أونيۏرسيتي تيكنولوجي مارا
UNIVERSITI TEKNOLOGI MARA

GROUP PROJECT

CSC583 : ARTIFICIAL INTELLIGENCE ALGORITHMS

NO	STUDENT NAME	STUDENT ID	GROUP
1	NUR AFRINA BINTI ZAINUDI	2023168623	CDCS2303C
2	NAZIRA KHAIRUNNISA BINTI JAMALUDIN	2023115371	
3	NUR IZZAH MAISARAH BINTI ROSTAM	2023149775	
4	EZATI WARDINA BINTI MURDI	2022463974	

SUBMISSION DATE: 16 JULY 2023

LECTURER'S NAME : MISS NORZILAH BINTI MUSA

TABLE OF CONTENTS

1. DESCRIPTION OF PROBLEM AND OBJECTIVES

1.1. Problem

Animal Classification is needed because there are so many animals in the world, it would be nearly impossible to keep track of them all if they are not organized into groups. Animal classification is used to group and classify various types of animals according to their shared traits and evolutionary connections. In the case for this project, we choose the group of animals that includes wolves, lions, tigers, bears, cats, dogs, elephants, goats, and horses. Understanding the links and variety within this group of mammals is made possible by the classification of animals.

The system's purpose is developing a computer-based system that can automatically analyze and classify images of these mammal species based on their visual characteristics. Let's take a look at the situation now and the wildlife conservation group in charge of observing and researching various mammal species in a certain area. The organization gathers a lot of pictures from various projects like citizen science, animal surveys, and camera traps. The difficulty they encountered was the laborious and time-consuming operation of manually classifying and sorting these photos by species.

By using animal classification of CNN images, wildlife conservation organizations can develop models that learn from datasets of labeled images. The model goes through training to learn to recognise patterns and visual cues unique to each mammal species. Once trained, the model can automatically analyze fresh photos and classify them according to their species.The approach saves conservationists a large amount of time and effort, allowing them to devote more resources to other critical tasks such as data analysis, habitat monitoring, and conservation planning. It improves their ability to process and manage vast numbers of wildlife photographs efficiently.

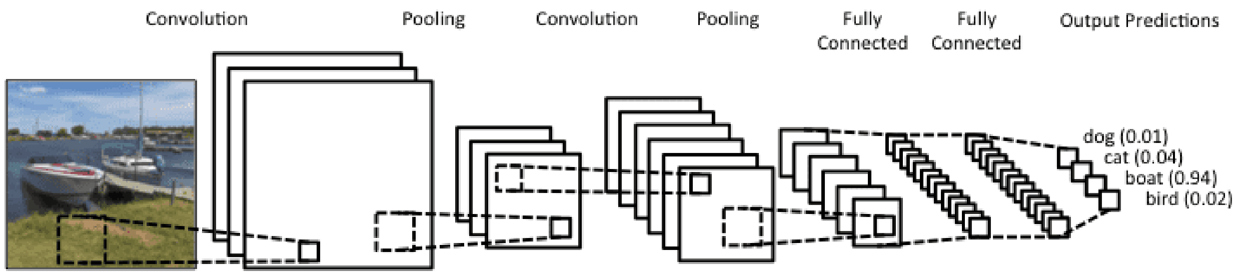
Animal classification systems can also be used by educational institutions and instructors to improve teaching and learning experiences. It is an excellent tool for exposing pupils to different mammal species through interaction with the system and comprehending the visual clues that separate them.

In summary, the purpose of using image animal classification for mammals in the class of bear, cat, dog, elephant, goat, horse, lion, tiger, and wolf is to automate the process of analyzing and categorizing images of these species. The approach saves time, improves accuracy, promotes public participation, and enhances educational experiences for animal conservation organizations, researchers, citizen scientists, educators, and the general public.

1.2. Objectives

1. To identify animals in each category for classification.
2. To identify the name of animals regarding the photo that we input.
3. To apply Convolutional Neural Network (CNN) in real life problems.

2. TECHNIQUE USED



For animal classification, we employ the CNN (Convolutional Neural Network) approach since it is particularly well-suited to picture analysis applications. CNNs are deep learning algorithms that recognise and extract patterns and features from images. This program will prompt the user to input an image and this Deep Learning Algorithm will assign importance (learnable weights and biases) to various aspects or objects in the image and be able to differentiate one from the other.CNNs are intended to learn and extract relevant features from images automatically. For example,they can recognise patterns, shapes, textures, and other visual traits that help distinguish between animal species.

CNNs, as opposed to classic neural networks, take advantage of the local connection pattern found in images. A convolutional layer's neurons process a tiny local portion of the input image, allowing the network to detect spatial correlations between nearby pixels. Furthermore, parameter sharing is a powerful feature of CNNs. Using the same set of weights for multiple positions in the input image allows the network to learn to detect a certain feature regardless of its position, improving the model's efficiency and generalization.

Pooling fully connected layers in convolutional neural networks play a crucial role in reducing the spatial dimensions of the feature maps while retaining the most important information. They achieve this by down-sampling the feature maps using operations such as max or average pooling. Pooling helps to make the model more robust to variations in the input data, increases its translation invariance, and reduces the computational complexity of subsequent layers.

