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

Hasib-Al Rashid*
hrashid1@umbc.edu
University of Maryland, Baltimore County
Baltimore, Maryland, USA

Pretom Roy Ovi*
povi1@umbc.edu
University of Maryland, Baltimore County
Baltimore, Maryland, USA

Aryya Gangopadhyay
University of Maryland, Baltimore County
Baltimore, Maryland, USA

Tinoosh Mohsenin
University of Maryland, Baltimore County
Baltimore, Maryland, USA

ABSTRACT

With the emergence of Artificial Intelligence (AI), new attention has been given to implement AI algorithms on resource constrained tiny devices to expand the application domain of IoT. Multimodal Learning has recently become very popular with the classification task due to its impressive performance for both image and audio event classification. This paper presents *TinyM^{2}Net* - a flexible system algorithm co-designed multimodal learning framework for resource constrained tiny devices. The framework was designed to be evaluated on two different case-studies: COVID-19 detection from multimodal audio recordings and battle field object detection from multimodal images and audios. In order to compress the model to implement on tiny devices, substantial network architecture optimization and mixed precision quantization were performed (mixed 8-bit and 4-bit). *TinyM^{2}Net* shows that even a tiny multimodal learning model can improve the classification performance than that of any unimodal frameworks. The most compressed *TinyM^{2}Net* achieves 88.4% COVID-19 detection accuracy (14.5% improvement from unimodal base model) and 96.8% battle field object detection accuracy (3.9% improvement from unimodal base model). Finally, we test our *TinyM^{2}Net* models on a Raspberry Pi 4 to see how they perform when deployed to a resource constrained tiny device.

KEYWORDS

Multimodal Learning, TinyML, Micro AI, COVID-19 Detection, Battle Field Object Detection.

ACM Reference Format:
Hasib-Al Rashid, Pretom Roy Ovi, Aryya Gangopadhyay, and Tinoosh Mohsenin. 2022. TinyM^{2}Net: A Flexible System Algorithm Co-designed Multimodal Learning Framework for Tiny Devices. In *Proceedings of tinyML Research Symposium (tinyML Research Symposium'22).* ACM, New York, NY, USA, 7 pages.

*Both authors contributed equally to this research.

tinyML Research Symposium'22, March 2022, San Jose, CA

1 INTRODUCTION

Artificial Intelligence (AI) has a huge impact on our daily lives now-a-days. In our daily lives, AI has brought convenience and ease of use to the table. The AI devices are now able to perform computationally intensive tasks and eliminate human error from the system to a large extent, making this convenience possible. We see AI techniques and devices being used in domains such as medical diagnosis, security and combat fields, robotics, vision analytics, knowledge reasoning, navigation, etc. today. 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.

Modern IoT and wearable devices, such as activity trackers, environmental sensors, images, and audio sensors can generate large volumes of data on a regular basis. Modern AI is increasingly reliant on data from numerous sources in order to produce more accurate findings. Learning process for human is multimodal. We human can take our decision by processing different modalities of data. To mimic human-like behavior, AI 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. Multimodal learning has two key advantages. To begin, several sensors observing the same data can produce more robust predictions, as recognizing changes in it may need the presence of both modalities. Second, the integration of many sensors enables the capture of complementing data or trends that individual modalities may miss. However, increased model parameters and computations limit multimodal learning to be adopted for resource constrainded edge and tiny ML applications.

Commodity MCUs and MPUs have a very limited resource in terms of memory (SRAM) and storage (Flash) budget. A typical MCU has an SRAM of less than 512kB, which is insufficient for installing the majority of off-the-shelf deep learning networks. Even on more capable hardware such as the Raspberry Pi 4, configuring inference to run in the L2 cache (1MB) can dramatically increase energy efficiency. These new issues add to the difficulty of performing efficient multimodal learning inference with a low peak memory consumption. In this paper, we address this challenge and implement multimodal learning on tiny hardware. We take advantages of state-of-the-art compression techniques and combined them with computationally relaxed layers to implement energy efficient multimodal learning on tiny processing hardware. We proposed a flexible system algorithm co-designed framework *TinyM^{2}Net* which is re-configurable in terms of input data modality and data shapes, number of layers, filter sizes etc. hyper-parameters for the sake of application requirements. We evaluated *TinyM^{2}Net* with two different case-studies: audio processing with multimodal audios and object detection with multimodal images and audios. *TinyM^{2}Net* is then implemented on commodity tiny MPU, Raspberry Pi 4 to measure real-time performance on tiny hardware. The main contributions of this paper are as follows:

