scientific reports

Check for updates

OPEN

Optimising TinyML with quantization and distillation of transformer and mamba models for indoor localisation on edge devices

Thanaphon Suwannaphong[1]✉, Ferdian Jovan[2], Ian Craddock[1] & Ryan McConville[1]

This paper proposes small and efficient machine learning models (TinyML) for resource-constrained edge devices, specifically for on-device indoor localisation. Typical approaches for indoor localisation rely on centralised remote processing of data transmitted from lower powered devices such as wearables. However, there are several benefits for moving this to the edge device itself, including increased battery life, enhanced privacy, reduced latency and lowered operational costs, all of which are key for common applications such as health monitoring. The work focuses on model compression techniques, including quantization and knowledge distillation, to significantly reduce the model size while maintaining high predictive performance. We base our work on a large state-of-the-art transformer-based model and seek to deploy it within low-power MCUs. We also propose a state-space-based architecture using Mamba as a more compact alternative to the transformer. Our results show that the quantized transformer model performs well within a 64 KB RAM constraint, achieving an effective balance between model size and localisation precision. Additionally, the compact Mamba model has strong performance under even tighter constraints, such as a 32 KB of RAM, without the need for model compression, making it a viable option for more resource-limited environments. We demonstrate that, through our framework, it is feasible to deploy advanced indoor localisation models onto low-power MCUs with restricted memory limitations. The application of these TinyML models in healthcare has the potential to revolutionize patient monitoring by providing accurate, real-time location data while minimising power consumption, increasing data privacy, improving latency and reducing infrastructure costs.

Keywords TinyML, IoT, Indoor localisation

As the healthcare industry increasingly relies on digital data to optimise patient outcomes, the ability to accurately track and monitor individuals within indoor environments becomes vital. Precise indoor localisation enhances patient monitoring, improves safety, and provides better healthcare outcomes by ensuring timely and accurate data is available to inform medical decisions and interventions[1]. By accurately determining the location of patients, healthcare providers can monitor the elderly or those with cognitive impairments[2] to prevent wandering and ensure timely assistance in emergencies[3]. This technology also facilitates efficient asset tracking[4], optimising the use of medical equipment and reducing operational inefficiencies. Moreover, indoor localisation supports personalised healthcare[5] and after treatment monitoring[6] by collecting detailed movement patterns that can inform tailored interventions and treatments.

Typically, accurate indoor localisation systems rely on a large machine learning models deployed with a centralised remote server using data collected from low-powered battery-based edge devices such as wearables. This requires raw data to be communicated off device leading to increased latency, potential privacy concerns, and higher operational costs for the whole system. Advancements in tiny machine learning (TinyML), the field focused on deploying machine learning models on resource-constrained devices[7], have provided a means for implementing advanced machine learning models directly on resource-constrained edge devices. TinyML enables the execution of machine learning algorithms on small, low-power devices, making it possible to implement advanced functionalities like indoor localisation while adhering to stringent resource constraints typical of wearable devices and IoT sensors. This has potential in various applications including healthcare by

[1]School of Engineering Mathematics and Technology, University of Bristol, Bristol, UK. [2]School of Natural and Computing Sciences, University of Aberdeen, Aberdeen, UK. ✉email: aloe.th.suwannaphong@bristol.ac.uk

enabling more efficient, lower-cost and secure solutions for continuous patient tracking, personalised care and health monitoring.

On-device indoor localisation offers several benefits for in-home health monitoring, including enhanced privacy and security, as sensitive location data is processed locally rather than transmitted over networks. Additionally, it decreases reliance on external infrastructure, lowering costs and increasing system robustness and reliability in diverse environments. On-device processing also enables smart battery management, reducing power consumption by putting the device into sleep mode during periods of inactivity, such as when users are stationary indoors, asleep in their bedrooms, or outside the monitoring area. This smart system extends the operational longevity of monitoring devices per charge cycle. Thus, on-edge indoor localisation is a valuable technology supporting the seamless integration of digital health solutions into everyday patient care routines.

Despite the promise of TinyML, developing efficient indoor localisation models for in-home settings remains challenging. Low-power micro-controller units (MCUs) have limited memory, processing power, and battery life, which restrict the size and complexity of deployable models. This study addresses these challenges by optimising and evaluating machine learning models specifically tailored for in-home localisation on resource-constrained devices. We compare Mamba-based architectures, designed for the in-home localisation task, with state-of-the-art (SoTA) transformer models. Among these, the MDCSA Transformer model has demonstrated superior performance in noisy, real-world datasets, including in-home environments which align with the main focus of our study[8]. While MDCSA serves as a strong baseline, we also design Mamba-based architectures, which are known for their efficiency in sequential tasks and are evaluated here for the first time in an in-home indoor localisation context.

In this work, our primary focus is on developing small, efficient models for in-home indoor localisation to operate on resource-constrained devices. Unlike real-time localisation systems, which prioritise latency, our system targets healthcare applications where real-time performance is not critical. For example, in room-level localisation, there is natural latency for an individual to move between rooms, and the system can log locations using RSSI data collected a few seconds earlier. This makes memory and model size the primary constraints, rather than inference speed.

Model compression techniques are important for creating small and efficient models that can run on resource-limited devices, especially for in-home indoor localisation. Since this study focuses on reducing model size rather than improving inference speed, these techniques are helpful for meeting the memory constraints of low-power devices. We use two common methods: quantization and knowledge distillation. Quantization works by lowering the precision of the model's weights and activations, which reduces the model size while keeping its performance nearly the same[9]. Knowledge distillation helps by transferring what a large, complex model (teacher) has learned to a smaller, simpler model (student), allowing it to perform similarly with fewer resources[10]. By applying these techniques, we aim to create models that are small enough to fit on low-power MCUs while still being effective for in-home healthcare applications.

In proposing the appropriate MCU from available market options, our focus has been on minimising energy consumption to extend monitoring periods, while ensuring sufficient memory capacity to load and execute the indoor localisation model entirely within the MCU's RAM. Table 1 shows that devices with 64KB RAM or less tend to have particularly low active-mode current, making them more suitable, such as the ADuCM3029, RL78/G13, CC2650 and STM32L4. Therefore, this study aims to develop a tiny indoor localisation model with size less than 64 KB.

The primary objectives of this paper are to develop a highly efficient, small-scale indoor localisation model that fits within the memory constraints of low-power MCUs and to provide a thorough evaluation of quantization and knowledge distillation as model compression techniques to achieve this goal. We aim to evaluate the performance of a SoTA transformer model and a Mamba-based model for in-home localisation, applying the aforementioned compression techniques to optimise their sizes for deployment on resource-constrained devices.

This provides insight into the most effective strategies for achieving efficient on-device localisation within the limitations of low-power MCUs, with the aim of advancing the development of robust and energy-efficient healthcare monitoring systems that seamlessly integrate into users' daily routines. The main contributions of the paper are as follows:

MCU names	RAM (KB)	Flash (KB)	Active-mode current
2 KB RAM or less			
MSP430FR5969	2	64	1.6 mA
ATmega328P	2	32	1.5 mA
2 to 64 KB RAM			
CC2650	20	128	2.9 mA
RL78/G13	32	512	2.11 mA
ADuCM3029	64	128	0.96 mA
STM32L4	64	256	4.0 mA
More than 128 KB RAM			
MAX32630	512	2048	10.1 mA
EFM32	128	1024	10.5 mA

Table 1. Examples of low-power MCUs suitable for wearable devices.

- We propose a small and efficient model for indoor localisation suitable for resource-constrained devices by compressing state-of-the-art transformer-based models.
- We are the first to propose Mamba based architectures for indoor localisation, demonstrating that even without model compression, it is suitable for devices with less than 32 KB of memory.
- We systematically evaluate indoor localisation models for in-home datasets and large, multi-building datasets, all with model sizes under 64 KB, suitable for deployment on low-power MCUs, providing benchmarks to guide model selection based on hardware constraints.

Related work

Indoor localisation has seen significant advancements with the development of high-performing models that leverage complex algorithms and extensive computational resources[11]. Models such as those based on deep learning have shown great accuracy and robustness in determining precise indoor locations[8]. However, a major limitation of these state-of-the-art models is their size and computational demand, making them unsuitable for deployment on edge devices with limited resources. The requirement for high computational power and large memory footprints restricts their application in real-time, on-device scenarios.

Tiny Machine Learning (TinyML) has emerged as a promising solution to address the challenges of deploying machine learning models on resource-constrained devices[12]. TinyML techniques aim to reduce the size and computational requirements of models while maintaining their performance. This approach is particularly beneficial for applications like indoor localisation, particularly for healthcare purpose, where real-time processing and low power consumption are critical.

