



MINI PROJECT

BAMS

Backyard Animal Monitoring System

TEAM 9 (AIML) :

K. SOHAN REDDY (100521729055)

GUZAIL AAMER (100521729023)

Project Summary

Here are some rough stats around the problem we are trying to solve :

210

million packages are lost and
often damaged

40%

of the people experience
trespassing of neighbours'
pets

29%

of the residents report
incidents annually



Problem Statement

Detecting and monitoring wildlife activity in one's backyard can be challenging, yet it holds significant value for wildlife enthusiasts, conservation efforts, and home security. Traditional methods rely on manual observation, which is often inefficient and can miss crucial events. This project aims to develop an automated system that employs deep learning techniques to detect and classify animals captured in images or video footage from a backyard camera. The system will enhance our understanding of local wildlife, support conservation initiatives, and provide valuable data for homeowners concerned about security and animal behaviour on their property.



Measuring Impact

This project not only advances the field of deep learning but also offers a practical solution for homeowners to better understand their backyard ecosystem, enhance security, and actively participate in wildlife conservation.

Requirements

OpenCV

OpenCV (Open Source Computer Vision Library), often abbreviated as cv2 in Python, is an open-source computer vision and machine learning software library. It provides a wide range of tools and functions for various computer vision tasks, such as image and video processing, object detection, facial recognition, and more. OpenCV is popular for its extensive set of functions and support for multiple programming languages, making it a valuable resource for developers and researchers working on computer vision applications. In Python, "cv2" is commonly used as an alias to access OpenCV's functions and classes.

TensorFlow

TensorFlow is an open-source machine learning framework developed by Google. It's designed to provide a comprehensive ecosystem for building and deploying machine learning and deep learning models. TensorFlow is widely used for a range of applications, including image and speech recognition, natural language processing, and various other tasks in the field of artificial intelligence.

One of TensorFlow's primary features is its flexibility. It allows developers to create, train, and deploy machine learning models on various platforms, including CPUs, GPUs, and even specialized hardware like TPUs (Tensor Processing Units). TensorFlow provides a high-level API called Keras, which simplifies the process of building and training neural networks.

TensorFlow's computational model is based on data flow graphs, where nodes represent mathematical operations, and edges represent the data that flows between them. This enables efficient parallel processing and distribution of workloads across different hardware.

