INSTITUTE OF SPACE TECHNOLOGY

Computer Vision and Mobile Application Development

Project Report

Topic: Visual Pollution Detection

Group Members:

Asad Abbas

(200901024)

Faraz Ahmad Qureshi

(200901045)

Semester 6th

BSCS – 01

Section B

Course Instructor:

Sir Tufail

Abstract

This report presents the outcomes of a project focused on detecting visual pollution through the integration of computer vision techniques and mobile application development. The project was divided into two main phases: the development of a computer vision model and the creation of a mobile application to deploy the model. In the computer vision component, we explored different datasets and machine learning models, with the primary objective of optimizing the model for mobile device deployment. Our journey involved multiple iterations, each accompanied by its own set of challenges and valuable insights. Initially, we experimented with YOLOv8 and YOLOv5s models, but after extensive testing, we achieved our desired model training using the pascal_voc dataset, which we subsequently implemented in our application. Concurrently, we undertook the development of a mobile application using Kotlin in Android Studio. Our initial attempts were confronted with several obstacles, including compatibility issues with models, dependencies, and frequent application crashes. However, through persistent efforts and an iterative development approach, we successfully created a fully functional application capable of real-time object detection through both camera view and image analysis from the gallery. The culmination of our project is a mobile application that can promptly identify and highlight visual pollution, thus contributing to the broader goal of environmental preservation. This report provides an in-depth account of our journey, including the challenges we encountered, the solutions we implemented, and the knowledge we acquired throughout the project.

Table of Contents

Computer Vision

Introduction

This report provides a comprehensive overview of our computer vision project, which aimed to detect visual pollution on roads. The project involved the use of various datasets and machine learning models, with the primary goal of deploying the final model on mobile devices.

First Attempt: YOLOv8 and YOLOv5s

Dataset and Initial Training

The first dataset we used consisted of **24,000 training images** and **7,000 validation** images. These images were categorized into three classes: barriers, sidewalks, and potholes. We initially trained our model on *YOLOv8* for three epochs. However, this resulted in low accuracy.

Transition to YOLO v5s and Results

Due to the unsatisfactory results with *YOLOv8*, we transitioned to training the model on *YOLOv5s*. Despite hardware limitations, we managed to train the model another eight times, achieving an accuracy of approximately 0.6% after a total of 20 epochs.

The model concluded its training and demonstrated capability in one of the key areas for our research requirements; it appeared knowledgeable and capable of detecting objects in unseen data. We successfully converted it to *TensorflowLite* model to distribute our work on mobile and edge devices. However, this is where we met our first obstruction. We were unable to deploy it on android because of the fundamental difference in *YOLO* and *Tensorflow* based models in ML. Our model appeared to cause issues with the number of Output Tensors in the *TensorflowLite* model’s input and output as it was trained with *Yolov5s* which usually results in an odd number of output tensors compared to *Tensorflow* which preserves the original shape of the dataset. After trying out various techniques and methods including model reshaping as well as image transformation before inference, our techniques did not give us a workable solution. Moreover, our research discovered that *TensorflowLite* is considered a deprecated system of machine learning models however because there wasn't any other alternative just yet, we had to find a way.

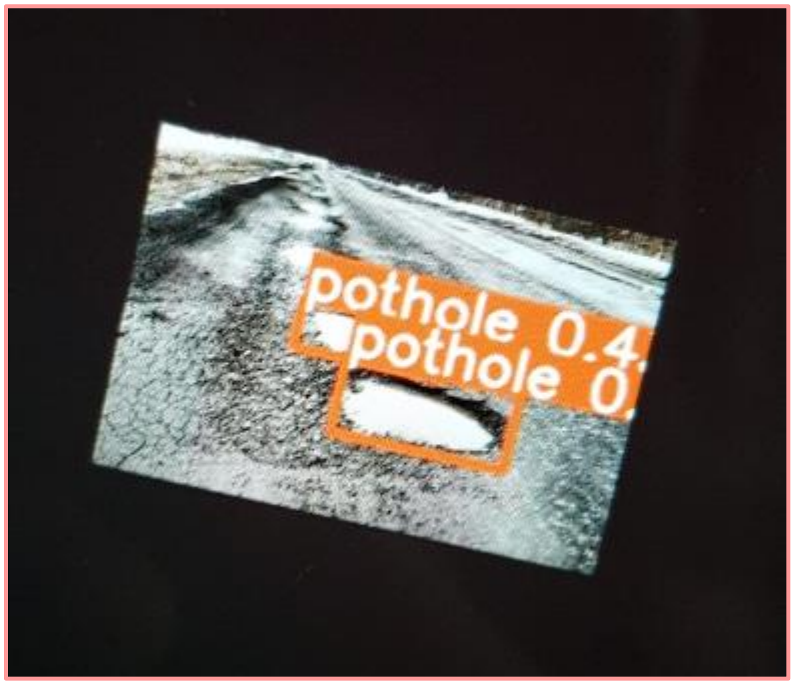


Figure 1 YOLO Object Detection Model

Second Attempt: Reduced Dataset and Parameter Misconfigurations

Dataset Adjustment and Training

In our second attempt, we used the same dataset but reduced the number of images for training and validation by half. We trained the model for a total of 18 epochs.

