TinyM^{2}Net: A Flexible System Algorithm Co-designed Multimodal Learning Framework for Tiny Devices

Hasib-Al Rashid[1], Pretom Roy Ovi[1], Carl Busart[2], Aryya Gangopadhyay[1] and Tinoosh Mohsenin[1]

1. University of Maryland, Baltimore County

2. U.S. Army Research Laboratory

Motivation

- To integrate AI in our day-to-day life, it is being implemented on resource constrained mobile and edge platforms.
- With the exponential growth of resource constrained micro-controller (MCU) and micro-processor (MPU) powered devices, a new generation of neural networks has emerged, one that is smaller in size and more concerned with model efficiency than model accuracy.
- These low-cost, low-energy MCUs and MPUs open up a whole new world of tiny machine learning (tinyML) possibilities.
- We can directly do data analytics near the sensor by running deep learning models on very tiny devices, greatly expanding the field of AI applications.
- To mimic human-like behavior, tinyML algorithms should integrate multimodal data as well.
- Multimodal learning combines disparate, heterogeneous data from a variety of sensors and data sources into a single model.
- In contrast to standard unimodal learning systems, multimodal systems can convey complimentary information about one another, which becomes apparent only when both are integrated into the learning process.
- Thus, learning-based systems that incorporate data from many modalities can generate more robust inference or even novel insights, which would be unachievable in a unimodal system.
- However, increased model parameters and computations limit multimodal learning to be adopted for resource constrained edge and tiny ML applications.
- In this research, we address this challenge and implement multimodal learning on tiny resource constrained hardware.

Why TinyM^{2}Net?

- TinyM^{2}Net is a novel flexible system-algorithm co-designed multimodal learning framework for resource constrained devices.
- TinyM^{2}Net that can take multimodal inputs (images and audios) and be re-configured for application specific requirements.
- TinyM^{2}Net allows the system and algorithms to quickly integrate new sensors data that are customized to various types of scenarios.

Contribution Towards tinyML Implementation

- Performed network architecture optimization with depthwise separable CNN (DS-CNN) which reduces both the memory requirements and required computations.
- Performed model compression with mixed-precision quantization with the purpose of decreasing memory size for resource constrained hardware implementation while maintaining accuracy.
- Evaluated proposed TinyM^{2}Net for two different case-studies.
  - Case-study 1 includes Covid-19 detection from multimodal cough and speech audio recordings.
  - Case-study 2 includes battlefield object detection using multimodal images and audios.
- Implement TinyM^{2}Net on commodity microprocessor unit, Raspberry Pi 4. We measured inference time while it was in use, as well as providing the appropriate power profiling to ensure that our system is adaptable. To be called a real-time implementable tinyML system, TinyM^{2}Net meets all the requirements.

