

# **Search By Image In a Camera Trap Dataset**

**Akin Kaki**

MSc Advanced Computer Science

**School of Computer Science and Informatics  
Cardiff University**

**Supervisor: Dr Charith Perera**

July 2022

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Motivation</b>	<b>6</b>
<b>3</b>	<b>Methodology</b>	<b>7</b>
3.1	Identification of Images belonging to a Location . . . . .	12
3.2	Search by Image . . . . .	13
3.3	Identification of animals . . . . .	14
3.4	Using Normalized images for CNN . . . . .	15
3.5	Web application . . . . .	16
<b>4</b>	<b>Dataset</b>	<b>17</b>
<b>5</b>	<b>Results and discussion</b>	<b>17</b>
5.1	Identification of images belonging to same location: . . . . .	18
5.2	Search By Image . . . . .	19
5.3	Identify images with animals . . . . .	20
5.4	Using normalized data for camera trap . . . . .	22
<b>6</b>	<b>Limitations</b>	<b>23</b>
<b>7</b>	<b>Related work</b>	<b>24</b>
<b>8</b>	<b>Conclusion and Future work</b>	<b>26</b>
<b>9</b>	<b>Reflection</b>	<b>28</b>
9.1	Adapting different concepts . . . . .	28
9.2	Time management . . . . .	28
9.3	Learning from mistakes . . . . .	28
9.4	Overcoming Challenges . . . . .	29
<b>10</b>	<b>Lessons learned</b>	<b>29</b>
<b>11</b>	<b>References</b>	<b>29</b>
<b>12</b>	<b>Figures</b>	<b>31</b>

## Abstract

Camera trap images were widely used in the study of wildlife, but most of the time, the location information of the images (Camera traps) was kept unknown. This can be due to using low-cost cameras that don't have a GPS sensor or to keep the location of rare species hidden. This information is lost once the researcher finishes their work. However, this causes an issue when new research is done using the same data or when a researcher uses an older dataset due to a lack of access to real-time data directly from the forest. To solve this problem, we propose a technique that uses a few images from a particular location to generate a background image and compare it with the rest of the data to find images from the exact location and develop a proof of concept to implement it. Some manual work is required initially. While it can also be solved using machine learning algorithms, these models generally create tags for images that might give a false positive when searching for a location, as the same animal can be present in two different locations. The models can tag the animals instead of the location. Initial data required to train the model is enormous and need to be manually identified, defeating the purpose of automation. we tested this method with some custom data and got an accuracy of 88.48% while using the Machine learning model only got an accuracy of 44.68%. Once the whole dataset is processed using this approach, a single image can be used to find to which location it belongs (search by image) and achieved an overall accuracy of 95.6%.

Camera-trap datasets are not open sourced sometimes, in that case manual processing of the whole dataset (which might contain 100000 to millions of images) or once the researcher identifies images from a location, manual process is required to determine the presence of animals in the images. Machine learning models cannot be used sometimes as they might only have low power devices and/or no internet access in a remote location, or the dataset is a closed source. So, we used the same process of comparing the background image to identify animal presence in the images. We tested on a subset of custom dataset we used in the above process and achieved an accuracy of 71.48%.

Normalizing the dataset is a common process used in machine learning to convert the dataset into a normal distribution. This process is done to eliminate the data representing extreme conditions. However, normalizing the camera trap data, where the animal doesn't occupy much area of the image, might result in loss of information about the animal and may result in reduced accuracy when training the model to identify the images with animals. To prove this, we trained a simple CNN using normalized data and non-normalized data and compared the results. we were able to achieve an accuracy of 49.79% when using the normalized dataset as compared to using the non-normalized dataset, which achieved an accuracy of 52.92% proving the statement

**GITHUB LINK:** <https://github.com/Akin1998/Dissertation.git>

**Keywords-** Background image, Machine learning, Camera Trap dataset, Time lapse dataset, Normalization, CNN(convolutional neural network), Static camera

# 1 Introduction

Camera traps have become increasingly popular in the study of wildlife. Cameras will be placed at a certain location in the forest. These cameras are either motion-activated using sensors (camera trap dataset) or take images at regular intervals (time lapsed). Because of this, a lot of redundant, empty images (images without animals) are produced as the sensors can also be activated by external factors like wind, rain, etc. Most of the time, these cameras don't have a GPS sensor to keep the location of the animal's a secret to avoid poaching. However, this causes an issue when using those images for research as the location information is lost, and the research carried out by (Di Bitetti MS et al. [7]) suggests that location information is quite important when studying about animals. Only the researchers that are currently studying have this information, and it is lost once they finish their study. Biologist carrying out research at a later stage or who do not have direct access to the forest needs to conduct their research using the older datasets that were previously collected from the forest and can only use photographs of animals where two or more different species of animals are located in the same image when studying about wildlife and how different species interact with each other as the location information of the image is lost, this prevents them from researching how different animals interact with each other in a particular location (research on habitat). This issue is further amplified when studying about rare species and how they interact with other animals. To address this issue, we have developed a technique that can identify the images belonging to the same location or allow users to search for images in past datasets belonging to the same location using a small set of data (a few images) belonging to that location that researchers have collected. Researchers can do this by checking for an animal if the dataset is labelled and manually going through a few images to identify it (e.g., they can monkey in the dataset and manually go through a few images to collect images from a particular location of the forest), if the dataset is not labelled, they need to find some images based on how they images are stored in the folder (i.e., images might be organized in folders based on a date, etc.).

As mentioned by (Zivkovic, Z. et al.[6]), an applicable assumption of static camera surveillance is that a statistical model can well describe the pixels of the background image. camera traps can also be considered static camera surveillance as static cameras are used in monitoring the animals, and this principle can be used to generate a common background image. The redundant images, while not useful for the researchers since they are taken from a static camera, can be used and compared with each other by pixels in the same location background image will be created, which will then be used to find similar images by setting a threshold to account for small changes caused by wind, rain, etc. or identify images that are taken in the same location as the background in all those cases is same. While similar approaches have been used in a time-lapse setting to identify the presence of an animal in a timelapse images/video (where cameras take images with a set delay). This proves to be very challenging in a camera trap setting where the cameras can be activated by many random events like wind and rain, and these events differ a lot from place to place. So, machine learning trained to identify a location cannot be used in a different location where conditions are different it also requires a lot of data for each location to train the model, which has to be manually collected. It may also associate an animal with a location and return images of the



particular animal instead of the images from the required location. instead, this technique uses the redundant data generated by the activation of the cameras by those random events to create an empty background image and uses this background image to search for images taken from the same location using pixel-to-pixel comparison from a dataset that was not processed by this process or where the location information is not labelled. Initially, some manual process is required to identify a few images from a particular location, as mentioned above, and nighttime images must be separate from daytime images since most of the pixels in a nighttime image are just black and so will match with any other nighttime image even if it's not from the same location. The daytime images will then be used to identify all the other images in the dataset that belongs to the same location. While this approach cannot identify the name or the GPS coordinates of the images, this process can be used to group all the images that can be from the same locations, i.e., categorize the forest into sub-regions from the older dataset by either using newly collected images from the same location or manually identifying images from the same location in the older dataset. we have also tested the performance of this approach by comparing the results obtained from a CNN trained on the same dataset.

A similar method used to identify the images from the same location can also be used if an animal is present in the image. Biologists spend a lot of time classifying the images with and without animals. Most of these images do not contain any animals and are just redundant as multiple factors like wind, rain, etc., can activate the camera traps, which make the camera captures a redundant image camera set up in a time-lapse setting need to take more images at regular intervals with the smaller time gap between them to make sure the required animal is present in at least few frames. Both image capture methods result in a lot of redundant data. About 70% of all data generated by the camera traps are redundant. Although there are a lot of machine learning algorithms developed to automate this task, researchers (biologists) usually work in remote locations (closer to forests) when conducting their research and have little to no internet access or might want to keep the data private for the safety of animals and/or might have low power devices which cannot be used to run a computer vision. Since this method is based on generating a background image and then using it, this can be run on any machine as it does not require much processing power. Since we generated an empty background image in the above process of searching for images in the same location, we can use this background image and compare it with the rest of the images and considers the image to contain an animal if it falls within a specific set threshold limit accounting for foreground objects. This limit depends on the animal we are searching for, and the area of the image the animal occupies (i.e., bigger animals occupy more area of the image, and so the lower limit of the threshold needs to be low as compared to the smaller animal which occupies less area). We are also required to set a threshold for pixels to account for minor changes between the background image and the other images in the dataset. While multiple factors can affect the performance (factors like different brightness due to weather conditions etc.). The performance of this approach is, although it is not as good as the state-of-the-art machine learning models designed to solve the problem for camera trap images, it can still give a good result based upon the noises in the image itself. This method needs users (biologists) to use images belonging to the same location, which the first task can obtain. Even in worst-case situations, it can help eliminate some of the

redundant data using less processing power, thereby reducing the work of classifying the images for the researcher.

Computer vision models require a lot of processing power and use a lot of images for training to get better accuracy. Issues can arise when the training data set contain images with extreme conditions. And manually looking at these extreme conditions in a training dataset is not a good option, and in some cases, due to the enormous size of the datasets might not even be possible. These extreme conditions are usually eliminated from the images by normalizing all the images in the dataset after pre-processing the data by using mean and standard deviation. Normalization, in general, converts all the pixels of the same location into a normal distribution and can eliminate very high-intensity noise and low-intensity noise in the pixels. Pixels having a normal distribution which makes it easier for the machine learning algorithm (visualizer or CNN) to process the image, but this might cause some issues when training a model to identify the animal in a camera trap dataset as the animal occupy a fraction of the area of the image. Most of the dataset consists of empty images. Normalizing this data can remove the animal from the image. Since I have already created a mean and standard deviation image while creating a background image, I use those images to normalize the whole dataset and use this normalized data to check how it affects the performance of the machine learning model when training on a camera trap dataset to identify the presence of an animal in the image. While this test is not directly valuable for this project, based on the results, this approach of normalizing the data in a camera trap setting can be used while making improvements to the identification of animal's presence of this technique in the future while also improving the already existing methods developed to solve this task.

