Classification of Animals

To identify species of animals and birds in video clips as part of a wildlife and environmental monitoring project, i would apply the following steps:

1.Data Collection

Since i would need dataset to train the model on, i would start by collecting data, in this case images and videos captures by the cameras. I will have to make sure that those images and videos cover a large variety of animal and species in different environmental conditions(daytime, nighttime, different weather conditions, etc.).

2. Data Preprocessing

Before applying machine learning algorithms, I need to preprocess the video data:

a. Video to Frame Conversion: Convert the video clips into individual frames. Each frame will be treated as an individual image for analysis. To reduce data size, extract a frame every second because a video is typically 30/60 frames per second and if each frame is fed into the algorithm it would be a lot of data to process. I would use openCV to convert video to images. Also use openCV to identify motion and put rectangles around the animal and then pass that image for labelling. Example: This is just basic code to identify moving objects in a frame and it draws a rectangle around it.

```python
import cv2
import numpy as np

cap = cv2.VideoCapture(1)
cap.set(cv2.CAP_PROP_FPS, 0.25)
ret, frame1 = cap.read()
ret, frame3 = cap.read()
k = 0
while cap.isOpened():
    diff = cv2.absdiff(frame1, frame3)
    gray = cv2.cvtColor(diff,cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray,(5,5), 0)
    _, thresh = cv2.threshold(blur, 20, 255, cv2.THRESH_BINARY)
    dilated = cv2.dilate(thresh, None, iterations=3)
    contours, _ = cv2.findContours(dilated,cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    for contour in contours:
        (x, y, w, h) = cv2.boundingRect(contour)

        if cv2.contourArea(contour) < 12000:
            continue

        cv2.rectangle(frame1, (x,y), (x+w, y+h), (0,255,0), 2)

    #cv2.drawContours(frame1, contours, -1,(0,255,0), 2)
    #cv2.imwrite(f'frame_{k}.jpg', frame1)
    k += 1
    cv2.imshow('feed', frame1)
    frame1 = frame3
    ret, frame3 = cap.read()

    if cv2.waitKey(40) == 27:
        break
```

b. Image Enhancement: Enhance the quality of frames to improve feature extraction. Techniques like contrast adjustment, noise reduction, and image resizing can be applied.

c. Labelling: I would also need to label the data, if multiple species are present in the i would have to label the data accordingly. Labelling in this case would mean annotating the images/frames as it would help the Neural Network to accurately identify the species and would be crucial for multi-species identification.

