A

Report

on

"*Wild Eye: Wildlife Detection on The Road*"

Prepared by

Bigyan Sapkota

Ali Azgar Katha

Jyoti Bisht

Under the guidance of

Himanshu Patel

Submitted to

The Artificial Intelligence and Data Analytics Program

School of Information and Communications Technology

Saskatchewan Polytechnic

CAPSTONE PPOJECT REPORT

APRIL 2024

ABSTRACT

Recently there have been numerous accidents on the road involving wildlife. This not only affects the humans and their life, but it also poses a great threat to the wildlife present on the area. These road accidents disturb the natural movement and habitat of the animals and creates a problem in managing a smart and safe traffic. In this modern era, Artificial Intelligence plays a vital role to increase safety and efficiency in almost every field. Our project "**Wild Eye: Wildlife Detection on The Road**" is an innovative AI system to monitor roads to detect wildlife and ensure the safety of both animals and humans. This innovative solution is a customized system that will detect animals with the help of AI and generate warning to the drivers to take appropriate actions. It enhances the safety measures on road, animal welfare and ensure smart traffic management by notifying authorities and drivers about the animal presence on the road. Our project is based on CNN (convolutional Neural Network) architecture, and we used TensorFlow to build and train model. We collected dataset, processed it, and built the required model to detect and identify wildlife on the road. This project aims to ensure road safety and protect wildlife from the accidents and helps to build a smart traffic management. We considered several ethical aspects of animal safety, privacy on this project on the development process.

ACKNOWLEDGEMENT

We would like to express our gratitude to our instructor Mr. Himanshu Patel for guiding and providing feedback throughout the project development period. We would also like to thank AIDA department at Saskatchewan Polytechnic for providing this opportunity to work on this innovative idea.

I. Introduction

A. Project Overview

In this Wildlife Detection project, we will be using Artificial Intelligence technology to detect wild animals on the road and classify them. And then notifies driver by warning text message on dashboard so that they can be more conscious while driving. It will be a step to create safer circumstances for both humans and animals by enhancing road safety, animal welfare and smart traffic management and reduce wildlife related road accidents.

B. Problem Statement

The incidence of road accidents involving wildlife is increasing, posing risks to both road safety and wildlife conservation. This initiative aims to improve road safety and safeguard wildlife by providing timely warnings to drivers. Current systems lack the accuracy and specificity required to precisely detect the presence of animals and ensure their safety, creating a gap in their intended functionality. Our project seeks to reduce this gap and enhance the effectiveness of systems addressing these concerns.

C. Project Objective:

Our project aims to overcome the following problems.

1. Rising Wildlife-Related Road Accidents: The escalating number of road accidents involving wildlife is a significant issue. This poses threats to the safety of human. drivers and the well-being of wildlife.
2. Threat to Road Safety and Wildlife Preservation: Mentioned accidents not only jeopardize road safety but also pose a threat to wildlife preservation efforts. The dual impact on both humans and animals emphasizes the severity of the issue.

II. Related Work

- This study is used to identify object not limiting itself to animals. It used upgraded SSD and multilayered CNN model for identification. This study has been primarily done for images and later upgraded for moving images also. On our project, we will try to implement this recognition on video from the road and primarily focus on animals. [Vaishnavi, K.]
- In this study, algorithm based on YOLOv5 has been used to identify animals. Standard images clicked have been used to train the model and a lack of consideration on environmental changes, animals natural habitats haven been given much attention. The initial study was not able to clearly detect and identify overlapping objects, and they have proposed a new updated algorithm to tackle this issue. We will try to take this in consideration to resolve the overlapping object detection issue. [Zhang, Mingyu]
- This research has been done to identify small object present in single shot images. They have compared YOLOv3 and Retina net approach, where YOLOv3 was two times faster in processing time but Retina net was two time accurate. This project work has its limitation on what kind of bojects it identifies as it has been trained with a limited dataset. We will try to avoiud this on our project as there are a lot of easily available reliable dataset for animal. [Tivelius, Malcolm.]

III. Methodology

A. Data Collection

To carry out our project, we require a well-annotated dataset that encompasses diverse scenarios, wildlife, lighting variations, and changing weather conditions. The dataset should consist of meticulously labeled images and video frames. Our data collection approach will be methodical and systematic, ensuring the acquisition of comprehensive real-world data with ample annotations. Given the time constraints, our strategy involves seeking existing datasets from similar studies and projects. If necessary, we will engage in the annotation process for these datasets. Our search will encompass related studies, university projects, similar ongoing initiatives, and platforms such as Kaggle to find datasets that align with our project requirements. Initial research has identified several example datasets, including COCO (Microsoft Common Objects in Context), ImageNet (https://www.image-net.org/), and https://www.kaggle.com/datasets/andrewmvd/animal-faces

Finally, we come up with the data set of 12 classes with approximately 5000images. That data was well-annotated.

https://universe.roboflow.com/kiran-kumar-g-rir6u/wildlife-security-system/dataset/2

