Energy-efficient Cross-camera Communication for Classification with Overlapping Field-of-View

ECE M202A: Embedded Systems, Fall 2021
Final Project Presentation
https://zutierrez.github.io/ecem202a_project/report

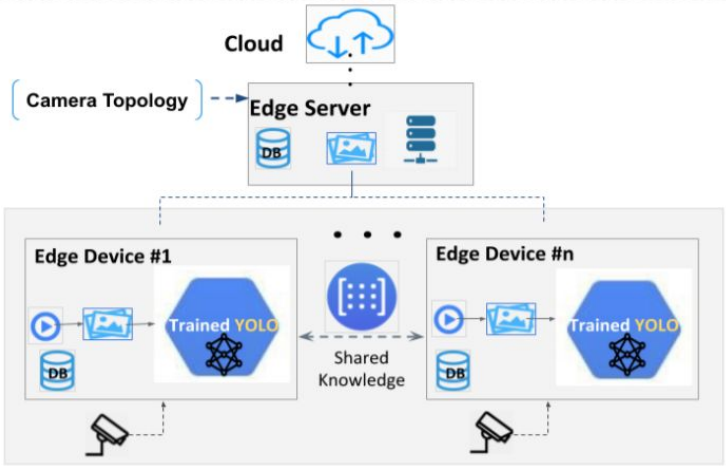
Presenters:
Zion Gutierrez (zutierrez@g.ucla.edu)
Sung Yoon Jung (sungyoon@g.ucla.edu)

UCLA Samueli School of Engineering

State-of-the-Art and Prior Research

CONVINCE: Collaborative Cross-Camera Video Analytics at the Edge

1) Collective entity of cameras capture takes advantage of **spatio-temporal** correlations
2) **Knowledge sharing** to conserve power

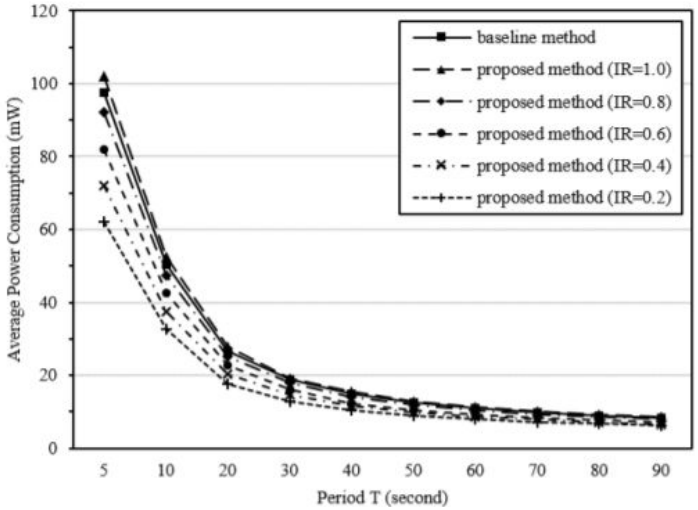


CONVINCE Framework Overview
(H. B. Pasandi and T. Nadeem)

State-of-the-Art and Prior Research

CamThings: IoT Camera with Energy-Efficient Communication by Edge Computing based on Deep Learning

1) Only **images of interest** are transmitted to Cloud
2) **Frequency** of image capture is adjusted
3) Edge computing based on **DNNs**



Improved Power Consumption via CamThings Framework
(J. Lim, J. Seo, and Y. Baek)

Overall Project Goals and Specific Aims

To optimize ***power consumption*** of multi-camera image classification using cross-camera communication.

