Real-Time Retail Product Classification on Android Devices inside the Caper AI Cart

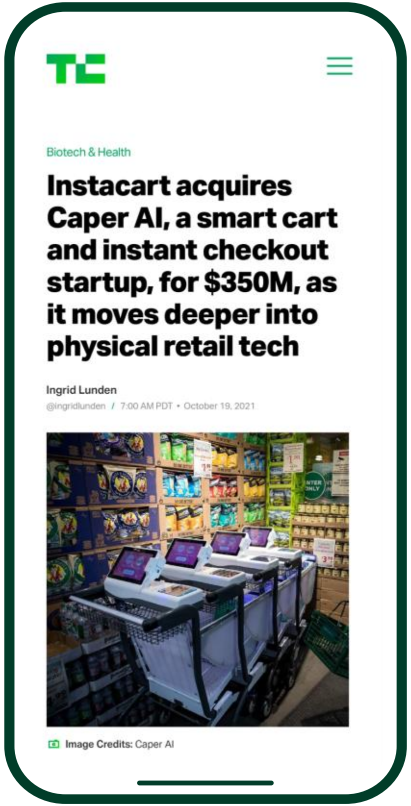
David Scott

Senior Machine Learning Engineer

Instacart (Caper)

Table of Contents

Caper @ Instacart



Biotech & Health

Instacart acquires Caper AI, a smart cart and instant checkout startup, for $350M, as it moves deeper into physical retail tech

Ingrid Lunden
@ingridlunden / 7:00 AM PDT • October 19, 2021

Image Credits: Caper AI

Bloomberg Subscribe

Technology | Deals

Instacart to Buy Smart Grocery Cart Startup in Biggest Deal Yet

- $350 million deal for Caper AI aims to expand store offerings
- CEO says Instacart will be 'more focused' on growth before IPO

Caper AI grocery carts. *Source: Caper AI*

By Jackie Davalos +Follow
October 19, 2021, 8:00 AM MDT

Instacart Inc. is acquiring Caper AI, a startup that makes self-checkout shopping carts, for $350 million in cash and stock in an effort to tap more areas of growth ahead of the online

A Seamless Shopping Experience

embedded VISION SUMMIT®

Caper Cart

1. **Sensor Fusion & AI Integrated**
   Providing customers with a more convenient way to shop

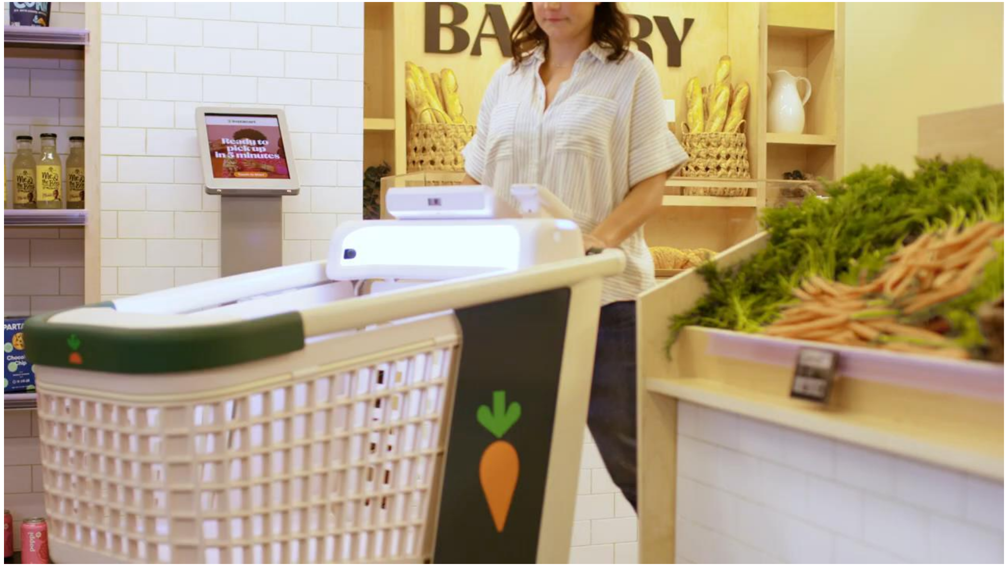
2. **Weights & Measures Enabled**
   Robust and certified

3. **Embedded Location Systems**
   Streamlining the shopper experience

instacart

Computer Vision and AI @ Caper

In order to unlock a magical shopping experience, we enable seamless AI that detects and recognizes products in less than 0.5 seconds



