Machine Learning based Real Time UAV Detection using Smartphone

Swatter

Joonki Rhee

Gwangwon Kim

Hyunjong Jang

Minseop Shin

Justin Allange

PURDUE UNIVERSITY | Polytechnic Institute

Contents

Introduction

Real-time UAV Detection using Smartphones

Introduction

Team Introduction

Joonki Rhee

Computer Science

Kyonggi Univ.

Team Leader

Server Developer

Gwangwon Kim

Industrial Engineering, Data Engineering

Kyonggi Univ.

Research

Main AI Developer

Hyunjong Jang

Software Engineering

Dongseo Univ.

Research

Writing Paper

Minseop Shin

Software Engineering

Dongseo Univ.

Research

App Developer

Justin Allange

Computer and Information Technology

Purdue University

Research & App Developer

Introduction

Acronyms

- **ML : M**achine **L**earning
- **CNN : C**onvolutional **N**eural **N**etworks
- **RF : R**adio **F**requency
- **YOLO : Y**ou **O**nly **L**ook **O**nce
- **MFCC : M**el **F**requency **C**epstral **C**oefficient
- **NN : N**eural **N**etwork
- **GNB : G**aussian **N**aïve **B**ayes algorithm
- **KNN : K N**earest **N**eighbor algorithm
- **SVM : S**upport **V**ector **M**achine algorithm
- **ReLU : Re**ctified **L**inear **U**nit
- **TCP: T**ransfer **C**ontrol **P**rotocol
- **UAV : U**nmanned **A**erial **V**ehicles
- **CAGR : C**ompound **A**nnual **G**rowth **R**ate

Introduction

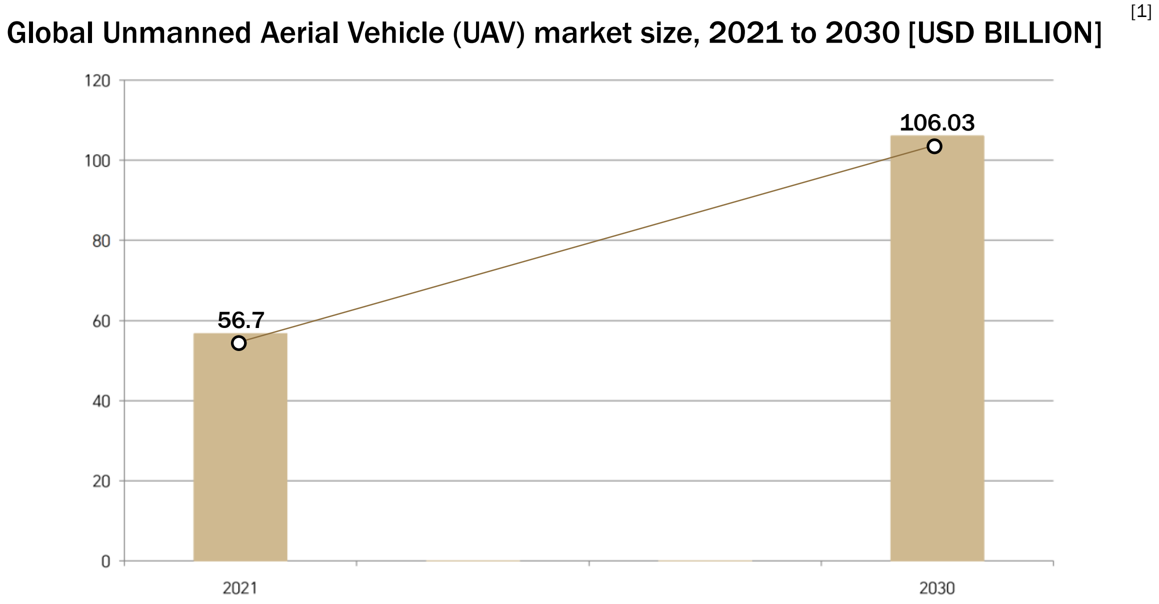
Project Motivation

Drone for delivery

Drone for photography

Introduction

Project Motivation



[1] B. Aamir, "Unmanned Aerial Vehicle (UAV) market are expected to reach US$ 106.03 billion by 2030 - astute analytica," GlobeNewswire News Room, 17-Nov-2022. [Online]. Available: https://www.globenewswire.com/en/news-release/2022/11/17/2558200/0/en/Unmanned-Aerial-Vehicle-UAV-Market-to-Reach-US-106-03-Billion-by-2030-Astute-Analytica.

