Software Technology 1
Capstone Project

Student ID number & Student Name	Mitchell Starr - u3260016
Unit name	Software Technology 1
Unit number	4483
Unit Tutor	Girija Chetty
Assignment name	ST1 Capstone Project – Semester 2 2023
Due date	29/10/2023
Date submitted	

Student declaration

I certify that the attached assignment is my own work. Material drawn from other sources has been appropriately and fully acknowledged as to author/creator, source and other bibliographic details.

Table of Contents

Github Repository Link

https://github.com/Dysteiq/ST1-Capstone-Project

Introduction

This report describes the details and process of Software Technology 1 UG Capstone Project. The project was to create a deep learning model which could classify images using computer vision. The exact project is titled Lions or Cheetahs - Image Classification. The task at hand was to develop a model using deep learning which could then later be deployed and used to classify images of either lions or cheetahs into their respective categories. Deep learning is a subsection of machine learning that can be used to detect patterns and perform classification tasks. Deep learning can be used for a very wide variety of things, anything from self-driving vehicles to translating languages. This report will go into further detail about the steps of deep learning in the subsequent sections of this report.

Dataset Description

The dataset I was assigned was Lions and Cheetahs image classification. It contains 100 lion images and 100 cheetah images. This dataset is used for image classification including developing and testing deep learning algorithms, along with comparing the effectiveness of those deep learning algorithms. It is a publicly available dataset on the Kaggle repository. The dataset is quite poor quality however as it contains many images that are neither lion or cheetah, it contains monkeys, giraffes, statues and other animals and objects, this might make the model worse, due to the poor quality dataset that will be used to train it.

Methodology

This section of the report will explain the methodology used to prepare, create, test and deploy the model.

Stage 1: Exploratory Data Analysis

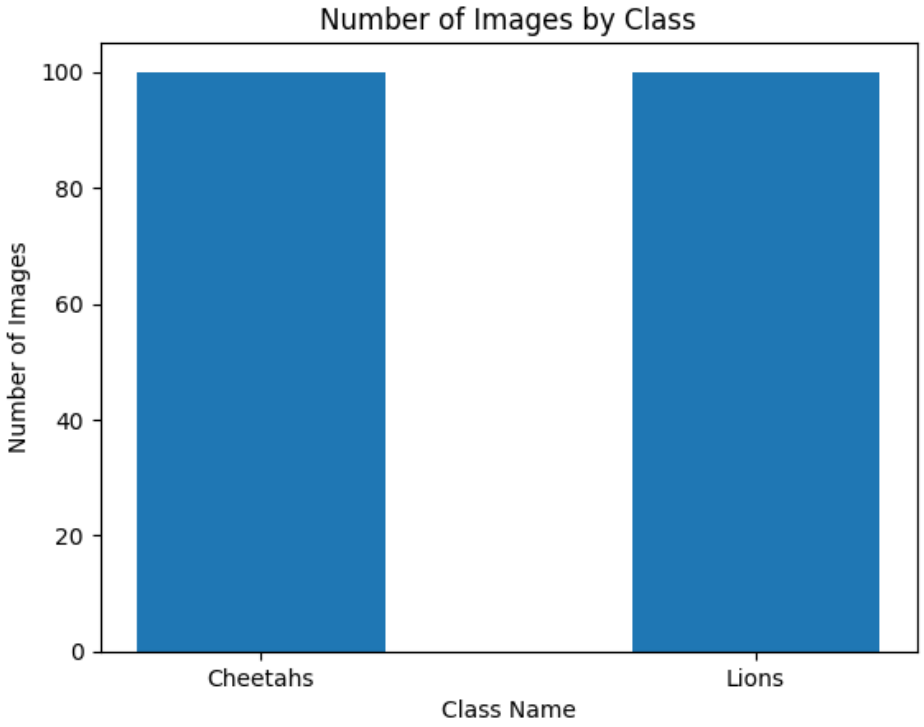
EDA is the process of analysing data to determine their main different characteristics. This is important to define a ground level and familiarise ourselves with the dataset. To perform EDA we analyse different statistics of the images, this can also include augmenting the datasets before analysing, to flesh out our dataset a little more. However it's important to not augment the images too much otherwise the data will be irrelevant to the original classes. For this reason I did not augment my data for the EDA stage. The statistics I analysed between lions and cheetahs came from these 5 questions:

- Can we display a random sample of the dataset?

- Can we determine the total number of images in each dataset?
- Can we graph the distribution of image sizes in each category? (width x height)
- Can we graph the distribution of contrast and brightness for each category?
- Can we display the average colour distribution for each category?

Total Number of Images

It’s important to be able to graph the total number of images to ensure all images are loaded correctly, and there's no corrupted files or other errors. Displayed below is a graph of the total images in each class. As shown in the bar graph below, all images have been loaded correctly. The python libraries used with this program were os; to join the directories, and Matplotlib; to create the graph.



Random Sample

The purpose of a random sample is to sample the dataset, quickly viewing a user specified number of random images. The program is very simple, it asks the user for a specified number of images, and then displays that number of images from each class back to the user. A successful test, showing one of each image is pictured below. The python libraries used for this program were os; to join two directories together, openCV; to load the images and random; to choose the random sample of images.