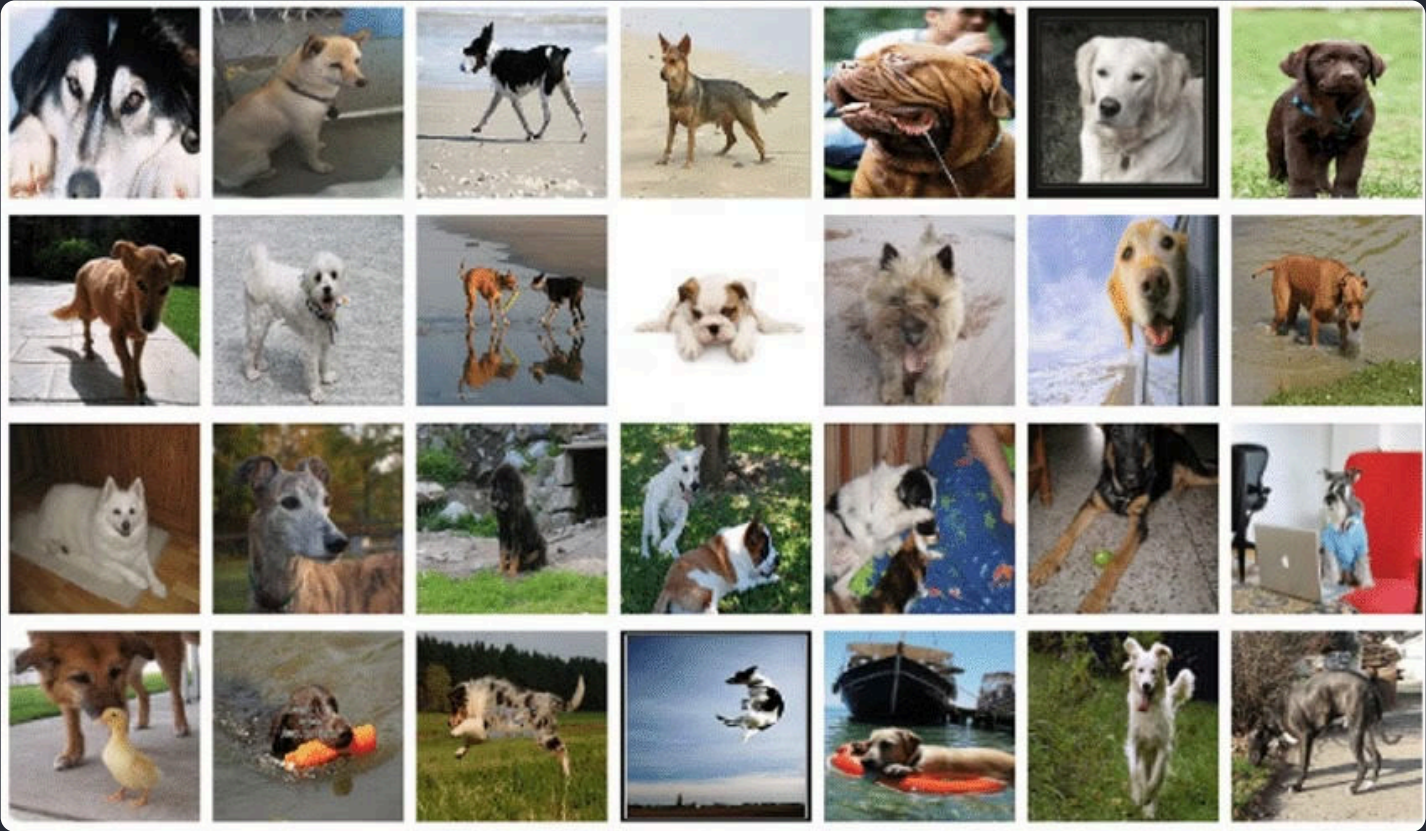
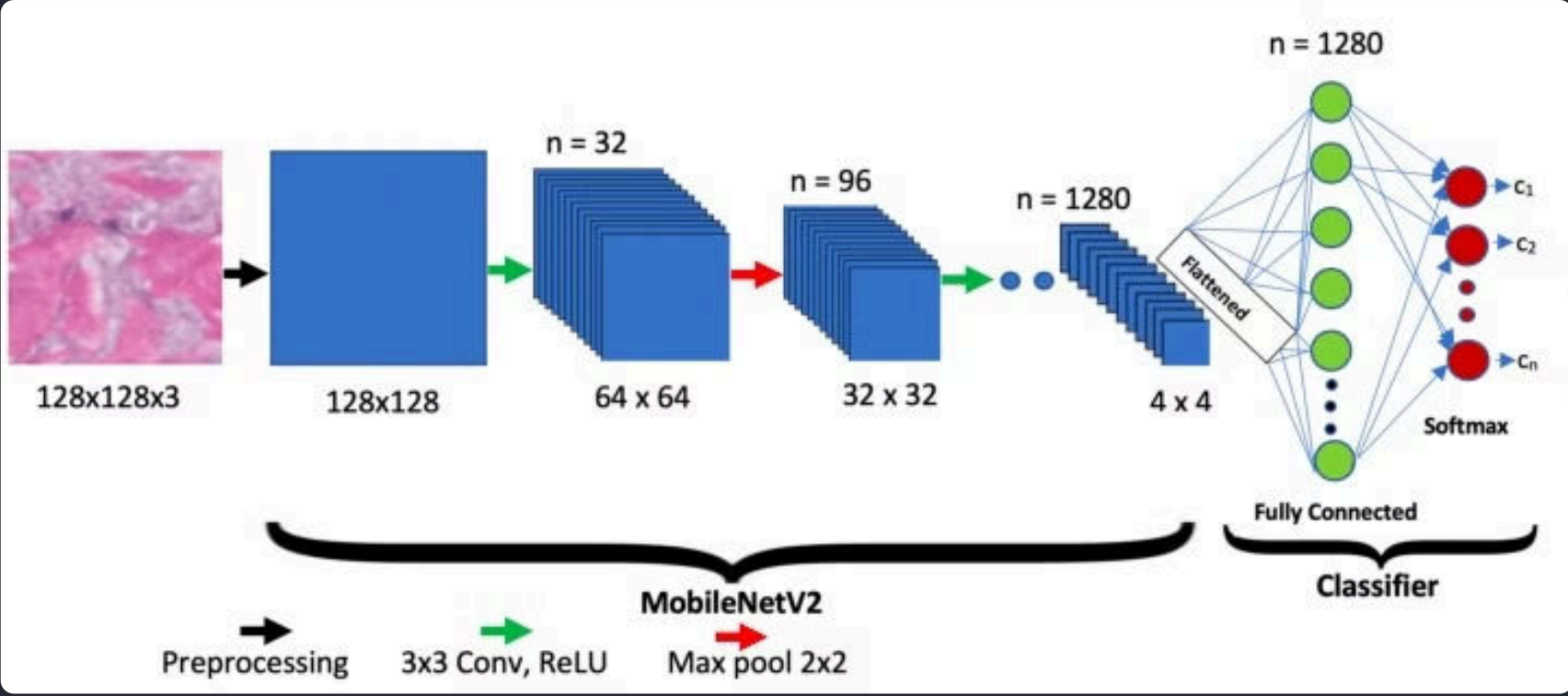
Overall, TensorFlow is a powerful and widely adopted framework in the field of machine learning and deep learning, used by researchers, data scientists, and developers to create and deploy machine learning models for a wide range of applications.

NumPy

NumPy is a Python library for numerical and scientific computing. It provides efficient ways to work with large arrays of data and includes a wide range of mathematical functions. NumPy is crucial for various fields, including data analysis, machine learning, and scientific research, as it simplifies complex numerical operations and improves performance.

MobileNetV2

MobileNetV2 is a lightweight and efficient deep learning model designed for mobile and embedded devices. It's optimized for tasks like image recognition and object detection, making it ideal for applications on smartphones and other resource-constrained platforms. It achieves high accuracy while using fewer resources by employing specialized convolutional operations.