Convolutional Neural Networks have revolutionized the field of computer vision and deep learning. Their ability to automatically learn and extract meaningful features from images and utilize local connectivity, parameter sharing, and hierarchical learning makes them highly effective in various tasks like image classification, object detection, and image recognition.

The inclusion of fully connected layers at the end of the network facilitates accurate classification by mapping the learned features to specific class labels. With their wide-ranging benefits, CNN in deep learning continues to pave the way for advancements in computer vision and contribute to developing AI systems that can understand and interpret visual data with remarkable precision.CNN represents a huge breakthrough in image recognition. They're most commonly used to analyze visual imagery and are frequently working behind the scenes in image classification. Image classification is the process of taking an input (like a picture) and outputting a class (like "cat") or a probability that the input is a particular class ("there's a 0.9% accuracy that this input is a cat").

3. INPUT, PROCESS AND OUTPUT

3.1. Input

For our project, we have used 9 classes of mammals namely bears, cats, dogs, elephants, goats, horses, lions, tigers and wolves. We use 99 bear datasets, 99 datasets cat, 99 dog dataset, 99 elephant dataset, 99 elephant dataset, 99 goat dataset, , 99 horse dataset, 99 lion dataset, 99 tiger dataset and 99 dataset wolf which makes a total of 891 data sets. The dataset that we use in this system is an existing dataset that we have downloaded from the kaggle website and adapted the coding so that it can be used and read by this system. **Figure 3.1.1** shows the data set we split according to the class of mammals it represents. We have divided these datasets into two sets according to their respective needs without duplicating the datasets. 712 which is 80% of datasets has been used for training purposes while the other 179 which is 20% datasets has been used for testing. The reason we are doing so is to prevent overfitting in machine learning. Due to this scenario, the machine's ability to generalize data will decline. **Figure 3.1..2**
shows the datasets of our project.

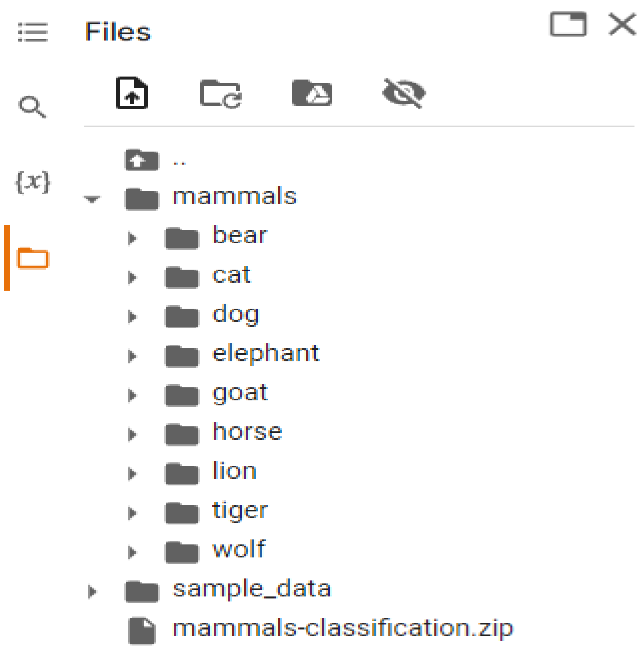
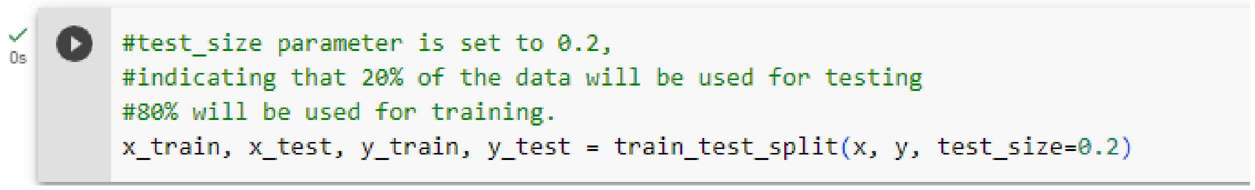


Figure 3.1.1 The datasets according to mammal classes.