- Propose *TinyM^{2}Net*, 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.
- Perform network architecture optimization, and mixed-precision quantization with the purpose of decreasing computation complexity and 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 of the requirements.

2 RELATED WORKS

Authors in [17] presented a high level overview on the optimization techniques of deep neural networks (DNNs) for tinyML on device inferences. TinyML model optimization includes different algorithms of Parameter Search, Sparsification and Quantization techniques. Element-wise pruning[19], Structured Pruning [4, 13] these techniques reduces the unimportant weights and compress the models to be implemented on tinyML devices. Extreme low precision quantization [3, 14] and mixed precision quantization [9, 11, 27, 28, 30] is adopted by the researchers to decrease the memory requirements of the DNN models. MCUNet [15], MicroNets [6] are proposed to deploy DNN models on micro-controller units (MCU). Recently, multimodal learning attracts researchers to improve the classification accuracy of the models integrating different modalities of data fusion [1, 7, 16]. However, implementation of multimodal learning into resource constrained tiny hardware is very limited due to its large model sizes. We present a novel multimodal learning framework *TinyM^{2}Net* which is system algorithm co-designed and different model compression techniques were implemented to compress the large multimodal models to be implemented on tiny devices.

3 TinyM^{2}Net FRAMEWORK

Figure 1 shows the proposed *TinyM^{2}Net* framework along with its detailed architecture. Based on the case studies we mention in section 5, *TinyM^{2}Net* is able to integrate two different modalities of data and classify them. Proposed *TinyM^{2}Net* is designed based on mainly convolutional neural networks (CNN). CNN performed very promising with images and audio data classification previously which is the reason behind choosing CNN as our base model.

We will evaluate *TinyM^{2}Net* on two different case studies described in details in section 5. Case-study 1 is detecting COVID-19 signatures using two different modalities of audio recordings, cough sound and speech sound. Case-study 2 is based on detecting battle field object using images and audios. While processing audio data, we have divided the whole audio into shorted window frames, the size of those frames are variable based on application requirements. Then the window frames are converted into Mel-Frequency Cepstral Coefficients (MFCCs) spectrograms. In the next step, two different data modalities, whether be images or audios, are sent to the CNN layers for feature extraction. The number of layers of our CNN layers can be adjusted to suit application specific requirements. Maxpooling layers is used to reduce the size of the feature map. A number of fully interconnected layers are applied once the output is flattened to achieve the needed tiny feature map size in order to isolate enough information with linkages between nodes. The outputs from two parallel feature extraction layers from two data modalities are then concatenated and processed through fully connected layers to produce the final label. The activation function for each layer is a rectified linear unit (ReLU). Softmax activation function is used to generate a probability distribution for the final layer.

4 MODEL COMPRESSION FOR TINY DEVICES

Traditional CNN models are very notorious for being bulky in terms of memory and computation requirements. To implement CNN on low powered embedded tiny devices, researchers proposed various compression techniques which results in highly optimized CNN models. Our proposed *TinyM^{2}Net* adopts different model compression techniques which optimizes the network architectures and memory requirements. *TinyM^{2}Net* adopts Depthwise Separable CNN (DS-CNN) to reduce the computation from traditional CNN layers. To have improvement on memory requirements, *TinyM^{2}Net* adopts low precision and mixed-precision (MP) quantization. We emphasize on MP quantization as uniform low precision quantization degrades model accuracy.

