Embedding-Assisted Attentional Deep Learning for Real-World RF Fingerprinting of Bluetooth

Anu Jagannath [1] and Jithin Jagannath [2]

[1]Affiliation not available
[2]ANDRO Computational Solutions

October 30, 2023

Abstract

A computationally efficient framework to fingerprint real-world Bluetooth devices is presented in this work. Despite the active research in this topic, a generalizable framework suitable for real-world deployment in terms of performance in a new and evolving environment as well as hardware efficiency of the architecture is lacking. We propose an embedding-assisted attentional framework (Mbed-ATN) suitable for fingerprinting actual Bluetooth devices and analyze its generalization capability in different settings and demonstrate the effect of sample length and anti-aliasing decimation on its performance. The embedding module serves as a dimensionality reduction unit that maps the high dimensional 3D input tensor to a 1D feature vector for further processing by the ATN module. Furthermore, unlike the prior research in this field, we closely evaluate the complexity of the model and test its fingerprinting capability with real-world Bluetooth dataset collected under a different time frame and experimental setting while being trained on another. Our study reveals a 7.3x and 65.2x lesser memory usage with the proposed Mbed-ATN architecture in contrast to Oracle at input sample lengths of M=10 kS and M=100 kS respectively. Further, the proposed Mbed-ATN showcases a 16.9x fewer FLOPs and 7.5x lesser trainable parameters when compared to Oracle. Finally, we show that when subject to anti-aliasing decimation and at greater input sample lengths of 1 MS, the proposed Mbed-ATN framework results in a 5.32x higher TPR, 37.9 % fewer false alarms, and 6.74x higher accuracy under the challenging real-world setting.

Embedding-Assisted Attentional Deep Learning for Real-World RF Fingerprinting of Bluetooth

Anu Jagannath, *Senior Member, IEEE,* Jithin Jagannath, *Senior Member, IEEE*

***Abstract*—A computationally efficient framework to fingerprint real-world Bluetooth devices is presented in this work. Despite the active research in this topic, a generalizable framework suitable for real-world deployment in terms of performance in a new and evolving environment as well as hardware efficiency of the architecture is lacking. We propose an embedding-assisted attentional framework (Mbed-ATN) suitable for fingerprinting actual Bluetooth devices and analyze its generalization capability in different settings and demonstrate the effect of sample length and anti-aliasing decimation on its performance. The embedding module serves as a dimensionality reduction unit that maps the high dimensional 3D input tensor to a 1D feature vector for further processing by the ATN module. Furthermore, unlike the prior research in this field, we closely evaluate the complexity of the model and test its fingerprinting capability with real-world Bluetooth dataset collected under a different time frame and experimental setting while being trained on another. Our study reveals a $7.3\times$ and $65.2\times$ lesser memory usage with the proposed Mbed-ATN architecture in contrast to Oracle at input sample lengths of $M = 10$ kS and $M = 100$ kS respectively. Further, the proposed Mbed-ATN showcases a $16.9\times$ fewer FLOPs and $7.5\times$ lesser trainable parameters when compared to Oracle. Finally, we show that when subject to anti-aliasing decimation and at greater input sample lengths of 1 MS, the proposed Mbed-ATN framework results in a 5.32$\times$ higher TPR, 37.9 % fewer false alarms, and $6.74\times$ higher accuracy under the challenging real-world setting.**

***Index Terms*—RF fingerprinting, Bluetooth, Deep learning, Embedding module, Attention mechanism**

I. Introduction

RADIO frequency (RF) fingerprint based on the hardware imperfections of the emitter circuit serves as an excellent tool or watermark to distinguish between devices manufactured by the same manufacturer even while transmitting the same message. In the present day and evolving Internet of Things (IoT) era where numerous wireless devices emerge everyday, the wireless security and the privacy of data shared across the spectrum accessed by these devices is a growing concern [1], [2]. The various approaches towards RF fingerprinting to enhance security of wireless devices that utilize wireless standards such as WiFi, Bluetooth (BT), and LoRa are an actively researched topic [3]–[6]. However, the application of deep learning (DL) especially a lightweight deployable framework that improves generalization capability for fingerprinting real-world BT devices is lacking. In [3], the authors synthetically generate WiFi signals using GNU Radio software from USRP X310 radios instead of actual IoT devices and report performance improvement only with impairments introduced at the transmitter side and significantly poor performance of 35.96 % classification accuracy without the induced impairments. Such induced perturbations mask the actual RF circuitry signatures and hinders easy adoption for use with the billions of already deployed IoT devices. The authors of [5] study WiFi-based drone detection with actual drone emitters rather than synthetically generated emissions. In [7], a 1D AlexNet and ResNet architectures are adopted to fingerprint 7 DJI M100 drones. The generalization test performed here involves training and testing on different bursts of the emission collected during the same time frame. A ZigBee emitter fingerprinting with Differential Constellation Trace Figure (DCTF) using a LeNet-5 CNN model was proposed in [8]. The ZigBee emissions are collected, trained, and tested on the same time frame and testbed setup. However, these works do not perform the generalization test where the classifiers are trained with data obtained from *a certain time frame and testbed setup* and tested on another unseen time frame and setup, quoted in our article as TTDDL scenario.