Please enter the number of images: 1
Cheetahs

Lions

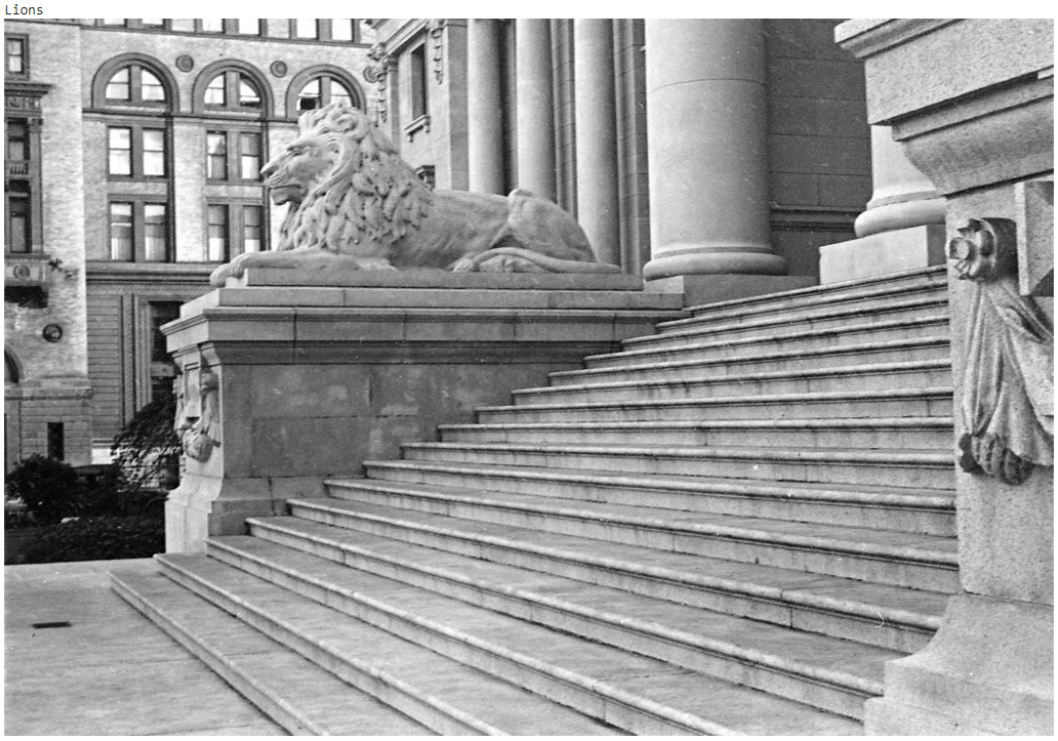
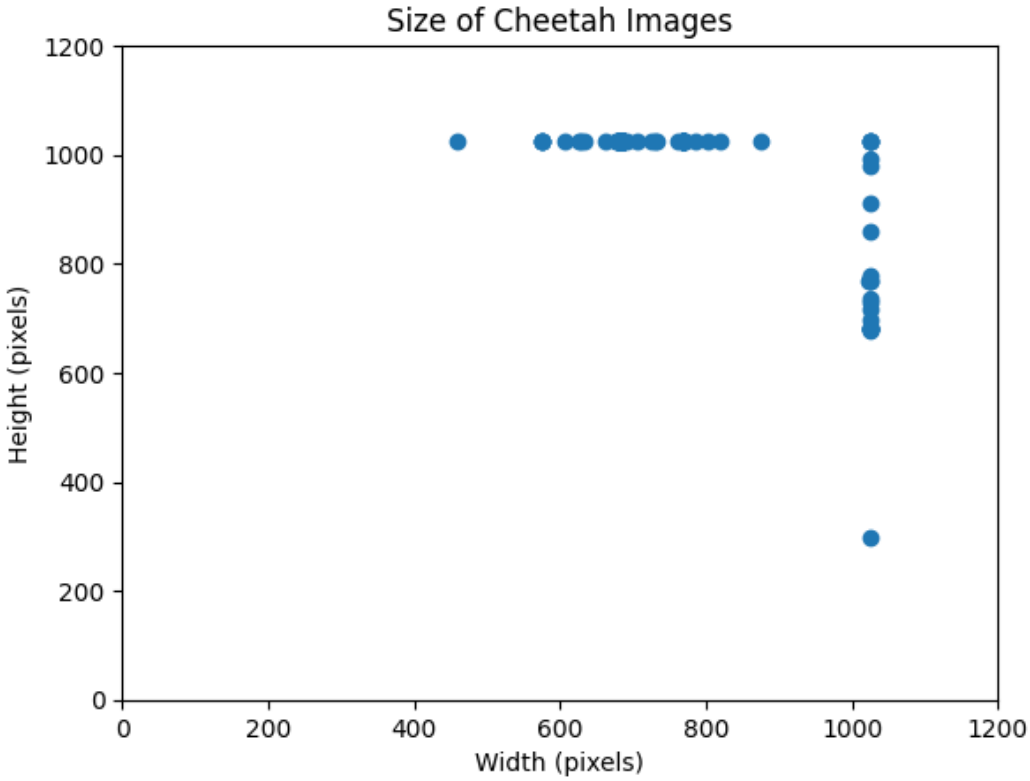
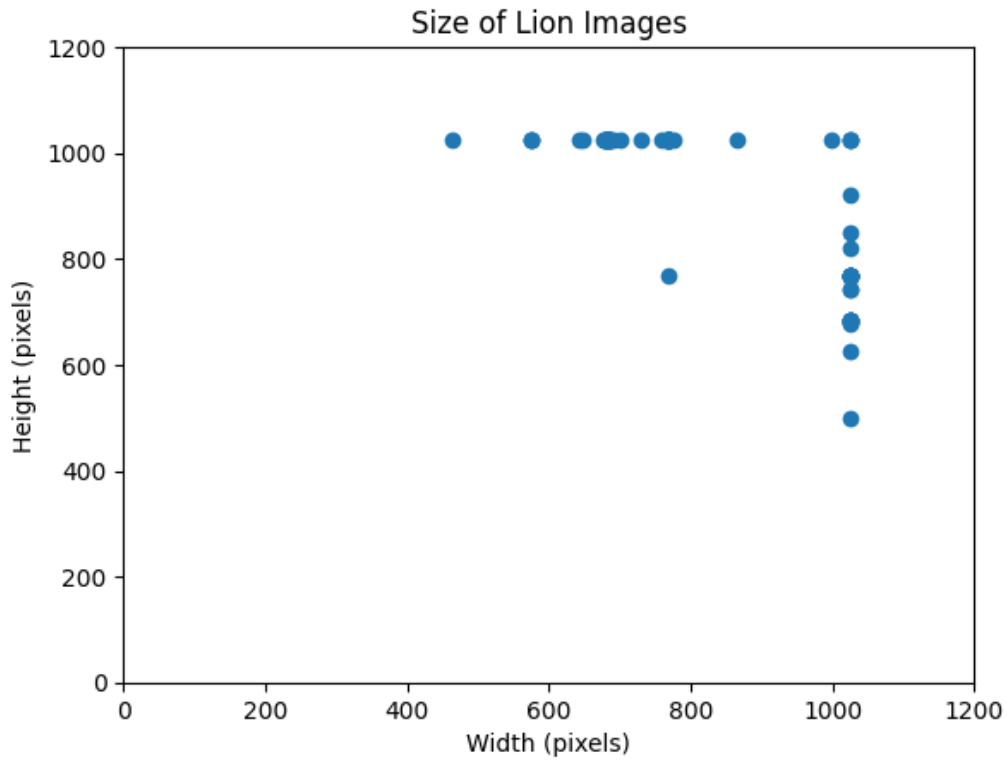


Image Sizes

I graphed the image sizes in two forms, once with a scatterplot to show the width and height of the images, and then once by multiplying width and height together to get the total number of pixels, then graphing that in a histogram we can see the distribution of total pixels across the two datasets. Curves of best fit were also included, however they become more relevant with later image analysis, not so much with the dimensions due to many of the images following a 2:3 or 3:4 aspect ratio, and being already resized to have their maximum side be 1024 pixels. This also affected the scatterplots, as it gave straight lines along each of the 1024 lines for height and width. The python libraries used for this program were os; to join the directories, OpenCV; to read the images, Matplotlib; to graph the results and Seaborn; to create the line of best fit.




