Reflection
4ZP6: DieSpy

Team #9

Jackson Cassidy
Wyatt Habinski
Christian Majid
Paul Puscas

April 4, 2025

Github Repo

Achievements

DieSpy is an android app that leverages machine learning to detect and count dice rolls in real time using a mobile camera, while also providing a platform for seamless collaboration. Built using Android Studio, and TensorFlow Lite, the app integrates a custom detection agent trained on a YOLOv10 model, achieving precision and recall rates above 95%. Multiplayer functionality is enabled through party features, with real time updates and data synchronization using Firebase Firestore.

Requirements Completion

Functional Requirements

P	Requirement	Completion
P0	Camera functionality	**100% Complete:** Camera live video feed and averages it over 10 frames
	Dataset	**100% Complete**
	Detect and Identify D6s, Count number of D6s, Sum D6	**100% Complete:** The camera detects and identifies dice, counts the number of each value rolled, and the total sum.
	Offline mode	**100% Complete:** Provides camera functionality and detection without needing an account or a network connection.
P1	Scan from Live Video Feed	**100% Complete:** Camera live video feed and averages it over 10 frames. Has a manual "Capture" button to save the data, and allows for data editing in the event of a misinput.
	Login screen, option to Host or Join	**100% Complete:** Users have a fully functional account system using an online database, and can create and join different parties at will.
	Connect to host and view results	**100% Complete:** Users can either join a party with a passcode regardless of distance, or join a nearby party using bluetooth.
	Maintain account/turn system, i.e track rolls by who's turn it is	**100% Complete:** Users must create an account and login to play, and rolls are tracked by user when they are logged.

*We were also able to complete 5/6 of our P2's

What Worked

The key feature of our app, Dice Detection with the camera, worked absolutely perfectly. We set up a YOLOv10 model that detects the number of dice cross referenced, and averaged over 10 frames for consistency. The results were extremely good, with precision and recall rates over 95% in testing. Additionally, we were able to easily store our model in a TFLite file, which allowed for very easy embedding of our model in the app.

Furthermore, our decision to design the detection around an android app worked well. CameraX was easy to integrate, and Android provided access to many libraries that we used for other features, such as Firebase and BLE. Additionally, android provides strong processing power that is able to run our TFLite small model. Our app provides a seamless user interface that would not be possible on another medium, such as a desktop.

Finally, our use of a cloud database via Google Firebase significantly improved the accessibility of our app. Storing everything on the cloud allowed for chats, logs, and player info to be accessible from any device with real time listeners and updates. This made all our functionality much more convenient and secure, as you could now access all our features at their maximum potential convenience, with the only requirement being an internet connection for online mode.

What Did Not Work

Originally, we planned a local host-client model, but it was harder to program and limited users and parties to specific devices. Instead, we switched to Google Firebase for cloud hosting, simplifying our code and enabling remote connections.

We first tried using WifiDirect (WifiP2P) for local joins with TCP sockets for data, but host switching was cumbersome and the library was unreliable, with connections being difficult to maintain. We then switched to Bluetooth Low Energy (BLE), which was easier to develop and offered more stable, longer-range connectivity.

What We Learned

This project primarily taught us what it would be like to create a project from concept to final delivery. We had the chance to put much of our previous knowledge together, including Object Oriented Programming, ML training, Concurrent Programming, Cloud Storage, and API use. While we previously used these individually in different courses, we had a chance to use these in tandem with other “soft skills”, such as requirements engineering, system design, and use-case analysis.

We also learned about how much effort goes into non-technical portions of the project. With most of the year being spent in the design phase, we appreciated the role that all of our documentation played in the SDLC. We previously underestimated how hard it would be to just code from scratch: our documentation provided an unexpected but much needed roadmap.