Point-Cloud Based Image Classification

Mrudani Pimpalkhare, Bhavya Ahuja
International Institute of Information Technology, Hyderabad

Mithun Rameshbabu
International Institute of Information Technology, Hyderabad

Eshaan Sharma
International Institute of Information Technology, Hyderabad

***Abstract*—As part of our coursework in the Embedded Systems Workshop, we explored the capabilities of the Qualcomm Integrated Development Kit (QIDK). Our project focused on implementing a Point Cloud Image Classification Model on the QIDK, contributing to a larger initiative led by CVEST and CVIT, IIIT Hyderabad..**

AS A METHOD TO contribute to Road Safety Applications, the CVEST(Centre for VLSI and Embedded Systems) and CVIT (Centre for Visual Information Technology) labs, implemented a Point Cloud Image Classification using FMCW Radar. Their work presents millimeter wave (mmWave) radar sensing based weather-agnostic, real time object classification system for road safety applications. A unique experimental point cloud data set comprising about 15,000 samples across three categories (pedestrians, 2-wheeler, and 4-wheeler) is also created and made available in the public domain. Experimental results show that the ML model suggested by our analysis requires about 70 MB of memory and the proposed system achieves 96accuracy in real time while classifying pedestrians, 2-wheeler and 4-wheeler vehicles.[1]

The core model that is used for classifying objects, as the name suggests, uses point cloud images. These images, generated using FMCW Radar and internal working are converted from 3D to 2D point cloud images, and then sent to the model, and analysed, as seen in Figure 1 (Page 2). Our team's objective was to modify the model according to our requirements, and then create an Android App out of it, which is compatible with the QIDK.

The Qualcomm Innovators Development Kit (QIDK) is a robust platform designed to support the rapid prototyping and deployment of advanced AI and machine learning applications. It leverages the power of Qualcomm's cutting-edge processors and neural processing units (NPUs) to facilitate efficient, high-performance computation for on-device AI workloads. One of the many advantages of utilising a QIDK is its high efficiency, low power consumption and scalable AI performance, which We were able to appreciate these advantages while making our application.

Digital Object Identifier 10.1109/MCE.YYYY.Doi Number
Date of publication DD MM YYYY; date of current version DD MM YYYY

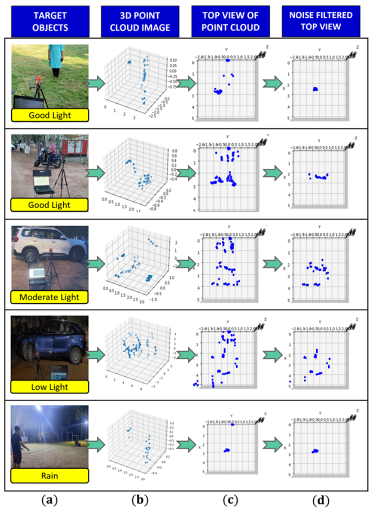


FIGURE 1. (a) Experimental setup for data collection under different environ mental conditions (b) 3D Point cloud image generation (c) Top View image generation for the same point cloud image (d) Noise filtered from the top view image

INITIAL STEPS

The raw model and testing-training data (as point cloud images) were provided to us. However, we had to make this model compatible with our system, and the test images that were given to us. This included one major part : Converting the images into binary images, that contained only the blue dots (these dots represent segments of the object we wish to detect.) This phase is called the preprocessing of the image, and initially it was assumed that these blue dots would be pure blue. This did tend to give incorrect results, even if the assumption was rightfully correct. We modified the code to check for a range of blue shades instead of pure blue, and we succeeded. Finally we trained the model, which used SVM and PCA, to get the following results, as shown in Table 1.

TABLE 1. Evaluation Metrics for Logistic Regression and SVM

Model	Class	Recall	F1-Score	Accuracy
3*Logistic Regression	0	1.00	1.00	3*0.92
	1	0.98	0.88	
	2	0.84	0.91	
3*SVM	0	1.00	1.00	3*0.94
	1	0.98	0.90	
	2	0.87	0.92	

CREATING THE APP

The next aim is to run this trained model on an Android App. This required saving the trained model, and converting it into a format that is supported by Android Apps. We chose TensorFlow Lite(TfLite) as it is the most optimised for mobiles and embedded devices, has lightweight framework, and flexible interface. We saved our trained model into a .pkl file, and then converted it into a .tflite format, which was used by our app. The user basically uploads a point cloud image, and the predicted result is displayed on the screen.