Size of Lion Images
Height (pixels)
Width (pixels)
0
200
400
600
800
1000
1200

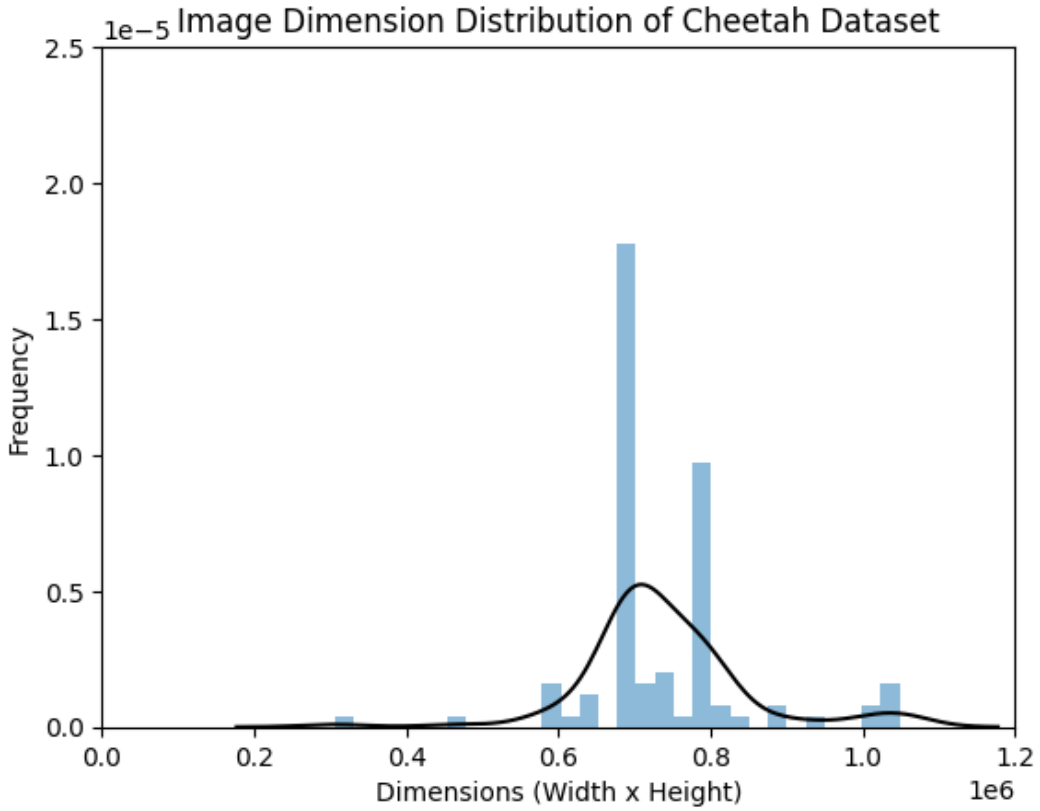
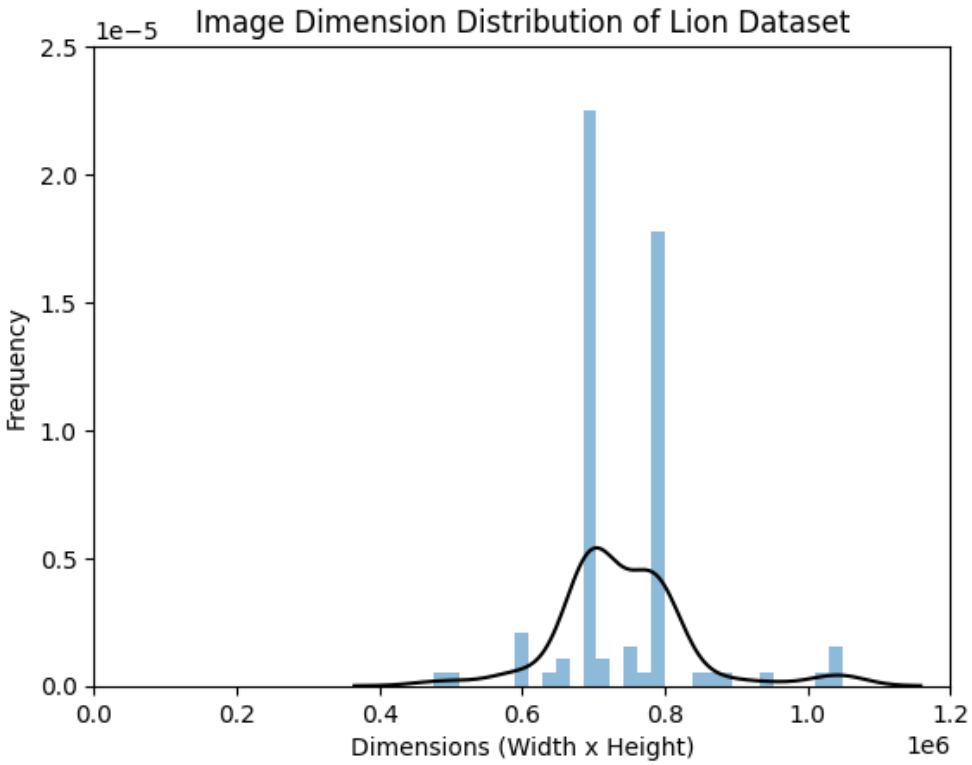
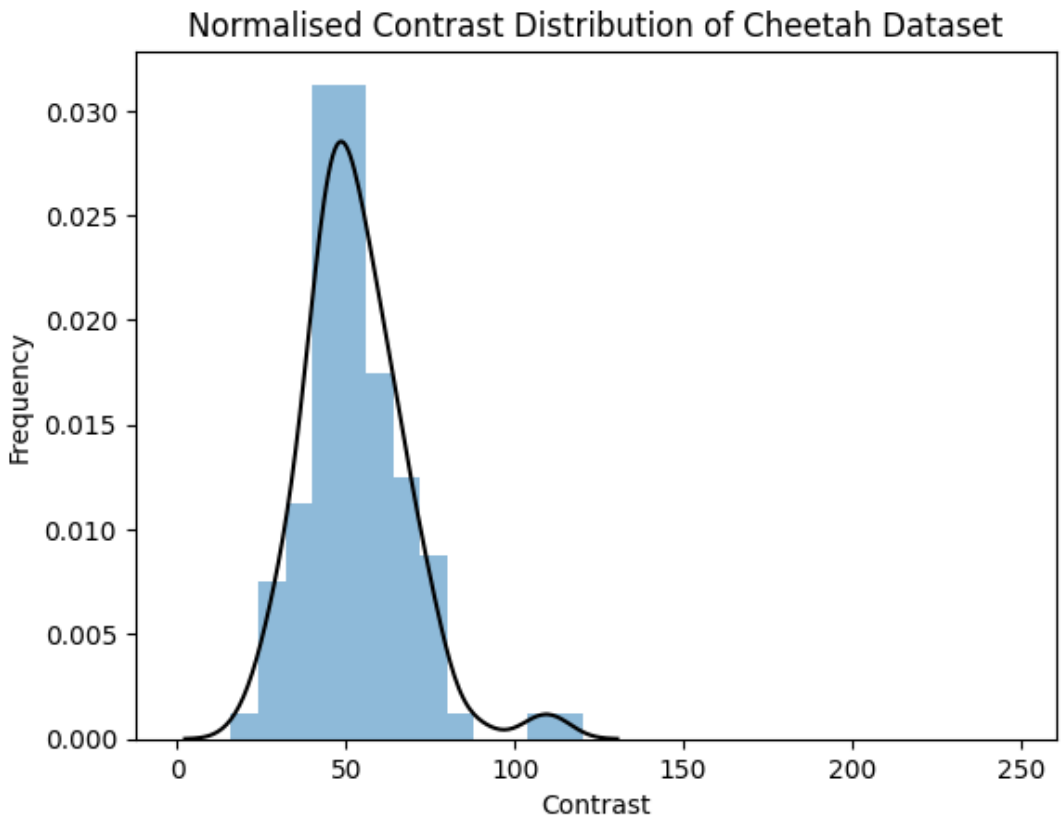