The authors of [9] conduct CNN-based WiFi fingerprinting on a custom as well as large-scale DARPA dataset using a channel equalization approach. Here, the authors perform a generalization test where only the time frame of the testing dataset is different from the training, reported in their article as Train on One Test on Another (TOTA) scenario. The authors report a accuracy of 23.2 % with their Baseline CNN model. In [10], the authors employ a triplet loss based CNN model to fingerprint base stations transmitting either of 5G New Radio, LTE, or WiFi waveforms. However, these base stations are software-defined radio (USRP B210) based rather than real-world base stations and synthentically generate the waveforms with MATLAB LTE, WLAN, or 5G toolboxes. In this work, the authors perform a generalization test by training and testing on different days. However, the multipath effect, fading, and orientation experienced by the emissions under our challenging TTDDL scenario is closer to the deployment setting faced in the real-world setting. Note that the proposed Mbed-ATN framework attains a 46.5 % accuracy in fingerprinting the challenging frequency hopping BT emitters under the TTDDL setting.

While the vast RF fingerprinting literature delves into waveforms such as ADS-B, WiFi, LoRa, and Zigbee [2], a robust DL based approach to fingerprint BT devices capable of handling unseen configuration is still lacking [6]. The core challenge stems from the rapid frequency hopping nature of the BT emitters. In this work, for the first time, we introduce an unique embedding assisted attentional framework (Mbed-

Anu Jagannath is with Northeastern University and Marconi-Rosenblatt AI/ML Innovation Laboratory, ANDRO Computational Solutions, LLC.

Jithin Jagannath is with University at Buffalo and Marconi-Rosenblatt AI/ML Innovation Laboratory, ANDRO Computational Solutions, LLC.

ATN) for fingerprinting BT emitters and evaluate it in depth.

Unlike existing literature, we comprehensively evaluate the model's complexity, prediction capability, and generalization merit. We measure the generalization power of the proposed DL model by testing the DL model with unseen data obtained from a different time frame and testbed set up compared to the training data.

Our contributions are summarized below,

- We propose for the first time, an embedding assisted attentional framework for fingerprinting BT devices.
- We collect real-world BT emissions from actual IoT devices under two indoor laboratory scenarios in a rich multipath propagation and non-line-of-sight settings.
- We demonstrate the lightweight nature of the proposed DL model in order to validate its practical deployment capability.
- We present the evaluation results of the proposed DL model in contrast to the benchmark with RF data collected under the different time frame and different experimental setup than the training data.
- Finally, we comprehensively evaluate the effects of input tensor length and anti-aliasing filtering in the prediction capability of the proposed DL model.

II. Embedding Assisted RF Fingerprint Extractor

In this section, we elaborate the design of the proposed embedding assisted RF fingerprint extractor (Mbed-ATN) which enables it to classify BT emitters in unseen challenging environment. The proposed Mbed-ATN is a deep learning framework which adopts a convolutional neural network (CNN)-based embedding module (Mbed) that serves as the feature extractor and dimensionality reduction module. The Mbed module maps the high dimensional BT signal input tensor to a one dimensional (1D) 1024×1 vector which feeds into a CNN and gated recurrent unit (GRU)-based attentional (ATN) classifier. The ATN module extracts the spatial and sequential patterns in the input vector allowing it to efficiently isolate the fingerprint from other confounding factors. This unique Mbed-ATN framework that combines the advantages of CNN and GRU in extracting the unique emitter characteristics is shown in FigXX. Emphasizing the significance of deployment of the Mbed-ATN in real-world operational scenarios, we enforce a lightweight architecture that can generalize well to the real-world environment.

Input Data Preprocessing: We denote the time domain BT signal of length N samples captured by the receiver as $y(t)|_{i=1}^{N}$. In our previous work [6] and as an ongoing study, we have empirically determined that BT emitter fingerprinting requires larger input sample lengths and additional features in the input tensor for acceptable classification accuracy. The capture length in this study is intentionally kept large enough ($N = 40$ MS) to experiment with data segmentation and other signal processing required to determine the input format that yields an acceptable fingerprinting accuracy. To keep up with the usual trend of processing short input sample lengths [9], [11], in our previous work we attempted data segmentation where the signal present regions of the captured vector are chunked in to 1024×1 inphase-quadrature (IQ) samples. However, such short signal lengths were proven to be insufficient to capture the challenging hopping pattern of the BT signal. Hence, a longer sample length with appropriate features was deemed necessary. We subject the captured BT signal to the following operations to generate a $3 \times M$ tensor.

$$\mathbf{Y}^{3\times M} = \mathfrak{F}\left[\hat{y}(t)_{t=1}^{M}\right] \tag{1}$$

$$= \begin{bmatrix} |\hat{y}(t)_{t=1}^{M}| \\ \angle\hat{y}(t)_{i=1}^{M} \\ PSD\left(\hat{y}(t)_{t=1}^{M}\right) \end{bmatrix} \tag{2}$$

where $\hat{y}(t)_{t=1}^{M}$ is the downsampled version of $y(t)_{t=1}^{N}$, the first two rows contain the magnitude and phase of the decimated signal $\hat{y}(t)$ and the third row is the power spectral density (PSD) of the decimated signal.

