One Size Does Not Fit All: Multi-Scale, Cascaded RNNs for Radar Classification

DHRUBOJYOTI ROY* and SANGEETA SRIVASTAVA*, The Ohio State University, USA
ADITYA KUSUPATI, University of Washington, USA
PRANSHU JAIN, Indian Institute of Technology Delhi, India
MANIK VARMA, Microsoft Research India, India and Indian Institute of Technology Delhi, India
ANISH ARORA, The Ohio State University, USA and The Samraksh Company, USA

Edge sensing with micro-power pulse-Doppler radars is an emergent domain in monitoring and surveillance with several smart city applications. Existing solutions for the clutter versus multi-source radar classification task are limited in terms of either accuracy or efficiency, and in some cases, struggle with a trade-off between false alarms and recall of sources. We find that this problem can be resolved by learning the classifier across multiple time-scales. We propose a multi-scale, cascaded recurrent neural network architecture, MSC-RNN, comprised of an efficient multi-instance learning (MIL) Recurrent Neural Network (RNN) for clutter discrimination at a lower tier, and a more complex RNN classifier for source classification at the upper tier. By controlling the invocation of the upper RNN with the help of the lower tier conditionally, MSC-RNN achieves an overall accuracy of 0.972. Our approach holistically improves the accuracy and per-class recalls over machine learning models suitable for radar inferencing. Notably, we outperform cross-domain handcrafted feature engineering with purely time-domain deep feature learning, while also being up to ~3× more efficient than a competitive solution.

CCS Concepts: • **Computing methodologies** → **Neural networks**; *Supervised learning by classification*; • **Computer systems organization** → **Sensor networks**; **Real-time system architecture**; *System on a chip.*

Additional Key Words and Phrases: Radar classification, recurrent neural network, low power, edge sensing, range, joint optimization, real-time embedded systems

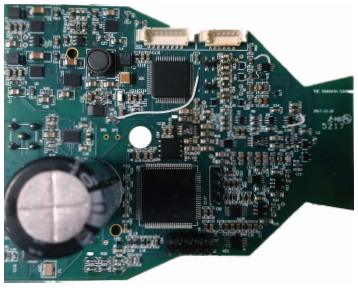
ACM Reference Format:
Dhrubojyoti Roy, Sangeeta Srivastava, Aditya Kusupati, Pranshu Jain, Manik Varma, and Anish Arora. 2020. One Size Does Not Fit All: Multi-Scale, Cascaded RNNs for Radar Classification. *ACM Trans. Sensor Netw.* 37, 4, Article 111 (August 2020), 27 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Both authors contributed equally to this research.

This submission is a journal extension of the ACM BuildSys 2019 publications [35, 38] by the same authors. Differences from the aforementioned publications are highlighted in Section 1.

Authors' addresses: Dhrubojyoti Roy, roy.174@osu.edu; Sangeeta Srivastava, srivastava.206@osu.edu, The Ohio State University, 2015 Neil Avenue, Columbus, Ohio, USA, 43210; Aditya Kusupati, kusupati@cs.washington.edu, University of Washington, 185 E Stevens Way NE, Seattle, Washington, 98195, USA; Pranshu Jain, anz178419@cse.iitd.ac.in, Indian Institute of Technology Delhi, Hauz Khas, New Delhi, 110016, India; Manik Varma, manik@microsoft.com, Microsoft Research India, 9, Vigyan 1st floor, Lavelle Road, Bengaluru, Karnataka, 560001, India , Indian Institute of Technology Delhi, Hauz Khas, New Delhi, 110016, India; Anish Arora, arora.9@osu.edu, The Ohio State University, 2015 Neil Avenue, Columbus, Ohio, USA , 43210, The Samraksh Company, 5980 Venture Dr, Dublin, Ohio, USA, 43017.

1550-4859/2020/8-ART111 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn



(a) Micro-power pulse-Doppler radar system

(b) Solar harvested Signpost platform supporting low power sensors [3]

Fig. 1. The micro-power pulse-Doppler radar (PDR) device can be independently deployed or interfaced with existing multi-sensor smart city platforms such as Signpost (figure adapted from [3], Copyright ©2019 ACM, Inc.)

1 INTRODUCTION

With the rapid growth in deployments of Internet of Things (IoT) sensors in smart cities, the need and opportunity for computing increasingly sophisticated sensing inferences on the edge has also grown. This has motivated several advances in designing resource-efficient sensor inferences, particularly those based on machine learning and especially deep learning. The designs, however, encounter a basic tension between achieving efficiency while preserving predictive performance that motivates a reconsideration of state-of-the-art techniques.

In this paper, we consider a canonical inference pattern, namely discriminating clutter from several types of sources, in the context of a radar sensor. This sort of N+1−class classification problem, where N is the number of source types, has a variety of smart city applications, where diverse clutter is the norm. These include triggering streetlights smartly, monitoring active transportation users (pedestrians, cyclists, and scooters), crowd counting, assistive technology for safety, and property surveillance. As an example, streetlights should be smartly triggered on for pedestrians but not for environmental clutter such as trees moving in the wind. Similarly, property owners should be notified only upon legitimate intrusions but not for passing animals.

The radar is well-suited in the smart city context as it is privacy preserving in contrast to cameras. Moreover, it consumes low power (~15mW), because of which it can be deployed at operationally relevant sites with little dependence on infrastructure, using, for instance, a small panel solar harvester or even a modest sized battery, as shown in Figure 1. Experiences with deploying sensors in visionary smart city projects such as Chicago's Array of Things [6, 42] and Sounds of New York City [5] have shown that wired deployments on poles tend to be slow and costly, given constraints of pole access rights, agency coordination, and labor unions, and can sometimes be in suboptimal locations. Using a low-power sensor that is embedded wirelessly or simply plugged in to existing platforms while imposing only a nominal power cost simplifies smart city deployment.

Table 1 illustrates an efficiency-accuracy trade-off for the canonical inference pattern with $N = 2$, wherein clutter is distinguished from human and other (i.e., non-human) sources. The more accurate deep models, the Convolutional Neural Network (CNN) [24] and the Long Short-Term Memory (LSTM) [18], that we machine-learned for this 3-class classifier from a reference dataset are significantly less efficient, in terms of speed and therefore power consumption. In contrast, the more efficient shallow solution, Support Vector Machine (SVM), is significantly less accurate. While the SVM classifier has been implemented to operate in near real-time on the Cortex-M3 single-microcontroller processor in the device depicted in Fig. 1(a), neither the CNN nor the LSTM per se yield a near real-time implementation. To implement deep models in near real-time on the M3, we therefore consider model optimization with recent state-of-art-techniques such as fast gated RNNs (FastGRNN)

Table 1. Trade-offs in accuracy and runtime efficiency for the 3-class radar problem (window length 1s, feature computation overhead ignored for SVM, dataset and machine architecture details are in Section 5)

ML Model	Accuracy	FLOPS	Fits on Cortex-M3?
SVM (15 features)	0.85	37K	Yes
LSTM	0.89	100K	No
CNN (1s FFT)	0.91	1.3M	No
EMI-LSTM	0.90	20K	No
EMI-FastGRNN	0.88	8K	Yes

[28] and Early-exit Multi-Instance RNNs (EMI-LSTM and EMI-FastGRNN) [10]. However, Table 1 illustrates that the trade-off remains: the best accuracy we achieve, namely with FastGRNN, has significantly lower efficiency than the best efficiency achieved, namely with EMI-FastGRNN, but the latter's accuracy is comparatively worse.

Problem Statement. In this work, we investigate alternative optimizations of deep models for the above classification task that achieve both high accuracy and speed. In doing so, we do not wish to sacrifice the recall performance for achieving high precision. For instance, radar sensing applications require that the clutter recall be very high so that there are minimal false alarms. However, a solution that restricts false alarms at the cost of detectability (i.e., low source recall, where a source could be either human or non-human) would be undesirable as it would have limited applicability in the smart city contexts discussed above.

Solution Overview. The $N+1$-class radar problem, where the +1-class is clutter, conflates discrimination between classes that are conceptually different. In other words, discriminating clutter from sources has a different complexity from that of disambiguating source types. This insight generalizes when the sources themselves are related by a hierarchical ontology, wherein different levels of source types involve concepts of correspondingly different complexity of discrimination. By way of example, in the 3-class clutter vs. human vs. non-human classification problem, discriminating clutter from sources turns out to be simpler than discriminating the more subtle differences between the source types. Using the same machine architecture for 3 classes of discrimination leads to the accuracy-efficiency trade-off, as the last two rows of Table 1 indicate. A more complex architecture suffices for discriminating among source types accurately, whereas a simpler architecture more efficiently suffices for discriminating clutter from sources, but hurts the accuracy of discriminating between source types.

We, therefore, address the problem at hand with an architecture that decomposes the classification inference into different hierarchical sub-problems. For the 3-class problem, these are: (a) Clutter vs Sources, and (b) Humans vs. Non-humans *given* Sources. For each sub-problems we choose an appropriate learning architecture; given the results of Table 1, both architectures are forms of RNN albeit *with learning at different time-scales*. The lower tier RNN for (a) uses a short time-scale RNN, the Early-exit Multi-Instance RNN (EMI-FastGRNN) [10, 28], whereas the higher tier for (b) uses a longer time-scale RNN, a FastGRNN [28], which operates at the level of windows (contiguous, fixed-length snippets extracted from the time-series) as opposed to short instances within the window. The upper tier uses the features created by the lower tier as its input; for loss minimization, both tiers are jointly trained. To further improve the efficiency, we observe that source type discrimination needs to occur only when a source is detected and clutter may be the norm in several application contexts. Hence, the less efficient classifier for (b) is invoked only when (a) discriminates a source: we refer to this as cascading between tiers. The joint training loss function is refined to emulate this cascading. We call this architecture *Multi-Scale, Cascaded RNN (MSC-RNN).*

Contributions. Our proposed architecture exploits conditional inferencing at multiple time-scales to jointly achieve superior sensing and runtime efficiency over state-of-the-art alternatives. To the best of our knowledge,

this approach is novel to deep radar systems. For the particular case of the 3-class problem, MSC-RNN performs as follows on the Cortex-M3:

Accuracy	Clutter Recall	Human Recall	Non-human Recall	FLOPS
0.972	1	0.92	0.967	9K

Its accuracy and per-class recalls are mostly better than, and in remaining cases competitive with, the models in Table 1. Likewise, its efficiency is competitive with that of EMI-FastGRNN, the most efficient of all models, while substantially outperforming it in terms of sensing quality. We also validate that this MSC-RNN solution is superior to its shallow counterparts not only comprehensively, but at each individual tier as well. The data and training code for this project are open-sourced at [36, 37].