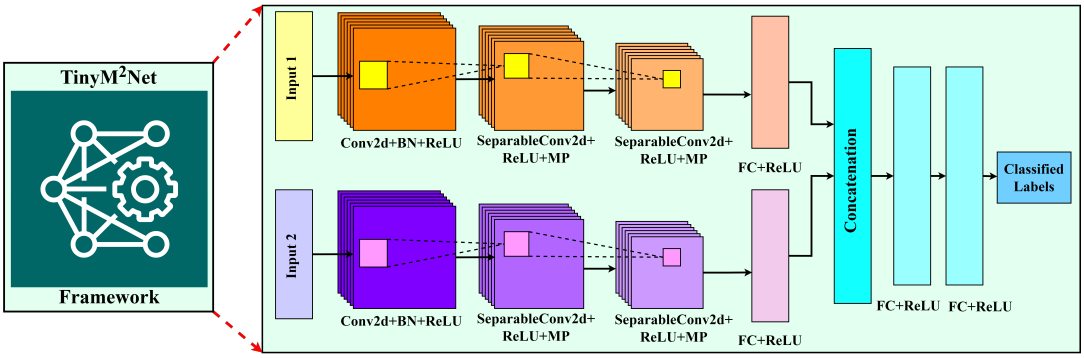

Figure 1: The proposed *TinyM^{2}Net* framework for multimodal learning on tiny hardwares. *TinyM^{2}Net* is flexible in terms of number of input, number of layers and hyper-parameters based on application specific requirement. New data modality suited for various settings can be readily included into the device. Some of the input information is images, other input data can be auditory.

Table 1: Number of parameter and required computations equations for different types of convolution layers.

CNN types	No. of Parameters	No. of Computations
Traditional	$M \times D_k^2 \times N$	$M \times D_k^2 \times D_p^2 \times N$
Depthwise Separable	$M \times D_k^2 + M \times N$	$M \times D_p^2 \times D_k^2 + M \times D_p^2 \times N$

4.1 Network Architecture Optimization with DS-CNN

Figure 3 presents the conventions and the techniques that is done in traditional CNN and DS-CNN . In traditional CNN, if the input is of size $D_f \times D_f \times M$ and N is the number of filters having a size $D_k \times D_k \times M$ then output of this layer without zero padding applied is of size $D_p \times D_p \times M$. If the stride for the convolution is S then D_p is determined by the following equation:

$$D_p = \frac{D_f - D_k}{S} + 1 \tag{1}$$

In this layer, the filter convolves over the input by performing element wise multiplication and summing all the values. A very important note is that depth of the filter is always same as depth of the input given to this layer. The computational cost for traditional convolution layer is $M \times D_k^2 \times D_p^2 \times N$ [10].

Depthwise Separable convolution is a combination of depthwise and pointwise convolution [12]. In contrast to traditional CNNs, which apply convolution to all M channels at once, depthwise operations only apply convolution to a single channel at a time. So here the filters/kernels will be of size $D_k \times D_k \times 1$. As there are M channels at the input, M numbers of such filters are needed. This will produce a output of size $D_p \times D_p \times M$. A single convolution operation require $D_k \times D_k$ multiplications. Since the filter are slided by $D_p \times D_p$ times across all the M channels. The total number of computation for one depthwise convolution comes to be $M \times D_p^2 \times D_k^2$. In point-wise operation, a 1×1 convolution is applied on the M channels. The filter shape for this operation will be $1 \times 1 \times M$. If we use N such filters, the output shape becomes $D_p \times D_p \times N$. One convolution operation in this needs $1 \times M$ multiplications. The total number of operations for one pointwise convolution is $M \times D_p^2 \times N$. Therefore, total computational cost of one depthwise separable convolution is $M \times D_p^2 \times D_k^2 + M \times D_p^2 \times N$ [10]. Table 1 shows the equations required for calculating the number of parameters and number of computations for each of the traditional CNN and DS-CNN layers. Here $D_k \times D_k$ is the size of the filter, $D_p \times D_p$ is the size of the output, M is number the of input channels and N is the number of output channels.