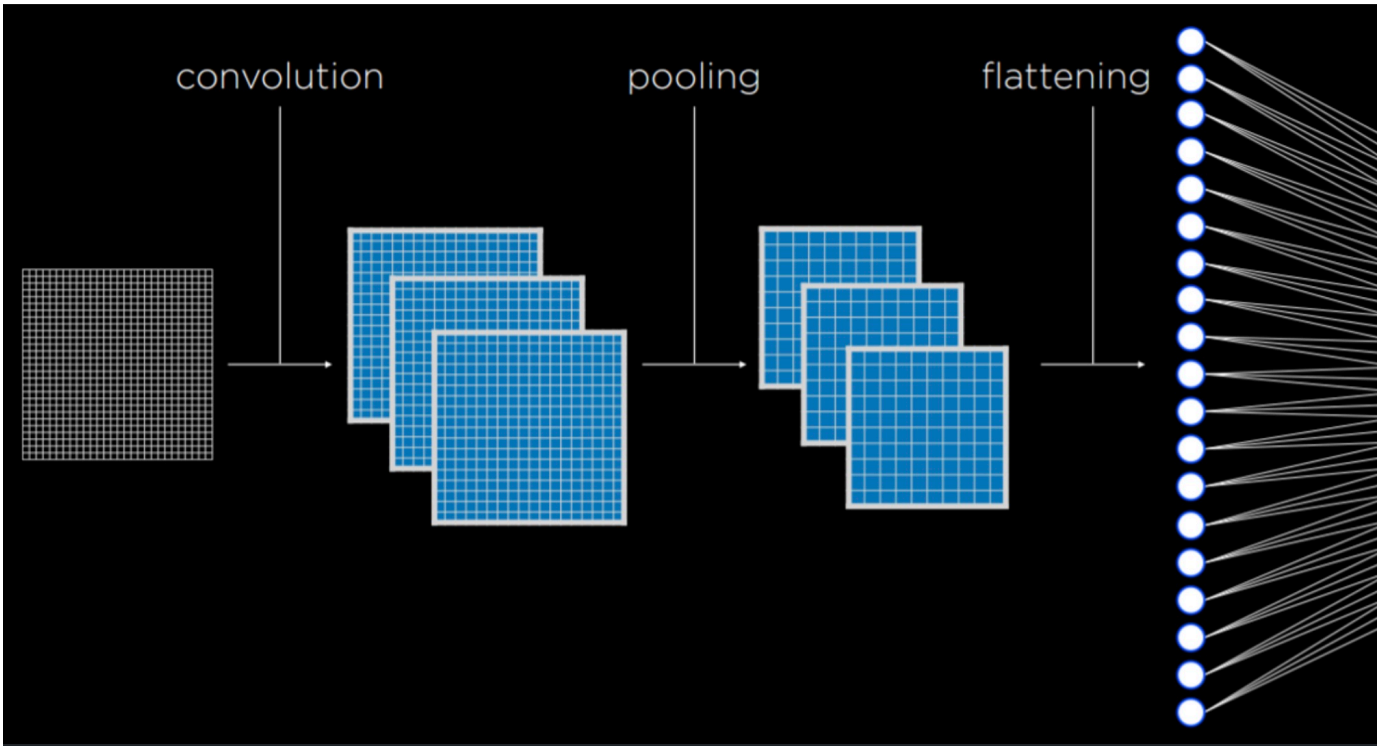
d. Data Augmentation: Since the amount of images to train on would be less, i would apply data augmentation to augment the dataset by applying transformations like rotation, cropping, and flipping to create variations of the images.

3. Training

a. Split the Data: After preprocessing I would divide the dataset into training, validation and test sets.

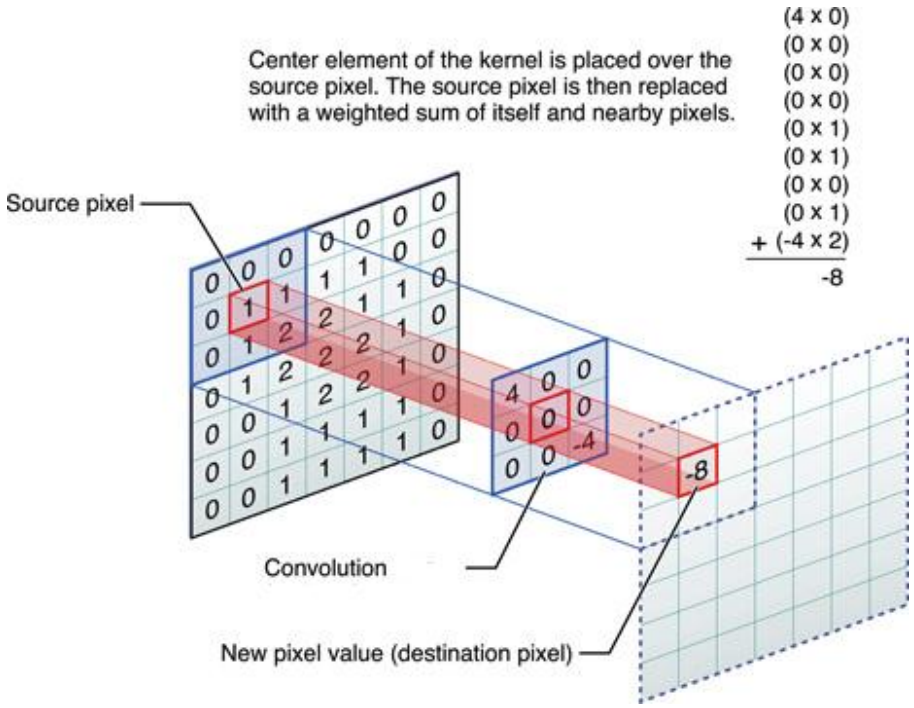
b. Model Architecture: Since I am training on images, a CNN architecture using Tensorflow/Keras. A suitable loss fucntion like categegorical cross-entropy for multi-class classification and optimizer(mostly Adam).