Introduction

Project Motivation

Drone with bomb

Drone with gun

Introduction

Project Motivation

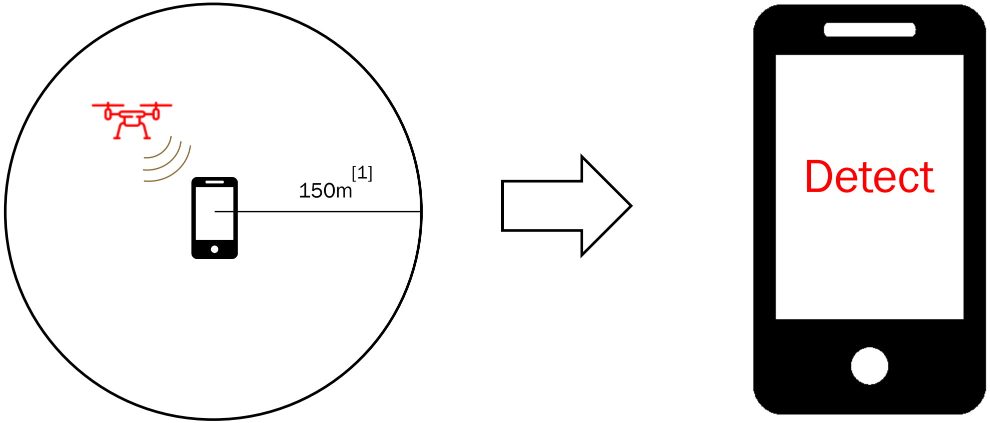
Drone assassination attempt on President

Drone attempts to smuggle out of prison

Introduction

Project Goal

- Real Time UAV Detection with Smartphone



[1] B. Taha and A. Shoufan, "Machine Learning-Based Drone Detection and Classification: State-of-the-Art in Research," in IEEE Access, vol. 7, pp. 138669-138682, 2019, doi: 10.1109/ACCESS.2019.2942944.

RELATED WORKS

Real-time UAV Detection using Smartphones

RELATED WORKS

Relevant literature for detecting Unmanned Aerial Systems

- **Vision based** UAV detection - B.-G. Han el al[1]. YOLO algorithm employs convolutional neural networks (CNN) to detect objects in real-time. This solution requires huge labeled datasets and have problem with visual data's noise.

- **Radar based** - B Torvik et al. [2] proposed 100% accuracy results by simple nearest neighbor approach for binary classification between UAVs and birds. However, this method has Radar Cross-Section and range limitation[3].

- **Radio frequency**-based UAV detection use RF signals from controller and achieved 80~95% accuracy with ML learning techniques. However, this solution fails when the drone is operated in autonomous mode[2].

[1] B. –G. Han et al. "Eesign of a Scalable and Fast YOLO for Edge –Computing Devices", Sensors, Bvol. 20, no. 23, 2020.
[2] B. Torvik, K. E. Olsen and H. Griffiths, "Classification of birds and uavs based on radar polarimetry", *IEEE geoscience and remote sensing letters* , vol. 13, no. 9, pp. 1305-1309, 2016.
[3] B. Taha and A. Shoufan, "Machine learning-based drone detection and classification: State-of-the-art in research", *IEEE Access*, vol. 7, pp. 138669-138682, 2019.

RELATED WORKS

Acoustic node for detecting Unmanned Aerial Systems

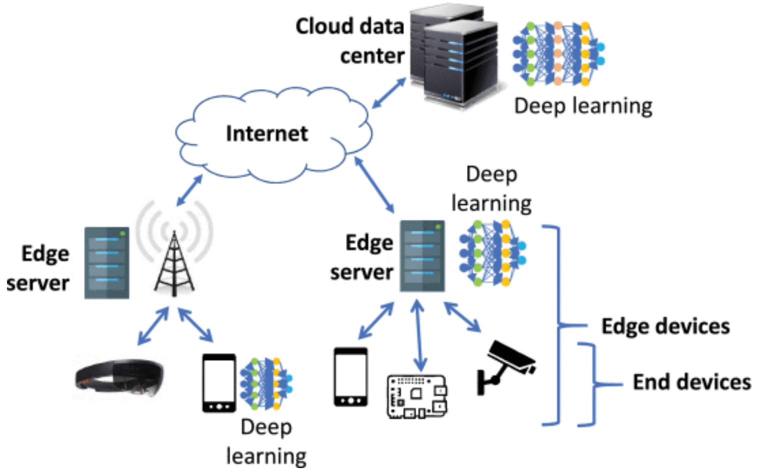
- The rotation of the drone’s **rotor blades produces a humming sound** that can be sensed and recorded, even within the range of human hearing.
- Y Wang *et al.*[1] Machine learning algorithm and **MFCC** were applied to detect UAVs.
- **Using features for classification** provide explanations for understanding how the ML classification was produced.
- This solution provided 78% accuracy and fewer computational resources than others.

[1] Y. Wang et al. "A Feature Engineering Focused System for Acoustic UAV Detection", 2021 Fifth IEEE International Conference on Robotic Computing (IRC), 2022.

RELATED WORKS

Acoustic UAV detection on Edge device

- 3 Challenges : Latency, Scalability, and Privacy [1]
- TensorFlow Lite was proposed for mobile and embedded devices, with mobile GPU support.
- In contrast to cloud computing, edge computing's latency is significantly lower, as large quantities of data do not have to travel through a backhaul network to the cloud[2].



