [3] [4]. In turn, this had led people to reconsider how they manage their data. People have had to return to the tedious and manual practise of filtering through their old photographs in order to free up space for new content.

The significance of photo selection isn't limited to storage issues. The rise of social media and influencer-culture has culminated in entire businesses that require carefully selected photographs. *Instagram*, a photo-sharing social media platform, has become the most prominent hub for influencer-based marketing. Its popularity has spawned many Instagram curation apps [5] [6], which helps users and businesses plan out their posts to work towards a cohesive profile aesthetic [7]. This has also caught the attention of data scientists who have investigated what makes certain images perform better online than others [8].

Image aesthethic quality assessment is the process of automatically assessing the

[1] Both *Google* and *Apple* accounts are required to use their respective app stores and *Apple* doesn't support alternatives app stores.

visual aesthetic rating of an image. This area of research has seen recent improvements in the last 5 years due to the advancements in AI, computer vision, and machine learning. On the AVA dataset[2] alone we have seen accuracy rates of up to 83% when predicting the aesthetic quality rating of images, See figure 1.1.

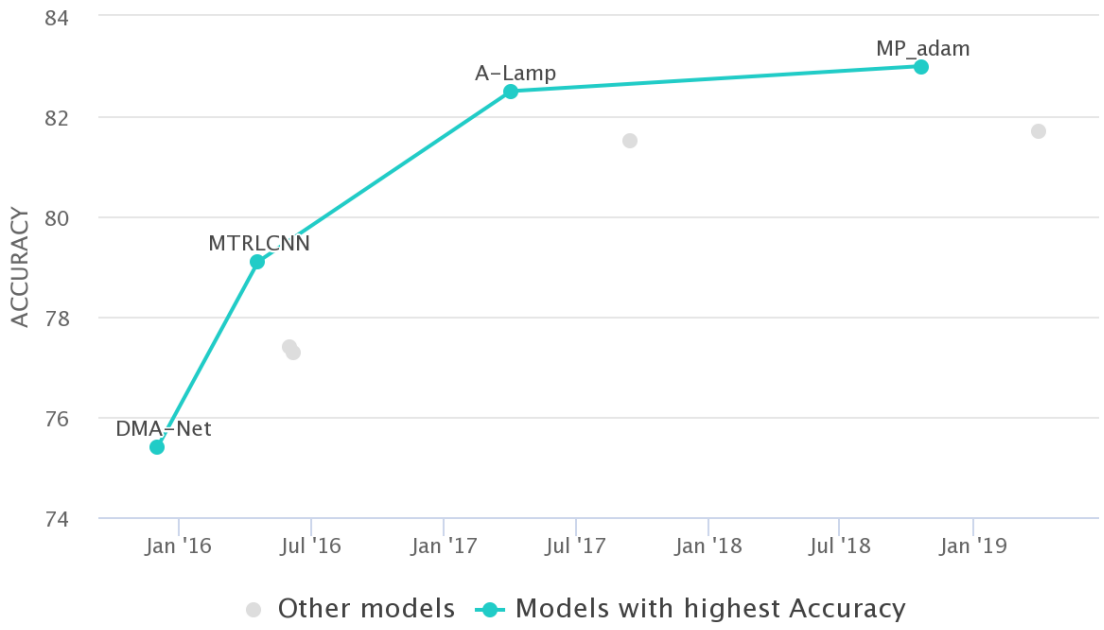


Figure 1.1: Accuracy of models trained on the AVA dataset [1]

1.1.2 Related Work

Trophy Camera In 2017, fine artist Dries Depoorter and professional photographer Max Pinckers developed a camera "that can only make award winning pictures" [9]. This camera was built using a Raspberry Pi and was programmed to only save award-winning photographs, which were subsequently uploaded to a website: trophy.camera [10]. The AI was trained using previous winning photographs from the WPPY (World Press Photo's of the Year) contest (*Warning: Website includes images some might find disturbing, including death, extreme violence, and suffering*) [11]. By comparing the photos featured on *trophy.camera* and the previous winning photographs of WPPY, you can see the effectiveness of the project is questionable. This isn't surprising, given the stark difference in subject matter and the photography skills of the general public versus the award-winning photographers. The majority of the photographs submitted to the WPPY are taken by professional photographers using high-end cameras (See figure 1.2c) and mostly depict global conflict. In contrast, the Trophy Camera is limited to the walls of a gallery, using a low-end camera, and the photos are taken by the general public (See figure 1.2b). This is an example where the data used for training doesn't match the desired use-case of the project, resulting in

[2]The AVA (Aesthetic Visual Assessment) dataset is the most common dataset used for aesthetic quality assessment.

images that fail to achieve an “award winning” appearance.

(a) Trophy Camera [9]

(b) Photo taken with Trophy Camera [10]

(c) Example of photo used to train Trophy Camera [12]

Figure 1.2: Trophy Camera

Archillect A combination of the words “archive” and “intellect”, Archillect [2] is an AI that automatically curates visually stimulating content found online. To do this, Archillect uses an algorithm and a list of of keywords that searches for posts and pages, then crawls pages linked from original results to find new images and gain relevant contextual knowledge and learn new keywords. Ultimiately, Archillect learns what images create visual stimulation for people on social media, reposting them and updating the algorithm according ”likes” and other user engagement metrics.

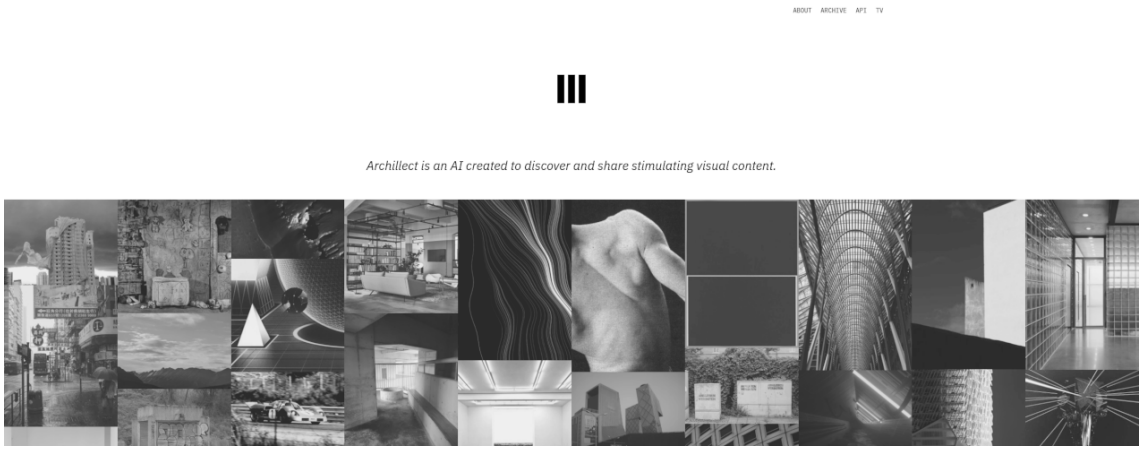


Figure 1.3: archillect.com homepage [2]

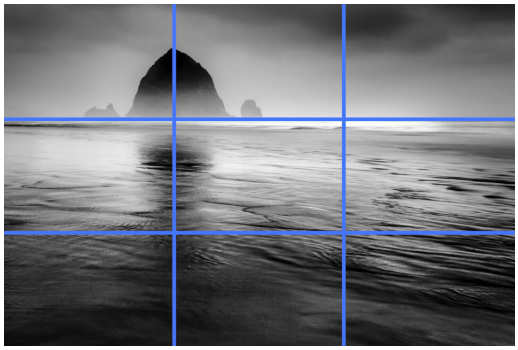
1.1.3 Motivation

Historically, the most common and successful approaches to grading the aesthetic quality of images has been through machine learning. This involves creating a neural network, typtically a CNN (Convulution Neural Network), and training it with professional photography based datasets. The most common of these datasets being AVA (Aesthetic Visual Analysis), which is comprised of a set of professionally taken photographs and an aesthetic rating provided by numerous professional photographers that acts as a ground truth for aesthetic quality.

Many people believe aesthetics are subjective. We can observe this when the aesthetics of certain art pieces can often be contested and contraversial. Considering the percieved subjective nature of aesthetics, there seems to be certain ideas that are almost universally accepted as aesthetic. In art we often hear about composition rules like the golden ratio, rule of thirds, and symmetry which are meant to invoke a positive sentiment. See figure 1.4. For others, aesthetics can't be reduced down to a set of predifined rules and many people welcome images that break the natural conventions of professional photography.

(a) Golden Ratio [13]

(b) Symmetry [14]



(c) Rule of Thirds [15]

(d) Shallow Depth of Field [16]

Figure 1.4: Common photographic techniques

1.1.3.1 Applications

Automatic image selection is a useful concept that can be used in many applications. The main motivation for the project was to use the system as a acessibility tool, allowing those who are less technically or physically abled to take good aesthetic-quality images. They use a head-mounted camera and record a journey our outing and the tool could automatically reduce the footage into an album of good-looking photos for them to enjoy.

Professional photographers could use automatic image selection to remove technically poor images from a large collection of images from a photoshoot. This process would help alleviate much of the manual filter photographers have to do post photoshoot.

1.1.3.2 Personal motivations

An important motivation for this project and it's direction is my interest in machine learning. The CS36220 Machine Learning [17] module acted as a good introduction to the topic, but I wanted to get some personal experience with deep learning as it's an area of computer science I find really interesting.

1.2 Analysis

1.2.1 Problem Description

This project aims to explore the question of "Can we use a computer to make accurate aesthetic judgements on images?". In order to create a program to detect and select aesthetics images from a video file, the video file needs to be broken down into individual frames, then each frame needs to be analysed against certain aesthetic features and rules like (rule of thirds, contrast, brightness, focus). Each of these filters will need to be implemented in such a way that allows any order and combination of filters to be used. Lastly, a machine learning approach will be taken to rate and rank the remaining images. For this we can use a CNN (Convolution Neural Network) which will need to be trained on professional photographs as they can be considered a good base for what is considered a good "aesthetic" quality.

1.2.2 Approach

The specific approach taken for this project is to use a combination of machine learning and conventional image processing techniques. This is a good balance between performance and accuracy as the conventional image processing techniques can filter out the poor technical images (blurry, low contrast, extreme exposure etc.) therefore limiting the amount of processing required by the machine learning model at the end of the pipeline.

1.2.3 Alternative Approaches

This project could also be approached from an exclusively coventional computer vision approach, which would require writing specific code to recognise aesthetic photographic techniques (like rule of thirds and vanishing point: see figure 1.4) and using those to help determine the aesthethic quality of an image.

