Multi-Modal Model on Edge

Android app for near real-time zero-shot object detection

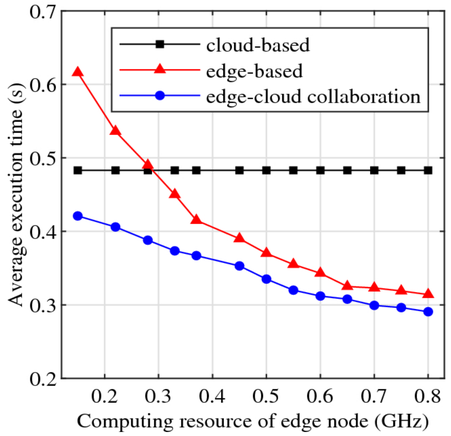
Aksara Bayyapu, Naman Gupta, Mihir Bala (Mentor)

Carnegie Mellon University Computer Science Department

Introduction

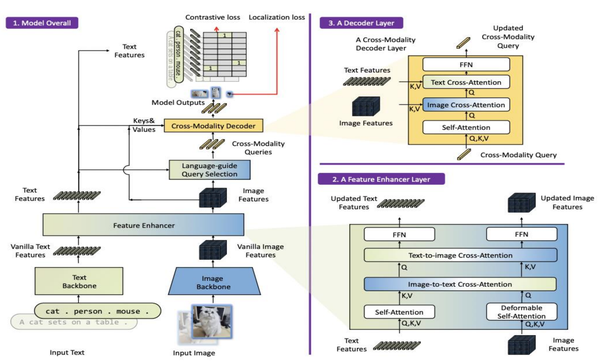
Edge computing minimizes latency by processing data locally on cloudlets, rather than relying on cloud servers, making it ideal for mobile applications. Cloudlet is a small-scale data center located at the edge of a network, designed to provide computing power and services directly to nearby devices like smartphones and wearables.

This project demonstrates the feasibility of running multi-modal deep learning models on edge devices to achieve near real-time inference for applications like zero-shot image captioning, object recognition, and image segmentation.



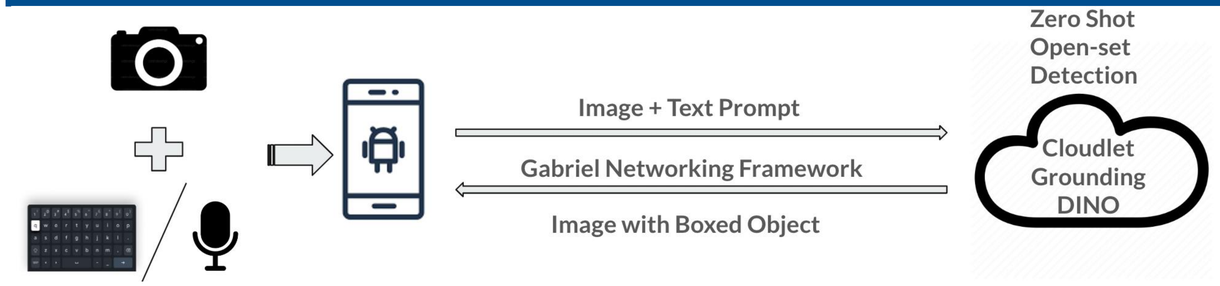
Existing Work

Grounding Dino: A state-of-the-art open-set, zero-shot object detection model that integrates language and vision modalities to detect arbitrary objects specified by human inputs. For this project we have used Grounding DINO running in the cloudlet for near real-time object detection. Grounding DINO framework:



Gabriel Platform: A framework designed for low-latency communication between mobile devices and nearby cloudlets. For this project we have leveraged Gabriel for communication between the Edge Device (android smartphone) and the cloudlet.

Design Architecture



We have developed an Android application that takes dual mode input. From the Camera we get the image/video feed (vision modality) and from the keyboard we get the user prompt (text modality). The application sends both the inputs (via Gabriel networking) to the cloudlet for further processing.

Our Cloudlet is a hosting a GPU (1080 Ti) for ML workload acceleration. The Grounding DINO model runs on the GPU cores to provide accelerated inferencing. The result of the model is the image/video annotated with box(es) around the object, wherever the object is present in the frame. The model understands what object to detect based on the text prompt from user.