Other salient findings from our work are summarized as follows:

(1) Even with deep feature learning purely in the time-domain, MSC-RNN surprisingly outperforms handcrafted feature engineering in the amplitude, time, and spectral domains for the source separation sub-problem. Further, this is achieved with 1.75-3× improvement in the featurization overhead.
(2) The Tier 1 component of MSC-RNN, which classifies legitimate sources from clutter, improves detectability by up to 2.6× compared to popular background rejection mechanisms in radar literature, even when the false alarm rate is controlled to be ultra-low.
(3) MSC-RNN seems to tolerate the data imbalance among its source types better than compared EMI-RNN models. In particular, it enhances the non-dominant human recall by up to 20%, while simultaneously maintaining or improving the dominant non-human recall and overall accuracy.

In contrast to [35, 38], we also provide the following additional evaluations of our solution:

(1) We show that the MSC-RNN joint optimization strategy improves the non-dominant class (human) recall over competitive training approaches such as independent training of the two tiers by up to 21% without compromising other metrics. We also show that pre-training the two tiers prior to commencing the joint training step is crucial for achieving a much better optimization of the loss objective.
(2) Through a series of regression experiments, we bolster our claim that deep feature learning in the time domain alone can essentially replace feature handcrafting for radar classification. We show that the upper tier of MSC-RNN is able to faithfully approximate the top engineered features across time, amplitude, and frequency domains from the raw radar time series, with an MAE of 0.09−0.13, and an MSE of 0.02−0.03.
(3) We also discuss how the non-dominant class recalls can be further improved in Tier 2 by strengthening the end softmax layer post-hoc with a resource-efficient and powerful tree classifier, Bonsai [25], at no additional storage overhead.

Organization. In Section 2, we present related research and outline the basics of micro-power radar sensing in Section 3. In Section 4, we detail the various components in our solution and discuss the training and inference pipelines. We evaluate our solution extensively in Section 5, and provide pertinent discussion including implementation details in Section 6.3 respectively. We conclude and motivate future research in Section 7.

2 RELATED WORK

Shallow and Deep Radar Sensing. Micro-Doppler features have been used in myriad radar sensing applications ranging from classification [16, 21, 29] to regression [15]. Most of these applications employ the short-time Fourier transform (STFT) input representation for analyzing micro-Doppler signatures. Although shallow classifiers can be computationally cheaper than deep solutions, the spectrogram generation over a sliding window for the STFT incurs significant computational overhead for real-time applications on single microcontroller devices. In order

to decrease the computational overhead, different feature extraction methods [12, 19] have been investigated in the past. Notably, feature engineering not only requires sophisticated domain knowledge, but also may not transfer well to solutions for other research problems. Moreover, selection of relevant and non-redundant features requires care for sensing to be robust [34]. In recent years, there has been significant use of deep learning in radar applications. Most works with architectures like CNNs and autoencoders use spectrogram-based input [20, 22]. The authors of [30] generate a unique spectral correlation function for the Deep Belief Network to learn signatures from. The pre-processing needed in these applications and the resulting model sizes make them unsuitable for single microcontroller devices. Therefore, we use raw time-series data in conjunction with RNN variants to make our deep learning solution faster and more efficient in addition to avoiding spectrogram computation altogether.

Efficient RNN. The ability of RNNs in learning temporal features makes them ubiquitous in various sequence modeling tasks. They, however, often suffer from training instabilities due to the exploding and vanishing gradient problem (EVGP) [32]. Gated RNNs like LSTMs [18] and GRUs [8] circumvent EVGP and achieve the desired accuracy for a given task, but compromise on model sizes and compute overheads, making them unattractive for real-time, single microcontroller implementations. Recently, FastGRNN [28] has been proposed to achieve prediction accuracies comparable to gated RNNs while ensuring that the learned models are significantly smaller for diverse tasks. The hierarchical classifier solution we present in this paper is based on this architecture.

Multi-Instance Learning and Early Classification. MIL is a weakly supervised learning technique that is used to label sub-instances of a window. MIL has found use in applications from vision [43] to natural language processing (NLP) [23]. It enables a reduction in the computational overhead of sequential models like RNNs by localizing the appropriate activity signature in a given noisy and coarsely-labeled time-series data along with early detection or rejection of a signal [10]. We use it as our lower-tier classifier for clutter versus source discrimination.

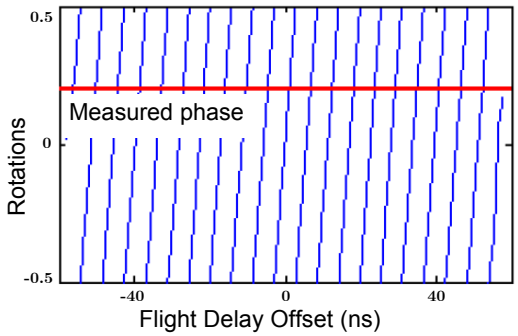
Multi-Scale RNN. One of the early attempts to learn structure in temporally-extended sequences involved using reduced temporal sequences [17] to make detectability over long temporal intervals feasible in recurrent networks [31, 39]. With the resurgence of RNNs, multi-scale recurrent models are being used to discover the latent hierarchical multi-scale structure of sequences [9]. While they have been traditionally used to capture long-term dependencies, we use them to design a computationally efficient system by incorporating different scales of temporal windows for the lower- and upper-tier RNNs. By conditioning the Tier 2 classifier, which works on longer windows and is hence bulkier, we make sure that the former is invoked only when necessary, i.e., when Tier 1 predicts a source.

Compression Techniques. Sparsity, low-rank, and quantization have been proven to be effective ways of compressing deep architectures like RNNs [27, 44] and CNNs [14, 26]. We incorporate low-rank representation, Q15 quantization, and piecewise-linear approximation [28] to make MSC-RNN realizable on Cortex-M3 microcontrollers.

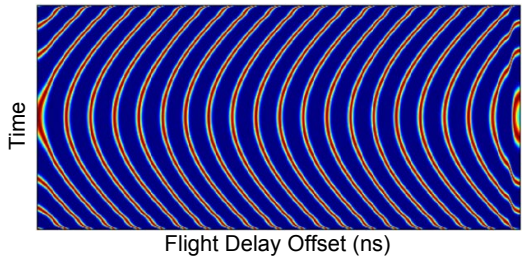
3 RADAR AND CLASSIFIER MODELS

3.1 Micro-power Radar Model

The monostatic pulsed Doppler radar (PDR) sensor [40] in Figure 1(a) has a pulse repetition frequency of 3MHz, bandwidth of nearly 100 MHz, a center frequency at about 5.8 GHz, and power consumption of 38 mW. Its radiation pattern is rather omnidirectional, yielding a maximum detection range of ~13 m. It detects movements with radial velocities between 2.6 cm/s and 2.6 m/s; this range encompasses several kinds of motion such as human walks, animal gaits (walking, ambling or trotting), and slow-moving vehicles. The on-board radar antenna radiates with a 60° conical coverage in the z-plane, allowing the radar to be tethered to tree branches or posts at a



(a) Possible ranges for a given phase



(b) Evolution of likelihood for a passing target

Fig. 2. Relative, but not absolute, ranging is possible over the likely trajectories for a target passing through the radar-mote field of view.

modest height. A notable radar artifact is its anisotropic antenna pattern, i.e., different lobes have different signal return strengths with the front lobe typically being the strongest direction. It is thus possible that the return from a larger target (e.g., human) on a weaker lobe is comparable to that from a smaller target (e.g., dog) on a stronger lobe. The radar response is low pass filtered to 100 Hz; hence the output is typically sampled at rates over 200Hz (in our experimental setting, at 256 Hz).

The output signal from the radar is a complex value, i.e., with in-phase (I) and quadrature (Q) components. The radar return depends upon all reflecting objects in the scene, which include not only displacing foreground targets but also clutter that is static or moving in-situ such as trees and bushes. When a target moves within the detection range, the phase of the complex signal changes according to the direction of motion (specifically, the phase decreases as the target moves away from the radar and increases as it approaches the radar). Notably, relative but *not* absolute ranging of the source is possible: this is shown in Figure 2(a), where a measured output phase (indicated by the red line) places the source at any one of many possible locations consecutively separated by an order of wavelength, with rather low likelihood of being in between (Figure 2(b)). Consequently, its relative displacement can be estimated with high accuracy (typically, sub-cm scale), and a rich set of features in the amplitude, time, and frequency domains can be derived by tracking its phase evolution over time.

3.2 Classifier Architectures

3.2.1 Input and Feature Representation. The radar classifier system uses the aforementioned complex time-series as input. Extant end-to-end architectures for micro-power radar sensing mostly eschew deep feature learning

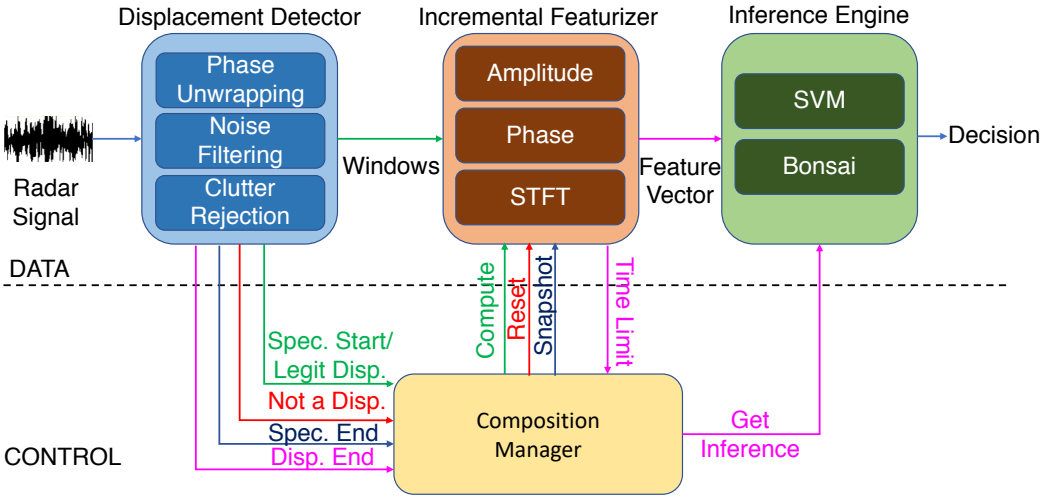


Fig. 3. SVM classifier data and control planes; control signal-response pairs are color coded

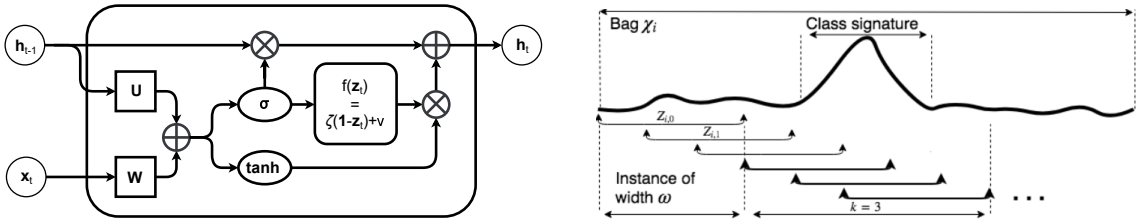
for cheap handcrafted feature engineering in the amplitude, time, and spectral domains [15, 34]. However, these solutions incur significant featurization overhead; this is exemplified in Table 2 on 1-second snippets extracted from the complex time-series. Even ignoring the SVM computation latency, it can be seen that the main computation bottleneck is this incremental overhead which results in >30% duty cycle on the Cortex-M3, of which ~10% constitutes the FFT overhead alone.