## 2 Motivation

Cameras are set inside forest; these cameras are usually sensor activated or time lapsed (usually video files). Biologist (researchers) use the images generated by these cameras to study about wildlife and their habitat. However, not all researchers can access live data directly from forest. In such cases, they use already available datasets to conduct their study. Most of the time, the GPS information is not collected from the forest as this can put the animals in danger (giving direct location to poachers etc.) this becomes a problem for the researchers who are using the available dataset as they need some information regarding the place to study about how the wildlife interact with other animals and their habitat. Time lapsed datasets overcome this problem since all the images are taken from the same location so although the GPS information is not available researchers can still understand the habitat of wildlife since all the interactions of the animals will be present in the time lapsed video, but time lapse usually can only get animals in few frames which might not provide enough information for the study. Camera traps activated by sensors on the other hand can capture a lot more images of the animals but depending the how they are sorted (sorted based on animals if already labeled, sorted based on date etc.), images belonging to same location will be jumbled across different folders. As a result, researchers cannot access all the required information (images). Although machine learning models can be trained

to identify images from the same location, to train the model, we need lots of data from one location that is unavailable (since GPS information is lost). So, we require manual work to identify those images which takes much time (since datasets are usually very huge containing 100000's images to millions) and is not a good option. Machine learning models trained on one location (forest) cannot be used on another location and so will not be helpful.

The focus of this project is to develop a technique that searches the whole dataset to find all the images belonging to the same location (it cannot identify the GPS coordinates) and use proof of concept to test it. The idea is, since researchers will have some images regarding the target animal for their research, this technique uses those images to create an empty background image of that location (this is possible due to the cameras being static) and then use the generated background image to search the whole dataset to find images with the similar background giving researchers the data they need about the location. Once they have the data from a particular location, this technique will also help in identifying the images with animals if the dataset isn't labelled. On the other hand, once this procedure processes the whole dataset and a background image is created, researchers can then use a single image to search for the images from the required location for their research.

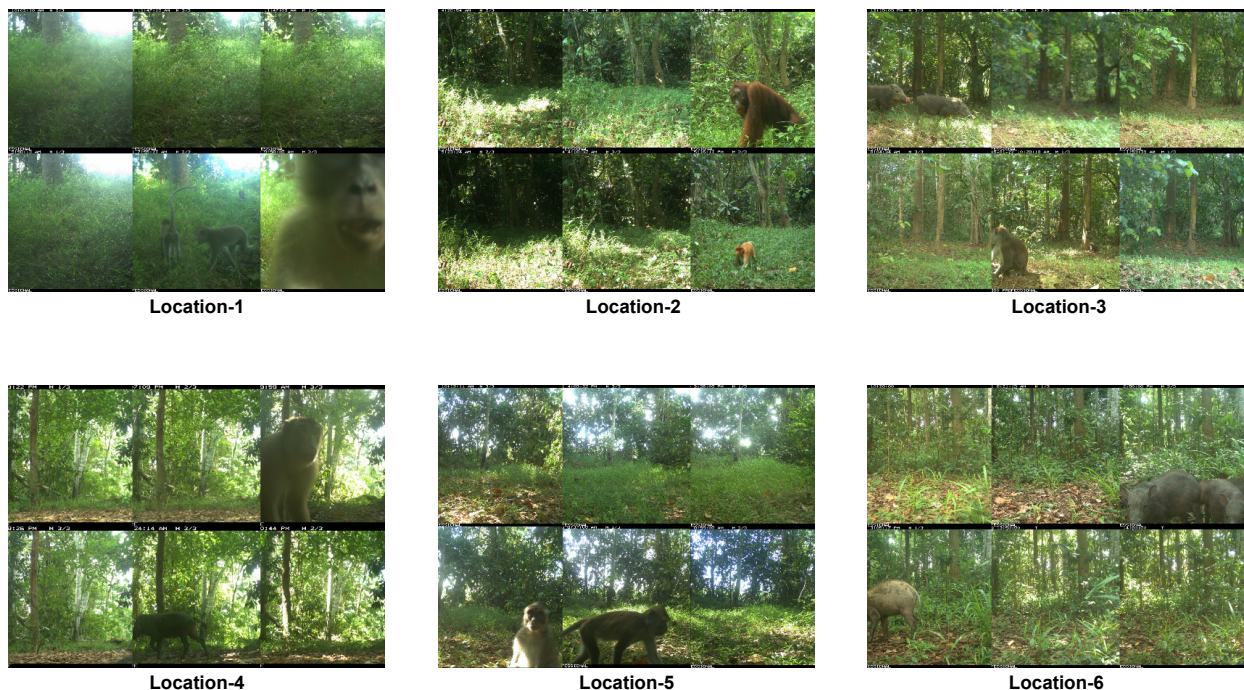


Figure 1: Different locations present in the dataset

### 3 Methodology

Cameras are set in the forest, which can be either activated by a sensor (camera trap dataset) or takes images on a regular interval (timelapse dataset). Researchers use the images from

these cameras and conduct their research on wildlife. Usually, these approaches produce lots of redundant data due to being activated by random events (wind, rain, etc.). About 70% of the data is redundant and cannot be used by the biologists (users) to conduct their research. The idea of this project is that these redundant images can be used to identify similar images and can also be used to search for images taken in the same location. The images here are identical to a static camera surveillance and so were used to create a clear empty neutral background image as the background in a static camera surveillance can be well described by a statistical model (**Zivkovic, Z. et al.[6]**), while accounting for all the minor changes which can be caused due to change in lighting, little wind, etc. present in each of the images and will be used to compare with other images with a set threshold. The threshold values will be based on how much of the background needs to match to consider two images are from the same location and slight changes in each pixel of every image (to adjust for brightness) when compared to the neutral background image. Initially, some manual process is required to identify some images from the same location (same camera). Since cameras are usually fixed and are not moved, images from a particular camera are considered as an image from one specific location and can be compared by comparing the pixels in the same location.

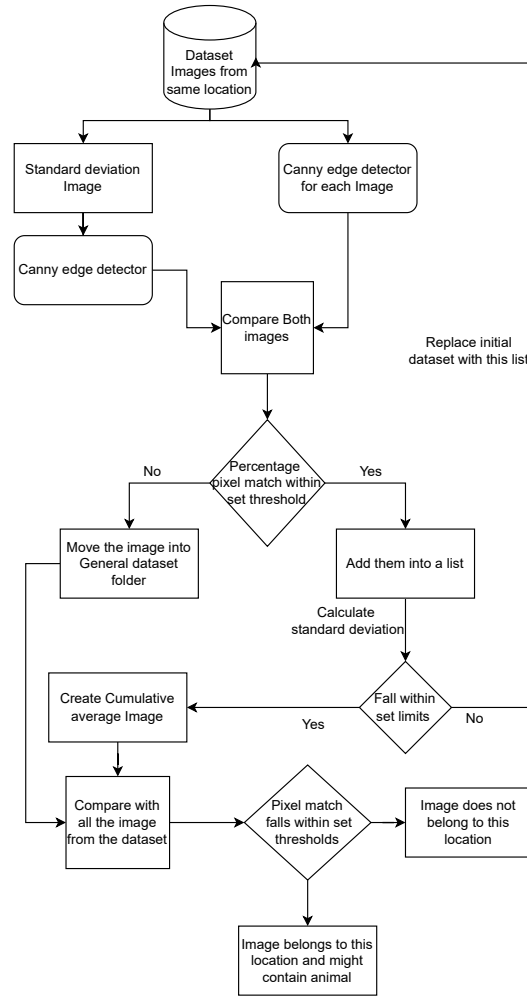


Figure 2: Flowchart Representation of Methodology

Once we have few images from a particular camera, the images first need to be filtered into day and night since this proof of concept is designed to only identify the day images. As there is not enough information from a nighttime image since most of the image is dark or might just be in a grey scale (if the camera contains night vision technology), which makes comparison of the images incredibly difficult as the dark pixels can easily match when comparing pixel to pixel with a set limit to account for minor changes resulting in a false positive. Methods of obtaining the images are explained in Dataset. After separating the images, it then reads all the images and converts their dimensions to (512 x 512 x 3). This is done to reduce the total number of comparisons, reducing the amount of memory required. Using higher dimension images can give much better results when enough memory is available or can wait for a longer time to get better results. The standard deviation of all the images, as shown Figure 3, is then calculated as

$$SD = \sqrt{((\sum (I(i, j)k - U(i, j)^2)/N)}$$

Where:

- SD = standard deviation
- $I(i, j)k$  = Pixel of position (i, j) of image k
- $U(i, j)$  = Pixel of position (i, j) of Cumulative average image
- N = Total number of images
- k value is of the range 1,N

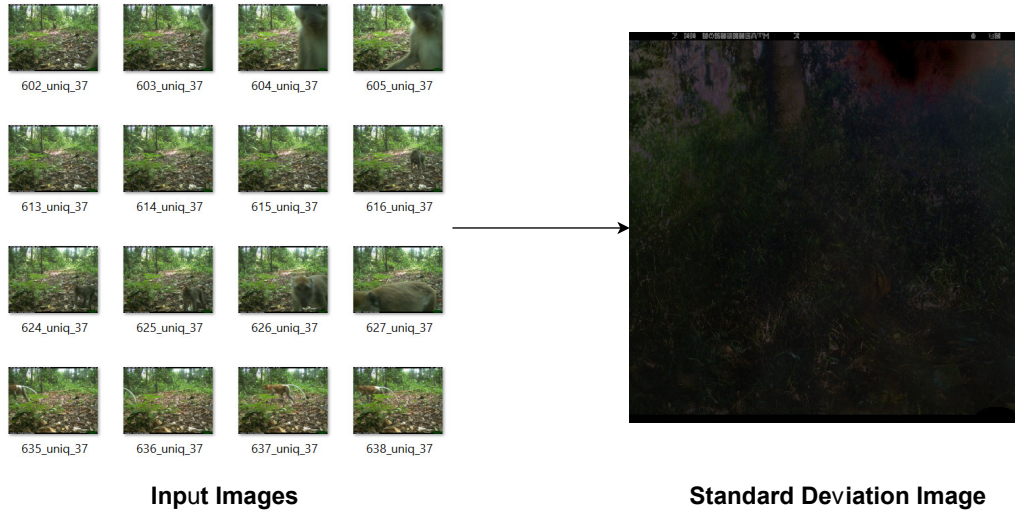


Figure 3: Standard Deviation Image

The standard deviation will give us information on how the pixels are dispersed in comparison to the mean as mentioned in (reference-[8]) i.e., we can use it to find pixels that



deviate the most compared to other pixels in the same location since most of the images are empty, the mean image can represent an empty background image if at least the significant noise images are removed (images with animals, different lighting conditions, etc.). The mean is calculated using all the images, including the ones where extreme conditions are noticeable in this step. We generally need a lower standard deviation so that the pixels of the images used to create an empty background image are less dispersed. The images responsible for higher standard deviation are usually those with animals or images with extreme weather conditions (sun shining directly on the camera lens, water present on the lens, etc.). In the following steps, we eliminate these images by using the canny edge detector as results from (Joshi, M.[9]), shows that the performance of the canny edge detector is better than other edge detection algorithms.