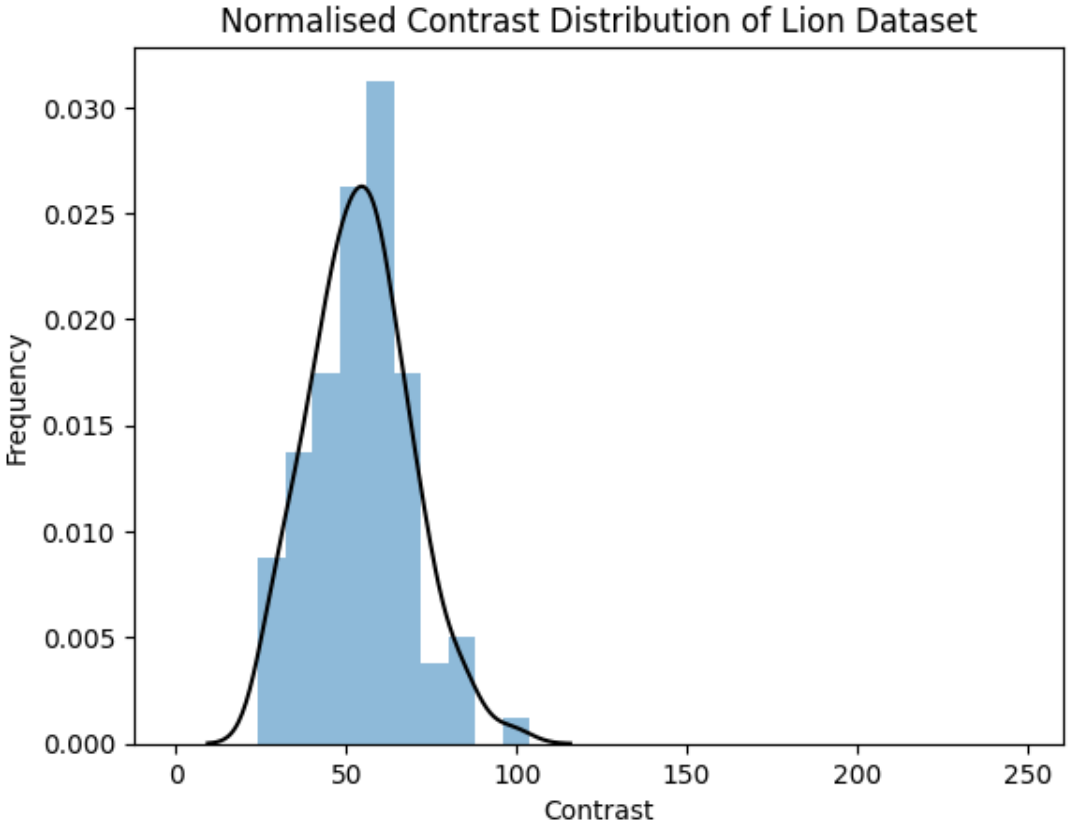
Image Dimension Distribution of Cheetah Dataset
1e−5
Frequency
Dimensions (Width x Height)
1e6
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0
1.2