This result is then relayed back to the handheld device (client), and the android application shows the result (frame with boxes around the objects) on the display.

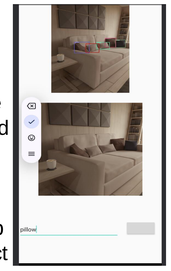
Since we are using Gabriel platform which is optimized for low-latency communication between mobile device and the cloudlet and with GPU acceleration we are getting a very low latency (~2 seconds) to see the result of this compute heavy operation.

This suggests that with the above hybrid Edge based approach we can have a good balance between latency and performance to see near real-time processing of compute heavy workloads, especially AI and ML inferencing.

Screenshots of using the application.

Text prompts given by the user are Window (left) and Pillow (right).

Original image is at the bottom, result is at the top with box around the object



Result

Process	Time (in ms)
Network Latency - RTT	200-300*
Processing Latency (images)	2000-2500*

The above table shows the time taken in two major components of executing a request in our application. The latency to get the result back to the client depends on network latency (round trip time) and the latency to process the request on the server. The server processing latency is measured using time profiling code in the server script, and RTT is calculated by subtracting total time (measured at client) with processing time. Image Processing time is ~90% of total latency and increases with video feeds.

Typical network latency using Gabriel is in the range of 80-150ms for most applications over Wi-Fi connections. The time taken can vary because of physical proximity to cloudlet, network congestion and whether the wireless medium is Wi-Fi or 5G/LTE.

The processing time on the cloudlet depends on the workload complexity, for example Lightweight tasks like symbolic representation extraction or guidance generation may take only a few milliseconds. However, computationally intensive tasks, such as deep neural network inference, can take several seconds. This time can be reduced by using a more powerful machine in the cloudlet which has dedicated hardware acceleration (GPUs).

Challenges and Learnings

We faced the following challenges during the project and most of them resulted in new learnings for us:

- **Learning Curve**: We had no prior experience with Android Development, or working with Deep Neural Network Models. We quickly ramped-up on Android development for the client side, Basics of Gabriel for networking and how to use Grounding DINO for the server side.
- **Limited Hardware**: Cloudlet GPU struggled to provide real-time inferencing even with single frame photo. For video processing the latency would not be close to near real-time, defeating the purpose of using the cloudlet and not offloading to the cloud.
- **Containerizing**: Learnt to containerize the server-side code using docker, so that if can be seamlessly migrated to any machine, potentially a higher performance machine to reduce the processing latency for video feeds. The challenge was to manually port multiple environment dependencies to the docker which are required to to execute the server-side code.
- **Publishing the App**: For developer using personal account, Google mandates a 2-week rigorous testing with 20 users, before the app can be made available to download. Logistics of working with 20 users is a challenge.

Future Work

A lot can be done to extend the work which we have done so far. If time permits, we would like to explore the following:

- We have added speech to text input functionality on android device to ease the user experience of providing text prompt instead of typing. We can further explore integration with voice assistants (e.g. Google Assistant) for hands-free operation where the app can be controlled using the assistant.
- Optimize server-side code deployment using containerization tools like Docker. We have already made progress towards this and currently in between the process of deploying to a bigger cloudlet server.
- Once the project is docked, we can Investigate hardware upgrades or alternative architectures (hybrid model of opportunistic offloading between cloudlet and cloud) for better performance with complex workloads.
- Publishing the Android app on play store, so that it can be downloaded and used by anyone with an Android device. This will help in extensive user testing. Our application is currently in the 2-week closed testing phase, as per the policies of Google Play Console.
- Finally, we will document all the progress we made and create our code's GitHub repository, so that it can be referred in future and the above tasks can be carried forward.

References

https://www.researchgate.net/figure/Latency-of-Cloud-and-Edge-computing-of-Application-and-database-server_fig3_344218742

https://arxiv.org/pdf/2303.05499

https://www.cs.cmu.edu/~satya/docdir/chen-sec2017.pdf

https://github.com/IDEA-Research/GroundingDINO

https://github.com/cmusatyalab/gabriel/tree/master/examples/round_trip

https://github.com/cmusatyalab/gabriel

https://play.google.com/console/about/publishingoverview/

https://developer.android.com/courses

Aksara Bayyapu - abayyapu@andrew.cmu.edu
Naman Gupta - namang2@andrew.cmu.edu