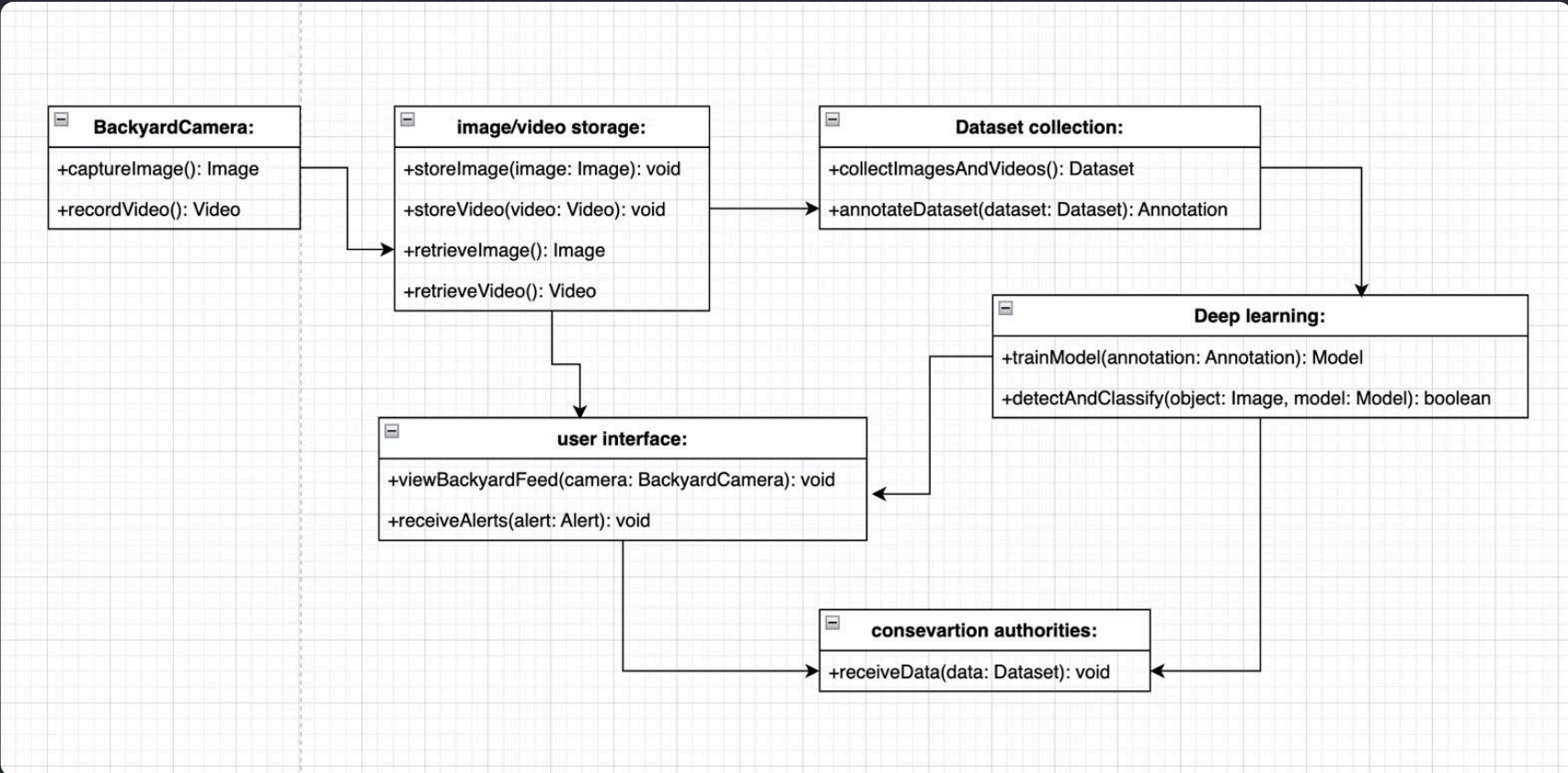
ImageNet Dataset

Size of the dataset: Over 14 million images

Categories: More than 20,000 categories

Twilio

Twilio is a cloud communications platform that provides a range of tools and APIs (Application Programming Interfaces) for developers to integrate various communication capabilities into their applications. These capabilities include sending and receiving SMS (text messages), making and receiving phone calls, and handling other forms of real-time communication like video and chat.



Design

Here's a textual representation of the key components and their relationships for your backyard monitoring system :

Backyard camera

Methods:

- captureImage() : Image**
- recordVideo() : Video**

Image/Video storage

Methods:

- storeImage(image: Image) : void**
- storeVideo(video: Video) : void**
- retrievelImage() : Image**
- retrieveVideo() : Video**

Dataset collection

Methods:

- collectImagesAndVideos() : Dataset**
- annotateDataset(dataset: Dataset) : Annotation**

Deep Learning

Methods:

- trainModel(annotation: Annotation) : Model**
- detectAndClassify(object: Image, model : Model) : Boolean**

UI

Methods:

- viewBackyardFeed(camera: BackyardCamera) : void**
- receiveAlerts(alert: Alert) : void**

Relationships

1 Backyard camera

captures images and records videos, which are stored in the Image/Video Storage component.