A machine learning only approach could also be taken but for the context of this project this would be computationally wasteful. Due to the nature of a continous video recording, many frames will be blurry or under exposed due to lighting changes. These images are technically poor in quality and processing hundres of them using a CNN

would be wasteful when they could be more easily discarded using conventional image processing techniques.

1.2.4 Aim

The aim of the project is to develop a piece of software that will take a video or a set of images as an input, process them through filtering, and output a subset of ”aesthetic” images or frames. The set will be processed through filters which will be implementated as part of the project.

1.2.5 Objectives

- Context specific research into machine learning
  - Research suitable machine learning frameworks
  - Research how to start building a CNN
- Setup tools for development
  - Host a Gitea instance to hsot the project’s repository
  - Create repository for project
  - Mirror repository to GitLab.com for availability
  - Setup WoodpeckerCI for CI/CD workloads and connect it to the project repository
  - Setup pipelines for automated testing in WoodpeckerCI
- Research convential image processing techniques
  - Quantify brightness
  - Quantify contrast
  - Quantify focus
  - Depth of field detection
  - Vanishing point detection

1.2.5.1 Research Questions

- How effective is the machine learning approach compared to a more conventional one?
- Which order of filtering is most effective?
- Can a CNN be used to make aesthetic judgements of an image?

Chapter 2

Experiment Methods

2.1 Hypothesis

Hypothesis: Can a CNN and an a set of image processors make accurate aesthetic judgments on images.

Chapter 3

Software Design, Implementation and Testing

3.1 Development Process

Originally, the plan was for the project to follow a scrum-like methodology, but due to the research-based nature of the project and working with new technologies – it was difficult to estimate the time and effort required for certain tasks.Naturally, the weekly meetings with the project supervisor acted as a weekly review of the work done and what work was to be done the following week. When the project repositories were set up, a Kanban board was set up alongside it. This was used to keep track of the progress of tasks and features. Milestones were created to group small tasks together that lead to a bigger task being completed.

3.2 Languages and Frameworks

3.2.1 Python

Python [18] was selected as the programming langauge for the project due to it's popularity in data science and it's dynamic and weakly typed nature. This enables an easier transisition from project ideas to working code. It was also chosen for it's use in machine learning as it seems to be the main language in most of the top machine learning libraries.

3.2.2 PyTorch

The most common frameworks for deep learning are *TensorFlow* [19], *PyTorch* [20], and *Keras* [21].

TensorFlow, arguably the most known of the three, is developed by Google. It's mostly

used in Python but can also be used with C++, Java, and Javascript. It provides a powerful framework for ML (Machine Learning) and AI (Artificial Intelligence) tasks. Although, TensorFlows primarily focuses on creating deep neural network and training.

Keras is a high-level deep-learning API used to interface with TensorFlow. It's main aim is to simplify the process of developing deep neural networks for use in quick experimentation.

The last of the three, *PyTorch*, is primarily developed by Facebook and acts as the main competitor to TensorFlow. It's gained popularity within the research community due to it's ease of use and gradual learning curve compared to TensorFlow.

The decision was made to use PyTorch as machine learning isn't the primary aspect of this project. The reduced complexity should speed up development time.

3.3 Software Tools and Technologies

3.3.1 VS Code

The IDE/Editor used for this project is VS Code[1]. This was chosen as it was a familiar and popular IDE/Editor with good plugin support and features.

3.3.2 Docker

Docker containers were used to host tools like Gitea and Woodpecker on a private server to aid with development. They were also used to create reproducible development and testing environments[2].

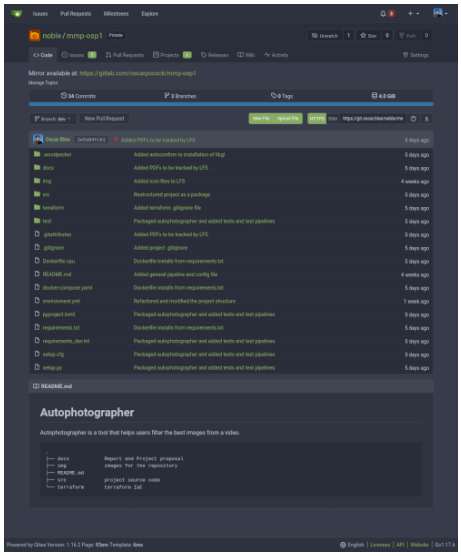
3.3.3 Gitea

For version control this project uses git and Gitea as an open-source and self-hostable git host. Gitea also includes workflow features like bug tracking via "issues", kanban boards (see figure 3.2), and plugins to work with other tools (see section 3.3.4). The source code for this project is hosted on a personal instance of Gitea hosted at git.oscar.blue. The project's repository is also mirrored[3] to gitlab.com for availability.

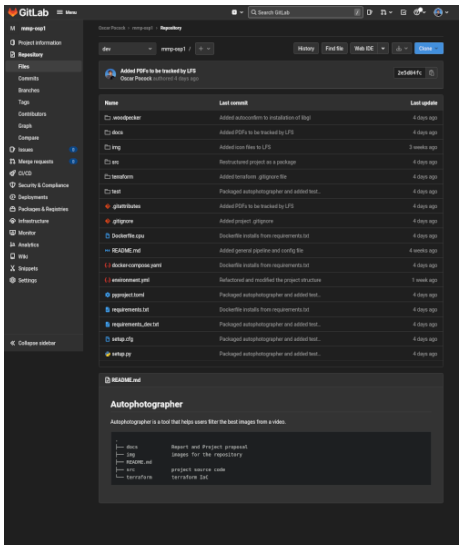
[1] More specifically: VS Codium. A libre-binary version of Microsoft's VS Code. It's built from VS Code's source albeit with tracking and telemetry disabled and then distributed at binaries. It's essential VS Code, so it'll continued to be referenced as such.

[2] a Dockerfile and Docker-compose file are supplied with the source code for this project.

[3] Sync interval is every 8 hours.



(a) Original Gitea version



(b) Mirror Gitlab version

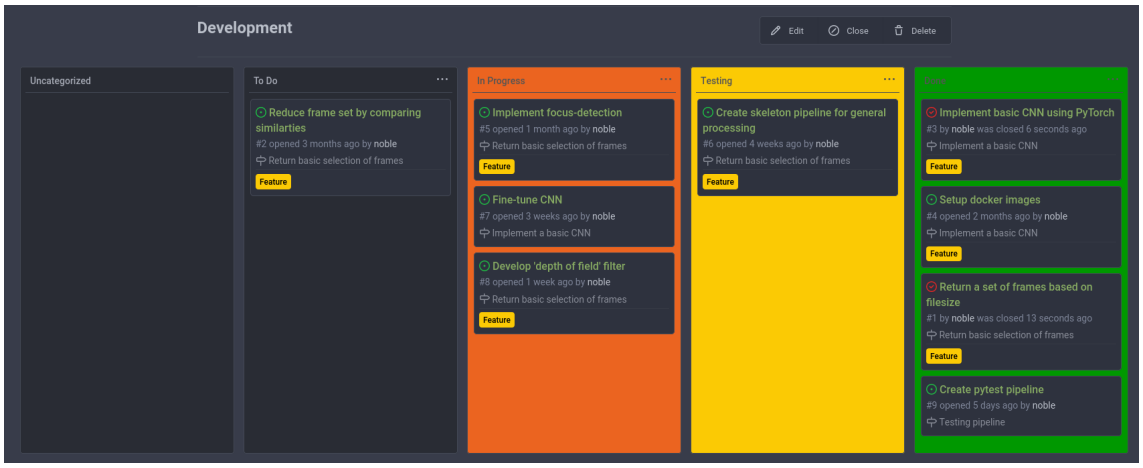


Figure 3.2: Gitea Kanban board

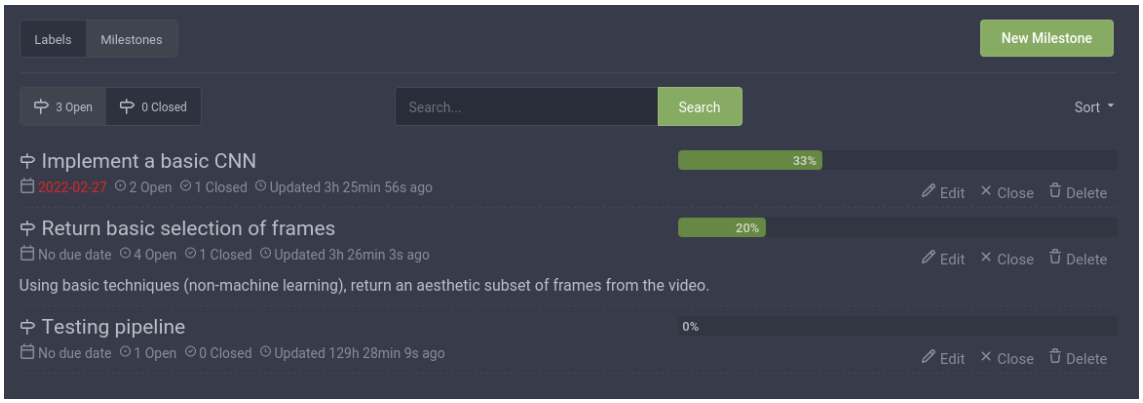


Figure 3.3: Project milestones

3.3.4 WoodpeckerCI

WoodpeckerCI, an open-source community fork of DroneCI, is a CI/CD (Continuous Integration/ Continous Deployment) platform used to create automated piplines for tesing and building. A personal instance was hosted at woodpecker.oscar.blue and linked to the project's git repository for CI/CD.