Example: A basic example of a CNN



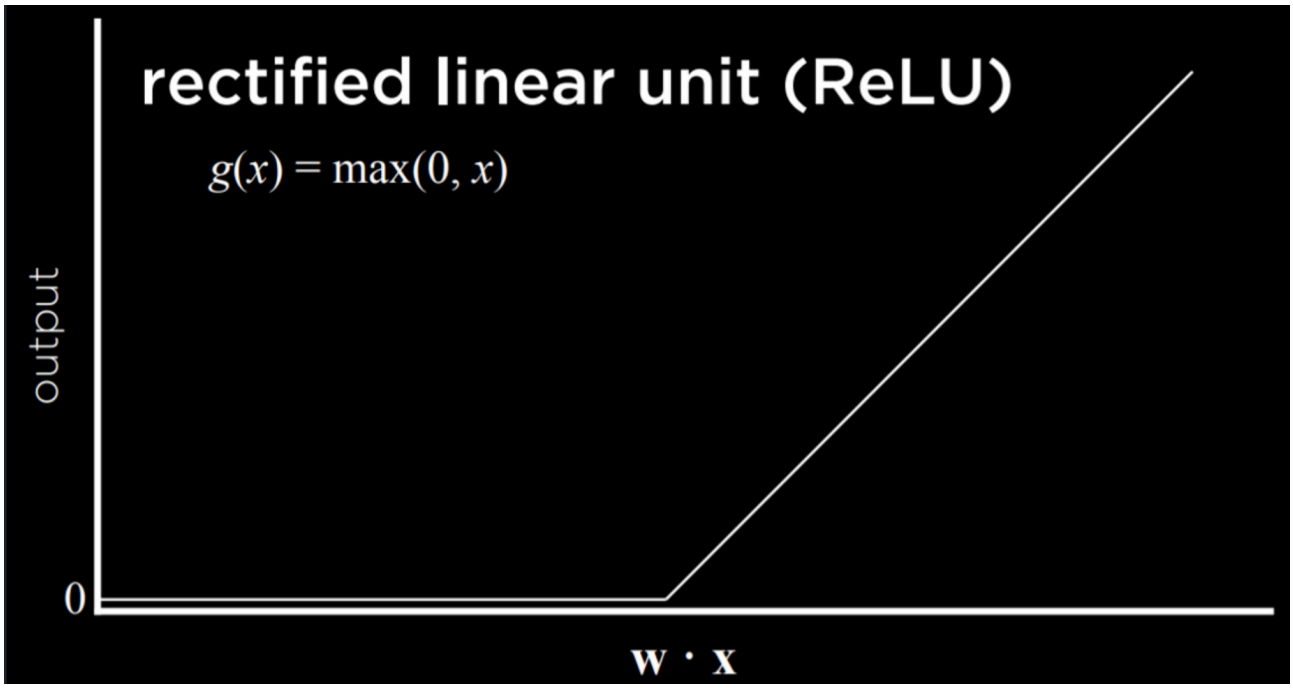
```
model = tf.keras.models.Sequential()

model.add(tf.keras.layers.Conv2D(32,(3,3),activation="relu", input_shape = (IMG_WIDTH, IMG_HEIGHT,3)))

model.add(tf.keras.layers.MaxPooling2D())

model.add(tf.keras.layers.Conv2D(32,(3,3),activation="relu", input_shape = (IMG_WIDTH, IMG_HEIGHT,3)))

model.add(tf.keras.layers.MaxPooling2D())

model.add(tf.keras.layers.Conv2D(64,(3,3),activation="relu", input_shape = (IMG_WIDTH, IMG_HEIGHT,3)))

model.add(tf.keras.layers.MaxPooling2D(pool_size=(3,3)))

model.add(tf.keras.layers.Flatten())

model.add(tf.keras.layers.Dense(100, input_shape = (IMG_WIDTH, IMG_HEIGHT, 3), activation = "relu"))

model.add(tf.keras.layers.Dropout(0.5))

model.add(tf.keras.layers.Dense(NUM_CATEGORIES, activation="softmax"))

model.compile(
    optimizer="adam",
    loss = "categorical_crossentropy",
    metrics=["accuracy"]
)
```