Model compression techniques like quantization, pruning, low-rank factorization, and knowledge distillation are designed to reduce model size, each with its own set of limitations. The summarised comparison of these model compression techniques is presented in Table 2. Pruning, which removes less significant weights, can create sparse models that are harder to optimise and may lose accuracy due to the removal of crucial connections[13]. Moreover, it often requires specialized hardware to support sparse operations, making it impractical for low-power micro-controllers (MCUs). Low-rank factorization, which approximates weight matrices with lower-rank versions, can struggle to retain the model's expressiveness, particularly in complex tasks, leading to potential performance drops[14]. This technique is typically more effective in large models with very high performance, rather than in smaller models like those used in this study. In contrast, quantization and knowledge distillation offer more practical solutions for TinyML applications, as they do not require specialized hardware and are more effective at reducing the size of smaller models while maintaining good performance.

Quantization and knowledge distillation are among the most popular TinyML techniques for model compression[15]. Quantization reduces model size and computational load by decreasing the precision of the numbers representing the model's parameters, which can instantly reduce the model's size by about half[16]. Its straightforward application allows for easy implementation, especially when compressing already small and efficient models for deployment on resource-constrained devices. Knowledge distillation, on the other hand, involves transferring knowledge from a larger model (the teacher) to a smaller one (the student) by training the student to mimic the behaviour of the teacher[10]. Although this process requires additional training, it offers flexibility in the size and architecture of the models, allowing the teacher to be a complex, high-performing model while the student remains a tiny, simple neural network. This flexibility is crucial for developing models that meet the stringent constraints of TinyML applications. Thus, we investigate these two techniques to achieve small and efficient indoor localisation models aimed to operate in low-power MCU devices.

There have been several attempts to apply these techniques to compress indoor localisation models for edge devices while maintaining accuracy. For instance, studies have applied quantization to develop tiny indoor localisation models on MCU devices using various models such as DNN[4,17–19], LSTM[19], and one-layer MLP[20]. Similarly, some studies have evaluated the potential of knowledge distillation to develop on-device indoor localisation models using simple neural network models such as CNN[21,22], and DNN[23,24]. While these studies successfully developed compact and efficient models for indoor localisation, they primarily used simple neural network models not specifically designed for sequence modeling tasks like RSSI localisation.

The Transformer model[25], with its core attention layer, has been highly successful in sequence modeling due to its effectiveness in handling information-dense data within a context window. This capability makes it suitable for modeling complex data. Transformers have been widely adopted in various fields, particularly in natural language processing and sequence data analysis[26]. Notably, the transformer-based model, called MDCSA, has achieved state-of-the-art performance in indoor localisation, particularly when tested on complex, real-world datasets[8]. MDCSA outperformed other leading models in time-series data tasks, as evidenced in its comparative analysis table. Given their high performance, Transformers have significant potential for developing efficient models for on-device localisation. Therefore, our study uses MDCSA as a strong baseline.

Characteristic	Quantization	KD	Pruning	Low-rank factorization
Memory reduction	High	High	High (sparsity)	Moderate
Hardware requirements	None	None	Require	None
Ease of implementation	Easy	Moderate	Moderate	Complex
Performance preservation	Moderate	High	Moderate	Low
Suitability for TinyML	High	High	Low	Low

Table 2. Comparison of model compression techniques for TinyML applications.

Another recent advancement in machine learning for sequence modeling that potentially benefits the development of TinyML models is Mamba[27], a novel class of structured state space models (SSMs). Mamba is competitive with the well-known transformer model as it combines the strengths of RNNs and CNNs to efficiently handle sequence modeling with linear scaling in sequence length. Mamba is also lightweight compared to transformers due to its SSM-based structure, which offers potential for on-device applications, particularly on resource-constrained MCU devices. It is a SoTA architecture for time-series tasks, with proven performance in various applications. For instance, Ahamed et al.,[28] showcases Mamba's efficiency, even in resource-constrained settings. To the best of our knowledge, Mamba has not been previously studied within the context of TinyML for indoor localisation. This presents a unique opportunity for our research to explore the potential of Mamba in developing an efficient, compact model for indoor localisation using RSSI data.

In this study, we propose a Mamba-based architecture tailored for indoor localisation and apply model compression techniques, comparing its performance with MDCSA, the state-of-the-art transformer-based indoor localisation model. This work addresses a gap in the TinyML field, where most existing studies focus on simpler neural networks like CNNs and DNNs. By exploring advanced machine learning models such as Mamba and Transformers, we aim to utilise their capabilities to improve the efficiency and accuracy of indoor localisation models for resource-constrained devices.

In summary, our methodology will involve Mamba and Transformer models for on-device indoor localisation using quantization, knowledge distillation, and hybrids, to achieve the best model under limited device constraints.

Datasets

Our primary focus is on in-home localisation, and we evaluate our approaches using a BLE RSSI dataset collected from four residential homes[29]. To provide additional context and assess generalisation, we also include the UJIIndoorLoc dataset, which uses Wi-Fi RSSI collected from three university buildings[30]. These two datasets complement each other, offering insights into different indoor localisation scenarios. While the in-home dataset serves as our main use case, the UJIIndoorLoc dataset offers a complementary perspective by testing the broader applicability of our approach in a multi-building, multi-floor setting. Together, these datasets enable both context-specific optimisation for in-home localisation and insights into generalisation across diverse environments. The summarised details of these datasets are presented in Table 3.

In-home localisation dataset

The in-home localisation dataset is a public dataset collected from four residential houses and uses BLE signals for indoor localisation[29]. This dataset serves the purpose of indoor positioning systems within home environments which is increasingly common for healthcare purposes[31].

The dataset includes four houses: House A (one-bedroom apartment), Houses B and D (two-bedroom, two-floor houses), and House C (the largest, three-bedroom, two-floor house). Each area within the houses is labelled at the room level, with some larger rooms classified into multiple classes (e.g., Living Area A and Living Area B). Additional classes include hallways, stairs, and outside gardens. House A, B, C, and D contain 4, 11, 9, and 10 classes, respectively.

The dataset employs wrist-worn accelerometers transmitting BLE signals at 5Hz. Raspberry Pi-based access points (APs) receive these signals and record RSSI values from the person wearing the device. The RSSI values range from -110 to 0, indicating signal strength. Each room is equipped with at least one AP for room-level localisation. Larger rooms, however, may have multiple APs strategically placed to ensure accurate signal reception throughout the space.

Data collection utilized an automated system incorporating binary floor tags placed at one-meter intervals and a chest-strapped camera capturing images as participants moved throughout the houses. This method provided precise location labels.

The dataset is organized into two experiments: *fingerprint* and *free-living*. In the fingerprint experiments, controlled experiments were conducted where participants followed scripted paths to systematically visit every area within the houses. This method ensured comprehensive coverage for training and validation purposes. Conversely, the Free-living section captured participants' daily activities within the residences, reflecting their natural movement patterns and interactions with the environment. Houses A, B, C, and D have scripted fingerprint measurements of 39, 117.8, 82.0, and 71.0 min. The corresponding unscripted living recordings are 49.6, 47.2, 237.0, and 178.4 minutes.

For model training and validation, 75% of the fingerprint data is used to train the model, with the remaining 25% used for validation. The entire free-living dataset is reserved for testing purposes, providing real-world scenarios and unseen data to evaluate the trained model's performance.

Name	Building type	No. of APs	No. of classes	Scripted data	Unscripted data	RSSI signal
House A	1-bedroom apartment	8	4	39 mins	49.6 mins	BLE
House B	2-bedroom, 2-floor house	11	11	117.8 mins	47.2 mins	BLE
House C	3-bedroom, 2-floor house	11	9	82 mins	237 mins	BLE
House D	2-bedroom, 2-floor house	11	10	97 mins	178.4 mins	BLE
UJIIndoorLoc	3 multiple-floors building	520	59	19,937 samples	1111 samples	Wi-Fi

Table 3. Summary of dataset details.

UJIIndoorLoc dataset

The UJIIndoorLoc dataset[30] was collected for evaluation of indoor positioning systems using WLAN/WiFi fingerprinting in larger spaces. It covers multiple buildings and floors at Jaume I University (UJI), serving as a reliable testbed for developing and validating precise multiple building indoor localisation models. We use this dataset to extend our evaluation beyond in-home localisation, demonstrating the efficacy of our approach across larger and multiple buildings, rather than limited to single home environments.

The UJIIndoorLoc database contains three buildings within the university, each with multiple floors. Two of the buildings have four floors each, while the third building comprises five floors. The dataset includes location information in terms of longitude, latitude, floor number, and building ID, facilitating the prediction of the exact position within the building.

The dataset was collected using 25 different Android devices, which acted as the signal transmitters. The receivers were 520 wireless access points (APs) scattered throughout the buildings. The signal type used in this dataset is WiFi, with the signal frequency ranging from 0.1 Hz to 1 Hz due to varying user activities and device usage. Signal strength is recorded as negative integer values, ranging from -104 dBm (indicating an extremely weak signal) to 0 dBm. A positive value of 100 is used to indicate instances where a wireless access point was not detected.