2 The Dataset Collection

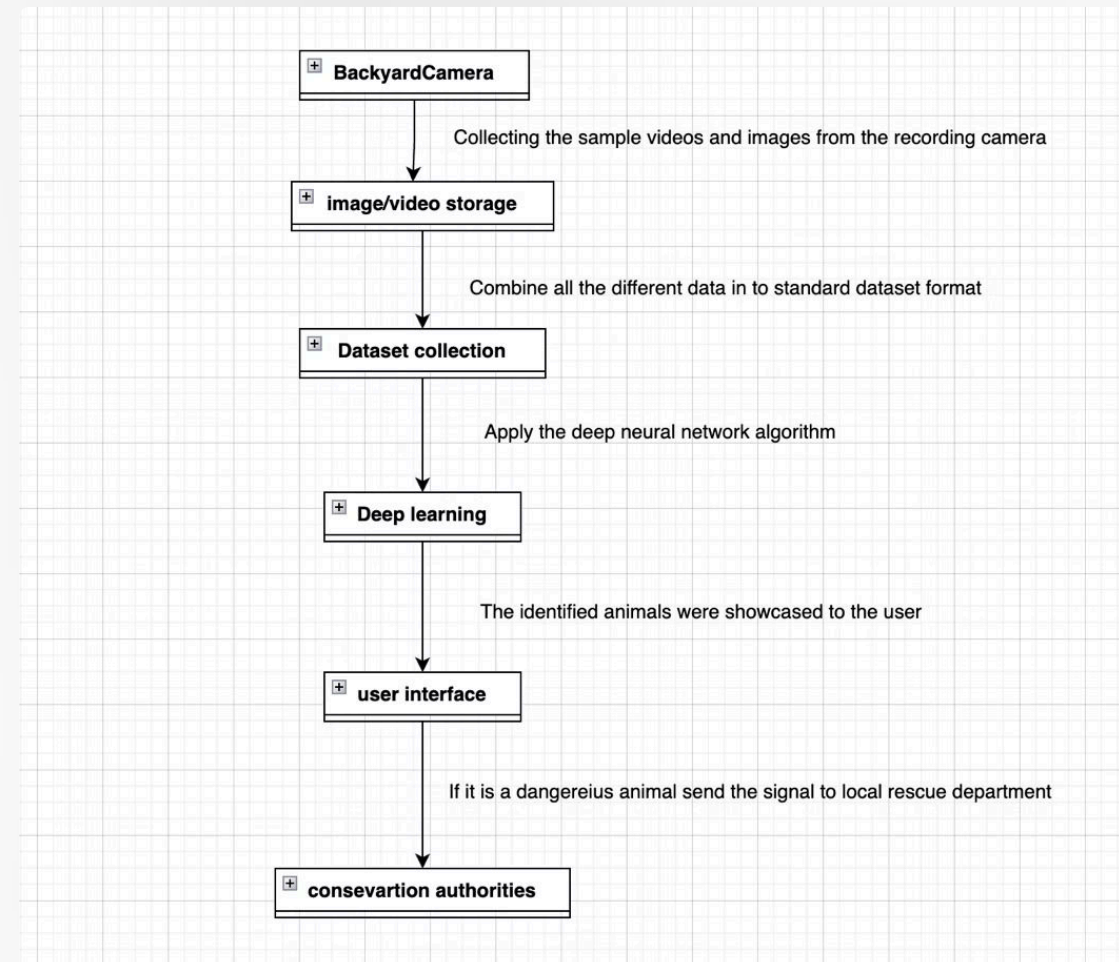
component collects images and videos, creating a dataset, and then annotates the dataset.

3 Deep Learning

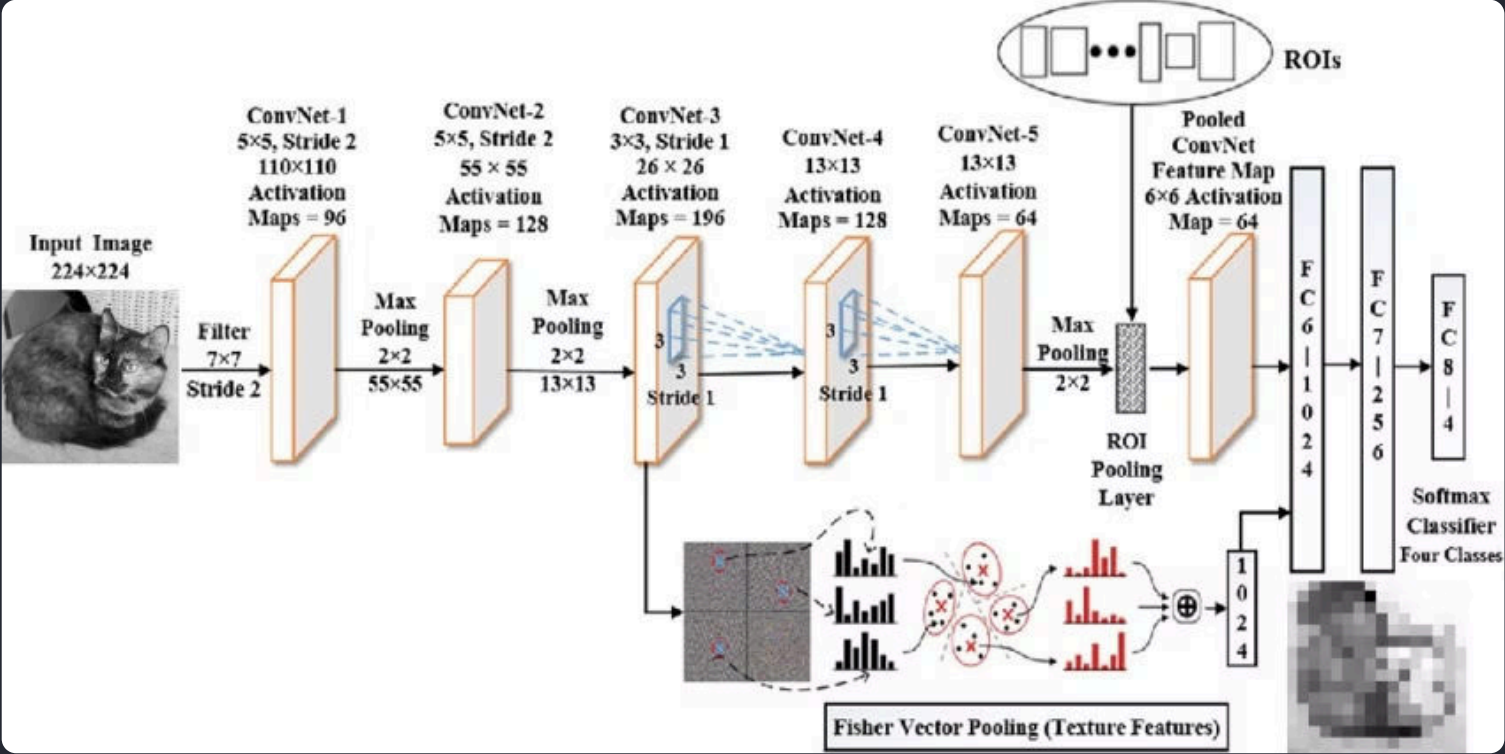
component uses the annotated dataset to train the model for object detection and classification.