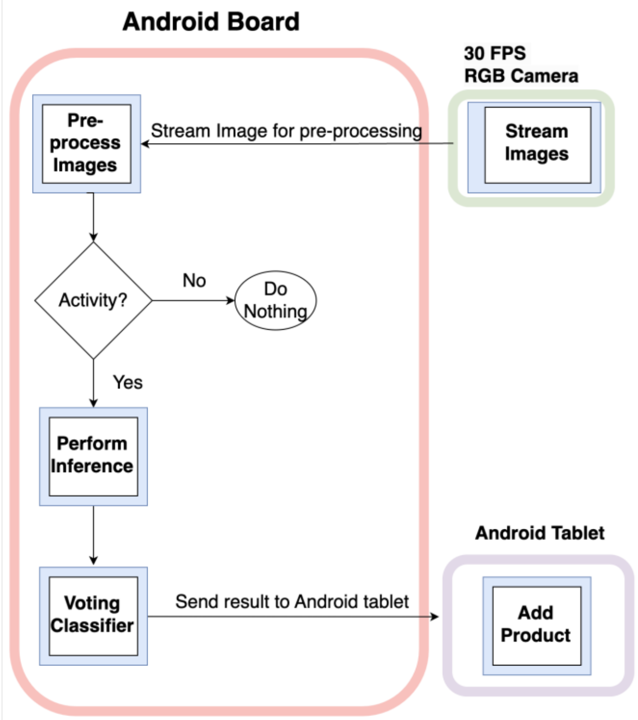
Current Challenges for Such Products

1. **High-throughput model with high accuracy**
   Ensure smooth and efficient detection
   Avoid delays or disruptions in user experience

2. **Limited by system constraints**
   Limited CPU resources shared by Android services
   Model speed can impact overall app degradation

3. **User experience**
   How can we utilize the hardware + software resources we have to improve our user experience without requiring re-educating our customer?

Proposed Approach



1. **Generate image stream and process directly on Android board**
   Utilize 30 FPS red-green-blue (RGB) camera to stream images for processing
2. **Skip processing images on Jetson GPU**
   Utilize the processing power of our Android board to stream and process images directly on Android
3. **Deploy using TensorFlow Lite (TFLite)**
   Train and convert our PyTorch models to TFLite for model inference

Proposed Approach

1. **User adds product under our camera**
   Get a stream of images that are background removed
2. **Perform inference on each image**
   Android board takes in images as an input and returns embeddings
3. **Utilize a voting classifier with embeddings**
   - Generate one set of embeddings per frame from a single network
   - Utilize ground truth embeddings to calculate highest similarity in class

instacart

Approach to Reduce Resource Usage

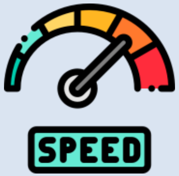
Approach to Reduce Resource Usage: Camera

Frame skipping

Do we have an accuracy reduction or lose important information by skipping frames?

Re-using circular buffers

Other services utilize the same camera;
we can re-use buffers



Reducing the resolution of incoming image stream

Since our model utilizes 224 [w, h] images, is it possible to modify the camera firmware to give us smaller images?

Approach to Reduce Resource Usage: System

Threading

How many threads is necessary to maximize the performance of the system?

Perfetto traces

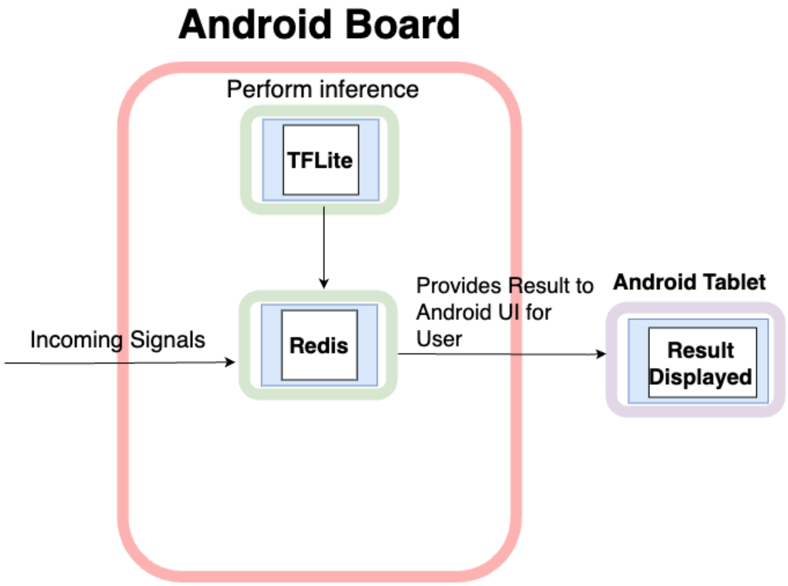
Utilized to help pinpoint heavy system resource usages on CPU and RAM

Reducing embeddings dimensions

By decreasing the dimensions and # of our output embeddings, we reduced system usage

Approach to Integrate with Android System

Approach to Integrate with Android System



1. **Utilizing co-routines and state machine learning model**
   Break thread if issue arises, avoiding impact on other apps
2. **Incoming signals via Redis**
   Combine information across services to give best result
3. **TFLite integration**
   - Feed model and perform embedding similarity
   - Utilize TFLite built in optimizations for on-device inference

instacart

Results and Next Steps

Android Performance Metrics

CPU usage

Get the best performance while minimizing CPU usage for other Apps

Model throughput

How can we optimize model throughput while maintaining precision and recall?

Camera speed

Higher FPS gets us better recognition

More FPS = More CPU; how to balance?