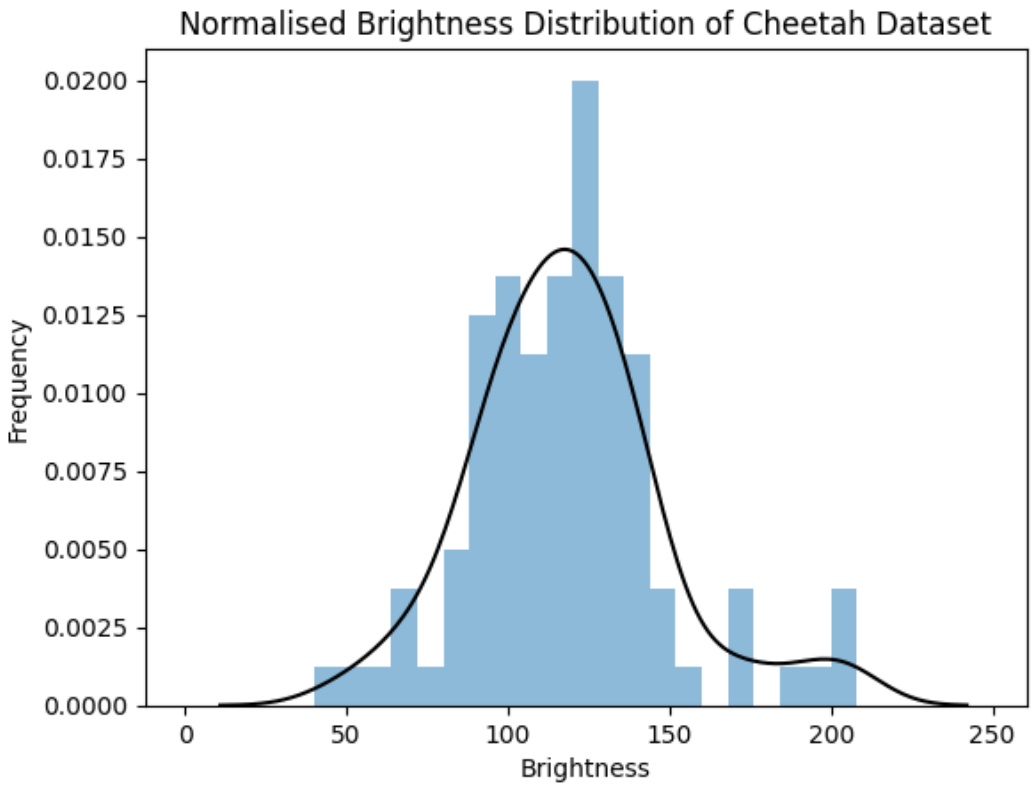


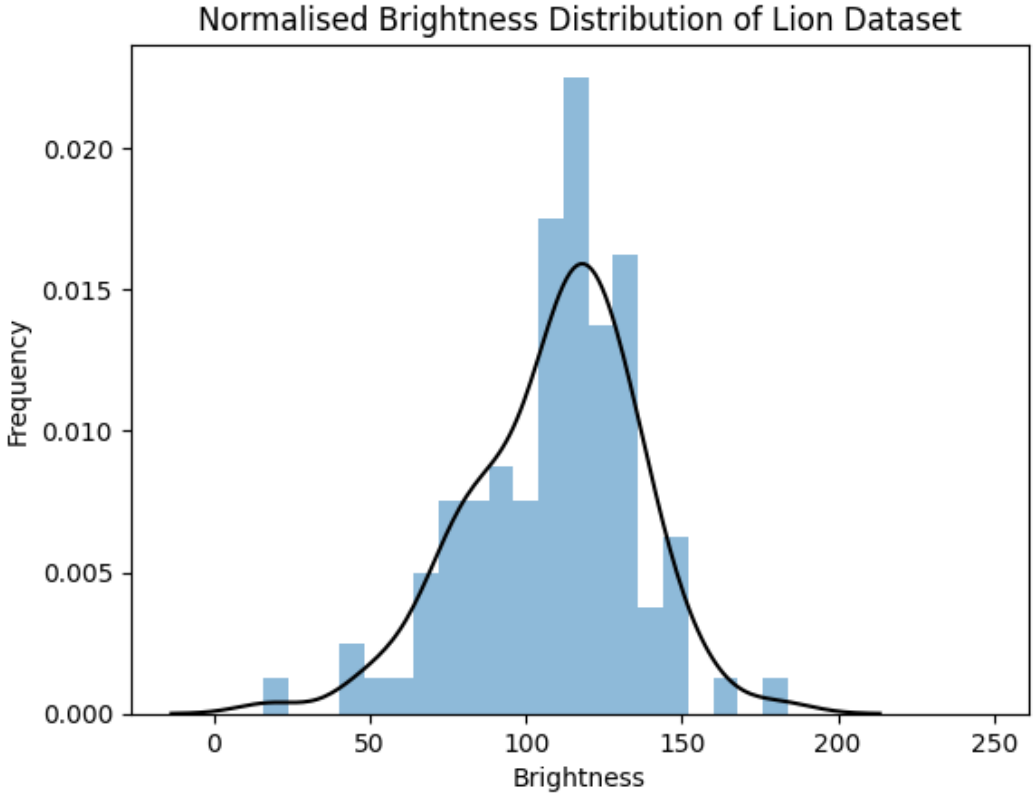
Average Contrast and Brightness

I analysed the average contrast and brightness over the images to determine some differences between the two datasets using python libraries to calculate and visualise the data. First, I'll detail contrast, it was calculated by first converting the colour images to grayscale, which made them single channel, then determining the highest and lowest brightness pixel and then subtracting the lower from the higher, a higher contrast value comes from a larger difference between these two values. Surprisingly the average contrasts were very similar between the two datasets. As for brightness, I used the grayscale images from before and calculated their average value. I found that the lion images were slightly brighter on average than the cheetahs. The python libraries used for this program were os; to join the directories, OpenCV; to read the images, NumPy; to calculate the array for the bins for the histogram, Matplotlib; to graph the results and Seaborn; to create the line of best fit. This data analysis was crucial to prepare myself for creating and training the model, as it was important to see the image statistics before any augmentation.


Normalised Contrast Distribution of Cheetah Dataset
Frequency
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
50
100
150
200
250
Contrast

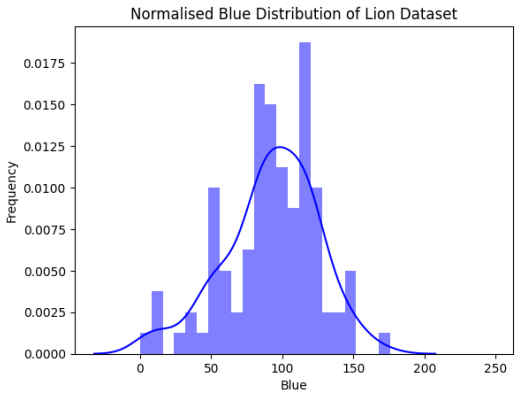
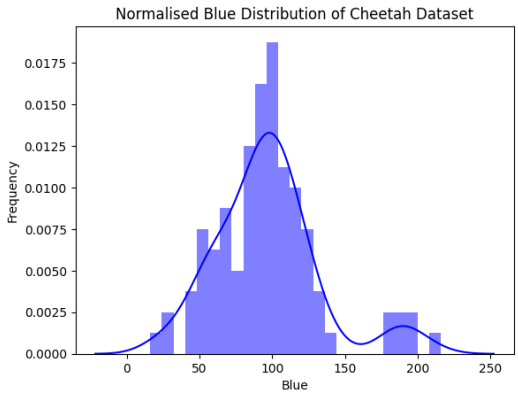
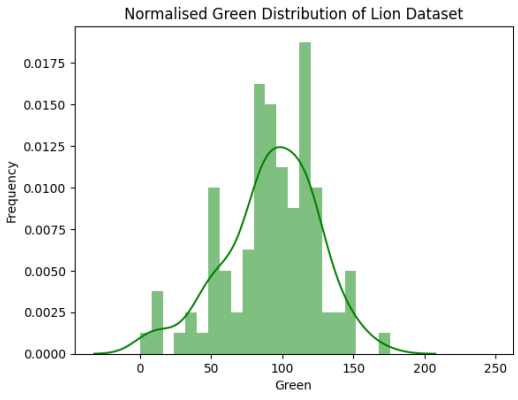
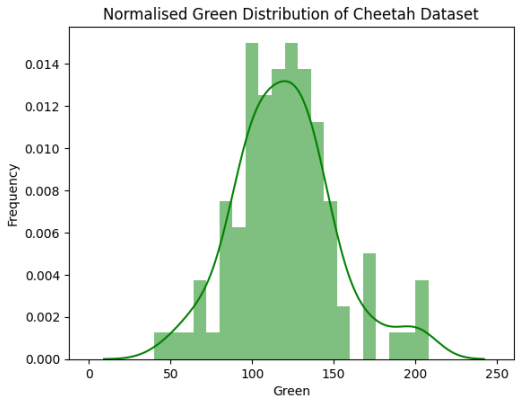

Normalised Contrast Distribution of Lion Dataset
Frequency
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
50
100
150
200
250
Contrast

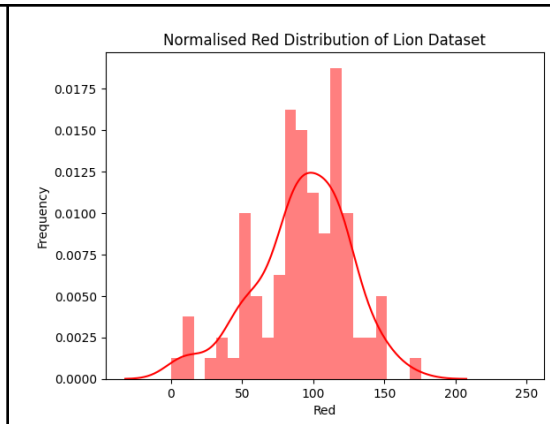
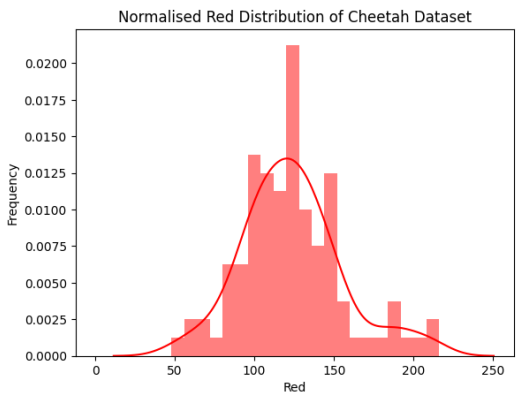

Normalised Brightness Distribution of Cheetah Dataset
Frequency
0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
0
50
100
150
200
250
Brightness