Image Convolution: Image convolution is applying a filter that adds each pixel value of an image to its neighbors, weighted according to a kernel matrix. Doing so alters the image and can help the neural network process it.

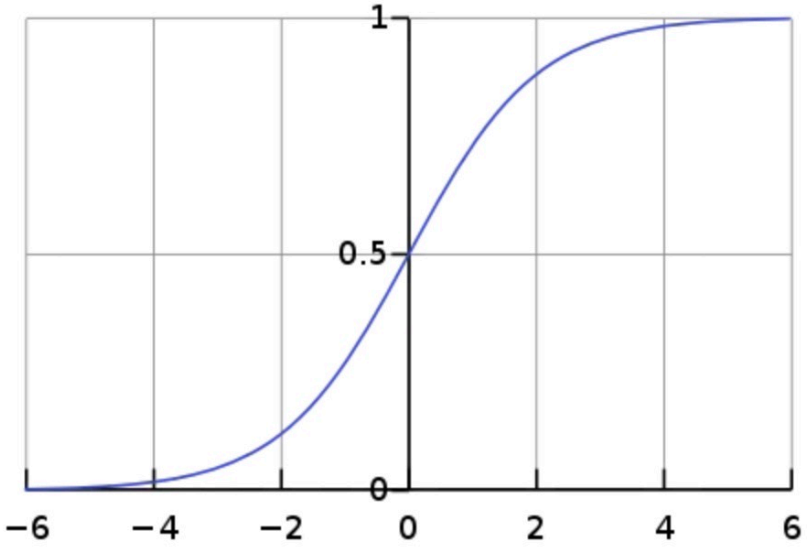


Activation Function: In this case I am using ReLU(rectified linear unit) as the activation function.
Here's how the ReLU activation function works:
For positive input values (x>0), ReLU returns the input value itself, essentially acting as the identity function. This means if the input is positive, it remains unchanged.

