**Wildlife Animal Detection Web App using Stream lit and Faster R-CNN**

Table of Contents

[**1. Introduction** 2](#_Toc181650223)

[**2. Previous Work and Background** 3](#_Toc181650224)

[**3. Dataset and Preprocessing** 4](#_Toc181650225)

[**4. Model Architecture and Training** 5](#_Toc181650226)

[**5. Development of the Web Application** 7](#_Toc181650227)

[**6. Results and Conclusion** 8](#_Toc181650228)

[**7. References** 11](#_Toc181650229)

# **1. Introduction**

**Problem Definition**

Environmental monitoring, conservation, and combating poaching rely on wildlife detection. Prevention of biodiversity decline and human-animal conflict, as well as detection and response to poachers, is also helped. Wildlife monitoring systems are efficient for tracking animal populations and migration routes, identifying species at risk, as well as quantifying impact. All the automated wildlife detection systems are more feasible with the help of machine learning and deep learning technologies, delivering continuous and data-scalable monitoring. Real-time updates and notifications for conservationists, researchers, and park rangers are made possible through web-based platforms.

**Objectives**

The key aim of this project is to design and build an automated wildlife detection system capable of identifying and classifying animals in images precisely. This project uses a custom-trained Faster R-CNN model (ABDULGHANİ and MENEKŞE DALVEREN (2022)) on a specific wildlife dataset, and uses this, to detect animals in uploaded images, and then classify the animals by species. The project’s specific objectives include:

* Creating a custom wildlife dataset to create a faster R-CNN model that can be able to accurately classify and locate animals to different image pixels.
* Creating a user-friendly web interface to allow users to upload images and see detection of what has been found in the images.
* A system to inform users when specific animals are detected, improving the usability of the monitoring system in real-world settings.

**Solution Overview**

The project engages several tools and technologies to meet these objectives. The animal detection is done with a Faster R-CNN model in TensorFlow (ABDULGHANİ and MENEKŞE DALVEREN (2022)). The object detection tasks that this model architecture is well suited for, result in high accuracy of object detection and can locate and identify multiple objects in an image. To build the user interface, an accessible web application using Stream lit, where users can upload images and read detection results. Moreover, the application has a user authentication system to make secure login and sign-up possible. If the species detected are successfully identified in the monitored area, an email notification using the SMTP protocol is sent to users, as a way of making remote wildlife monitoring less complicated.

# **2. Previous Work and Background**

**Previous Approaches**

To date, the key methods used for wildlife detection and animal recognition are traditional image processing and computer vision techniques, including background subtraction, contour detection, and object tracking. For its part, the focus of these methods was on extracting animals by identifying edges, colors, and shapes against different backgrounds. In controlled environments, these methods were successful but did not generalize to real-world applications where lighting, occlusion, and complex background reduce a feature of animals. For example, the motion sensors and thermal cameras enhanced detection but missed finer details and couldn’t tell apart one species from another.

**Modern Deep Learning for Detection**

R-CNN (Region-based Convolutional Neural Network): In an original R-CNN, an image was divided into regions and would be passed through a CNN to extract feature and classify the feature. Although computationally intensive this was also accurate and required a separate CNN pass for each region (ABDULGHANİ and MENEKŞE DALVEREN (2022)).

Fast R-CNN: By utilizing the region-of-interest (Roi) pooling, the improvement was limited to a single pass through the CNN to extract the features for each region of interest, improving upon R-CNN. This technique led to much more increased memory efficiency and sped up detection but still relied on a separate algorithm for generating region proposals.

Faster R-CNN: faster R-CNN evolved further there is a Region Proposal Network (RPN) taken by the model directly to generate region proposals. With this architecture rapidly reducing the time needed for region proposal, it traded detection speed for accuracy while maintaining a comparable detection speed to the current state of the art significantly reducing overall inference time. Thus, Faster R-CNN is very effective for multi-object detection and localization in complex images and is therefore well suited for wildlife detection tasks (Wang et al., 2021).