3.2.2 Shallow Classifier Architecture. As shown in Figure 3, a prototypical shallow radar classifier system consists of three subsystems: (i) a *displacement detector* for discriminating clutter vs. sources, (ii) an *incremental featurizer*, (iii) an *end inference engine* that discriminates source types, and (iv) a *composition manager* that handles their interactions. The displacement detector thresholds unwrapped phase over incoming windows of radar data ($\frac{1}{2}$s or 1s) to detect legitimate source displacements in the scene, filtering in-situ clutter that tends to yield self-canceling phase unwraps. When a source displacement is speculatively detected, the featurizer is invoked till the current displacement ends or a pre-specified time limit is reached. The final feature vector is fed to an end classifier such as SVM [34]. Note that incremental feature computation overhead *is* the primary impediment in realizing efficiency in these systems, hence techniques like replacing the heavy SVM classifier with the much lighter Bonsai [25], or observing longer displacements to run inference infrequently do not alleviate this problem.

Table 2. Computation overheads in a shallow (SVM) radar solution on Cortex-M3 (10 features, 1s windows)

Component	Latency (ms)
FFT	80
Incremental feature computation	212
SVM inference (700 SVs)	55

3.2.3 Deep Classifier Architecture. In the interest of designing resource efficient solutions, in this work, we use purely time-domain learning instead of all-domain feature engineering. While we preserve the aforementioned



(a) FastGRNN (cascaded) (b) EMI-FastGRNN (always on)

Fig. 4. FastGRNN & EMI-FastGRNN (images from [10, 28])

classifier hierarchy in our solution, we replace the simple “ensemble" with a principled 2-tier RNN approach. In the next sections, we present our proposed architecture and discuss how our approaches to deep feature learning can be used to successfully resolve the above issues.

4 2-TIER DEEP CLASSIFIER ARCHITECTURE

MSC-RNN is a multi-scale, cascaded architecture that uses EMI-FastGRNN as the lower-tier clutter discriminator and FastGRNN as the upper-tier source classifier. While EMI-FastGRNN efficiently localizes the source signature in a clutter prone time-series ensuring smaller sequential inputs along with early classification, FastGRNN reduces the per-step computational overhead over much heavier alternatives such as LSTM. We begin with the relevant background for each of these components.

4.1 Candidate Classifiers

FastGRNN. FastRNN [28] provably stabilizes RNN training by helping to avoid EVGP by using only two additional scalars over the traditional RNN. FastGRNN is built over FastRNN and it extends the scalars of FastRNNs to vector gates while maximizing the computation reuse. FastGRNN also ensures its parameter matrices are low-rank, sparse and byte quantized to ensure very small models and very fast computation. FastGRNN is shown to match the accuracies of state-of-the-art RNNs (LSTM and GRU) across various tasks like keyword spotting, activity recognition, sentiment analysis, and language modeling while being up to 45x faster.

Let $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T]$ be the input time-series, where $x_t \in \mathbb{R}^D$. The traditional RNN's hidden vector $\mathbf{h}_t \in \mathbb{R}^{\hat{D}}$ captures long-term dependencies of the input sequence:

$$\mathbf{h}_t = \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}) \tag{1}$$

Typically, learning $\mathbf{U}$ and $\mathbf{W}$ difficult due to the gradient instability. FastGRNN (Figure 4(a)) uses a scalar controlled peephole connection for every coordinate of $\mathbf{h}_t$:

$$\mathbf{h}_t = (\zeta(1 - \mathbf{z}_t) + \nu) \odot \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}_h) + \mathbf{z}_t \odot \mathbf{h}_{t-1} \tag{2}$$

$$\mathbf{z}_t = \sigma(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}_z) \tag{3}$$

Here, $0 \leq \zeta, \nu \leq 1$ are trainable parameters, and $\odot$ represents the vector Hadamard product.

EMI-RNN. Time-series signals when annotated are rarely precise and often coarsely labeled due to various factors like human errors and smaller time frames of activities themselves. EMI-RNN [10] tackles the problem of signal localization using MIL, by splitting the i^{th} data window into instances $\{Z_{i,\tau}\}_{\tau=1,\ldots T-\omega+1}$ of a fixed width ω (Figure 4(b)). The algorithm alternates between training the classifier and re-labeling the data based on the learned classifier until convergence. A simple thresholding scheme is applied to refine the instances: in each

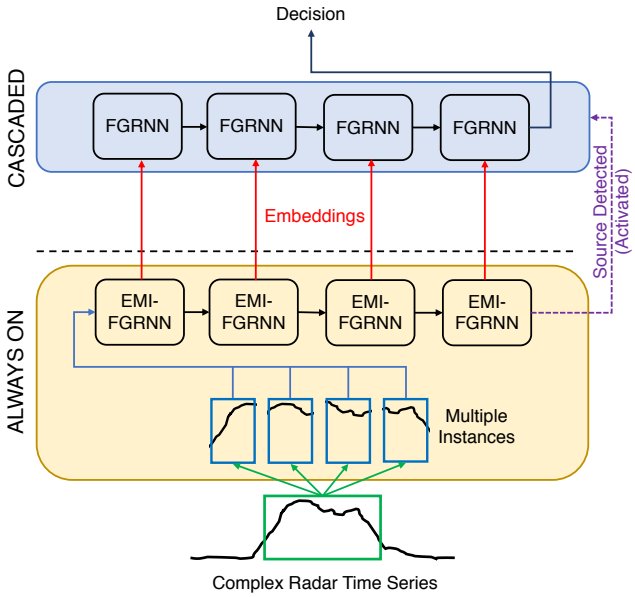


Fig. 5. MSC-RNN architecture – the lower EMI-FastGRNN runs continuously, while the higher FastGRNN is invoked only for legitimate displacements

iteration, k consecutive instances are found with maximum prediction sum for the class label. Only these instances are included in the training set for the next iteration. Here, k is a hyperparameter that intuitively represents the number of instances expected to cover the source signature. In the end, EMI-RNN produces precise signal signatures which are much smaller than the raw input, thus reducing the computation and memory overhead over the traditional sequential techniques. EMI-RNN also ensures early detection of noise or keywords thereby removing the need of going through the entire signal before making a decision. When combined, EMI-FastGRNN provides very small models along with very fast inference for time-series classification tasks. Codes for FastGRNN [28] & EMI-RNN [10] are part of Microsoft Research India's EdgeML repository [11].

4.2 MSC-RNN Design

While EMI-RNN is by itself equipped to handle multi-class classification efficiently, we find its accuracy and non-dominant source recall to be sub-optimal for the radar time-series, especially at smaller hidden dimensions and shorter window lengths. FastGRNN, on the other hand, is a relatively heavier solution to be used as a continuously running 3-class discriminator. To redress this trade-off, we make the following observations:

(i) clutter, which yields self-canceling phase, can be rejected at a relatively shorter time-scale,
(ii) disambiguating source types from their complex returns is a harder problem requiring a potentially longer window of observation, and
(iii) the common case in a realistic deployment constitutes clutter; legitimate displacements are relatively few.

MSC-RNN, therefore, handles the two sub-problems at different time-scales of featurization (see Figure 5): the lower tier, an EMI-FastGRNN, discriminates sources from clutter at the level of short instances, while the upper one, a windowed FastGRNN, discriminates source types at the level of longer windows. Further, the upper tier is invoked *only* when a source is discriminated by the lower tier and operates on the instance-level embeddings generated by the latter.

4.2.1 Joint Training and Inference. The training of the lower tier inherits from that of EMI-training. We recap its training algorithm [10], which occurs in two phases, the MI phase and the EMI phase. In the MI phase, where the source boundaries are refined in a clutter-prone window, the following objective function is optimized:

$$\min_{f_l, s_i} \frac{1}{n} \sum_{i,\tau} \mathbb{1}_{\tau \in [s_i, s_i+k]} \ell(f_l(Z_{i,\tau}), y_i) \tag{4}$$

Here, ℓ represents the loss function of FastGRNN, and the classifier f_l is based on the final time-step in an instance. In the EMI phase, which incorporates the early stopping, the loss $\mathcal{L}_{\text{EMI}}$ is obtained by replacing the previous loss function with the sum of the classifier loss at every step: $\min \sum_i \sum_{t=1}^{T} \ell(w^T o_{i,t})$, where w is the fully connected layer and $o_{i,t}$ the output at step t. The overall training proceeds in several rounds, where the switch to the EMI loss function is typically made halfway in.

For training the upper tier, in keeping with the divide-and-conquer paradigm of MSC-RNN, the Tier 2 FastGRNN cell should only learn to separate the source types, while ignoring instances of training data that are clutter. Therefore, we devise a conditional training strategy that captures the cascading behavior. To achieve this, the standard cross-entropy loss function of the upper tier is modified as:

$$\min_{f_u} \frac{1}{n} \sum_{i} \mathbb{1}_{y_i \neq -1} \ell(f_u(\mathcal{E}(\{Z_{i,\tau}^{tr}\})), y_i) \tag{5}$$

where f_u represents the upper classifier, and $\mathcal{E} : \mathbb{R}^{(T-\omega+1)\times\omega\times F} \rightarrow \mathbb{R}^{(T-\omega+1)\times H_l}$ represents the instance-level embedding vector from EMI-RNN with a hidden dimension of H_l (here, F represents the feature dimension for the radar time-series). Intuitively, this means that the upper loss is unaffected by clutter points, and thus the tiers can be kept separate.

The training algorithm for MSC-RNN is outlined in Algorithm 1. The two tiers are first separately initialized using their respective loss functions, and in the final phase, both are jointly trained to minimize the sum of their losses. Inference is simple: the instance-level EMI-RNN stops early with a decision of "Source" when a probability threshold $\hat{p}$ is crossed; $\geq k$ consecutive positives constitute a discrimination for which the cascade is activated.

5 COMPARATIVE & TIER-WISE EVALUATION

5.1 Datasets

Table 3(a) lists the radar source and clutter datasets collected in various indoor and outdoor environments, which are used in this work. Some of these locations are documented in Figure 6; small or crammed indoor spaces such as office cubicles have been avoided to prevent the radar returns from being adversely affected by multi-path effects and because they are not central to the smart city scenarios. A partial distribution of displacement durations is provided in Figure 7(a). Each data collect has associated with it the corresponding ground truth, recorded with motion-activated trail cameras or cellphone video cameras, with which the radar data was correlated offline to "cut" and label the source displacement snippets appropriately[1]. The datasets have been balanced in the number of human and non-human displacement points where possible, and windowed into snippets of 1, 1.5, and 2 seconds which correspond to 256, 384, and 512 I-Q sample pairs respectively. We note that due to the duration of collections and differences in average displacement lengths, etc., humans are underrepresented in these datasets compared to the other labels. Table 3(b) shows the number of training, validation, and test points for each of these window lengths on a roughly 3:1:1 split. Currently, only the cattle set has multiple concurrent targets; efforts to expand our datasets with target as well as radar type variations are ongoing.