TinyM^{2}Net Framework

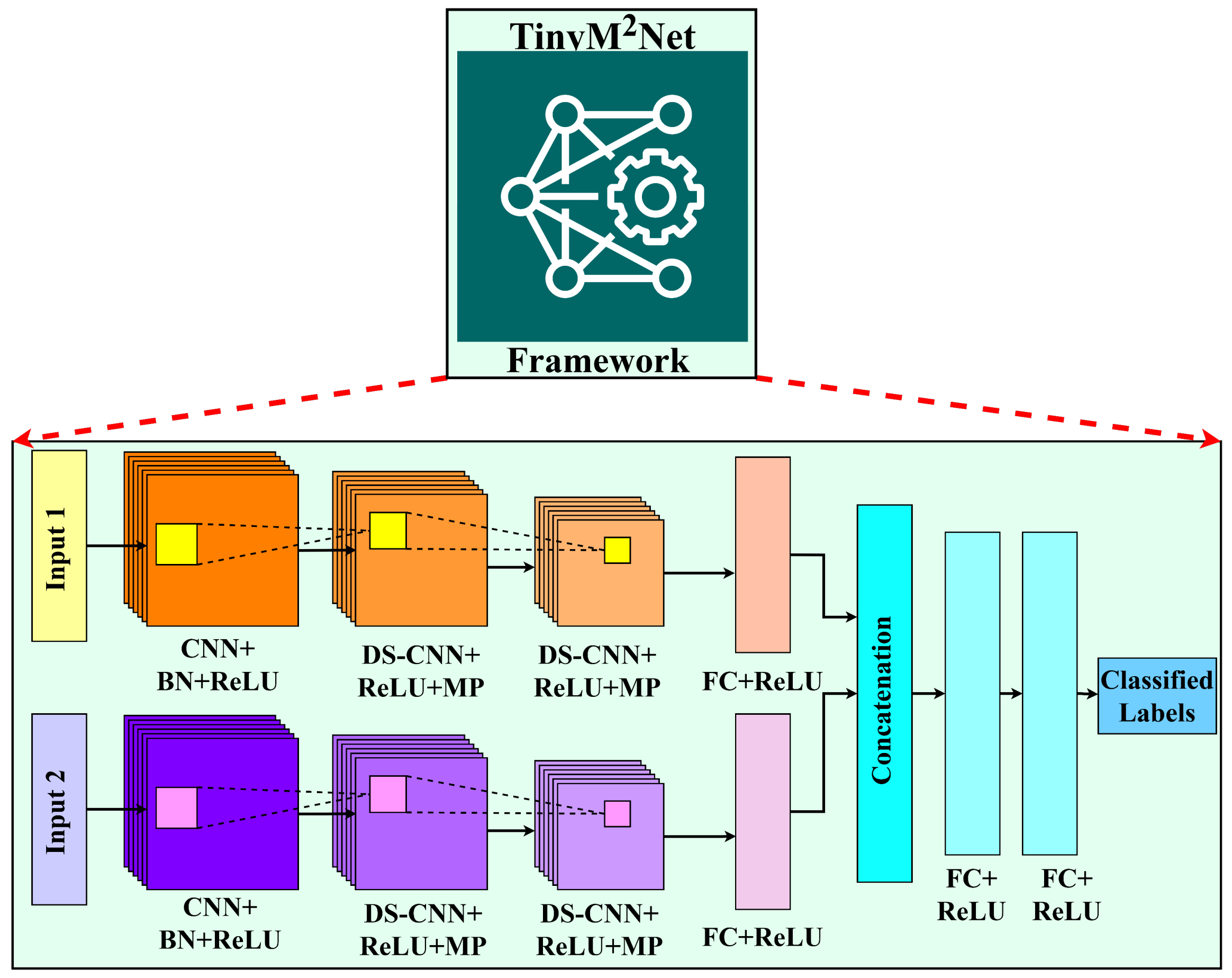
