CROSSCOUNT: A Deep Learning System for Device-free Human Counting using WiFi

Osama T. Ibrahim, Walid Gomaa, and Moustafa Youssef

Abstract—**Counting humans is an essential part of many people-centric applications. In this paper, we propose CROSS-COUNT: an accurate deep-learning-based human count estimator that uses a single WiFi link to estimate the human count in an area of interest. The main idea is to depend on the temporal link-blockage pattern as a discriminant feature that is more robust to wireless channel noise than the signal strength, hence delivering a ubiquitous and accurate human counting system. As part of its design, CROSSCOUNT addresses a number of deep learning challenges such as class imbalance and training data augmentation for enhancing the model generalizability.**

Implementation and evaluation of CROSSCOUNT in multiple testbeds show that it can achieve a human counting accuracy to within a maximum of 2 persons 100% of the time. This highlights the promise of CROSSCOUNT as a ubiquitous crowd estimator with non-labour-intensive data collection from off-the-shelf devices.

Index Terms—**Crowd Counting, Human Counting, Recurrent Neural Networks, Device-free identification, Sequence Classification.**

I. INTRODUCTION

MONITORING the human count in a given area of interest is a crucial part in many pervasive applications such as smart guiding in museums, energy management in smart buildings, indoor analytics, and people evacuation in emergency situations. For example, in a retail store, lightening and air conditioning can be automatically adjusted based on the clients' density in each section. Furthermore, the occupancy statistics can assess the store sections that attract more visitors to plan for future business [1].

Due to its importance, human-counting has attracted the attention of the research community. For example, computer vision researchers, aided with the recent advancement in deep learning, presented high-accuracy human counting systems [2]–[5]. Images/videos captured by cameras from the area of interest are processed to estimate the human count. Most of these systems are built using Convolutional Neural Networks (CNN) with various architectures and feature optimization techniques. After the network is trained with a large set of labeled images/videos, the human density is estimated by processing images/videos of the crowd using the trained net. However, vision-based systems require high installation cost, suffer from blind spots and occlusion issues, require high computational power, are limited in functionality

in poor lightening conditions, and raise privacy concerns. In addition, they cannot work through the walls. This *through-wall* capability is highly-desirable in a number of applications such as law enforcement [6].

To address these issues, algorithms based on analyzing the RF signals have been introduced. In particular, they analyze the signal received from the already-installed wireless infrastructure. RF-based systems can be either device-based or device-free. In device-based systems, each human target must be equipped with a device, such as a cell phone [7], [8], which limits the system ubiquity. On the other hand, in device-free systems, the number of human targets inside an area is estimated by analyzing their impact on the wireless links covering this area, without requiring them to carry any device. The concept of RF device-free localization was introduced in 2007 [9], presenting human counting as one of the challenges that faces the newborn technique. Since then, a number of device-free counting techniques have been introduced based on different features and machine learning algorithms [10]–[13].

In an early study, Nakatsuka et al. [10] demonstrated the feasibility of using Received Signal Strength (RSS) of radio links to estimate human counting. The authors showed empirically that increasing the number of people leads to higher variance in the RSS signal of a single RF link. Based on that, they derived a linear formula that relates the human count to the RSS average and variance. The counting functionality in the Nuzzer system [11] extended this model to work on a large scale. Based on observations, the authors showed that the variance of a single link is not enough to differentiate clearly between the human count classes. Therefore, they proposed to use the average relative variance of 12 WiFi links to count only up to two persons with 81% accuracy and up to three persons with 75% accuracy. Adding more complication to the model, Yoshida et al. [12] examined non-linear regression to capture the relation between the RSS and the number of people. Specifically, they used a Gaussian kernel to perform regression of RSS absolute values of 10 wireless links. Depatla et al. [13] propose a probabilistic approach to calculate the RSS probability mass function (PMF) of one link as a profile for each case of human count. When testing using an RSS vector, they compare its PMF with the pre-calculated profiles and report the nearest one, achieving 25% exact counting accuracy with estimation error of 2 or less 63% of time.