Canny edge detector is applied to the standard deviation image. The canny edge detector is a multi-stage algorithm that can be used to detect a wide range of edges. The image generated by the canny edge detector is shown in Figure 4. This generates a binary image with 0 responding to black pixels and 1 responding to white pixels. This is compared to a white image of dimensions (512 x 512) to obtain the percentage of white pixels in the image since only white pixels in both images match, allowing us to count the number of white pixels. This is set as an upper limit. The lower limit needs to be manually adjusted. Higher the value of the lower limit, the higher the chance for that image to contain major deviation from the other images. This can be due to animals blocking the camera, bad weather conditions (fog, heavy rain), etc.

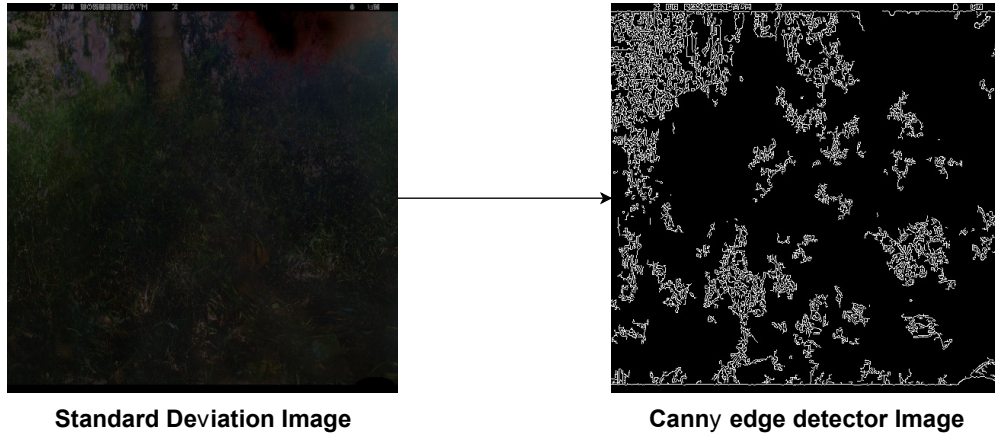


Figure 4: Canny edge Image

The canny edge detector is applied to every image. All the black pixels of images were converted into green, as shown in Figure 5, except for the standard deviation image, to avoid matching black pixels between the standard deviation image and the current image. However, because of this conversion to black pixels to green, the binary canny edge image becomes an "RGB" image. So, the standard deviation image is also converted into the "RGB" shape (512x512x3) by converting the binary value of black pixels from 0 to (0,0,0) and white pixel value from 1 to (255,255,255). Once the images are in the same shape, i.e.,

the same RGB value for a white pixel, the image is compared with standard deviation by subtracting the pixels at the same position creating a NumPy array. Since all the black pixels were converted into green pixels for the image, it returns 0 only if the two pixels are white. The image is eliminated if the percentage number of zeros in the NumPy array falls into the set threshold. The threshold's lower limit can be manually adjusted or calculated directly by comparing the percentage match of white pixels when comparing the mean and standard deviation images, while the higher limit is set by comparing the Standard deviation image with Black image (counting number of white pixels).

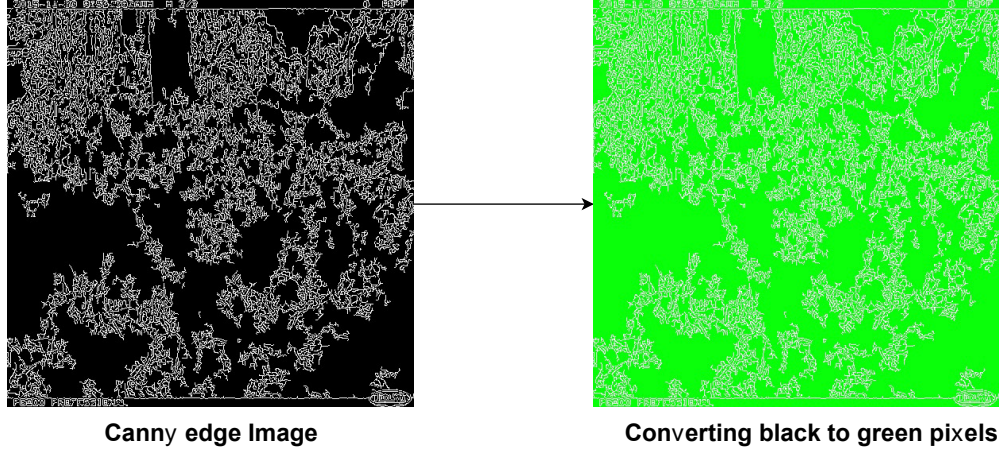


Figure 5: Conversion of black pixels to green

The eliminated images are usually the ones with more noise which can be caused due to animals, weather factors, etc. Even after removing the images that contain more noise/higher standard deviation, the images that remain will still have some minor noise elements which can be caused due to wind, lighting, etc. in order to remove this cumulative average of pixels in the same location is calculated which will reduce the effect of any minor noise in the pixel and this final image is considered as a background image. The above process of calculating standard deviation and removing images responsible for higher standard deviation can be repeated until a desired standard deviation is achieved. The cumulative average image can then be calculated to create the desired background image. The cumulative average is calculated as following:

$$U = (\sum (X(i, j)k)/N$$

Where:

- U = Cumulative average Image
- X(i, j)k = pixel of position (i, j) of image k
- N = Total number of images
- k value is of the range 1,N

Mixture of Gaussian approach was used by (Weinstein, B. (2015) [3]). to calculate the background image. However, Mixture of Gaussian approach requires a sequence of images that cannot be obtained when using sensor-activated cameras, as images are generated at random times by random triggers. If the dataset is from a time-lapse setting, using Mixture of Gaussian for background image generation will give better results as the approach can be used to update the background image after a pre-defined number of frames, and this is only possible when the dataset is time lapsed or when sequence is available in a camera trap dataset as a result of multiple activation by various factors.

### 3.1 Identification of Images belonging to a Location

The above procedure (section 3) was used to identify the images belonging to the same location. This can be done once we have some images from a particular location. This data can be new and recently collected (if having access to real-time data and removing all the night-time images) or manually searched from an existing dataset as mentioned in the Dataset. Once the required data is obtained, a background image will be created using the procedure described above. After creating the background image, it is used to be compared with all the other images present in the other dataset in which we are looking. If the percentage of pixels matching the image from the dataset falls into the set threshold, the image is considered to be from the same location. The threshold value (pixel match threshold) is set based on how much of the image can be considered foreground and how much background is visible. All the animals are considered foreground, and the percentage of foreground changes depending on how many areas the animal can occupy (e.g., if there is an elephant in the image, it might occupy about 70% of the whole image, so the lower limit of threshold is set to a value closer to 30%. Likewise, if there is a small monkey might only occupy 20% of the image and so the threshold is set to 80%). Sometimes even smaller animals can occupy a larger area of the image if they are close to the camera.

The images are compared as following:

$$NP = 0, \text{ if } : X(i, j)k - U(i, j) < T$$

$$NP = 1, \text{ if } : X(i, j)k - U(i, j) > T$$

$$Percentage = (number\ of\ zero's\ in\ NP / size\ of\ NP) * 100$$

Where:

- NP = NumPy array
- $X(i, j)k$  = pixel of position (i, j) of image k
- $U(i, j)$  = pixel of cumulative average image at location (i, j)
- N = Total number of images
- k value is of the range 1,N
- T = pixel comparison threshold value to account for minor changes



- Percent = percentage of similarity between the two images

A threshold “T” called pixel comparison threshold, as shown in subsection 3.1 was set to account for minor changes between the calculated background image and the image used for comparison. Minor changes can be caused due to different lighting conditions throughout the day, wind, a slight drizzle, etc. a value of about 15% to 20% should also be considered in the pixel match threshold as there might be images that might not contribute much for the standard deviation, however, is still too big to be considered as a minor change when compared with background image pixel present in the same location and as such cannot be accounted when setting the pixel threshold “T” as shown in Figure 7, where the image is bright that some green pixels look almost white. However, there are very few to make a huge difference in standard deviation while too big change in values when compared to pixel as green pixel values are usually (0,255,0). In contrast, white pixel values are usually (255,255,255).

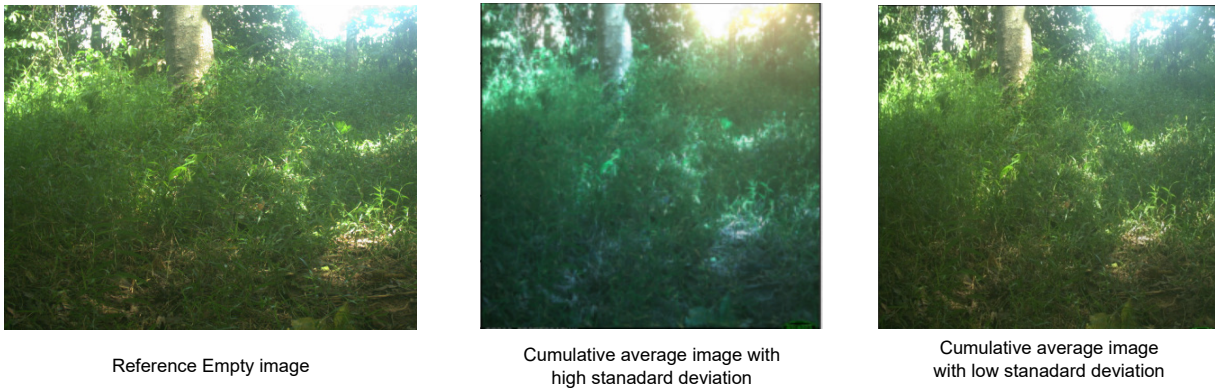


Figure 6: Cumulative average

### 3.2 Search by Image

Alternatively, once the above process is done and a whole dataset was classified, the user can provide a single image that is not a nighttime image. This image will then be compared to all the available background images created from the process mentioned above. The user can also provide upper and lower thresholds based on the size of the animal (if present in the image) or how much of the background should match to consider its image from the same location (since there can be different factors that can affect the comparison as mentioned above) or they can choose to use the default threshold which is set to (30% to 70%). Once we find a background image within the set threshold, all the images that belong to the set of the background image will be returned. If no background images fall within the set threshold values, we can say that the image used for searching belongs to none of the locations in the available dataset. This is one advantage as compared to a machine learning algorithm where we need to train the model to so it knows what kind of images belongs to a class and what images belong to none of the classes, which can be challenging in case of camera trap images as most of them are taken in the forest and might not look different from one another unless compared pixel by pixel in the same location and so has a high chance of the

wrong classification when it comes to identifying image that does not belong to any of the location used in the training data. Since “belong to none” needs to be treated as a separate class in the case of machine learning, additional data that is not present in the data needs to be collected to represent the class “belong to none” as all the images in the dataset will belong some class since being from the same forest and so belongs to some location.