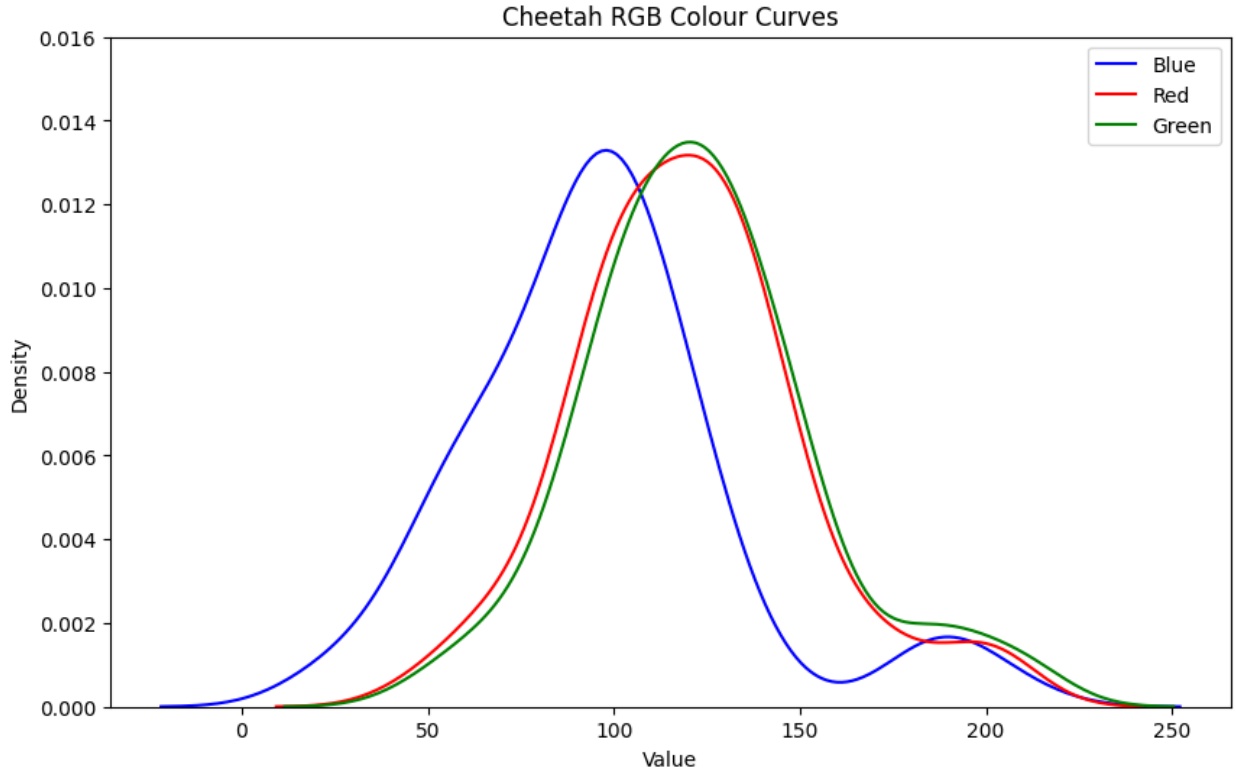

Normalised Brightness Distribution of Lion Dataset
Frequency
0.020
0.015
0.010
0.005
0.000
0
50
100
150
200
250
Brightness

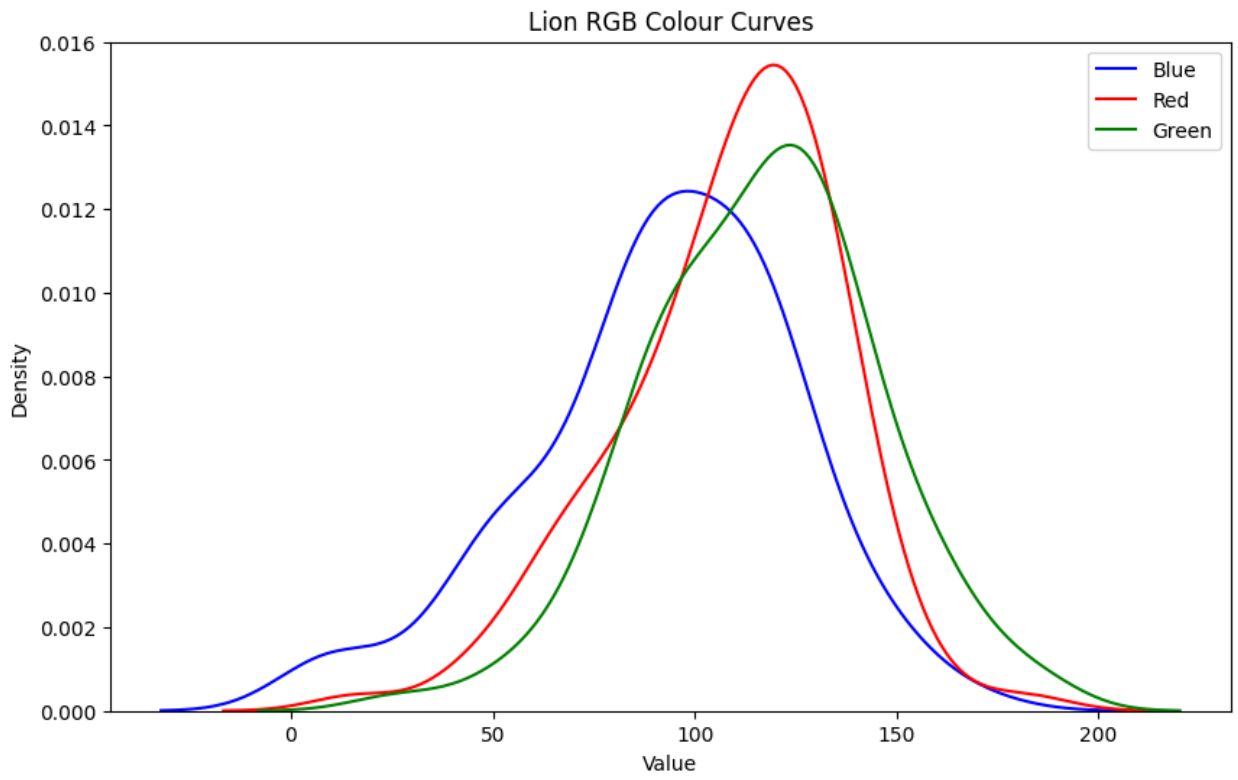
Average RGB Colour Distribution

I calculated the average colour distribution by splitting the blue, green, red image into its respective colours, then calculating the mean value for each colour over the entire image, and appending that to a list, then finally graphing the distribution of each colour. As you can see in the graphs below, lions had higher values on average, which is likely due to their higher brightness. Lions also had more red on average, which I suspect is due to their manes contributing a large amount of red, and also taking up space which would be replaced with either grass or sky, both non-red colours. The python libraries used for this program were os; to join the directories, OpenCV; to read the images, NumPy; to calculate the array for the bins for the histogram and calculate the mean of the colour values, Matplotlib; to graph the results and Seaborn; to create the line of best fit.

	Lion	Cheetah
Blue		
Green		

Red	 Normalised Red Distribution of Lion Dataset Frequency Red	 Normalised Red Distribution of Cheetah Dataset Frequency Red


Cheetah RGB Colour Curves
Blue
Red
Green
Density
Value



Stage 2: Predictive Data Analysis

- Provide details about the neural network architecture, including the number of layers, type of layers (convolutional, pooling, fully connected, etc.), and any specific hyperparameters you used.
- Describe how you split the data into training, validation, and test sets.
- Explain the training process, including the optimizer used, learning rate, batch size, and the number of epochs.
- Include any data augmentation techniques you applied during training.

The model architecture I used was using Keras and TensorFlow on Google Colab. I trained the model using repeated applications of 10 epochs, saving or discarding based on how well the model improved. I augmented the data using these parameters:

```
- rescale=1./255,
-     rotation_range=40,
-     brightness_range=[0.2, 1.8],
-     shear_range=0.2,
-     zoom_range=0.2,
-     horizontal_flip=True,
```

```
-       vertical_flip=True
```

This is done to artificially increase the number of images in the dataset to improve its effectiveness when training the model. I did not augment the colour of the images due to having such a small dataset, I thought it might cause overfitting by making the images too different to real lions and cheetahs, as they are quite similar in their own right, both being members of large cats. The image size I used was 128x128 and it ran for 10 epochs each time, for a total of about 60 epochs. The model will then display the loss and accuracy for each model to the user. A successful training and validation test can be seen below.