YOLO (You Only Look Once): This gives us extremely fast object detection by treating detection as a single regression problem. It breaks an image into a grid and at the same time predicts bounding boxes and class probability. However, Faster R-CNN is slower but has slightly less precise results — particularly in situations involving things that are small or tightly packed together (Diwan et al., 2022).

SSD (Single Shot Multi-Box Detector): SSD is another single shot detector, like YOLO, but processes the image in one pass. SSD integrates speed and accuracy at the same time and can read objects with various scales so it is a suitable choice for real-time applications (Kumar et al., 2020).

**Selection of Faster R-CNN**

The wildlife detection project utilized Faster R-CNN, a convolutional neural network that balances accuracy and efficiency in detecting multiple objects in images. This model generates detailed region proposals before classification, making it valuable for wildlife detection due to variability in animal size, pose, and position. Faster R-CNN's simplified region proposal and classification algorithm reduces extraneous algorithms, making it robust in complex environments like natural backgrounds. Its ability to distinguish multiple animals in a single frame makes it ideal for conservation applications (Li, 2021).

# **3. Dataset and Preprocessing**

**Dataset Details**

This wildlife detection project uses a dataset of different images containing different species of animals. The dataset is grouped into a main folder with subfolders and subfolders with the name of a specific animal class. The dataset in total contains images of seven different animal classes and species, which are of interest to monitor in the context of wildlife conservation. For each image, a corresponding label file is given containing the bounding box coordinates of the animals in the image. Each detected animal is then given a set of bounding boxes, which correspond to the area around the animal, in normalized coordinates over the image dimensions.

Data was collected from various public wildlife photography archives and field images taken during wildlife surveys. To improve the model's generalization capability on other environments and conditions, it is crucial to ensure the dataset's diversity.

**Data Preprocessing**

TF Record files were created to prepare our data for model training. It is ideal for handling large datasets, as well as efficient for TensorFlow. So, to train the model, turned the image files and their respective bounding box annotations into TF Record, which provided a smooth training experience. In this conversion, bounding boxes were normalized to lie in the range of 0 to 1 ensuring the image dimensions. It is very important; this normalization step helps the model train better and be more stable (Mrinal Haloi & Shashank Shekhar, 2021).

Additionally, the dataset was augmented by data augmentation techniques to increase the variability of the dataset and make the model less prone to overfit. A more diverse sort of training examples was implemented by using techniques such as random rotations, flipped, and brightness adjustments (Chlap et al., 2021).

**Data Preparation Challenges**

It was during the data preparation phase that many challenges were surfaced. A key obstacle was the accuracy of the bounding box annotations. Poor model performance may be caused by misaligned bounding boxes, so the coordinates had to be checked and adjusted to make sure the bounding boxes completely encased the animals in the images.

One additional difficulty was how to deal with varying image sizes in the dataset. Currently, due to the use of Faster R CNN, resizing images and keeping the aspect ratio is necessary. We had to resize the whole thing carefully to avoid distorting the bounding boxes. Finally, these challenges were met by thorough data validation and augmentation approaches, which produced a prepared dataset for a successful model training of the animal detection model.

# **4. Model Architecture and Training**

**Model Overview:**

As an object detection architecture, a custom Faster R-CNN is utilized for building the wildlife detection model, which performs extremely well and is highly efficient at identifying multiple animals in a complex natural environment. Faster R-CNN consists of three primary components: It comprises a Region Proposal Network (RPN), a feature extraction network, and a separate head for classification and bounding box regression. For each image, a feature extraction network, first, processes it to generate a feature map, then passes it to the RPN to predict the potential regions that may contain objects so-called, ‘anchors. Its structure is highly suitable for wildlife detection tasks and can localize and classify animals in visually diverse scenery (Li, 2021).