### 3.3 Identification of animals

Using the cumulative average image, we can identify if an animal is present in the image. However, for this to be done, all the images given to this proof of concept should belong to the same location, and nighttime images should be removed. Images belonging to the same location can be obtained as mentioned in subsection 3.1, to the location. The background image was obtained using the procedure mentioned in the section 3. Once after calculating the background image (cumulative average image), a threshold was set based on the assumed size of the animal, i.e., the bigger animal will occupy more area of the image, and so we need to set the lower limit to  $(100\%-x)$  assuming that animal might take up to  $x\%$  of the image, the upper limit needs to be in the range of 70%-75% this is done because there can be about 15% to 20% variation from the background image to the image we compare and at least 10% for the animal itself (so the variation limit becomes 25% to 30%). While the background image averages out all the minor issues like wind, rain, etc., all the images will not have these minor issues. As mentioned at the end of subsection 3.1, even though we set a pixel threshold limit in order to account for minor changes in the pixels of the background image and current image used for comparison, we still might have conditions where the changes are too small to be removed using standard deviation and too big to consider in pixel variation because of this, it becomes extremely difficult when using this method to differentiate between the animal occupying less area (less than 15% to 20% in general) of the image (small animal or located far from camera) and images where variations are lower than standard deviation but higher to be eliminated using pixel comparison threshold. Even if the animal occupies most of the image (maybe more than 70%), it can also make the camera lose focus on the background and/or change brightness, contrast which makes the background much different from what it’s supposed to look like and might not consider it as required image.

Once we set the upper and lower threshold, the background image will then be compared with the rest of the image as mentioned in "NP."(subsection 3.1) If the image falls into the set pixel comparison threshold, that image is returned as an image with the animal. While this method of identifying an animal in an image is not effective as a state-of-the-art machine learning model for this task, it can still be used to identify if the animal is present or not and will give a better result if the dataset consists of fewer disturbances as it might also classify bright images (or images with major noise) as shown in Figure 7 as an image with the animal because some of the green pixels (leaves color) are almost white which might result in the image falling into the set pixel match threshold.



Figure 7: Bright Image

### 3.4 Using Normalized images for CNN

Since already calculating a cumulative average and standard deviation of images, the other idea of this project is if using normalized images can decrease the accuracy of a computer vision algorithm (machine learning) compared to using the images directly when using a camera trap dataset. Images are usually divided by 255 as a preprocessing step in a computer vision algorithm. This is done as pixel values lie between 0-255, and this division will change the values to 0-1. However, other steps must be taken to remove extreme conditions from the dataset. Images are normalized as shown below in “**Ni**”, using mean and standard deviation. Since normalization can be used to remove pixels with very high and low intensities and convert all the pixels to have a normal distribution, this step can eliminate the extreme conditions present in the

$$Ni = (X(i, j)k - U(i, j)) / SD(i, j)$$

Where:

- $Ni$  = normalized image
- $X(i, j)k$  = pixel of position (i, j) of image k
- $U(i, j)$  = pixel of cumulative average image at location (i, j)
- $N$  = Total number of images
- $SD(i, j)$  = pixel of position (i, j) of standard deviation image
- k value is of the range 1, N

All the images of the dataset are normalized using the above formula to convert all the pixels into a normal distribution. Other pre-processing steps (cropping, applying filters, etc.) can also be applied to these normalized images, like the images where this step wasn't done. However, in this test, no other pre-processing steps have been done apart from converting all the images into shape (512,512,3) before being normalized. And a simple CNN, as shown in Figure 22, was created, and the dataset was split into 70% training and 30% testing set. And the CNN is trained separately with both datasets that were not normalized and the Normalized dataset and tested. No development set was used as this was just to test the performance of the CNN when using a normalized dataset.

### 3.5 Web application

A front end for this application was also created using a tool called **anvil.works**. This can be used to develop application directly using Jupyter notebook as the backend while also using python to create a simple user interface (HTML and CSS is required when creating a complicated interface). Using this, I developed a simple prototype web application that allows users to upload images as input. The user first needs to enter the class names separated by a comma (",") without any space, as shown in Figure 19. and click on submit. The application then creates the folders for those classes in the backend and asks the user to upload Images of each class separately, as shown in Figure 20. And the images will be uploaded into the respective folders, and the application will then automatically do all the above-mentioned procedures in order to generate a background image, and this will be saved as a NumPy file (.npy extension) to avoid any loss of data while saving the background image. NumPy file will save the image in NumPy array format with all the information required to reconstruct the image even on a different platform. Future versions of the applications will allow the users to set the threshold values that are suitable for them. The current version has a pre-set threshold which might not be ideal for different kinds of datasets.

The users will then be able to upload a different image to check to which class it belongs (Search by image), as shown in Figure 21. the application will also return "None" if the image does not belong to any uploaded classes. The users can also check if animals are present in the uploaded classes, as shown in Figure 21. As mentioned above, multiple factors can affect when trying to retrieve the images where an animal is present, but it can still help the users (biologists) to some extent. This can also be updated so that the users can upload some recently collected images and use them to reverse search in an older dataset that was not labeled.

This web app is designed based on the procedure mentioned above of creating a background image as its backend and so cannot be used for different datasets (non-camera trap datasets). To generate the background image, some manual work is required initially to ensure all the images in a certain class belongs to the same location of the camera trap dataset. No night images are present in the dataset. This web app can be easily hosted on the user's machine to run locally, allowing users to process closed source data without uploading to the internet or use this web application when they do not have good internet access. Since this does not require more processing power, this can also be used on a low power device

like raspberry pi.

## 4 Dataset

I have created a custom dataset which is a subset of the DGFC dataset. About 7200 images belonging to 6 locations (calling them location-1, location-2, location-3, location-4, location-5, and location-6) were manually identified (1200 images per location). Fifty images from each location were selected at random and set aside. These will be used to test this project's search by image aspect and so was not be used when generating the background image . Another 8000 random images were also chosen to represent the additional data found in the Camera trap datasets, from a dataset consisting of about 3.4 million images. Few images pre-location was selected to be used as an initial dataset. And the rest of the images were kept in a single folder. This is an unlabelled dataset, so Images from the same location are identified manually.

In this case, I had access to different folders where each researcher had stored their data to conduct their research, and some manual search was required to identify these images. 70% of all the images chosen are empty similar to normal camera trap datasets that consist of about 70% of empty images, i.e., only 2160 images contain animals out of 7200 images, 360 images per location contain animals, and 8000 images are chosen randomly containing 2400 images with animals and 5600 empty images, this will be used to simulate additional data present in the dataset and when training the CNN. No nighttime image was selected here as this method is developed only to identify images during the day.

In the case of a labelled dataset, images can be manually identified by searching for the particular animal the research focuses on, and manually looking at those images to identify images from the same location as looking for a single animal, at least some images where the animal is present will be from the same location and only a few images are required initially (looking for images with less noise, although it might take a while, can give better results) and once identifying some images from the same location (at least 50 is recommended based upon the results), then this method automatically searches for the images in the same location.

## 5 Results and discussion

Initially, 50 images belonging to each location were chosen randomly (excluding the 50 images set aside for search by image), 300 images in total. The initial value for standard deviation to eliminate images was directly calculated using the mean of images as the mean will have the lowest number of pixels matching with standard deviation.

## 5.1 Identification of images belonging to same location:

Background cumulative average image is generated by the process as shown in Figure 2, using the above threshold values and 50 images. The pixel comparison threshold value 'T' in [NP] was set to 25.5. i.e., there can be about 10% variation in a pixel since the pixel value ranges from 0-255. This will be used when comparing pixel to pixel. And a limit of 30% was set to identify the image, so if 30% of the pixels from the background image match with the pixels of the image at the same location we are comparing, then that image is considered an image from the same location. 30% is selected, assuming that an animal might occupy a maximum of 70% area of the image due to the animal being closer to the camera or a large animal.

With the above set thresholds and using this technique, we were able to correctly identify 6371 images out of 7200 images belonging to any one of the six classes reaching an accuracy of **88.48%** and did not classify any of the 8000 randomly chosen images as belonging to any one of the six locations. So, in total 14,371 images were correctly classified reaching an accuracy of 94.54% as shows in Figure 8. No images were miss classified as belonging to a different location, i.e., 0 images were classified as False positive. This result increased to correctly identifying 6512 images (excluding the 8000 random images) reaching an accuracy of **90.44%** when the limit was set to 25%, i.e., at least 25% of all the pixels match with the generated background image when comparing the images as mentioned in subsection 3.1, and no image was misclassified. When using 25 randomly selected images instead of 50, it correctly identified 5729 images with a pixel match threshold of 30%, reaching 79.5% accuracy. When the threshold was changed to 25%, 5982 images were correctly identified, getting an accuracy of 83.08%, and no image was miss classified in either of the cases.

In all the cases, 50 images belonging to one class were given (or 25), and all results the images returned were manually checked and verified to make sure they belonged to the same location. Looking at the wrongly classified images (images classified as belonging to none), the animals sometimes being very close to the camera and occupying most of the area, or light source shining directly to the camera changing the pixel intensities of the image. As the accuracy can also depend on the initially selected images, the images, in this case, were randomly selected rather than choosing images with overall better conditions and accuracy of all the classes were added together to get a overall accuracy.