[1]The radar dataset, which we have open-sourced, does not include individually identifiable information of living individuals and is thus not considered research with human subjects per 45 CFR §46.102(e)(1)(ii).

Algorithm 1: MSC-RNN training algorithm

Input: Multi-instance training data $\{\{Z^{tr}_{i,\tau}\}_{\tau=1,\ldots T-\omega+1}, y^{tr}_i\}_{i=1,\ldots n}$, the number of rounds n_r, the number of epochs n_e per round, k

Training:

Freeze FastGRNN, unfreeze EMI-FastGRNN
repeat
  Train **EMI-FastGRNN**($\{\{Z^{tr}_{i,\tau}\}, \mathbb{1}_{y^{tr}_i \neq -1}\}$) for $n_r \times n_e$ epochs
until convergence
Freeze EMI-FastGRNN, unfreeze FastGRNN
repeat
  Train **FastGRNN**($\{\mathcal{E}(\{Z^{tr}_{i,\tau}\}, y^{tr}_i)\}$) for $n_r \times n_e$ epochs, minimizing loss $\frac{1}{n}\sum_i \mathbb{1}_{y_i \neq -1}\ell(f_u(\mathcal{E}(\{Z^{tr}_{i,\tau}\})), y_i)$
until convergence
Unfreeze both EMI-FastGRNN and FastGRNN
for $r \in n_r$ **do**
  if $r < \frac{n_r}{2}$ **then**
    $\mathcal{L}_{\text{lower}} \leftarrow$ MI-loss
  else
    $\mathcal{L}_{\text{lower}} \leftarrow$ EMI-loss
  end if
  repeat
    Train **MSC-RNN**($\{\{Z^{tr}_{i,\tau}\}, \mathbb{1}_{y^{tr}_i \neq -1}\}$) for n_e epochs minimizing loss $\mathcal{L}_{\text{lower}} + \frac{1}{n}\sum_i \mathbb{1}_{y_i \neq -1}\ell(f_u(\mathcal{E}(\{Z^{tr}_{i,\tau}\})), y_i)$
  until convergence
end for

5.2 Evaluation Methodology

Our proposed architecture is compared with existing shallow radar solutions that use feature handcrafting in the amplitude, phase and spectral domains, as well as with other MIL RNNs. In all cases involving RNNs, the radar data is represented purely in the time-domain. The models chosen for this evaluation are:

(a) **2-tier SVM with phase unwrapped displacement detection.** Phase unwrapping [13] is a widely used technique in radar displacement detection due to its computational efficiency. The idea is to construct the relative trajectory of a source by accumulating differences in successive phase measurements, whereby clutter can be filtered out. We contrast MSC-RNN with a two-tier solution proposed in [34], which uses a robust variant of phase unwrapping with adaptive filtering of clutter samples.
(b) **3-class SVM.** A clutter vs human vs non-human SVM solution that uses feature handcrafting.
(c) **EMI-FastGRNN.** An EMI version of FastGRNN (Section 4).
(d) **EMI-LSTM.** An EMI version of the LSTM. Note that this is a much heavier architecture than the former, and should not be regarded as suitable for a microcontroller device.

Since shallow featurization incurs high incremental overhead, real-time micro-power radar solutions typically only compute the highest information features. For the SVM solutions, training is performed with the best 15 features selected by the *minimal-redundancy-maximal-relevance* (mRMR) criterion [33].

For the MIL experiments, the windowed data from Table 3(b) is further reshaped into instances of length 48×2 samples with a fixed stride of 16×2, where 2 refers to the number of features (I and Q components of radar data).

(a) Public park (b) Indoor amphitheater

(c) Parking garage bldg. (d) Building foyer

Fig. 6. Some locations where source and clutter data was collected for experiments

For example, for 1-second windows, the shape of the training data for MIL experiments is (17055, 14, 48, 2), and the shape of the corresponding instance-level one-hot labels is (17055, 14, 3). In the interest of fairness and also to avoid a combinatorial exploration of architectural parameters, we present results at fixed hidden sizes of 16, 32, and 64. For MSC-RNN, the lower tier's output (embedding) dimension and upper tier's hidden dimension are kept equal; however, in practice, it is easy to parameterize them differently since the former only affects the latter's input dimension.

5.2.1 Hyperparameters. Table 4 lists the hyperparameter combinations used in our experiments. For the upper-tier source discrimination comparison in Section 5.3.3, FastGRNN is also allowed to select its optimum input length from 16, 32, and 64 samples.

The selection of the EMI hyperparameter k merits some discussion, in that it controls the extent of "strictness" we assign to the definition of *displacement*. A higher k makes it more difficult for a current window to be classified as a source unless the feature of interest is genuinely compelling. Expectedly, this gives a trade-off between clutter and source recall as is illustrated in Figure 7(b). As explained in Section 1, controlling for false positives is extremely important in radar sensing contexts such as intrusion detection. Hence, we empirically set k to 10, the smallest value that gives a clutter recall of 0.999 or higher in our windowed datasets.

5.3 Results

5.3.1 Comparative Classifier Performance. We compare the inference accuracy and recalls of MSC-RNN, with the RNN and shallow solutions outlined in Section 5.2.

Recall that we have purposefully devised a purely time-domain solution for source discrimination for efficiency reasons, since one of the main components of featurization overhead is that of FFT computations. Figure 8 compares MSC-RNN with engineered features in the amplitude, time, and spectral domains that are optimized

Table 3. Radar evaluation datasets

(a) Source displacement counts and clutter durations

Environment	Data Type	
	Type	Count
Building foyer	Human, Gym ball	52, 51
Indoor amphitheater	Human, Gym ball	49, 41
Parking garage bldg.	Human	268
Parking lot	Human, Car	50, 41
Indoor soccer field	Human, Gym ball	90, 82
Large classroom	Human, Gym ball	48, 50
Cornfield	Human, Dog	117, 85
Cattle shed	Cow	319
Playground	Clutter	45 mins
Parking garage bldg.	Clutter	45 mins
Public park	Clutter	45 mins
Garden	Clutter	45 mins
Lawn	Clutter	20 mins

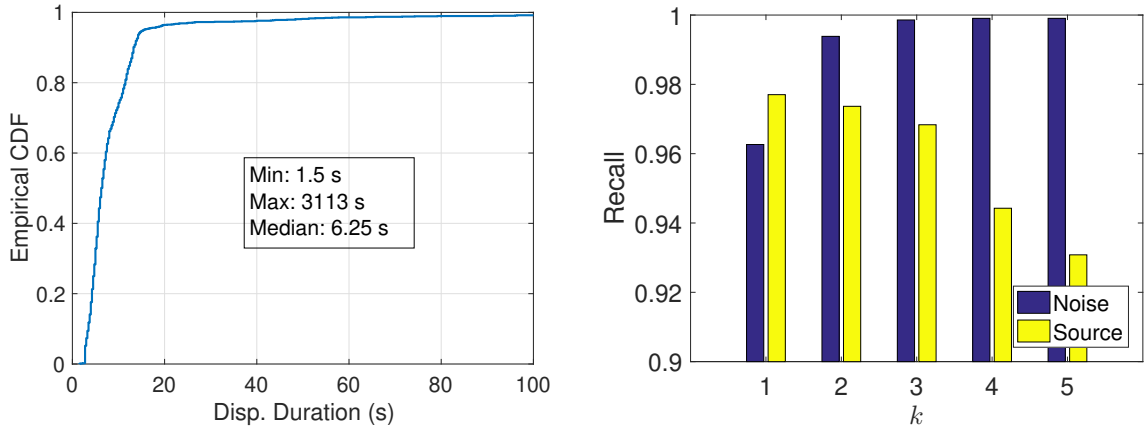
(b) Windowed data from (a) showing number of training, validation, and test points

Window Length (s)	#Windows		
	Training	Validation	Testing
1	17055	5685	5685
1.5	11217	3739	3739
2	8318	2773	2773

Table 4. Training hyperparameters used

Model	Hyperparameter	Values
EMI/FastGRNN	Batch size	64, 128
	Hidden size	16, 32, 64
	Gate non-linearity	sigmoid, tanh
	Update non-linearity	sigmoid, tanh
	k	10
	Keep prob. (EMI-LSTM)	0.5, 0.75, 1.0
	Optimizer	Adam
SVM	c	1e-3, 1e-2, 0.1, 1, 10, 100, 1e3, 1e5, 1e6
	γ	1e-3, 1e-2, 5e-2, 0.1, 0.5, 1, 5, 10

for micro-power radar classification. For the two-tier SVM, the source recalls for increasing window sizes are inferred from Figure 9 (discussed in Section 5.3.3). We find that MSC-RNN significantly outperforms the 2-tier



(a) Disp. Duration CDF (partial)

(b) Impact of k on EMI Recalls

Fig. 7. Source detected duration CDF for the data in Table 3(a) and how the hyperparameter k in 2-class EMI affects their detection (1-second windows)

SVM solution in terms of human and non-human recalls, even with features learned from the raw time-series. Similarly, for the 3-class case, our solution provides much more stable noise robustness and is generally superior even to the much heavier SVM solution.

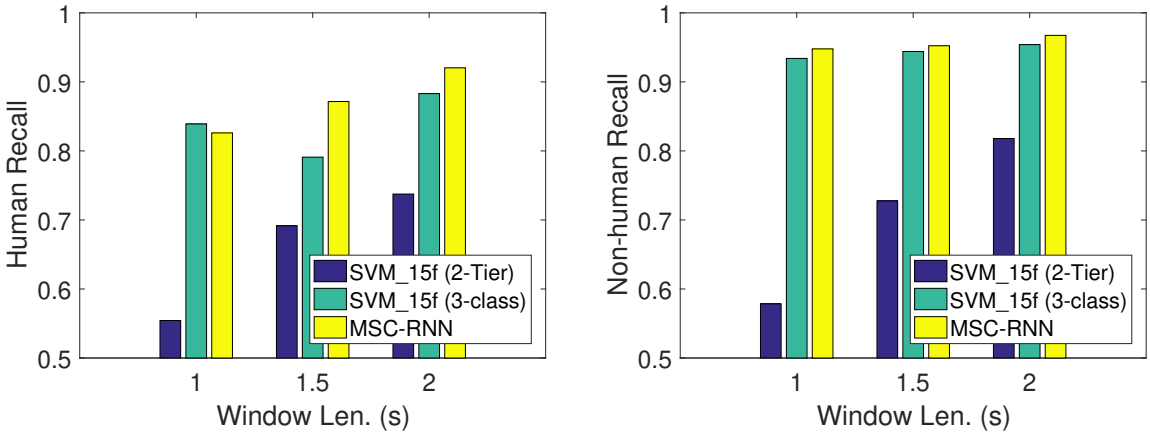
5.3.2 Runtime Efficiency Comparison - MSC-RNN vs. Feature Handcrafting. Table 5 lists the runtime duty cycle estimates of MSC-RNN versus shallow SVM alternatives in two deployment contexts with realistic clutter conditions, supported by usage statistics of a popular biking trail in Columbus, OH [2]. While the 2-tier SVM understandably has the lowest duty cycle due to a cheap lower tier, it is not a competitive solution as established in Section 5.3.1. The 3-class SVM, on the other hand, is dominated by the feature computation overhead. While the 48×2 MSC-RNN formulation is about 1.75× as efficient as using handcrafted features, it is possible to reduce instance-level computations even further by using longer input vectors and reducing the number of iterations. As an example, an alternate formulation of MSC-RNN with a 16-dimensional input vector at the instance level is 3× more efficient than feature engineering.