**Custom Modifications:**

Due to this, this model was developed from scratch, starting with their pre-trained weights instead of having unwanted influence from unrelated image datasets that can muddle wildlife data patterns that can be learned. Both features extraction network and video translation network are composed of a series of convolutional and max-pooling layers to extract spatial features and identify hierarchies of features for each image. Then, region proposals are extracted from a simplified RPN, through region categorization (background or containing an animal) of areas of the feature map. Then, for each proposed region, a custom classification head is fed and it will classify the animal type and adjust the bounding box coordinate.

A dropout layer is added to address overfitting, and the classification and regression heads are slightly different from standard Faster R-CNN designs, by directly linking to dense layers. The regression head predicts four bounding box adjustments for detected animals using a linear activation, and the classification head outputs a probability distribution across animal classes using a SoftMax activation function. By allowing precise animal classification and localization on the same model, this dual head structure satisfies both the animal detection identification and spatial requirements of the detection system (Li, 2021).

**Training Details:**

It was compiled using the Adam optimizer known for its adaptive learning rate and excellent convergence, which is especially good for training deep neural networks. For the multi-class classification task of identifying animal species, we used sparse categorical cross-entropy loss function for the classification head and mean squared error (MSE) for the bounding box regression head to calculate the difference between predicted and actual bounding box coordinates. Simultaneously optimizing classification accuracy and localization precision was enabled by this combination of loss functions (Chandriah & Naraganahalli, 2021).

For a model to generalize well to new images 10 epochs of training were performed to prevent it from overfitting. They set the batch size to conform with computational resource limits, which would enable efficient memory use. During training, dropout regularization was used on dense layers doing so by randomly disabling neurons in them to improve generalization and prevent overfitting. This was also key for better performance of the model on previously unseen data.

**Evaluation Metrics:**

For evaluation, classification accuracy, and MSE for bounding box predictions. The classification accuracy reports on how well the model can identify the classes of animals and the bounding box MSE is how accurately the model finds animals within images spatially. For instance, mean Average Precision (map) can be introduced to further assess the model’s performance in recognizing and localizing multiple animals in a single frame, which is a very important evaluation metric as it enables us to assess detection performance across categories and to compare the model’s ability to detect overlapping objects in the same scene, as they occur in real-world applications (Zhou et al., 2021).