B. Data Preprocessing

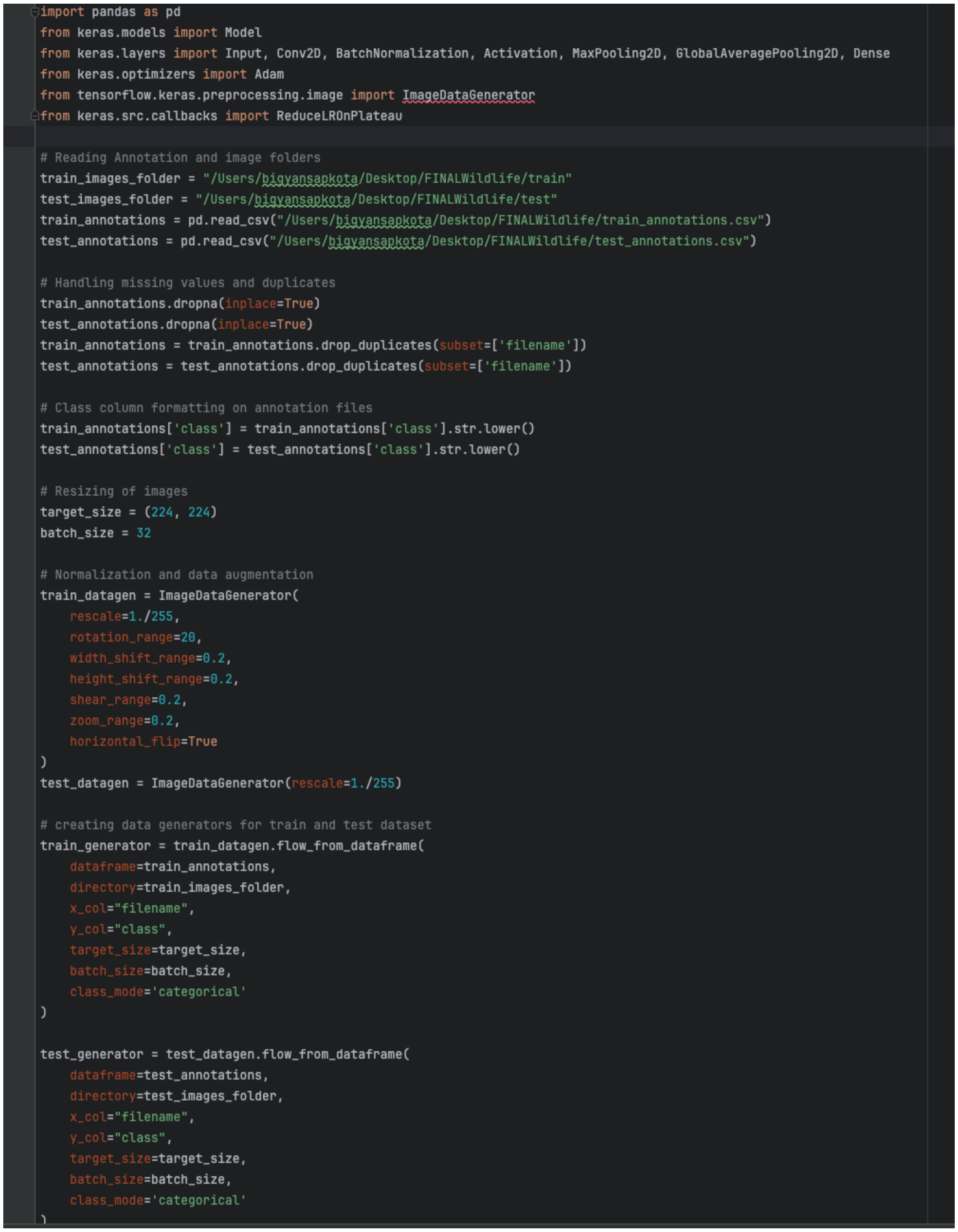
Our approach involves developing a model for wildlife detection on roads using a Convolutional Neural Network (CNN) with TensorFlow. To mitigate overfitting, we'll apply data augmentation. OpenCV will handle real-time image and video processing, while Scikit-learn will complement the workflow for calculations and data splitting.

Train_annotation_path and Test_annotation_path are the annotated files with file_name and class names columns. Test_image_folder and test_images_folder are the test and train images respectively. Then we read the csv files with pandas' library. We also removed the null and duplicates and changed the class names to lower case.

Image was originally of size (640 ,640) which will be changed by target_size = (224, 224) and we are reducing it for processing efficiently and less memory usage also it is the standard sizes of input images. Batch Size is the number of samples that will be passed through the network at one time.

we are doing data augmentation. ImageDataGenerator is a utility class of keras library. This class is designed for data preprocessing and argumentation during the training of deep learning models. Rescale is the parameter used for feature scaling. It will scale the pixel values of images by diving each pixel value by 255. It will rescale the pixel value to be 0 to 1.

Flow_from_dataframe() is the method class ImageDataGenerator for generating baches of augmented data when file name and label are stored in data frame.



```python
import pandas as pd
from keras.models import Model
from keras.layers import Input, Conv2D, BatchNormalization, Activation, MaxPooling2D, GlobalAveragePooling2D, Dense
from keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.src.callbacks import ReduceLROnPlateau

# Reading Annotation and image folders
train_images_folder = "/Users/bigyansapkota/Desktop/FINALWildlife/train"
test_images_folder = "/Users/bigyansapkota/Desktop/FINALWildlife/test"
train_annotations = pd.read_csv("/Users/bigyansapkota/Desktop/FINALWildlife/train_annotations.csv")
test_annotations = pd.read_csv("/Users/bigyansapkota/Desktop/FINALWildlife/test_annotations.csv")

# Handling missing values and duplicates
train_annotations.dropna(inplace=True)
test_annotations.dropna(inplace=True)
train_annotations = train_annotations.drop_duplicates(subset=['filename'])
test_annotations = test_annotations.drop_duplicates(subset=['filename'])

# Class column formatting on annotation files
train_annotations['class'] = train_annotations['class'].str.lower()
test_annotations['class'] = test_annotations['class'].str.lower()

# Resizing of images
target_size = (224, 224)
batch_size = 32

# Normalization and data augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)
test_datagen = ImageDataGenerator(rescale=1./255)

# creating data generators for train and test dataset
train_generator = train_datagen.flow_from_dataframe(
    dataframe=train_annotations,
    directory=train_images_folder,
    x_col="filename",
    y_col="class",
    target_size=target_size,
    batch_size=batch_size,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_dataframe(
    dataframe=test_annotations,
    directory=test_images_folder,
    x_col="filename",
    y_col="class",
    target_size=target_size,
    batch_size=batch_size,
    class_mode='categorical'
)
```

C. Model Development

After all the data preprocessing had been completed, we moved to another phase of development. This phase is the backbone of any machine learning development where we build the model per our requirements. The first step of model development is to choose the best model that satisfies our needs and works perfectly with the dataset we have. For classification problem, we had numerous model options to choose from. We had models like DenseNet, VGGNet, MobileNet, EfficientNet, Sequential and InceptionV3 etc. that we could use for our project. Most of these models are computationally expensive and models like MobileNet sacrifices accuracy for efficiency. Some models like sequential and MobileNet also are better aligned for smaller datasets. Thus, we decided not to proceed with any of these models and while searching for the best model we came to ResNet. ResNet is a model based on residual blocks and its ability to handle skip connections helps to mitigate the vanishing gradient problem making it perfect model for our project. If we were able to use the pre trained ResNet model and finetune it with our data, we could have achieved better accuracy. But, due to our project requirements, we had to build the ResNet like model from scratch with residual blocks and layers.