Table 5. Estimated featurization duty cycle comparison on ARM Cortex-M3

Architecture	**Est. Duty Cycle (Cortex-M3)**	
	97% Clutter	**98% Clutter**
MSC-RNN (Inp. dim.=2)	21.00%	20.00%
MSC-RNN (Inp. dim.=16)	10.87%	10.70%
2-Tier SVM	2.05%	1.70%
3-Class SVM	35.00%	35.00%

5.3.3 Tier-wise Evaluation. We next compare the lower-tier and upper-tier classifiers individually to their shallow counterparts in the 2-tier SVM solution.

Tier 1 Classifier. Figure 9 compares the probabilities of missed detects versus displacement durations for the 3-outof-4 displacement detector and the EMI component of our solution (for a principled approach to choosing



(a) Human Recall

(b) Non-human Recall

Window Length (s)	Accuracy		Clutter Recall	
	SVM_15f (3-class)	MSC-RNN	SVM_15f (3-class)	MSC-RNN
1	0.851	0.944	0.758	0.999
1.5	0.934	0.954	0.996	0.999
2	0.959	0.972	0.999	1.000

(c) Accuracy and Clutter Recall (3-class SVM and MSC-RNN)

Fig. 8. Classification comparison of purely time-domain FastGRNN with two SVM solutions: (a) a 2-tier system using a phase unwrapped clutter rejector as the lower tier, and (b) a 3-class SVM. Both use 15 high information features handcrafted in the amplitude, time, and spectral domains

parameters for the former, refer to Appendix A) at hidden sizes of 16, 32, and 64. It can be seen that, for the shortest cut length of 1.5 s in the dataset, the detection probability is improved by up to 1.5× (1.6×) over the 3-outof-4 detector with false alarm rates of 1/week and 1/month respectively even when the false alarm rate (1−test clutter recall) of EMI is 0, which translates to a false alarm rate of <1 per year. Further, the EMI detector converges to 0 false detects with displacements ≥2.5 s, and is therefore able to reliably detect walks 2.6× shorter than the previous solution. Therefore, it is possible to restrict false positives much below 1/month while significantly improving detectability over the M-outof-N solution. Since the clutter and source datasets span various backgrounds (Figure 6), MSC-RNN offers superior cross-environmental robustness.

Tier 2 Classifier. We now show that the gains of MSC-RNN over the 2-tier SVM solution are not, in fact, contingent on the quality of the underlying displacement detector for the latter. For this experiment, we train a 2-class FastGRNN on embeddings derived from the Tier 1 EMI-FastGRNN. Table 6 compares its performance with the upper-tier SVM from the latter when trained with the best 15 cross-domain features obtained from the raw radar samples. It can be seen that the purely time-domain FastGRNN still generally outperforms the 2-class SVM on all three metrics of accuracy, human recall, and non-human recall. Thus, it is possible to replace feature engineering with deep feature learning and enjoy the dual benefits of improved sensing and runtime efficiency for this class of radar applications.

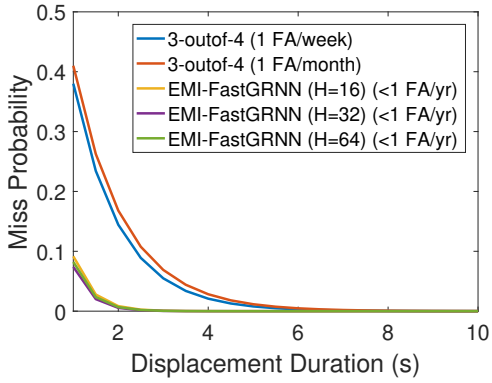


Fig. 9. Comparison of miss probabilities versus displacement durations of Tier 1 classifier vs. 3-outof-4 phase unwrapped displacement detector

Table 6. Independent of the Tier 1 classifier, the Tier 2 source-type classifier outperforms the SVM

Window Length (s)	Accuracy		Human Recall		Non-human Recall	
	SVM _15f	Fast-GRNN	SVM _15f	Fast-GRNN	SVM _15f	Fast-GRNN
1	0.93	0.93	0.90	0.90	0.93	0.94
1.5	0.93	0.93	0.90	0.93	0.95	0.95
2	0.93	0.96	0.86	0.96	0.96	0.97

5.3.4 Evaluation of MSC-RNN Joint Training Strategy. MSC-RNN can be trained in a couple of ways to arrive at the final model. Algorithm 1 presents the best strategy among the following:

(1) The two tiers are jointly trained with a combined loss function in the last round,
(2) The two tiers are independently trained prior to training the joint optimization objective (*warm start*) as opposed to training purely with the combined loss (*cold start*).

We next evaluate the impact of each of these strategies. The analysis is done in two different epoch settings, with the number of rounds n_r and the number of epochs per round n_e set to 50 (100) and 50 (50) respectively, and the best results are reported.

Joint versus Independent Training. We compare Algorithm 1 with an alternate training strategy where the EMI layer is first trained with loss $\mathcal{L}_{\text{lower}}$ freezing the upper FastGRNN, and then the upper tier is trained with loss $\frac{1}{n}\sum_i \mathbb{1}_{y_i \neq -1}\ell(f_u(\mathcal{E}(\{Z_{i,\tau}^{tr}\})), y_i)$ freezing the lower tier. This is very similar to steps 1-8 in Algorithm 1, with one difference. Since MSC training *encompasses* independent training with early exit through convergence as its pre-processing step, we instead let the purely independent training run for the full $n_r \times n_e$ epochs without early exit for fairness' sake. At the end of training, the checkpointed model with the best validation accuracy is used to generate the test results.

The results are presented in Table 7. While it is clear that joint training per Algorithm 1 generally maintains or improves upon all metrics, especially human recall, two salient observations can be made:

(1) For relatively simpler models and when the input sequence is short, the joint optimization strategy is sometimes an overkill. This is illustrated through an inversion in the human recall improvement at H=16

Table 7. Joint training the two tiers of MSC-RNN generally improves/maintains all metrics over training them independently

Window Length (s)	Accuracy		Clutter Recall		Human Recall		Non-human Recall	
	MSC (Joint Tr.)	MSC (No Joint Tr.)	MSC (Joint Tr.)	MSC (No Joint Tr.)	MSC (Joint Tr.)	MSC (No Joint Tr.)	MSC (Joint Tr.)	MSC (No Joint Tr.)
H=16								
1	0.928	0.9278	0.999	0.999	0.750	0.778	0.945	0.933
1.5	0.946	0.910	0.999	0.999	0.845	0.693	0.944	0.930
2	0.955	0.915	1	1	0.853	0.681	0.955	0.931
H=32								
1	0.943	0.932	0.999	0.999	0.823	0.761	0.947	0.951
1.5	0.954	0.929	0.999	0.999	0.872	0.775	0.952	0.936
2	0.972	0.921	1	1	0.920	0.713	0.967	0.934
H=64								
1	0.944	0.936	0.999	0.999	0.826	0.829	0.948	0.930
1.5	0.954	0.932	0.999	0.999	0.872	0.809	0.952	0.929
2	0.965	0.925	1	1	0.904	0.742	0.960	0.930

on 1 second windows. However, this simple setting typically yields overall suboptimal results. And when the machine is more complex and has longer sequences to memorize, independent training fails to find good optima; consequently, training performance degrades with increasing window lengths. Joint training, however, improves the optimization significantly and its performance increases monotonically with window length at each hidden dimension.

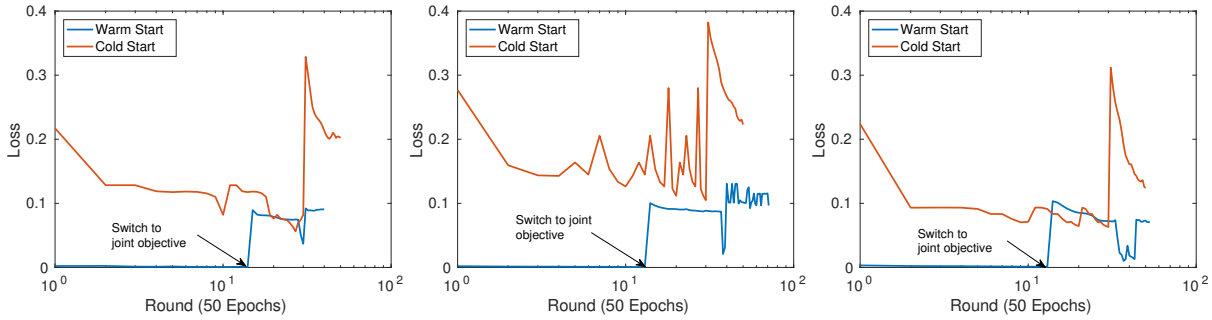
(2) For longer window lengths, the number of training data points is also reduced significantly (Table 3(b)). From Table 7, it is apparent that the lack of data also affects training performance of the complex, two-tier MSC-RNN model in the absence of a joint optimization strategy. Though it is not easy to decouple the effect of having to memorize longer sequences from that of reduced data, joint training does seem to offer significantly better resilience to the latter.

Impact of Pre-Training MSC-RNN Tiers. We compare Algorithm 1 with a variant where the combined loss objective of $\mathcal{L}_{\text{lower}} + \frac{1}{n}\sum_i \mathbb{1}_{y_i \neq -1}\ell(f_u(\mathcal{E}(\{Z^{tr}_{i,\tau}\})), y_i)$ is optimized from the very first epoch, and tabulate the results in Table 8. As before, the cold start approach is allowed to train for the full $n_r \times n_e$ epochs without early exit for reasons of fairness.