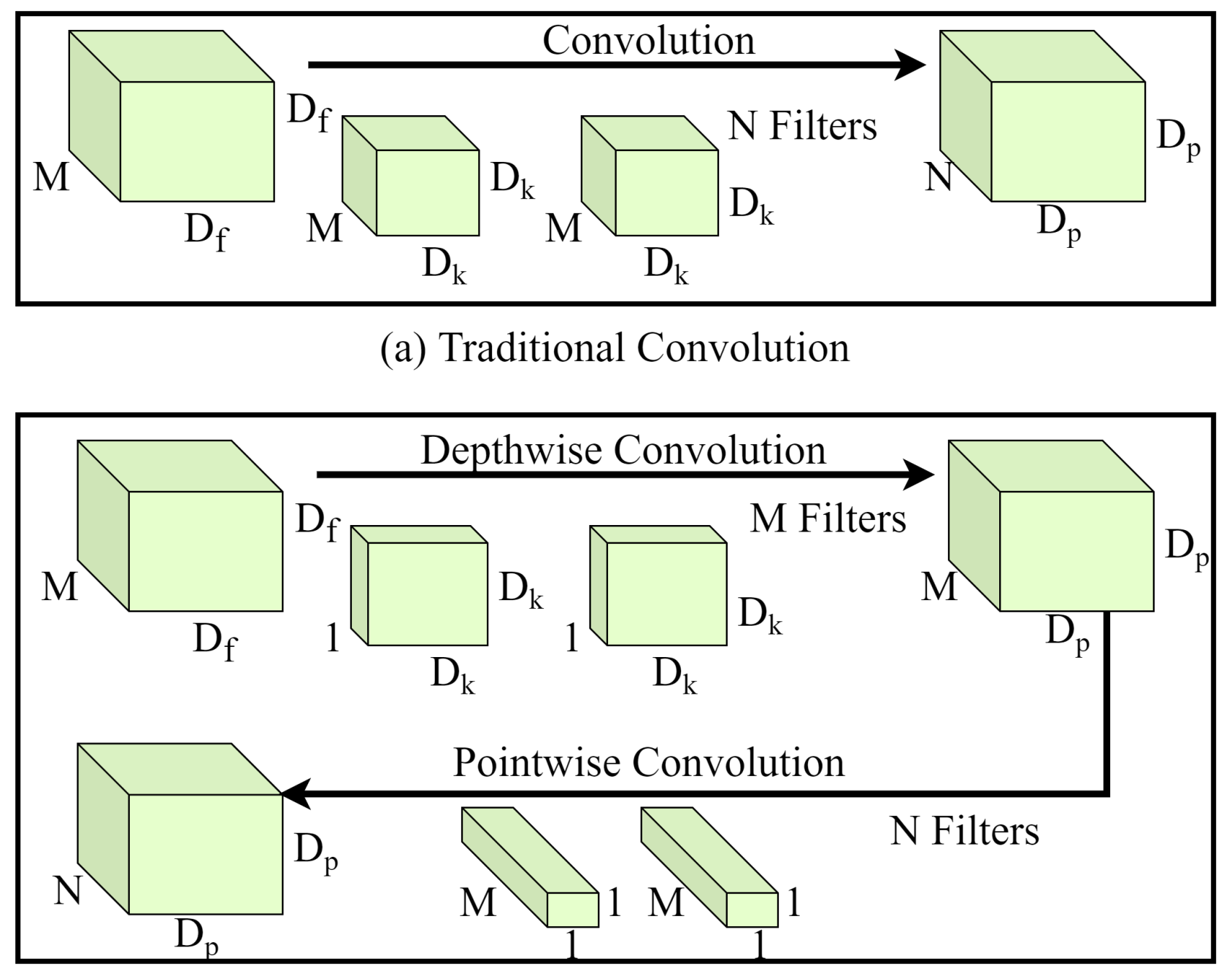