Figure 2 demonstrates a screenshot of one of the results obtained.

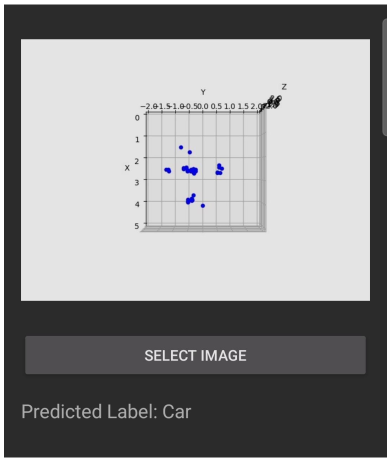


FIGURE 2.

OUR JOURNEY

All the members of the team were new to running, training and testing models, along with creating Android Apps. Hence creating an app which runs a model was not a one-step job. Our journey towards creating our final project successfully consisted of the following milestones:

A. Running, training and testing any model on our own, locally.
B. Running an app succesfully using a pretrained model
C. Creating an app out of scratch, using a very simple model that doesnt require too much logic.
D. Training a model, and building an app that uses it. (our end product)

MODEL

Running, training and testing

Our aim was to understand how certain models work, and how to take care of the software and dependencies that are needed. We also wanted to understand how the analysis of certain models work. For local training, we used YOLO (You Only Look Once), which is a real-time object detection system that predicts bounding boxes and class probabilities directly from full images in one evaluation. Here, we learnt how to use Google Collab's NVDIA GPUs to improve training speed.

We used Kaggle's YOLO-v10 vehicle detection system, consisting on classes like ambulances, bikes, cars, etc. On training an testing, we obtained the analysis reports, consisting of a confusion matrix and various parameters that determine the efficiency of a model.

We also successfully managed to Train a SVM model that detects objects, and it works on the dataset provided by the TA

The script builds an SVM-based image classifier to identify vehicle types (hatchback, motorcycle, pickup, sedan, SUV). Images are resized, flattened, and split into training and testing sets. The model is optimized using GridSearchCV and achieves a high accuracy, saving the best version as a .pkl file. It also allows real-time prediction for new images, displaying probabilities for each category. This efficient pipeline is ideal for vehicle classification tasks

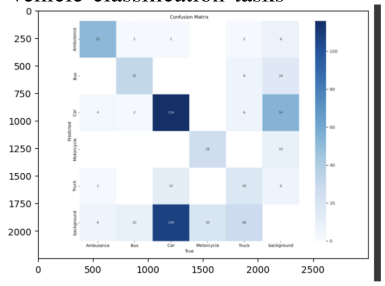


FIGURE 3. Confusion Matrix

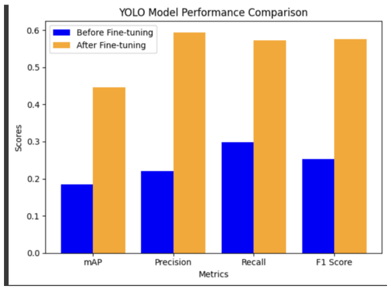


FIGURE 4. Confusion Matrix

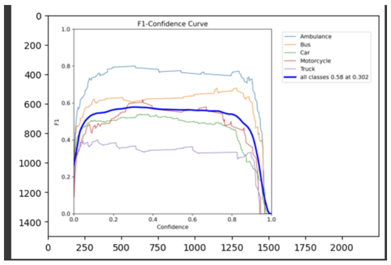


FIGURE 5. Confidence Curve

APP OUT OF A PRETRAINED MODEL

Pose Detection Model

The application uses a pose detection model to analyze the user's performance in real-time. This model, built using the Google ML Kit, identifies key landmarks on the user's body, such as elbows, wrists, knees, and ankles. These landmarks are used to calculate angles and other metrics that represent the user's pose. Reducing Images to Key Features

The pose detection model works by reducing images to key features, such as the angles of key landmarks. This allows for efficient comparison between frames and accurate pose identification. The model utilizes a specific topology called BlazePose, which offers 33 human body keypoints, providing a more detailed understanding of the user's posture compared to traditional methods.

BlazePose Topology

The BlazePose topology provides a richer representation of the human body, incorporating keypoints for hands and feet, which is crucial for accurate pose analysis.