Embedding Module: We resort to the powerful feature extraction capability of CNNs to process the input tensor $\mathbf{Y}$. The Mbed module acts as a dimensionality reduction step in mapping the large 3D input tensor to a condensed 1D feature embedded vector. It treats the 3D input tensor as a 3-channel input and adopts 1D convolutional kernels to encode the dependencies between the adjacent samples in each input channel. The Mbed module has three convolutional layers with 1D kernels which feeds into the dense layers. The dense layers generate the 1024×1 feature embedded vector $\mathbf{f}$ for further processing by the ATN module. The architectural detail of the Mbed module is shown in Table I. We resort to using parametric ReLU (PReLU) activation function in the convolutional layers as it has shown considerable improvement when the negative values are not zeroed out citeXX. The PReLU performs non-linear mapping of an input x as in equation 3.

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ ax, & \text{if } x \leq 0 \end{cases} \tag{3}$$

Here, a is the trainable parameter and hence, the name PReLU. The dense layer utilizes ReLU activation. Unlike PReLU, the ReLU maps all negative values to 0, or in other words, when the $a = 0$ in equation 3, the function is equivalent to ReLU.

TABLE I: Architectural detail of Mbed module.

Input $3 \times M$
Conv (100,1,10) - Stride (1,10) $\forall M < 1e6$ - Stride (1,20) $\forall M \geq 1e6$
Conv (50,1,6) - Stride (1,3) $\forall M < 1e6$ - Stride (1,6) $\forall M \geq 1e6$
Maxpool (1,8) Dropout 0.5
Conv (40,1,10) - Stride (1,10) $\forall M < 1e6$ - Stride (1,5) $\forall M \geq 1e6$
Maxpool (1,5) - active for $M \geq 1e6$ Dropout 0.5
Dense 1024
Activation: Conv Layers - PReLU, Dense Layer - ReLU

Attentional module: We resort to adopting a modified version of attentional mechanism to extract the inter-dependencies in the samples and *focus only on the relevant portions of the samples* with fewer layers. Our experiments show that the adoption of attentional module can outperform deep network architectures and preempt the need for denser networks. The ATN module is a hybrid model that combines the benefits of

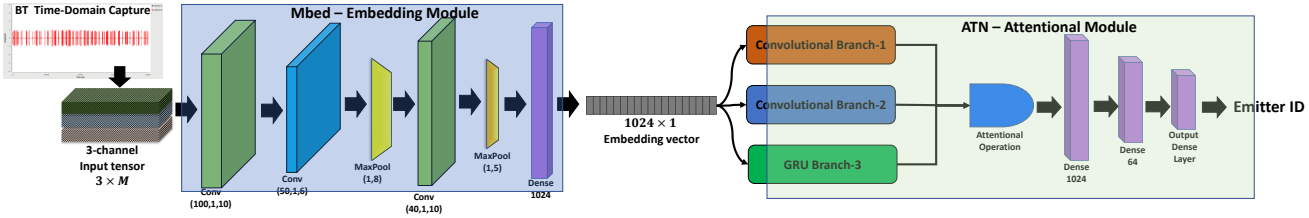


Fig. 1: Proposed Scalable Mbed-ATN framework

CNN and GRU. While the CNN performs 1D convolutions on the embedding vector $\mathbf{f}$ to capture the timing relationship of the samples, the GRU extracts the before-after timing dependencies of the samples. We consider this as a pivotal step in characterizing and comprehensively extracting the fingerprint features especially owing to the hopping nature of the BT waveform as it traverses the multipath propagation channel. GRU is an efficient form of long short term memory

TABLE II: Architectural detail of ATN module.

Input 1024×1
Branch-1: Conv (15,1,7) - Stride 1 - Padding 1 - PReLU - Dropout 0.1 Conv (32,1,7) - Stride 1 - PReLU - MaxPool (1,2) - Dropout 0.5
Branch-2: Conv (15,1,3) - Stride 1 - Padding 1 - PReLU - Dropout 0.1 Conv (32,1,3) - Stride 1 - PReLU - MaxPool (1,2) - Dropout 0.5
Branch-3: GRU hidden size = 80, #layers = 3, Dropout = 0.5
Dense 1024 - PReLU - Dropout 0.2
Dense 64 - PReLU - Dropout 0.2
Dense 10 - Softmax

(LSTM) since it uses only two gates - *Update* and *Reset* - instead of three gates as in LSTM. Further, GRU does not possess an internal memory or an output gate. Therefore, GRU uses fewer training parameters and memory and hence trains faster than LSTM. The update ($\mathbf{u}_t$) gate controls the amount of past information that needs to be carried over to the next state. The reset ($\mathbf{r}_t$) gate determines the amount of previous history that needs to be forgotten. The GRU unit are defined by the following set of equations,

$$\mathbf{u}_t = \sigma\big(\mathbf{W}_u\mathbf{x}_t + \mathbf{R}_u\mathbf{h}_{t-1} + \mathbf{b}_u\big) \tag{4}$$

$$\mathbf{r}_t = \sigma\big(\mathbf{W}_r\mathbf{x}_t + \mathbf{R}_r\mathbf{h}_{t-1} + \mathbf{b}_r\big) \tag{5}$$

$$\hat{\mathbf{h}}_t = \tanh\big(\mathbf{W}_h\mathbf{x}_t + \mathbf{R}_h\big(\mathbf{r}_t \odot \mathbf{h}_{t-1}\big) + \mathbf{b}_h\big) \tag{6}$$

$$\mathbf{h}_t = (1 - \mathbf{u}_t) \odot \mathbf{h}_{t-1} + \mathbf{u}_t \odot \hat{\mathbf{h}}_t \tag{7}$$

where $\mathbf{x}_t$ is the input vector, $\mathbf{W}_i$ and $\mathbf{R}_i$ are the weight matrices, $\mathbf{b}_i$ the bias vector, $\mathbf{h}_t$ indicates candidate hidden state, $\tanh(\circ)$ is the hyperbolic tangential activation function, and $\sigma(\circ)$ is the sigmoid activation function.