Data for the UJIIndoorLoc dataset were collected from more than 20 different users using two Android applications: CaptureLoc and ValidationLoc. These applications interfaced with reference map services published on an ArcGIS server, which provided detailed geographic information of the building interiors and training reference points. For the training set, 18 users captured RSSI signals from 924 reference points using the CaptureLoc application, resulting in a dataset with 19,937 samples. The validation set involved 14 users who collected data for approximately 20 minutes in each building, capturing WiFi signals randomly which might come from locations not included in the training set, resulting in 1,111 samples.

For model training and validation, 75% of the training set was used to train the model, while the remaining 25% was used to validate its performance. The validation set, consisting of 1,111 samples, was used as an independent testing set. This approach ensured that the model was tested on previously unseen data, enhancing the robustness and reliability of the localisation predictions.

Data pre-processing

In-home dataset

We handle missing values and prepare the dataset for analysis to ensure the integrity and consistency of the data. Missing data are initially addressed using forward filling with a value of the latest timestamp for 1 second given the limited movement capability within a home. Missing data at this rate is typically due to dropped packets by the hardware. Any remaining gaps were represented by -120, a value beyond the feasible range of RSSI readings, to clearly indicate missing entries (such as when the person was out of the home). We also applied a windowing technique with 4-second windows and a 50% overlap to capture temporal patterns. Lastly, min-max normalization was used to scale data values between 0 and 1, ensuring uniform feature scaling and facilitating effective model training for room classification.

UJIIndoorLoc dataset

This dataset has already managed missing data using a placeholder value of 100 to indicate undetected signals, thus removing the need for further handling of missing entries. Given the dataset's irregular sampling rate, ranging from as low as 0.1 Hz to as high as 1 Hz, we opted to forgo windowing techniques to avoid compromising the accuracy of location predictions through upsampling. Min-max normalization was applied to scale the data within a 0 to 1 range, ensuring consistent feature representation. Additionally, we assigned class labels to each area within the building, categorizing data points by specific locations and floors. As a result, the dataset consist of 59 classes throughout all floors of the three buildings. This is done to prepare the dataset for a classification task, as our objective is to classify the data points into distinct areas rather than predicting continuous values. This facilitates the localisation process and enhances the predictive modelling of spatial data.

Methodology

Model compression techniques

We apply two model compression techniques-quantization and knowledge distillation-to develop a compact and efficient indoor localisation model suitable for low-power MCU devices. Quantization is particularly advantageous due to its straightforward application, allowing for easy implementation without additional steps, especially when compressing already small models. Knowledge distillation, while requiring additional training steps, offers greater flexibility in the size and architecture of the models. This approach enables the use of a complex, high-performing teacher model to train a smaller, simpler student model, which is crucial for meeting the constraints of the target device in TinyML applications.

Quantization

Quantization is a technique used to reduce the size and computational requirements of machine learning models, making them suitable for deployment on resource-constrained devices[9]. This method works by decreasing the precision of the numbers used to represent the model's parameters, effectively reducing the memory footprint and computational complexity. They are seeing increasing use by those deploying Large Language Models (LLMs) to (still) high powered machines[32]. Nonetheless, we will use it here to apply them to tiny devices.

In a typical machine learning model, parameters are represented using 32-bit floating-point numbers (FP32). Quantization replaces these high-precision numbers with lower-precision alternatives, such as 8-bit integer (int8)

representations. This reduction in bit-width translates directly into smaller model sizes and faster computations, as lower-precision arithmetic operations require fewer computational resources.

To convert from floating-point to integer values, the original floating-point values are multiplied by a scaling factor and then rounded to the nearest whole number, as shown in Eq. (1).

$$Q = \text{round}\left(\frac{x}{\text{scale}} + \text{zero\_point}\right) \tag{1}$$

$$\text{scale} = \frac{\text{max\_float} - \text{min\_float}}{\text{max\_int} - \text{min\_int}} \tag{2}$$

$$\text{zero\_point} = \text{round}\left(-\frac{\text{min\_float}}{\text{scale}}\right) \tag{3}$$

The *scale* is a floating-point scaling factor for converting values from higher to lower precision. For example, the usual case is converting FP32 values (scale -3.4×10^{38} to 3.4×10^{38}) to int8 values (scale 0 to 255). The $zero\_point$ is an integer value that maps to the real zero in the quantized value range. It adjusts the range of the quantized values to ensure that zero in the floating-point range is accurately represented in the integer range. Various quantization approaches differ in how they determine these quantization parameters.

We apply post-training quantization to develop the TinyML model. The post-training quantization involves training the model with high precision and then converting the weights and/or activations to lower precision after training. This approach is straightforward and can be applied to pre-trained models without requiring access to the original training data, making it appropriate for those who want to take SoTA models and apply them in resource constrained settings. For our task we investigate two main types of post-training quantization: static and dynamic quantization.

Static quantization quantizes the weights and activations of the model during the model conversion process. This method requires a calibration step, where the model is run with a representative dataset to determine the optimal scaling factors for the weights and activations. We apply the LLM.int8() static quantization method[32] to achieve a TinyML model for indoor localisation. The LLM.int8() quantization method improves the performance of large-scale models by handling outlier features with 16-bit floating point (FP16) precision, while quantizing non-outlier values into 8-bit integers (int8) for memory efficiency. The process involves three main steps: extracting outliers, applying vector-wise quantization to non-outliers, and combining mixed-precision results. This approach maintains efficiency while preserving accuracy for extreme values, specifically targeting the quantization of linear layers in transformers where the weights and activations can be extreme.

In our study, we apply the LLM.int8() static quantization to compress our Transformer- and Mamba-based models for indoor localisation into a TinyML-friendly size. The LLM.int8() method is particularly beneficial for our model because it efficiently handles outlier features, which are common in the complex signal environments typical of indoor localisation tasks. By using FP16 precision for these outliers and int8 for other values, we maintain the model's accuracy while significantly reducing its size. For example, it can reduce the size of a linear layer by up to 75%, dependent on the number of outliers. We note that this method only works on linear layers, as it was originally designed for large language models that rely heavily on linear layers, making it suitable to apply to our Transformer and Mamba models which mainly consist of multiple linear layers. This approach is critical for ensuring that our models, which includes linear layers in the attention head of the Transformer and structure state space layer of the Mamba, performs reliably even with the extreme variations in signal strength encountered in diverse indoor settings.

Dynamic quantization, in contrast, only quantizes the model weights ahead of time and leaves activations in their original precision during the calibration process. At runtime, activations are dynamically quantized as they pass through the model. This reduces the size of the model weights and speeds up model execution. We use the PyTorch library for post-training dynamic quantization. This method is ideal for situations where model execution time is dominated by loading weights from memory rather than computing matrix multiplications, which is true for Transformer models with small batch sizes, such as the one used in this study.

Dynamic quantization differs from LLM.int8() as it does not treat outliers separately and is a tensor-wise process rather than vector-wise like LLM.int8(). The tensor-wise quantization scheme applies a single scale and zero-point to the entire weight matrix. Each tensor matrix has distinct quantization parameters, allowing each layer to be quantized independently based on its data distribution. This method is simpler and computationally less intensive compared to vector-wise quantization. It is typically used when uniform quantization across the entire weight matrix is sufficient, thus we compare this approach with the LLM.int8() to investigate the optimal technique to develop TinyML for indoor localisation. This dynamic quantization supports `nn.Linear`, `nn.LSTM`, `nn.RNN`, `nn.GRU`, and `nn.Embedding`, making it suitable for our models which mainly consist of linear layers.

Knowledge distillation

Knowledge distillation (KD) is a model compression technique in which a smaller, less complex model (referred to as the "student") is trained to replicate the behaviour of a larger, more complex model (referred to as the "teacher")[10]. The process involves training the student model to mimic the teacher model's predictions rather than relying on the original training data labels. This approach allows the student model to achieve similar performance levels to the teacher model while significantly reducing computational resources required for inference.

Traditional model compression techniques, such as pruning and quantization, primarily focus on reducing model size by removing redundant parameters or lowering numerical precision. While effective, these methods

may lead to a substantial loss in model accuracy, particularly for complex tasks. In contrast, knowledge distillation leverages the rich information embedded in the teacher model's output to train the student model, providing more nuanced guidance and helping the student model to generalize better. KD also offers significant flexibility in both the size and architecture of the teacher and student models, making it a superior model compression technique.

The teacher model can be any large, high-accuracy model with no constraints on its architecture, allowing the use of state-of-the-art models. The student model, in contrast, can be smaller and simpler, tailored to the specific constraints of the deployment environment, such as limited memory and computational power. This flexibility enables KD to significantly reduce the student model's size and computational complexity without substantial performance degradation. The architectures of the teacher and student models can differ, with the teacher being a complex model like a state-of-the-art transformer, and the student being a simpler neural network, such as a small transformer with a minimal attention core. This adaptability makes KD highly suitable for various use cases and deployment constraints.