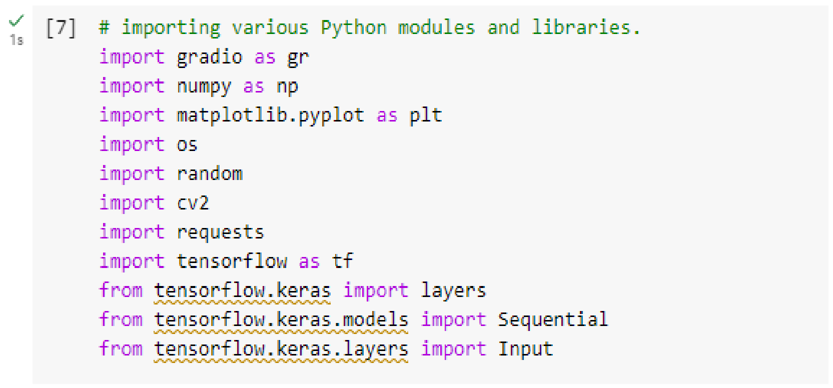
```
#test_size parameter is set to 0.2,
#indicating that 20% of the data will be used for testing
#80% will be used for training.
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

Figure 3.1.2. The datasets.

3.2. Process

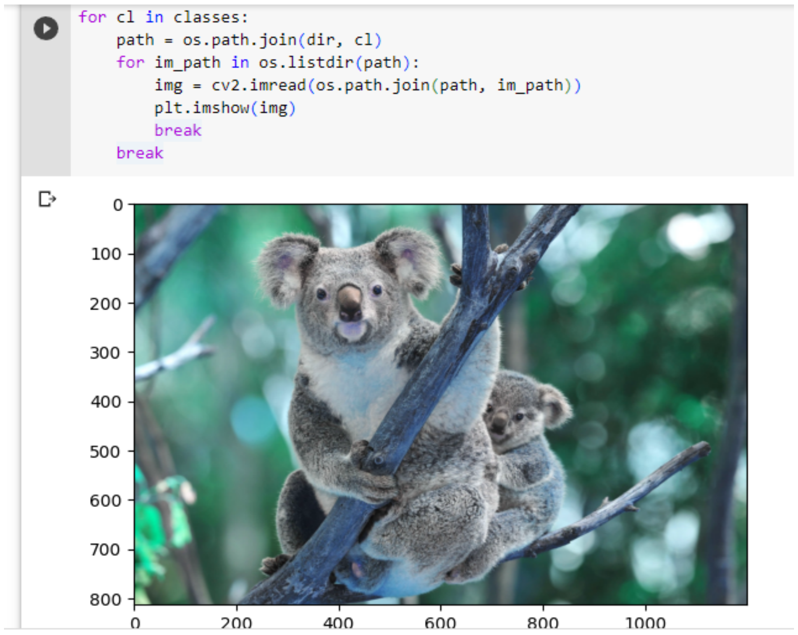
For our project we use Colab Notebooks to run the source code. After we have collected our datasets, we process the image by using Convolutional Neural Network (CNN) technique.y. Here we will briefly state what are the code segments all about. Any significant segment will be explained succinctly. But first, we need to be reminded that, before the learning process, there are a few steps we need to complete.

1. Load a Python script that imports various libraries and modules commonly used for working with data, images, and machine learning models.



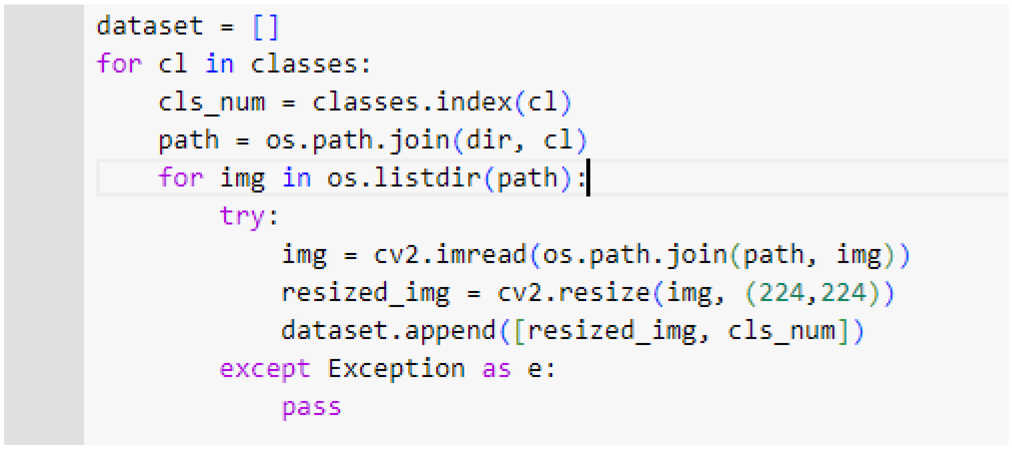
```
[7] # importing various Python modules and libraries.
    import gradio as gr
    import numpy as np
    import matplotlib.pyplot as plt
    import os
    import random
    import cv2
    import requests
    import tensorflow as tf
    from tensorflow.keras import layers
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Input
```

2. Iterates over a list of classes, constructs a file path for each class, reads an image from the corresponding path, and displays the image using Matplotlib.



```
for cl in classes:
    path = os.path.join(dir, cl)
    for im_path in os.listdir(path):
        img = cv2.imread(os.path.join(path, im_path))
        plt.imshow(img)
        break
    break
```

3. Creates a dataset by processing images from different classes using OpenCV. It loops through the classes, reads and resizes the images, and adds them to a dataset with their class labels. Errors are ignored, allowing the code to continue. The resulting dataset contains resized images and class labels.



```
dataset = []
for cl in classes:
    cls_num = classes.index(cl)
    path = os.path.join(dir, cl)
    for img in os.listdir(path):
        try:
            img = cv2.imread(os.path.join(path, img))
            resized_img = cv2.resize(img, (224,224))
            dataset.append([resized_img, cls_num])
        except Exception as e:
            pass
```

4. Edit the code as a function that processes a dataset and separates the images

and corresponding labels into separate lists. It is useful when preparing data for training a machine learning model.