As in the architectural diagram in Fig.1, the input ($\mathbf{f}$) to the ATN module feeds into two convolutional branches, and a GRU branch. The notations $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{G}$ denote the operations of the first convolutional branch, second convolutional branch, and the GRU branch respectively. The layer details of the branches are presented in Table II. The output from the convolutional branches are vectorized (flattened) form of their respective feature maps. The GRU branch is a many-to-1 type of GRU whose output is also a vector. The operations of the ATN module are governed by the following set of equations,

$$\mathbf{o}_1 = \mathcal{C}_1\big(\mathbf{f}\big) \qquad \mathbf{s} = \mathrm{SiLU}\big(\mathbf{o}_3\big) \tag{8}$$

$$\mathbf{o}_2 = \mathcal{C}_2\big(\mathbf{f}\big) \qquad \mathbf{a} = \big[\mathbf{o}_1; \mathbf{o}_2; \mathbf{s}\big] \tag{9}$$

$$\mathbf{o}_3 = \mathcal{G}\big(\mathbf{f}\big) \tag{10}$$

where $\mathbf{s}$ is the scoring vector function approximation obtained by applying Sigmoid Linear Unit (SiLU) activation to the output from GRU branch. The SiLU activation [12] multiplies the input (x) by its sigmoid activation ($\sigma(x)$). The operator ; indicates vector concatenation. The final attentional vector $\mathbf{a}$ is generated by concatenating the outputs from the convolutional branches with the scoring vector $\mathbf{s}$ as in equation (9). This scoring vector is fed into the subsequent Dense layers for final softmax emitter classification.

Training Mbed-ATN framework: We train the end-to-end Mbed-ATN framework as in Algorithm 1. The Mbed module is initially trained to classify the emitters by adding an output softmax Dense layer to the architecture in Table I. This layer is dropped after training and the 1024×1 feature vector $\mathbf{f}$ is fed to the ATN module. The ATN module is then trained independently while keeping the weights of the Mbed module unchanged. The modules are trained for a maximum epochs of 2000 with Adam optimizer at a learning rate of 0.0001. The network convergence is monitored during the training process and the parameters are frozen at the best point of convergence.

III. Experimental Evaluation

Real world IoT Datasets: We consider real-world IoT testbed for the practical application and evaluation of the proposed RF fingerprinting framework. We collect the BT emissions from 10 IoT emitters in two challenging settings in an indoor multipath environment with other unavoidable interferences and obstacles rendering a rich multipath propagation scenario. A passive listener USRP X300 tuned into a 2 MHz bandwidth of a 2.414 GHz center frequency is streaming samples at the rate of 2 MSps. The USRP X300 is outfitted with a UBX160 daughterboard and a VERT2450 antenna affixed to the RX2 antenna port. The respective emitters are positioned to transmit the BT bursts over the duration of the capture while the receiving radio records 40 MS in one capture. The BT waveform is a challenging waveform in itself owing to the frequency hopping nature which hops at the rate of 1600 hops/second over the 2.4 GHz ISM band. This implies

Algorithm 1 Backpropagation to train Mbed-ATN framework

Train Mbed module:
Initialize network weights Θ_{Mbed}.
for `epoch = 1 to MAX_EPOCHS` **do**
  for `steps = 1 to STEPS` **do**
    Input batch $\mathbf{x}$ and **Compute** loss $\ell_{Mbed}(\Theta_{Mbed})$ [standard forward pass]
    Compute gradients $\nabla \ell_{Mbed}(\Theta_{Mbed})$
    Update weights
    $\Theta^*_{Mbed} \longleftarrow \Theta_{Mbed}$ [standard backward pass]
  end for
  Stop training once model stops learning (starts to diverge)
end for
Freeze the Mbed module with learned weights Θ^*_{Mbed}
Eliminate the output softmax Dense layer of Mbed module and feed the 1024×1 feature vector $\mathbf{f}$ to ATN module.
Train Mbed-ATN module:
Initialize network weights $\Theta^*_{Mbed}, \Theta_{ATN}$.
for `epoch = 1 to MAX_EPOCHS` **do**
  for `steps = 1 to STEPS` **do**
    Input batch $\mathbf{x}$ and **Compute** loss $\ell_{Mbed-ATN}(\Theta^*_{Mbed}, \Theta_{ATN})$ [standard forward pass]
    Compute gradients $\nabla \ell_{Mbed-ATN}(\Theta^*_{Mbed}, \Theta_{ATN})$
    Update weights of ATN module
    $\Theta^*_{ATN} \longleftarrow \Theta_{ATN}$ [standard backward pass]
  end for
  Stop training once model stops learning (starts to diverge)
end for

the signal will be periodically visiting the tuned in BT channel making it a harder waveform to capture and fingerprint. The shorter input sample lengths therefore cannot comprehensively capture the emitter characteristics and will therefore need larger sample lengths [6].

Setup 1: Here the emitters and the receiving radio are positioned in line-of-sight (LoS) settings. The separation between the emitter and receiver are varied from 1.6 ft to 9.8 ft in steps of 0.8 ft.