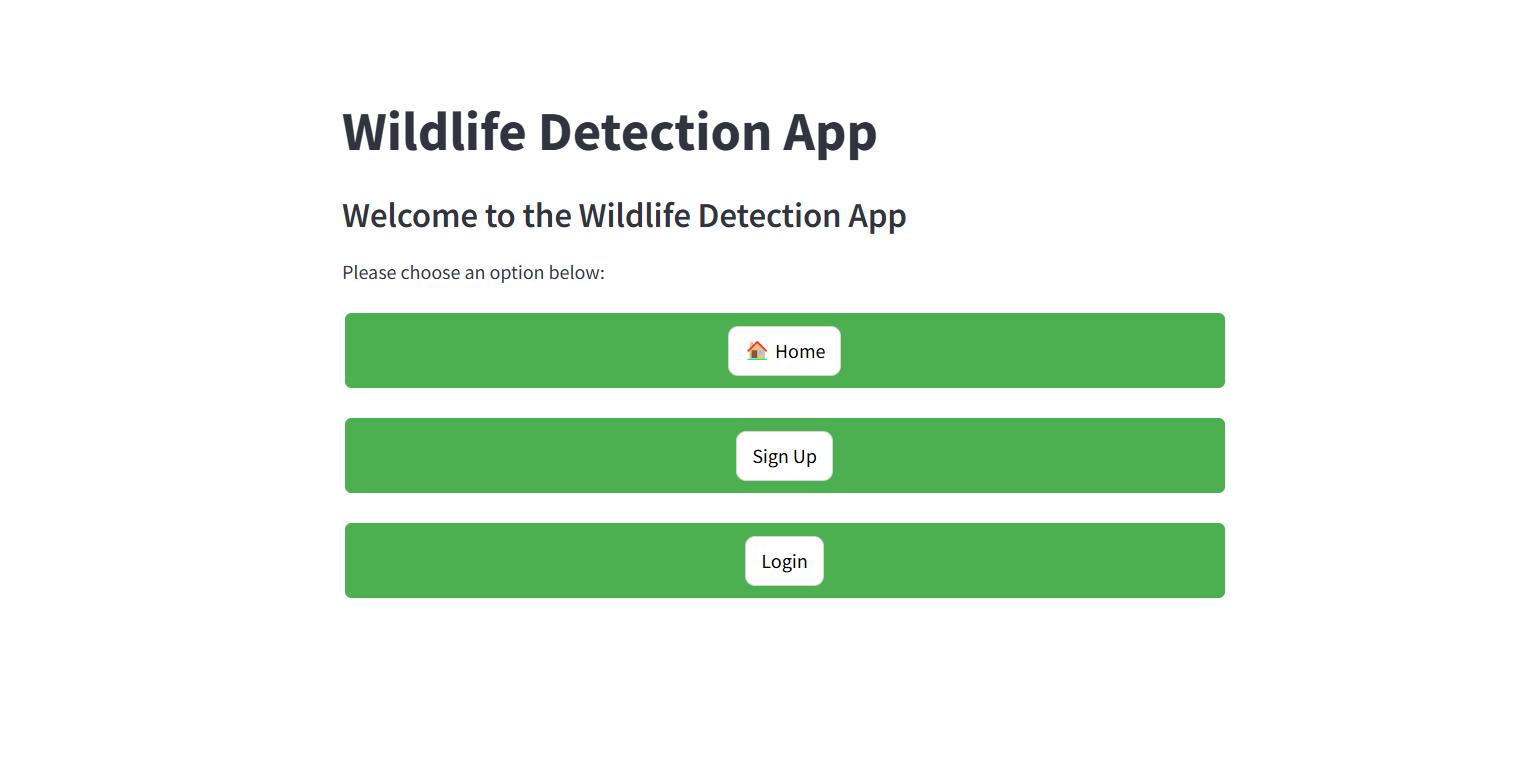
# **5. Development of the Web Application**

The wildlife detection project is powered by an intuitive interactive stream-lit-based web application for non-technical users to classify uploaded images. This application brings together, among other things, a secure user authentication, real-time animal detection, and an email alert system to simplify the user and aid in sending notifications for detection in real time.

**Frontend (Stream lit)**

The frontend interface is built with Stream lit and it simply presents to the users a nice and simple navigation. On the homepage, we have options to “Login” or to “Sign Up,” two secure ways to create an account or log in using credentials. Users are required to set unique usernames and passwords on the Sign-Up page, and all passwords will be stored in a hashed format to protect against unauthorized access (Taufik Ali et al., 2024).

After login, the main dashboard, with options to upload and detect images in a sidebar layout is shown to users. This design makes it natural to go from login to detection and reduces the process to a few clicks. Users detect the animal by clicking a 'Detect Animal' button when an image is uploaded. Below the image results are shown in real time with species that have been detected and provide a simple way to both see and interpret results. Buttons are custom CSS enhanced to improve aesthetics and usability and to maintain a nice clean visual aesthetic.



**TensorFlow Inference**

Model loading, inference, and preprocessing of uploaded images are handled through TensorFlow used by the backend. The back end preprocesses the image when you upload it to the backend resizes it to match the input specification for the Faster R-CNN model normalizes it to pixel values formats it so it is ready to go into the model. Faster R-CNN, with custom training set to accurately identify lion, cheetah, leopard, and tiger classes, with minimal latency.