It can be seen that the cold start approach is significantly worse at all window lengths and all hidden sizes, most prominently in terms of human recall performance. This can be explained by the fact that MSC-RNN has essentially two objective functions, one at each tier, and trivially combining them can be unstable since the two objectives have to be optimized simultaneously. Instead, the warm start technique used by Algorithm 1 pre-trains the two tiers *alternately* at the start to stabilize the space before joint optimization (steps 1-8). Doing so reduces the learning complexity, and the learnt parameters of Tier 1 or Tier 2 (whichever is frozen) help drive the other stage towards a common optimum. In the end, joint optimization (steps 9-18) with both tiers unfrozen fine-tunes these representations and is seen to boost the performance significantly. This is corroborated through Figure 10 which shows the training loss landscapes of MSC-RNN joint training at H=64, with and without alternating

Table 8. Joint training of MSC-RNN with pre-training the two tiers (*warm start*) is better than training the joint optimization objective from the very first epoch (*cold start*)

Window Length (s)	Accuracy		Clutter Recall		Human Recall		Nonhuman Recall	
	Joint Tr. (Warm Start)	Joint Tr. (Cold Start)	Joint Tr. (Warm Start)	Joint Tr. (Cold Start)	Joint Tr. (Warm Start)	Joint Tr. (Cold Start)	Joint Tr. (Warm Start)	Joint Tr. (Cold Start)
H=16								
1	0.928	0.910	0.999	0.999	0.750	0.710	0.945	0.922
1.5	0.946	0.910	0.999	0.999	0.845	0.771	0.944	0.896
2	0.955	0.935	1	1	0.853	0.795	0.955	0.933
H=32								
1	0.943	0.919	0.999	0.999	0.823	0.772	0.947	0.914
1.5	0.954	0.926	0.999	0.999	0.872	0.749	0.952	0.942
2	0.972	0.925	1	1	0.920	0.732	0.967	0.936
H=64								
1	0.944	0.930	0.999	0.999	0.826	0.749	0.948	0.951
1.5	0.954	0.929	0.999	0.999	0.872	0.786	0.952	0.932
2	0.965	0.948	1	1	0.904	0.866	0.960	0.934



(a) Window Length=1s (b) Window length=1.5s (c) Window length=2s

Fig. 10. Training loss landscapes of MSC-RNN joint optimization with (*warm start*) and without (*cold start*) pre-training of the two tiers (H=64)

pre-training per round (n_e=50). For the warm start variant, the point where the training switches to the combined loss objective is indicated, while the cold start algorithm uses the same loss function throughout. It is clear that, for all three window lengths, the warm start approach yields more stable training and finds better minima in the joint optimization phase.

5.3.5 MSC-RNN Architectural Justification. Figure 11(a)-(c) contrasts our model with 3-class EMI-FastGRNN and EMI-LSTM, for fixed hidden sizes of 16, 32, and 64 respectively. It can be seen that MSC-RNN outperforms the monolithic EMI algorithms on all three metrics of accuracy, non-human and human recalls (with one exception

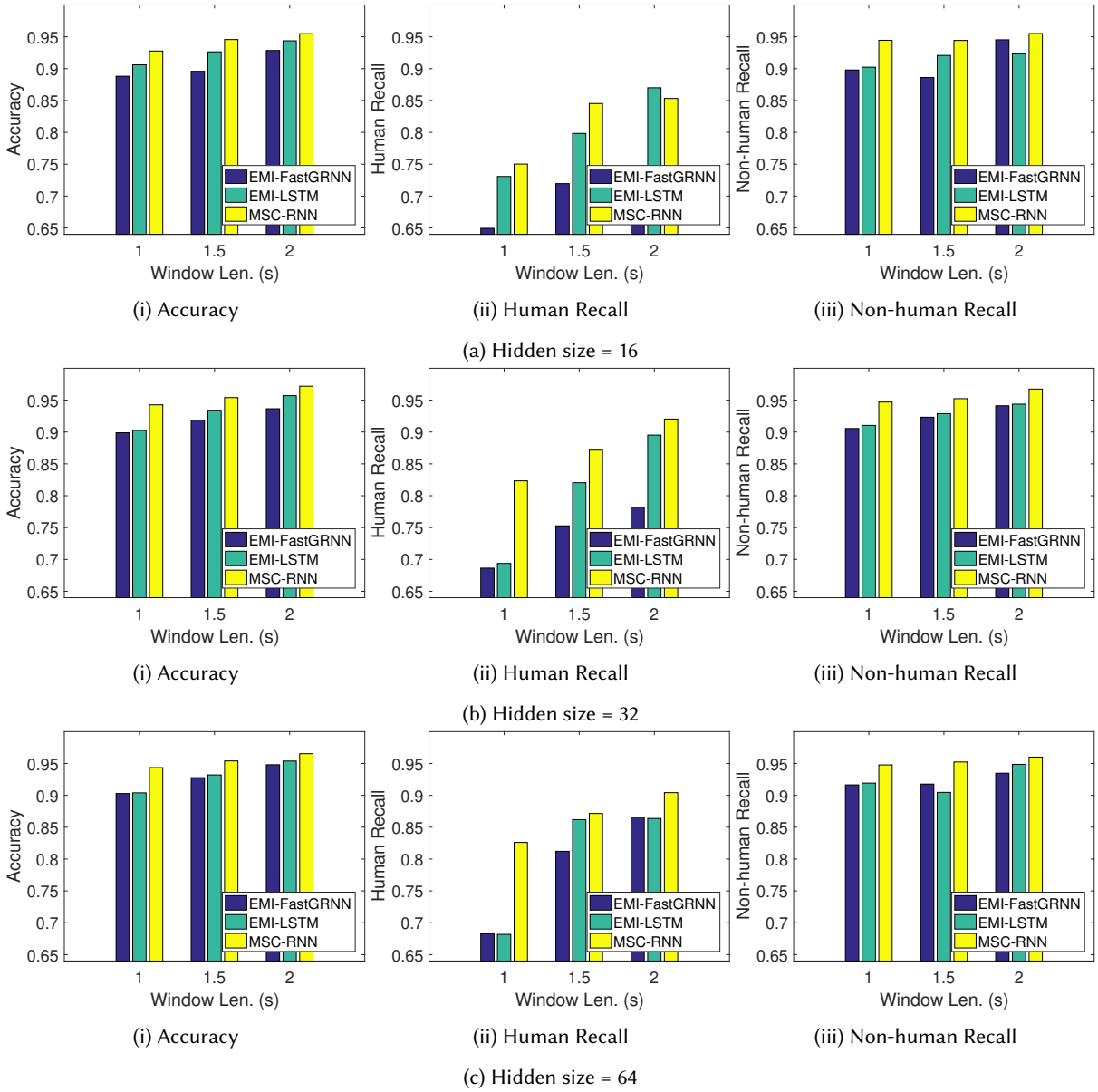


Fig. 11. Sensing performance comparison of MSC-RNN with EMI-FastGRNN and EMI-LSTM

for EMI-LSTM). Notably, cascading significantly enhances the non-dominant class recall over the other methods, especially for larger hidden sizes, and therefore offers better resilience to the source type imbalance in radar datasets.

6 DISCUSSION

6.1 Can Deep Feature Learning Replace Feature Engineering for Radar Classification?

In this section, we present a deeper defense of our claim that RNN feature learning purely in the time domain can essentially replace feature handcrafting in the amplitude, time, and frequency domains. We establish this through an experimental set up to learn the top 16 handcrafted features selected by the mRMR algorithm [33] from the raw radar time series. We find that a single FastGRNN cell is able to approximate the feature vectors with reasonably high accuracy. In these (regression) experiments, we optimize the Mean Squared Error (MSE) loss, given by:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}i)^2$$

Since no other FC layers are introduced post featurization, the RNN's hidden size is fixed to the feature dimension (16). In addition to the training and validation MSE, we also report the Mean Absolute Error (MAE), given by:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}i|$$

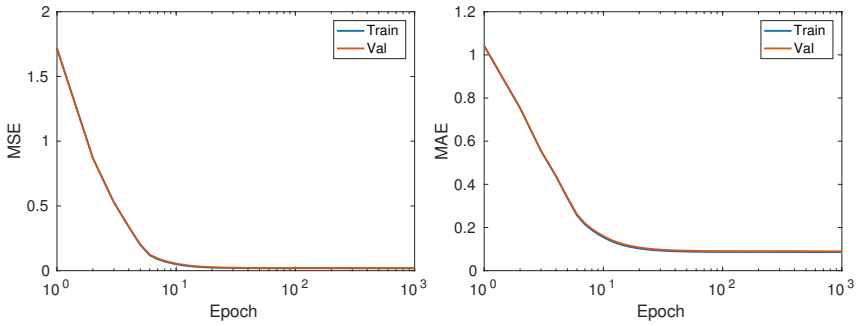
Figure 12(a)-(c) illustrates the training progression for 1000 epochs at window lengths of 1s, 1.5s, and 2s respectively, while the corresponding test performances are presented in Table 9. Given that the norm of the engineered features is approximately 2.5, it is clear that FastGRNN is able to learn the top handcrafted features used in a mote-scale shallow radar solution in a stable manner (the slight degradation with longer input sequences can be attributed to the hidden size being constrained). Interestingly, most of the relevant handcrafted features are in the frequency domain (cf. Table 11 in Appendix B), which the RNN can learn purely from raw data. This can be explained by the fact that STFT, and features derived from it, are essentially linear transformations on the time domain signal and can therefore be faithfully approximated by a non-linear featurizer.

Table 9. Test loss (MSE) and MAE of FastGRNN that approximate the top 16 cross-domain handcrafted features (ℓ_2 norm ≈2.5) for radar classification using time domain input

Window Length (s)					
1		**1.5**		**2**	
MSE	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
0.020	0.090	0.026	0.115	0.031	0.127

6.2 Improving Non-Dominant Recalls Further - Post-hoc Strengthening of Tier 2 Classifier

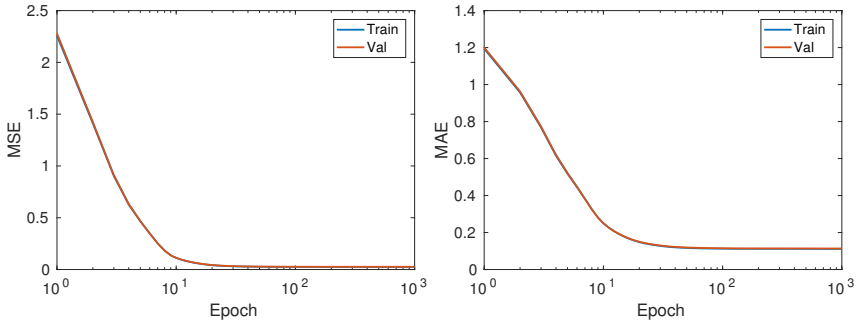
The MSC-RNN non-dominant (human) class recalls can, in fact, be improved further while maintaining accuracies and dominant class recalls and without increasing the storage overhead. We achieve this by re-training the upper tier post-hoc with a stronger end classifier (this is not done for the simpler clutter rejection common case, since it is a slightly heavier step than computing softmaxes on EMI embeddings). Specifically, we use *Bonsai* [25] to replace the last layer of the Tier 2 FastGRNN. Bonsai learns a single, short, sparse tree classifier with powerful non-linear predictors at internal as well as leaf nodes, designed for use on resource constrained devices. It first projects the feature vectors into a low-dimensional space, and the parameters of this projection matrix are learnt jointly with all tree parameters. Equation 6 describes the Bonsai inference computation for a feature vector $\mathbf{x}$ that is projected to a low dimension P using a sparse matrix $\mathbf{Z}$.