All the 7200 images belonging to the six classes were used to test the performance using a machine learning model. The additional 8000 images were not used since they represent the class 'does not belong,' which has to be treated as a separate class when using a CNN. The dataset is split into 70% training and 30% test sets. The model was trained to 100 epochs and set callbacks to stop if validation accuracy did not change for five epochs. The performance of CNN was observed. The model stopped training after 61 epochs and achieved an accuracy of 44.68%, which is significantly lower when compared to the accuracy achieved above. This can be because most of the locations look similar within the forest, and data was not enough to train the model. as a result, CNN could not learn more features about the data. However, it could also be due to this being a simple CNN and is not specifically designed for this task. However, designing a CNN for this task requires us to have a lot of data for a given location,



which is impossible unless the data already has a location value and a CNN trained for one location cannot be used in a different location. The Precision values are [0.4552, 0.3695, 0.4854, 0.4229, 0.4571, 0.4740] and Recall values are [0.325, 0.2833, 0.5555, 0.5333, 0.5028, 0.48055] since we have 6 locations (6 classes) and the results for the 30% test set are shown in Figure 9

Confusion matrix								
Predicted	Location-1	1179 7.73%						1179 100% 0.00%
	Location-2		871 5.71%					871 100% 0.00%
	Location-3			1041 6.83%				1041 100% 0.00%
	Location-4				1200 7.87%			1200 100% 0.00%
	Location-5					1062 6.97%		1062 100% 0.00%
	Location-6						1018 6.68%	1018 100% 0.00%
	None	71 0.14%	529 3.18%	109 1.04%		102 1.13%	102 1.13%	8000 52.48% 0.00%
sum_col		1200 98.23%	1200 97.43%	1200 98.23%	1200 98.23%	1244 95.37%	1200 95.17%	8000 94.27% 5.73%
		Actual						sum_lin

Figure 8: Confusion matrix generated using the proposed method

Confusion matrix								
Predicted	Location-1	117 5.42%	41 1.40%	22 1.35%	28 1.35%	10 0.80%	25 1.35%	257 45.33% 58.47%
	Location-2	21 1.40%	102 4.72%	45 2.90%	20 1.35%	32 2.40%	15 0.60%	276 35.95% 68.54%
	Location-3	32 2.40%	20 1.40%	200 9.26%	20 1.15%	30 1.70%	61 2.92%	412 66.54% 59.28%
	Location-4	22 2.87%	21 2.70%	21 1.35%	192 8.89%	43 1.80%	32 2.40%	454 42.99% 57.71%
	Location-5	20 1.70%	20 2.01%	20 1.35%	21 1.35%	181 8.38%	32 1.40%	396 46.71% 64.28%
	Location-6	20 2.15%	40 2.15%	22 1.35%	24 1.35%	20 1.00%	173 8.01%	365 47.48% 65.68%
sum_col		360 94.50%	360 94.50%	360 95.54%	360 94.54%	360 90.28%	360 91.60%	2160 44.68% 59.32%
		Actual						sum_lin

Figure 9: Confusion matrix generated using CNN

## 5.2 Search By Image

Here we use the 50 images per class that were set aside in the above process (300 images in total). Once I identified all the images in a location (images belonging to each of the 6 locations) as mentioned in the above process, I only selected the images identified in the above process when using 50 images and a pixel match threshold of 25%, i.e., 90.44% of the total data. This is done since, search by image can only be done once the above process (Identification of images belonging to same location) is done for whole dataset as in real-world applications, the number of images might be in the range of 100000's to 1000000's and manually identifying all the images from a location is very time-consuming. However, 50 images can be identified as mentioned in section 4, which might take few minutes to an hour.

Once the images belonging to each location are identified, a new Background image is generated based on all the identified images. This image will be used to compare with any new image. All 50 images were compared with this new background image with a pixel match threshold of 25% as we achieved higher accuracy in the above process and pixel comparison threshold (mentioned in subsection 3.1) value "T" of **[NP]** set to 25.5 (10%). The proof of concept shows the location number (1, 2, 3, 4, 5, 6 for location-1, location-2, location-3, location-4, location-5, and location-6, respectively) and returns none if the image does not belong to any of the locations which were manually verified. This method correctly identi-

fied all 50 images for each location using the above threshold values, giving an accuracy of 100%. When the pixel match threshold value was changed from 25% to 30%, this method was able to correctly identify 48,50,47,48,48,46 Images for location-1, location-2, location-3, location-4, location-5, and location-6, respectively. However, no image was miss classified, so the total accuracy was considered 95.6% (sum of correctly classified images/ total number of images). In both cases (threshold=25% and threshold=30%), none of the images were misclassified as belonging to a different location. It returned belongs to none for the seven images when the pixel match threshold is set to 30%. Another 50 images were randomly selected from the 8000 images that do not belong to any location, and this method correctly returned belongs to none for both the threshold values (25% and 30%).

While using 50 images is not a good test, in this case, this approach will most likely only be used to identify to which group a single image belongs, as it only returns the location name (based on how they are categorized in the previous process) the images returned can then be used for the research. So, using 50 images is still a good enough test in this case. The machine learning model was not used in this case to compare the results, as the above model only achieved an accuracy of 44.68% as this current process is an extension of subsection 5.1

### 5.3 Identify images with animals

Once all the Images are identified using the process subsection 3.1, a background image is created using all the available images. The background image is compared with all the images. The pixel match threshold is set to be 30% to 70%, i.e., for an image to be considered as “animal present in the image,” at least 30% of the pixels and at most 70% of the pixels of the image should match with the newly created background image. These limits are set under the assumption that the animal might occupy 30% to 70% area of the image based on its distance and size. Anything above 70% match is considered empty. The pixel comparison threshold “T” of [NP](subsection 3.1) is set to 25.5 (10%). Only the 7200 images classified into animals present in each of the 6 locations (1200 images per location), along with the respective background image generated were used here since this requires all the images to belong to the same location (8000 random images were not used). Images of each location are passed separately. This approach correctly identified 147, 148, 282, 346, 324, 297 images with animals for location-1, location-2, location-3, location-5, location-5, and location-6, respectively, out of 360 images with animals for each location and achieved an overall accuracy of 71.48%. The results in each case were manually verified, and confusion matrices were generated. Confusion matrices for location-1 to location-6 are shown in Figure 10 to Figure 15.



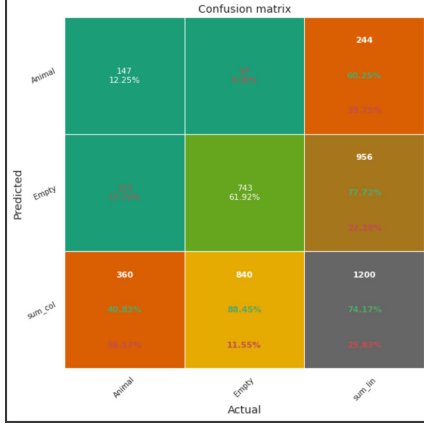


Figure 10: Confusion matrix, Location-1

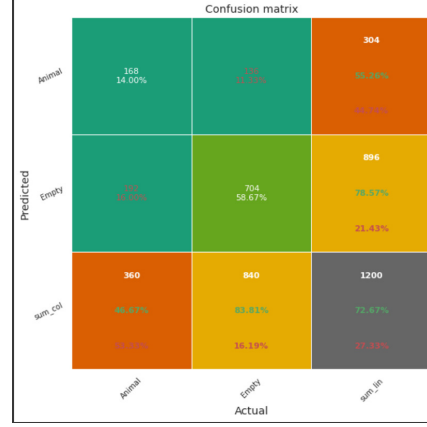


Figure 11: Confusion matrix, Location-2

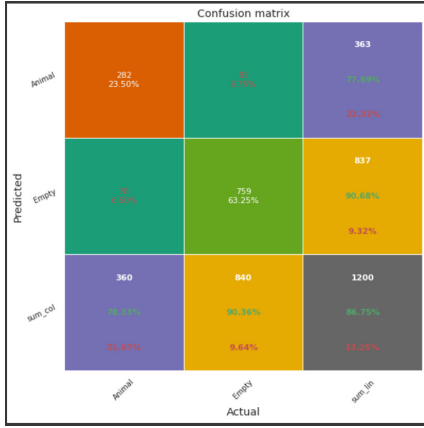


Figure 12: Confusion matrix, Location-3

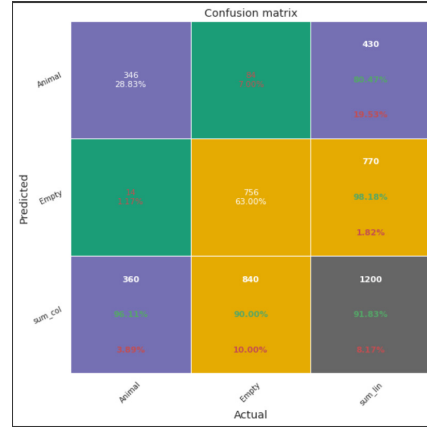


Figure 13: Confusion matrix, Location-4

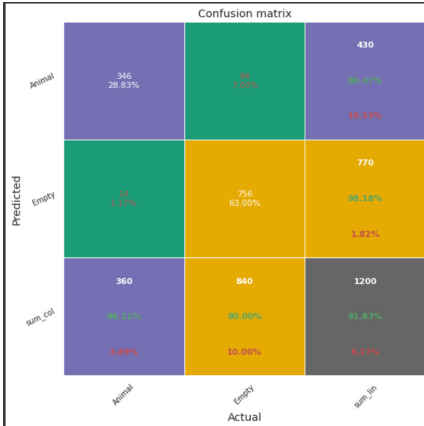


Figure 14: Confusion matrix, Location-5

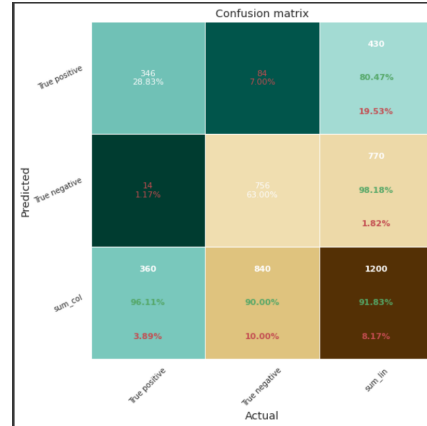


Figure 15: Confusion matrix, Location-6

From the above confusion matrices, we can see that about 11.11% of the images with animals were classified as images without animals. By manually verifying those images, most of the time, the animals present in these images occupy a total area of less than 25% of the image. The threshold we set was that the animal needs to occupy at least 30% of the image.