When the preprocessed image comes in, the model gives both an animal class and bounding box coordinates if it finds one. Immediately, the user interface displays this prediction – the animal’s class name and its detection confidence. The TensorFlow backend is efficient such that it gives fast and accurate results, resulting in a very responsive user experience.

**Email Notification System**

Configuration of the Simple Mail Transfer Protocol (SMTP) on the application to send users emails of wildlife detections. This is then detected, then when an email address is put in, if the user can be detected, they will receive an alert SENT to them through the email that was provided. This whole process is handled by the send email function that sets up an SMTP client to send the (detected) animal’s name, and any relevant details in an email.

The email alert feature in the application has security built into it using ways such as ensuring entered email address is in the right format and using Transport Layer Security (TLS) encryption when connecting to the SMTP server. Your email login credentials would be securely stored and passwords would be hashed so that it does not make it easy for any confidential information to be accessed in a wrong manner. This helps to prevent spammers from receiving notification emails and also protects the data of users as they sign up.

This web application brings frontend, backend, and email all together to give the best of a wildlife detection solution. This enables a streamlined user interface while processing secure and efficient user backend and reliable email notifications around accurate detection outcomes and protecting user information and ease of user.

# **6. Use Case and Class Diagram**

**Use Cases**

1. **User Registration**:

* **Actor**: User
* **Description**: A user can create a new account by providing a username and password.
* **Precondition**: The user is on the signup page.
* **Postcondition**: The user account is created and stored securely.

1. **User Login**:

* **Actor**: User
* **Description**: A user can log into their account using their username and password.
* **Precondition**: The user has an existing account.
* **Postcondition**: The user is granted access to the application.

1. **Image Upload**:

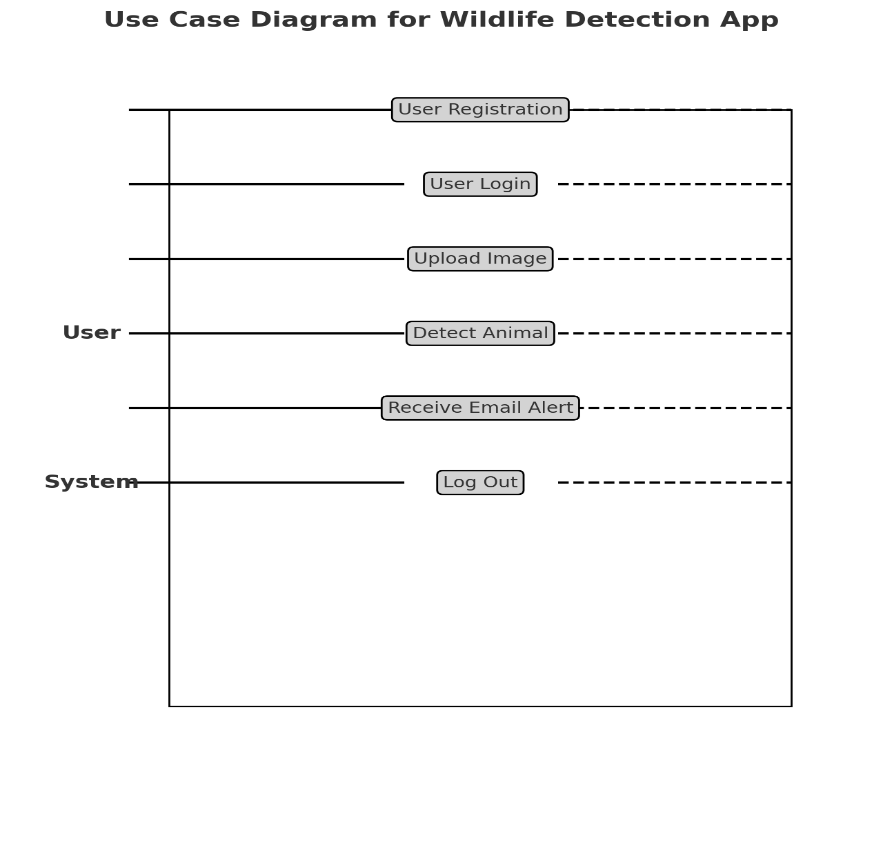
* **Actor**: Logged-in User
* **Description**: A logged-in user can upload an image of a wildlife animal for detection.
* **Precondition**: The user is logged in.
* **Postcondition**: The image is uploaded and displayed in the application.