4 The User Interface

allows users to view the backyard camera feed and receive alerts.



R-CNN

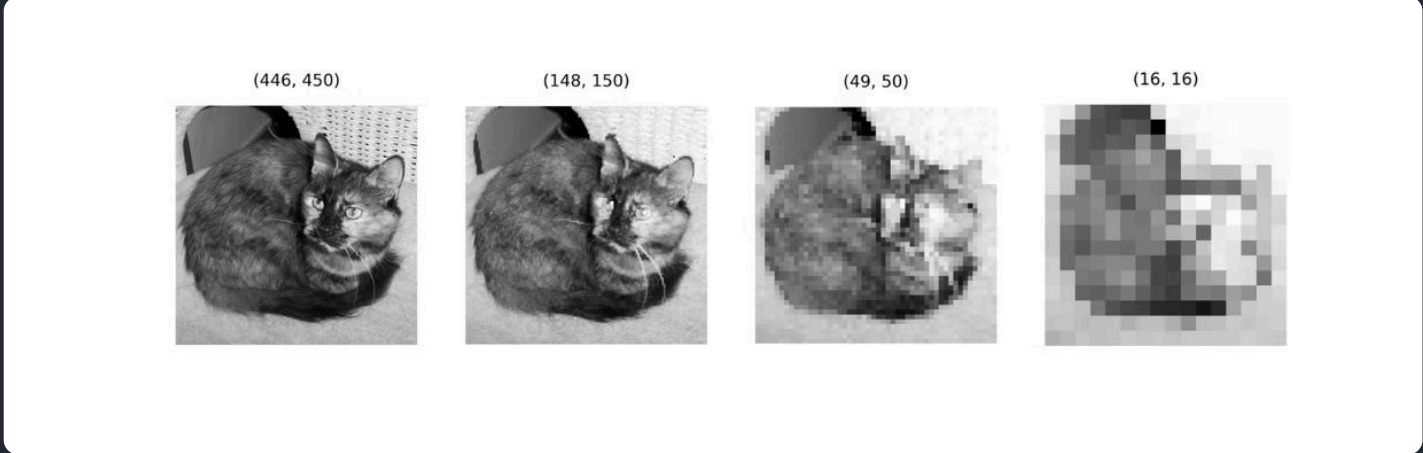
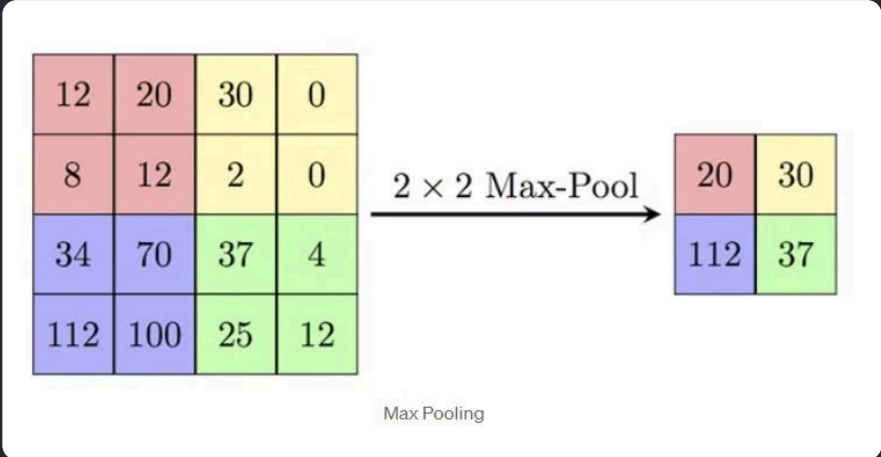


| Layer | | Feature Map | Size | Kernel Size | Stride | Activation |
|--------|-------------|-------------|---------------|-------------|--------|------------|
| Input | Image | 1 | 227x227x3 | - | - | - |
| 1 | Convolution | 96 | 55 x 55 x 96 | 11x11 | 4 | relu |
| | Max Pooling | 96 | 27 x 27 x 96 | 3x3 | 2 | relu |
| 2 | Convolution | 256 | 27 x 27 x 256 | 5x5 | 1 | relu |
| | Max Pooling | 256 | 13 x 13 x 256 | 3x3 | 2 | relu |
| 3 | Convolution | 384 | 13 x 13 x 384 | 3x3 | 1 | relu |
| | Convolution | 384 | 13 x 13 x 384 | 3x3 | 1 | relu |
| 5 | Convolution | 256 | 13 x 13 x 256 | 3x3 | 1 | relu |
| | Max Pooling | 256 | 6 x 6 x 256 | 3x3 | 2 | relu |
| 6 | FC | - | 9216 | - | - | relu |
| 7 | FC | - | 4096 | - | - | relu |
| 8 | FC | - | 4096 | - | - | relu |
| Output | FC | - | 1000 | - | - | Softmax |