We perform knowledge distillation using a distillation loss to guide the student model to mimic the behaviour of the teacher model, as demonstrated in Eq. (4):

$$L = (\alpha)L_S + (1-\alpha)L_D \tag{4}$$

Here, α is the proportion between student loss (L_S) and distillation loss (L_D). L_S is calculated from the student predictions matching the raw labels, while L_D measures the match between teacher and student predictions. Various options for distillation loss include KL divergence, MSE, cross-entropy, and cosine similarity[33]. In this study, we use cross-entropy, as it is the most suitable for our model's output.

The final layer of our model is a Convolutional Random Field (CRF) layer, which produces a class prediction as the final output without showing the probability distribution. Therefore, we select categorical cross-entropy (CCE) loss as the distillation loss. Our L_D is calculated using CCE loss between student and teacher class predictions, as shown in Eq. (5):

$$L_D = -\sum_{i=1}^{N} P_{ti} \times \log(P_{si}) \tag{5}$$

For N data points, P_{ti} is the teacher's categorical prediction and P_{si} is the student's categorical prediction for the i^{th} data point.

During training, the student learns from both the hard labels of the training data and the soft targets produced by the teacher model. We use CRF loss as the student loss and CCE loss as the distillation loss, which measures the difference between the student and teacher outputs. The final loss function is a weighted combination of the student and distillation losses, and we experiment with different proportions between these two loss functions. We found that α=0.1 provides the highest performance; therefore, the KD results presented in this study all use this α value.

Indoor localisation model

Multihead dual convolutional self-attention (MDCSA), a transformer-based model

We apply the SoTA Multihead Dual Convolutional Self-Attention (MDCSA) model developed by[8] for indoor localisation. The MDCSA model differs from typical transformers by combining convolutional layers with self-attention mechanisms, specifically designed to handle time-series data for indoor localisation. This hybrid architecture allows MDCSA to capture both local temporal patterns and long-term dependencies, making it more effective at managing multivariate features and filtering out noise compared to standard transformers.

The MDCSA model includes four main components: Positional Embedding (PE) for transforming RSSI data into spatial and temporal embeddings, Dual Convolutional Self-Attention (DCSA) for combining self-attention with causal convolutions to capture local temporal patterns, Multihead DCSA for learning diverse patterns with varied kernel sizes, and a Conditional Random Field (CRF) layer for ensuring consistent room-level predictions by considering temporal dependencies.

In this study, we aim to develop a TinyML model for indoor localisation by experimenting with two main hyperparameters that determine the model size: hidden size (H) and layer (L). The hidden size refers to the embedding size of all MDCSA components (including PE, and the linear layer inside DCSA components), while the layer represents the number of MDCSA layers. We investigate various MDCSA-based architectures by adjusting these parameters to find an optimal balance between model performance and memory constraints.

Mamba-based model

Mamba is a recent class of state space models (SSMs) designed to efficiently and effectively handle sequence modeling by applying a selective mechanism to SSMs[27]. Other architectures like transformers face computational inefficiencies and challenges in modeling long-range dependencies due to their quadratic scaling with sequence length. Mamba addresses these issues by integrating selective state spaces (selective SSM), which combine the strengths of recurrent neural networks (RNNs) and convolutional neural networks (CNNs), to achieve linear scaling in sequence length which is much less than transformer while maintaining high performance across various data modalities, particularly on dense modalities like language[34] and genomics[35]. Mamba's use of efficient computation strategies, such as kernel fusion and parallel processing, further supports its suitability for real-time processing in resource-constrained environments.Thus, Mamba is suitable to handle the information-

dense RSSI data and the lightweight SSM base makes Mamba well-suited for our application in on-device health monitoring systems, where efficient handling of sequential data and computational cost-effectiveness are keys.

The key architecture of Mamba consists of three main components: linear projection, sequence transformation, and nonlinearity. Since selective SSMs are standalone sequence transformations, they can be flexibly incorporated into neural networks. Inspired by the Transformer architecture, which typically is an interleaving of linear attention with an MLP (multi-layer perceptron) block, Mamba simplifies this by merging linear projection and sequence transformation into a single Mamba block. These blocks are stacked uniformly, as shown in Fig. 1. In this study, we use MLPs for the linear projection, convolution and selective SSM for sequence transformation, and SiLU/Swish activation and multiplication for nonlinearity. The number of Mamba layer (blocks) in the stack directly relates to the model size, so we experiment with different numbers of Mamba layers, referring to this parameter as layer (L).

In addition to the basic Mamba block, we incorporate linear embedding and CRF classification layers, as illustrated in Fig. 1, to enhance the model's performance for localisation tasks. Similar to the MDCSA model, the CRF layer is added at the end of the architecture to perform location classification due to its high capability for sequence labeling tasks. The linear embedding layer maps the raw input data into a hidden latent space with a desired dimension before processing it through the stack of Mamba blocks. This hidden dimension also directly defines the Mamba model size, so we experiment with the appropriate dimension to achieve a practically good performance under the limited constrains. We refer to this parameter as hidden size (H) similar to the MDCSA model.

Experimental setting

Our experiments will follow two stages. We will first train an effective indoor localisation model, before modifying it to work within the limited memory constraints available (e.g., via quantization or distillation).

Baseline models

This experiment involves measuring the performance of different model configurations. We train transformer and Mamba models of various sizes to discover the baseline performances without compression. Each model is trained separately for each dataset and separately for each house of the in-home dataset.

MDCSA: we investigate various MDCSA architectures by adjusting the H and L to observe the model performance and model size. Starting with the original architecture from the referenced paper[8], which uses three layers of kernel sizes 1, 4, and 7 with a hidden size of 256 (referred to as 'H256L3'), we then systematically reduce the model size to the smallest possible. We experiment with H = [256, 128, 64, 32, 16, 8, 4, 2] and L = [3, 1]. For clarity, we refer to any model with three layers of kernel sizes 1, 4, and 7 as a 'L3 model' and any model with one layer of kernel size 1 as a 'L1 model.'

Mamba: We explore different sizes of Mamba by varying the H and the L. We experiment with H = [1, 2, 4, 8, 16, 32, 64, 128], combined with L = [1, 2, 4]. For example, the smallest model consists of a hidden size of 1 and 1 Mamba layer, referred to as 'H1L1' following the naming convention. The model performances reach their limit at the hidden size 128 with 4 layers as the performance does not significantly improve much further so we did not extend the model size any larger.

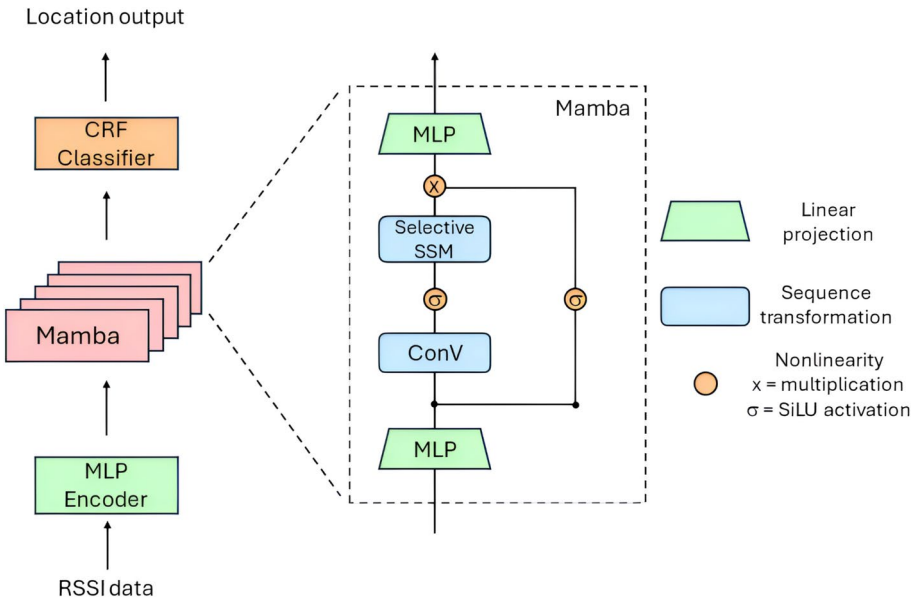

Fig. 1. Mamba architecture for RSSI classification.

Model compression
This experiment reduces the model size from the baseline model to achieve a suitable model size under the limited memory constraints.

Quantization: We develop TinyML models from the baselines using quantization technique by performing static quantization and dynamic quantization to convert the full models from FP32 precision to int8 precision. We quantize only the linear layers due to the prominence and computational expense of linear layers in Transformer- and Mamba-based models.

KD: For each dataset, we perform KD involving MDCSA and Mamba models, i.e., there are MDCSA teacher and Mamba teacher models for MDCSA student and Mamba student models, for each dataset and each house of the in-home dataset. We identify the most suitable teacher model by selecting the baseline model with the highest performance. Then, we distill the teacher's knowledge to smaller student models derived from the rest of the baseline models.