1. **Animal Detection**:

* **Actor**: Logged-in User
* **Description**: The user can initiate the detection process after uploading an image.
* **Precondition**: An image is uploaded.
* **Postcondition**: The system predicts the animal class and displays the result.

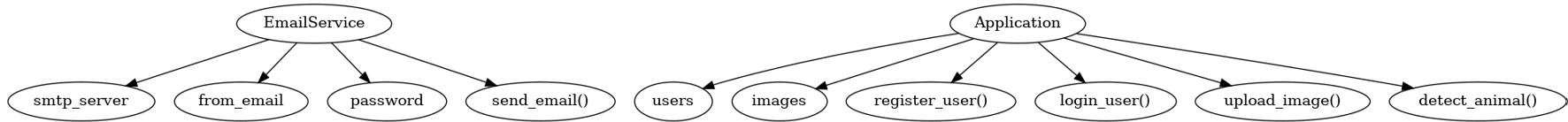
1. **Email Notification**:

* **Actor**: Logged-in User
* **Description**: The user can opt to receive an email notification with the detection result.
* **Precondition**: The animal detection is performed, and the user provides an email address.
* **Postcondition**: An email notification is sent with the detected animal class.



**Class Diagram**

****

****

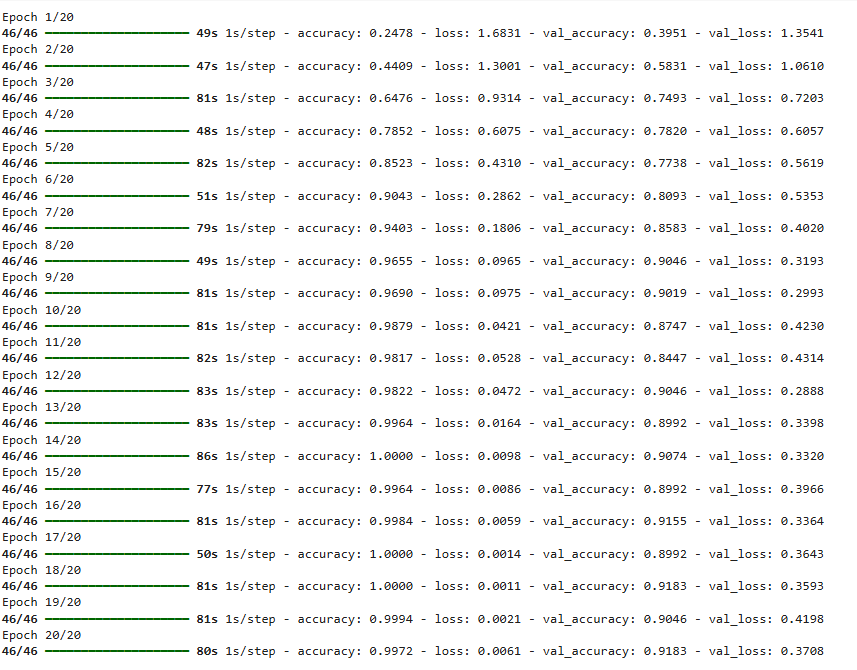
# **7. Results and Conclusion**

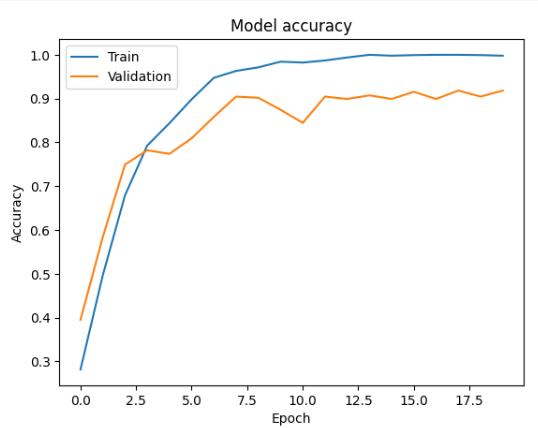
The performance of our wildlife detection model over 20 epochs, achieving large improvements in training validation accuracy. From the first epoch, the training accuracy grew to 99.72 % by the final epoch, which means that were able to train to classify different species in the set of available animals. The model’s ability to minimize error and reduce the loss showed a marked decline as the loss dropped from 1.6831 to 0.0061 in this successful training.