Results and Issues

Unfortunately, due to some misconfigurations of parameters, the best precision we achieved was 0.4. We hypothesized that the precision could have improved with more epochs.

Third Attempt: New Dataset and Improved Results

Dataset Selection and Training

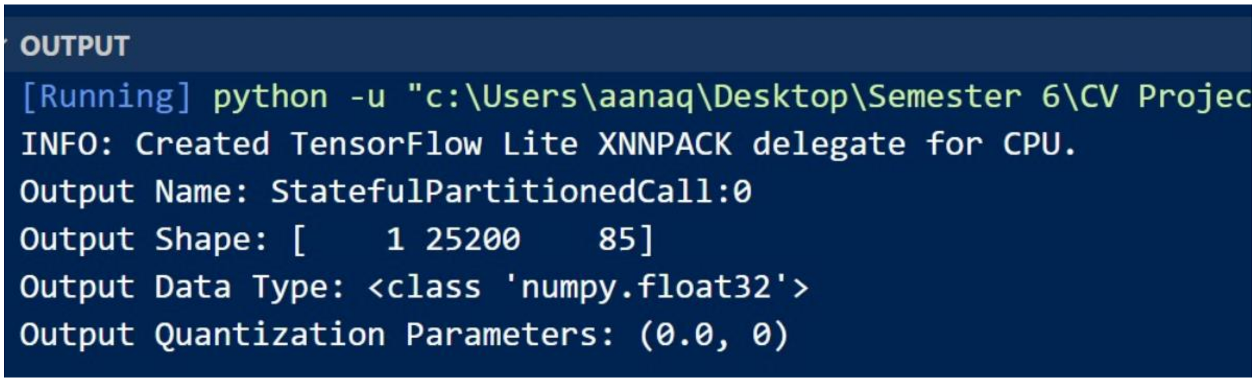
For our third attempt, we decided to use a different dataset, which only included a 'garbage' class. We downloaded this dataset from *Roboflow* and trained our model again on *YOLOv5s.*

Results

This time, we achieved significantly better results, with a precision of 0.8. The model was successful in detecting objects in unknown data, demonstrating its improved performance. We successfully converted it to TensorFlow Lite however we were unable to deploy it on Android because of issues with Output Tensors as the TFLite models trained with Yolo v5s usually result with one output tensors which was unsupported by TFlite Model Maker as well as several other platforms where we could deploy it.

Main Reason for failed Deployment:

The screenshot below shows that for all the models that we trained in yolov5s, which were converted into TF-Lite resulted in the output tensors being just one. As we visualized from **https://netron.app/** website. We tried multiple times to make our application work in android studio by creating numerous applications but all used to fail when loading our model.



```
OUTPUT
[Running] python -u "c:\Users\aanaq\Desktop\Semester 6\CV Projec
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Output Name: StatefulPartitionedCall:0
Output Shape: [     1 25200     85]
Output Data Type: <class 'numpy.float32'>
Output Quantization Parameters: (0.0, 0)
```

Our YOLOv5s model showcased 3 output tensors; these were addressed as P3,P4 and P5. Meanwhile the output tensors for Tensorflow, MLkit or TensorflowLite in this case were fundamentally different and used an older method to address detection boxes and find the object on screen.

Successful Deployment of Model on Mobile Devices.

Dataset Selection and Training

For our fourth and final attempt, we decided to use the same dataset as last time, which only included a 'waste' class. We downloaded this dataset from *Roboflow* and trained our model again by using annotations and images in *Pascal_voc* format. The annotations were different than yolo as xml was included. After our research we were able to train a TFRecord temporary model which was compatible with our tf_lite_conversion. However, due to deprecation, it was almost impossible to develop and train a modern tf_lite_model. However, a discovery and breakthrough in the eleventh hour made our project switch gears and we trained not only one but two garbage detection models which were able to work accurately with our application. A similar approach was taken in training the pothole / road condition dataset/model..

Results

The generated model was in a format which was supported by TFLite and included four output tensors. Then we further converted it into a tf_lite model which was able to work with android studio after we generated metadata for it. We have used this model in our mobile application for live camera detection and image detection from gallery. The TFLite models trained with Yolov5s usually result with one output tensors which was unsupported by TFlite Model Maker as well as several other platforms where we could deploy it but this one successfully worked in our android application.

Conclusion

Throughout the course of this phase, we learnt valuable lessons about the importance of dataset selection, parameter configuration, and the choice of machine learning model. Despite some challenges, we made significant progress and achieved promising results in our final attempt. Future work will focus on further improving the precision of our model and exploring its deployment on mobile devices.

Developing The Mobile Application

Goal: Creation of Basic Mobile Application to display Our Model

Introduction

Alongside our computer vision project, we embarked on a journey to develop a mobile application that could utilize our trained model for real-time visual pollution detection. The primary language used for this development was Kotlin in Android Studio. This report will detail the challenges we faced, the attempts we made, and the final product we developed.