Key Landmarks

These landmarks include points on the torso, arms, legs, and face, enabling a comprehensive understanding of the user's pose. (Figure 6)

APP FROM SCRATCH : SIMPLE

We used a very simple scikit based model (given our final project also uses scikit), and made sure that the model does not contain a lot of complex logic, since our main focus was to learn how to build the basic app. We referred to the following link : https://towardsdatascience.com/deploying-scikit-learn-models-in-android-apps-with-onnx

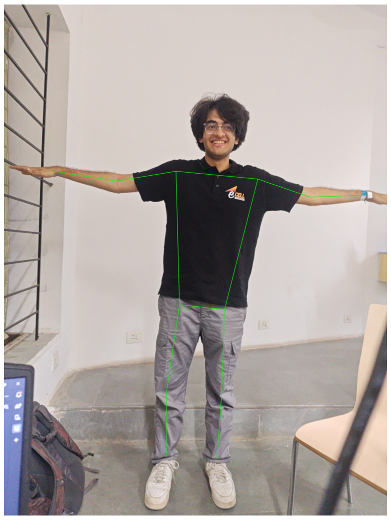
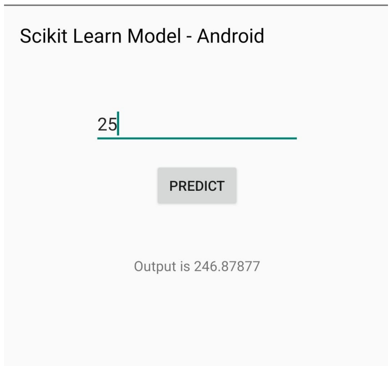


FIGURE 6. Proper Identification of pose

This is a basic model, that runs and predicts data using a .csv file. The code was quite straightforward, when it came to the logic. This way we got to learn about the core of making an Android Studio App, rather thatn worrying about the model functionality

We faced challenges with dependencies but were finally able to run the app, as shown in Figure 7


Scikit Learn Model - Android

25

Output is 246.87877

FIGURE 7. scikit model app from scratch

INDIVIDUAL CONTRIBUTIONS

Mrudani Pimpalkhare

1. Creating an Android app from scratch using Scikit
2. Creating the android app for point-cloud based image classification
3. Training the main model, and converting it into ONNX format(using this method was rejected during the final deployment of our app)
4. Running the yoga pose detection model 5. Report

Bhavya Ahuja

1. Creating the android app for point-cloud based image classification
2. Training the main model, and converting it into TfLite format(used this method during the final deployment of our app)
3. Running the SVM model 4. Presentation

Eshaan Sharma

1.Running and Finetuning the YOLO Models for classfication and statistics
2. Building the app and running the yoga pose detection app
3. Training SVM model on dataset given by TA and Statistics
4. SVM and Linear regression for Image classification
5. Presentation and report

Mithun Rameshbabu

1. Presentations and reports
2. Training the initial SVM Model with point cloud images

CONCLUSION

In conclusion, through various resources and learning methods, we were able to successfully contribute to a Machine Learning Project by training a model and building an app based on it, from scratch.

ACKNOWLEDGMENTS

We thank Prof. Abhishek Srivastava for providing us with an opportunity to take up this project and giving us guidance throughout the course.

Also Sri Rama Rathan Reddy, a CVEST honors student under Prof. Abhishek, along with our TA Ayush Raghuvanshi

We would also like to thank CVEST and CVIT for guiding us by providing the code and datasets.

REFERENCES

1. Point Cloud based Object Classification System using FMCW Radar for Road Safety Applications Srayan Sankar Chatterjee, Pranjal Mahajan, Adithya Sunil Edakkadan, Jewel Benny, Kiruthika K*, Arpit Sahni, Ravi Kiran S* and Abhishek Srivastava

Mrudani Pimpalkhare is a CSE student pursuing her second-year at IIIT Hyderabad, contact her at mrudani.pimpalkhare@students.iiit.ac.in

Bhavya Ahuja is a CSD student pursuing his second-year at IIIT Hyderabad, contact him at bhavya.ahuja@research.iiit.ac.in

Eshaan Sharma is a CSD student pursuing his second-year at IIIT Hyderabad, contact him at eshaan.sharma@research.iiit.ac.in

Mithun Rameshbabu is a CSE student pursuing his second-year at IIIT Hyderabad, contact him at mithun.rameshbabu@students.iiit.ac.in