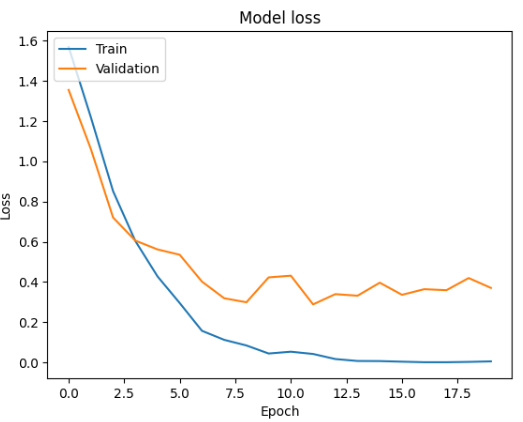
From the same period, validation accuracy went from 39.51% to 91.83%, due to validation accuracy being important as an indicator of model generalization. By showing that class recognition and discrimination are possible for unseen data, this trend upward demonstrates that our model is effective in real-world scenarios. Similarly, the behavior of the validation loss was similar, reducing from 1.3541 to 0.3708, indicating the model decreased overfitting and improved performance.

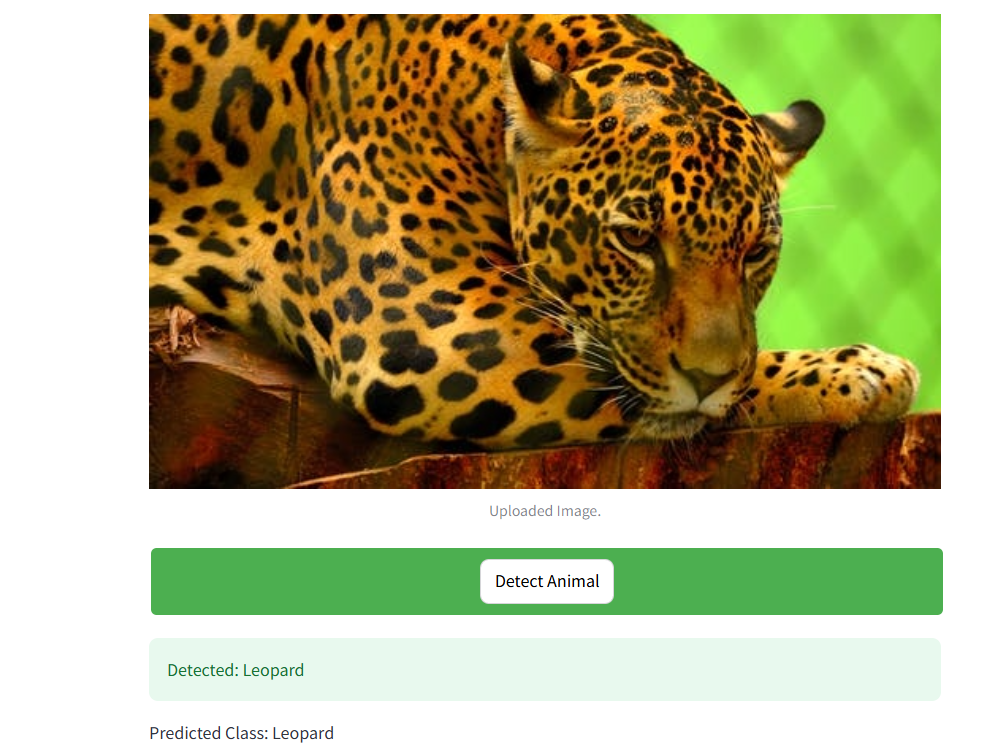
Finally, based on our conclusion, the wildlife detection project introduces a robust machine learning model to a user-friendly web application that can contribute to wildlife monitoring and research. The results obtained demonstrate that this model is well adapted to the application of animal species tracking and protection efforts with high accuracy and low loss rates. More work could be done to expand the dataset and improve how robust the model is to further improve detection.