The main limitation of the above approaches is that the area of interest should be covered by a large number of WiFi links in order to achieve an acceptable counting accuracy, which is not the case in many wireless environments and applications. One possible solution to resolve this trade-off

Osama T. Ibrahim is with the Wireless Research Center, Egypt-Japan University of Science and Technology (E-JUST), Alexandria and the Computer Systems and Engineering Dept., Zagazig University, Zagazig, Egypt

Walid Gomaa is with the Computer Science and Engineering Dept., E-JUST and Alexandria University, Alexandria, Egypt

Moustafa Youssef is with Alexandria University, Alexandria, Egypt

between links density and counting accuracy is to use channel state information (CSI) instead of RSS [14]–[16], where the data of all RF sub-carriers of every WiFi link is available for processing. Unfortunately, unlike RSS, reading CSI data is not widely supported by all wireless cards. In addition, both RSS- and CSI-based techniques cannot work-well in through-the-wall scenarios, due to the attenuation of the RF signal.

Recently, Depatla et al. [17] proposed to utilize the WiFi link blockage events as a discriminate feature instead of features depending on the RSS exact values. In particular, the system embeds the inter-arrival times between the link line of sight (LoS) blockages into a renewable stochastic process that models the human motion mathematically. In addition to providing through-wall counting, the blockage pattern performs well in case of counting moving targets; which incur more RSS variance affecting the estimation model. However, this system accuracy significantly degrades in a number of real-world scenarios, as we quantify in Section IV, due to the oversimplified assumptions in the used mathematical model. These simplifications include discarding the order of inter-arrival times which is an important part of the context information and simulating the human motion to generate the model training data. Besides, the proposed mathematical model is tailored for special cases of testbeds where the WiFi link is in the middle and aligned with the area of interest. For any different setup, the mathematical model in [17] does not fit leading to deteriorated performance.

In this paper, we present CROSSCOUNT, a through-wall human counting system that leverages a Recurrent Neural Network (RNN) to map a sequence of link inter-blockage temporal pattern to the human count using a single WiFi link. The idea is that, the higher the number of people in an area of interest the shorter the time between blocking a single WiFi link and vice versa. As part of CROSSCOUNT design, we introduce different modules to address practical issues such as reducing the labour-intensive calibration required for training a deep learning model as well as handling the imbalance in the number of training cases between the different counting classes.

Implementation and evaluation of CROSSCOUNT in different testbeds show that it can provide the exact human count 59% of the time. This increases to 100% to within two persons difference in count. This is achieved using the information of only a single WiFi link, highlighting CROSSCOUNT promise as a ubiquitous through-the-wall human counting system. To sum up, the main contributions of this work are threefold:

- We present the architecture and details of CROSSCOUNT: a deep learning system that leverages the temporal blockage information of a single WiFi link to provide accurate device-free human counting.
- Beneath the folds of CROSSCOUNT, we propose a novel technique for training data augmentation and class balancing to significantly decrease the data collection overhead.
- We implement the CROSSCOUNT and thoroughly evaluate its performance in clear and cluttered WiFi testbeds by counting up to 10 persons.

The rest of the paper is organized as follows. Section II provides an overview on how CROSSCOUNT works. Section III gives the details of CROSSCOUNT components and how it deals with different practical challenges. We evaluate the system performance in Section IV. Finally, Section VI concludes the paper and discusses future directions.

II. SYSTEM OVERVIEW

In this section, we present a typical scenario of how CROSSCOUNT works and give an overview about the information flow through its main modules. The details of these modules are described in subsequent sections. We assume an indoor area covered by a single WiFi link whose transmitter and receiver are behind the walls. There is an unknown number of people that are moving casually inside this room. Given only the RSS readings over time at the receiver, our goal is to infer the human count inside this area of interest.

The CROSSCOUNT system architecture is depicted in Fig. 1. The basic idea of CROSSCOUNT is to map a sequence of link-blockage time to the estimated count based on the observation that the more people in the area of interest the shorter the time between link blockages. To do this mapping, CROSSCOUNT leverages a recurrent neural network. In particular, for discretized time, the input sequence to the RNN is a stream of binary values which are 1 at the time instances when an LoS blockage event encountered and 0 otherwise. The output of the RNN is the estimated count.