[Deep learning on edge devices and cloud data centers]

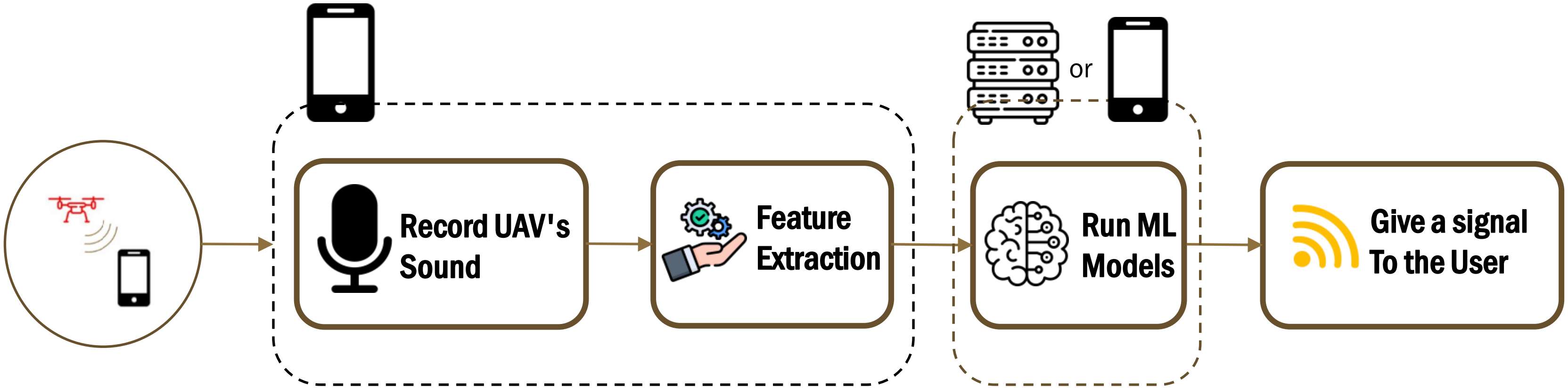
[1, Fig 1] J. Chen and X. Ran, "Deep learning with edge computing: A review", Proc. IEEE, vol. 107, no. 8, pp. 1655-1674, Jul. 2019.
[2] P. Joshi et al. "Enabling All In-Edge Deep Learning: A Literature Review" 2023, IEEE Access (Volume: 11)

Methodology

Real-time UAV Detection using Smartphones

Methodology

Overview of two methods for drone detection



Methodology

2.1 Drone Types

- **Autel Evo 2**
  - Weight : 1192 g
  - Max Speed : 20 m/s

- **DJI Phantom 4**
  - Weight : 1380 g
  - Max Speed : 20 m/s

Methodology

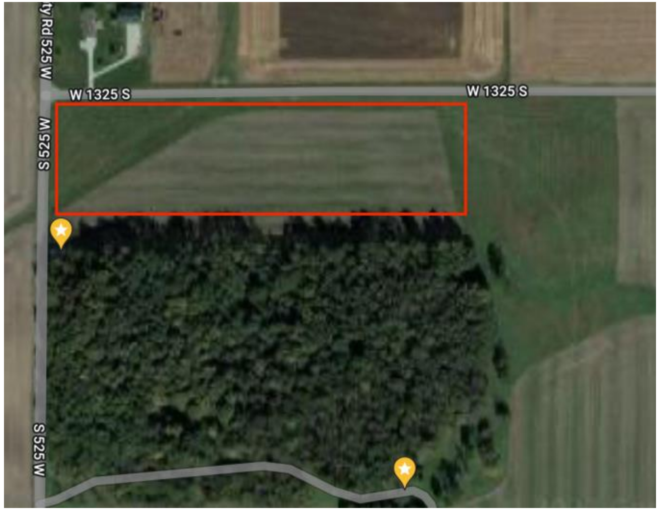
Location of data collection

- Location

  - New Richmond, Indiana,
    (40.2227062, -87.0000169)

- Time

  - February 13, 06:00 A.M



Methodology

Sound data samples

Audio Type	Number of Samples	Total Time(sec)
UAV	200	500
Noise	200	500
Total	400	1000

5 seconds per each sample

Methodology

Feature extraction

TABLE II
DRONE ACOUSTIC FEATURE EXTRACTION METHOD

Feature	Shape
chroma_stft	12
chroma_cqt	12
chroma_vqt	12
mel	128
mfcc	40
rms	1
centroid	1
bandwidth	1
contrast	7
flatness	1
bandwidth	1
rolloff	1
poly_shape	2
tonnetz	6
zero_crossing	1

- Feature extraction methods

⇒ **Python library, Librosa**

→ Models can understand the sound through Feature extraction

Methodology

Selecting ML Algorithm

- Machine Learning is our main method of UAV detection.
- Using ML algorithms is suitable for audio classification.
- In another paper[1], 4 algorithms were selected.