6.47% of the empty images were classified as images with animals. These images contained either brightly illuminated spots or other disturbances like raindrops on the camera lens and were wrongly classified as expected.

The machine learning model, as shown in Figure 22, was also trained to identify images with animals. However, the results are not compared since a simple CNN was created to test the performance when using normalized data vs. using data without normalizing, as state-of-the-art machine learning models designed for this particular task can identify with an accuracy close to 95%. In contrast, this identification of animals using this approach is designed for specific purposes when the researcher unable to use a machine learning model (due to lack of internet in a remote location and/or not having a powerful device).

## 5.4 Using normalized data for camera trap

A simple CNN was created, as shown in Figure 22 was used to compare the performance of CNN on the task of identifying images with animals when using Normalized data and when the data is not normalized. All the images (both 7200 images and 8000 images) were used. The images are manually separated into images with animals and images without animals. A total of 4560 images contain animals 10640 images are empty (70% of images are empty, and 30% of images contain animals). All the images are initially divided by 255 to convert the pixel values from 0-255 to 0-1, then normalized using mean and standard deviation as mentioned in subsection 3.4, and then this normalized data is used to train then CNN for 100 epochs. The dataset is split into 70% training and 30% test sets. Early stopping is called if the validation accuracy did not change for 5 consecutive epochs. Batch size is set to 200 images. The CNN stopped training after 61st epoch due to early stopping and got a validation accuracy of 49.79%, (Precision [0.2914, 0.6919], Recall [0.4706, 0.5096]) and the results are as shown in the Figure 16

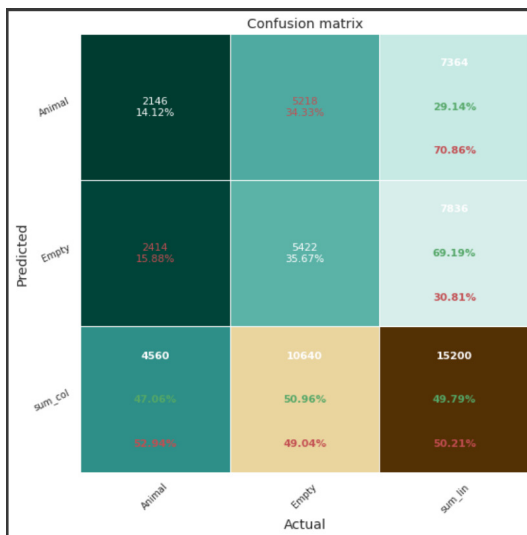


Figure 16: Using normalized data

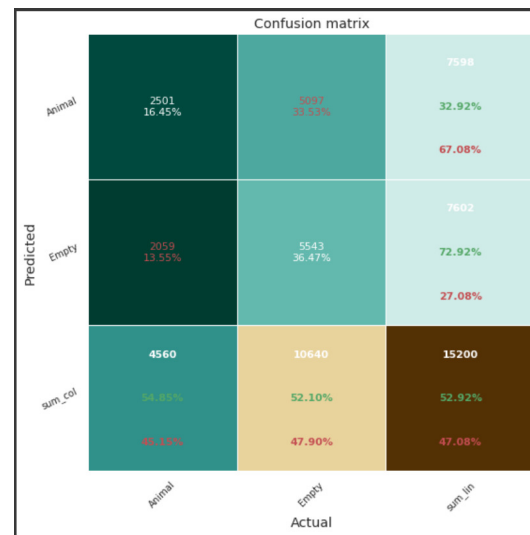


Figure 17: Using non-normalized data

A similar setting is used again, except after dividing the images with 255, the data is not normalized this time. The model stopped training after the 69th epoch due to no change validation accuracy for 5 consecutive epochs and got a validation accuracy of 52.92%, (Precision [0.3292, 0.7291], Recall [0.47041, 0.6197]). The results are shown in Figure 17. Looking at the above results, we can see about a 3.19% increase in accuracy when the data is not normalized. This increase in accuracy can be because information about some animals could be lost when normalized. Since the camera trap dataset consists of approximately 70% of empty images (approximately), when the pixels of the same position in all the images are converted into a normal distribution, the pixels with animals being the minority might lose some information and so results in lower accuracy. The overall accuracy of this CNN is quite low (the state-of-the-art models designed for this task have an accuracy of 95%). This is because this model is designed only to check the difference in performance when using a normalized camera trap dataset. While a 3.13% difference might be considered negligible in some cases, even a 1% increase in accuracy for identifying animals in camera trap images can be considered huge because of the huge datasets (1% of data might contain at least 10000 images).

## 6 Limitations

This method cannot be used to identify nighttime and grey-scale images. This technique requires the “RGB” value of the pixel to create a background. Because of this, data collected during nighttime or data from cameras fitted with night vision technology cannot be used. As a result, this method will not be helpful for research on Nocturnal wildlife (active nighttime animals).

No lossy compression techniques should be applied to the image. All the input images given should be full-size images (even 4k images should not be downgraded). As this technique is based pixel to pixel comparison at the same location, lossy compression results in loss of pixel values which are essential for this comparison. So measures must be taken when downloading/uploading to prevent loss of information.

This cannot be used for the identification of small animals or animals located far from the camera. In general, this method cannot identify if an animal occupies less area of the image due to various factors like bright spots and shadows, which are indistinguishable for this approach and needs to be accounted for in every image.

Even if two cameras are placed in the same location, as this approach is based on static camera surveillance, this cannot identify the two cameras as the same location, and manual work is needed in such cases to identify it (like categorizing them as location-1a, 1b).

Initial data given should be manually verified to make sure they are from the same location. If not, the background image generated not be a proper representation of the actual background, or it might not create a background at all, which can result in either incorrect classification of data or no classification at all

## 7 Related work

In order to automate the task of recognizing individuals in an elephant population, (**A. Ardovini et al**[5]), developed a technique that can detect elephants based on the nicks present on the borders of the ears. The users need to input a few reference points which approximately detect the elephant’s head position, scale, and orientation. The system then performs edge detection using canny edge detection algorithm. The user is then required to point to the initial and final points of the nicks, and the algorithm calculates the shortest path between them. If this is not the desired output, the user must add or delete points to make it into the desired output. Once this is done, this can be used to query images of elephants containing the same nicks, or this can also be uploaded into the database. This is specifically developed to identify an individual elephant from the group of elephants and cannot be used to identify similar images or the presence/absence of an animal in an image. But a similar method can be used to compare the images and find similarities and differences between them. This information can, in turn, be used to check if two images are from the same location or not and the presence/absence of an animal in an image.

Algorithms to constantly update the parameters of a Gaussian mixture model were developed by (**Zivkovic, Z. et al.**[6]), in order to identify and remove of background on a surveillance system with the assumption background of an image exhibit a regular behavior that can be described by a statistical model when captured from a static camera. A time-period  $T$  is chosen to update the background image, and for each new sample after a time-period  $T$ , the training data is updated, and new intensities are calculated based on some defined equations. A parameter constant  $\alpha$  is introduced to limit the influence of the older Frames on the Frames after time-period  $T$ . a pixel from a new image is considered to be background if its value can be described by the density function. Else can be considered foreground. This approach also updates objects that are present in the foreground for more extended periods of time as background and removes them from the background of newer frames, fully adapting to the observer scene. The kernel method used is much simpler compared to older ones and seems to be a better approach for simple static scenes, but this approach requires a continuous sequence of frames. eMammal is a subscription-based service created by (**Z. He et al.** [4]), to classify camera trap images. This approach analyses the sequence of images using a three-step process. It first analyses the images, detects animal from the image, and separates them from the background then features like body size, appearance, speed, entry angle, etc. are extracted then a pattern layer is used to identify specific patterns for different species, and all this information is used to generate a computer recommendation. These results are then verified by a volunteer who provides necessary changes, and finally, these results are further analyzed by an expert to ensure the correctness of the final decision. The accuracy of the developed application is the same as human-verified accuracy since it has two stages of human verification. This is developed in time-lapse images where the animal is present in multiple frames or camera trap images that can be sequenced and is not ideal to be used for time-lapse images where the animal is usually present in one

or two frames or camera trap images where the sequence pattern of images cannot be created.

Methods to partially eliminate nontarget recordings to reduce the workload by detecting changes in the pixel values frame by frame were developed by (K.R.R. et al.[2]). The first two frames were removed from the recordings because of the instability of light. The videos were down sampled on both x, y spatial dimensions time t by a factor of 5,5,10, respectively, to remove the noise, the moment of vegetation, etc., and the research was mainly targeted at beavers. The pixels of every frame are compared with each other, and the recording was classified as nontarget if the pixel values didn't vary much throughout the video. This method can discard up to 76% of the nontarget recordings and can be used to identify motion in camera trap videos which in turn can be used to identify animals by converting the sequence of images into videos, and this cannot be used for a sensor activated camera trap as there is no sequence of images. Motion meerkat, developed by (Weinstein, B. (2015) [3]) was an Improvement on (K.R.R. et al.[2]). And it was developed with the goal of reducing the video to a pool of reasonable and potentially useable frames. It compares each frame to a background image using either accumulated averaging or a 'mixture of Gaussian.' The accumulated background approach can be used when there are fewer frames in a video, or the background varies highly. Mixture of Gaussian compares each pixel of a frame with the previous pixel in the same location, and when it's not within the specified limits, that pixel is classified as foreground. In this way, the background image will be updated over time to account for the changes in illumination. However, this also faces the same issue as it cannot be used in a setting where there is no sequence of images. In order to overcome these issues, (Jennifer L. et al [1]) developed the method called AnimalFinder: a semi-automatic system for the identification of animals in time-lapse images. This program takes a set of images, separates them based on day and night, converts day images into greyscale, and uses canny edge detector algorithm to identify the lines in the images. It then looks for large areas with low line intensities, as animals usually have smooth surfaces, applies a colour saturation mask, and then analyses it based on size and shape. A median filter of pixel size 40 is first used on the night images, and then canny edge detector is applied. Then from this point, classification for both day and night images becomes the same. A threshold value is used to ignore false positive pixels as objects like rocks and bushes also contain smooth surfaces. Finally, pixel lines excluding the ignored pixels are counted, and those with a count greater than two standard deviations of respective subjects are counted as positive. Results are verified from 20 different camera sites consisting of 65,291 images and using threshold values between 0.01 and 0.95. And at a threshold of 0.95, 95% of deer images and, 97% of wild pig images, 94% of raccoon images were correctly classified. This requires manual classification of day and night images, and different threshold values are needed for different animal species which must be tested. This application is developed for Time-lapse images where the sequence of images was present and might not work well for sensor-activated camera trap images.