Strategies:

1) Reducing the rate of capture
2) Transmitting only images of interest
3) Preventing redundant image transmission

Accomplishments

- Deployed machine learning model on edge-computing device
  - Arduino classified images using MobileNet V1 and TensorFlow Lite
- Arduino-Arduino cross-communication
  - Transmitted/received classified object labels between devices
- Arduino-to-WiFi-device image transmission
  - Transmitted images from the Arduino to the host device (computer)
- Analysis of power consumption for different scenarios
  - Found voltage and current draw of Arduino for multiple cases

Image Collecting and Model Training

Datasets: 128x128 images of Faces (Flickr-Faces-HQ) and other miscellaneous objects (Linnaeus 5)

Image Processing: 96x96 crop and data augmentation

128x128 image 96x96 image

MobileNet V1 0.1: 53.2K RAM and 101K ROM (default settings and optimizations)

87% accuracy on test set, **20** training cycles, 1,000 human samples, 1,000 unknown → Transfer Learning

- **Speed:** inference time < 300 ms on average
- **Low Compute:** Runs on Arduino Nano 33 BLE Sense with 256KB (nRF52840)
- **Compatibility with Arduino Nano:** lightweight library for edge devices

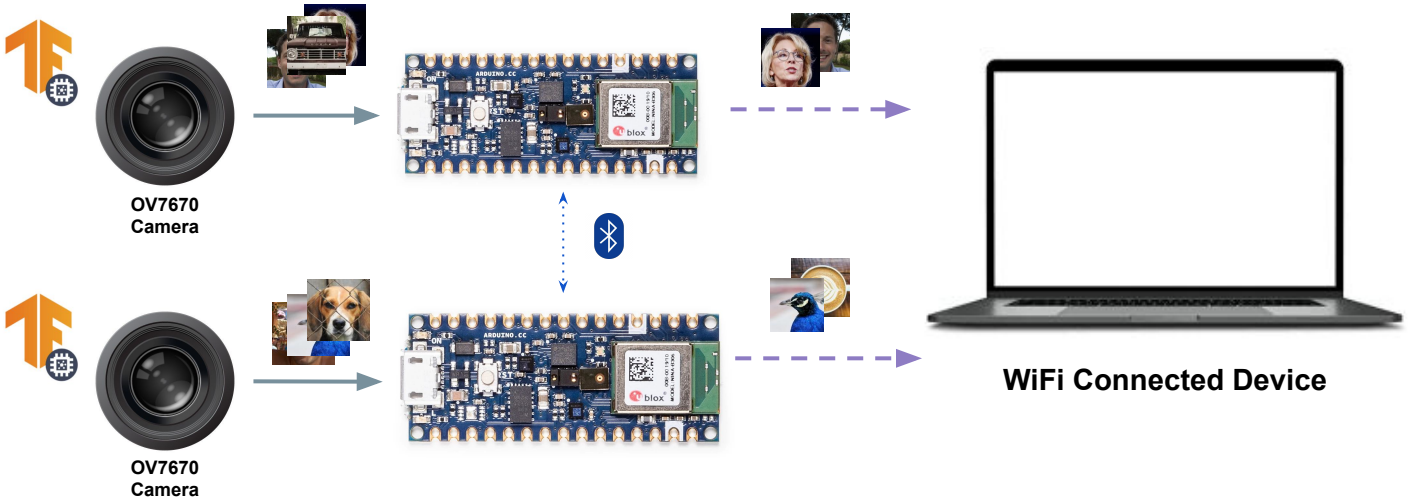
Cross-Camera Communication via BLE

- BLE 5 Characteristics and Constraints:
  - BLE 5 has practical net data rate of around 1.4 Mbps
  - Central Peripheral Model for Data Sharing
  - Constant advertisement of data over connection requires additional power
  - Images (280x320) are too large for BLE so label are transmitted in their place

- Arduino-Arduino communication
  - Central device sends classified label (integer value) to the peripheral device, which can prevent redundant transmissions to cloud

Hardware System Overview:

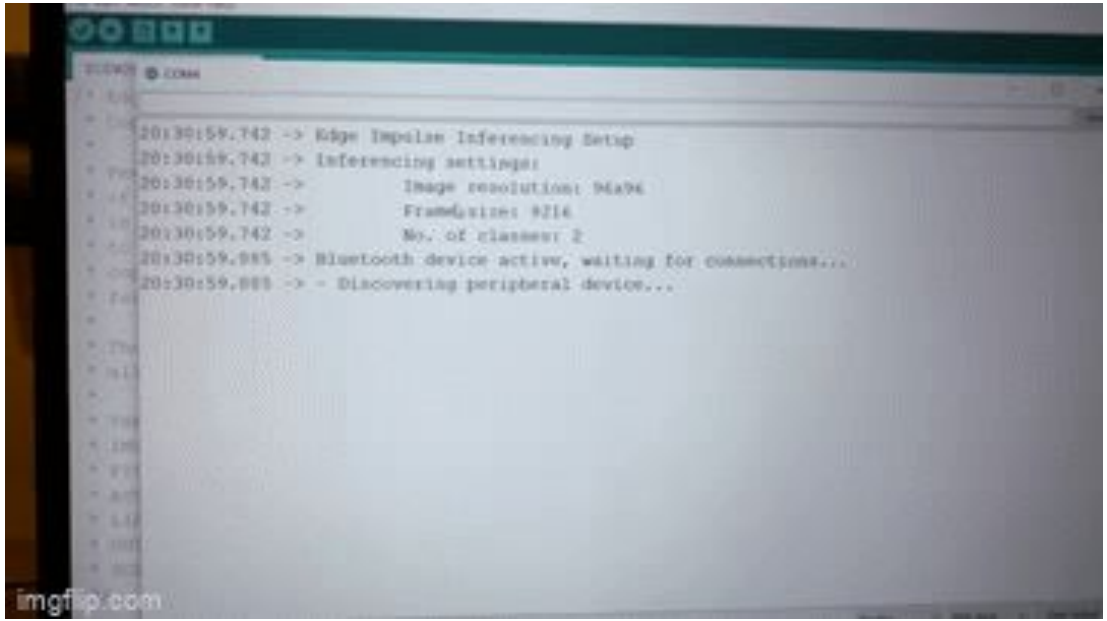
Arducam OV7670, 0.3 MP Camera Module
Arduino Nano 33 BLE Sense



Classification and Communication

Model Performance: 51 ms inference, 66.1k Peak RAM, 107.7K Flash

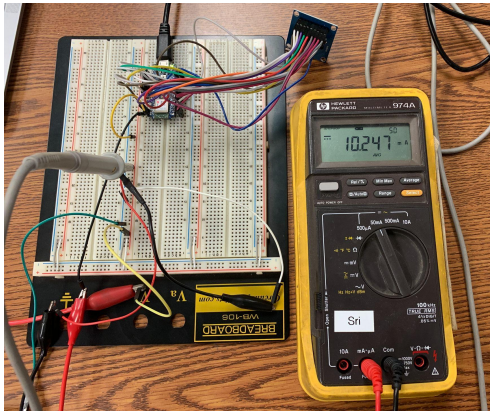
BLE Communication: Arduino 0.314 ms Inference and Label Sharing

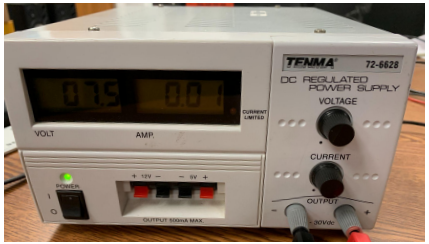


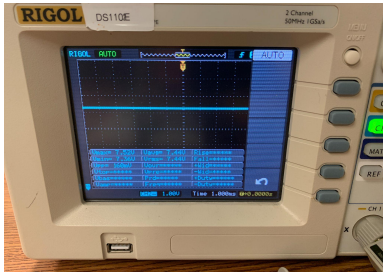
BLE Communication Demonstration: Arduino Nano 33 BLE Sense Central Device advertises label to Peripheral Device

Evaluation Set Up

- TENMA Power Supply
  - V = 7.5V (supply > 5.0V selected to override USB voltage)
  - USB current contamination
- RIGOL DS1102E Oscilloscope
  - Used to measure voltage
- HP974A Multimeter
  - Inserted in series to measure current
- 4 Evaluation Cases
  - Base case
  - Classification only
  - Classification with Bluetooth
  - Image transmission







Lab Instruments for Power Analysis:
Evaluation Setup and Methods

Evaluation and Results (Case 0)