4.2 Model Quantization

To have lesser memory requirements, model quantization is now attracting to the researchers to design tinyML models. The accuracy of a model can be significantly degraded if it is uniformly quantized to low bit precision. It is possible to address this with mixed-precision quantization in which each layer is quantized with different bit precision. The main idea behind mixed precision quantization is to keep sensitive layers at higher precision and insensitive layers at lower precision. However, the search space for this bit setting grows exponentially as the number of layers increase, which is challenging. Various methods have been offered to deal with this enormous search area. Reinforcement learning (RL) and Neural Architecture Search (NAS) have recently been presented as efficient approaches for searching the search space. As a result, these methods [9, 27, 28] often require a huge amount of computational resources and their performance is highly dependent on hyperparameters and even initialization. An algorithm known as Integer Linear Programming (ILP) is employed in [11, 30]. ILP is very light weight and gives result within second. We adopted ILP and formulated our problem following the methodology described in [30], simplifying some constraints to get the mixed precision settings for our *TinyM^{2}Net*. ILP equations were solved using a python module called Pulp.

To tackle the accuracy degradation with extreme low bit precision quantization, we chose two different bit precision settings ($X = 2$), INT4 and INT8 for our *TinyM^{2}Net* framework. As our *TinyM^{2}Net* is flexible in terms of number of layers, for a model with Y layers, the search space for ILP becomes X^Y. ILP will find the best bit precision choices from X^Y search spaces to have optimal trade-off between model perturbation Ω and user specific constraints i.e. model size and Bit Operations, BOPS. Each of these bit-precision options has the potential to produce different model perturbation. We assumed the perturbation for each layer are independent to

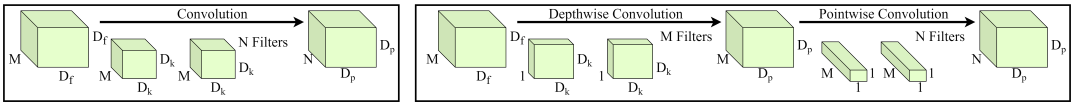

Figure 2: Detailed operations inside traditional CNN and DS-CNN.

each other [30] (i.e., $\Omega = \sum_{i=1}^{Y} \Omega_i^{x_i}$, where $\Omega_i^{x_i}$ is the perturbation of i-th layer with x_i bit). This enables us to pre-calculate the sensitivity of each layer independently, with only XY computations required. Hessian based perturbation, presented in [30] is used as sensitivity metric. Minimizing this sensitivity, ILP tries to find the right bit precision settings. The ILP equations would be as follows: Objective:

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

Subject to:

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

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

Here, $\mu_i^{(x_i)}$ denotes the size of the i-th layer with x_i bit quantization and $\beta_i^{(x_i)}$ is the corresponding BOPS required for computing that layer. All the equations are adopted from [30]. We have considered the bit precision for both weights and activations to be same so that the mathematical operations become efficient. The overall MP quantization process is summarized in figure [somethin].

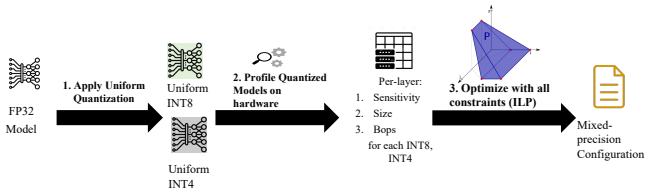

Figure 3: Finding Mixed Precision Bit Setting Using Integer Linear Programming (ILP)

5 TinyM^{2}Net EVALUATION

We evaluated proposed *TinyM^{2}Net* with two very important real-world case studies: COVID-19 detection from multimodal audios and Battlefield Object Detection from multimodal images and audios. Both of the case studies proves to be important sectors where tinyML implementations is much required.

5.1 Case Study 1: Covid Detection from Multimodal Audio Recordings

Combining numerous data sources has always been a high priority topic, but with the advent of new AI-based learning algorithms, it has become critical to combine the complementary capabilities of distinct data sources for effective diagnosis, treatment, prognosis, and planning in a variety of medical applications. With the onset of COVID-19 pandemic, patient pre-screening from passively recorded audios has become an active area of research. Therefore, a bunch of unimodal and multimodal COVID-19 audio dataset have been presented [5, 8, 20, 23]. The ultimate goal in this research is to provide COVID-19 pre-screening mobile or tiny devices. Figure 4 shows the highlevel overview of the evaluation of *TinyM^{2}Net* in terms of COVID-19 detection.