ResNet Model:

ResNet is a deep learning architecture based on residual blocks and layers. The primary purpose of this model is to tackle the vanishing gradient problem which usually occurs during the training process of the model. The residual blocks in ResNet have shortcut skip connections to help in mitigating vanishing gradient. Most of the other models have the vanishing gradient problem which refers to the condition where gradients become super small on the backpropagation process through different layers. This makes the learning process difficult. To tackle this issue ResNet has skip connection or shortcut paths which allows the gradient to flow directly in the network. It gives an option to skip multiple layers and can connect output to input layer directly. If there are issues learning the characters, then the network can come back to identity mapping. Multiple residual blocks are stacked together, and the more blocks the deeper the model becomes.

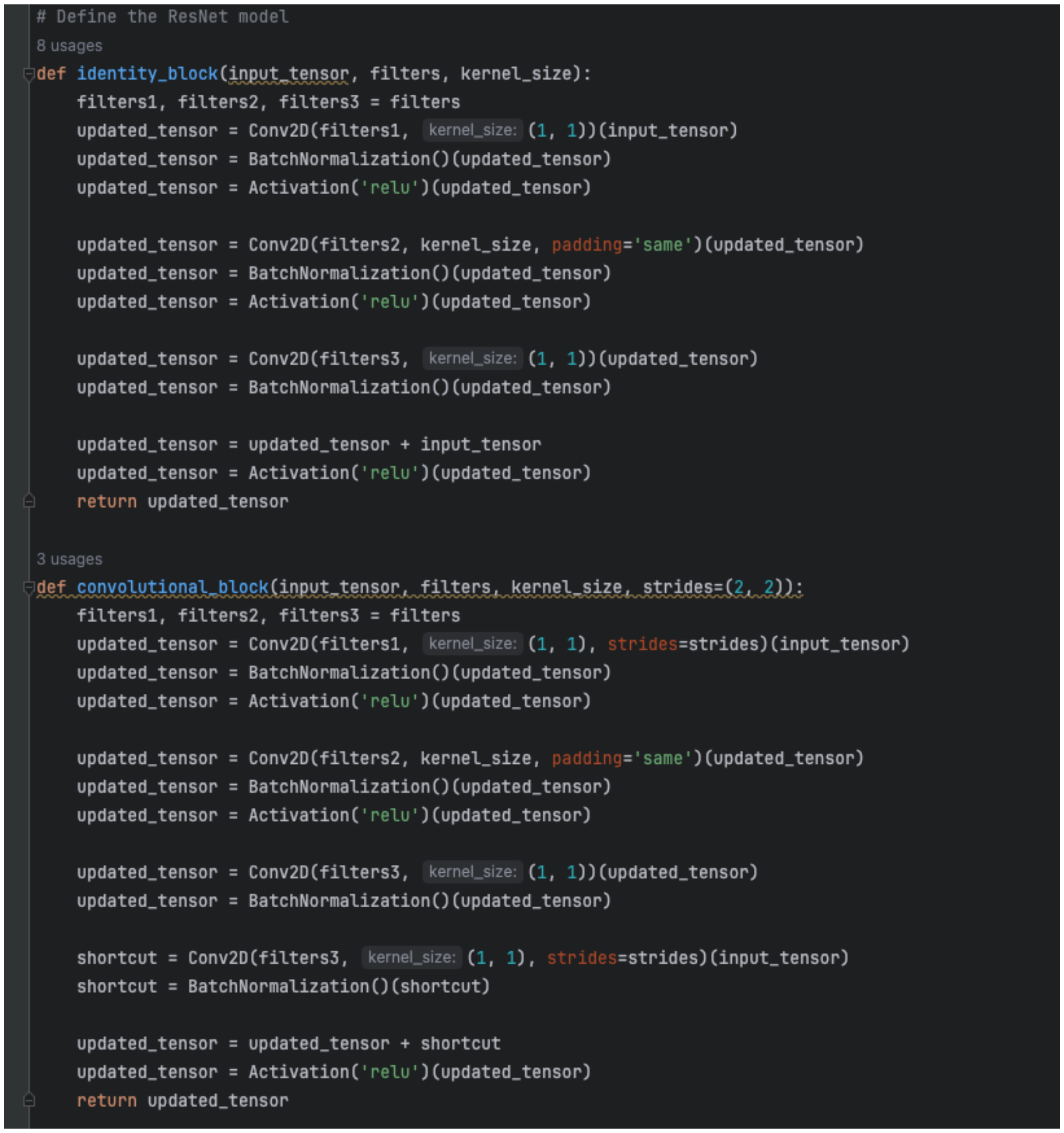
Residual Block:

Residual blocks are the backbone of ResNet model, as the name itself abbreviates to Residual Network. The primary goal of these residual blocks is to make the model learn about the difference in output and input rather than learning the output directly. The difference between output and input is termed as residual mapping. So, the model learns this residual mapping, learns how the input and output have differed and this information gets backpropagated to learn. The residual blocks contain two major parts, the shortcut path and the residual path. Firstly, input is passed through a convolutional layer with small kernel size to learn few features. Then we apply batch normalization to normalize the activation, and ReLU activation function to introduce non-linearity. Then another convolutional layer is applied to extract more features. A skip connection is introduced where the output of the input can be directly added to the final output. For example, if x is the input,f(x) is the output of first layer and h(x) is the output of second layer, and a shortcut path is introduced, then the mathematical representation is h(x) = f(x)+ shortcut. This allows residual learning and allows gradient to flow easily during backpropagation.