Name of Algorithm	Hyperparameters
NN (**N**eural **N**etwork)	Learning rate = 0.001, epochs = 15
GNB (**G**aussian **N**aïve **B**ayes)	Default
KNN (**K**-**N**earest **N**eighbor)	N_neighbors = 6, others are default
SVM (**S**upport **V**ector **M**achine)	C = 10, kernel as linear

[1] Y. Wang et al, "A Feature Engineering Focused System for Acoustic UAV Detection," *2021 Fifth IEEE International Conference on Robotic Computing (IRC)*, Taichung, Taiwan, 2021, pp. 125-130

Methodology

Server Development

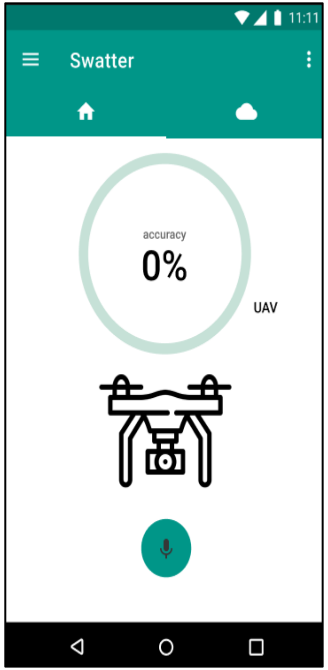
CentOS 7 | C++ 17 | Python/C API

1. Install CentOS 7 on VMWare 17
2. Implement server program
3. Embed Python inside a C++ program
4. Use Transmission Control Protocol (TCP)

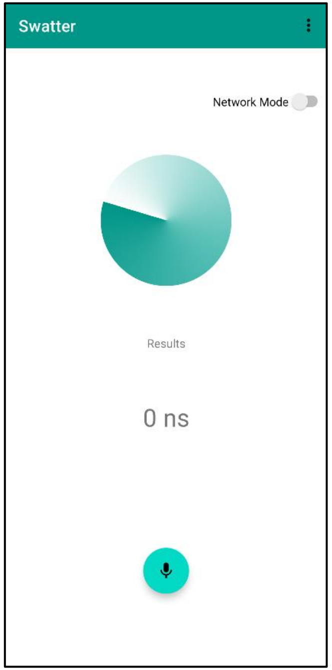
Methodology

Application Development

- Model converted using TensorFlow lite
- Program specifications
  - Kotlin (java), Android API 27
- Three main functions
  - Record sound
  - Draw result of UAV Inference
  - Transmit sound data to cloud server



[UI before Develop]



[UI after Develop]

Progress

Real-time UAV Detection using Smartphones

Progress

Verifying Feature Extraction

- Feature extraction using sample data
  - Librosa built-in data is used
- Verified that shapes are matched with previous paper[1]

```python
## Verify shape of features
df = pd.DataFrame(index = ['MFCC','mel','chroma_stft','contrast','tonnetz']
                  ,columns = ['Shape'])
#df.index.name = 'Features'

## sample data
y, sr = librosa.load(librosa.ex('trumpet'))
print(y)

## MFCC
mfcc = librosa.feature.mfcc(y, sr=sr)
mfcc = np.mean(mfcc.T,axis=0)
# print(mfcc)
df.iloc[0] = mfcc.shape[0]

## mel
mel = librosa.feature.melspectrogram(y,sr=sr)
mel = np.mean(mel.T, axis=0)
df.iloc[1] = mel.shape[0]

## chroma_stft
chroma_stft = librosa.feature.chroma_stft(y,sr)
chroma_stft = np.mean(chroma_stft.T, axis=0)
df.iloc[2] = chroma_stft.shape[0]

## contrast
stft = np.abs(librosa.stft(y))
contrast = librosa.feature.spectral_contrast(S=stft,sr=sr)
contrast = np.mean(contrast.T, axis=0)
df.iloc[3] = contrast.shape[0]

## tonnetz
tonnetz = librosa.feature.tonnetz(y=librosa.effects.harmonic(y),sr=sr)
tonnetz = np.mean(tonnetz.T, axis=0)
df.iloc[4] = tonnetz.shape[0]

print(df)
```

[Program for verifying feature extraction]