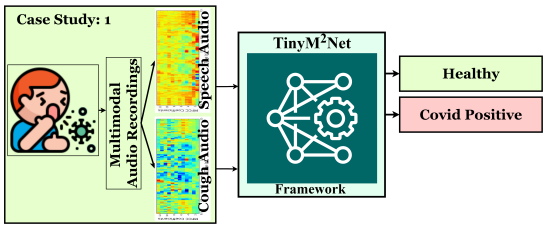

Figure 4: The proposed *TinyM^{2}Net* framework to detect COVID-19 from two different modalities of audios, cough audios and speech audios.

5.1.1 ***Dataset Description.*** We have used the dataset from the COVID-19 cough sub-challenge (CCS) and the COVID-19 speech sub-challenge (CSS) from the Inter Speech 2021 ComParE challenge [22]. This dataset is a subset of the bigger dataset [8] collected by University of Cambridge. In this dataset there are 929 cough audios from 397 participants and 893 speech recordings from 366 participents. 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.

5.1.2 ***Experimental Setups, Results and Analysis.*** To create a balanced multimodal dataset we have taken 893 cough recordings from 366 participants matching their IDs mentioned in the metadata so that we have both cough and speech recordings contributing from a same person. Then we have divided the audios into 2 sec audio chunks and produces 6000 random samples out of that. Then we converted them into MFCC spectrogram and passed them to *TinyM^{2}Net*. *TinyM^{2}Net* process two different modalities of audios with its parallel CNN layers, extracts features, combines them and classify at the end as binary classification. We have used the 1st layer as traditional CNN and the later layers as DS-CNN. The detailed network architecture is mentioned in table 2.

We trained our model with categorical cross-entropy loss and Adam optimizer. We achieved 90.4% classification accuracy with FP32 bit precision. We then quantize our model to uniform 8-bit and 4-bit precision and achieved 89.6% and 83.6% classification accuracy. Our MP quantization technique improves the classification accuracy to 88.4%. The best baseline accuracies for unimodal COVID-19

Table 2: Details of the network architecture for multimodal COVID-19 detection

Layers	Description	Output
Input Layer	Cough Audio MFCC Vector	203×20 × 1
Input Layer	Speech Audio MFCC Vector	333×13×1
Conv2D	Kernels = 16 ×(3×3) - BN - ReLU	201×18×16
Conv2D	Kernels = 64 ×(3×3) - BN- ReLU	333×13×64
Separable Conv2D	Kernels = 32 ×(3×3) - ReLU	199x16x32
Separable Conv2D	Kernels = 32 ×(3×3) - ReLU	329×9×32
MaxPooling2D	Pool size = (3×3), 20% Dropout	66×5×32
MaxPooling2D	Pool size = (2×2), 20% Dropout	164×4×32
Separable Conv2D	Kernels = 32×(3×3) - ReLU	64×3×32
Separable Conv2D	Kernels = 16×(3×3) - ReLU	162 × 2 ×16
MaxPooling2D	Pool size = (×3), 20% Dropout	21×1×32
MaxPooling2D	Pool size = (2×2), 20% Dropout	81×1×16
Flatten	21×1×32	672
Flatten	81×1×16	1296
Dense	Neurons = 32 - ReLU - 20% Dropout	32
Dense	Neurons = 32 - ReLU - 20% Dropout	32
Concatenate	32 + 32	64
Dense	Neurons = 256 - ReLU - 20% Dropout	256
Dense	Neurons = 128 - ReLU - 20% Dropout	128
Dense	Neurons = 2 - Softmax	2

Table 3: Details of the network architecture for multimodal battle field object detection