```
x =  []
y = []
for pic, label in dataset:
    x.append(pic)
    y.append(label)
```

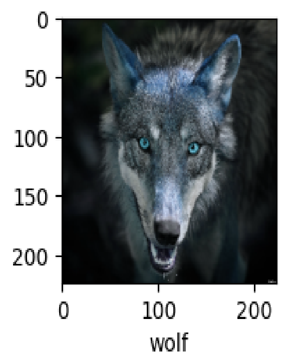
5. Splits the input features (x) and corresponding labels (y) into training and testing datasets using the train_test_split function. The resulting datasets are assigned to x_train, x_test, y_train, and y_test, which can be used for model training and evaluation.

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

6. Display an image along with its label using Matplotlib. Specifies the size of the figure to be created and parameter sets the font size of the label to 10. It creates a figure, displays the image at the specified index, and sets the x-axis label to the corresponding label for that image.

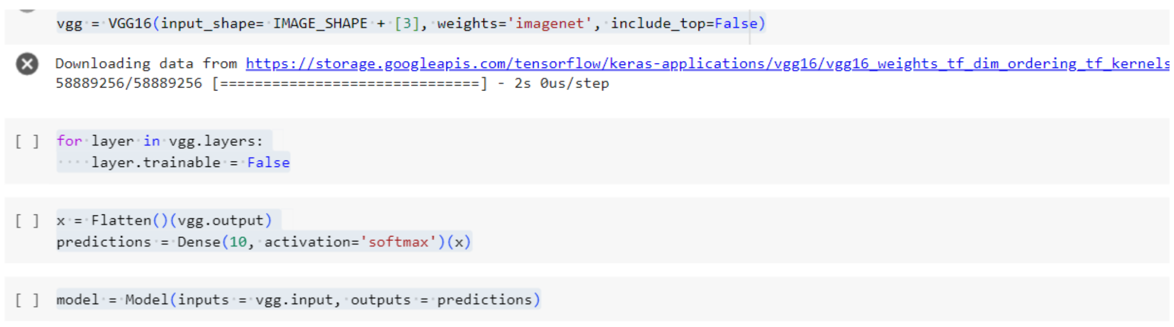
```
def img_show(pic, label, index):
    plt.figure(figsize=(15,2))
    plt.imshow(pic[index])
    plt.xlabel(translate[label[index]], fontsize=10)
```

```
[ ] img_show(x_test, y_test, 8)
```



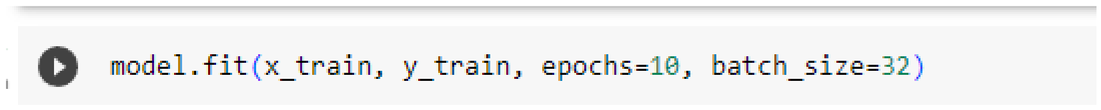
7. Initializes a VGG16 model with pre-trained weights and excludes the fully

connected layers. Adds new layers on top of the pre-trained model, including a Flatten layer and a Dense layer for classification. The resulting model is stored in the model variable and can be used for training or making predictions.



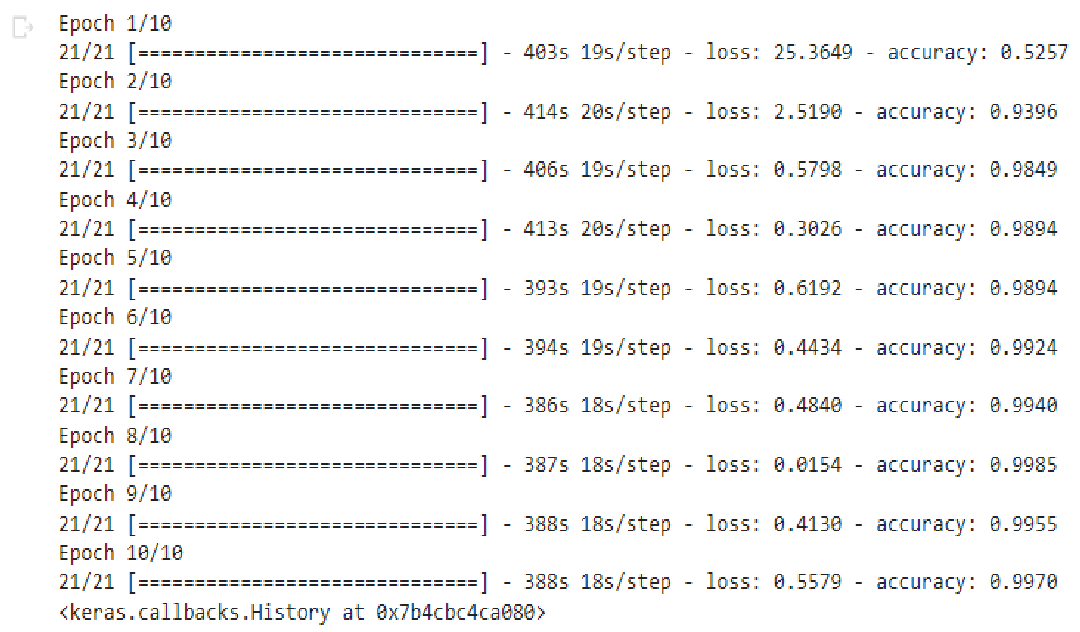
```
vgg = VGG16(input_shape= IMAGE_SHAPE + [3], weights='imagenet', include_top=False)

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels
58889256/58889256 [==============================] - 2s 0us/step

[ ] for layer in vgg.layers:
        layer.trainable = False

[ ] x = Flatten()(vgg.output)
    predictions = Dense(10, activation='softmax')(x)