Hybrid compression combination
We also investigate the combination of KD and quantization to obtain the benefits of both techniques. KD often leads to improved performance and generalization in the student model and the student can be very tiny as KD is flexible in term of model size. Then quantization can further reduce the size of the tiny student model to significantly decrease memory usage and computational requirements. When combined, these techniques enable the creation of compact, efficient models that maintain high accuracy and performance, making them well-suited for deployment on resource-constrained devices.

Evaluations
We evaluate our experimental models using three metrics: macro F1 score (due to the imbalanced nature of the task), accuracy (to assess overall correctness across all predictions), and model size (due to the requirement for the model to fit on the device).

Overall F1 score
The F1 score measures the model's performance on overall classification, emphasizing the balance between precision and recall. It provides an average class performance across all classes, which is particularly valuable for our in-home dataset where some rooms have less training data and are more challenging to classify, yet they are crucial for accurate localisation (e.g., stairs in a home). Unlike overall accuracy, which may not adequately represent class-specific performance, the F1 score accounts for the importance of these harder-to-classify locations. We compare the F1 score between the baseline and compressed models derived quantization, KD and hybrid. The aim is to achieve an acceptable performance within the device constraints.

Accuracy
Accuracy provides a straightforward measure of how often the system correctly identifies the room, reflecting the overall correctness of predictions. While F1 score offers insights into class-specific performance, accuracy is an intuitive metric that is particularly relevant for real-time applications where consistent correctness is critical. We assess how accuracy changes across baseline and compressed models to evaluate the trade-off between memory constraints and performance. Comparing accuracy and F1 score also highlights the model's ability to balance predictions across frequently and infrequently visited rooms.

Model size
This metric is essential for developing a TinyML model, as the model needs to fit within the device's limited memory constraints. According to the acceptable memory constraints outlined in Table 1, the target sizes are either 64 KB or 32 KB. These constraints provide a good balance between RAM usage and active-mode current, which is important for a long-term indoor monitoring task. Therefore, our goal is to create models that fit within these memory limits. We measure the model size in KB and report the number of parameters. Balancing model size with performance is critical to ensure that the model can be deployed effectively on these resource-constrained devices.

Results and discussions
TinyML for in-home localisation
Our investigation into the performance of different SoTA architectures and model compression techniques for in-home localisation under the constraints of TinyML devices revealed several key insights. Overall, the results highlight the robustness of the quantized MDCSA model for in-home localisation under a 64 KB RAM constraint, while a smaller Mamba model is more suitable for environments with even more restrictive memory constraints. Quantization consistently proved to be a valuable technique in reducing model size without sacrificing performance in our task.

The MDCSA model demonstrated the highest performance within the 64 KB RAM limitation. This model achieved a size of 44 KB after quantization with an F1 score ranging from 73.84 to 84.36 across different houses, as presented in Tables 4, 5, 6 and 7. The Mamba model, although competitive in terms of both size and performance, was unable to surpass the MDCSA model unless the device constraints were extremely tight, such as less than 32 KB. Under such conditions, the Mamba model outperformed the transformer-based MDCSA model with the performance ranging from F1 score of 72.79 to 83.89 across different houses. These results suggest that the MDCSA model excels in capturing detailed patterns due to its combination of convolution and self-attention through the Multihead DCSA architecture, resulting in an exceptional performance under the acceptable memory constrain. In contrast, the simpler and more compact selective structure state space

		Baseline			Static quant			Dynamic quant			Distillation			Distill + static quant		
		F1	Acc	Size	F1	Acc	Size	F1	Acc	Size	F1	Acc	Size	F1	Acc	Size
Model name	Number of params	(%)	(%)	(KB)	(%)	(%)	(KB)	(%)	(%)	(KB)	(%)	(%)	(KB)	(%)	(%)	(KB)
Under 64 KB																
MDCSA: H16L1	10588	84.20*	97.73	64*	**84.36**	97.57	44	83.15	97.15	43	82.84*	97.52	64*	82.86	97.63	44
Mamba: H32L1	10392	82.71	**98.01**	47	79.05	96.59	26	82.18	97.74	25	77.27	97.14	47	77.32	97.05	26
Under 32 KB																
MDCSA: H8L1	2812	83.33	97.74	27	83.89	**97.93**	26	**84.76**	97.27	25	76.93	96.04	27	76.88	95.99	26
Mamba: H8L1	1432	80.49	96.44	12	79.41	97.27	12	80.46	97.77	12	82.73	97.76	12	82.95	97.78	12

Table 4. House A: baseline VS compressed models for the best MDCSA and mamba models within the limited constrains. The distillation results came from model that shows the highest baseline validation performance which are H32L4 for mamba (96.59% F1 validation, 177 KB, 40248 parameter) and H256L1 for transformer (98.31% F1 validation, 10436 KB, 2565148 parameters). The * means model size exceeds memory constraint but the compressed version of the same model is acceptable.

		Baseline			Static quant			Dynamic quant			Distillation			Distill + static quant		
		F1	Acc	Size	F1	Acc	Size	F1	Acc	Size	F1	Acc	Size	F1	Acc	Size
Model name	Number of params	(%)	(%)	(KB)	(%)	(%)	(KB)	(%)	(%)	(KB)	(%)	(%)	(KB)	(%)	(%)	(KB)
Under 64 KB																
MDCSA: H16L1	10874	74.14*	82.11	65*	73.84	**81.90**	44	**74.11**	81.06	43	74.56*	82.02	65*	73.98	81.68	44
Mamba: H16L2	7263	72.09	79.90	38	70.27	78.60	29	70.71	79.87	28	71.33	80.19	38	71.24	80.12	29
Under 32 KB																
MDCSA: H8L1	3018	67.52	77.02	28	67.75	76.95	26	67.11	76.21	25	69.12	79.19	28	69.19	79.43	26
Mamba: H8L1	1631	72.28	**80.74**	12	72.71	80.71	13	71.73	79.91	13	72.73	79.94	12	**72.79**	79.96	13

Table 5. House B: baseline VS compressed models for the best MDCSA and mamba models within the limited constrains. The distillation results came from model that shows the highest baseline validation performance which are H32L2 for mamba (88.00% F1 validation, 92 KB, 20783 parameters) and H256L1 for transformer (94.57 % F1 validation, 10447 KB, 2567834 parameters). The * means model size exceeds memory constraint but the compressed version of the same model is acceptable.

		Baseline			Static quant			Dynamic quant			Distillation			Distill + static quant		
		F1	Acc	Size	F1	Acc	Size	F1	Acc	Size	F1	Acc	Size	F1	Acc	Size
Model name	Number of params	(%)	(%)	(KB)	(%)	(%)	(KB)	(%)	(%)	(KB)	(%)	(%)	(KB)	(%)	(%)	(KB)
Under 64 KB																
MDCSA: H16L1	10796	83.42*	94.84	65*	83.44	94.75	44	**83.64**	94.03	43	83.38*	95.19	65*	83.22	**95.10**	44
Mamba: H32L1	10723	79.12	93.60	49	79.14	93.62	26	75.46	93.50	25	77.64	93.15	49	77.75	93.12	26
Under 32 KB																
MDCSA: H8L1	2956	72.91	90.87	28	73.20	91.07	26	73.17	90.70	25	78.13	92.45	28	78.51	92.52	26
Mamba: H8L1	1571	78.64	93.37	12	78.59	93.35	13	77.06	93.28	13	**80.60**	93.44	12	77.96	**93.47**	13

Table 6. House C: baseline VS compressed models for the best MDCSA and mamba models within the limited constrains. The distillation results came from model that shows the highest baseline validation performance which are H64L4 for mamba (88.88% F1 validation, 545 KB, 132259 parameters) and H256L1 for transformer (94.79% F1 validation, 10445 KB, 2567276 parameters). The * means model size exceed memory constrain but the compressed version of the same model is acceptable.

architecture of the Mamba model is better suited for stricter memory limitations, maintaining reasonable accuracy in highly constrained environments.

When considering accuracy, we observe that all models achieve higher accuracy than F1 scores across all houses, highlighting the class imbalance in the dataset. This suggests that the models frequently classify majority classes correctly, which boosts the accuracy metric, while they still face challenges in correctly classifying minority classes, as reflected in the lower F1 scores. Such class imbalance is typical for in-home data, as certain

Model name	Number of params	Baseline			Static quant			Dynamic quant			Distillation			Distill + static quant		
		F1 (%)	Acc (%)	Size (KB)	F1 (%)	Acc (%)	Size (KB)	F1 (%)	Acc (%)	Size (KB)	F1 (%)	Acc (%)	Size (KB)	F1 (%)	Acc (%)	Size (KB)
Under 64 KB																
MDCSA: H16L1	10834	75.68*	90.56	65*	75.68	**90.53**	44	**75.71**	90.41	43	74.74*	89.79	65*	74.82	89.84	44
Mamba: H32L1	10776	71.85	89.17	49	72.33	89.02	26	69.89	88.96	25	71.05	88.53	49	70.98	88.46	26
Under 32 KB																
MDCSA: H8L1	2986	63.78	83.45	28	63.55	83.47	26	64.10	83.40	25	66.13	83.42	28	66.04	83.47	26
Mamba: H16L1	3848	72.38	89.12	21	**73.17**	**89.13**	16	70.03	89.01	16	72.00	88.65	21	71.96	88.54	16

Table 7. House D: baseline VS compressed models for the best MDCSA and mamba models within the limited constrains. The distillation results came from model that show the highest baseline validation performance which are H64L4 for mamba (89.20% F1 validation, 545 KB, 132344 parameters) and H256L1 for transformer (96.62% F1 validation, 10445 KB, 2567554 parameters). The * means model size exceeds memory constraint but the compressed version of the same model is acceptable.

areas, like stairs, are less frequently visited in real-world scenarios but are critical for healthcare applications due to their vulnerability.