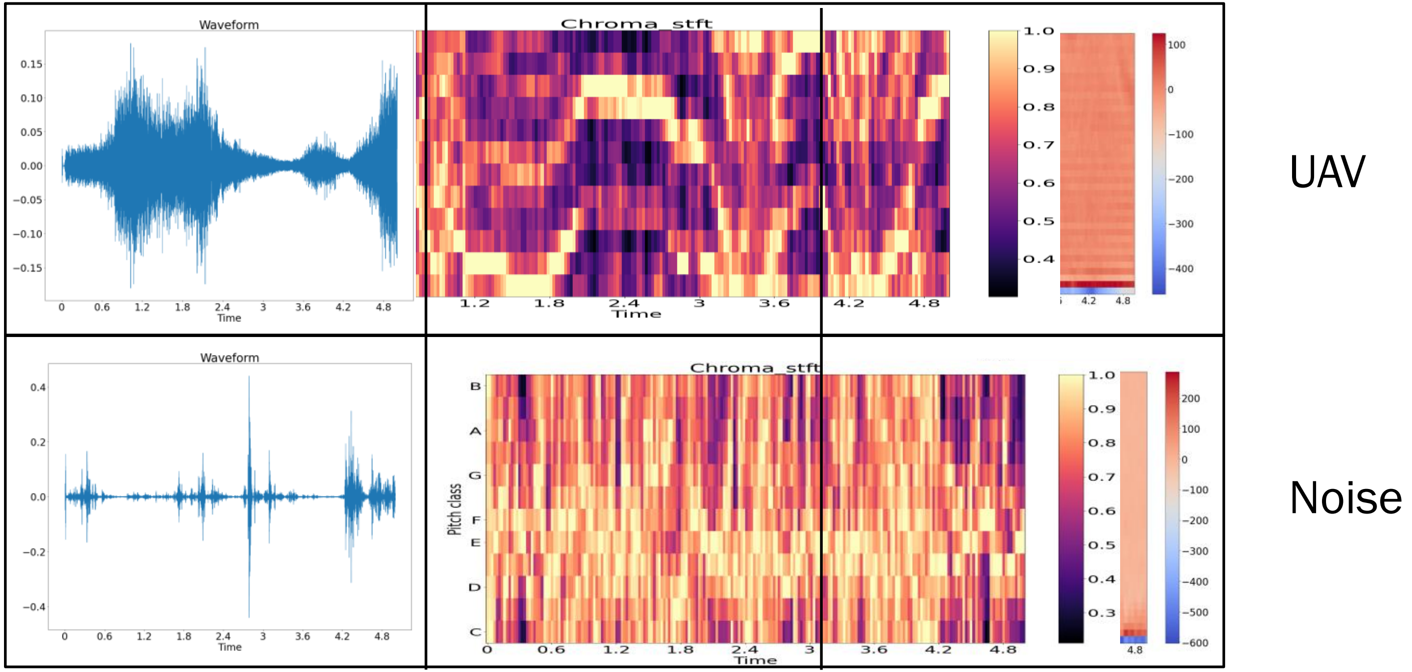
	Shape
MFCC	40
mel	128
chroma_stft	12
contrast	7
tonnetz	6

[Results of program]

[1] Y. Wang et al, "A Feature Engineering Focused System for Acoustic UAV Detection," *2021 Fifth IEEE International Conference on Robotic Computing (IRC)*, Taichung, Taiwan, 2021, pp. 125-130

Progress

Feature Visualization



Progress

Model training

- **Models**
  - NN (Tensorflow)
  - SVM, KNN, GNB (scikit-learn)

- **Data**
  - X (Predictor Variable) : Extracted feature
  - Y (Outcome Variable) : Autel Evo2 (0)

    DJI Phantom4 (1)

    Noise (2)

- **Train vs Test**
  - 0.8 : 0.2 ratio

- **Computing Power**
  - Laptop (Ryzen 5 5625U)

Progress

Testing Result

Feature	Accuracy	Recall	Precision	F1 Score
1. chroma_stft	0.81	0.81	0.83	0.80
2. mel	**0.89**	**0.89**	**0.89**	**0.88**
3. mfcc	0.86	0.86	0.88	0.86
4. contrast	0.76	0.76	0.75	0.75
5. tonnetz	0.78	0.78	0.79	0.76

[NN model Benchmark]

Feature	Accuracy	Recall	Precision	F1 Score
1. chroma_stft	0.86	0.86	0.86	0.86
2. mel	0.86	0.86	0.87	0.86
3. mfcc	**0.96**	**0.96**	**0.96**	**0.96**
4. contrast	0.76	0.76	0.77	0.75
5. tonnetz	0.79	0.79	0.81	0.78

[SVM model Benchmark]

- NN model
- SVM model

Progress

Testing Result

Feature	Accuracy	Recall	Precision	F1 Score
1. chroma_stft	0.80	0.80	0.81	0.79
2. mel	0.85	0.85	0.85	0.85
3. mfcc	**0.93**	**0.93**	**0.93**	**0.92**
4. contrast	0.82	0.82	0.84	0.82
5. tonnetz	0.80	0.80	0.80	0.79

[KNN model Benchmark]

- KNN model

Feature	Accuracy	Recall	Precision	F1 Score
1. chroma_stft	**0.91**	**0.91**	**0.91**	**0.91**
2. mel	0.66	0.66	0.77	0.67
3. mfcc	0.88	0.88	0.88	0.87
4. contrast	0.84	0.84	0.86	0.83
5. tonnetz	0.89	0.89	0.89	0.89