Figure: The proposed TinyM^{2}Net framework for multimodal learning on tiny hardware. TinyM^{2}Net is flexible in terms of number of input, number of layers and hyper-parameters based on application specific requirement. New data modality suited for various settings can be readily included into the device. Some of the input information can be images, other input data can be auditory.

Network Architecture Optimization with DS-CNN



(a) Traditional Convolution

(b) Depthwise Separable Convolution

Figure: Detailed Operations inside traditional CNN and DS-CNN

Model Compression with Mixed-Precision Quantization

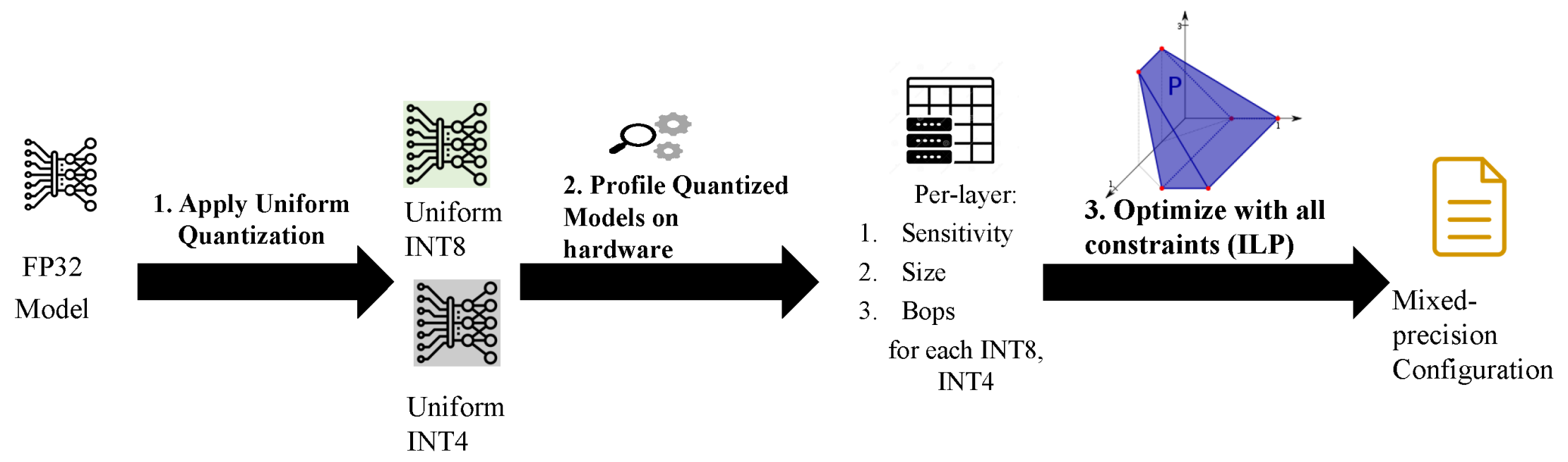

FP32 Model
1. Apply Uniform Quantization
Uniform INT8
Uniform INT4
2. Profile Quantized Models on hardware
Per-layer:
1. Sensitivity
2. Size
3. Bops
for each INT8, INT4
3. Optimize with all constraints (ILP)
P
Mixed-precision Configuration