3.3.4.1 Woodpecker pipelines

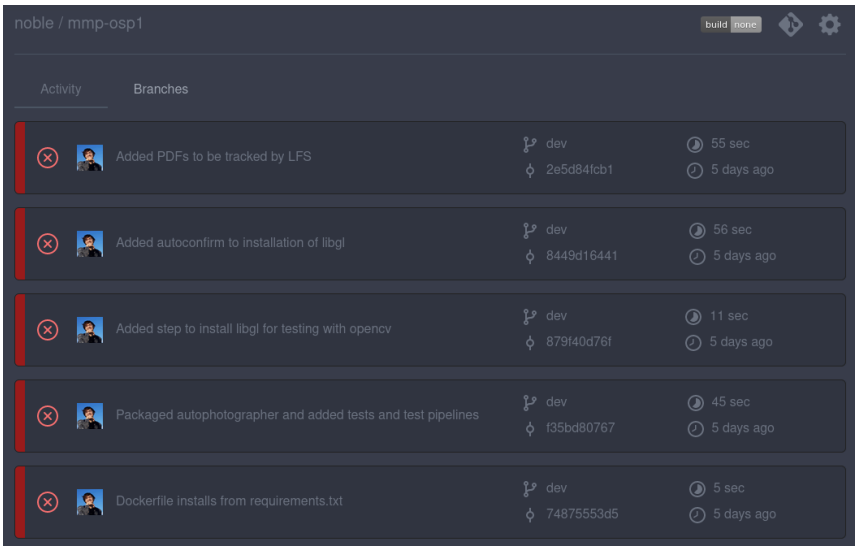


Figure 3.4: Woodpecker Pipelines

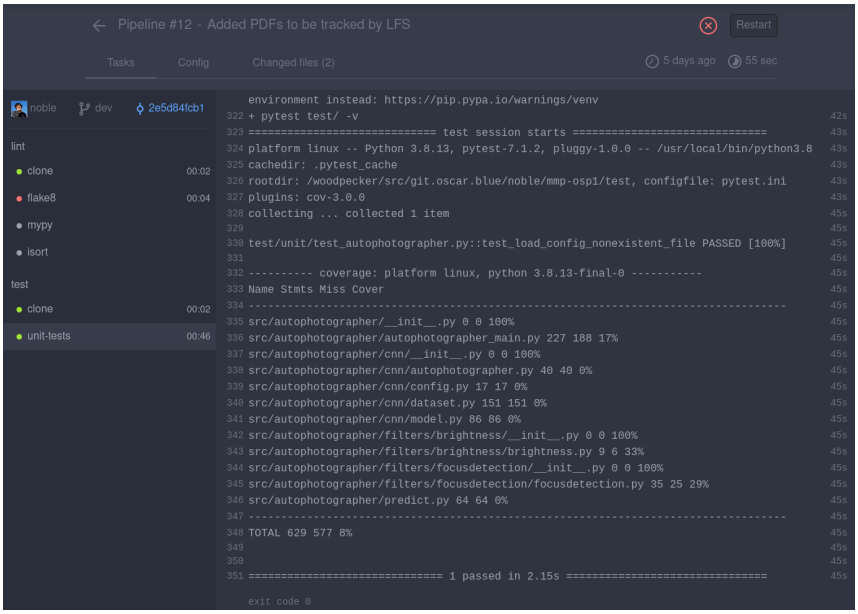


Figure 3.5: A Woodpecker Pipeline

Two pipelines were created, one to establish automated testing and the other for source code linting.

3.3.5 Terraform

When training the CNN, Terraform was used to create GPU-enabled cloud compute resources on the fly so that it could easily be built up and torn down during training. Although this code was barely used in the end due to moving away from cloud solutions (See section 3.5.4.3 for details).