CROSSCOUNT operates in three stages: preparing the training data, training the deep network, and finally classifying the input sequence by leveraging the trained network. During the first stage, the training sequences are collected *manually* in a light-weight process by recording the timestamps of the WiFi link virtual LoS blockages by a *single* walking human *without reading the link RSS*. However, during the online counting stage, the link-blockage events are calculated *automatically* by processing the captured WiFi link RSS stream at the receiver.

The typical way of collecting the training data in literature is to try all the combinations of the number of people in the area of interest [10], [12], [14]–[16]. This makes the training task labor intensive and limits the systems ability to detect a large number of people [18]. To address the scale and overhead of collecting the training data, CROSSCOUNT introduces a new technique that depends on collecting the training samples using **only a single person**. This single-person training data is then processed to automatically generate the training sets for all the other count classes. Specifically, during the *Training Data Preparation* stage, the **Training Data Collector** module records the LoS blockage events caused by a *single* person moving in the environment to generate the training data for the single-human count class. Thereafter, the **Training Data Synthesizer** module processes the collected data to generate the training set for multiple-user classes.

The generated data also suffers from the imbalanced data distribution for the different count classes. In addition, deep learning models require large amounts of training data. The **Training Classes Balancer** module tackles both of these issues by augmenting the training set with synthesized data.
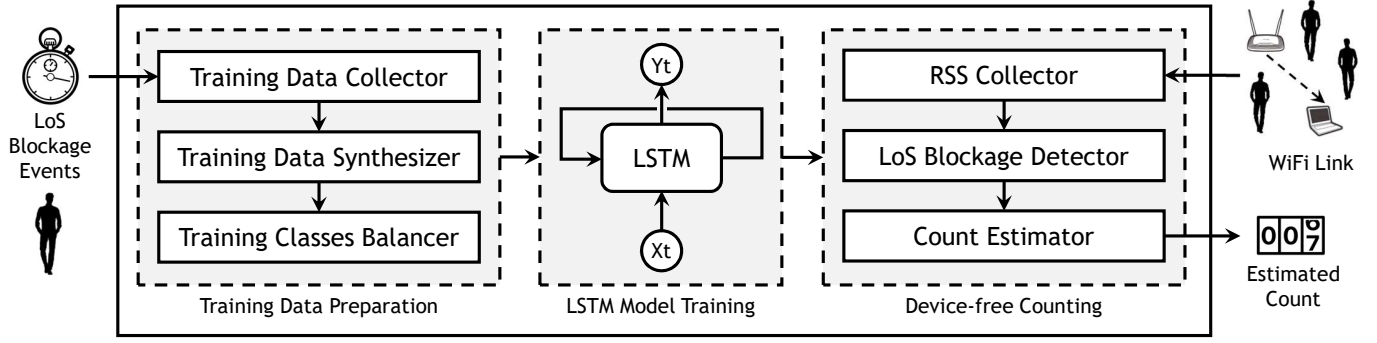
Fig. 1. CROSSCOUNT system architecture.

This also enhances the system generalizability and increases its ability to deal with the noisy characteristics of wireless channels.

CROSSCOUNT then trains a Long Short-Term Memory (LSTM) RNN network to characterize the blockage pattern of each human count class. The data preparation and training phases are done offline only once while deploying the system.

During the online system operation in the *Device-free Counting* stage, CROSSCOUNT estimates the unknown crowd count by extracting their LoS blockages sequence pattern and classifying it using the trained LSTM model. Specifically, the **RSS Collector** reads the signal strength received over the WiFi link for a specific time window. The **LoS Blockage Detector** module processes the RSS values and estimates the LoS blockage events timing encountered along this window in quantized time units. This generates a binary link blockage sequence that has a 1 in the time slot that the link was blocked inside the current window and zero otherwise. The input binary sequence is then passed to the **Count Estimator** module which uses the learned LSTM network to estimate the human count.