- Convolutional layer - abstracts image as a feature map using filters and kernels
 - Kernel - collection of small matrices
 - Predefined size set by parameters

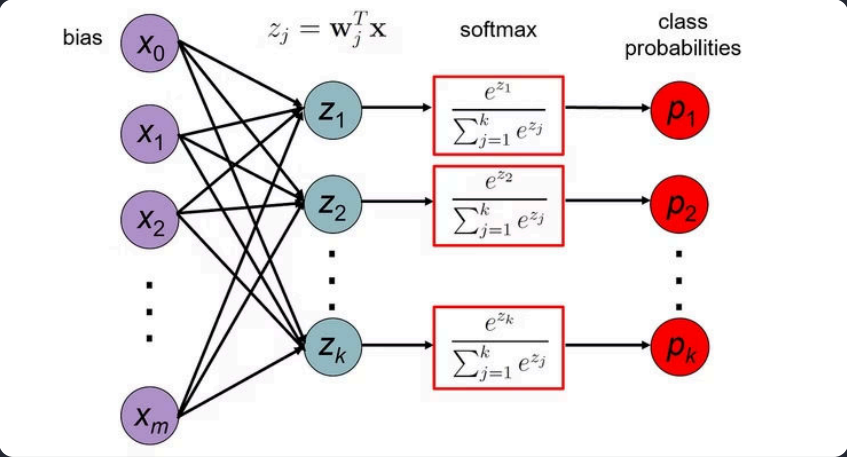
Region-based CNN - helps extract regions using ROI Pooling

- ROI Pooling layer - helps downsample feature map and summarize the presence of features
 - Progressively reduces spatial size to reduce parameters in the network
 - Max Pooling - takes maximum feature value and eliminates all other smaller features thereby reducing size of kernel

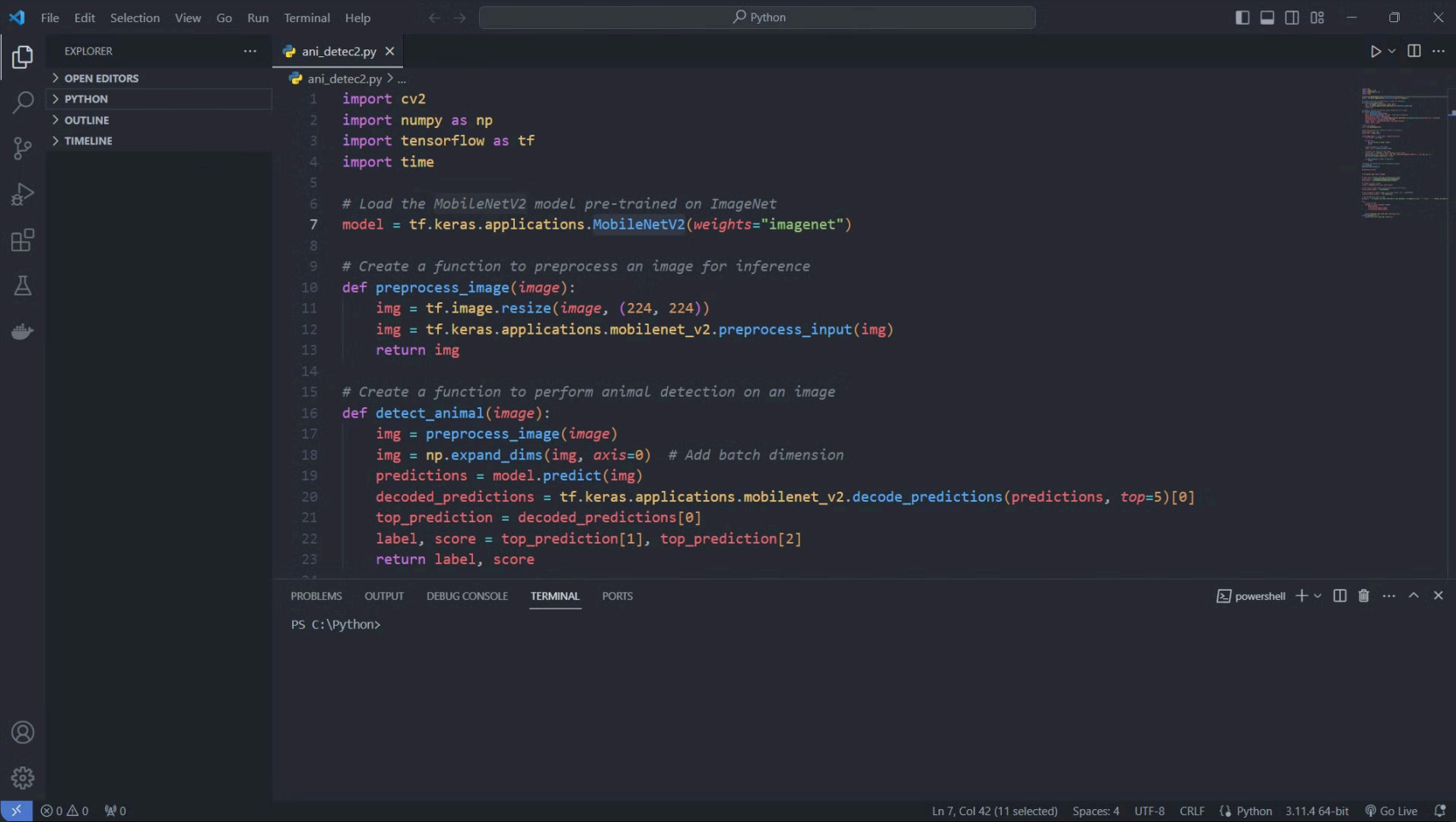


- The final feature map is sent through flattening and dense layers which initiates the learning process
- Softmax classifier uses activation function as given below