“Reset” Arduino Program

- Average Current = 9.0mA

“Blink” Arduino Program

- Without USB: 17.38mA / 19.88 mA
- With USB: 17.36 mA / 19.83 mA

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Code Snippet: Arduino Blink Program for Base Power Consumption

→ USB current contamination is negligible (< 0.1 mA)

Evaluation and Results (Case 1)

Image Classification Model Program

- Instantaneous Current:
  - **Task Flow:** Taking photo (2s) → inference (0.3s) → Delay (2s)
  - 26.8mA → 27.8mA → 24.3mA
- Average Current = ~26 mA

```
20:51:28.054 -> Inferencing settings:
20:51:28.054 ->         Image resolution: 96x96
20:51:28.054 ->         Frame size: 9216
20:51:28.054 ->         No. of classes: 2
20:51:28.054 ->
20:51:28.054 -> Starting inferencing in 2 seconds...
20:51:30.057 -> Taking photo...
20:51:33.241 -> Predictions (DSP: 10 ms., Classification: 328 ms., Anomaly: 0 ms.):
20:51:33.241 ->     person:     0.000000
20:51:33.241 ->     unknown:    0.996094
```

Evaluation and Results (Case 2)

Classification + BLE Communication Program

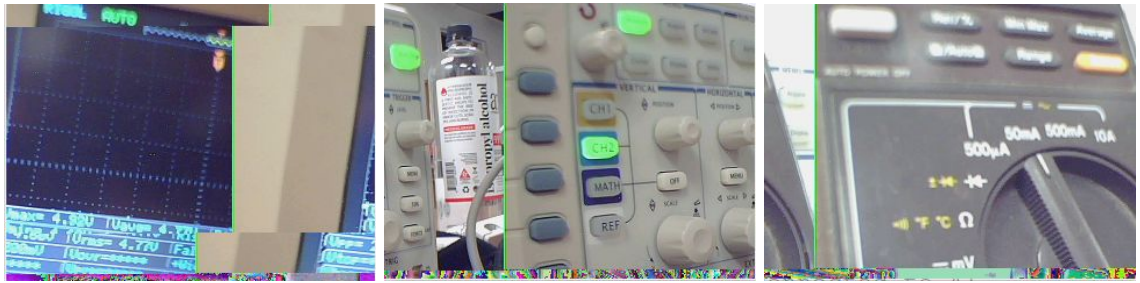
- Instantaneous Current:
  - **Task flow:** searching → connect → delay → take photo/inference → disconnect/reconnect
  - 13.3 mA → 10.3 mA → 28-29 mA → 31-32 mA → 20-21 mA
- Average Current:
  - **Before BT connection:** 13.3 mA
  - **After BT connection:** 28 mA

```
20:31:39.343 -> - Connecting to peripheral device...
20:31:39.416 -> * Connected to peripheral device!
20:31:39.416 ->
20:31:39.416 -> - Discovering peripheral device attributes...
20:31:40.658 -> * Peripheral device attributes discovered!
20:31:40.658 ->
20:31:40.658 -> * Device connected and starting inference!!
20:31:40.658 ->
20:31:40.658 ->
20:31:40.658 -> Starting inferencing in 2 seconds...
20:31:42.641 -> Taking photo...
20:31:46.214 -> Predictions (DSP: 18 ms., Classification: 584 ms., Anomaly: 0 ms.):
20:31:46.214 ->     person:     0.000000
20:31:46.214 ->     unknown:    0.996094
20:31:46.214 -> Choosing Label... - UNKNOWN detected
20:31:46.214 -> * Writing value to eventImage characteristic: 0
20:31:46.253 -> * Writing value to gesture_type characteristic done!
20:31:46.253 ->
20:31:46.253 -> * Device connected and starting inference!!
```

Evaluation and Results (Case 3)

Images Transmission from Arduino to WiFi Device Program

- Average Current:
  - **Idle**: 25 mA
  - **Take photo:** 29 mA
  - **Transmit photo binary:** 27 mA