III. THE CROSSCOUNT SYSTEM

In this section, we provide the details of the main CROSS-COUNT modules, including diminishing the training data collection overhead, class balancing, and enhancing the system generalizability. We start by an overview of the CROSSCOUNT training process.

A. CROSSCOUNT *Training Process Overview*

As a supervised machine learning system, each sample in the CROSSCOUNT training set is sequence of the link blockage times which is labeled with the number of persons generating this sequence. The link blockage time is mapped to a bit map, where ones represent instances of time the link was blocked in a discrete time slotted system (Fig. 2).

The direct way to collect these samples is for the specified number of persons to move inside the area of interest for a certain time window, w, record the time instances they cross the LoS, and label the generated sequence with the human count. This should be repeated for each and every crowd count class, i.e. number of humans inside the area of interest. Moreover, each class should have a large number of training samples to generate a well-trained deep network.
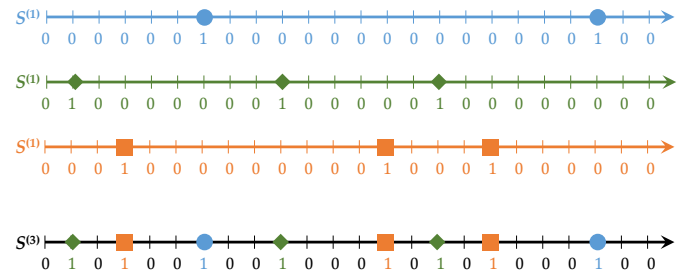

Fig. 2. An example describing the superposition of 3 single-person sequences ($S^{(1)}$) to generate a 3-persons sequence ($S^{(3)}$). The circles, diamonds, and squares represent the LoS blockages of the first, second, and third sequence respectively.

Accordingly, collecting the training data is a labor-intensive and time consuming task. Some previous work, e.g. [13], [17], proposed a human motion model to simulate this task. However, the mathematical model assumes simple motion that does not capture real life complex scenarios, affecting the system accuracy as we quantify in Section IV-D. CROSSCOUNT resolves this challenge by collecting the training samples for only a *single* user, form which the whole training data for any arbitrary number of humans inside the area of interest is extrapolated.

1) Training Data for Single Target: The *Training Data Collector* module allows a single person to move randomly inside the area of interest while recording the timestamp each time she crosses the WiFi link virtual LoS. Note that the training process does not require processing the RSS of the link as it is based on visually recording when the user crosses the virtual line between the transmitter and receiver. This is because CROSSCOUNT depends only on the time of link blockage and not on the specific RSS.

Assuming a discrete time space, a training sample takes w time steps, to be collected. This is repeated m times to collect different training sequences, each of length w. We call these manually collected samples the "*the original training samples*".

2) Reducing Training Data Collection Overhead for Multiple Persons: CROSSCOUNT generates the training samples for the **multi-person** classes using the collected data from the **single-person** case as described in the previous section. To do that, we assume that each person is moving independently

TABLE I
AN EXAMPLE ON THE IMBALANCED TRAINING DATA FOR 0-5 COUNT CLASSES. NOTE THAT THE ZERO-COUNT CLASS HAS JUST ONE EXAMPLE REPRESENTED BY THE BIT SEQUENCE OF ALL ZEROS, I.E NO LINK BLOCKAGE EVENTS IN AN EMPTY AREA OF INTEREST.

Count Class	0	1	2	3	4	5
No. of Samples	1	15	105	455	1365	3003

Fig. 3. CROSSCOUNT network diagram.

of the others. As a result, the LoS blockage sequence of a user is independent from the others and the blockage pattern for **multiple** persons can be calculated as the *superposition* of the individual persons' blockages. In particular, CROSSCOUNT randomly selects k sequences out of the m collected samples of a single person and consider each of them as a link blockage sequence for a different user. The k-person count class training sample is synthesized from the superposition of these k sequences. Fig. 2 shows an example on how to generate a training sequence for 3 persons from 3 single-person training sequences. For each time instance, the superposed sequence reports an LoS blockage if any of the single-person sequences encounters a blockage at this instance. For each multi-person count class, the *Training Data Synthesizer* module applies the superposition technique over all the available combinations of the m original training samples to generate the higher count classes training set.