3.4 Design

3.4.1 Overall Architecture

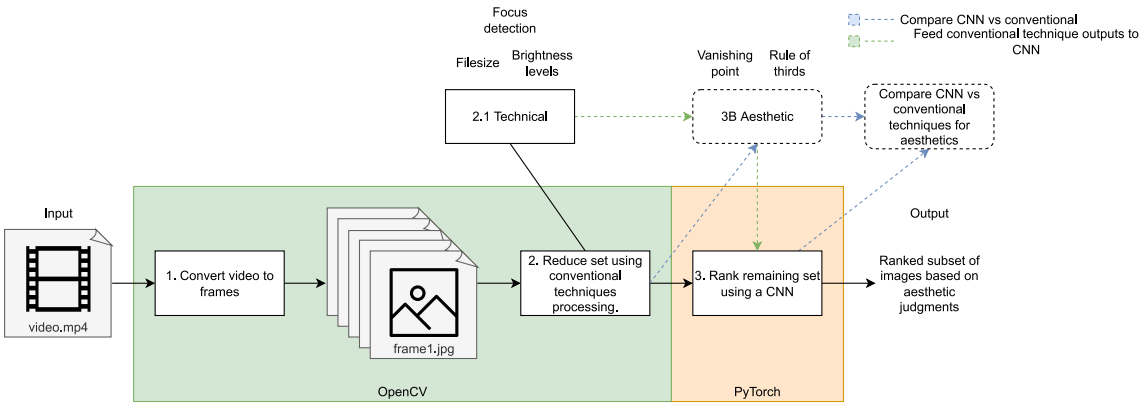


Figure 3.6: High Level Architecture

3.4.2 Detailed design

3.4.2.1 CNN model architecture

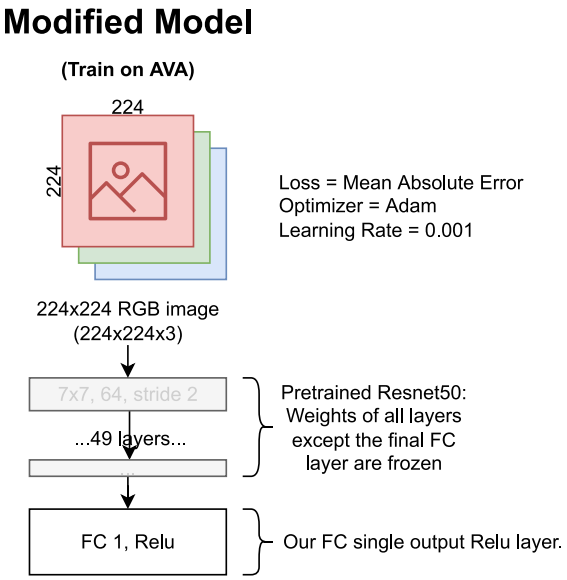


Figure 3.7: Simplified CNN Architecture

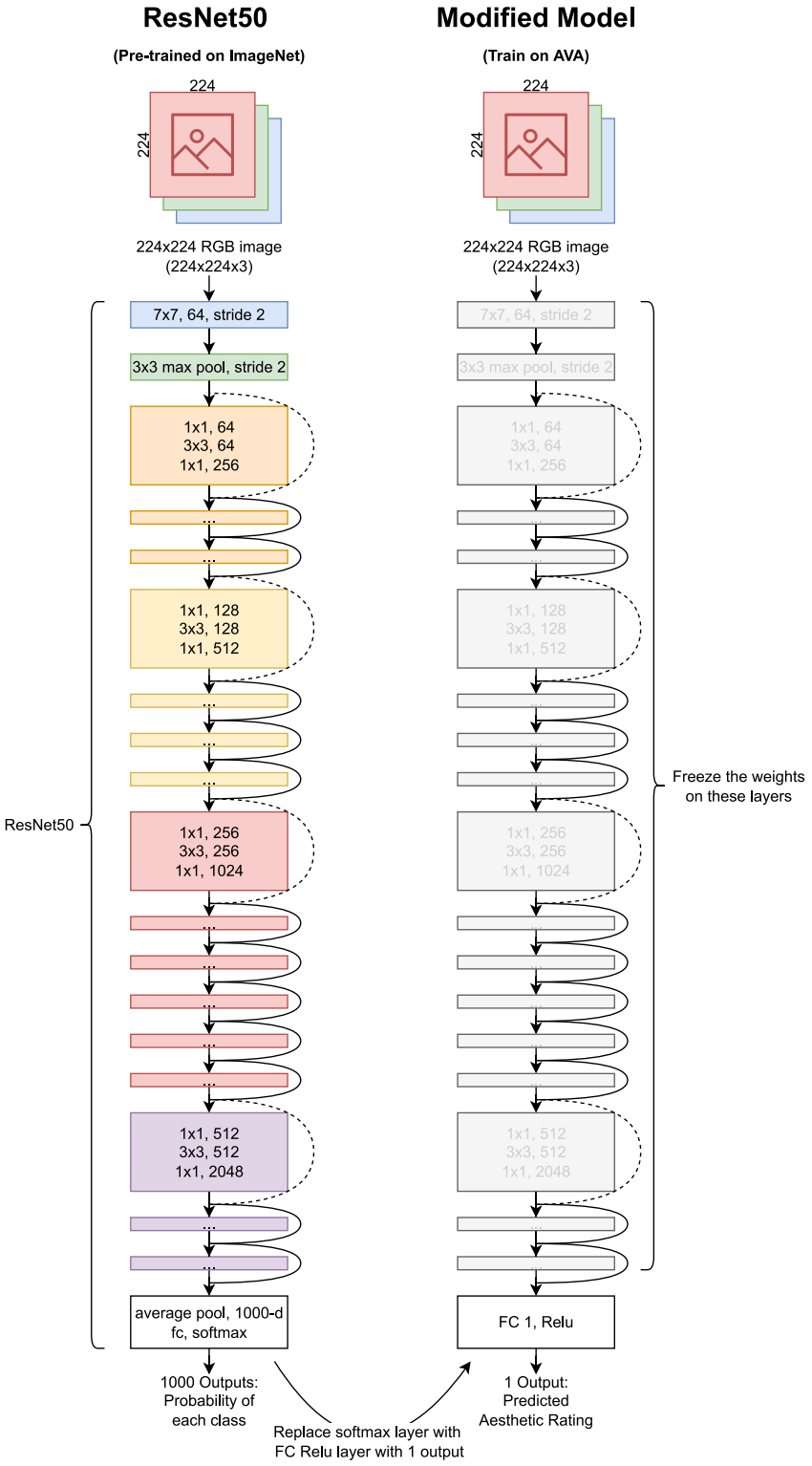


Figure 3.8: CNN Architecture

3.4.2.2 Functions

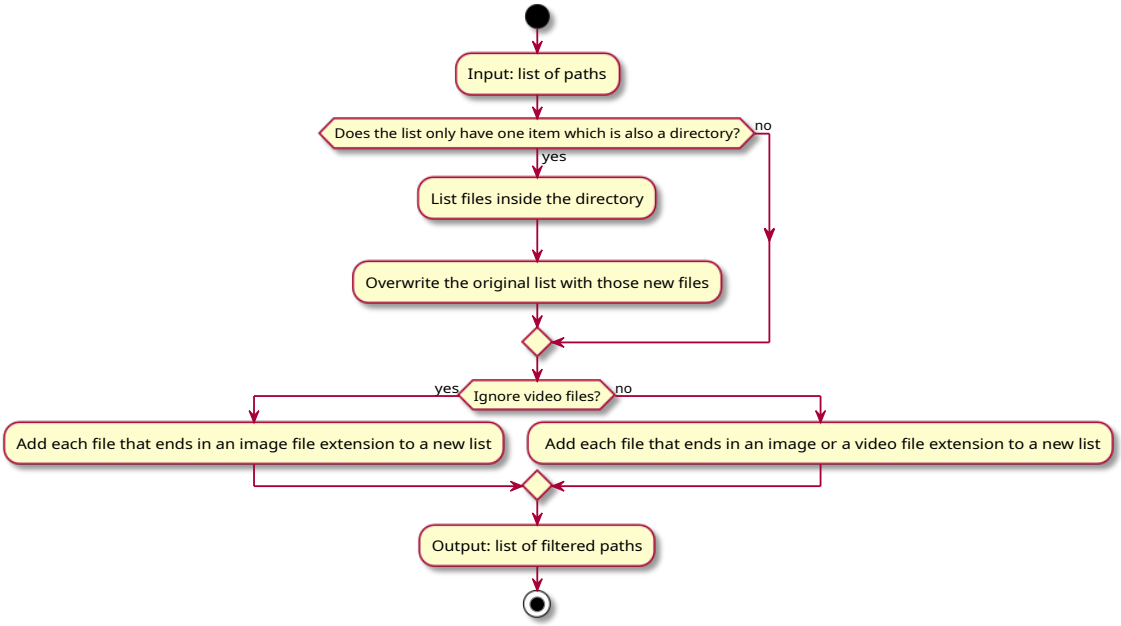


Figure 3.9: filter_paths() function

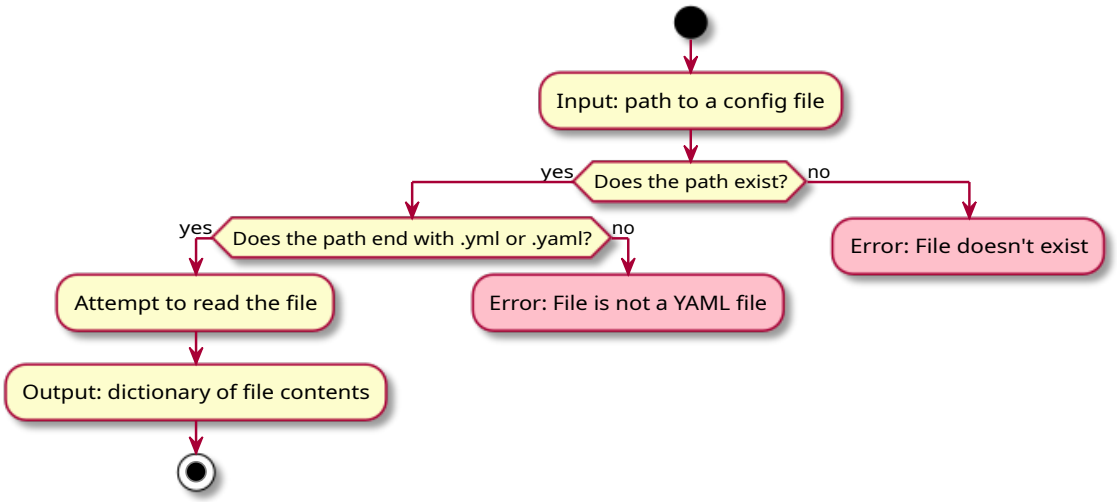


Figure 3.10: load_config() function

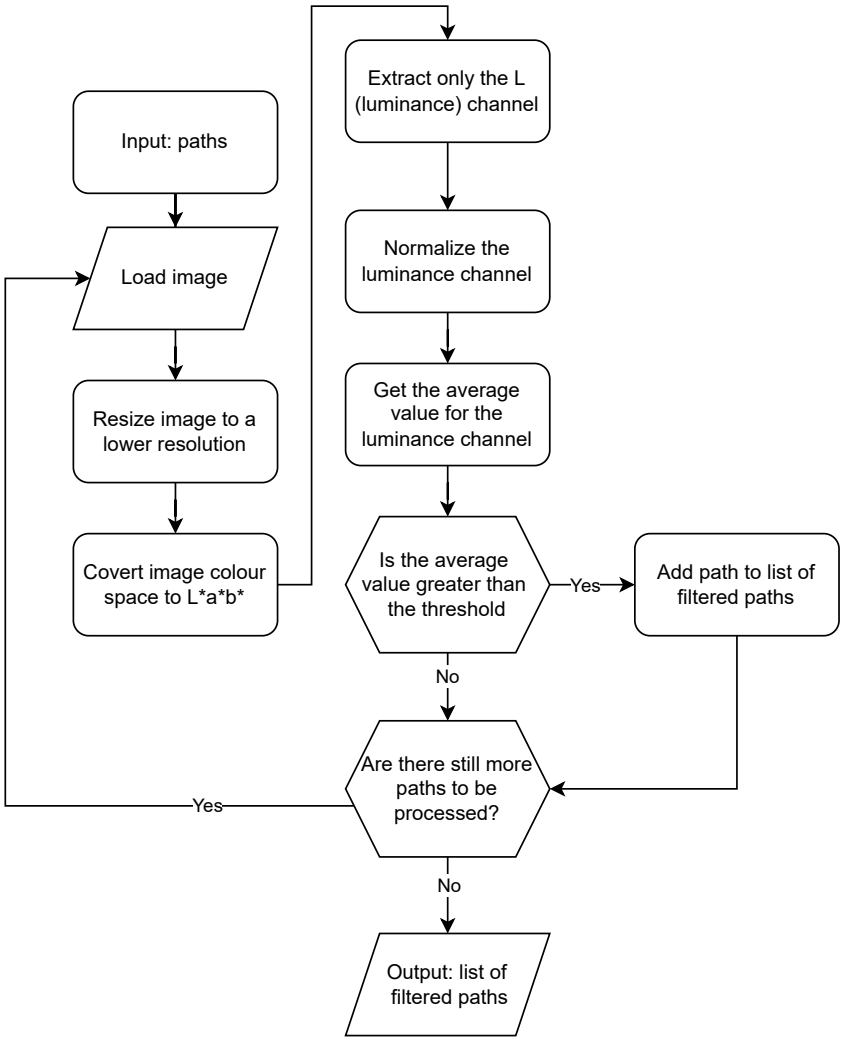


Figure 3.11: filter_brightness() function

3.4.3 Datasets

AVA The most common dataset used for analysing image aesthetics is the AVA (Aesthetics Quality Assessment) [22]. This dataset is comprised of over 250,000 photographs taken from DPChallenge [23] including a varied selection of metadata including: a large number of aesthetic ratings from users for each photograph, over 10 photography style labels (macro, rule of thirds, shallow depth of field etc.), and over 60 semantic lables (landscape, animal, wedding etc.).

In the paper released alongside the dataset [22], it references that the dataset is available at www.lucamarchesotti.com/ava. Although, this link has been down for the past 5 years. To work around this, there are many scripts online which will scrape DPChallenge to build the dataset for you. Others have decided to upload the entire dataset online for quicker access and from preventing users from being rate-limited by DPChallenge.com while scraping. For this project the dataset was downloaded from

MEGA (mega.nz/folder/9b520Lzb#2gIa1fgAzr677dcHKxjmtQ) which was provided by a GitHub repository [24] which also included AVA download-helper scripts.

AADB Another common dataset is AADB (Aesthetics and Attributes Database) [25]. Unlike AVA, AADB uses photos from Flickr [26] an image sharing site which is less targeted towards professional photographers and therefore hosts a more varied range of aesthetic quality compared to DPChallenge. Although AADB uses only 10,000 images compared to AVA's 250,000. In AADB, rater identity is anonymously recorded a tracked across different photos which they use in their sampling strategy to contexualize the subjectivity of aesthetic tastes.

As the project is aimed to help the general public to select high aesthetic quality frames from videos, it may make more sense to use the AADB dataset as it contains more amateur photography compared to AVA. Although, AVA has the advantage of being a larger dataset. Both datasets could be helpful in working with the project.

3.5 Implementation

3.5.1 Configuration

A YAML configuration file[4] was created to enabled quick changes to the filters and pipeline options. YAML was selected as it's a human readable and allows comments unlike JSON. The main options were:

- **filters** - declare which filters you want to use, and in which order (default)
- **CNNrank** - declare whether you want to rank the remaining images using a CNN (default: False)
- **ignore_video** - declare wether you want to ingore video files when loading in paths (default: False)
- **brightness_options** - declare options for brightness filter
- **filesize_options** - declare options for filesize filter
- **contrast_options** - declare options for contrast filter
- **focus_options** - declare options for focus filter

3.5.2 Pipeline

The overall pipeline takes in a set of paths and will applying image processing in layers to filter the lower aesthetic quality images and thus reducing the set of paths (see figure 3.6).

[4] See example configuration file in Apendix B section 2.1.3.

- Loads in the video or image
- Runs through each filter and ouputs a subset of the list of paths

3.5.3 Filters

In general, each filter takes in a list of paths and some options (for thresholds etc). The filter will attempt to reduce down the list of paths based on it's filtering technique and threshold. The reasoning behind each filter taking a list of paths as an input was to enable future functionality which would process the frame in context to the other frames. Based on the filter type, some filters will only have a lower bound (e.g discarding filesizes below a certain threshold) while others will include both a lower and higher bound (e.g. discarding images that are too dark or too bright in the brightness filter).

3.5.3.1 Filesize

When comparing file sizes it's safe to assume that larger file sizes contain more information, more information might allude to a more interesting and thus more likely to hold a higher aesthetic quality. When filtering by filesize, only the lower bound is considered. In other words: only filter out images below a certain filesize. To do this, set a threshold and compute the filesize of a given file. If the filesize is lower than the threshold then the image is discarded.

3.5.3.2 Brightness

To quantify an image's brightness the L*a*b* colour space is used. This colour space is meant to reflect human perception of light and colour. In the case where brightness is only concerned, the L (Luminance) channel can be used as a comparable metric to human perception of brightness. To use this effectively, the image is first converted to L*a*b, then the pixel values for the L channel are normalised by dividing each pixels value by the maximum value, then the average of pixel brightness is calculated. This will give us our average luminance of the image. Here brightness should have a lower and upper bound. As some images can be over exposed or under exposed. Both extremes result in poor image quality. Therefore, we can set a percentile band removing the images that land in the extreme upper and lower bounds.

3.5.3.3 Contrast

To detect contrast an existing function in the skimage [27] library called "is_low_contrast" method. This method will take an image and a threshold and return true or false depending on if the contrast is above or below the threshold value. Again, like brightness, contrast has lower and upper bounds that need to be removed as an image can have too much contrast or too little. When researching how to calculate an image's contrast levels, the following post was found and followed as a guide [28].