Layers	Description	Output
Input Layer	Audio MFCC Vector	44×13×1
Input Layer	Image Vector	32×32×3
Conv2D	Kernels = 64×(3 ×3) - BN - ReLU	44×13×64
Conv2D	Kernels = 64 ×(3×3)- BN - ReLU	32×32×64
Separable Conv2D	Kernels = 32 ×(3×3) - ReLU	44×13×32
Separable Conv2D	Kernels = 32 ×(3×3) - ReLU	32×32×64
MaxPooling2D	Pool size = (2×2), 20% Dropout	22×6×32
MaxPooling2D	Pool size = (2×2), 20% Dropout	16×16×32
Separable Conv2D	Kernels = 64 ×(3×3) - ReLU	22 × 6 ×64
Separable Conv2D	Kernels = 64 ×(3×3) - ReLU	16 × 16 ×64
MaxPooling2D	Pool size = (2×2), 20% Dropout	11×3×64
MaxPooling2D	Pool size = (2×2), 20% Dropout	8×8×64
Flatten	11×3×64	2112
Flatten	8×8×64	4096
Dense	Neurons = 64 - ReLU - 20% Dropout	64
Dense	Neurons = 64 - ReLU - 20% Dropout	64
Concatenate	64 + 64	12
Dense	Neurons = 64 - ReLU - 20% Dropout	64
Dense	Neurons = 4 - Softmax	4

detection from cough audio and speech audio were 73.9% and 72.1% [**?**]. Our *TinyM^{2}Net* outperforms the baseline results by 14.5%.

5.2 Case Study 2: Battlefield Component Detection from Multimodal Images and Audios

Research in the field of computer vision has always focused on the detection of specific targets in an image. Research on the identification of armored vehicles in the battlefield environment, as well as the deployment dynamics, recognition and tracking, precise strike, and so on, are critical military objectives. There are still many difficulties in detecting armored vehicles on the battlefield because of the complication of the environment [29, 31]. Authors in [18, 24] proposed multimodal learning approach in object detection task. We have presented a novel multimodal learning approach in battlefield object detection based on image and audio modality. Figure 5 shows the highlevel overview of the evaluation of *TinyM^{2}Net* in terms of battlefield object detection.

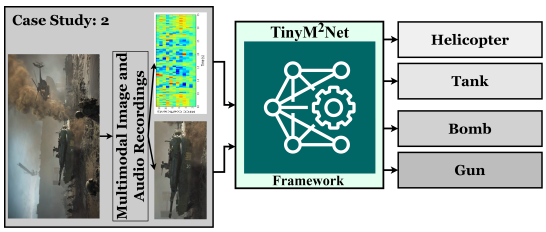

Figure 5: The proposed *TinyM^{2}Net* framework to detect four different battlefield objects, Helicopter, Tank, Bomb, Gun from two different modalities, images and audios.

5.2.1 ***Dataset Description.*** As open-sourced dataset for research on battlefield environment is very limited, we have created our own dataset for this case-study. This also contributed to the novelty of our work. 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 [2, 21, 25, 26] from where we extracted the images and corresponding audios of Helicopter, Bomb, Gun, and Tank. Sampling rate of the image extraction was one frame per second. We have collected the images in .jpg format. We extracted in total 2745 images for all the 4 classes. On the contrary, sampling frequency was 22050Hz for 1 sec audio. Length of our audio was 1 sec. We then converted the audio signal into MFCCs. We extracted in total 2745 MFCC spectrogram for 4 classes. The total number of extracted images and corresponding audios were as follows: Helicopter-1066, Gun - 1008, Bomb - 161, Tank- 510.

5.2.2 ***Experimental Setups, Results and Analysis.*** We passed both the input and corresponding audio MFCCs to *TinyM^{2}Net*. *TinyM^{2}Net* process two different modalities with its parallel CNN layers, extracts features, combines them and classify at the end as multiclass classification. We have used the 1st layer as traditional CNN and the later layers as DS-CNN. The detailed network architecture is mentioned in table 3.