Additionally, we found that the model with the highest accuracy did not always achieve the highest F1 score. However, models with the highest F1 scores generally achieved accuracy values very close to the highest accuracy model, with differences often less than 1%. This finding suggests that selecting a model based on its F1 score is more appropriate for this task, given the imbalanced nature of the dataset. Prioritising F1 score ensures better overall performance across all classes, while still maintaining acceptable accuracy.

Selecting the appropriate model and compression technique based on specific memory constraints is crucial for achieving optimal performance in real-world applications. The MDCSA model benefits from its more complex architecture and ability to capture detailed temporal and spatial patterns, which is feasible within a 64 KB constraint. In contrast, the Mamba model, being inherently simpler and more compact, is better suited for environments with stricter memory limitations, where the overhead of the more complex MDCSA model becomes a challenge.

The quantization process effectively reduced the model size by nearly half while maintaining performance comparable to the baseline. This highlights the efficacy of quantization in deploying models on devices with stringent memory constraints. Conversely, knowledge distillation did not yield significant advantages. Despite expectations that this method would elevate the performance to match that of a larger teacher model, the results showed only marginal improvements or occasional declines in performance, depending on the house. Although the model size is significantly reduced from the teacher model, this technique requires long training times involving training an optimal teacher model and distill the knowledge to a suitable student model, thus it is not an efficient solution in this scenario.

In conclusion, the MDCSA model with a hidden size of 16 and a single layer, along with quantization, is identified as the optimal choice for in-home localisation when the RAM constraint is 64 KB. For devices with a more restrictive 32 KB RAM limitation, a one-layer Mamba model with a hidden size of 8 or 16 is preferable. Model compression does not significantly enhance performance nor considerably reduce the size of the Mamba model, thus it is not considered essential in these scenarios.

TinyML for large building localisation (UJIindoorLOC)

In the context of UJIindoorLoc, a dataset characterized by a large building and an extensive set of 520 APs, our study shows that the Mamba model performs best within the 64 KB RAM constraint, significantly outperforming other models, which is the opposite to the smaller in-home dataset that the MDCSA outperforms Mamba under this condition. The most effective Mamba model, utilizing knowledge distillation, achieved a size of 44 KB and an F1 score of 64% (Table 8). In comparison, the MDCSA model showed notably poorer performance, not only in terms of accuracy but also in terms of size, which exceeded 64 KB even after quantization. This can be attributed to the more challenging dataset, as MDCSA was originally designed for in-home localisation within smaller buildings where the RSSI signal is less complicated due to fewer APs and a smaller building size.

Given the expansive nature of the data and the large number of APs, models designed for UJIindoorLoc tend to be larger than those optimised for in-home localisation. Consequently, under a stricter 32 KB RAM constraint, these models performed poorly, achieving F1 scores of less than 45%. This highlights the challenge of adapting complex models to environments with significant spatial coverage and numerous data points.

Similar to the house dataset, most models in the UJIIndoorLoc dataset demonstrate higher accuracy than F1 scores, further emphasising the imbalance nature of the task. However, the MDCSA experiences a significant accuracy drop of 20-40% with static quantization, despite its F1 score remaining relatively stable. This suggests that static quantization disproportionately affects the model's ability to correctly classify majority classes, while its performance on minority classes, which dominate the F1 calculation, remains relatively unaffected. This could be attributed to the sensitivity of transformer-based architectures to precision loss in critical weights or activations that represent majority class distinctions. In contrast, dynamic quantization, which applies quantization during inference rather than pre-quantizing the model weights, shows minimal impact on accuracy. This suggests that

Model name	Number of params	Baseline			Static quant			Dynamic quant			Distillation			Distill + static quant		
		F1 (%)	Acc (%)	Size (KB)	F1 (%)	Acc (%)	Size (KB)	F1 (%)	Acc (%)	Size (KB)	F1 (%)	Acc (%)	Size (KB)	F1 (%)	Acc (%)	Size (KB)
Exceed 64 KB																
MDCSA: H16L1	23290	45.81*	63.37	115*	46.00*	19.05	68*	45.95*	63.52	67*	48.18*	65.5	115*	48.40*	18.60	68*
Under 64 KB																
Mamba: H32L1	32111	70.02*	82.97	134*	61.25	78.08	58	69.8	77.35	95	71.51*	83.65	134*	62.10	78.49	58
MDCSA: H8L1	10978	19.62	33.3	60	19.63	9.42	45	19.40	32.97	44	17.49	30.87	60	17.32	9.78	45
Mamba: H8L2	10847	63.27	**79.06**	52	56.28	72.37	40	61.45	72.12	46	63.60	78.49	52	53.65	71.72	40
Mamba: H8L1	9543	62.62	77.57	44	53.47	71.08	31	60.5	70.22	37	**64.00**	78.74	44	55.19	72.57	31
Under 32 KB																
Mamba: H4L1	6475	**44.56**	**67.21**	31	39.88	61.96	27	43.76	61.02	29	41.99	66.25	31	36.14	59.77	27
Mamba: H2L1	5020	9.16	31.47	25	8.90	29.30	26	9.22	28.89	25	9.79	31.76	26	9.02	30.53	26

Table 8. UJIIndoorLoc: baseline VS compressed models for the top performance MDCSA and mamba models within the limited constrains. The distillation results came from model that shows the highest baseline validation performance which are H128L1 for mamba (94.62% F1 validation, 783 KB, 194447 parameters) and H256L3 for transformer (97.52% F1 validation, 39400 KB, 9802058 parameters). The * means model size exceed memory constrain but the compressed version of the same model is acceptable.

dynamic quantization preserves more critical information by adapting the quantization process to runtime requirements.

Unlike the in-home dataset, where quantization proved beneficial, the reduction in model performance is more significant for the UJIindoorLoc dataset because the larger dataset is more complicated and challenging. This suggests that more techniques are required to improve the post-quantization performance, such as fine-tuning the quantize model[36], Quantization-Aware Training[37] or adaptive rounding[38]. These techniques are proven effective but require additional time and resources to implement, as they involve retraining or adjusting the model to mitigate the performance loss caused by quantization. While effective, these approaches introduce a trade-off between model complexity and development time, highlighting the need for careful evaluation when targeting highly constrained environments, especially for larger and more complex datasets like UJIindoorLoc.

While quantization did not perform well here, in contrast, knowledge distillation showed a slight improvement in model performance across most model architectures, though with some limitations. The performance enhancements did not match the large teacher models when the model size was significantly decreased. This suggests that while knowledge distillation is effective in transferring knowledge from a larger model to a smaller one, there is a trade-off between model size and performance. The student models were able to capture the general patterns learned by the teacher, but fine-grained details were often lost as the model size decreased. This highlights the need for further refinement in the distillation process, such as incorporating techniques like layer-wise distillation[39], to better retain performance in extremely small models. This represents a trade-off, where performance improvements require an extensive time investment in the knowledge distillation process.

In conclusion, for the UJIindoorLoc dataset, the one-layer Mamba model with a hidden size of 8 and knowledge distillation is the best option under the 64 KB RAM constraint. This model offers a balanced compromise between performance and size, making it suitable for deployment on MCU devices within this constraint. However, developing an efficient TinyML model within the 32 KB RAM constraint for this large dataset remains challenging. The large number of APs demand bigger model sizes to handle high-dimensional inputs, indicating that further model compression or input size reduction may be required to meet such memory constraints without compromising the performance.

Limitations of quantization for TinyML models

In this study, we observed the effectiveness and limitations of model quantization in the context of TinyML, specifically focusing on models intended for deployment on devices with highly constrained memory resources for indoor localisation. The findings indicate that quantization, while generally effective in reducing model size for larger models, can be ineffective on a tiny baseline model.

The quantization may not provide size reduction benefits when applied to models that are already very small. For instance, the mamba model of H8L1 in house A has size of 12 KB which remain about the same after quantization. This is because an initial size is close to the minimum required for deployment so there is no significant reduction in size through quantization. In some cases, quantization did not reduce the model size at all or, paradoxically, increased it, as showed in the mamba model of H8L1 in house B and C that the model size increase from 12 KB to 13 KB after quantization.