[GNB model Benchmark]

- GNB model

Progress

Model Conversion

- To implement models into Android

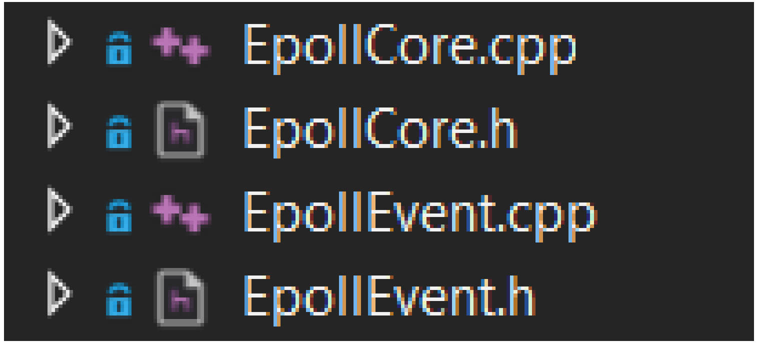
- HDF5 (Hierarchical Data Format 5)
- PB (Protocol Buffer)
- TFLITE (Tensorflow Lite)

Progress

Server program

```
for (int32 i = 0; i < 5; i++)
{
    GThreadManager->Launch([&service]()
        {
            DoWorkerJob(service);
        });
}
```

[Multiple threads start in main function]



[Classes to control epoll]

- Use Multithreading for smooth service
- Share the jobs to be processed to each thread
- Epoll is used for reducing waste and increasing performance

Packet protocol

```cpp
enum PacketNum : uint16
{
    C_LOGIN = 1000,
    S_LOGIN = 1001,
    C_AUDIO_DATA = 1002,
    S_DETECTION_RESULT = 1003,
};
```

- Received byte data is converted to packet
- Two protocols are used for the service

```cpp
struct PKT_C_AUDIO_DATA
{

    uint16 packetSize; // Common header,
    uint16 packetId; // Common header,
    uint16 featureOffset; // Address of
    uint16 featureCount = 40;
```

[Packet from a client to the server]

```cpp
struct PKT_S_DETECTION_RESULT
{
    uint16 packetSize; // Common header
    uint16 packetId; // Common header
    uint8 result;
};
```

[Packet from the server to the client]

Progress

Embedding Python program

```
#define PY_SSIZE_T_CLEAN
#include <python3.9/Python.h>

class MLManager
{
public:
    MLManager();
    ~MLManager();

    void Init();
    int8 RunModel(const float features[], uint16 featureCount);
```

[A class for call ML model written in python]