[ ] model = Model(inputs = vgg.input, outputs = predictions)
```

8. Train the model and calculate its accuracy on testing data. Finally, this code segment is where the training process could be done.



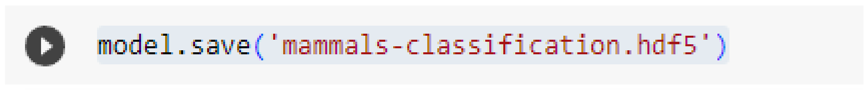
```
model.fit(x_train, y_train, epochs=10, batch_size=32)
```

When this segment is executed, it will show the training and learning process. This training process will take some time, as there are so many images in the dataset loaded. It took us nearly 1 hour to complete the training. From this terminal message, we can clearly see the accuracy of each epoch calculated from the code.



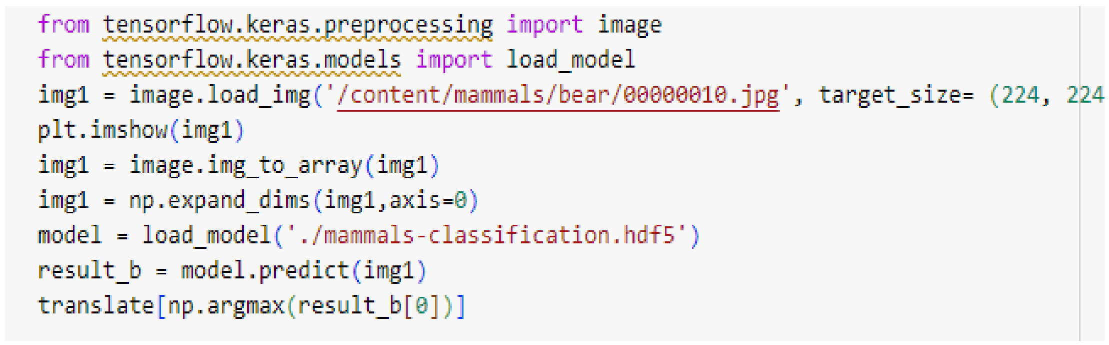
```
Epoch 1/10
21/21 [==============================] - 403s 19s/step - loss: 25.3649 - accuracy: 0.5257
Epoch 2/10
21/21 [==============================] - 414s 20s/step - loss: 2.5190 - accuracy: 0.9396
Epoch 3/10
21/21 [==============================] - 406s 19s/step - loss: 0.5798 - accuracy: 0.9849
Epoch 4/10
21/21 [==============================] - 413s 20s/step - loss: 0.3026 - accuracy: 0.9894
Epoch 5/10
21/21 [==============================] - 393s 19s/step - loss: 0.6192 - accuracy: 0.9894
Epoch 6/10
21/21 [==============================] - 394s 19s/step - loss: 0.4434 - accuracy: 0.9924
Epoch 7/10
21/21 [==============================] - 386s 18s/step - loss: 0.4840 - accuracy: 0.9940
Epoch 8/10
21/21 [==============================] - 387s 18s/step - loss: 0.0154 - accuracy: 0.9985
Epoch 9/10
21/21 [==============================] - 388s 18s/step - loss: 0.4130 - accuracy: 0.9955
Epoch 10/10
21/21 [==============================] - 388s 18s/step - loss: 0.5579 - accuracy: 0.9970
<keras.callbacks.History at 0x7b4cbc4ca080>
```

9. Saved the trained neural network model and make a dictionary to map to the output classes.

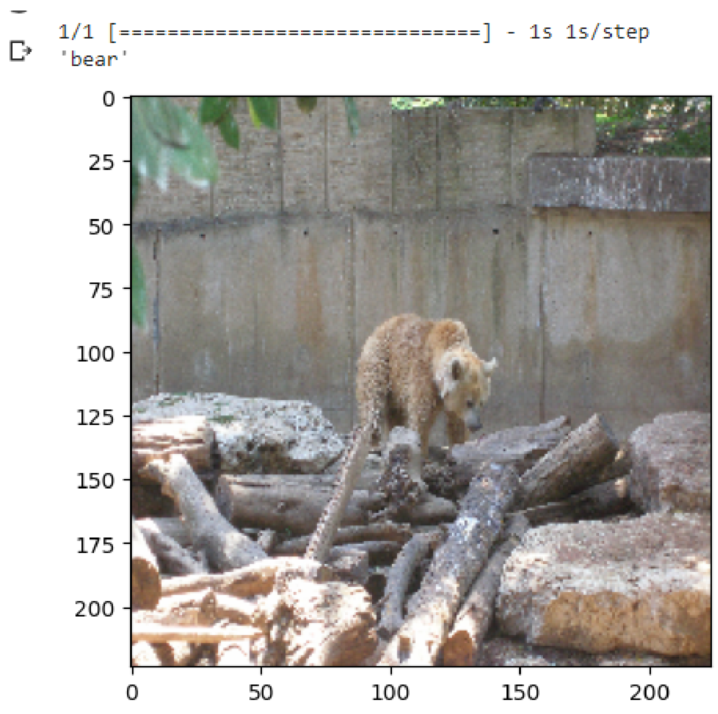


```
model.save('mammals-classification.hdf5')
```

This is a function for loads a trained model and making predictions on a specific image. It loads a trained model from a file, loads an image, makes predictions using the loaded model on the image, and retrieves the predicted class label based on the highest prediction probability.