This counterintuitive result occurs due to the intrinsic overhead of the quantization process outweigh the benefits of reducing the precision of the model weights. Despite storing the model weights in a compressed int8 format, quantized models still require 32-bit floating point numbers for the scale and zero-point parameters used in dequantization during inference as shown in Fig. 2. The presence of these 32-bit parameters is necessary to accurately map the lower precision weights back to their original precision range, which is crucial for maintaining

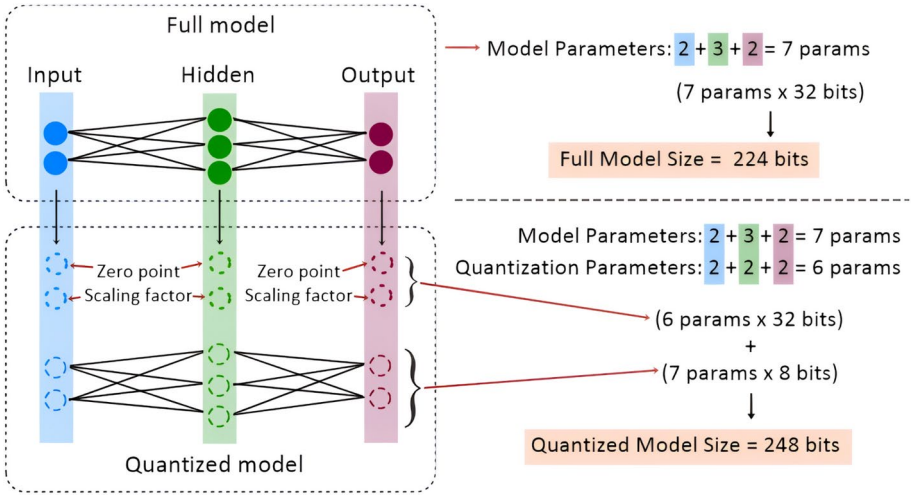

Fig. 2. Illustration shows an example case when a quantized model is bigger than the full model due to the overhead from quantization parameters. This usually occurs when quantizing a very small model, indicating the limitation of the quantization method.

the model's performance. However, in very small models, this fixed overhead can represent a significant portion of the total model size, thereby reducing the overall effectiveness of the quantization process.

For TinyML applications where the primary goal is to deploy models on devices with extremely limited memory, these findings underscore the importance of considering the initial model size and the potential overhead introduced by quantization. It highlights that while quantization can be a powerful tool for reducing model size in larger models, its benefits are not universally applicable and may not be suitable for very small models intended for deployment in highly constrained environments.

Limitations of knowledge distillation for TinyML models

KD was explored in this study to reduce model size while maintaining performance, but the results showed only limited improvements. We attribute the suboptimal performance of KD in this study to the simplicity of the distillation method employed. Specifically, our KD approach distilled knowledge only from the final layer of the teacher model, disregarding potentially valuable information from intermediate layers or the relational structure between data samples. The significant size disparity between the teacher model and the student model, limited to 64 KB or less, likely contributed to this limitation, as the distilled knowledge from the teacher's output alone may not have been sufficient to fully train the student model.

More advanced KD methods could address this limitation. Feature-based knowledge distillation[40], relation-based knowledge distillation[41], and strategies such as auxiliary architectures or adaptive distillation to reduce the performance gap[42] are promising approaches. For example, layer-by-layer distillation could enable a more comprehensive transfer of knowledge to the student model, improving its performance.

However, this study identified Mamba-based architectures and quantization as more efficient and computationally less demanding alternatives. Implementing advanced KD methods would require significant computational resources and time, particularly given the computational complexity of layer-wise distillation and the effort involved in training such systems. In contrast, Mamba with quantization offers a practical, efficient solution for TinyML applications.

While advanced KD techniques were not pursued further in this study, their potential for improving performance is acknowledged. Future research will investigate these methods to optimise the knowledge distillation process and enhance the performance of TinyML models.

Complexity and practical challenges of model deployment

Deploying machine learning models on MCU-based edge devices presents several complexities and practical challenges that must be carefully addressed. One critical factor is device compatibility, ensuring the model aligns with the hardware limitations of the target MCU, such as memory capacity, processing power, and supported data types (e.g., 8-bit integers or floating-point numbers). Resource constraints are another major challenge, as edge devices typically have limited RAM, Flash, and computational power.

Another important consideration is the handling of input data. Edge devices may collect data in formats or precision levels that require preprocessing steps to ensure compatibility with the model. For instance, raw RSSI signals may need normalisation or noise reduction before inference. Additionally, the selection of an appropriate deployment framework, such as TensorFlow Lite Micro, ONNX Runtime, or proprietary SDKs, plays a crucial role in translating trained models into formats executable on edge devices. Beyond initial deployment, scalability and maintenance pose challenges, particularly when scaling deployments to multiple devices or updating models in the field to reflect retraining efforts.

To address these challenges, advanced optimisation techniques are crucial for adapting models to resource-constrained environments, as demonstrated in this study, which achieved significant model size reduction while maintaining performance. Profiling and testing are essential to evaluate memory usage, inference time, and power consumption on the target hardware. Developing efficient, lightweight preprocessing pipelines tailored to specific data requirements can further enhance deployment efficiency. Additionally, utilising deployment frameworks like TensorFlow Lite Micro or CMSIS-NN can facilitate seamless integration into edge devices.

While this study primarily focuses on compressing the Mamba and Transformer models for RSSI-based indoor localisation, addressing deployment challenges remains an important next step. Future work will include validating these models on physical edge devices, assessing their performance under real-world operating conditions, and refining deployment strategies to ensure robustness and scalability.

Future work

While this study demonstrates the feasibility of deploying Mamba-based architectures and SoTA Transformer models for indoor localisation on resource-constrained TinyML devices, there are important limitations that need to be addressed in future work. A key limitation of this study is that all evaluations were conducted on high-performance computing environments rather than directly on physical low-power MCUs. This discrepancy matters because high-performance environments often do not accurately reflect the real-world constraints of TinyML devices, such as limited memory, processing power, and battery life. For example, inference latency and energy consumption can vary significantly between simulations on powerful hardware and actual deployment on MCUs. Furthermore, robustness under operational constraints, such as fluctuating power supply or thermal conditions, can only be effectively assessed through deployment on actual devices. Addressing these gaps in future research will be essential to validate the practical applicability and efficiency of the proposed models in real-world healthcare scenarios, ensuring they meet the stringent demands of resource-constrained settings.

This study focuses on KD and quantization to compress indoor localisation models for resource-constrained devices, but alternative techniques like structural pruning could also be considered. KD has consistently demonstrated its effectiveness in reducing model size while maintaining performance, often outperforming pruning in various scenarios[43,44], particularly at extreme compression levels[45]. Structural pruning, which removes entire neurons or channels, could potentially complement KD by further reducing model complexity. However, it introduces challenges, as structural pruning demonstrates significantly lower efficiency than unstructured pruning[46], which could be due to disruptions in the model's representational capacity caused by altering its architecture. Given these trade-offs, this study prioritised KD and quantization as practical and efficient solutions. Exploring the combination of KD with pruning could be an interesting direction for future research, potentially offering a more refined approach to model optimisation.

Expanding evaluations to include datasets with diverse building layouts and environmental conditions will also improve the generalisability of the proposed approach, ensuring its applicability across various indoor localisation use cases and healthcare settings. Furthermore, exploring a broader set of comparisons with additional models could add value to the study. Although this work primarily focuses on demonstrating the feasibility and optimisation of indoor localisation models for low-power edge devices, future research could examine additional SoTA models and methodologies to provide more comprehensive evaluations.

Conclusion

In this paper, we presented an in-depth analysis of developing TinyML models for on-device indoor localisation. Our focus was on developing models that fit within the strict memory constraints of low-power MCUs while maintaining high performance and efficiency. With model compression techniques such as quantization and knowledge distillation, we demonstrated that significant reductions in model size are possible without sacrificing, and in some cases improving, localisation performance. Our findings showed that the transformer model, after quantization, provides strong performance under the 64 KB RAM constraint, achieving an optimal balance between model size and indoor localisation performance. Moreover, the Mamba model, designed to be more compact, excelled under even tighter constraints, such as a 32 KB RAM limit, proving its effectiveness in highly resource-limited environments without model compression. The integration of on-device processing further enhances privacy, reduces latency, and lowers operational costs, making it a practical and efficient solution for continuous patient tracking, monitoring, and personalised care. This work provides a framework for converting large SoTA models into tiny models that effectively on highly resource-constrained devices.

Data availability

The In-Home Localisation dataset is available at https://doi.org/10.6084/m9.figshare.6051794.v1. The UJIIndoorLoc dataset is available at https://archive.ics.uci.edu/dataset/310/ujiindoorloc.

Code availability

Codes: https://github.com/AloeUoB/tinyML_indoor_localisation.