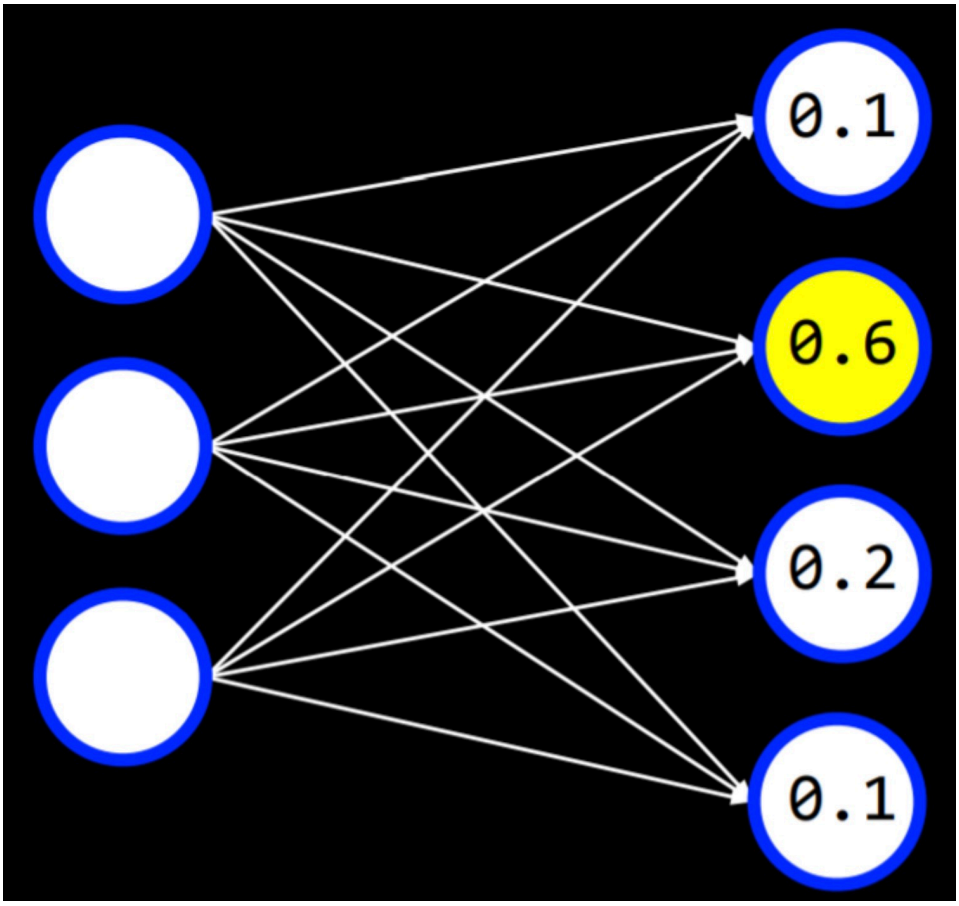
For negative input values (x≤0), ReLU returns 0. In other words, it "zeros out" any negative values.



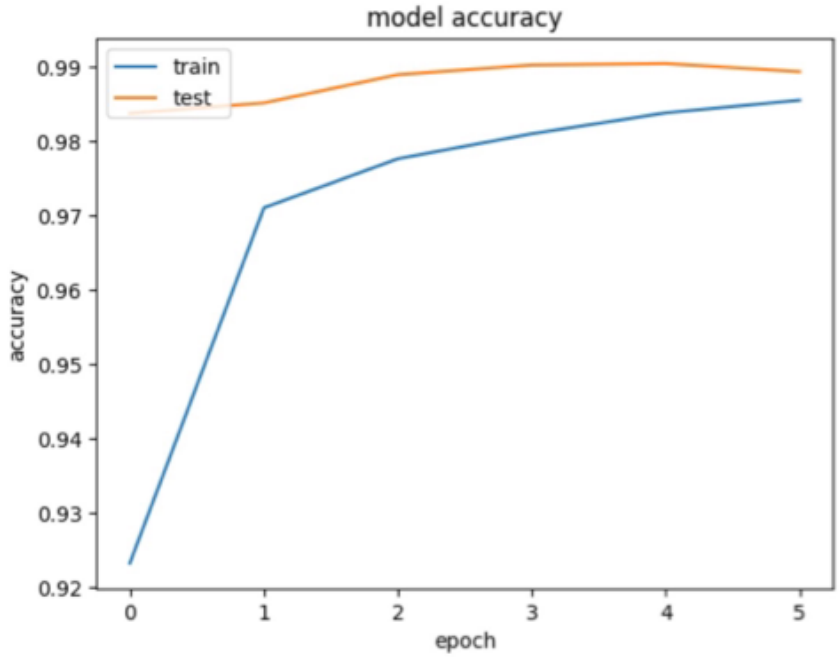
I the last layer I would use the Softmax activation function as it squashes its input values into the range [0,1], which is interpreted as probability. Last layer will return the probability of various species in the image.