Residual Blocks in Our Model:

Our model is based on ResNet as well. But we have opted to create different residual blocks from scratch rather than using any pre trained model. Firstly, we introduced an identity block. Here we defined an identity block which takes input tensor, filters and kernel size as the arguments, and the filter is then expanded to three filters, 1,2, and 3. Inside this block, we introduced three convolutional layers each taking respective filters, followed by batch normalization to normalize activation and a ReLU activation function to introduce non-linearity. At the end the output of the third and the last convolutional layer is added with input tensor to give the output of this identity block. Then we defined another residual block as convolutional block. This block has the same component as the identity block above but in addition it introduces a shortcut or skip connection. The shortcut is added to the output of the third and the last convolutional layer to give the output on this convolutional block. This skip connection helps to resolve the vanishing gradient problem by allowing the gradient to pass easily during the backpropagation.

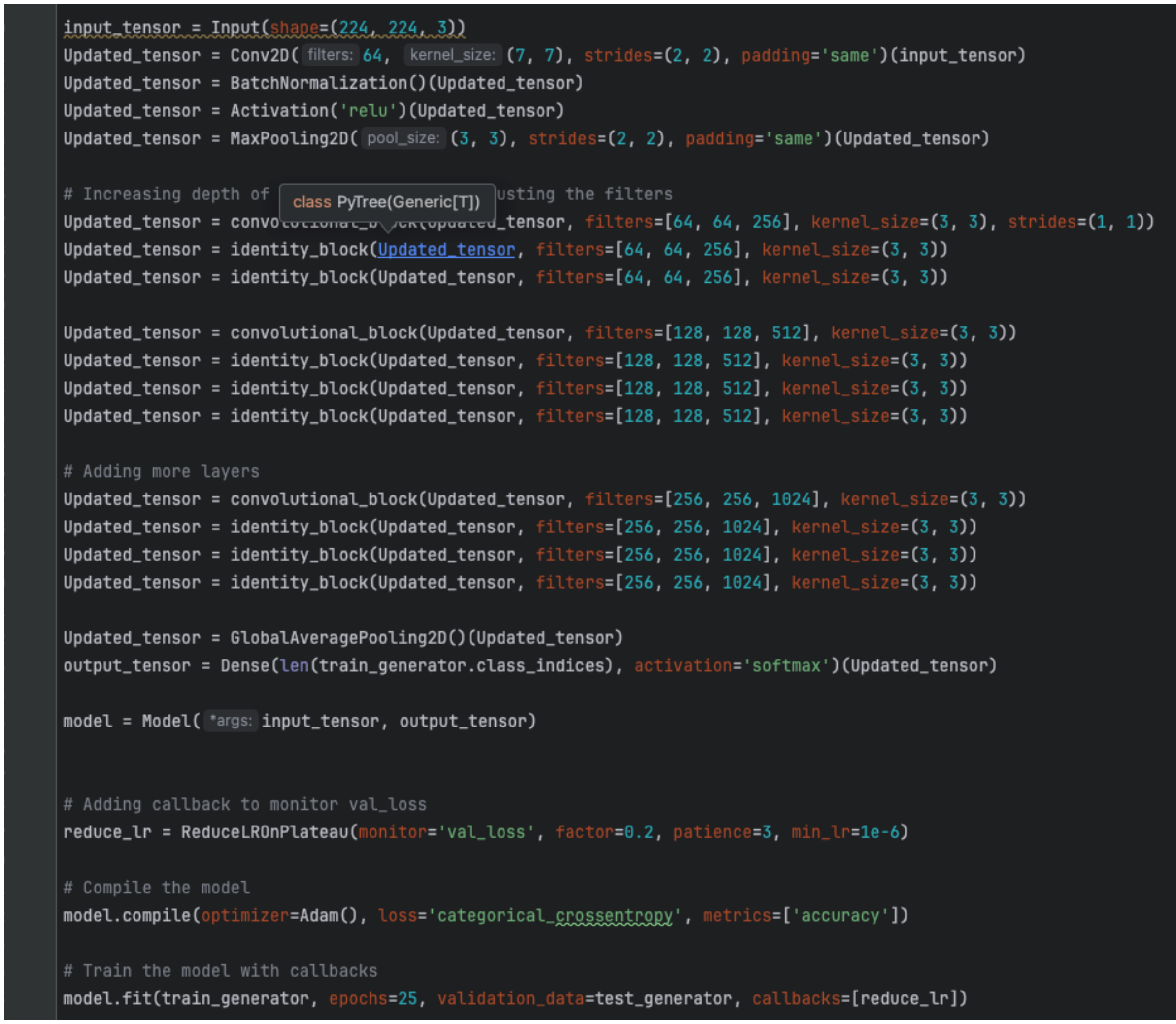
Now we have defined an input tensor with shape (224,224,3) meaning dimension of the image (224,224) and has 3 color channels typically RGB. Then, we applied a convolutional layer on the input_tensor with 64 filter, [7,7] kernel size which means convolutional operation being performed using 7/7 filter, and [2,2] stride meaning each time the filters move 2/2 pixels on each direction. Padding same has been applied to ensure the output has same dimension as the input. This is followed by batch normalization to normalize inputs of a layer to stabilize and accelerate the training process and ReLU activation is applied to introduce nonlinearity. Then maxpooling2d is applied which helps to reduce the spatial dimension of feature map. This takes input tensor representing feature map from the convolutional layer, and maxpooling acts on small rectangle nonoverlapping regions. On each window, it extracts the maximum value which helps it to gather the most important feature.



```
# Define the ResNet model
8 usages
def identity_block(input_tensor, filters, kernel_size):
    filters1, filters2, filters3 = filters
    updated_tensor = Conv2D(filters1, kernel_size: (1, 1))(input_tensor)
    updated_tensor = BatchNormalization()(updated_tensor)
    updated_tensor = Activation('relu')(updated_tensor)

    updated_tensor = Conv2D(filters2, kernel_size, padding='same')(updated_tensor)
    updated_tensor = BatchNormalization()(updated_tensor)
    updated_tensor = Activation('relu')(updated_tensor)

    updated_tensor = Conv2D(filters3, kernel_size: (1, 1))(updated_tensor)
    updated_tensor = BatchNormalization()(updated_tensor)

    updated_tensor = updated_tensor + input_tensor
    updated_tensor = Activation('relu')(updated_tensor)
    return updated_tensor

3 usages
def convolutional_block(input_tensor, filters, kernel_size, strides=(2, 2)):
    filters1, filters2, filters3 = filters
    updated_tensor = Conv2D(filters1, kernel_size: (1, 1), strides=strides)(input_tensor)
    updated_tensor = BatchNormalization()(updated_tensor)
    updated_tensor = Activation('relu')(updated_tensor)

    updated_tensor = Conv2D(filters2, kernel_size, padding='same')(updated_tensor)
    updated_tensor = BatchNormalization()(updated_tensor)
    updated_tensor = Activation('relu')(updated_tensor)

    updated_tensor = Conv2D(filters3, kernel_size: (1, 1))(updated_tensor)
    updated_tensor = BatchNormalization()(updated_tensor)

    shortcut = Conv2D(filters3, kernel_size: (1, 1), strides=strides)(input_tensor)
    shortcut = BatchNormalization()(shortcut)

    updated_tensor = updated_tensor + shortcut
    updated_tensor = Activation('relu')(updated_tensor)
    return updated_tensor
```