The research was carried out by (Di Bitetti MS et al. [7]), to study the density, habitat use, and activity patterns of ocelots, a medium-sized neotropical cat. They used a methodology based on photographic records of individual ocelots obtained from camera traps forest was categorized to explore the possibility that ocelots prefer rainfall and conducted a

series of studies that showed that they preferred tropical and sub-tropical forests and dense and thorn shrubs in order to study the density of ocelots in the study area, they used a tool called capture, which provided the individual count of ocelots with a 95% confidence limits.

## 8 Conclusion and Future work

In this project, I have developed a technique that allows Biologists (users) to use some images they have that belong to a particular location and search for images taken in the same location in the whole dataset and developed a proof of concept to test it. This will require some manual process at first. This method achieved an accuracy of 88.48% for the task of identifying images from same location when the pixel match threshold was set to 30%, and 50 images were used initially. In contrast, a simple CNN developed was only able to achieve an accuracy of 44.68%, which is significantly lower. Using the same principle, a single image can be searched to identify to which location the image belongs if the whole dataset was organized (categorized) using this technique. 300 images (50 images per location) were used to test the performance of this approach for this task (search by image) and got an accuracy of 100% when the pixel match threshold value was set to 25%, but when it is set to 30%, the accuracy achieved was 95.6%. Although 50 images per location are considered low, only 1 image will usually be used when searching to find to which location it belongs. This technique initially generates a background image which will be compared with all the other images in the dataset for both the tasks mentioned. This background image generated was also used to identify the presence of the animal in the image once the images from a particular location were identified. The accuracy achieved for this task was 71.48%. Although state-of-the-art machine learning models achieved an accuracy of 95% for the same task, this can still be used in cases where machine learning models cannot be used (e.g., researchers close to the forest might have a low-power device and/or no internet access). Finally, I also checked how normalization could affect a CNN's performance when explicitly trained for the identification of animals, which can further increase the performance of the state-of-the-art models. I have also developed a web application that makes this process easier. The web app allows users to directly upload images and classify them. The researchers can then use this data and check whether the other data they have belongs to any of these classified data or not. It also allows the users to check if animals are present in the images or not and search by image in the dataset once the whole data has been classified.

The pixel comparison threshold "T" for **[Ni]** is set to 10% for all the processes mentioned above to account for minor changes in the pixel values due to different conditions. Increasing this value can increase accuracy as more pixels can be considered the same but can also result in the wrong classification since increasing it might result in allowing completely different pixels. Figure 18 shows how different the pixel can look with only a 10% difference in value. The 10% value is chosen by experimenting with the initial 50 images.

However, this technique requires some amount of manual work to identify images from the same location initially. Also, the separation of day and night images must be done manually. The future work for this approach focuses on developing machine learning models to auto-

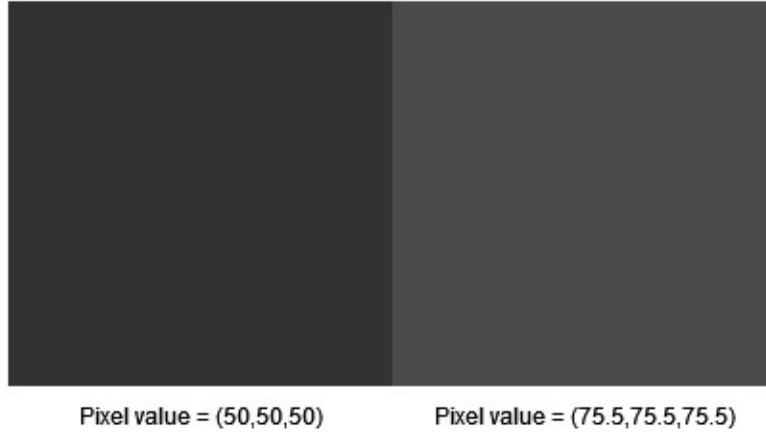


Figure 18: Pixel with 10% difference

mate some of the manual tasks. Light weight neural networks which can run on low-power devices like raspberry pi can be created and integrated into this. These neural networks can be used to identify and separate daytime images from nighttime images reducing some of the manual work required. In conditions where the researchers have high-power devices or not using a closed source dataset, using cumulative averaging to generate a background image can be replaced by a machine learning algorithm that goes through each pixel of every image and selects the best pixel for that position to create a better background image. However, this could be challenging as there is no sequence of images since multiple random events can trigger the sensors. Similar machine learning algorithms can also be developed to automatically set the threshold values based on the input images and update the threshold values for pixel intensities when required when comparing pixels.

While camera traps, images are not generally sequenced. Once identifying all the images present in the same location using the developed proof of concept, the metadata present in the image contains information about the date and time at which the image was created and a sequence of all these images similar to a time-lapse setting even though it will not be a perfect sequence can be created. A mixture of gaussian can then be used to identify and create a better background image which can also be updated after a set time period. This generated image can then be used to identify furthermore images that have the same background, information about time can be extracted from the new images, and this can then be used to approximately select which background image suits that time period. This will also help in better identification of the foreground, thereby allowing to better identify the presence of animals in an image, but this depends on how good the sequence of the images is when organized by this method and might give a bad result if the background image generated by this sequenced image is not good, since various factors can trigger the sensors and only a few images will be used to create the background image at a given time.

The interface of the web app will also be updated where the user can upload more images from a different location and automatically map this to the existing images the user has

already uploaded or add more images to already existing location (existing categories), allowing the users to set the thresholds manually as currently, the web application uses default values which might not be accurate in some cases (e.g., studying wildlife in deserts, etc.) a slide mechanism can be implemented that allows users to set the pixel intensity threshold values in order to better suit when search to which location a particular image belongs.

## 9 Reflection

with strong interest in computer vision and data processing, and research, we developed this project to solve some of the problems faced by biologist. however, this project became very challenging due to the following reasons.

### 9.1 Adapting different concepts

The goal is to produce a technique that will allow search by image in a camera trap setting and check if normalization of data reduces the accuracy of machine learning in a camera trap setting. Although machine learning sounds like a good solution, it proved to not be useful in this case and I have to research and learn different concepts of mathematics and how they can be applied in terms images and background image generation static camera surveillance (time lapsed images) and develop a different technique that can achieve the goal of the proposed task.

### 9.2 Time management

A lot of time was spent creating the required dataset, creating, and experimenting on various approaches that can achieve the proposed goal, manually analyzing all the results which took a lot of time since all images (12800 in total, but differed based on the data used for each solution) has to be manually verified, understand the drawbacks of current method and creating solutions to overcome the challenges faced. Many things went wrong especially due to dataset being camera trap as there is no research that focus on this particular dataset without using machine learning as a result all the ideas have to be created from scratch by analyzing solutions present in timelapse datasets and experimented which were time consuming.

### 9.3 Learning from mistakes

Multiple issues arise while trying to solve this task as a result of different methods experimented to solve this task and each having its own disadvantages. And I initially focused on trying to create new methods which adapts existing solutions to similar problems to the current problem . Instead, later I started focusing more on the problem statement and get better understanding of the existing solutions and its drawbacks, then create a solution to the problem statement rather than coming up with new methods and experimenting them. Dataset has to be manually verified every time. Doing this for experimentation of all the



initially created methods was challenging. Later this was made easier by spending more time initially in categorizing the data into different groups (brightness, size of animal, weather conditions, etc.) which made it easier to analyze the results since the performance is largely based on different attributes of the image.

## 9.4 Overcoming Challenges

The Initial approach of was to develop a technique that uses machine learning which allows us to search the dataset to identify images from the same location within the forest. However, there is no available dataset with GPS information of the forest and all required data needed to be manually processed. This raised issue because Computer vision models needs a lot of data in general to get better accuracies which need to be manually extracted from the datasets and this defeats the purpose since the goal is to develop solutions that use little data. Different forests need to be trained and so model developed for one cannot be used in another forest. Because of all these drawbacks, new ideas have to be developed and experimented.

A lot of statistical concepts and how they apply in case of data being images has to be understood.

All the available research that focus on the camera trap dataset uses machine learning. So, solution has to be developed from scratch understanding how statistics apply in static camera surveillance and timelapse datasets.

## 10 Lessons learned

**Machine learning might not always be the right solution.** In this case manually extracting the data required for training means going through almost all of the data by which the target audience would have already gotten the data before even training. And model trained on one forest cannot be used for a different forest without retraining which means manually extracting the data again. making it not a good solution in this case.

**Not to over complicate things**, as I have experimented with different techniques (like trying to use Mixture of gaussian for background image generation using the initial data) all of which were quite difficult to be implemented and spent a lot of time understanding Different statistical concepts and their application to images this was quite challenging because of all the complicated mathematics involved. However, and in the end a much simpler solution proved to be better than all the complicated things.

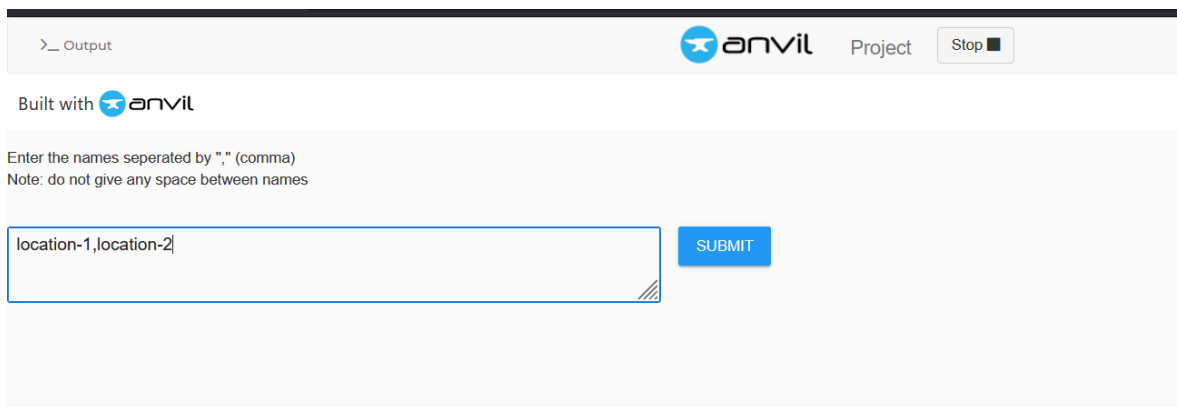
## 11 References

1. Jennifer L.Price Tack, B. S. (2016). AnimalFinder: A semi-automated system for animal detection in time-lapse camera trap images. Ecological Informatics Vol 36,