3.5.3.4 Focus

There are a couple of popular methods for detecting if an image is blurry or not. For detecting blurry images there are two common methods. Laplacian method and fast farrier transform. The laplacian variance method. This method can actually be implemented fairly easily using cv2. You can compute the variance of the laplacian of an image then you can just take the standard deviation squared. A low variance indicates that there aren't many edges in the image, something we can see in blurry pictures. While a high variance means there are a lot of edges and therefore the image is less likely to be blurry. Although this method alone won't allow us to predict if an image with a low depth of field, with a subject is in focus or not as it will just return as "this image is somewhat blurred" which doesn't tell us much. This method works well when you can compute an good level of focus before hand and then you can remove the outliers "(like for a consistent video feed). The following guides were used as a resource for this methods [29] [30].

3.5.4 CNN

3.5.4.1 CNN architecture

Originally, the plan was to build a CNN from scratch using PyTorch. This required a lot of research as the CS36220 Machine Learning [17] module didn't cover the details of CNNs nor their practical implementation. To gain a better theory understanding, Stanford University's online lecture material covering deep learning software [31] and CNN architecture [32] was used. For practical knowledge several sources PyTorch's official tutorial for training [33] and pyimagesearch's guide on training a CNN [34] were used a foundation for understanding the how to use PyTorch – which is also where the idea of using a separate configuration file for storing variables for batch sizes and validaton splits was inspired from. While attempting to build a CNN, existing CNN approaches [35] [25] were researched in detail to build foundational knowledge of how to tackle the shared problem of judging image aesthetic quality. It soon became clear that most sucessful models were using transfer learning instead of building a CNN from scratch. Transfer learning is the process of using the knowledge learned by an existing model and transferring it to a new model which is attempting to solve a similar issue.

After understanding its benefits, a transfer learning approach to creating a CNN was adopted. This approach was more suited for the project due to the time contraints and the approach's increased probability to create a good working model.

ResNet50 [36] was chosen as the base model as ResNet based networks have historically performed really well in transfer learning due to their large amounts of hidden layers. The large amount of hidden layers led to it winning all classification and detection competitions in both ILSVRC (ImageNet Large Scale Visual Recognition Challenge) and Microsoft's COCO (Common Objects in Context) in 2015. ResNet50 is a ResNet model with 50 hidden layers (See figure 3.8) and 1000 different classes. It is trained on the ImageNet [37] dataset, a large dataset comprised of over 14 million annotated images. This dataset is used in ILSVRC as a benchmark for models in classification and object

detection.

Although this project's problem could be implemented as a classficiation problem in the sense of "Is this image of high aesthetic quality?" with two classes "Yes" and "No", we would achieve better insight through a regression-based model asking "How high is the aesthetic quality of this image?" which would return a rating between 0 and 10. To learn specifically how to implement a regression-based model, a couple of guides [38] [39] surrouding the building of regression-based models using the real estate dataset [40]. Transfering the knowledge from the ResNet50 model and altering the model to fit our problem requires a few steps. The pyimagesearch's guide for transfer learning in PyTorch for classification problems [41] was also used as a guide and much of the structure of the model.py code is inspired by this post.

1. Load ResNet50 model
2. Freeze all layers except the last
3. Remove final layer
4. Add a fully connected layer with one output and a relu function
5. Train using new problem data (AVA dataset)

The original plan was to train the two models, once with the AVA dataset and once with the AADB dataset and compare their accuracy. Due to time contrainst and the other aspects of the project being neglected, this had to be dropped and only the AVA dataset was used in the end.

3.5.4.2 Working with the dataset

The first step to working with the AVA dataset is to load in the data in the format we need it. As the images in AVA were rated from a scale of 1-10 we also want our model to ouput a rating between 1-10. The way AVA exposes these ratings is by showing the count of each rating (see figure 3.12). In other words, for each image it shows you how many users rated the image (1/10, 2/10 etc.). For this project's use-case, these counts of ratings aren't very useful. They need to be processed into a single value. To do this, each rating is given a multipler (a rating of 1 is multiplied by 1, rating of 2 is multiplied by 2 etc.). The sum of all of these new multipied ratings is then dividing by the total number of casted votes. This should give us a single value rating between 1-10 (see figure 3.13).

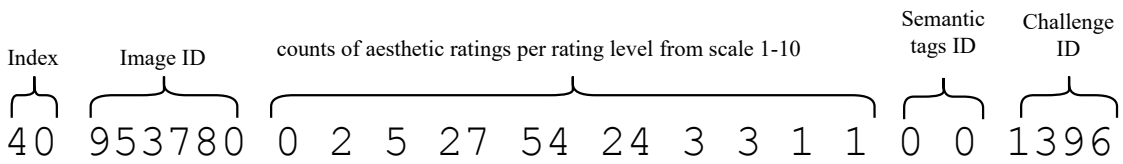


Figure 3.12: AVA Dataset entry

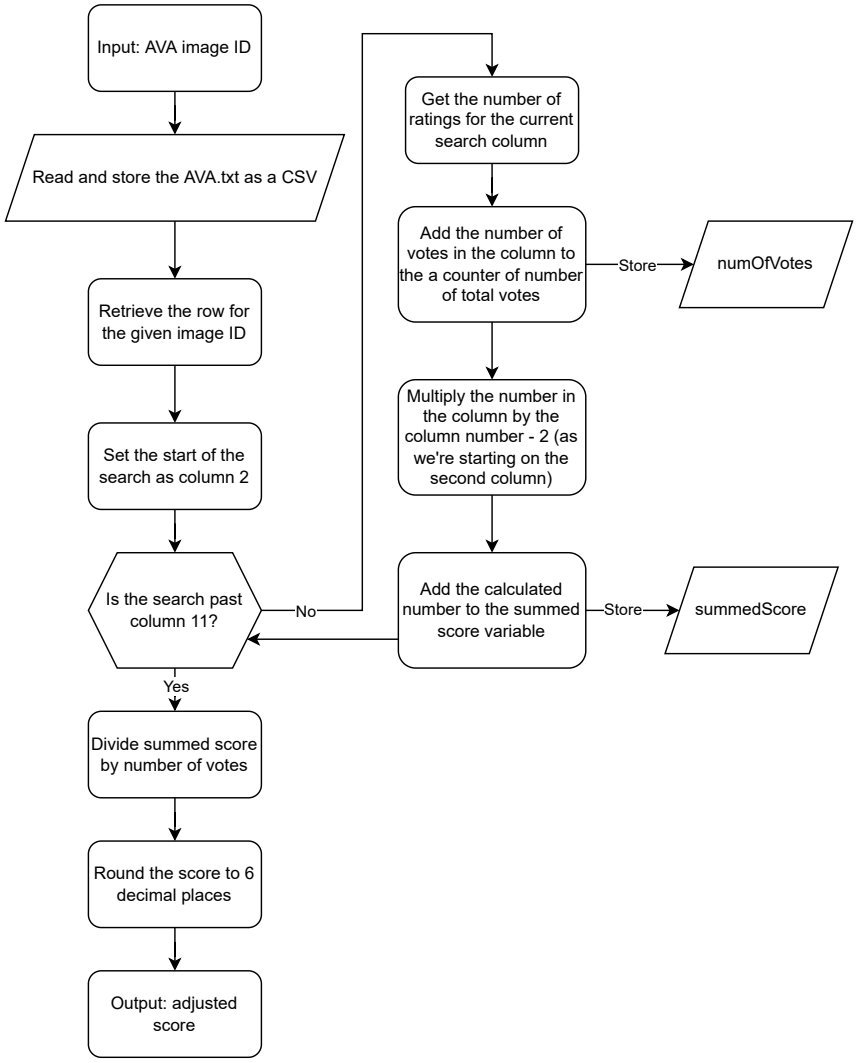


Figure 3.13: calculate_image_rating() method

3.5.4.3 Training model

In order to train the model, PyTorch's in-built training functionality was used. Intially, the model was trained for 20 epochs on the CPU to validate that there weren't any runtime errors. In order to run the training for 20 epochs on a CPU it took just over 4 hours, meaning 2000 epochs would take over 400 hours or roughly 16 and a half days to train on a CPU. In order to speed up processing GPU-compute power was required. PyTorch has long supported using CUDA on Nvidia GPUs for accelerated workloads but has only recently implemented support for AMD's equivalent: ROCm[5] [42] [43]. ROCm also has a lack of installation targets - mostly packaging the stack for enterprise Linux distributions like RHEL (Red Hat Enterprise Linux), SLES (SUSE Linux Enterprise Server), and Ubuntu[6]. Fortunately, ROCm is open source, unlike Nvidia's counterpart – CUDA. This

[5] As of writing ROCm support for PyTorch is still in beta.
[6] Ubuntu requires the HWE (Hardware Enablement) kernel.

meant that theoretically ROCm could be compiled from source and installed manually. When this was attempted, the machine compiling the package ran out of memory and halted. ROCm was abandonned and the choice to move forward with CUDA was made. As the computer used for the project didn't have an Nvidia GPU, the option of using cloud computing to was explored.

The next step involved researching a number of cloud providers which offered GPU compute and comparing prices. AWS (Amazon Web Services), the biggest global cloud provider to date, was unrealistically expensive so GCP (Google Cloud Platform), Azure, and Linode were considered instead. Although each of these platforms advertise GPU computer services, each platform has a GPU quota which is automatically locked to zero. This require contacting support to have the restriction lifted or in the case of Linode there's an additionally requirement to have $100 or more in transactions before asking to remove the lock. After contacting GCP support, the quota for GPUs was increased to one which enabled the CNN model to be trained using Google's cloud GPU compute.

As part of the project and the drive for creating an optimal development environment, IaC (Infrastructure as Code) was used to rapidly automate deployments of cloud resources, minimising cost and saving time when training. Terraform, a cloud agnostic IaC tool, was used to define the architecture for the cloud resources required to run a training workload. Unexpectedly, working with GCP was harder than expected due to it's novel methods. By default, GCP creates a user account for a newly created server and disables password authentication. Instead, GCP creates an SSH key pair and stores the private key within the GCP platform which isn't viewable by the user. To log in, users have to use the gcloud CLI application which connects to their GCP account which then in turn fetches the private SSH key and automatically logs the user in. Due to this convoluted process, it was difficult to automate the post-installation commands required to set up the training environment. Further issues arised when the decision was made to use Google's official machine learning images which were advertised as "the easiest way to get started because these images already have the NVIDIA drivers and CUDA libraries pre-installed" [44]. In reality, CUDA was not preinstalled. Instead, an installation script ran on boot which would install the aforementioned drivers and libraries. Unfortunately, Google's script clashed with the project's post-installation script as both of them used apt. This would result in the project's script locking apt when installing it's dependencies, preventing Google's script from using apt to install the drivers. Google's script would then terminate but wrongfully output that the drivers had been installed.