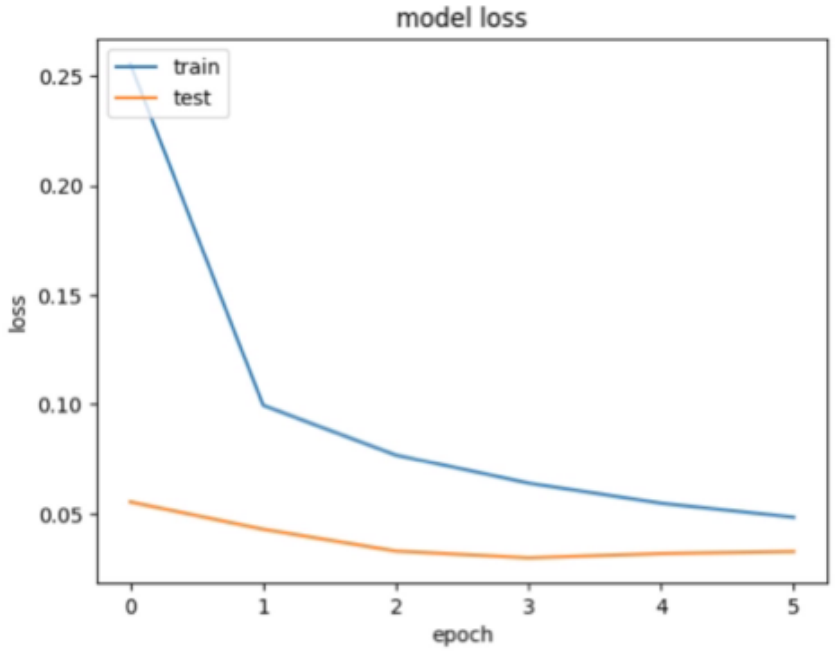


Example How output layer might look like after softmax activation:



c. Train the model on training training data and monitor its performance on the validation set, adjust the hyperparameters
based on performance of the model. Use matplotlib or tensorboard to check models performance.
Example: This is the model accuracy and loss after training the model on the MNIST dataset.





4. Evaluation

Evaluate the model's performance on the test set using metrics like accuracy, precision, recall, and F1-score.
I can further evaluate the model by passing new images to it and looking at its prediction. This will help us to fine tune the model with the help of new data.

Accuracy: It represents the number of correctly classified data instances over the total number of data instances.

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

Consider a model trained to classify between a lion and a cheetah, it was given 90 images of lion and 10 images of cheetah, it classified all 90 images as lions but also classified the images of cheetahs as lions. So the accuracy will be 90% but using accuracy in such scenarios will lead to misinterpretation of results.Since accuracy is not a good metric when the data set is unbalanced. That's why we calculate Precision, it is positive predicted value.

$$Precision = \frac{TP}{TP + FP}$$

Precision should ideally be 1 (high) for a good classifier. Precision becomes 1 only when the numerator and denominator are equal i.e TP = TP +FP, this also means FP is zero. As FP increases the value of denominator becomes greater than the numerator and precision value decreases

Recall is also known as sensitivity or true positive rate and is defined as follows:

$$Recall = \frac{TP}{TP + FN}$$

Recall should ideally be 1 (high) for a good classifier. Recall becomes 1 only when the numerator and denominator are equal i.e TP = TP +FN, this also means FN is zero. As FN increases the value of denominator becomes greater than the numerator and recall value decreases
So ideally in a good classifier, we want both precision and recall to be one which also means FP and FN are zero. Therefore we need a metric that takes into account both precision and recall. F1-score is a metric which takes into account both precision and recall and is defined as follows:

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

F1 Score becomes 1 only when precision and recall are both 1. F1 score becomes high only when both precision and recall are high.