Figure: Finding Mixed Precision Bit Setting Using Integer Linear Programming (ILP)

- We pre-calculate the sensitivity of each layer independently.
- Hessian based perturbation, presented in [1] is used as sensitivity metric.
- Minimizing this sensitivity, Integer Linear Programming is used to find the right bit precision settings, solving following equations.

Objective:

$$\min \{x_i\}_{i=1}^{Y} \sum_{i=1}^{Y} \Omega_i^{(x_i)}, \tag{2}$$

Subject to:

$$\sum_{i=1}^{Y} \mu_i^{(x_i)} \leq Model\ Size, \tag{3}$$

$$\sum_{i=1}^{Y} \beta_i^{(x_i)} \leq BOPS\ Limit, \tag{4}$$

Case Study 1: Covid Detection from Multimodal Audio Recordings

- We used the dataset from the Inter Speech 2021 ComParE challenge [2].
- In this dataset there are 929 cough audios from 397 participants and 893 speech recordings from 366 participants.
- Each recording included a COVID-19 test result that was self-reported by the participant.
- To build the two-class classification task, the original COVID-19 test results were mapped to positive (designated as 'P') or negative (designated as 'N') categories.

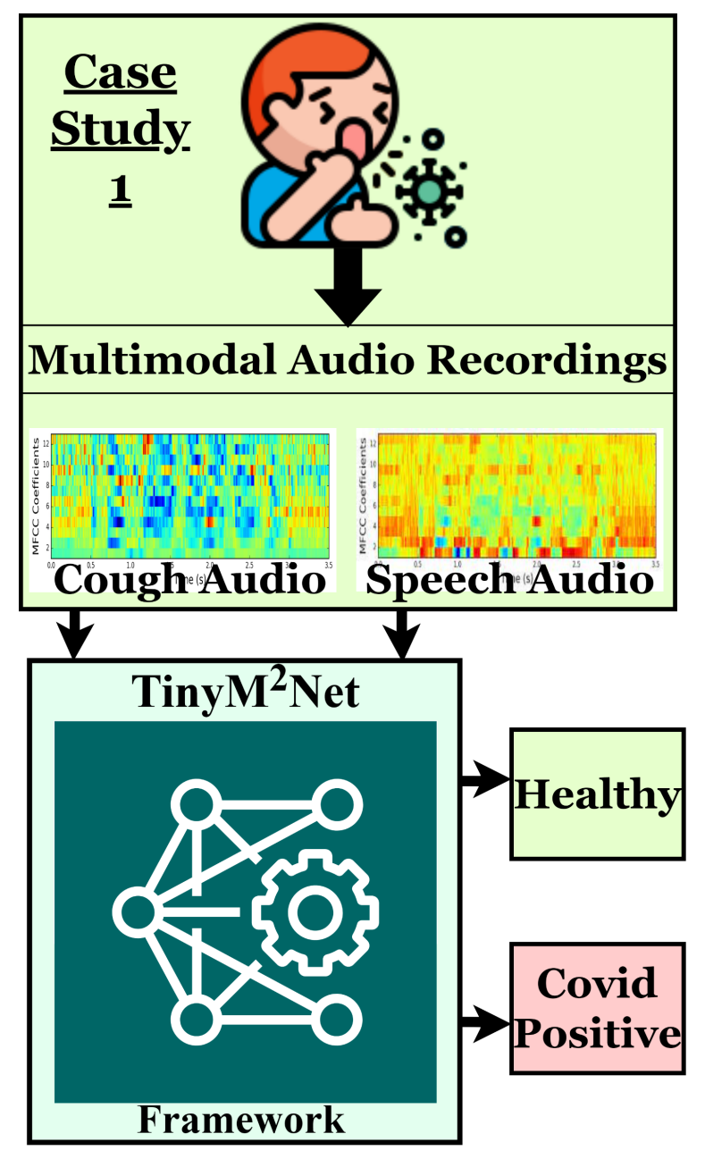


Figure: High-level Overview of Case-study 1

Table: Detailed network architecture for Case-Study 1

Layers	Description
Input Layer	Cough Audio MFCC Vector
Input Layer	Speech Audio MFCC Vector
Conv2D	Kernels = 16 × (3 × 3) – BN – ReLU
Conv2D	Kernels = 64 × (3 × 3) – BN – ReLU
SeparableConv2D	Kernels = 32 × (3 × 3) – ReLU
SeparableConv2D	Kernels = 32 × (3 × 3) – ReLU
MaxPooling2D	Size = (3 × 3) – 20% Dropout
MaxPooling2D	Size = (2 × 2) – 20% Dropout
SeparableConv2D	Kernels = 32 × (3 × 3) – ReLU
SeparableConv2D	Kernels = 16 × (3 × 3) – ReLU
MaxPooling2D	Size = (3 × 3) – 20% Dropout
MaxPooling2D	Size = (2 × 2) – 20% Dropout
Flatten	21 × 1 × 32
Flatten	81 × 1 × 16
Dense	Neurons = 32 – ReLU – 20% Dropout
Dense	Neurons = 32 – ReLU – 20% Dropout
Concatenation	32 + 32
Dense	Neurons = 256 – ReLU – 20% Dropout
Dense	Neurons = 128 – ReLU – 20% Dropout
Dense	Neurons = 2 – SoftMax

Case Study 2: Battlefield Object Detection from Multimodal Images and Audios

- As open-sourced dataset for research on battlefield environment is very limited, we have created our own dataset for this case-study.
- We have created a dataset for multiclass classification problem with 4 classes as Helicopter, Bomb, Gun, and Tank.
- We have selected 4 publicly available YouTube videos [3-6] from where we extracted the images and corresponding audios of Helicopter, Bomb, Gun, and Tank.