Setup 2: This setup is considered a challenging setting given the non-line-of-sight (NLoS) settings between the emitters and receiver. Here the emitter is placed at the four corners of the indoor laboratory while the receiver is placed at the center of the laboratory space. In this setup, the maximum separation between one of the corners and the receiver amounts to approximately 24.2 ft.

A. Key Performance Indicators

In this section, we specify the performance metrics that are used to evaluate the models. The predictions made by any DL model or in other words, the confusion matrix can be categorized into true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

1) *True positive rate (TPR) or Recall*: quantifies the positive predictions made by the model with respect to total positive predictions. For a multi-class classification, it is $TPR = \frac{\sum_i TP_i}{\sum_i (TP_i + FN_i)}$, where i denotes the class i.
2) *False positive rate (FPR)*: measures the false predictions of the model in proportion to the total false predictions. Its computed as $FPR = \sum_{i=1}^{L} \frac{FP_i/(FP_i+TN_i)}{L}$.
3) T*op-1 accuracy* (or balanced accuracy): is the arithmetic mean of the recall for each class.
4) *FLOPs*: accounts for the total number of floating point operations in the model.
5) *Model parameters*: measures the total number of trainable parameters in the model.
6) *Supported sample lengths*: the maximum measured input tensor lengths supported by the model without causing any out-of-memory (OOM) GPU errors.

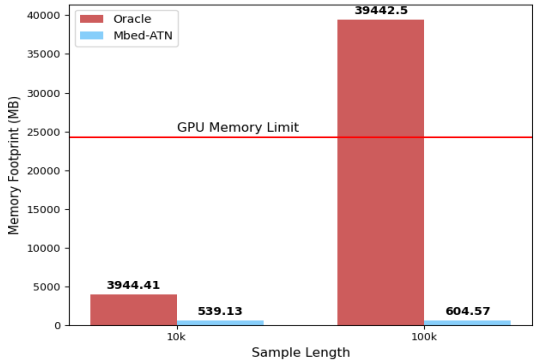
B. Complexity Analysis

Model complexity is an often overlooked factor by developers while designing and training deep learning (DL) models. According to a recent empirical study on 4960 failed DL jobs in Microsoft, 8.8% of the job failures were caused due to the depletion of GPU memory accounting for the largest category in all DL specific failures [13]. The challenging frequency hopping nature of BT waveform requires a scalable architecture that can process larger input sample lengths. This makes the model architecture challenging since it must be *large enough to process larger input samples but at the same time be lightweight for supporting commercial-off-the-shelf (COTS) deployment platforms*. In this section, we perform a systematic review of the memory footprint of the proposed Mbed-ATN model and benchmark it against Oracle [3].

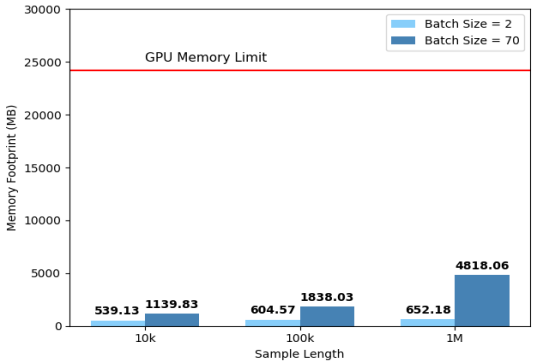
Most developers are oblivious of the memory consumption of a DL model and the steps that incur the most memory usage. Having a firm grasp of the memory usage of a model is impetus in designing efficient and lightweight DL models. In order to answer this critical practical usage question, we elucidate the maximum memory consumption of a model and demonstrate it on the proposed Mbed-ATN and Oracle models. The training of a DL model can be segmented into roughly five stages:

1) **Model Loading:** This stage involves moving the model parameters to the GPU memory. Here the current memory usage is the model memory.
2) **Forward pass:** Here the input batch is passed through the model and the intermediate activations are stored in memory for use by backpropagation. Here the current memory consumption is contributed by the model and the activations.
3) **Backward pass:** The gradients are computed from the end of the network to the beginning while discarding the saved activations during the traversal. The memory usage in this step is by the model and the gradients.
4) **Optimizer parameters:** The optimizer parameters are updated during the backpropagation. The parameters would vary depending on the type of learning algorithm such as Adam, RMSProp, etc. For example, Adam would estimate the first and second moments of the gradients. Here the memory is depleted by the model, gradients, and gradient moments.
5) **Training iterations:** Once the first iteration has passed and the optimizer has taken a step, the gradient and gradient moments are updated and stored in memory. So the maximum memory consumption in the subsequent training iterations will be in parts by the model, activations, gradients, and gradient moments.

We used the PyTorch framework and a Quadro RTX 6000 GPU in implementing and evaluating the models. Figure 2 demonstrates the GPU memory usage by Mbed-ATN model and Oracle with the same evaluation settings. In Fig.2a, we analyze the GPU memory usage when the input tensor lengths are configured to $M = 10$ kS and $M = 100$ kS. It can be seen that the memory usage rapidly scales up to trigger OOM with the Oracle model while the proposed Mbed-ATN maintains manageable and very low memory usage. This corresponds to a $7.3\times$ and $65.2\times$ lesser memory usage with Mbed-ATN architecture in contrast to Oracle at sample lengths of $M = 10$ kS and $M = 100$ kS respectively. This evaluation was carried out with a batch size of 2. A higher batch size of 70 and an input sample length $M = 1$ MS were not feasible with the Oracle model. However to provide more insight to the readers, we characterize the proposed Mbed-ATN at different batch sizes and sample lengths in Fig.2b. These analyses demonstrate the GPU memory usage with the proposed Mbed-ATN well under the GPU memory capacity.