Around the same time, the university offered access to use their GPU compute cluster on the condition that the project only used one GPU from their old server. The university's GPU compute cluster managed jobs using slurm [45] which enqueues an new job to the cluster. The resources on the old server limited it to job very few concurrent jobs, creating a scenario where existing jobs would have to finish before new ones could start. This was ultimately the best option considering it was being provided at no extra cost and the setup wasmuch more straight-forward compared to GCP.

The university's GPU cluster uses Anaconda [46] virtual environments to run experiements. Once a new environment was created and the depencies for the model training code were installed, a model was able to be trained. This code trained the model for 2000 epochs and saved a plot the train and validation loss.

To speed up processing time, a batch of the dataset was pre-processed with translations and saved to tensors. This was originally done to speed up runtime troubleshooting in the training code by cutting out the time it took to process the images before each run.

3.6 Testing

3.6.1 Overall Approach to Testing

Due to the explorative and experimental nature of this project, a lot of code was used temporarily or was CNN based which is difficult to write tests for. Therefore testing was left till the end when the project when there was more structure and finalised pieces.

3.6.2 Automated Testing

CI/CD pipelines were set up to facilitate automated testing using pytest and pytest's conv plugin which shows a breakdown of test coverage on the codebase.

3.6.2.1 Unit Tests

Only 4 unit tests were implemented. Three of the tests check the functionality to calculate a single value for the AVA ratings. As a lot of the functions in the this project rely on complex datastructures like panda DataFrames and images testing was quite difficult and the use of mocks was necessary.

As the ”calculate_image_rating()” function reads a CSV file and stores it as a dataframe, a mock dataframe had to be created for each test.

test_calculate_image_rating_with_no_ratings This test is checking that if all ratings for an image are 0 then the resulting single score would also be zero. This test intially failed and made aware the fact that by following the logic of the algorithm it would try to divide 0 by 0. This caused a runtime error and therefore a specific check had to be put into the function which checked if the summer score equated to 0. If it did, it would avoid diving by the total number of votes and instead just return 0.

test_calculate_image_rating_multiple_ratings This test will check that the algorithm accurately calculates a single rating value.

test_calculate_image_rating_incorrect_number_of_columns This test checks that the right exception is called when it attempts to calculate the image rating on an entry which has more than 15 columns which is the number of columns an entry had in the AVA dataset.

test_load_config_nonexistent_file This test checks if the right exception is thrown when an non-existent file is attempted to load using the ”load_config()” function in the autophotographer_main.py script. This test implemented very well as it relies on the filesystem of the user. If the user has a file named ”blahblahblah” on their system in the right location then the test will falsely fail.

3.6.3 CNN testing

When testing the CNN performance, a random small batch of 4 images is selected and then their rating is predicted and plotted. Although as the images are fetched from the validation loader, they are pre-processed which has led to processed images being shown in the reduced size they are scaled down to for training.

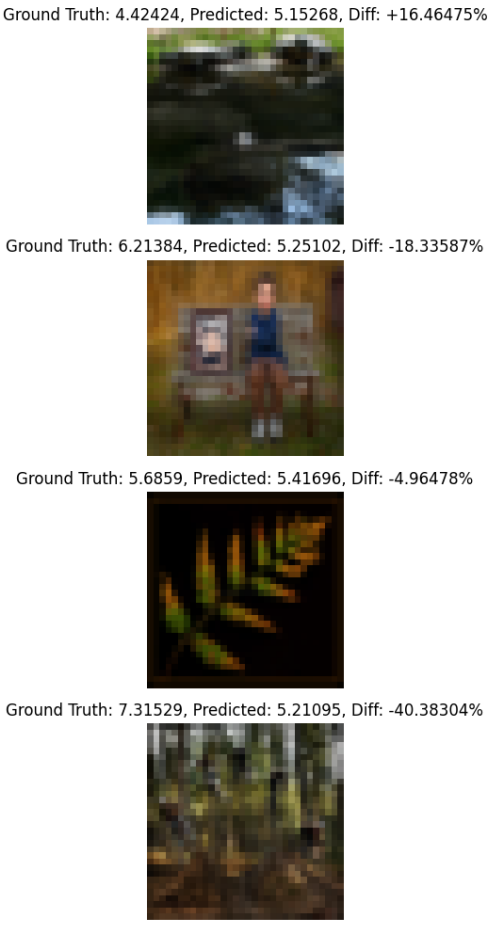


Figure 3.14: CNN Testing

As observed in figure 3.14, the CNN isn’t very accurate as predicting the aesthetical quality of these images and the variance in prediction/ground truth is quite large between images.

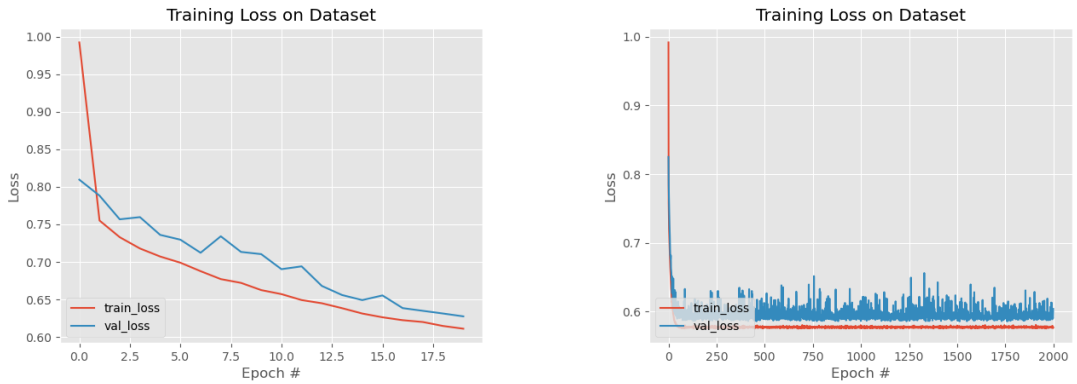
Chapter 4

Results and Conclusions

4.1 CNN

4.1.1 Training

The first training attempt for the CNN was only run for 20 epochs on a CPU to check for runtime errors. During these 20 epochs the loss gradients didn't plateau meaning the model could benefit from longer training. The next attempt was run for 2000 epochs on a GPU. Figure 4.1b shows that the training loss started to slow down around 30 epochs and plateaued at around 60. While the validation loss also plateaued around 60 epochs, the variance in loss was much greater. This most likely means the model has converged on an concept of aesthetic quality that doesn't match the concept it was set out to learn. As shown in figure 4.2 the resulting training loss was 0.576810 and the validation loss was 0.604089, this means that on average the model is roughly 60% off the ground truth with quite a high variance.



(a) 20 epochs

(b) 2000 Epochs

Figure 4.1: Training loss for model

```
5986
5987  100%|██████████| 1994/2000 [21:54:43<03:56, 39.44s/it][INFO] EPOCH: 1994/2000
5988  Train loss: 0.579094, Val loss: 0.588920
5989
5990  100%|██████████| 1995/2000 [21:55:23<03:17, 39.56s/it][INFO] EPOCH: 1995/2000
5991  Train loss: 0.577055, Val loss: 0.589209
5992
5993  100%|██████████| 1996/2000 [21:56:03<02:38, 39.68s/it][INFO] EPOCH: 1996/2000
5994  Train loss: 0.576771, Val loss: 0.589980
5995
5996  100%|██████████| 1997/2000 [21:56:43<01:59, 39.78s/it][INFO] EPOCH: 1997/2000
5997  Train loss: 0.576947, Val loss: 0.590138
5998
5999  100%|██████████| 1998/2000 [21:57:23<01:19, 39.84s/it][INFO] EPOCH: 1998/2000
6000  Train loss: 0.578944, Val loss: 0.591172
6001
6002  100%|██████████| 1999/2000 [21:58:03<00:39, 39.91s/it][INFO] EPOCH: 1999/2000
6003  Train loss: 0.575987, Val loss: 0.590269
6004
6005  100%|██████████| 2000/2000 [21:58:43<00:00, 39.97s/it]
6006  100%|██████████| 2000/2000 [21:58:43<00:00, 39.56s/it]
6007  [INFO] EPOCH: 2000/2000
6008  Train loss: 0.576810, Val loss: 0.604089
6009  [INFO] total time taken to train the model: 79123.39s
6010
```

Figure 4.2: Terminal output when training

4.1.2 Testing

4.1.2.1 Experiment 1

Due to the lack of tuning in the pipeline: experiment 1 is non-functional. The aim of experiment 1 is to pass a batch of of AVA images and their ratings and compared the average rating of the batch before and after filter. If the average rating has increased it tells us that the pipeline is doing some for of effective filtering. It's not a well thought out experiment so the results are difficult to quantify.

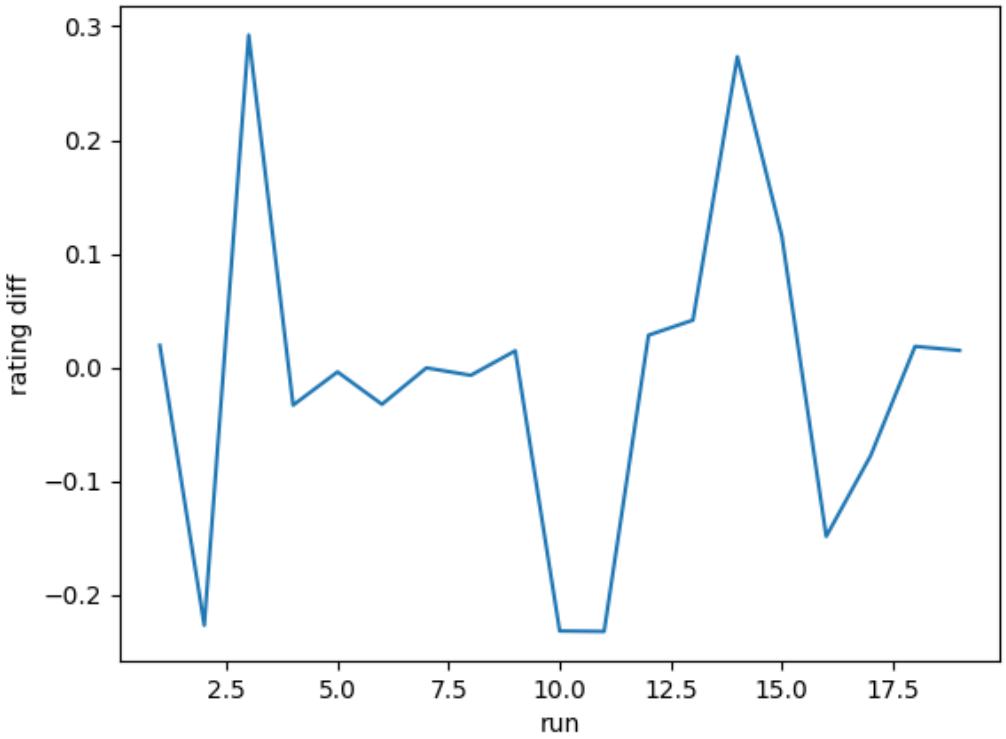


Figure 4.3: Experiment 1

4.1.2.2 Experiment 2

Due to the lack of tuning in the pipeline: experiment 2 is non-functional. The aim of this experiment was to see which filter could reduce the set the largest amount. The same set of images is passed through each filter once and the remaining images are plotted.