Now, we build deeper networks to extract more features correctly. Convolutional and identity blocks are stacked together with changing filters. By combining convolutional and identity blocks, we can get an effective way to build our network and train the model by mitigating the vanishing gradient problem using skip connections.

Then we introduced GlobalAveragePooling2D to reduce the spatial dimension by taking the average value of each feature map. It also helps to reduce the number of parameters in the model. Then output tensor is introduced with fully connected dense layer with the layers equal to the number of classes of animals. The activation function SoftMax is used which converts raw score to the probability for each class. After this we compiled the model with input and output tensors, with adam optimizer, categorical crossentropy for loss function and accuracy to monitor the model performance during training. Then we trained the model on train datasets with 50 epochs, and learning rate reduction callback has been used. This callback is used to adjust the learning rate dynamically, by reducing learning rate when monitored metric like val_loss has stopped improving. The factor and min_lr give the factor by which learning rate will be reduced and the minimum rate allowed respectively.



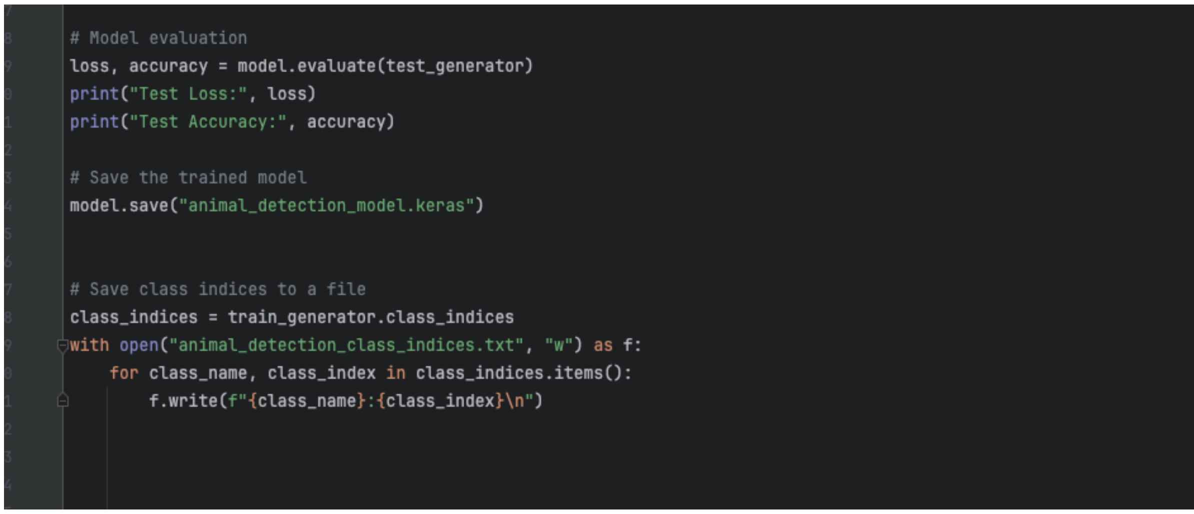
```
input_tensor = Input(shape=(224, 224, 3))
Updated_tensor = Conv2D( filters: 64,  kernel_size: (7, 7), strides=(2, 2), padding='same')(input_tensor)
Updated_tensor = BatchNormalization()(Updated_tensor)
Updated_tensor = Activation('relu')(Updated_tensor)
Updated_tensor = MaxPooling2D( pool_size: (3, 3), strides=(2, 2), padding='same')(Updated_tensor)

# Increasing depth of usting the filters
Updated_tensor = convolutional_block(Updated_tensor, filters=[64, 64, 256], kernel_size=(3, 3), strides=(1, 1))
Updated_tensor = identity_block(Updated_tensor, filters=[64, 64, 256], kernel_size=(3, 3))
Updated_tensor = identity_block(Updated_tensor, filters=[64, 64, 256], kernel_size=(3, 3))

Updated_tensor = convolutional_block(Updated_tensor, filters=[128, 128, 512], kernel_size=(3, 3))
Updated_tensor = identity_block(Updated_tensor, filters=[128, 128, 512], kernel_size=(3, 3))
Updated_tensor = identity_block(Updated_tensor, filters=[128, 128, 512], kernel_size=(3, 3))
Updated_tensor = identity_block(Updated_tensor, filters=[128, 128, 512], kernel_size=(3, 3))

# Adding more layers
Updated_tensor = convolutional_block(Updated_tensor, filters=[256, 256, 1024], kernel_size=(3, 3))
Updated_tensor = identity_block(Updated_tensor, filters=[256, 256, 1024], kernel_size=(3, 3))
Updated_tensor = identity_block(Updated_tensor, filters=[256, 256, 1024], kernel_size=(3, 3))
Updated_tensor = identity_block(Updated_tensor, filters=[256, 256, 1024], kernel_size=(3, 3))

Updated_tensor = GlobalAveragePooling2D()(Updated_tensor)
output_tensor = Dense(len(train_generator.class_indices), activation='softmax')(Updated_tensor)

model = Model( *args: input_tensor, output_tensor)


# Adding callback to monitor val_loss
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=3, min_lr=1e-6)

# Compile the model
model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model with callbacks
model.fit(train_generator, epochs=25, validation_data=test_generator, callbacks=[reduce_lr])
```