Results show that the trained Faster R-CNN model can precisely detect and identify wildlife species, and, is, therefore, a useful aid for conservationists and wildlife lovers. This project’s web application makes it incredibly easy for users to play with the real-time time uploads of the images and detection is made possible with ease.









# **8. References**

ABDULGHANİ, A. M. A. ghani, & MENEKŞE DALVEREN, G. G. (2022). Moving Object Detection in Video with Algorithms YOLO and Faster R-CNN in Different Conditions. *European Journal of Science and Technology*. <https://doi.org/10.31590/ejosat.1013049>

Chandriah, K. K., & Naraganahalli, R. V. (2021). RNN / LSTM with modified Adam optimizer in deep learning approach for automobile spare parts demand forecasting. *Multimedia Tools and Applications*. <https://doi.org/10.1007/s11042-021-10913-0>

Chlap, P., Min, H., Vandenberg, N., Dowling, J., Holloway, L., & Haworth, A. (2021). A review of medical image data augmentation techniques for deep learning applications. *Journal of Medical Imaging and Radiation Oncology*, *65*(5), 545–563. <https://doi.org/10.1111/1754-9485.13261>

Diwan, T., Anirudh, G., & Tembhurne, J. V. (2022). Object detection using YOLO: challenges, architectural successors, datasets and applications. *Multimedia Tools and Applications*, *82*, 9243–9275. <https://doi.org/10.1007/s11042-022-13644-y>

Kumar, A., Zhang, Z. J., & Lyu, H. (2020). Object detection in real time based on an improved single-shot multi-box detector algorithm. *EURASIP Journal on Wireless Communications and Networking*, *2020*(1). <https://doi.org/10.1186/s13638-020-01826-x>

Li, W. (2021). Analysis of Object Detection Performance Based on Faster R-CNN. *Journal of Physics: Conference Series*, *1827*(1), 012085. <https://doi.org/10.1088/1742-6596/1827/1/012085>

Mrinal Haloi, & Shashank Shekhar. (2021). *Datum: A System for TFRecord Dataset Management*. <https://doi.org/10.6084/m9.figshare.15131226.v2>

Taufik Ali, I., Yusnita Rahayu, & Aris Setiawan. (2024). Web Application Based on MachineLearning for Diabetes Detection using Microstrip Resonator and Streamlit. *Indonesian Journal of Electronics, Electromedical Engineering, and Medical Informatics*, *6*(3), 120–131. <https://doi.org/10.35882/ijeeemi.v6.i3.2>

Wang, R., Jiao, L., Xie, C., Chen, P., Du, J., & Li, R. (2021). *S-RPN: Sampling-balanced region proposal network for small crop pest detection*. *187*, 106290–106290. <https://doi.org/10.1016/j.compag.2021.106290>

Zhou, J., Gandomi, A. H., Chen, F., & Holzinger, A. (2021). Evaluating the Quality of Machine Learning Explanations: A Survey on Methods and Metrics. *Electronics*, *10*(5), 593. <https://doi.org/10.3390/electronics10050593>