First Attempt: Basic Structure and TensorFlow Lite Integration

Our first attempt at creating the mobile application resulted in a basic structure with working camera functionalities and the ability to load images from the gallery. However, when we tried to integrate TensorFlow Lite for model deployment, we encountered numerous crashes.

Challenges and Issues

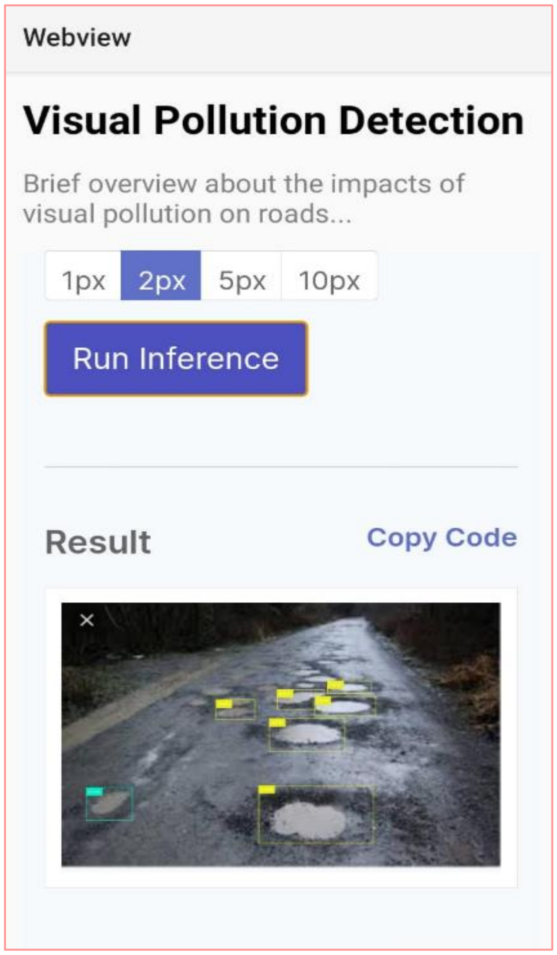
The main challenge during this phase was the lack of comprehensive guides on the internet. Most available resources were either outdated or had unsupported dependencies. Additionally, we faced several Gradle-related issues, which further complicated the development process. Despite these challenges, we managed to create a basic structure for our application, which was a significant first step.

Second Attempt: WebView and Roboflow Integration

In our second attempt, we created a fully functional application that utilized WebView to redirect to the Roboflow website. This allowed us to perform object detection on custom images by providing the API key and the model link.

Details

The application was designed with a user-friendly interface that allowed users to upload images for object detection. Once an image was uploaded, the application redirected to the Roboflow website, where the image was processed, and the results were displayed. This approach allowed us to bypass the issues we faced with TensorFlow Lite integration in our first attempt. However, it was not the ideal solution as it relied heavily on internet connectivity and the Roboflow platform.



Brief overview about the impacts of visual pollution on roads...

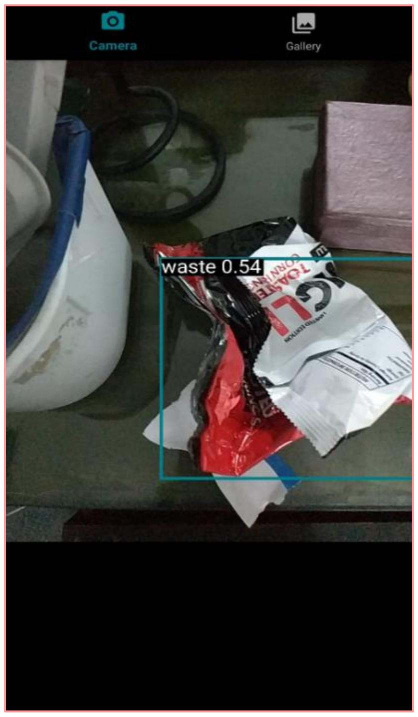
Figure 2 Detecting Potholes through model

Third Attempt: Successful Deployment of Model in Mobile Application

In our third and final attempt, we successfully developed the desired application, which utilized our trained model in a supported format. This application was capable of performing real-time object detection using the camera view, as well as detecting "waste/trash" related objects in images uploaded from the gallery.

Details

The application was designed with a clean and intuitive interface, making it easy for users to navigate and use. The real-time object detection feature allowed users to point their camera at any scene, and the application would identify and highlight any detected waste or trash objects. Similarly, the image upload feature allowed users to select images from their gallery, which the application would then analyze for waste or trash objects. We used a combination of mediapipe inferencing techniques as well as pretrained models to reduce the size of android application while increasing the performance of low-power/ edge devices. We use CPU based inferencing as mobile GPUs cannot handle too much load. Once the objects are detected we can start drawing bounding boxes around them based on the coordinates that are being returned.



Conclusion

The journey of developing a mobile application to deploy our trained model was filled with challenges and learning opportunities. Despite the hurdles, we managed to create a fully functional application that successfully utilizes our trained model for real-time visual pollution detection. Future work will focus on improving the application's performance, user interface, and expanding its capabilities to include more object classes.