Progress

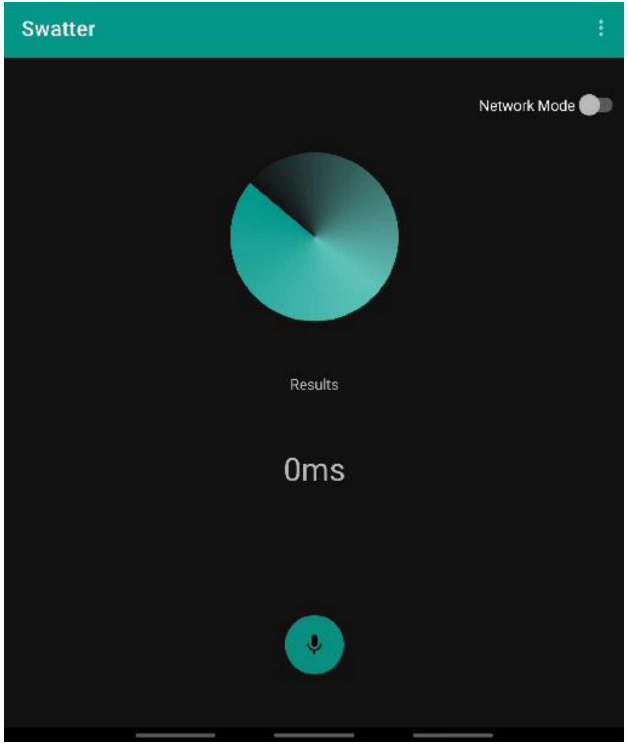
Application

- **Implementation**

  - Record Audio with Uncompressed Audio File

  - MFCC feature extraction

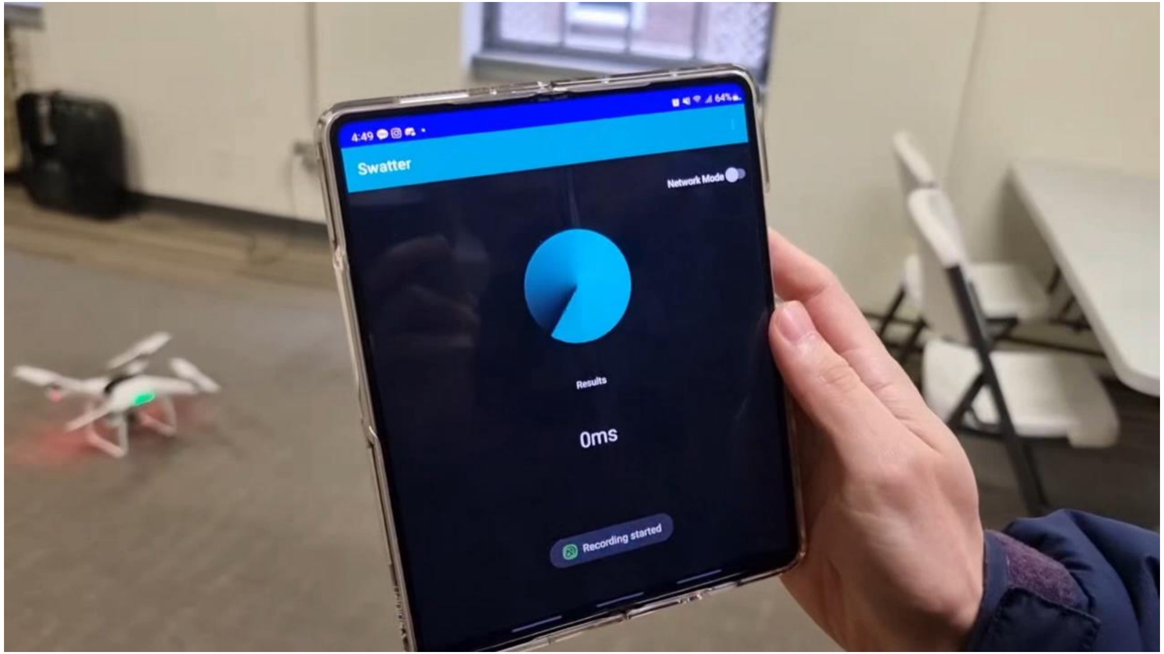
  - TensorFlow Lite Interpreter



[Application]

Progress

Demo video



Progress

Test between smartphones and server

	Smartphone	Server
Average of inference time	0.90525 ms	5270 ms

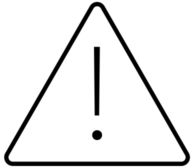
- Compare inference time.
- Using only smartphones more faster than using Server.

Future work

Real-time UAV Detection using Smartphones

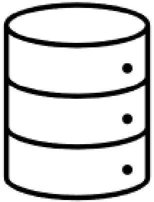
Future Work

Developing various functions on smartphone



- **Warning UAV's appearance**

  - Save GPS location
  - Share appearance to security



- **Collecting UAV's sound**

  - Restore audio file
  - Send to database

Reference

- [1] W. E. Forum, “The race for drone delivery just got a new leader: 3 key differences that will define who wins,” *Forbes*, 05-Sep-2020. [Online]. Available: https://www.forbes.com/sites/worldeconomicforum/2020/09/05/the-race-for-drone-delivery-just-got-a-new-leader-3-key-differences-that-will-define-who-wins/?sh=103ffc1714d7. [Accessed: 20-Feb-2023].
- [2] “Drone terrorism and software solutions to prevent catastrophes.” [Online]. Available: https://blog.dedrone.com/en/drone-terrorism-deadly-strikes-and-software-solutions-to-prevent-catastrophes. [Accessed: 20-Feb-2023].
- [3] B. Marcolini and C. Koettl, “How the drone attack on Maduro unfolded in Venezuela,” *The New York Times*, 10-Aug-2018. [Online]. Available: https://www.nytimes.com/video/world/americas/100000006042079/how-the-drone-attack-on-maduro-unfolded-in-venezuela.html. [Accessed: 20-Feb-2023].
- [4] C. Rogers, “Will the government ban these drones equipped with guns?,” *Fstoppers*, 21-Jul-2015. [Online]. Available: https://fstoppers.com/aerial/will-government-ban-these-drones-equipped-guns-76907. [Accessed: 20-Feb-2023].
- [5] R. Allain, “How do drones fly? physics, of course!,” *Wired*, 19-May-2017. [Online]. Available: https://www.wired.com/2017/05/the-physics-of-drones/. [Accessed: 23-Feb-2023].
- [6] I. Lee, “How to choose a safe and legal location to fly your drone,” *UAV Coach*, 24-Jun-2021. [Online]. Available: https://uavcoach.com/how-to-choose-location-to-fly-drone/. [Accessed: 23-Feb-2023].
- [7] “McAllister Park,” *Lafayette, IN*. [Online]. Available: http://lafayette.in.gov/Facilities/Facility/Details/McAllister-Park-35. [Accessed: 23-Feb-2023].
- [8] “Download,” *The CentOS Project*. [Online]. Available: https://www.centos.org/download/. [Accessed: 23-Feb-2023].
- [9] “C++17,” *cppreference.com*. [Online]. Available: https://en.cppreference.com/w/cpp/17. [Accessed: 23-Feb-2023].
- [10] “Type objects,” *Python documentation*. [Online]. Available: https://docs.python.org/3/c-api/typeobj.html. [Accessed: 23-Feb-2023].
- [11] Authors: Leah A. Wasser, “Hierarchical data formats - what is HDF5?,” *Hierarchical Data Formats - What is HDF5? | NSF NEON | Open Data to Understand our Ecosystems*. [Online]. Available: https://www.neonscience.org/resources/learning-hub/tutorials/about-hdf5. [Accessed: 23-Feb-2023].
- [12] “Protocol buffers,” *Wikipedia*, 28-Sep-2022. [Online]. Available: https://en.wikipedia.org/wiki/Protocol_Buffers. [Accessed: 23-Feb-2023].
- [13] “Tflite,” *PyPI*. [Online]. Available: https://pypi.org/project/tflite/. [Accessed: 23-Feb-2023].
- [14] M. Lee, “How to record WAV file on different devices,” *EaseUS*, 30-Nov-2022. [Online]. Available: https://recorder.easeus.com/screen-recording-tips/record-wav-file-on-windows.html. [Accessed: 23-Feb-2023].
- [15] “Feature extraction¶,” *Feature extraction - librosa 0.10.0 documentation*. [Online]. Available: https://librosa.org/doc/main/feature.html. [Accessed: 23-Feb-2023].
- [16] “Dev.java: The destination for java developers,” *Dev.java: The Destination for Java Developers*. [Online]. Available: https://dev.java/. [Accessed: 23-Feb-2023].