```
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import load_model
img1 = image.load_img('/content/mammals/bear/00000010.jpg', target_size= (224, 224
plt.imshow(img1)
img1 = image.img_to_array(img1)
img1 = np.expand_dims(img1,axis=0)
model = load_model('./mammals-classification.hdf5')
result_b = model.predict(img1)
translate[np.argmax(result_b[0])]
```



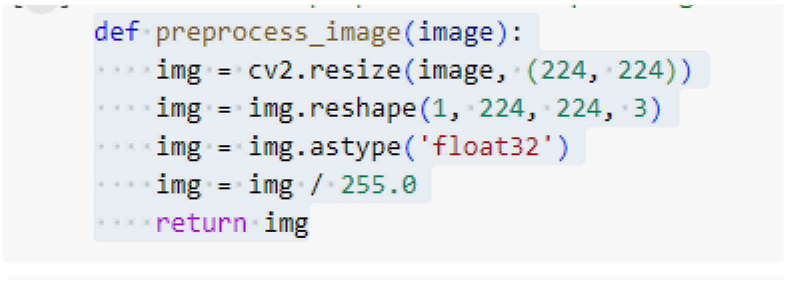
```
1/1 [==============================] - 1s 1s/step
'bear'
```

In summary, our program algorithm is mainly trained to learn the image and specify the objects based on the kaggle dataset that we've downloaded and imported into the google colab which consists of 891 images in 9 classes, with 99 images per class. There are 712 images for training and 179 images for the test.

3.3. Output

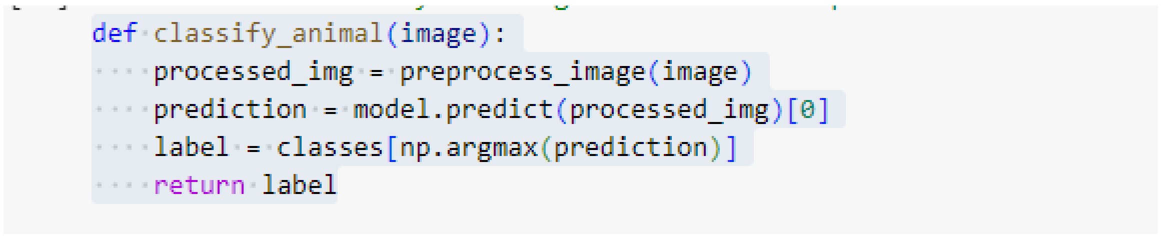
Here we attach the responsible coding segment program output using GUI functions. However there are some code segments that we need to run first to produce output through the GUI.

1. Defines a function named preprocess_image that preprocesses an input image before feeding it into a neural network model.



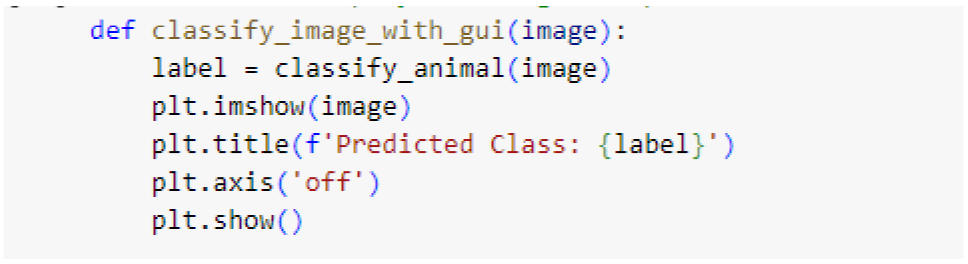
```
def preprocess_image(image):
    img = cv2.resize(image, (224, 224))
    img = img.reshape(1, 224, 224, 3)
    img = img.astype('float32')
    img = img / 255.0
    return img
```

2. Defines a function named classify_animal that takes an input image, preprocesses it, and uses a trained model to classify the animal in the image.



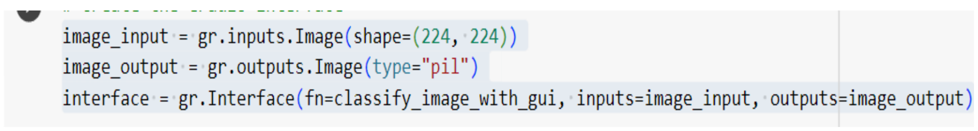
```
def classify_animal(image):
    processed_img = preprocess_image(image)
    prediction = model.predict(processed_img)[0]
    label = classes[np.argmax(prediction)]
    return label
```

3. Defines a function named classify_image_with_gui that takes an input image, classifies it using the classify_animal function, and displays the image along with the predicted class label using Matplotlib.



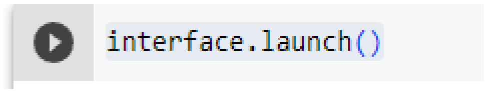
```
def classify_image_with_gui(image):
    label = classify_animal(image)
    plt.imshow(image)
    plt.title(f'Predicted Class: {label}')
    plt.axis('off')
    plt.show()
```

4. Creates an interface object using gr.Interface . It sets the input and output objects to handle image inputs and PIL image outputs, respectively. This allows users to interact with the GUI to provide images as input and visualize the output of the classification process.



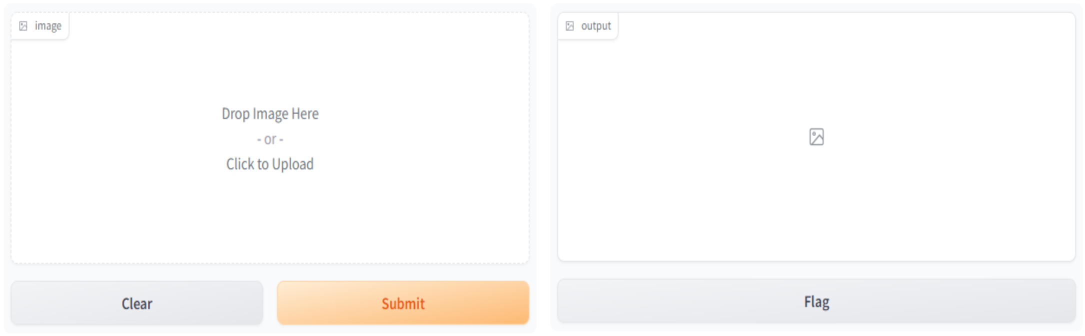
```
image_input = gr.inputs.Image(shape=(224, 224))
image_output = gr.outputs.Image(type="pil")
interface = gr.Interface(fn=classify_image_with_gui, inputs=image_input, outputs=image_output)
```

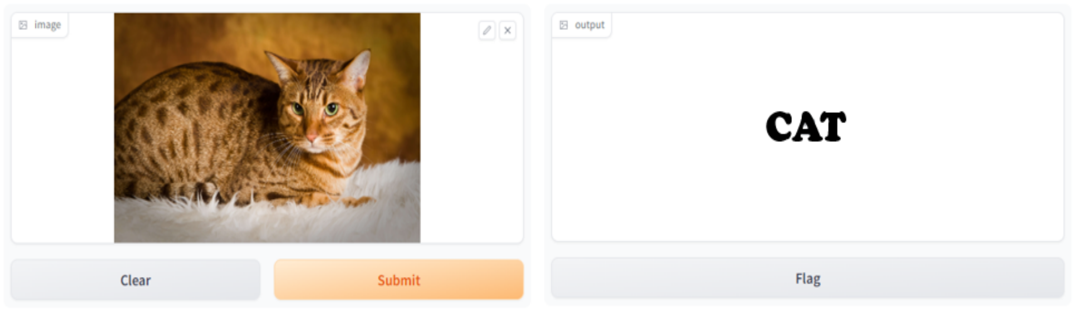
5. Launches the graphical user interface (GUI) that was defined using the gradio.