We trained our model with categorical cross-entropy loss and Adam optimizer. We achieved 98.5% classification accuracy with FP32 bit precision. We then quantize our model to uniform 8-bit and 4-bit precision and achieved 97.9% and 88.7% classification accuracy. Our MP quantization technique improves the classification accuracy to 97.5% which is very comparable to both 8-bit and 32-bit quantized models. We achieved 93.6% classification accuracy with unimodal (only image data) implementation. Our multimodal approach improved the object detection accuracy to 3.9%.

Table 4: Summary of the *TinyM²Net* Framework Evaluation Results

Case Studies	Quantization	Accuracy (%)	Model Size (KB)
1	Floating Points	90.4	845
1	W8A8 (uniform 8)	89.6	216
1	**W4/8A4/8 (MP)**	**88.4**	**145**
1	W4A4 (uniform 4)	83.6	107
2	Floating Points	98.5	1605
2	W8A8 (uniform 8)	97.9	407
2	**W4/8A4/8 (MP)**	**96.8**	**269**
2	W4A4 (uniform 4)	91.3	205

Table 5: Implementation of the TinyM²Net framework to resource constrained Raspberry Pi 4 device

Case Studies	Inference Time (S)	Power (mW)
1	1.2	798
2	1.7	959

6 TinyM²Net RUNNING ON TINY DEVICES

The inference stage must be implemented on resource constrained tiny devices in order to make the *TinyM²Net* system real-time. We implemented *TinyM²Net* on Rasberry Pi 4 which has quad-core Cortex-A72 (ARM v8) and 2GB LPDDR4 memory. Performance evaluation of the *TinyM²Net* on resource constrained Raspberry Pi 4 was based on two metrics: inference time and power consumption during inference. The capacity of a framework to run in real time is determined by its inference time or running time. 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. We also need to consider the model's power profile when deploying it in the actual world. The running power of any deep model should be well within the device's sustainable range. Power consumption is calculated by deducting idle power from the peak power indicated during inference operation and reporting the result. For reporting, we use the metric unit milliwatt (mW), and a USB power meter has been employed. The inference time and required power during inference with the most compressed models are mentioned in table 5 for both of the case studies.

7 CONCLUSION

This paper presents *TinyM²Net*, a flexible system algorithm co-designed multimodal learning framework which employs as much as correlated information a multimodal dataset provides in an attempt to exploit deep learning algorithms evaluating two important tinyML evaluation case-studies: to detect the signature of COVID-19 into participents' cough and speech sound and battle field object detection from multimodal images and audios. 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 most compressed *TinyM²Net* achieves 88.4% COVID-19 detection accuracy and 96.8% battle field object detection accuracy. Finally, we test our *TinyM²Net* model on a Raspberry Pi 4 to see how they perform when deployed to a resource constrained tiny device.

8 ACKNOWLEDGMENT

We acknowledge the support of the U.S. Army Grant No. W911NF21-20076. We also acknowledge the support of the University of Maryland, Baltimore, Institute for Clinical and Translational Research (ICTR) and the National Center for Advancing Translational Sciences (NCATS) Clinical Translational Science Award (CTSA) grant number UL1TR003098.

REFERENCES

[1] Sharmeen M Saleem Abdullah Abdullah, Siddeeq Y Ameen Ameen, Mohammed AM Sadeeq, and Subhi Zeebaree. Multimodal emotion recognition using deep learning. *Journal of Applied Science and Technology Trends*, 2(02):52–58, 2021.

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

[3] Hande Alemdar, Vincent Leroy, Adrien Prost-Boucle, and Frédéric Pétrot. Ternary neural networks for resource-efficient ai applications. In *2017 international joint conference on neural networks (IJCNN)*, pages 2547–2554. IEEE, 2017.

[4] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(3):1–18, 2017.

[5] Piyush Bagad et al. Cough against covid: Evidence of covid-19 signature in cough sounds. *arXiv preprint arXiv:2009.08790*, 2020.

[6] Colby Banbury, Chuteng Zhou, Igor Fedorov, Ramon Matas, Urmish Thakker, Dibakar Gope, Vijay Janapa Reddi, Matthew Mattina, and Paul Whatmough. Micronets: Neural network architectures for deploying tinyml applications on commodity microcontrollers. *Proceedings of Machine Learning and Systems*, 3, 2021.

