Milestone M3 report

Current State of the Project:

During this milestone, our team focused on completing two primary tasks. The first task was to further improve the accuracy of the models and then combine all our models (the pretrained YOLO model, the game recognition model, and the game recommender model) in a single python script to test the complete workflow locally. The second task was to build a backend for the heavy computation required by our app, i.e., processing the inputs with the two ML models. The backend responds with recommendations upon and HTTP request to the server. This task also includes creating a system to allow the uploading of game images to the server and returning responses to the frontend app. In addition, we have also finished working on the User Interface for our Android App.

In order to build our backend, we used Node.js. We used the 'express' framework for setting up the server, 'multer' to process and store the uploaded image file, and 'spawn' in order to run python scripts. The communication (request and response) between our android app and the backend/server is to be as follows:

- Upon request, the app will connect to the backend/server.
- The user will upload (up to) 3 images of game covers using his camera or the camera roll.
- This will be converted into a bitmap and then a byte array stream which will be sent over and decoded in the backend into an image and then passed as inputs to our model.
- The models will compute the 3 recommendations and respond to the app with the strings of the game titles which will be conveyed to the user on the app.

Despite having a clear and focused plan as described above, much of this work is new to us as a team. We are working on building basic projects (to gain some experience by following online tutorials) and then recreating them in context of our project. As we go through this process, we find ourselves running into some problems. Hence to conclude, our current state of the project includes fully functioning ML models with desirable accuracy, a working backend, a UI for the android app, and a script that can perform test runs with all our ML models combined. The majority of the work remaining involves establishing communications between the backend and the frontend app, which we are working on and will have ready by the end of the final milestone.

Feature Changes to the Proposal:

(In the Risk & fallback plan part)

- If we were to face a lot of difficulties in image (to bitmap) to byte array conversion over the server, we might return just the recommended game titles and not the images.
- If we got nowhere in building a communication through Android Studio, we might switch to a different platform like StreamLit to build a web app (but this is very unlikely, given that we have a working UI).

Current Challenges/Bottlenecks:

- We are facing difficulties in maintaining quality of images when sending them over the network to the backend for processing. Whether it affects the accuracy of the model remains to be seen through further testing.
- Furthermore, we are in a race against time to have the app and backend working cohesively so that we have enough time to deploy the app on Google Cloud to allow it to be run not just locally.

Team Member Responsibilities:

- Aayush: Further tested and trained the game recognition model, compiled all the models in a single script to be tested locally as a module, worked on the Node.js backend.

- Harshil: Extensive research into deployment of the ML model on Google Cloud, building a backend using Flask (scrapped), worked on get and post requests from the app to the server, and the milestone report.

- Kolton: Completed the Frontend App UI, worked on the communication between the app and the server.

- Nick: Further tested and trained the game recommender model to get a desirable accuracy, remapped the game indices in the datasets to allow the models to run together, and did research on RetroFit, a library that establishes communication to the backend from an Android app.