```
interface.launch()
```

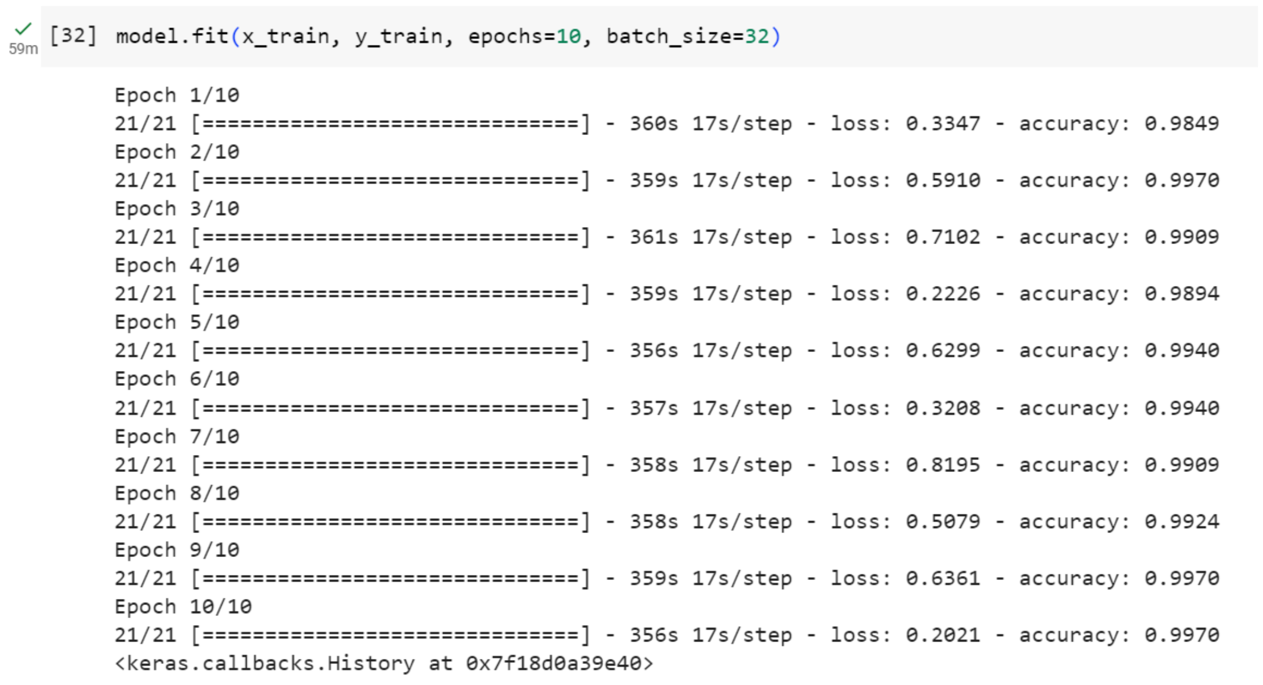
This is the output interface of the system, there are two separate boxes where one is the box to upload pictures and the other is where the name of the class mammal will be output.After the user uploads the selected image and clicks 'Submit', the program will display the possible answer for the picture selected. The output is simply the alphabetical name of the object detected inside the image.





4. EXPERIMENTAL RESULTS AND ANALYSIS

We have two files that require execution. The first file is intended for deep learning and training purposes, while the other file is responsible for the Graphic User Interface (GUI). Running the first program will display the training process, consisting of 10 epochs. Each epoch possesses its own accuracy value. The average accuracy for this program is calculated to be 0.9928. Although it may not be considered a significantly high accuracy, it remains within a manageable range.