- 145-151. Retrieved from <https://www.sciencedirect.com/science/article/pii/S1574954116301121#!>
2. K.R.R. Swinnen, J. R. (2014). A novel method to reduce time investment when processing videos from camera trap studies. PLoS One. Retrieved from <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0098881>
3. Weinstein, B. (2015). MotionMeerkat: integrating motion video detection and ecological monitoring. *Methods Ecol. Evol.*, 6, 357-362. Retrieved from <https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210X.12320>
4. Z. He, R. K. (2016). Visual informatics tools for supporting large-scale collaborative wildlife monitoring with citizen scientists. *IEEE Circuits Syst. Mag.* Retrived from <https://ieeexplore.ieee.org/document/7404334>
5. A. Ardochini, L. C. (2008). Identifying elephant photos by multi-curve matching. *Pattern Recogn* vol 41, 1867-1877. Retrieved from <https://www.sciencedirect.com/science/article/pii/S0031320307005031>
6. Zivkovic, Z. &. (2006). Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 773-780. Retrieved from <https://www.sciencedirect.com/science/article/pii/S0167865505003521>
7. Di Bitetti MS, P. A. (2006). Density, habitat use and activity patterns of ocelots (*Leopardus pardalis*) in the Atlantic Forest of Misiones, Argentina. *Journal of Zoology*, 153-163. Retrieved from <http://dx.doi.org/10.1111/j.1469-7998.2006.00102.x>
8. [https://www.nlm.nih.gov/nichsr/stats\\_tutorial/section2/mod8\\_sd.html](https://www.nlm.nih.gov/nichsr/stats_tutorial/section2/mod8_sd.html)
9. Joshi, M. (n.d.). Comparison of Canny edge detector with Sobel and Prewitt edge detector using different image formats. *International Journal of Engineering Research & Technology (IJER)*. Retrieved from. <https://www.ijert.org/comparison-of-canny-edge-detector-with-sobel-and-prewitt-edge-detector-using-different-image-formats>
10. <https://www.geeksforgeeks.org/mahotas-getting-mean-value-of-image/>
11. <https://www.icsid.org/uncategorized/what-is-mean-and-standard-deviation-in-image-processing>
12. <https://inside-machinelearning.com/en/why-and-how-to-normalize-data-object-detection-on-image-in-pytorch-part-1/>
13. Kekre, Hemant & Sonawane, Kavita. (2009). Standard Deviation of Mean and Variance of Rows and Columns of Images for CBIR. Retrived from [https://www.researchgate.net/publication/242684384\\_Standard\\_Deviation\\_of\\_Mean\\_and\\_Variance\\_of\\_Rows\\_and\\_Columns\\_of\\_Images\\_for\\_CBIR](https://www.researchgate.net/publication/242684384_Standard_Deviation_of_Mean_and_Variance_of_Rows_and_Columns_of_Images_for_CBIR).

14. Sung, Jungmin & Choi, Bong-Yeol & Ha, Yeong-Ho. (2014). Image Thresholding Using Standard Deviation. Proceedings of SPIE - The International Society for Optical Engineering. 9024. Retrieved from [https://www.researchgate.net/publication/263053264\\_Image\\_Thresholding\\_Using\\_Standard\\_Deviation](https://www.researchgate.net/publication/263053264_Image_Thresholding_Using_Standard_Deviation)
15. Moulden Bernard, Kingdom Fred & Gatley Linda F. (1990). The Standard Deviation of Luminance as a Metric for Contrast in Random-Dot Images. Retrieved from <https://journals.sagepub.com/doi/abs/10.1068/p190079>
16. Mishra, Drpawan & Saroha, Gyanendra. (2016). A Study on Video Surveillance System for Object Detection and Tracking. Retrieved from [https://www.researchgate.net/publication/299978232\\_A\\_Study\\_on\\_Video\\_Surveillance\\_System\\_for\\_Object\\_Detection\\_and\\_Tracking](https://www.researchgate.net/publication/299978232_A_Study_on_Video_Surveillance_System_for_Object_Detection_and_Tracking)
17. Cristani, M., Farenzena, M., Bloisi, D. et al. Background Subtraction for Automated Multisensor Surveillance: A Comprehensive Review. EURASIP J. Adv. Signal Process. 2010, 343057 (2010). Retrieved from <https://doi.org/10.1155/2010/343057>
18. Malini, R. & Vasanthanayaki, C.. (2013). Average mean based feature extraction for image retrieval. 2013 IEEE Conference on Information and Communication Technologies, ICT 2013. 208-213. Retrieved from [https://www.researchgate.net/publication/261355159\\_Average\\_mean\\_based\\_feature\\_extraction\\_for\\_image\\_retrieval](https://www.researchgate.net/publication/261355159_Average_mean_based_feature_extraction_for_image_retrieval)

## 12 Figures



The screenshot shows the Anvil web application interface. At the top, there is a header bar with the Anvil logo, the word "Project", and a "Stop" button. Below the header, the text "Built with anvil" is displayed. The main content area contains a text input field with the placeholder text "location-1,location-2". To the right of the input field is a blue "SUBMIT" button. Above the input field, there is a note: "Enter the names seperated by ',' (comma)" and "Note: do not give any space between names".

Figure 19: web-app class names

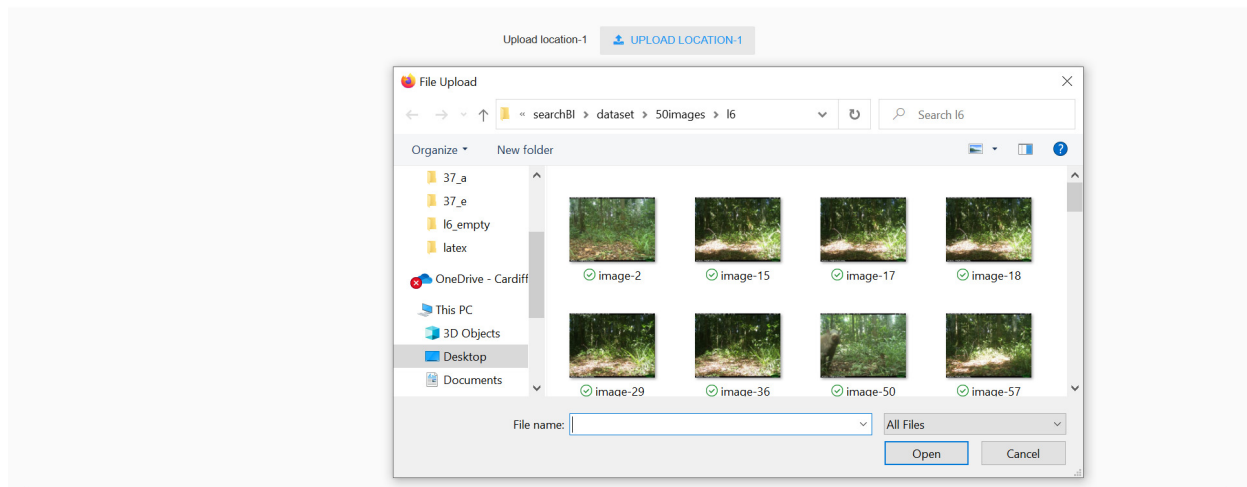


Figure 20: web-app uploading images

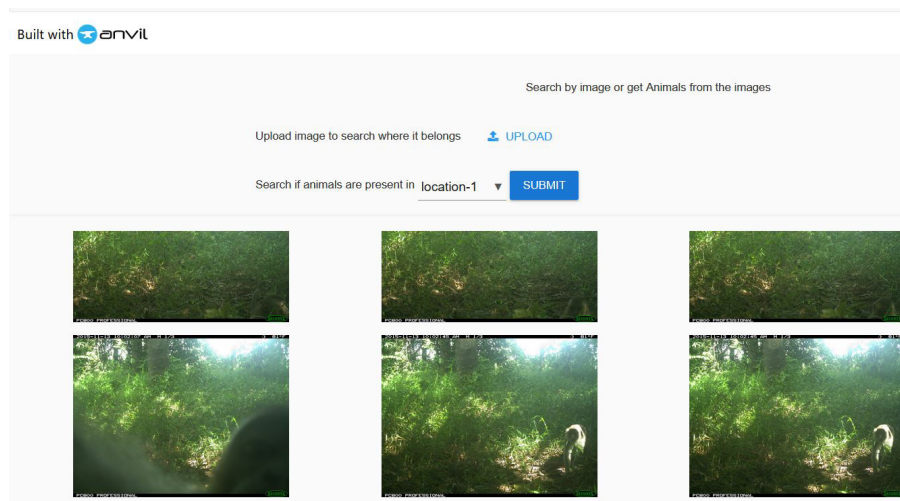


Figure 21: web-app check images

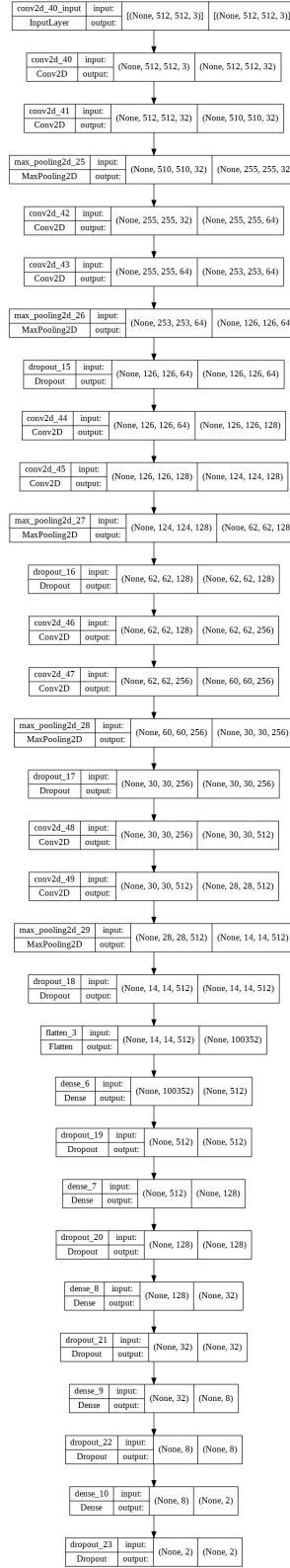


Figure 22: CNN structure