D. Evaluation

The model.evaluate() function will evaluate the performance of the model based on the test images provided by the ‘test_generator’. The generator provides batches of test images along with the labels that correspond to those images. It will return two values- ‘losses and 'accuracy', where ‘losses represent the errors on the testing datasets and ‘accuracy’ represents the proportion of correctly classified images out the total images in the testing datasets.

The model. Save () function is used to save the entire architecture, weights, and optimizer state of the trained model in a single file so that we can reuse it without having to retrain it from scratch. This is saved in a HDF5 file format commonly used in machine learning communities for storing large amounts of data.

‘train_generator.class_indices’ contains a dictionary mapping the class names to their corresponding integer indices. Next it will create and open a file named ‘model2_50epochs.txt’ in write mode and referred to as ‘f’. Then the loop will iterate over each item in the class_name dictionary and here ‘class_name’ represents the class name(such-’cat’,’dog’) and class_index represent the corresponding integer index assigned to that class. Next there will be a string which will contain the class name followed by a colon then the corresponding index.



```
# Model evaluation
loss, accuracy = model.evaluate(test_generator)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)

# Save the trained model
model.save("animal_detection_model.keras")


# Save class indices to a file
class_indices = train_generator.class_indices
with open("animal_detection_class_indices.txt", "w") as f:
    for class_name, class_index in class_indices.items():
        f.write(f"{class_name}:{class_index}\n")
```

E. Ethical Considerations

With any AI systems, there are some ethical considerations that we must address ourselves during our development. Here are some of the ethical considerations that we considered and will continue to focus on this project future scope.

- Wildlife privacy:

Wildlife has their own privacy to live in their natural habitat and roam freely. We will try not to disturb this natural habitat through this project by implementing rule on not capturing pictures and videos that may disturb their privacy.

- Impacts to wildlife behavior:

Continuous monitoring of wildlife could alter their natural living. We must mitigate this issue.

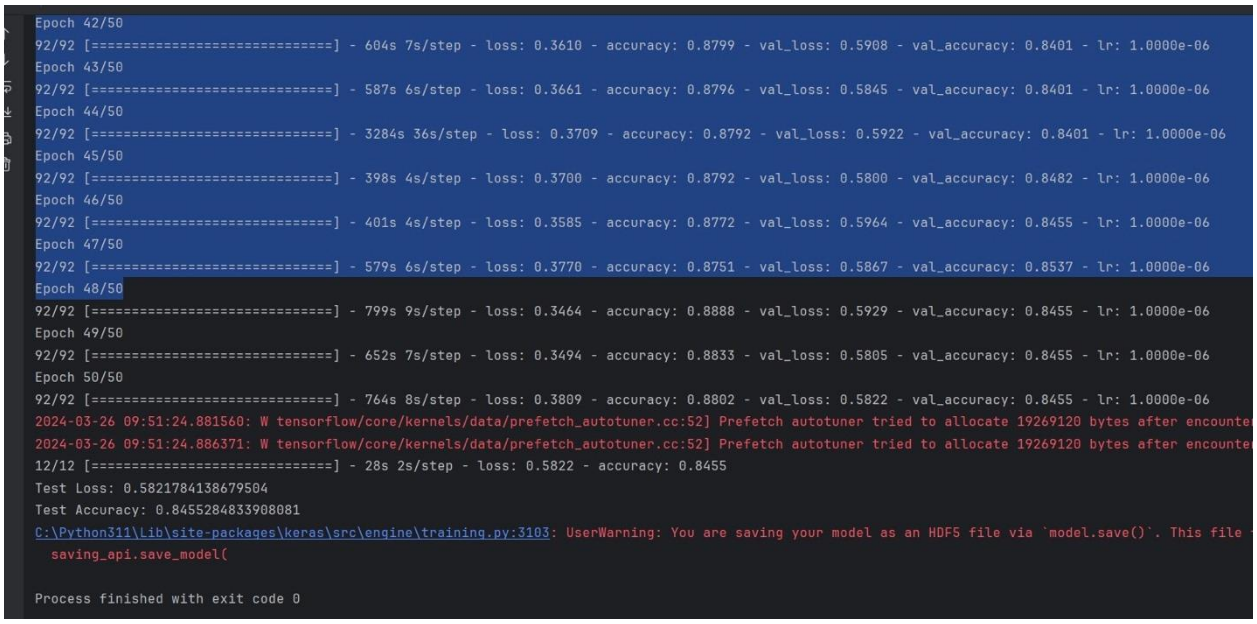
- Data security and privacy:

Collected data about animals and their presence must be secured to avoid unintended harm to the animals.

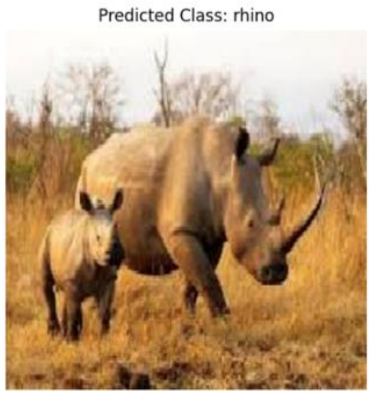
- Alert consequences that are unintended may affect the driver and their driving resulting human and wildlife harm.