```
[32] model.fit(x_train, y_train, epochs=10, batch_size=32)

     Epoch 1/10
     21/21 [==============================] - 360s 17s/step - loss: 0.3347 - accuracy: 0.9849
     Epoch 2/10
     21/21 [==============================] - 359s 17s/step - loss: 0.5910 - accuracy: 0.9970
     Epoch 3/10
     21/21 [==============================] - 361s 17s/step - loss: 0.7102 - accuracy: 0.9909
     Epoch 4/10
     21/21 [==============================] - 359s 17s/step - loss: 0.2226 - accuracy: 0.9894
     Epoch 5/10
     21/21 [==============================] - 356s 17s/step - loss: 0.6299 - accuracy: 0.9940
     Epoch 6/10
     21/21 [==============================] - 357s 17s/step - loss: 0.3208 - accuracy: 0.9940
     Epoch 7/10
     21/21 [==============================] - 358s 17s/step - loss: 0.8195 - accuracy: 0.9909
     Epoch 8/10
     21/21 [==============================] - 358s 17s/step - loss: 0.5079 - accuracy: 0.9924
     Epoch 9/10
     21/21 [==============================] - 359s 17s/step - loss: 0.6361 - accuracy: 0.9970
     Epoch 10/10
     21/21 [==============================] - 356s 17s/step - loss: 0.2021 - accuracy: 0.9970
     <keras.callbacks.History at 0x7f18d0a39e40>
```

After executing the program and conducting tests using various images, we encountered the expected occurrence of correct answers. However, we also obtained a few incorrect answers. Based on our experience with this algorithm, we have observed that the program struggles to accurately detect objects when an image contains multiple animals, leading to inaccurate results. For instance, the program sometimes misidentifies a bear as a lion or a bear as a cat. We believe that increasing the number of epochs can improve accuracy, but it would significantly increase the execution time since each epoch takes a while. Additionally, it's important to note that this program is only capable of detecting images within the predefined classes of the provided datasets.

5. CONCLUSION

In conclusion, Artificial Intelligence is a rapidly advancing field with immense potential and numerous applications across various industries. AI refers to the development of computer systems that can perform tasks that typically require human intelligence. Throughout this assignment, we have examined the capabilities of AI systems. These capabilities enable AI to process vast amounts of data, recognize patterns and make predictions of 9 classes of animal. The 9 classes are bear, cat, dog, elephant, goat, horse, lion, tiger and wolf. The input from the user will identify which category of the chosen images. It is either among the 9 classes. As this program succeeds in classifying images, then all the objectives of the system are achieved. However, in the aspect of data accuracy, it doesn't seem 100% accurate. When an image contains several animals, the program has trouble correctly identifying the animal, which causes inaccurate results.

6. FUTURE WORKS RECOMMENDATION

If we wish to use the same hardware to develop our projects, we must alter the prior number. Therefore, k-fold cross validation would be suitable as there are ways commonly used by other developers to obtain optimal levels and high levels of accuracy from training data sets at once. Different data sets should be used to test the method. We will be able to determine if the machine is trained properly or not. Moreover, producing hard data will result in the increase of the accuracy improvement. By producing hard data, we limit the images that have a high probability of being labeled incorrectly to be trained.Practise with all the photos to balance the accuracy of the exercise while checking the accuracy after practicing with the hard data and before returning to the usual activity. Because we don't have to wait for each cell in the source code to execute, this sneaky way can save time. Finally, we strongly advise using future advancements with hardware that is more powerful that supports higher folds.If the project is run on more advanced hardware, it is possible to quantify picture categorization effectiveness in other ways in addition to accuracy.

7. REFERENCES

Youtube

Goldbloom , A., Hamner, B. and Nicholas Gruen, N. (2017) *Find open datasets and Machine Learning Projects*, *Kaggle*.

https://www.kaggle.com/datasets

Kumar, Y. S., Manohar, N., & Chethan, H. K. (2015). Animal classification system: a block based approach. *Procedia Computer Science*, *45*, 336-343.

Manohar, N., Kumar, Y. S., Rani, R., & Kumar, G. H. (2019). Convolutional neural network with SVM for classification of animal images. In *Emerging Research in Electronics, Computer Science and Technology: Proceedings of International Conference, ICERECT 2018* (pp. 527-537). Springer Singapore.

8. APPENDIX

System : Google collab link:
https://colab.research.google.com/drive/1SqSM-5w4i3aYXZglgjDNC85h9amZB615?usp=sharing
Youtube Video Link:
https://youtu.be/yIG0CaeSOXA