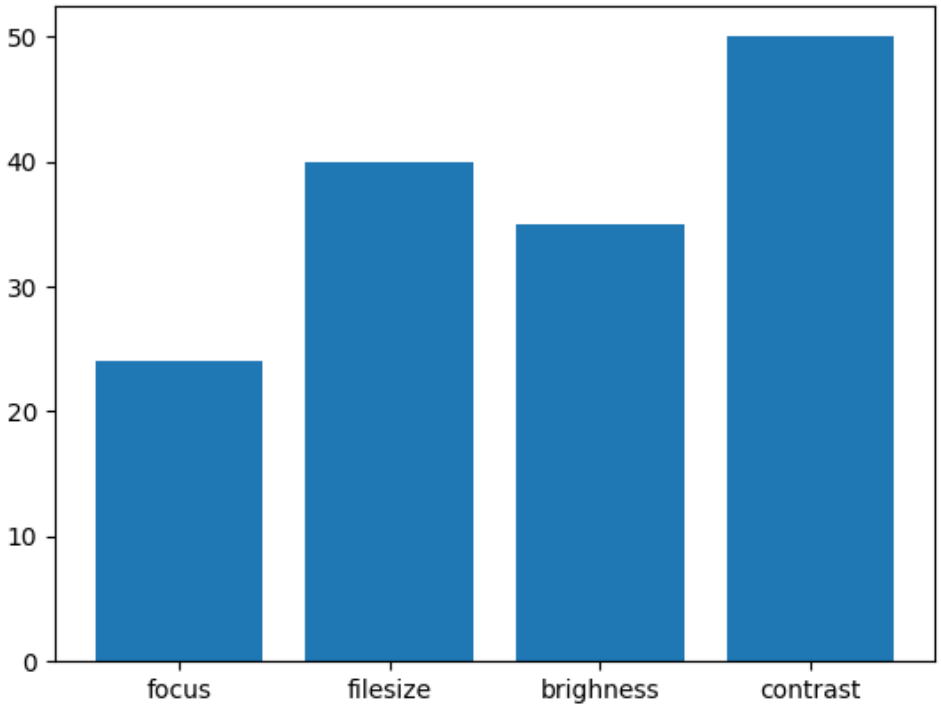


Figure 4.4: Experiment 2

4.1.2.3 Experiment 3

During the third experiment, there was an attempt to visualize how the model performed on real data. The model was used to predict the aesthetic quality rating of the frames in the videos provided by Dr Hannah Dee. The ratings were then ranked and the top 5 ranking and lowest 5 ranking entries were plotted. This experiment isn't an accurate metric of how well the model performs, but it useful to see how it performed on data that wasn't from the AVA dataset. Common accross all of the following examples is that the top 5 ranking images lay in the low-6s while the bottom 5 images lay in the high-5s. There doesn't seem to be as much variance as expected from the data, which includes a few shots in very poor lighting conditions and very blurry frames. This might possibly be an architecture issue with the training of the CNN resulting in most images being predicted in the range of 5-6.

Figure 4.5: overcastjuly-melindwr1

Figure 4.6: overcastjuly-melindwr2

Figure 4.7: sunnyaugust-camels_hump

Figure 4.8: sunnyaugust-diggers_end

Figure 4.9: sunnyaugust-drunken_druid

Figure 4.10: sunnyaugust-hippety_hop

Figure 4.11: sunnyaugust-melindwr1

Figure 4.12: sunnyaugust-melindwr2

Figure 4.13: sunnyaugust-spaghetti_junction

4.1.2.4 Experiment 4

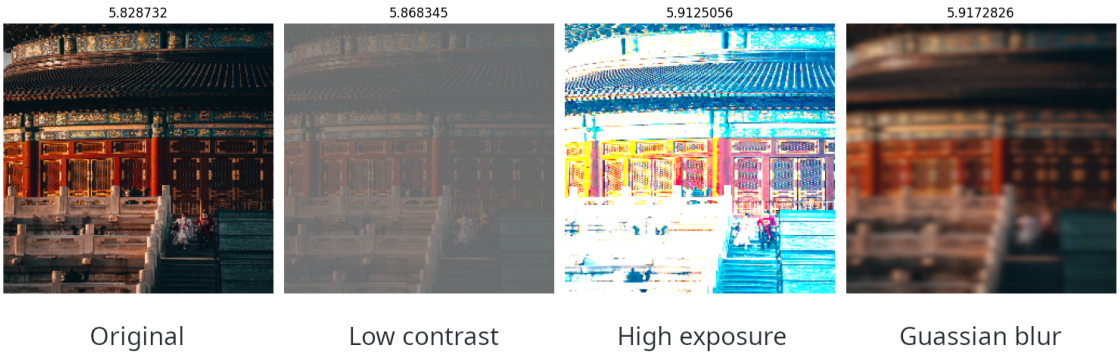


Figure 4.14: Artificial modifications and their ratings

In the 4th experiment. Aritificial data was used to test the model's performance. A base image [47] was used and then artificially edited with GIMP [48] to lower it's aesthetic quality. The first image image in figure 4.14 is the original photo, the second has had it's contrast lowered, the third has had it's exposure increased, and the fourth has had a gaussian blur applied to it. From the results it's clear to see the model has no understanding of the decline in quality in the edited picture. In this example all of the edited pictures actually score higher than the original picture.

4.2 Filters

4.2.1 Filesize

4.2.2 Contrast

4.2.3 Brightness

4.2.4 Focus

Chapter 5

Evaluation

5.1 Design Decisions

5.2 Development Process

Due to the lack of a strict development process, tasks weren't evaluated and took more effort than expected. This lead to the project pulling back its scope over the course of development. A stricter development process would have enabled the project to grow at a steadier pace and evaluating the work and time taken for each work item at the end of each week would have helped with the prioritising of tasks and adapting of scope.

5.3 Approach

The machine learning approach was chosen due to the attractiveness of the possible output rather than it's context as part of a major project. Going down the machine learning path required a lot learning a getting a good grasp of complex topics which took up a large portion of the project. In the end, more time was spent learning about CNNs than expected which lead to less time developing.

5.4 Project State

The project is considered as unfinished as there was much more planned to implement. The filters are basic in concept and in implementation and require using a manually determining a ”good” threshold and applying it. If the work were to be continued, the pipeline would process each frame of a video and use the distribution of values from a given filter to determine lower and upper bound outliers which would thn be removed automatically.

5.4.1 Implementation

The overall implementation of the project is incomplete, most of the code was hard-coded and static which made it difficult to work with. With more time the project would be restructured and refactorer to make the code easier to work with and develop further.

5.4.2 Testing

One of the biggest issues with the project is a lack of testing. Due to maining of the bad decisions with the approach of the project and the lack of a strict development process, testing was left until the very end.

5.4.3 Experiments

Many of the experiments only give artificial insight into the level of success of the project rather than providing meaningful metrics.

5.5 Report

The overall quality of the report is quite poor and many of its sections are limited in detail and scope. One of the difficulties that was presented in writing this report was a lack of experience in research-based approaches and report writing. This made it difficult to give the project a research-based context. Much of the detailed design is missing from the report and there's a lack of detail when talking about specific implementation issues.

5.6 Time management

The time management for the project was very poor and it was a struggle to finish everything that was intially planned. This mostly due to a lack detailed planning and a strict development process. This led to an ambiguity of how much work was required to finish the project which became clearer towards the deadline. Near the end of the project, it was difficult to balance project work and other areas of life and personal issues.

5.7 Futher Work

This technical work achieved in this project is a good foundation for further work. Alterations can be made to the CNN to attempt to improve it's accuracy with later

iterations. One way that this can be done is to explore fine-tuning the CNN, potentially unfreezing all the layers and training the model again.

If composition filters were implemented (golden ratio, rule of thirds, symmetry etc.) they could be used to boost the detected aesthetic features. If rule of thirds is loosely detected in an image, the option could be added to automatically crop the image in a non-destructive way[1] so that the rule of thirds has a stronger presence in the image. This could also be applied to filters like brightness and contrast but might be less successful due to the sensitivity in artificially enhancing them.

A good next step would be to refactor and restructure the code to allow plug-in filters, making filter development easier but also allowing other developers to write their own that can be dropped into the pipeline.

5.8 Personal conclusion

Overall this project has been more of a learning experience and an opportunity to work with new technology for me than a well planned experiment. I feel like I have gained a lot while also not having much to show for it.