(i) MSE (loss)

(ii) MAE

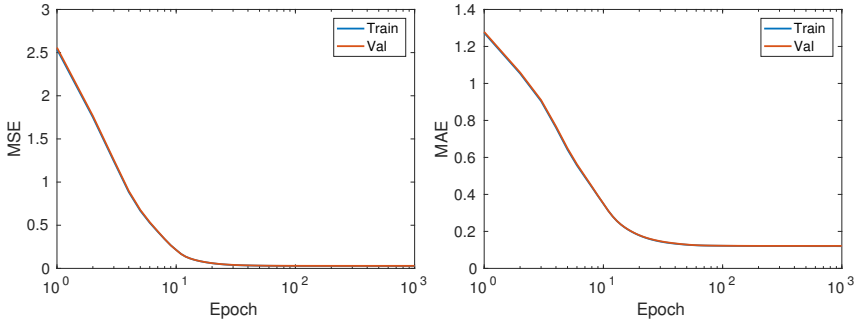
(a) Window Length=1s



(i) MSE (loss)

(ii) MAE

(b) Window Length=1.5s



(i) MSE (loss)

(ii) MAE

(c) Window Length=2s

Fig. 12. Evolution of training and validation loss (MSE) and MAE with a FastGRNN setup to approximate the top 16 cross-domain handcrafted features for radar classification, purely from time domain input. *N.B.:* The training and validation MSEs and MAEs almost overlap in each training, indicating no over-/underfitting.

$$y(\mathbf{x}) = \sum_k \mathbf{I}_k(\mathbf{x})\mathbf{W}_k^{\mathrm{T}}\mathbf{Z}\mathbf{x} \odot \tanh(\sigma\mathbf{V}_k^{\mathrm{T}}\mathbf{Z}\mathbf{x}) \tag{6}$$

Here, $\mathbf{x}$ is the upper-tier embedding vector, k is an index over the number of nodes in the Bonsai tree (7 for a 2-layer tree), $\mathbf{I}_k$ is an indicator function that is 1 if the node k is in the computation path, $\mathbf{W}_k$ and $\mathbf{V}_k$ are weight matrices associated with node k, and $\odot$ represents the Hadamard operation. For a projection dimension of P, the branching or $\mathbf{I}_k$ is determined by an additional P-dimensional vector $\boldsymbol{\theta}_k$ that is learned at each internal node, where the sign of $\boldsymbol{\theta}_k^{\mathrm{T}}\mathbf{Z}\mathbf{x}$ indicates whether the left or right child of the tree is taken. Note that each $\mathbf{W}_k$ and $\mathbf{V}_k$ also reduces to P-dimensional vectors for the binary classification problem.

Table 10. MSC-RNN human recalls can be further improved by replacing the fully-connected end classifier with a short, heavy-noded *Bonsai* tree with no additional storage overhead. Results shown are for the upper tier with H=64

Window Length (s)	Accuracy		Human Recall		Non-human Recall	
	MSC Tier 2 + Logistic	MSC Tier 2 + Bonsai	MSC Tier 2 + Logistic	MSC Tier 2 + Bonsai	MSC Tier 2 + Logistic	MSC Tier 2 + Bonsai
1	0.911	0.910	0.851	0.886	0.938	0.920
1.5	0.920	0.920	0.834	0.844	0.958	0.952
2	0.923	0.938	0.845	0.910	0.953	0.951

In this experimental setup, we use Bonsai classifiers of depth 2 and 3, and report the best results at each window length (Table 10) for H=64. To keep the sizes under 64×2 for the last layer, we keep the weights $\mathbf{W}_k$, $\mathbf{V}_k$, $\boldsymbol{\theta}_k$, as well as the projection matrix $\mathbf{Z}$ 90% sparse, and use projection dimensions of 13 and 9 for the two tree depths respectively. It can be seen than the human class recalls can be improved by up to 7% by strengthening the end classifier post-hoc. Note that only the upper tier is jointly re-learned with Bonsai in these experiments. Jointly optimizing both tiers of MSC-RNN *along with* Bonsai is a much more complex objective that can potentially yield further benefits; this is left as future work.

6.3 Low-power Implementation Details

The radar sensor described in Figure 1(a) uses an ARM Cortex-M3 microcontroller with 96 KB of RAM and 4 MB of flash storage. It runs eMote [41], a low-jitter near real-time operating system with a small footprint. We emphasize that energy efficient compute, not working memory or storage, is the bigger concern for efficient real-time operation. Hence, we take several measures to efficiently implement the multi-scale RNN to run at a low duty cycle on the device. These include low-rank representations of hidden states, integer quantization, and *piecewise-linear* approximations of non-linear functions. The latter in particular ensures that all the computations can be performed with integer arithmetic when the weights and inputs are quantized. For example, $\tanh(x)$ can be approximated as:

$$\mathrm{quantTanh}(x) = \begin{cases} x & , \text{if } |x| < 1 \\ \mathrm{sgn}(x) & , \text{otherwise} \end{cases} \tag{7}$$

, and $\mathrm{sigmoid}(x)$ can be approximated as:

$$\mathrm{quantSigm}(x) = \max(\min(\frac{x+1}{2}, 1), 0) \tag{8}$$

Similar to [28], the weight matrices at both tiers of MSC-RNN are trained with low-rank representations for faster inference. The RNN matrices W and U are represented as follows:

$$W = W^1(W^2)^{\mathrm{T}} \tag{9}$$

$$U = U^1(U^2)^{\mathrm{T}}, \tag{10}$$

, where the low rank hyperparameters r_w and r_u dictate that $W^1 \in \mathbb{R}^{D \times r_w}$, $W^2 \in \mathbb{R}^{\hat{D} \times r_w}$, whereas $U^1, U^2 \in \mathbb{R}^{\hat{D} \times r_u}$. This step reduces the net complexity over T timesteps from $O(T\hat{D}(D + \hat{D}))$ to $O(T(r_w(D + \hat{D}) + r_u\hat{D}))$.

The underlying linear algebraic operations are implemented using the CMSIS-DSP library [1]. While advanced ARM processors such as Cortex-M4 do offer floating point support, it should be noted that, for efficiency reasons, using sparse, low rank matrices and quantization techniques are beneficial in general.

7 CONCLUSION AND FUTURE WORK

In this work, we introduce multi-scale, cascaded RNNs for radar sensing, and show how leveraging the ontological decomposition of a canonical classification problem into clutter vs. source classification, followed by source type discrimination on an on-demand basis can improve both sensing quality as well as runtime efficiency over alternative systems. Learning discriminators at the time-scales relevant to their respective tasks, and jointly training the discriminators while being cognizant of the cascading behavior between them yields the desired improvement.

The extension of MSC-RNNs to more complicated sensing contexts is a topic of future work. Of interest are regression-based radar "counting" problems such as occupancy estimation or active transportation monitoring, where the competitiveness of MSC-RNN to architectures such as TCNs [4] could be insightful. We also believe that MSC-RNN could also apply to alternative sensing for smart cities and built environments where the sources have intrinsic ontological hierarchies, such as in urban sound classification [5].

8 ACKNOWLEDGEMENTS

We are indebted to Don Dennis, Prateek Jain, and Harsha Vardhan Simhadri at Microsoft Research India for their suggestions and feedback. The computation for this work was supported by the Ohio Supercomputer Center [7] project PAS1090, the IIT Delhi HPC facility, and Azure services provided by Microsoft Research Summer Workshop 2018: Machine Learning on Constrained Devices [2]. We thank Jihoon Yun and Saswata Dasgupta for their help in photographing the data collection locations.

REFERENCES

[1] [n. d.]. CMSIS-DSP Software Library. http://www.keil.com/pack/doc/CMSIS/DSP/html/index.html.
[2] [n. d.]. Olentangy Trail usage, Columbus, OH. https://www.columbus.gov/recreationandparks/trails/Future-Trails-(Updated)/.
[3] Joshua Adkins, Branden Ghena, Neal Jackson, Pat Pannuto, Samuel Rohrer, Bradford Campbell, and Prabal Dutta. 2018. The signpost platform for city-scale sensing. In *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 188–199.
[4] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018).
[5] Juan P Bello, Claudio Silva, Oded Nov, R Luke Dubois, Anish Arora, Justin Salamon, Charles Mydlarz, and Harish Doraiswamy. 2019. SONYC: A system for monitoring, analyzing, and mitigating urban noise pollution. *Commun. ACM* 62, 2 (2019), 68–77.
[6] Charles E Catlett, Peter H Beckman, Rajesh Sankaran, and Kate Kusiak Galvin. 2017. Array of things: A scientific research instrument in the public way: platform design and early lessons learned. In *Proceedings of the 2nd International Workshop on Science of Smart City Operations and Platforms Engineering*. 26–33.
[7] Ohio Supercomputer Center. 1987. Ohio Supercomputer Center. http://osc.edu/ark:/19495/f5s1ph73
[8] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).

[2] https://www.microsoft.com/en-us/research/event/microsoft-research-summer-workshop-2018-machine-learning-on-constrained-devices/

[9] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2016. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704* (2016).

[10] Don Dennis, Chirag Pabbaraju, Harsha Vardhan Simhadri, and Prateek Jain. 2018. Multiple instance learning for efficient sequential data classification on resource-constrained devices. In *Advances in Neural Information Processing Systems*. 10953–10964.

[11] Don Kurian Dennis, Yash Gaurkar, Sridhar Gopinath, Chirag Gupta, Moksh Jain, Ashish Kumar, Aditya Kusupati, Chris Lovett, Shishir G Patil, and Harsha Vardhan Simhadri. [n. d.]. EdgeML: Machine Learning for resource-constrained edge devices. https://github.com/Microsoft/EdgeML

[12] Dustin P Fairchild and Ram M Narayanan. 2014. Classification of human motions using empirical mode decomposition of human micro-Doppler signatures. *IET Radar, Sonar & Navigation* 8, 5 (2014), 425–434.

[13] Richard M Goldstein, Howard A Zebker, and Charles L Werner. 1988. Satellite radar interferometry: Two-dimensional phase unwrapping. *Radio science* 23, 4 (1988), 713–720.

[14] Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *arXiv preprint arXiv:1510.00149* (2015).

[15] Jin He and Anish Arora. 2014. A regression-based radar-mote system for people counting. In *2014 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 95–102.