Swatter

Thank you

polytechnic.purdue.edu

/ TechPurdue

Mentor

Eric T. Matson

Anthony H. Smith

Yaqin Wang

Minji Lee

Q&A

Real-time UAV Detection using Smartphones

Appendix

Real-time UAV Detection using Smartphones

Structure of Neural Network Model

- Neural network model structure
- High flexibility, depends on usage.
- 3 Dense layers
  - 128 nodes for 2 layers, 1 node for output
- 2 Activation layers
  - 'ReLU' function
- 2 Dropout layers
  - Dropout rate : 0.1

```
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 128)]             0

dense_3 (Dense)              (None, 128)               16512

activation_2 (Activation)    (None, 128)               0

dropout_2 (Dropout)          (None, 128)               0

dense_4 (Dense)              (None, 128)               16512

activation_3 (Activation)    (None, 128)               0

dropout_3 (Dropout)          (None, 128)               0

dense_5 (Dense)              (None, 1)                 129

=================================================================
Total params: 33,153
Trainable params: 33,153
Non-trainable params: 0
_________________________________________________________________
```

[Structure for NN model]

Appendix

Server program

- Epoll can notify to users when monitored file descriptor is ready
- On each call to select() or poll(), the kernel must check all of the specified file descriptors to see if they are ready.

-> When monitoring a large number of file descriptors that are in a densely packed range, the timed required for this operation greatly outweights [the rest of the stuff they have to do]

- The 3 system calls that let you ask Linux to monitor lots of file descriptors are poll, epoll and select.

Appendix

Server program

```
bool ClientPacketHandler::Handle_C_AUDIO_DATA(PacketSessionRef & session, BYTE * buffer, int32 len)
{
    // TODO
    BufferReader br(buffer, len);

    PKT_C_AUDIO_DATA* pkt = reinterpret_cast<PKT_C_AUDIO_DATA*>(buffer);

    if (pkt->Validate() == false)
        return false;

    float* features = pkt->GetFeatures();

    int8 result = GMLManager.RunModel(features, pkt->featureCount);

    cout << "Handle_C_Audio, Result: " << result << endl;
    return true;
}
```

[Handle the packet from and do follow-up]

Server program

```cpp
class Lock
{
    enum : uint32
    {
        ACQUIRE_TIMEOUT_TICK = 10000,
        MAX_SPIN_COUNT = 5000,
        WRITE_THREAD_MASK = 0xFFFF'0000,
        READ_COUNT_MASK = 0x0000'FFFF,
        EMPTY_FLAG = 0x0000'0000
    };

public:
    void WriteLock(const char* name);
    void WriteUnlock(const char* name);
    void ReadLock(const char* name);
    void ReadUnlock(const char* name);

private:
    Atomic<uint32> _lockFlag = EMPTY_FLAG;
    uint16 _writeCount = 0;
};
```

```
ClientPacketHandler.cpp
ClientPacketHandler.h
DetectingSession.cpp
DetectingSession.h
DetectingSessionManager.cpp
DetectingSessionManager.h
MLManager.cpp
MLManager.h
```

- Offer the detection operation
- Call ML model to classify and send results to client