Training Program Execution

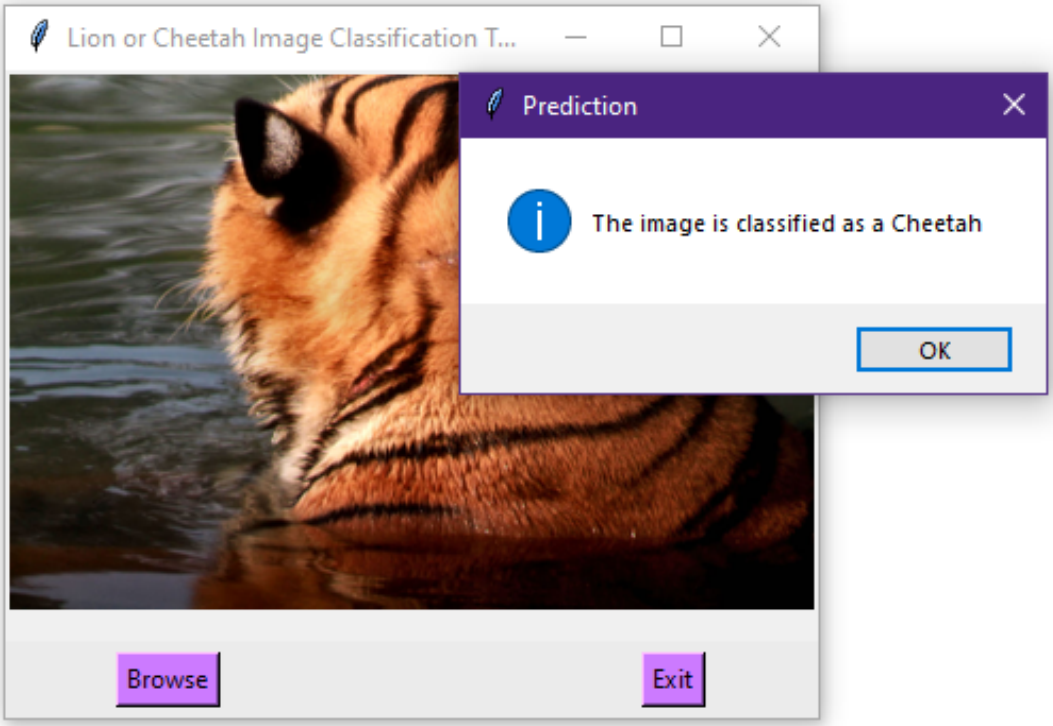
```
What do you want to do:
A: Train New Model (Overwrites saved model)
B: Fine Tune Current Model
C: Validate Current Model
X: Exit
B
Model successfully loaded
Found 200 images belonging to 2 classes.
Epoch 1/10
4/4 [==============================] - 23s 4s/step - loss: 0.6106 - accuracy: 0.6600
Epoch 2/10
4/4 [==============================] - 5s 790ms/step - loss: 0.6215 - accuracy: 0.6250
Epoch 3/10
4/4 [==============================] - 4s 779ms/step - loss: 0.6493 - accuracy: 0.6000
Epoch 4/10
4/4 [==============================] - 4s 889ms/step - loss: 0.5998 - accuracy: 0.6800
Epoch 5/10
4/4 [==============================] - 4s 773ms/step - loss: 0.6070 - accuracy: 0.6800
Epoch 6/10
4/4 [==============================] - 4s 857ms/step - loss: 0.6298 - accuracy: 0.6700
Epoch 7/10
4/4 [==============================] - 5s 1s/step - loss: 0.5912 - accuracy: 0.6750
Epoch 8/10
4/4 [==============================] - 4s 763ms/step - loss: 0.6188 - accuracy: 0.6850
Epoch 9/10
4/4 [==============================] - 4s 820ms/step - loss: 0.6120 - accuracy: 0.6200
Epoch 10/10
4/4 [==============================] - 5s 2s/step - loss: 0.6116 - accuracy: 0.7100
What do you want to do:
A: Save Model
B: Discard Changes
A
What do you want to do:
A: Yes, overwrite the old model
B: No, take me back
A
```

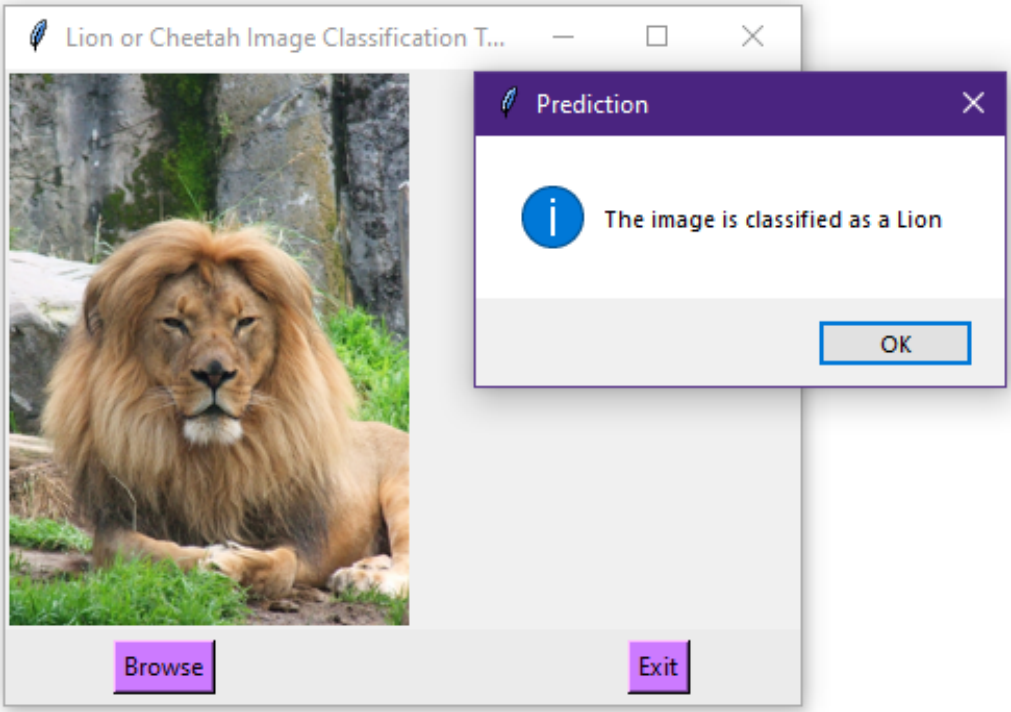
Validation Program Execution

```
What do you want to do:
A: Train New Model (Overwrites saved model)
B: Fine Tune Current Model
C: Validate Current Model
X: Exit
C
Model successfully loaded
Found 80 images belonging to 2 classes.
2/2 [==============================] - 5s 4s/step - loss: 0.4714 - accuracy: 0.8125
Average Loss: 0.4714
Accuracy: 81.25%
```

Stage 3: Deployment/Implementation

To deploy the model, I developed a user-friendly GUI application using tkinter, which significantly enhances the user experience compared to a console-based program. The program gives the user an easily navigated interface with a ‘Browse’ and ‘Exit’ button. The ‘Browse’ button will open a file open dialog box where the user can browse till they find an image. The program will then return its prediction on which class it believes the image belongs to and display it back to the user using a message box. The program also uses the ‘Exit’ button to exit the program, without requiring the ‘X’ in the top of the window. The GUI simplifies the user experience compared to a console program which most users would not be very experienced with, as that would require command line inputs to browse to an image file. It also gives the user a way to browse to their own images, rather than require the path to the image to be hard-coded or located through the command line as mentioned before. The message box gives the user a clear output which is not in the console. The image is also displayed in the tkinter window to further enhance the user experience by making it more obvious which image they have chosen.