[16] Jin He, Dhrubojyoti Roy, Michael McGrath, and Anish Arora. 2014. Mote-scale human-animal classification via micropower radar. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. 328–329.

[17] Geoffrey E Hinton. 1990. Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence* 46, 1-2 (1990), 47–75.

[18] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[19] Rios Jesus Javier and Youngwook Kim. 2014. Application of linear predictive coding for human activity classification based on micro-Doppler signatures. *IEEE Geoscience and Remote Sensing Letters* 11, 10 (2014), 1831–1834.

[20] Branka Jokanovic, Moeness Amin, and Fauzia Ahmad. 2016. Radar fall motion detection using deep learning. In *2016 IEEE radar conference (RadarConf)*. IEEE, 1–6.

[21] Youngwook Kim and Hao Ling. 2008. Human activity classification based on micro-Doppler signatures using an artificial neural network. In *2008 IEEE antennas and propagation society international symposium*. IEEE, 1–4.

[22] Youngwook Kim and Brian Toomajian. 2016. Hand gesture recognition using micro-Doppler signatures with convolutional neural network. *IEEE Access* 4 (2016), 7125–7130.

[23] Dimitrios Kotzias, Misha Denil, Phil Blunsom, and Nando de Freitas. 2014. Deep multi-instance transfer learning. *arXiv preprint arXiv:1411.3128* (2014).

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[25] Ashish Kumar, Saurabh Goyal, and Manik Varma. 2017. Resource-efficient machine learning in 2 KB RAM for the Internet of Things. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1935–1944.

[26] Sangeeta Kumari, Dhrubojyoti Roy, Mark Cartwright, Juan Pablo Bello, and Anish Arora. 2019. EdgeL^3: Compressing L^3-Net for mote scale urban noise monitoring. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 877–884.

[27] Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. 2020. Soft Threshold Weight Reparameterization for Learnable Sparsity. In *Proceedings of the International Conference on Machine Learning*.

[28] Aditya Kusupati, Manish Singh, Kush Bhatia, Ashish Kumar, Prateek Jain, and Manik Varma. 2018. FastGRNN: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network. In *Advances in Neural Information Processing Systems*. 9017–9028.

[29] Liang Liu, Mihail Popescu, Marjorie Skubic, Marilyn Rantz, Tarik Yardibi, and Paul Cuddihy. 2011. Automatic fall detection based on Doppler radar motion signature. In *2011 5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*. IEEE, 222–225.

[30] Gihan J Mendis, Tharindu Randeny, Jin Wei, and Arjuna Madanayake. 2016. Deep learning based Doppler radar for micro UAS detection and classification. In *MILCOM 2016-2016 IEEE Military Communications Conference*. IEEE, 924–929.

[31] Michael C Mozer. 1992. Induction of multiscale temporal structure. In *Advances in neural information processing systems*. 275–282.

[32] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International conference on machine learning*. 1310–1318.

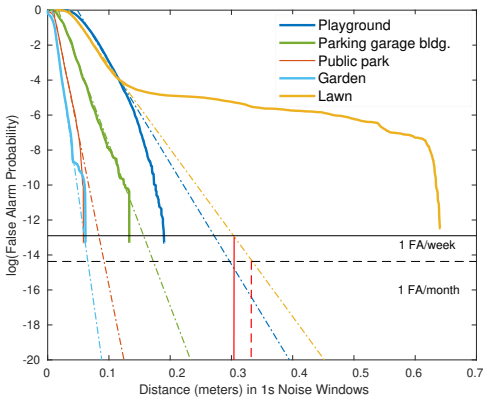
[33] Hanchuan Peng, Fuhui Long, and Chris Ding. 2005. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence* 27, 8 (2005), 1226–1238.

[34] Dhrubojyoti Roy, Christopher Morse, Michael A McGrath, Jin He, and Anish Arora. 2017. Cross-environmentally robust intruder discrimination in radar motes. In *2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. IEEE, 426–434.

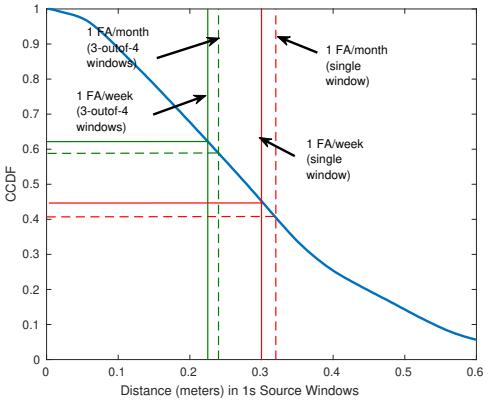
[35] Dhrubojyoti Roy, Sangeeta Srivastava, Pranshu Jain, Aditya Kusupati, Manik Varma, and Anish Arora. 2019. Lightweight, Deep RNNs for Radar Classification. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 360–361.

[36] Dhrubojyoti Roy, Sangeeta Srivastava, Aditya Kusupati, Pranshu Jain, Manik Varma, and Anish Arora. [n. d.]. Micro-power pulse-Doppler radar clutter and displacement source classification dataset. https://doi.org/10.5281/zenodo.3451407
[37] Dhrubojyoti Roy, Sangeeta Srivastava, Aditya Kusupati, Pranshu Jain, Manik Varma, and Anish Arora. [n. d.]. MSC-RNN: Multi-Scale, Cascaded RNNs for Radar Classification. https://github.com/dhruboroy29/MSCRNN
[38] Dhrubojyoti Roy, Sangeeta Srivastava, Aditya Kusupati, Pranshu Jain, Manik Varma, and Anish Arora. 2019. One size does not fit all: Multi-scale, cascaded RNNs for radar classification. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 1–10.
[39] Jürgen Schmidhuber. 1992. Learning complex, extended sequences using the principle of history compression. *Neural Computation* 4, 2 (1992), 234–242.
[40] The Samraksh Company. [n. d.]. BumbleBee Radar. https://goo.gl/ViMSiJ.
[41] The Samraksh Company. [n. d.]. .NOW with eMote. https://goo.gl/C4CCv4.
[42] UrbanCCD. [n. d.]. Array of Things. https://medium.com/array-of-things/five-years-100-nodes-and-more-to-come-d3802653db9f.
[43] Jiajun Wu, Yinan Yu, Chang Huang, and Kai Yu. 2015. Deep multiple instance learning for image classification and auto-annotation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3460–3469.
[44] Jinmian Ye, Linnan Wang, Guangxi Li, Di Chen, Shandian Zhe, Xinqi Chu, and Zenglin Xu. 2018. Learning compact recurrent neural networks with block-term tensor decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9378–9387.

A PARAMETER SELECTION FOR M-OUTOF-N DISPLACEMENT DETECTOR



(a) Clutter threshold selection for 1 FA/week and 1 FA/month



(b) Relaxation of per-window threshold through aggregation

Fig. 13. Shallow displacement detector parameter selection using the datasets from Table 3(a): here, M=3 and N=4

We discuss the parameter selection process for the unwrapped-phase displacement detector [34] referenced in Figures 8 and 9 in a principled manner. Figure 13(a) shows the cumulative distribution of unwrapped phase changes of environmental clutter, translated into real distance units, in various environments for 1-second integration windows from the clutter datasets in Table 3(a). The data is extrapolated using linear fitting on a logarithmic scale to estimate the required phase thresholds to satisfy false alarm rates of 1 per week and 1 per month respectively (derived using Bernoulli probabilities). We see that the unwrapped thresholds for 1 false alarm per week and month correspond to 0.3 and 0.32 m respectively. In this analysis, we fix the IQ rejection parameter at 0.9, which gives us the most lenient thresholds.

Figure 13(b) illustrates the CCDFs of phase displacements for all source types (humans, gym balls, dogs, cattle, and slow-moving vehicles) in our dataset combined, calculated over 1-second windows. Setting thresholds based on the previous analysis, the probability of false negatives per window is still significant. In practice, the algorithm

improves detection by basing its decision on 3-outof-4 sliding windows, where detectability improves since the threshold per window is now $\frac{3}{4}\times$ the original threshold. For 1 false alarm per week (month), the displacement threshold for the 3-outof-4 detector reduces to 0.22 m (0.24 m) per window, with an improved detection probability of 0.59 (0.62).

B TOP 16 HANDCRAFTED FEATURES SELECTED BY MAX-RELEVANCE MIN-REDUNDANCY

In this section, we list the top 16 amplitude, time, and frequency features that are selected by the mRMR algorithm for mote-scale Human vs. Non-human SVM classification out of a pool of 80 cross-domain features.

Table 11. Top 16 engineered features selected by mRMR for 1s, 1.5s, and 2s windows

1s			1.5s			2s		
Feature	**Threshold**	**Domain**	**Feature**	**Threshold**	**Domain**	**Feature**	**Threshold**	**Domain**
stdev(abs(signal))		Amplitude	stdev(abs(signal))		Amplitude	sum(abs(signal))		Amplitude
90th percentile phase diff.		Time	90th percentile velocity		Time	90th percentile velocity		Time
70th percentile phase diff.		Time	50th percentile velocity		Time	Distance		Time
50th percentile phase diff.		Time	Distance		Time	Time		Time
50th percentile velocity		Time	Time		Time	var(numHitBins)	14.144	Frequency
Distance		Time	var(numHitBins)	14.144	Frequency	max(numHitBins)/(time-8)	14.144	Frequency
Time		Time	max of excited frequencies	14.144	Frequency	max(numHitBins)/(time-8)	17	Frequency
var(numHitBins)	14.144	Frequency	var(numHitBins)	17	Frequency	var(numHitBins)	18.915	Frequency
median(numHitBins)/(time-4)	14.144	Frequency	var(numHitBins)	18.915	Frequency	max(numHitBins)/(time-8)	18.915	Frequency
var(numHitBins)	17	Frequency	max of excited frequencies	20	Frequency	max of excited frequencies	20	Frequency
median(numHitBins)/(time-4)	17	Frequency	var(numHitBins)	24.144	Frequency	max(numHitBins)/(time-8)	20	Frequency
var(numHitBins)	18.915	Frequency	Max of excited frequencies	25	Frequency	Total power	20	Frequency
Max of excited frequencies	18.915	Frequency	Width of excited frequencies	25	Frequency	var(numHitBins)	24.144	Frequency
Total power	24.144	Frequency	Total power	25	Frequency	max(numHitBins)/(time-8)	24.144	Frequency
Width of excited frequencies	25	Frequency	90th percentile of acceleration range		Time	Width of excited frequencies	25	Frequency
Total power	25	Frequency	90th - 10th percentile of velocity		Time	Total power	25	Frequency

While the amplitude and time domain features are straightforward to understand, some explanation of the frequency features is in order. Many of them are statistics on the number of *activated* bins in the STFT above a given power threshold across all frequencies in each time-step, referred to as *numHitBins*. Each frequency domain feature can be re-instantiated with different cutoff thresholds in the feature pool; the most relevant ones chosen by mRMR have their corresponding thresholds indicated in Table 11.