(a) GPU Memory Consumption of training Oracle and Proposed Mbed-ATN models with different sample lengths and Batch size=2.



(b) GPU Memory Consumption of training Proposed Mbed-ATN models with different sample lengths and batch sizes.

Fig. 2: GPU Memory Consumption of training Oracle and Proposed Mbed-ATN models. The red line indicates the memory capacity of the Quadro RTX 6000 GPU.

In order to shed more light into the model complexity from a deployment standpoint, we also evaluate the floating point operations (FLOPs) and number of trainable parameters of the proposed Mbed-ATN and Oracle for an input tensor length of $M = 10$ kS (Table III). The proposed Mbed-ATN showcases a $16.9\times$ fewer FLOPs and $7.5\times$ lesser trainable parameters when compared to Oracle. These experiments demonstrate the superior lightweight nature of the proposed Mbed-ATN model while being capable of supporting larger input sample length of 1 MS. Further, it also shows the scalability limitation of Oracle architecture for an input length $M > 10$ kS.

TABLE III: FLOPs analysis with benchmark.

Model	FLOPs	#Parameters	Supported Sample length
Mbed-ATN	**2.181G**	**0.034G**	**1M**
Oracle	36.87G	0.256G	10k

TABLE IV: Fingerprinting performance at $M = 10$ kS.

Scenario	Model	TPR	FPR	Top-1 Acc.
TTSD	Mbed	**0.762**	**0.027**	**0.775**
	Mbed-ATN	0.738	0.029	0.742
	Oracle	0.738	0.029	0.742
TTDDL	Mbed	**0.158**	**0.094**	**0.145**
	Mbed-ATN	0.079	0.103	0.075
	Oracle	0.079	0.103	0.069

C. Effect of Sample length

In this section, we critically evaluate how the length of the input tensor affects the performance of the fingerprinting framework. As mentioned previously, the capture length is $N = 40$ MS which is subsequently decimated to different sample sizes (M) such as 10 kS and 1 MS. In these evaluations, we also characterize the Mbed module's fingerprinting performance separately to showcase the need for the ATN unit.

Since it is only feasible to support (for the given hardware) a sample length of $M = 10$ kS with the Oracle model, we also present its fingerprinting performance in Table IV. We measure the performance of the models when they are trained and validated with dataset collected using *Setup 1* and tested with a portion of the test set obtained from unseen data collected from the same scenario. This evaluation is presented as the Train and Test Same Day (TTSD). Under the TTSD evaluation, the proposed Mbed model outperforms both Oracle and Mbed-ATN in terms of the TPR, FPR, and Top-1 accuracy. The performance of Mbed and Mbed-ATN were measured for a sample length of 1 MS Table V. Comparing the KPIs of these evaluations under the TTSD scenario, demonstrate an increase in TPR and Top-1 accuracy while lower FPR at higher sample lengths. These evaluations further portray that the significance of ATN module comes into play at larger sample length ($M = 1$ MS). This is intuitive as the GRU branch is able to decipher the time series relation better with longer sequence lengths. Table V shows a 2.8 % higher TPR, 23 % fewer false alarms, and 0.5 % greater top-1 accuracy with Mbed-ATN in contrast to Mbed unit alone under the TTSD condition.

D. Generalization on different location and time-frame setting.

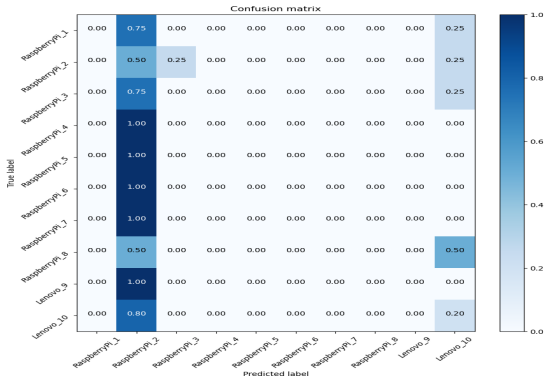
An important aspect in training and deploying a DL model for real-world applications is its generalization capability. The

TABLE V: Fingerprinting performance at $M = 1$ MS.

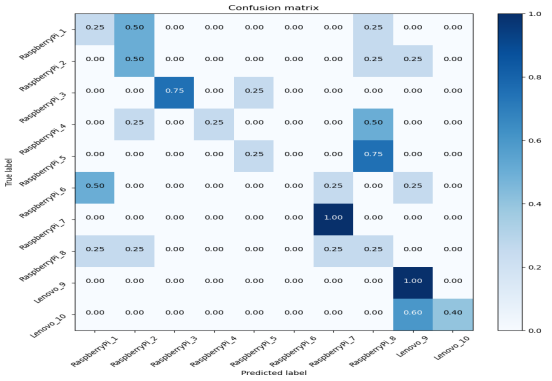
Scenario	Model	TPR	FPR	Top-1 Acc.
TTSD	Mbed	0.881	0.013	0.885
	Mbed-ATN	**0.905**	**0.01**	**0.91**
TTDDL	Mbed	0.105	0.095	0.175
	Mbed-ATN	**0.211**	**0.086**	**0.275**

fingerprinting literature has often resorted to evaluating this in terms of train one day and test another (TOTA) scenario where the DL model is trained and validated with one dataset while evaluating it with a test set collected on a different day (time frame) [9]. However, unlike the past works, we make it even more challenging by *testing on data collected from not just a different time frame but also under a different testbed setup*. We refer to this scenario as train and test different day and location (TTDDL). Under the TTDDL setting, the models are trained and validated with data collected from *Setup 1* and tested on data captured from *Setup 2*.