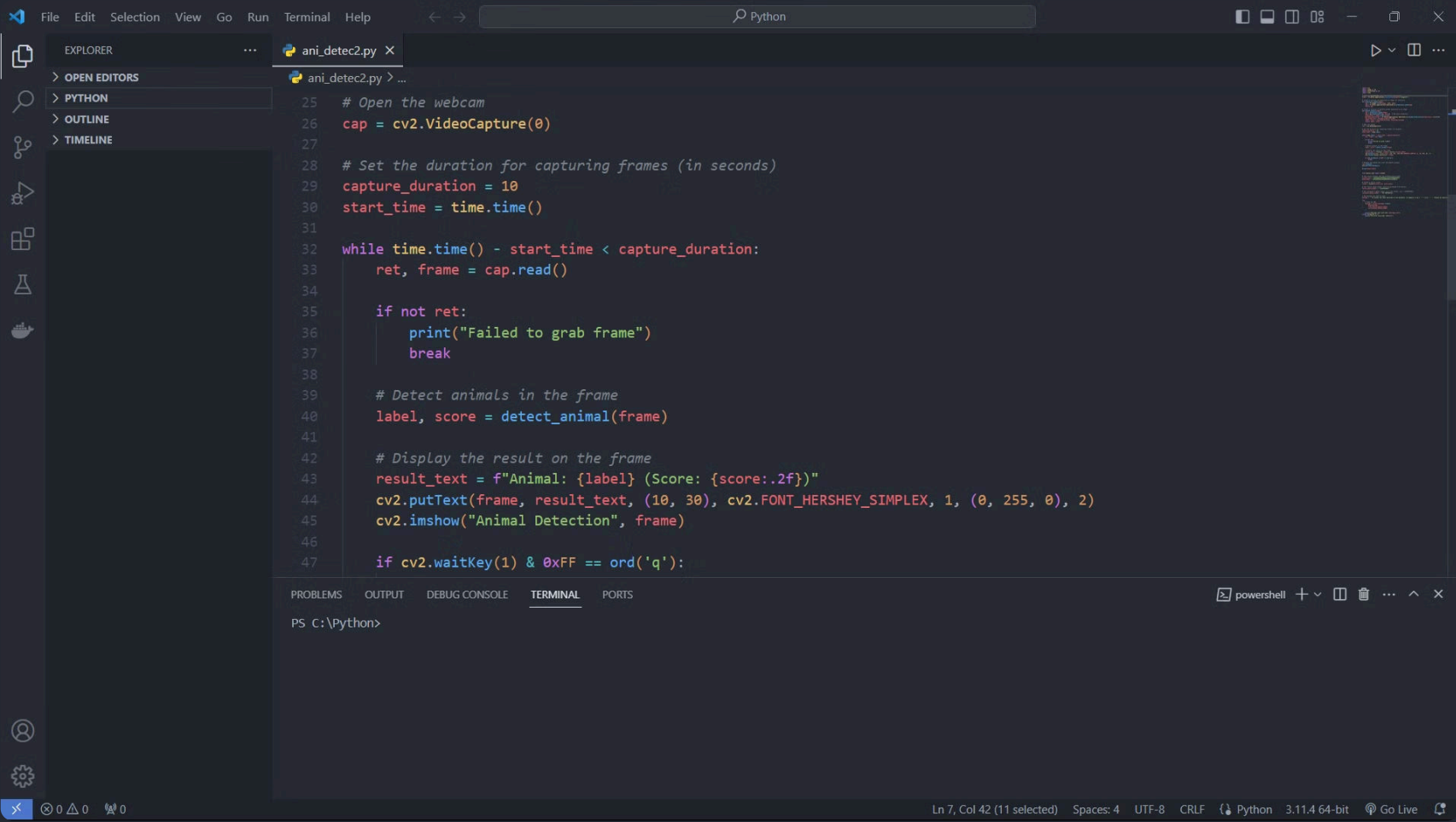
$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^N e^{z_j}}$$