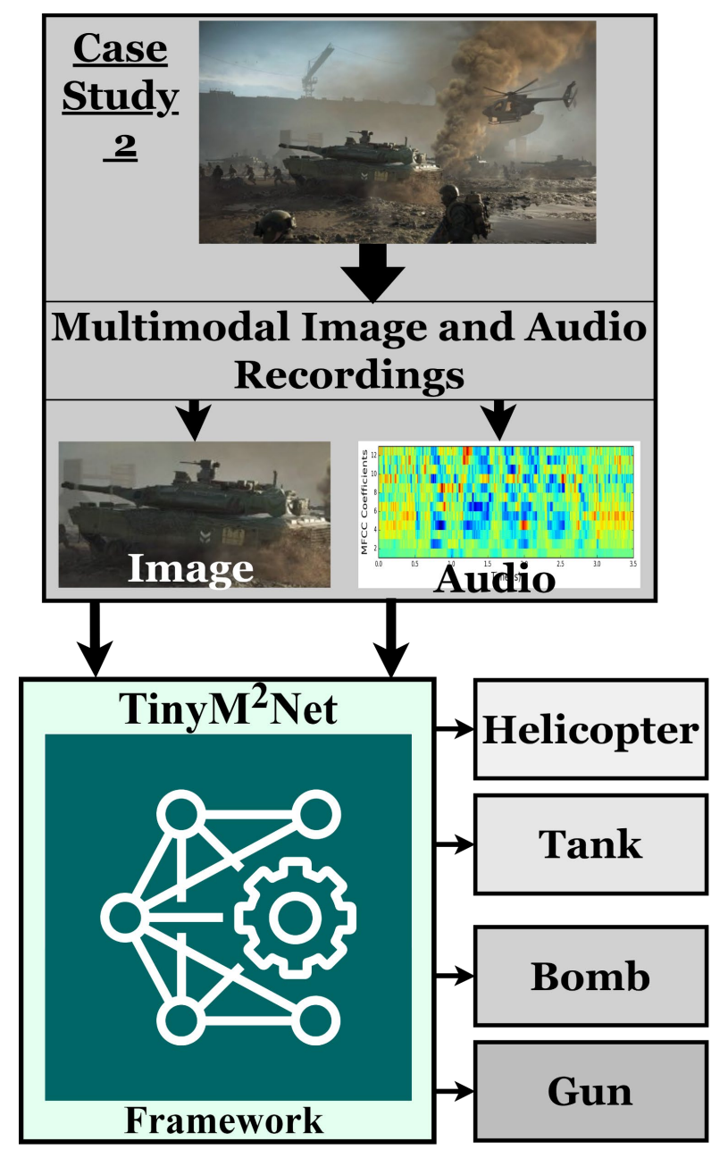


Figure: High-level Overview of Case-study 2

Table: Detailed network architecture for Case-Study 2

Layers	Description
Input Layer	Audio MFCC Vector
Input Layer	Image Vector
Conv2D	Kernels = 64 × (3 × 3) – BN – ReLU
Conv2D	Kernels = 64 × (3 × 3) – BN – ReLU
SeparableConv2D	Kernels = 32 × (3 × 3) – ReLU
SeparableConv2D	Kernels = 32 × (3 × 3) – ReLU
MaxPooling2D	Size = (2 × 2) – 20% Dropout
MaxPooling2D	Size = (2 × 2) – 20% Dropout
SeparableConv2D	Kernels = 64 × (3 × 3) – ReLU
SeparableConv2D	Kernels = 64 × (3 × 3) – ReLU
MaxPooling2D	Size = (2 × 2) – 20% Dropout
MaxPooling2D	Size = (2 × 2) – 20% Dropout
Flatten	11 × 3 × 64
Flatten	8 × 8 × 64
Dense	Neurons = 64 – ReLU – 20% Dropout
Dense	Neurons = 64 – ReLU – 20% Dropout
Concatenation	64 + 64
Dense	Neurons = 64 – ReLU – 20% Dropout
Dense	Neurons = 4 – SoftMax

Experimental Results

- We trained our model with categorical cross-entropy loss and Adam optimizer.

Table: Summary of the TinyM^{2}Net Framework Evaluation Results

Case Study	Quantization	Accuracy (%)	Model Size (KB)
1	Floating Point	90.4	845
1	W 8 A 8 (uniform 8)	89.6	216
1	**W 4/8 A 4/8 (MP)**	**88.4**	**145**
1	W 4 A 4 (uniform 4)	83.6	107
2	Floating Point	98.5	1605
2	W 8 A 8 (uniform 8)	97.9	407
2	**W 4/8 A 4/8 (MP)**	**96.8**	**269**
2	W 4 A 4 (uniform 4)	91.3	205

TinyM^{2}Net Running on Tiny Device

- The inference stage must be implemented on resource constrained tiny devices in order to make the TinyM^{2}Net system real-time.
- We implemented TinyM^{2}Net on Raspberry Pi 4 which has quad-core Cortex-A72 (ARM v8) and 2GB LPDDR4 memory.
- Performance evaluation of the TinyM^{2}Net on resource constrained Raspberry Pi 4 was based on two metrics: **inference time** and **power consumption** during inference.
- Data loading, model loading, and visual display of the final result all contribute to this inference time's length.
- The inference time was measured with the help of Raspbian OS's `time' function. We used a batch size of 1, which is the time it takes to process a single data point, to calculate the inference time.

Table: Implementation of the TinyM^{2}Net framework to resource constrained Raspberry Pi 4 device