The TTDDL evaluation in Table IV shows 83.5 % higher TPR, 6 % lower FPR, and 2.1 % higher top-1 accuracy with the proposed Mbed unit when compared to Oracle. The significance of ATN module is depicted in Table V where it achieves 2.6 % higher top-1 accuracy in contrast to Mbed unit.



(a) Oracle performance with the maximum supported sample length



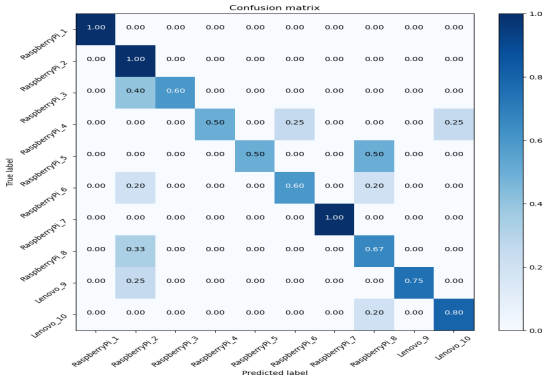
(b) Proposed Mbed-ATN architecture performance with maximum supported sample length

Fig. 3: Achievable BT fingerprinting performance on TTDDL (different day different location testbed setup)

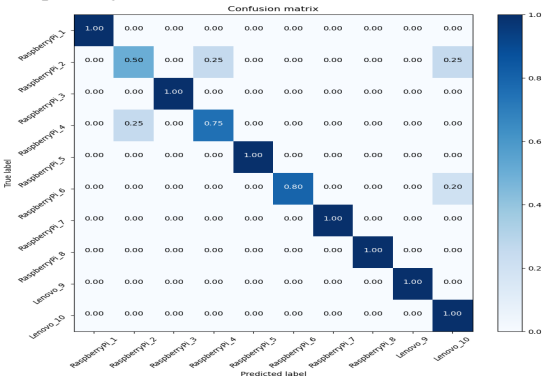
TABLE VI: Fingerprinting performance with anti-aliasing decimation (sample length = 1M).

Scenario	Model	TPR	FPR	Top-1 Acc.
TTSD	Mbed	0.905	0.011	0.905
	Mbed-ATN	0.905	0.011	0.905
TTDDL	Mbed	0.342	0.072	0.395
	Mbed-ATN	**0.421**	**0.064**	**0.465**

E. Effect of applying anti-aliasing decimation



(a) Oracle performance with the maximum supported sample length

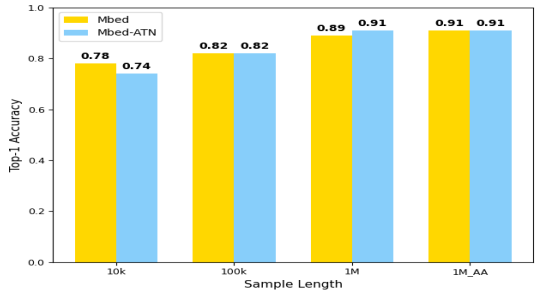


(b) Proposed Mbed-ATN architecture performance with maximum supported sample length

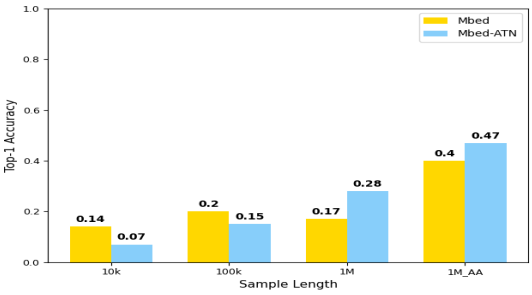
Fig. 4: Achievable BT fingerprinting performance on TTSD (same day same location testbed setup)

In this study, we evaluate the effect of anti-aliasing decimation as opposed to straightforward downsampling. For this, we decimate the 40 MS capture to 1 MS with an anti-aliasing order 8 Chebyshev type I filter. Here, the waveform is subject to anti-aliasing filtering prior to downsampling. The effects of anti-aliasing decimation is shown in Table VI. Here, we can evidently see the performance increase of the models with reference to the downsampling without anti-aliasing filtering in Table V. We show that Mbed-ATN achieves 23.1 % higher TPR, 11.1 % lesser false alarms, and 17.7 % higher accuracy compared to the Mbed module under the TTDDL setting. To truly understand, the effect of anti-aliasing, we contrast the Mbed-ATN framework's performance under the TTDDL setting in Table V and Table VI. Note the 99.5 % increase in TPR, 25.6 % drop in false alarms, and 69.1 % spike in the top-1 accuracy of the Mbed-ATN with the anti-aliasing

decimated samples. With respect to the TTDDL case in Table IV, the Mbed-ATN demonstrates a $5.32\times$ higher TPR, 37.9 % fewer false positives, and $6.74\times$ higher accuracy with the increased sample length subject to anti-aliasing filtering. In order to shed more light into this visually, we depict this increase in accuracy in Fig.5. Here, the label 1M_AA denotes a sample length of 1 MS with anti-aliasing decimation and we also show the accuracy with sample length of 100 kS. The improved generalization capability with longer sample length and anti-aliasing decimation under the challenging TTDDL setting with the adoption of ATN module can be clearly seen in Fig.5b. This study shows the combined effect of higher sample length and anti-aliasing on the performance of the proposed Mbed-ATN fingerprinting framework. Figures 3 and 4 show