[7] Bettyjo Bouchey, Jill Castek, and John Thygeson. Multimodal learning. *Innovative Learning Environments in STEM Higher Education: Opportunities, Challenges, and Looking Forward*, pages 35–54, 2021.

[8] Chloë Brown et al. Exploring automatic diagnosis of covid-19 from crowdsourced respiratory sound data. *arXiv preprint arXiv:2006.05919*, 2020.

[9] Morteza Hosseini and Tinoosh Mohsenin. Qs-nas: Optimally quantized scaled architecture search to enable efficient on-device micro-ai. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 11(4):597–610, 2021.

[10] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[11] Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Improving post training neural quantization: Layer-wise calibration and integer programming. *arXiv preprint arXiv:2006.10518*, 2020.

[12] Lukasz Kaiser, Aidan N Gomez, and Francois Chollet. Depthwise separable convolutions for neural machine translation. *arXiv preprint arXiv:1706.03059*, 2017.

[13] Carl Lemaire, Andrew Achkar, and Pierre-Marc Jodoin. Structured pruning of neural networks with budget-aware regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9108–9116, 2019.

[14] Shuang Liang, Shouyi Yin, Leibo Liu, Wayne Luk, and Shaojun Wei. Fp-bnn: Binarized neural network on fpga. *Neurocomputing*, 275:1072–1086, 2018.

[15] Ji Lin, Wei-Ming Chen, Yujun Lin, John Cohn, Chuang Gan, and Song Han. Mcunet: Tiny deep learning on iot devices. *arXiv preprint arXiv:2007.10319*, 2020.

[16] Mengmeng Ma, Jian Ren, Long Zhao, Sergey Tulyakov, Cathy Wu, and Xi Peng. Smil: Multimodal learning with severely missing modality. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2302–2310, 2021.

[17] Arnab Neelim Mazumder, Jian Meng, Hasib-Al Rashid, Utteja Kallakuri, Xin Zhang, Jae-sun Seo, and Tinoosh Mohsenin. A survey on the optimization of neural network accelerators for micro-ai on-device inference. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2021.

[18] Oier Mees, Andreas Eitel, and Wolfram Burgard. Choosing smartly: Adaptive multimodal fusion for object detection in changing environments. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 151–156. IEEE, 2016.

[19] Jian Meng, Shreyas Kolala Venkataramanaiah, Chuteng Zhou, Patrick Hansen, Paul Whatmough, and Jae-sun Seo. Fixyfpga: Efficient fpga accelerator for deep neural networks with high element-wise sparsity and without external memory access. In *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*, pages 9–16. IEEE, 2021.

[20] Lara Orlandic et al. The coughvid crowdsourcing dataset, a corpus for the study of large-scale cough analysis algorithms. *Scientific Data*, 8(1):1–10, 2021.

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

[22] 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.

[23] Neeraj Sharma et al. Coswara–a database of breathing, cough, and voice sounds for covid-19 diagnosis. 2020.

[24] Vishwanath A Sindagi, Yin Zhou, and Oncel Tuzel. Mvx-net: Multimodal voxelnet for 3d object detection. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7276–7282. IEEE, 2019.

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

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

[27] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8612–8620, 2019.

[28] Bichen Wu, Yanghan Wang, Peizhao Zhang, Yuandong Tian, Peter Vajda, and Kurt Keutzer. Mixed precision quantization of convnets via differentiable neural architecture search. *arXiv preprint arXiv:1812.00090*, 2018.

[29] Xie Xiaozhu and He Cheng. Object detection of armored vehicles based on deep learning in battlefield environment. In *2017 4th International Conference on Information Science and Control Engineering (ICISCE)*, pages 1568–1570. IEEE, 2017.

[30] 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.

[31] Guangdi Zheng, Xinjian Wu, Yuanyuan Hu, and Xiaofei Liu. Object detection for low-resolution infrared image in land battlefield based on deep learning. In *2019 Chinese Control Conference (CCC)*, pages 8649–8652. IEEE, 2019.