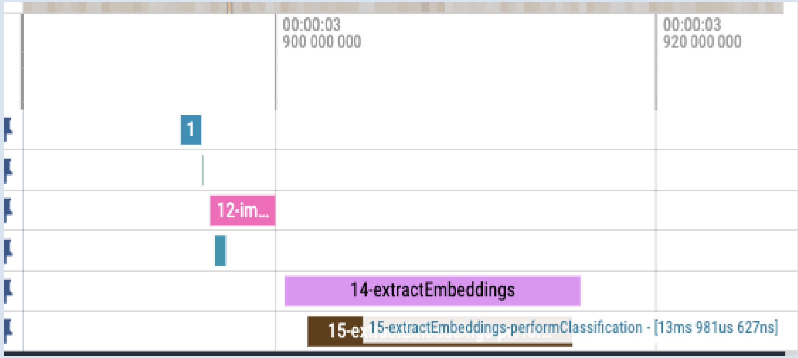
Android Performance Metrics — Model Throughput

Model throughput

We optimized our throughput by the following:

- Using INT8 instead of FLOAT32
- Utilizing 112x112 to train model instead of 224x224
- Decreasing output embedding size

Increase in speed of ~96% from 400ms!



Android Performance Metrics — Camera FPS

embedded VISION SUMMIT

Camera FPS

We optimized our image resolution + camera FPS by

- Using 640P images @ 30 FPS
- Re-using existing circular buffers from another service
- Worked with camera vendor to get custom firmware

Throughput + resolution investigation

```
sorted supported format(MJPG) w*h(320x240)@30.000000 fps
sorted supported format(MJPG) w*h(352x288)@30.000000 fps
sorted supported format(YUYV) w*h(640x480)@30.000000 fps
sorted supported format(MJPG) w*h(800x600)@30.000000 fps
sorted supported format(MJPG) w*h(1280x720)@30.000000 fps
sorted supported format(MJPG) w*h(1280x960)@30.000000 fps
sorted supported format(MJPG) w*h(1920x1080)@30.000000 fps
sorted supported format(MJPG) w*h(1600x1200)@15.000000 fps
sorted supported format(MJPG) w*h(2048x1536)@15.000000 fps
sorted supported format(MJPG) w*h(2592x1944)@15.000000 fps
sorted supported format(MJPG) w*h(2592x1944)@15.000000 fps
```

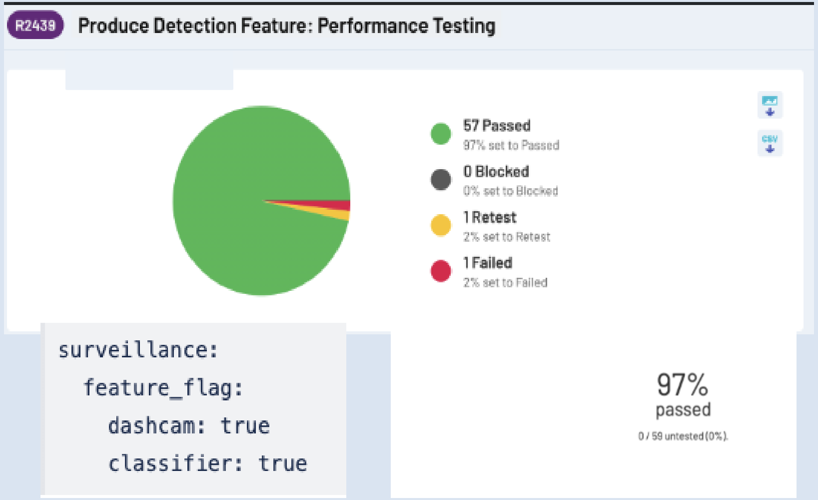
Android Performance Metrics — CPU Usage

CPU usage

We optimized CPU usage by optimizing our model & camera

- Skip intermediate steps, like image resizing
- Feed system with a queue of images
- Stress test across our system to ensure performance

Feature Testing from QA



Next Steps

1. **Further optimizations**
   Continuous improvement of model accuracy and speed

2. **Improving TFLite ecosystem and documentation**
   Contribute our learnings and findings to the sparse documentation for TFLite.

3. **Utilizing research to enhance user experience**
   Provide a seamless and delightful experience for users

Conclusions

1. **Solve your problems first with hardware**
   Optimizing hardware selection is a great place to start before software
2. **Optimizing throughput leads to better overall performance**
   Gains in throughput speed helped decrease CPU usage (through frame skipping)
3. **Deeply investigating pre-processing and model steps to maximize throughput**
   Spending time investigating pre-processing and model optimization was fruitful for the team

Resources

TensorFlow Lite Documentation:

- TensorFlow Lite Model Optimization Documentation
- TensorFlow Lite Quantization Documentation
- TensorFlow Lite Model Analyzer Documentation

Android Development for TensorFlow Lite:

- Quickstart for Android Development

Thank you for your time!
Look forward to answering your questions

David Scott

Senior Machine Learning Engineer

Instacart (Caper)

https://www.linkedin.com/in/davejscott/