Lion or Cheetah Image Classification T...
Prediction
The image is classified as a Cheetah
OK
Browse
Exit


Lion or Cheetah Image Classification T...
Prediction
The image is classified as a Lion
OK
Browse
Exit

Discussion

The strengths and weaknesses of the model and training program will be detailed in the table below:

Deployed Model Program

Strengths	Weaknesses
User friendly GUI	Can only classify one image at a time
Has error handling in case model doesn't load correctly or image is not the correct file type	Not 100% success rate as the dataset was quite poor, containing non-cheetah or lion images such as giraffes and I only did about 60 total epochs
High success rate, which can be validated using the training and validating program	
Models can easily be updated to better ones by changing the model name at the top of the code	
Displays output in an obvious message box	

Training and Validating Program

Strengths	Weaknesses
Augments data to ensure a well trained model	Does not have a GUI, requires the console for inputs and outputs
Can train both old and new models and then save those models	All file-paths are hard coded so would not work without changing the locations for other people
Can validate the current model	Did not have separate images for training and validation, only a subsection of the dataset.
Has error handling in case files cannot be found	

As detailed above, the GUI could be improved by allowing the user to input multiple images at a time, which would help the user greatly in case of bulk image classification. The actual model

can be improved by running more epochs, as only around 60 were completed in total, due to not wanting to overfit the data, however I could have increased it further. The dataset was also quite poor quality as it contained multiple images that were not part of either lions or cheetahs, there were giraffes, monkeys, statues and other animals or objects, this made it very hard for the program to tell the difference between lions and cheetahs, especially female lions, which lack a mane and therefore look quite similar to cheetahs. Also if the dataset had more than 100 images of each category it would have been good to separate the training and validation datasets in two, so that the model could be validated on completely new data. It also would have been good to include a ‘confidence’ score for when the image was classified, that way it would be clear whether the image was close to the threshold between lion and cheetah, or far from it.

Conclusion

With some small improvements this model would be significantly better. Features such as classifying multiple images at once, or a displayed confidence score would be great additions. Multi-image classification would revolutionise the model by allowing the simultaneous classification of images, improving its efficiency and usefulness for a wider range of applications. The confidence score would be excellent for eliminating false results, for example the image could have a ‘unsure’ category which it classifies the images into when the confidence score is low. This would also help with real world applications as it can prioritise the high certainty results for applications such as determining diseases in patients, any high certainty diseases can be flagged, while the uncertain ones can be reviewed by a human specialist. Overall, the program is able to tell the images apart most of the time, with some struggles on female lions, and a high error rate for the non-lion or non-cheetah images, which is to be expected, since they aren’t lions or cheetahs. This could be reduced by increasing the number of images in the datasets and also eliminating the images which are neither lion nor cheetah.

References

[1] MIKOŁAJFISH99, “Lions or cheetahs - image classification,” Kaggle, https://www.kaggle.com/datasets/mikoajfish99/lions-or-cheetahs-image-classification/data (accessed Oct. 28, 2023).

[2] sambler, “(python?) how to get the image editor to open a random image from a folder,” Blender Stack Exchange, https://blender.stackexchange.com/questions/76396/python-how-to-get-the-image-editor-to-open-a-random-image-from-a-folder (accessed Oct. 28, 2023).

[3] D. Rausch, “EDA for image classification,” Medium, https://medium.com/geekculture/eda-for-image-classification-dcada9f2567a (accessed Oct. 28, 2023).

[4] A. S. Gillis, E. Burns, and K. Brush, “What is deep learning and how does it work?: Definition from TechTarget,” Enterprise AI, https://www.techtarget.com/searchenterpriseai/definition/deep-learning-deep-neural-network (accessed Oct. 23, 2023).

[5] “What is exploratory data analysis?,” IBM, https://www.ibm.com/topics/exploratory-data-analysis (accessed Oct. 23, 2023).

Appendix

Logbook

Week 10

This week I did the tutorial challenge problems for this week, but didn’t do a heap of work on the capstone project, I mostly just familiarised myself with the dataset.

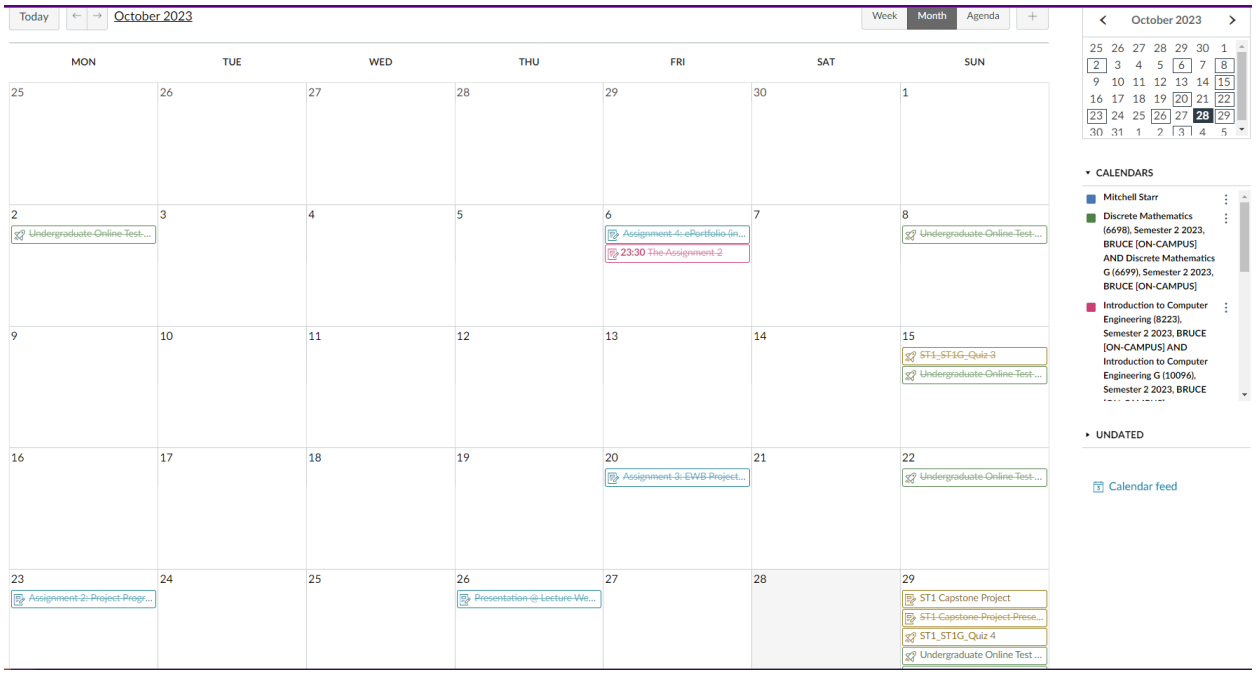
Week 11

This week I did the final tutorial challenge problem which took me a little while, so for this reason I still didn't do a heap of work on the capstone project. I also had many assignments for my other classes that I had to work on during this week so that made it hard too.

Week 12

This week I was unable to work on the assignment much because I was overwhelmed by assignments. I spent a heap of time on other ones, especially PPIE, but was able to do a fair bit

on the weekend. This was when I was finally able to start work on the project properly. I created all the EDA code on the 21st and 22nd. I also made a start on the presentation, finishing most of the slides off and about 75% of the speech.



Week 13

In week 13 I did the rest of the work training the model on Monday, and then created the GUI program on Tuesday and Wednesday. I presented the speech this week and then finally I also did the majority of the work on the report this week, finishing off the assignment.