[1] Saving the result as a new image, rather than overwriting the original

References

[1] “Papers with Code - AVA Benchmark (Aesthetics Quality Assessment).” [Online]. Available: https://paperswithcode.com/sota/aesthetics-quality-assessment-on-ava

[2] “Archillect.” [Online]. Available: https://archillect.com/

[3] “Updating Google Photos’ storage policy to build for the future,” Nov. 2020. [Online]. Available: https://blog.google/products/photos/storage-changes/

[4] B. Schoon, “G Suite’s unlimited Google Drive storage will be discontinued with Workspace,” Oct. 2020. [Online]. Available: https://9to5google.com/2020/10/08/google-workspace-drive-storage-limits/

[5] “PREVIEW - Plan your Instagram – Apps on Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.sensio.instapreview&hl=en_GB&gl=GB

[6] “UNUM — World’s easiest marketing tool.” [Online]. Available: https://www.unum.la/

[7] “Why Your Instagram Profile is the New Home Page.” [Online]. Available: https://later.com/blog/instagram-profile-home-page/

[8] K. Ding, K. Ma, and S. Wang, “Intrinsic Image Popularity Assessment,” *Proceedings of the 27th ACM International Conference on Multimedia*, pp. 1979–1987, Oct. 2019, arXiv: 1907.01985. [Online]. Available: http://arxiv.org/abs/1907.01985

[9] D. Depoorter and M. Pinckers, “Trophy Camera, 2017-2022.” [Online]. Available: https://driesdepoorter.be/trophy-camera/

[10] ——, “”trophy.camera”.” [Online]. Available: https://trophy.camera/

[11] “Home | World Press Photo.” [Online]. Available: https://www.worldpressphoto.org/

[12] “2007 Spencer Platt WY | World Press Photo.” [Online]. Available: https://www.worldpressphoto.org/collection/photo-contest/2007/spencer-platt/1

[13] “Golden ratio: A beginner’s guide | Adobe.” [Online]. Available: https://www.adobe.com/creativecloud/design/discover/golden-ratio.html

[14] “Symmetry in Photography: The Ultimate Guide to Using Symmetry in Your Photos,” Sept. 2020. [Online]. Available: https://www.photoworkout.com/symmetry-in-photography/

[15] “What is the Rule of Thirds and How to Use it to Improve Your Photos,” Mar. 2018. [Online]. Available: https://photographylife.com/the-rule-of-thirds

[16] “Understanding shallow depth of field photography | Adobe.” [Online]. Available: https://www.adobe.com/creativecloud/photography/discover/shallow-depth-of-field.html

[17] A. U. W. Team, “Current Modules by Department : Modules , Aberystwyth University,” last Modified: 2022-05-04. [Online]. Available: https://www.aber.ac.uk/en/modules/deptcurrent/CS36220/

[18] “Python.org.” [Online]. Available: https://www.python.org/

[19] Google Brain Team, “TensorFlow.org.” [Online]. Available: https://www.tensorflow.org/

[20] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan, “PyTorch.org.” [Online]. Available: https://www.pytorch.org

[21] “Keras: the Python deep learning API.” [Online]. Available: https://keras.io/

[22] N. Murray, L. Marchesotti, and F. Perronnin, “AVA: A large-scale database for aesthetic visual analysis,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012, pp. 2408–2415, iSSN: 1063-6919.

[23] “DPChallenge - A Digital Photography Contest.” [Online]. Available: https://www.dpchallenge.com/

[24] Fing, “AVA Dataset,” Feb. 2022, original-date: 2016-11-13T02:20:32Z. [Online]. Available: https://github.com/imfing/ava_downloader

[25] S. Kong, X. Shen, Z. Lin, R. Mech, and C. Fowlkes, “Photo Aesthetics Ranking Network with Attributes and Content Adaptation,” *arXiv:1606.01621 [cs]*, July 2016, arXiv: 1606.01621. [Online]. Available: http://arxiv.org/abs/1606.01621

[26] “Flickr.” [Online]. Available: https://www.flickr.com

[27] “skimage — skimage v0.19.2 docs.” [Online]. Available: https://scikit-image.org/docs/stable/api/skimage.html

[28] “Detecting low contrast images with OpenCV, scikit-image, and Python,” Jan. 2021. [Online]. Available: https://www.pyimagesearch.com/2021/01/25/detecting-low-contrast-images-with-opencv-scikit-image-and-python/

[29] “Blur detection with OpenCV,” Sept. 2015. [Online]. Available: https://www.pyimagesearch.com/2015/09/07/blur-detection-with-opencv/

[30] “OpenCV Fast Fourier Transform (FFT) for blur detection in images and video streams,” June 2020. [Online]. Available: https://www.pyimagesearch.com/2020/06/15/opencv-fast-fourier-transform-fft-for-blur-detection-in-images-and-video-streams/

[31] Stanford University School of Engineering, “Lecture 8 | Deep Learning Software,” Aug. 2017. [Online]. Available: https://www.youtube.com/watch?v=6SlgtELqOWc

[32] ——, “Lecture 9 | CNN Architectures,” Aug. 2017. [Online]. Available: https://www.youtube.com/watch?v=DAOcjicFr1Y

[33] “Training with PyTorch — PyTorch Tutorials 1.11.0+cu102 documentation.” [Online]. Available: https://pytorch.org/tutorials/beginner/introyt/trainingyt.html

[34] “PyTorch: Training your first Convolutional Neural Network (CNN),” July 2021. [Online]. Available: https://www.pyimagesearch.com/2021/07/19/pytorch-training-your-first-convolutional-neural-network-cnn/

[35] H.-J. Lee, K.-S. Hong, H. Kang, and S. Lee, “Photo Aesthetics Analysis via DCNN Feature Encoding,” *IEEE Transactions on Multimedia*, vol. 19, no. 8, pp. 1921–1932, Aug. 2017, conference Name: IEEE Transactions on Multimedia.

[36] “PyTorch: ResNet.” [Online]. Available: https://pytorch.org/hub/pytorch_vision_resnet/

[37] “ImageNet.” [Online]. Available: https://www.image-net.org/index.php

[38] “Keras, Regression, and CNNs,” Jan. 2019. [Online]. Available: https://www.pyimagesearch.com/2019/01/28/keras-regression-and-cnns/

[39] “Regression with Keras,” Jan. 2019. [Online]. Available: https://www.pyimagesearch.com/2019/01/21/regression-with-keras/

[40] “House Prices - Advanced Regression Techniques.” [Online]. Available: https://kaggle.com/competitions/house-prices-advanced-regression-techniques

[41] “PyTorch: Transfer Learning and Image Classification - PyImageSearch.” [Online]. Available: https://pyimagesearch.com/2021/10/11/pytorch-transfer-learning-and-image-classification/

[42] “PyTorch for AMD ROCm™ Platform now available as Python package.” [Online]. Available: https://pytorch.org/blog/pytorch-for-amd-rocm-platform-now-available-as-python-package/

[43] “ROCm™: Machine Learning.” [Online]. Available: https://www.amd.com/en/graphics/servers-solutions-rocm-ml

[44] “Create a VM with attached GPUs | Compute Engine Documentation | Google Cloud.” [Online]. Available: https://cloud.google.com/compute/docs/gpus/create-vm-with-gpus

[45] “Slurm Workload Manager - Overview.” [Online]. Available: https://slurm.schedmd.com/overview.html

[46] “Anaconda | The World’s Most Popular Data Science Platform.” [Online]. Available: https://www.anaconda.com/

[47] Unsplash, “Photo by ZHENYU LUO on Unsplash.” [Online]. Available: https://unsplash.com/photos/mhvL46eQis0

[48] “GIMP.” [Online]. Available: https://www.gimp.org/

[49] S. Kong, “Photo Aesthetics Ranking Network with Attributes and Content Adaptation,” Jan. 2022, original-date: 2016-06-05T06:08:10Z. [Online]. Available: https://github.com/aimerykong/deepImageAestheticsAnalysis

[50] Intel Corporation, Willow Garage, and Itseez, “OpenCV.” [Online]. Available: https://opencv.org

[51] “Image Quality Assessment,” Feb. 2022, original-date: 2018-06-12T14:46:09Z. [Online]. Available: https://github.com/idealo/image-quality-assessment

[52] “Most used social media 2021.” [Online]. Available: https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

[53] “There’s a way to pick the absolute best images for your content: Apply AI | TechCrunch.” [Online]. Available: https://techcrunch.com/2020/10/01/theres-a-way-to-pick-the-absolute-best-images-for-your-content-apply-ai/?guccounter=1

Appendices

Appendix A

Third-Party Code and Libraries

The following third party code and libraries were used to support this project[1]:

- **opencv-python** - provides many useful functions for working with images and video
- **numpy** - provides multi-dementional arrays and mathematical functions which can be applied to said arrays
- **matplotlib** - provides functions for plotting and exporting graphs
- **pandas** - provides advanced table data structures and mathematical functions to apply to them
- **PyYAML** - provides functions to read and write YAML files
- **scikit-image** - provides many functions for analysing images
- **tqdm** - provides progress bars
- **flake8** - (used in testing) provides linting, helping find inconsistencies with style
- **pytest** - (used in testing) a python-based testing framework
- **pytest-cov** - (used in testing) a plugin for pytest which will provide a report on test coverage
- **unittest** - (used in testing) the mock module of unittest was used to use mocks in testing
- **mypy** - (used in testing) provides linting for type checking
- **torch** - machine learning framework that provides tools for building and training models
- **torchvision** - library that contains datasets and models for neural networks

[1] This list does not include the libraries pulled down as dependencies of entries in this list.

Appendix B

Code Examples

2.1 Woodpecker pipelines

2.1.1 Lint pipeline

The lint pipeline was use to automatically report back on any linting issues with the code after a commit.

```
pipeline:
  flake8:
    image: python:3.8
    commands:
      - python3.8 -m pip install flake8
      - flake8 src/
  mypy:
    image: python:3.8
    commands:
      - python3.8 -m pip install mypy
      - mypy src/
  isort:
    image: python:3.8
    commands:
      - python3.8 -m pip install isort
      - isort --diff src/
branches: dev
```

2.1.2 Test pipeline

```
pipeline:
  unit-tests:
    image: python:3.8
```

```
    commands:
      - apt update -y && apt install libgl1 -y
      - python3.8 -m pip install -r ./requirements.txt
      - python3.8 -m pip install -r ./requirements_dev.txt
      - python3.8 -m pip install -e .
      - pytest test/ -v
branches: dev
```

2.1.3 Configuration file

```
---
# Configuration file for the autophographer tool

# List of filters to apply in order
# Note: Possible filters include: brightness, filesize, contrast, focus
filters:
  - brightness
  - filesize
  - contrast
  - focus

# Whether or not to apply CNN ranking
CNNrank: False

# Ignore video files and don't bother processing them into frames
# Note: Useful if directory contains original video and indivual
#   frames from video (prevents processing the same frames more than once)
ignore_video: True

# Options for brightness filter
brightness_options:
  threshold: 0.25

# Options for filesize filter
filesize_options:
  threshold: 0.35

# Options for contrast filter
contrast_options:
  threshold: 0.35

# Options for focus filter
focus_options:
  threshold: 0.5
...
```

2.1.4 pil_loader() method

This method was taken from a GitHub issue comment
https://github.com/python-pillow/Pillow/issues/835#issuecomment-53999355

```
def pil_loader(path):
    with open(path, 'rb') as f:
            image = Image.open(f)
    return image.convert('RGB')
```