Received: 7 October 2024; Accepted: 12 March 2025
Published online: 24 March 2025

References

1. Bhamare, M., Kulkarni, P. V., Rane, R., Bobde, S. & Patankar, R. Tinyml applications and use cases for healthcare. In *TinyML for Edge Intelligence in IoT and LPWAN Networks*, 331–353 (Elsevier, 2024).

2. García-Requejo, A., Pérez-Rubio, M. C., Villadangos, J. M. & Hernández, Á. Activity monitoring and location sensory system for people with mild cognitive impairments. *IEEE Sens. J.* **23**, 5448–5458 (2023).
3. García-Catalá, M., Rodriguez-Sánchez, M. C. & Martín-Barroso, E. Survey of indoor location technologies and wayfinding systems for users with cognitive disabilities in emergencies. *Behav. Inf. Technol.* **41**, 879–903 (2022).
4. Avellaneda, D., Mendez, D. & Fortino, G. A tinyml deep learning approach for indoor tracking of assets. *Sensors* **23**, 1542 (2023).
5. Poyiadzi, R. et al. Detecting signatures of early-stage dementia with behavioural models derived from sensor data. arXiv preprint arXiv:2007.03615 (2020).
6. McConville, R. et al. Vesta: A digital health analytics platform for a smart home in a box. *Futur. Gener. Comput. Syst.* **114**, 106–119 (2021).
7. Kulkarni, V. & Jujare, V. Tinyml using neural networks for resource-constrained devices. In *TinyML for Edge Intelligence in IoT and LPWAN Networks*, 87–101 (Elsevier, 2024).
8. Jovan, F. et al. Multimodal indoor localisation in parkinson's disease for detecting medication use: Observational pilot study in a free-living setting. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 4273–4283 (2023).
9. Gholami, A. et al. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, 291–326 (Chapman and Hall/CRC, 2022).
10. Hinton, G., Vinyals, O. & Dean, J. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015).
11. Méndez, D., Crovo, D. & Avellaneda, D. Machine learning techniques for indoor localization on edge devices: Integrating ai with embedded devices for indoor localization purposes. In *TinyML for Edge Intelligence in IoT and LPWAN Networks*, 355–376 (Elsevier, 2024).
12. Ray, P. P. A review on tinyml: State-of-the-art and prospects. *J. King Saud Univ. Comput. Inf. Sci.* **34**, 1595–1623 (2022).
13. Zhu, M. & Gupta, S. To prune, or not to prune: exploring the efficacy of pruning for model compression. arXiv preprint arXiv:1710.01878 (2017).
14. Sainath, T. N., Kingsbury, B., Sindhwani, V., Arisoy, E. & Ramabhadran, B. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, 6655–6659 (IEEE, 2013).
15. Abadade, Y. et al. A comprehensive survey on tinyml. *IEEE Access* (2023).
16. Polino, A., Pascanu, R. & Alistarh, D. Model compression via distillation and quantization. arXiv preprint arXiv:1802.05668 (2018).
17. Hayajneh, A. M. et al. Channel state information based device free wireless sensing for iot devices employing tinyml. In *2022 4th IEEE Middle East and North Africa COMMunications Conference (MENACOMM)*, 215–222 (IEEE, 2022).
18. Kotrotsios, K., Fanariotis, A., Leligou, H.-C. & Orphanoudakis, T. Design space exploration of a multi-model ai-based indoor localization system. *Sensors* **22**, 570 (2022).
19. Girolami, M., Fattori, F. & Chessa, S. A tinyml-approach to detect the proximity of people based on bluetooth low energy beacons. In *2023 19th International Conference on Intelligent Environments (IE)*, 1–4 (IEEE, 2023).
20. Jones, B., Raza, U. & Khan, A. Tiny but mighty: Embedded machine learning for indoor wireless localization. In *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, 176–181 (IEEE, 2023).
21. Mazlan, A. B., Ng, Y. H. & Tan, C. K. A fast indoor positioning using a knowledge-distilled convolutional neural network (kd-cnn). *IEEE Access* **10**, 65326–65338 (2022).
22. Mazlan, A. B., Ng, Y. H. & Tan, C. K. Teacher-assistant knowledge distillation based indoor positioning system. *Sustainability* **14**, 14652 (2022).
23. Putrada, A. G., Alamsyah, N., Pane, S. F., Fauzan, M. N. & Perdana, D. Knowledge distillation for a lightweight deep learning-based indoor positioning system on edge environments. In *2023 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, 370–375 (IEEE, 2023).
24. Al-Ahmadi, A. Knowledge distillation based deep learning model for user equipment positioning in massive mimo systems using flying reconfigurable intelligent surfaces. *IEEE Access* (2024).
25. Vaswani, A. et al. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **30** (2017).
26. Lin, T., Wang, Y., Liu, X. & Qiu, X. A survey of transformers. *AI open* **3**, 111–132 (2022).
27. Gu, A. & Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752 (2023).
28. Ahamed, M. A. & Cheng, Q. Timemachine: A time series is worth 4 mambas for long-term forecasting. arXiv preprint arXiv:2403.09898 (2024).
29. Byrne, D., Kozlowski, M., Santos-Rodriguez, R., Piechocki, R. & Craddock, I. Residential wearable rssi and accelerometer measurements with detailed location annotations. *Sci. Data* **5**, 1–14 (2018).
30. Torres-Sospedra, J. et al. Ujiindoorloc: A new multi-building and multi-floor database for wlan fingerprint-based indoor localization problems. In *2014 international conference on indoor positioning and indoor navigation (IPIN)*, 261–270 (IEEE, 2014).
31. Farahsari, P. S., Farahzadi, A., Rezazadeh, J. & Bagheri, A. A survey on indoor positioning systems for iot-based applications. *IEEE Internet Things J.* **9**, 7680–7699 (2022).
32. Dettmers, T., Lewis, M., Belkada, Y. & Zettlemoyer, L. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Adv. Neural. Inf. Process. Syst.* **35**, 30318–30332 (2022).
33. Chitty-Venkata, K. T., Mittal, S., Emani, M., Vishwanath, V. & Somani, A. K. A survey of techniques for optimizing transformer inference. *J. Syst. Arch.* 102990 (2023).
34. Zhang, X. et al. Mamba in speech: Towards an alternative to self-attention. arXiv preprint arXiv:2405.12609 (2024).
35. Guo, J. & Schwaller, P. Saturn: Sample-efficient generative molecular design using memory manipulation. arXiv preprint arXiv:2405.17066 (2024).
36. Hubara, I., Nahshan, Y., Hanani, Y., Banner, R. & Soudry, D. Accurate post training quantization with small calibration sets. In *International Conference on Machine Learning*, 4466–4475 (PMLR, 2021).
37. Jacob, B. et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2704–2713 (2018).
38. Nagel, M., Amjad, R. A., Van Baalen, M., Louizos, C. & Blankevoort, T. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, 7197–7206 (PMLR, 2020).
39. Liang, C. et al. Less is more: Task-aware layer-wise distillation for language model compression. In *International Conference on Machine Learning*, 20852–20867 (PMLR, 2023).
40. Romero, A. et al. Fitnets: Hints for thin deep nets. arXiv preprint arXiv:1412.6550 (2014).
41. Chen, Y., Wang, N. & Zhang, Z. Darkrank: Accelerating deep metric learning via cross sample similarities transfer. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 32 (2018).
42. Yang, C., Yu, X., An, Z. & Xu, Y. Categories of response-based, feature-based, and relation-based knowledge distillation. In *Advancements in Knowledge Distillation: Towards New Horizons of Intelligent Systems*, 1–32 (Springer, 2023).
43. Peer, D., Stabinger, S., Engl, S. & Rodríguez-Sánchez, A. Greedy-layer pruning: Speeding up transformer models for natural language processing. *Pattern Recogn. Lett.* **157**, 76–82 (2022).
44. Neill, J. O., Dutta, S. & Assem, H. Deep neural compression via concurrent pruning and self-distillation. arXiv preprint arXiv:2109.15014 (2021).

45. Chen, D. et al. Epsd: Early pruning with self-distillation for efficient model compression. *In Proceedings of the AAAI Conference on Artificial Intelligence* **38**, 11258–11266 (2024).
46. Malihi, L. & Heidemann, G. Matching the ideal pruning method with knowledge distillation for optimal compression. *Appl. Syst. Innov.* **7**, 56 (2024).

Acknowledgements
T.S. is supported by the Royal Thai Government scholarship provided by the Ministry of Higher Education, Science, Research and Innovation, Royal Government of Thailand.

Author contributions
All authors contributed to the study conception and design. T.S. wrote the code, conducted the experiments, analysed the results and prepared all figures. T.S., R.M. and F.J wrote the main manuscript text. R.M and I.C. supervised this project. All authors reviewed the manuscript.

Funding
This research was funded by the TORUS project [grant EP/X036146/1].

Declarations

Competing interests
The authors declare no competing of interest.

Additional information
Correspondence and requests for materials should be addressed to T.S.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.