(a) TTSD Scenario



(b) TTDDL Scenario

Fig. 5: Demonstrating effect of sample length and anti-aliasing decimation on Mbed-ATN fingerprinting performance.

the confusion matrices of the Oracle and proposed Mbed-ATN models at their maximum supported input tensor lengths under the TTSD and TTDDL experimental settings. The superiority of the proposed Mbed-ATN model in terms of the true positives, true negatives, false positives, and false negatives are evident in both the challenging TTDDL scenario and the TTSD setup. These evaluations validate the generalization and practical deployment capability of the proposed Mbed-ATN framework.

IV. Conclusion and Future Work

We proposed and presented a detailed analysis of Mbed-ATN, an embedding assisted attentional framework for enhancing the generalization capability of the fingerprinting architecture. The proposed model is capable of supporting large input tensor lengths of 1 MS while using significantly less GPU memory. The proposed Mbed-ATN utilizes $65.2\times$ lesser memory in contrast to the state-of-the-art Oracle architecture for an input length of $M = 100$ kS. Further, for a 10 kS sample length the Mbed-ATN utilizes $16.9\times$ fewer FLOPs and $7.5\times$ fewer trainable parameters with respect to Oracle. A detailed empirical study on the effect of higher sample length and anti-aliasing decimation was demonstrated for the proposed Mbed-ATN framework in showcasing the improved generalization capability of the model with the introduction of attentional learning. Unlike the existing literature, we resorted to the challenging different time frame and different experimental setup (TTDDL) scenario along with demonstrating the GPU efficiency of the model in validating the real-world deployment merit of the proposed framework.

References

[1] Q. Xu, R. Zheng, W. Saad, and Z. Han, "Device fingerprinting in wireless networks: Challenges and opportunities," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 94–104, 2016.

[2] A. Jagannath, J. Jagannath, and P. S. P. V. Kumar, "A comprehensive survey on radio frequency (rf) fingerprinting: Traditional approaches, deep learning, and open challenges," 2022.

[3] K. Sankhe, M. Belgiovine, F. Zhou, S. Riyaz, S. Ioannidis, and K. Chowdhury, "Oracle: Optimized radio classification through convolutional neural networks," in *Proc. of IEEE INFOCOM - IEEE Conference on Computer Communications*, 2019, pp. 370–378.

[4] N. Chen, A. Hu, and H. Fu, "Lora radio frequency fingerprint identification based on frequency offset characteristics and optimized lorawan access technology," in *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, vol. 5, 2021, pp. 1801–1807.

[5] I. Bisio, C. Garibotto, F. Lavagetto, A. Sciarrone, and S. Zappatore, "Unauthorized amateur uav detection based on wifi statistical fingerprint analysis," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 106–111, 2018.

[6] A. Jagannath, Z. Kane, and J. Jagannath, "RF Fingerprinting Needs Attention: Multi-task Approach for Real-World WiFi and Bluetooth," in *Proc. of IEEE Global Communications Conference (GLOBECOM)*, Rio de Janeiro, Brazil, December 2022.

[7] N. Soltani, G. Reus-Muns, B. Salehi, J. Dy, S. Ioannidis, and K. Chowdhury, "Rf fingerprinting unmanned aerial vehicles with non-standard transmitter waveforms," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 15 518–15 531, 2020.

[8] L. Peng, J. Zhang, M. Liu, and A. Hu, "Deep learning based rf fingerprint identification using differential constellation trace figure," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 1091–1095, 2020.

[9] A. Al-Shawabka, F. Restuccia, S. D'Oro, T. Jian, B. C. Rendon, N. Soltani, J. Dy, K. Chowdhury, S. Ioannidis, and T. Melodia, "Exposing the Fingerprint: Dissecting the Impact of the Wireless Channel on Radio Fingerprinting," *Proc. of IEEE Conference on Computer Communications (INFOCOM)*, 2020.

[10] G. Reus-Muns, D. Jaisinghani, K. Sankhe, and K. Chowdhury, "Trust in 5g open rans through machine learning: Rf fingerprinting on the powder pawr platform," in *IEEE Globecom 2020-IEEE Global Communications Conference*. IEEE, 2020.

[11] T. Jian, B. C. Rendon, E. Ojuba, N. Soltani, Z. Wang, K. Sankhe, A. Gritsenko, J. Dy, K. Chowdhury, and S. Ioannidis, "Deep learning for rf fingerprinting: A massive experimental study," *IEEE Internet of Things Magazine*, vol. 3, no. 1, pp. 50–57, 2020.

[12] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Networks*, vol. 107, pp. 3–11, 2018, special issue on deep reinforcement learning. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608017302976

[13] Y. Gao, Y. Liu, H. Zhang, Z. Li, Y. Zhu, H. Lin, and M. Yang, "Estimating gpu memory consumption of deep learning models," in *Proc. of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 1342–1352. [Online]. Available: https://doi.org/10.1145/3368089.3417050