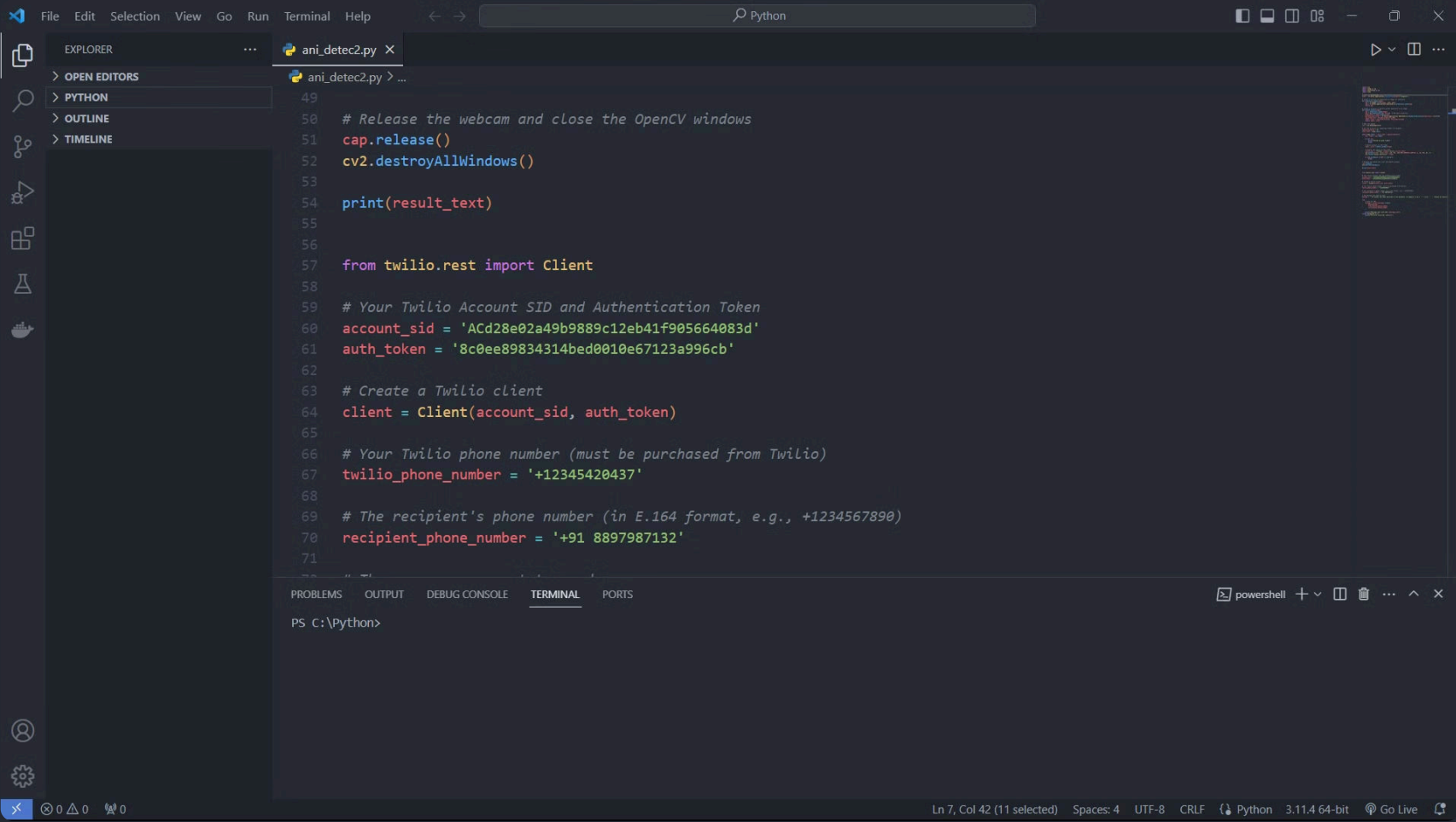
Implementation



```
1 import cv2
2 import numpy as np
3 import tensorflow as tf
4 import time
5
6 # Load the MobileNetV2 model pre-trained on ImageNet
7 model = tf.keras.applications.MobileNetV2(weights="imagenet")
8
9 # Create a function to preprocess an image for inference
10 def preprocess_image(image):
11     img = tf.image.resize(image, (224, 224))
12     img = tf.keras.applications.mobilenet_v2.preprocess_input(img)
13     return img
14
15 # Create a function to perform animal detection on an image
16 def detect_animal(image):
17     img = preprocess_image(image)
18     img = np.expand_dims(img, axis=0) # Add batch dimension
19     predictions = model.predict(img)
20     decoded_predictions = tf.keras.applications.mobilenet_v2.decode_predictions(predictions, top=5)[0]
21     top_prediction = decoded_predictions[0]
22     label, score = top_prediction[1], top_prediction[2]
23     return label, score
```



```
25 # Open the webcam
26 cap = cv2.VideoCapture(0)
27
28 # Set the duration for capturing frames (in seconds)
29 capture_duration = 10
30 start_time = time.time()
31
32 while time.time() - start_time < capture_duration:
33     ret, frame = cap.read()
34
35     if not ret:
36         print("Failed to grab frame")
37         break
38
39     # Detect animals in the frame
40     label, score = detect_animal(frame)
41
42     # Display the result on the frame
43     result_text = f"Animal: {label} (Score: {score:.2f})"
44     cv2.putText(frame, result_text, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
45     cv2.imshow("Animal Detection", frame)
46
47     if cv2.waitKey(1) & 0xFF == ord('q'):
```



```
49
50 # Release the webcam and close the OpenCV windows
51 cap.release()
52 cv2.destroyAllWindows()
53
54 print(result_text)
55
56
57 from twilio.rest import Client
58
59 # Your Twilio Account SID and Authentication Token
60 account_sid = 'ACd28e02a49b9889c12eb41f905664083d'
61 auth_token = '8c0ee89834314bed0010e67123a996cb'
62
63 # Create a Twilio client
64 client = Client(account_sid, auth_token)
65
66 # Your Twilio phone number (must be purchased from Twilio)
67 twilio_phone_number = '+12345420437'
68
69 # The recipient's phone number (in E.164 format, e.g., +1234567890)
70 recipient_phone_number = '+91 8897987132'
71
```

Output