IV. Result and Analysis

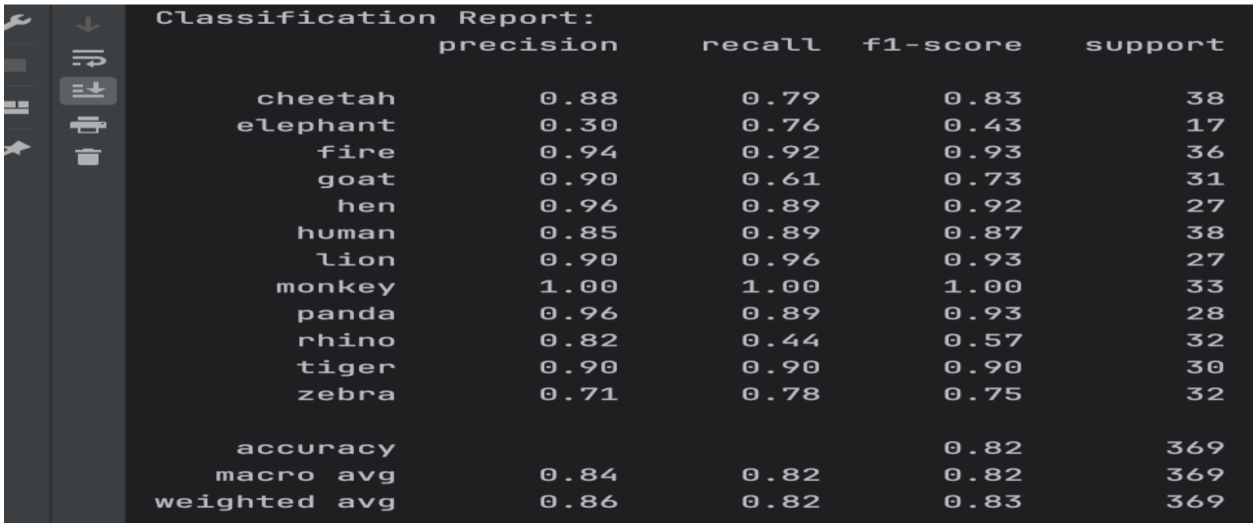
We evaluated our model to a set of 500 test images, and after running 50 epochs, on the training dataset we had minimal loss of 0.38 and accuracy of 0.88. Similarly on the test dataset, we had loss of 0.58 with the accuracy of 0.845. The to show the test result with single images , we initially we tested our model on input images. We provide image to the model and tried to see the predicted classes manually. Mostly the classification prediction was seen correct.



```
Epoch 42/50
92/92 [==============================] - 604s 7s/step - loss: 0.3610 - accuracy: 0.8799 - val_loss: 0.5908 - val_accuracy: 0.8401 - lr: 1.0000e-06
Epoch 43/50
92/92 [==============================] - 587s 6s/step - loss: 0.3661 - accuracy: 0.8796 - val_loss: 0.5845 - val_accuracy: 0.8401 - lr: 1.0000e-06
Epoch 44/50
92/92 [==============================] - 3284s 36s/step - loss: 0.3709 - accuracy: 0.8792 - val_loss: 0.5922 - val_accuracy: 0.8401 - lr: 1.0000e-06
Epoch 45/50
92/92 [==============================] - 398s 4s/step - loss: 0.3700 - accuracy: 0.8792 - val_loss: 0.5800 - val_accuracy: 0.8482 - lr: 1.0000e-06
Epoch 46/50
92/92 [==============================] - 401s 4s/step - loss: 0.3585 - accuracy: 0.8772 - val_loss: 0.5964 - val_accuracy: 0.8455 - lr: 1.0000e-06
Epoch 47/50
92/92 [==============================] - 579s 6s/step - loss: 0.3770 - accuracy: 0.8751 - val_loss: 0.5867 - val_accuracy: 0.8537 - lr: 1.0000e-06
Epoch 48/50
92/92 [==============================] - 799s 9s/step - loss: 0.3464 - accuracy: 0.8888 - val_loss: 0.5929 - val_accuracy: 0.8455 - lr: 1.0000e-06
Epoch 49/50
92/92 [==============================] - 652s 7s/step - loss: 0.3494 - accuracy: 0.8833 - val_loss: 0.5805 - val_accuracy: 0.8455 - lr: 1.0000e-06
Epoch 50/50
92/92 [==============================] - 764s 8s/step - loss: 0.3809 - accuracy: 0.8802 - val_loss: 0.5822 - val_accuracy: 0.8455 - lr: 1.0000e-06
2024-03-26 09:51:24.881560: W tensorflow/core/kernels/data/prefetch_autotuner.cc:52] Prefetch autotuner tried to allocate 19269120 bytes after encounte
2024-03-26 09:51:24.886371: W tensorflow/core/kernels/data/prefetch_autotuner.cc:52] Prefetch autotuner tried to allocate 19269120 bytes after encounte
12/12 [==============================] - 28s 2s/step - loss: 0.5822 - accuracy: 0.8455
Test Loss: 0.5821784138679504
Test Accuracy: 0.8455284833908081
C:\Python311\Lib\site-packages\keras\src\engine\training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file
  saving_api.save_model(

Process finished with exit code 0
```