Case Study	Inference Time (s)	Power (mW)
1	1.2	798
2	1.7	959

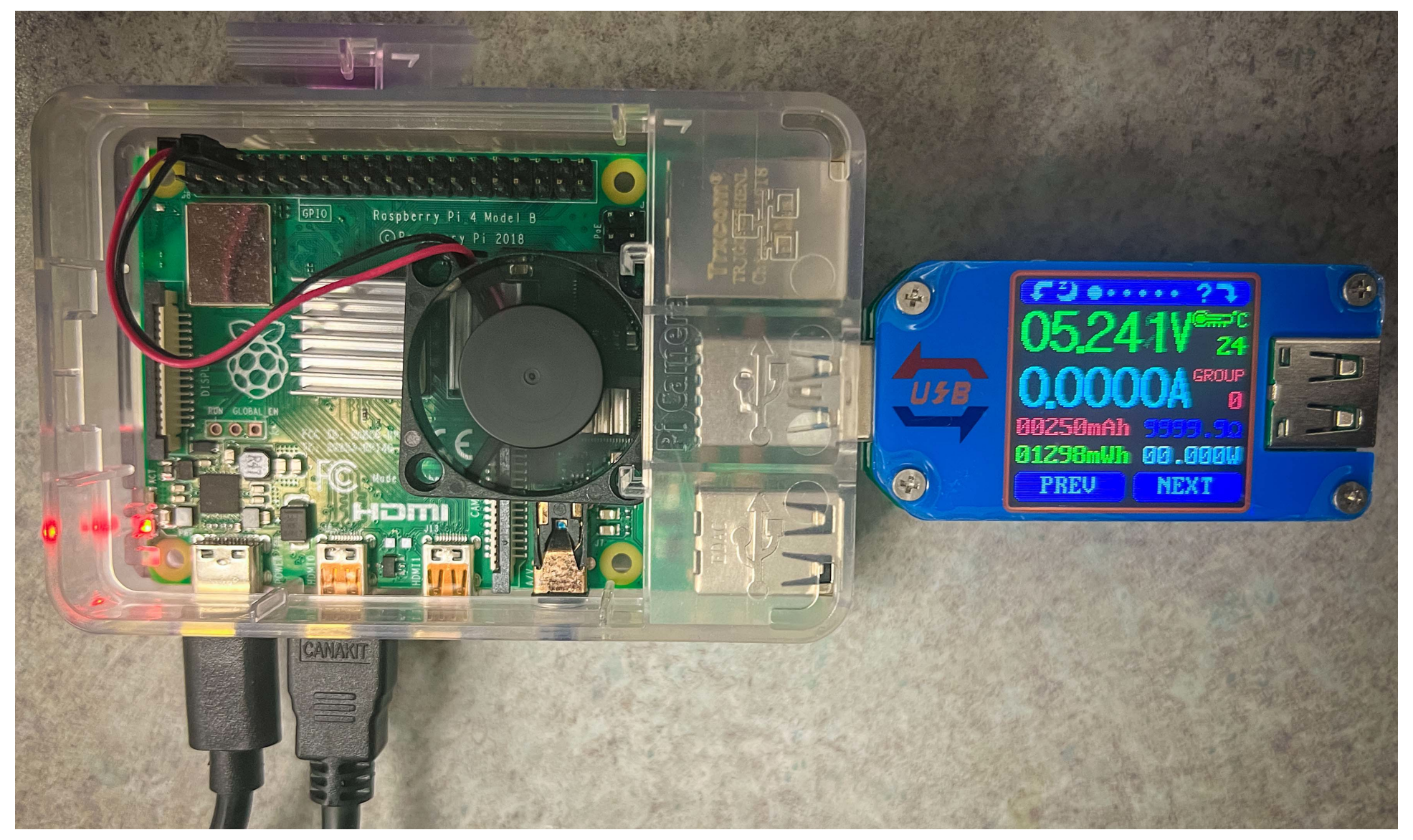


Figure : Implementation of the TinyM^{2}Net framework to resource constrained Raspberry Pi 4 device

Summary

- To implement into tiny hardware, extensive model compression was done in terms of networks architecture optimization and MP quantization (mixed 8-bit and 4-bit).
- The compressed TinyM^{2}Net achieves 88.4% accuracy in COVID-19 detection and 96.8% accuracy in battlefield object detection.
- We tested our TinyM^{2}Net model on a Raspberry Pi 4 to see how they perform when deployed to a resource constrained tiny device.

Acknowledgement

We acknowledge the support of the U.S. Army Grant No. W911NF21-20076.

References

[1] Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael Mahoney, et al. Hawq-v3: Dyadic neural network quantization. In International Conference on Machine Learning, pages 11875–11886. PMLR, 2021

[2] Björn W Schuller, Anton Batliner, Christian Bergler, Cecilia Mascolo, Jing Han, Iulia Lefter, Heysem Kaya, Shahin Amiriparian, Alice Baird, Lukas Stappen, et al. The interspeech 2021 computational paralinguistics challenge: Covid-19 cough, covid-19 speech, escalation & primates. arXiv preprint arXiv:2102.13468, 2021.

[3] Shawn Adams. Best helicopter sounds. top sounds that helicopters make, url: https://www.youtube.com/watch?v=e8backzbqzc.

[4] Sintonizar Productions. Real explosion sound effects asmr, url: https://www.youtube.com/watch?v=lhyhy5uioji&t=64s.

[5] Car News TV. The best tanks of world war ii start sound and ride, url:https://www.youtube.com/watch?v=kpvg5r7niso.

[6] PC Gaming Videos. Battlefield 5 gun sounds of all weapons, url: https://www.youtube.com/watch?v=tmwagle6pr8.