Final Results: Power Consumption Summary

- V_avg consistent in all cases
  - V_avg = 7.44V (over 10 ms), Vpk-pk = 120-160 mV
- USB current contamination is negligible

Case	Task	Min Current	Max Current	Avg. Current	Avg. Power
0	Reset	-	-	9 mA	**68 mW**
1	Classification	24.3 mA	27.8 mA	26 mA	**195 mW**
2	Classification + BLE	10 mA	32 mA	28 mA	**210 mW**
3	Image Transmission	-	-	27 mA	**203 mW**

Conclusions and Discussion

- Summary: Given a 50% redundancy rate, only 0.74% reduction in power consumption
  - **Total energy per frame = 35.9 mAh**
  - **Energy saved per frame = 0.267 mAh**
- This method does not significantly reduce power consumption in a 2-camera system executing on-board image classification and BLE communication in real time
- Reducing image transmissions by eliminating redundancies does not yield significant energy savings such that it compensates for power consumption of BLE communication
  - **BLE energy per frame = 2.58 mAh**
- Image classification & image transmission have similar power consumption (27-28 mA)
- Taking photos results in highest current spike (31-32 mA)

Challenges

Arduino Nano 33 BLE Sense Limitations

- **Low memory:** an lightweight model was necessary to avoid reaching memory capacity
- **Low processing power:** slow image capture and transmission during runtime (1 second)
- **Compilation speed:** large file took several minutes

OV7670 Camera Limitations

- **Low image quality:** tradeoffs made to switch to lower resolution camera decreased inference accuracy during execution at the edge
- **Unreliable connection:** images often became corrupted after several successful images. One camera produced constantly blurry images.

BLE Communication Issues

- **Unexpected disconnection:** Arduinos successfully connected but frequently lost connection due to memory constraints of running image classification model in conjunction with constant BLE advertising
- **Low throughput:** BLE 5 cannot send full image data in one transmission, images had to be split up into packets

Improvements & Future Work

Improvements:

- Utilize portable WiFi device (such as RPi) instead of computer as fog layer device
- Train model for finer granularity of on-device image classification (i.e. black dog vs. white dog)
- Improve latency and throughput by optimizing neural network architecture while maintaining low power consumption

Future Work:

- Develop overlapping field-of-view algorithm
- Analyze power consumption with portable power supply (battery) in real-life case study

References

1. H. B. Pasandi and T. Nadeem, “CONVINCE: Collaborative Cross-Camera Video Analytics at the Edge,” 2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), 2020, pp. 1-5. *https://ieeexplore.ieee.org/document/9156251*
2. J. Lim, J. Seo and Y. Baek, “CamThings: IoT Camera with Energy-Efficient Communication by Edge Computing based on Deep Learning,” 2018 28th International Telecommunication Networks and Applications Conference (ITNAC), 2018, pp. 1-6. *https://ieeexplore.ieee.org/document/8615368*
3. Zicheng Chi, Yan Li, Xin Liu, Yao Yao, Yanchao Zhang, and Ting Zhu. 2019. Parallel inclusive communication for connecting heterogeneous IoT devices at the edge. In Proceedings of the 17th Conference on Embedded Networked Sensor Systems (SenSys ‘19). Association for Computing Machinery, New York, NY, USA, 205–218. *https://doi.org/10.1145/3356250.3360046*
4. Linnaeus 5 Dataset: *http://chaladze.com/l5/*
5. Flickr Faces HQ Dataset (FFHQ): *https://github.com/NVlabs/ffhq-dataset*
6. Arduino IDE: *https://www.arduino.cc/en/software*
7. Edge Impulse CLI: *https://docs.edgeimpulse.com/docs/cli-installation*
8. ArduCAM Repository: *https://github.com/ArduCAM/Arduino*

Energy-efficient Cross-camera Communication for Classification with Overlapping Field-of-View

ECE M202A: Embedded Systems, Fall 2021
Final Project Presentation
https://zutierrez.github.io/ecem202a_project/report

Presenters:
Zion Gutierrez (zutierrez@g.ucla.edu)
Sung Yoon Jung (sungyoon@g.ucla.edu)

UCLA Samueli School of Engineering