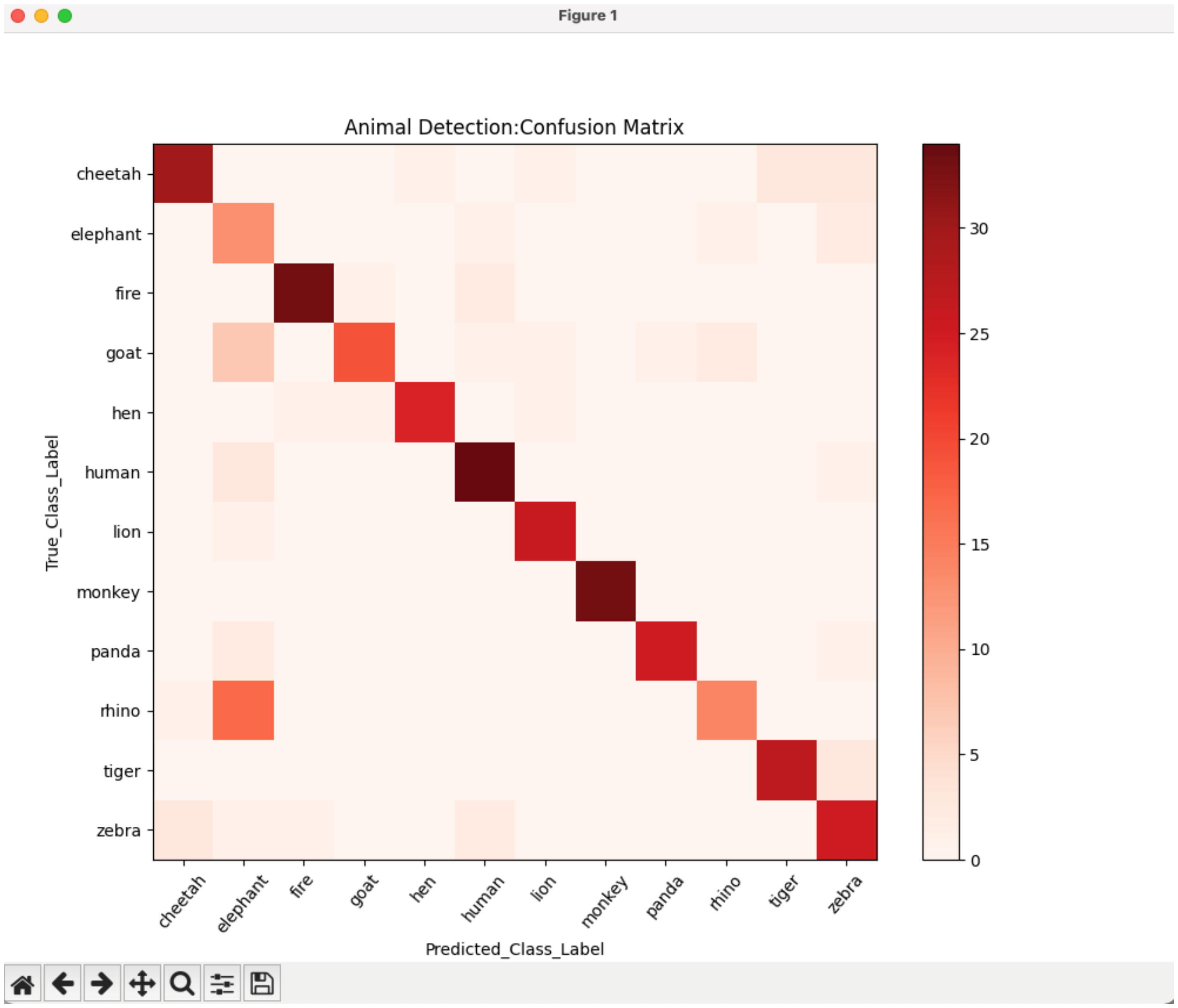


To evaluate our model, we printed the classification report on the test images as well and result is satisfactory as most of the result is as desired as shown below.



```
Classification Report:
              precision    recall  f1-score   support

     cheetah       0.88      0.79      0.83        38
    elephant       0.30      0.76      0.43        17
        fire       0.94      0.92      0.93        36
        goat       0.90      0.61      0.73        31
         hen       0.96      0.89      0.92        27
       human       0.85      0.89      0.87        38
        lion       0.90      0.96      0.93        27
      monkey       1.00      1.00      1.00        33
       panda       0.96      0.89      0.93        28
       rhino       0.82      0.44      0.57        32
       tiger       0.90      0.90      0.90        30
       zebra       0.71      0.78      0.75        32

    accuracy                           0.82       369
   macro avg       0.84      0.82      0.82       369
weighted avg       0.86      0.82      0.83       369
```

Then we created a confusion matrix to evaluate our model further. The result is as follows.



A. Conclusion

The project involves creating a wildlife security system using deep learning for image classification. We collected and preprocessed a dataset of wildlife images, trained a ResNet Model, and evaluated its performance. Currently, we have a trained model capable of classifying wildlife images with reasonable accuracy. Moving forward, we aim to enhance the model's accuracy and deploy the system in real-world scenarios for monitoring and protecting wildlife populations. Our goal is to contribute to wildlife conservation efforts by providing an effective and scalable solution for wildlife habitat monitoring and security.

B. Scope of Future Work

We could not provide the final completion to our project as our initial expectation due to time constraint. In future we want to achieve following goals.

- Increase the model accuracy with wide range of added animal categories.
- Increase the video effectiveness to capture wide and accurate frames.
- After getting preciseness on the animal detected, develop an warning system to warn the drivers with the detected animal
- Implement the project in real life.

C. References

Vaishnavi, K., et al. "Real-time Object Detection Using Deep Learning." *Journal of Advances in Mathematics and Computer Science* 38.8 (2023): 24-32.

Zhang, Mingyu, et al. "Real-Time Target Detection System for Animals Based on Self-Attention Improvement and Feature Extraction Optimization." Applied Sciences 13.6 (2023): 3987.

Tivelius, Malcolm. "Real-time Small Object Detection using Deep Neural Networks." (2021).

ImageNet, image-net.org/. Accessed 21 Jan. 2024.

"Animal Detect Object Detection Dataset and Pre-Trained Model by Artemis." Roboflow, universe.roboflow.com/artemis/animal-detect. Accessed 21 Jan. 2024.

Larxel. "Animal Faces." Kaggle, 22 May 2020, www.kaggle.com/datasets/andrewmvd/animal-faces.