3) Handling the Class Imbalance Problem: The *Training Data Synthesizer* uses the superposition technique to generate $\binom{m}{n}$ training samples for each n-count class. For instance, Table I lists the size of the training set for the 0 to 5-count classes as generated from $m = 15$ original training samples. There is only one training sample for the 0-count class which is a sequence of zeros as it does not encounter any LoS blockage during the counting window. The table shows that the class distribution is not uniform and is severely skewed. This imbalanced class data problem is well-known in machine learning [19], [20] and leads to a bias in the classifier towards the classes with large training data.

Traditionally, random oversampling and sub-sampling are simple and well-established solutions for this class imbalance problem [19]. Oversampling increases the minority classes by replicating the training samples, while sub-sampling decreases the majority classes by randomly removing some of its samples. Oversampling leads to overfitting the training data [21]. Therefore, in CROSSCOUNT, the *Training Data Balancer* module sub-samples the majority classes training set. In addition, CROSSCOUNT also augments the data from the minority classes by injecting noise into the training samples in order to improve the algorithm generalization capability and system robustness [22]. Noise is injected by randomly flipping some of the bits in the link-blockage bit steam input. This simulates injecting false link-blockage events or missing an actual link-blockage event, which may occur in reality due to the noisy wireless channel. This noisy training data increases CROSSCOUNT robustness and avoids overfitting.

The Training Data Synthesizer and Class Balancer modules are implemented as follows. The main input from the Training Data Collector is the m manually collected original single-person sequences, from which the multiple-persons training
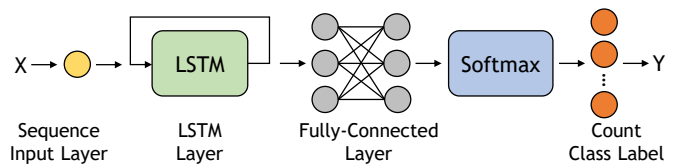
data is synthesized. Each training sequence is w seconds in length. For the zero-entity class, there is only one original training sample which is a sequence of zeros with length w, expressing that no LoS blockage happened along the time window. This sequence is noised by flipping any single bit along the sequence length. This generates a total of $w + 1$ training sample for the 0-count case, limiting all other classes to this size of training samples, to ensure a balanced training set. The training data for any other n-count class is generated as follows. A random combination of n original sequences is selected from the m inputs and a superposed training sample is synthesized from the bitwise logical OR of all the randomly-selected sequences. Another superposed sample can be generated by selecting another random sequence combination. We keep repeating this till meeting the class balancing condition where the number of generated sequences is equal to $w + 1$. If all the available $\binom{m}{n}$ sequence combinations are processed before meeting the balancing condition, the training set for the current n-class is augmented by data noising. A new training sequence is generated by selecting a random superposed sequence, flipping any random bit inside, and appending it back to the training set.

B. Model Training

Recurrent Neural Networks (RNN) model the contextual information in the input sequences. RNNs have been employed in many applications whose input data are sequences of features such as speech recognition [23] and handwriting alignment [24]. However, when dealing with long sequences, traditional RNNs suffer from the vanishing gradient problem during training [25]. Long Short-Term Memory (LSTM) [26] architecture is one of the most commonly-used solutions to this problem. Therefore, CROSSCOUNT leverages them to capture the contextual information in the link-blockage binary input sequence. Figure 3 shows the CROSSCOUNT LSTM architecture. The input sequence is one temporal bit stream generated from the deployed WiFi link, representing the link-blockage events as described in Section III. Therefore, we employ a one-dimensional input layer. Each crowd count is an output class of the network. Accordingly, the size of output layer is determined and preceded by a fully connected softmax for classifying among more than two output classes. The LSTM layer implements the tanh and sigmoid functions as the default configuration for cell state and gates activation respectively. The output layer implements cross entropy loss function.

During the Model Training phase, the LSTM network is trained using the original and synthesized training samples.
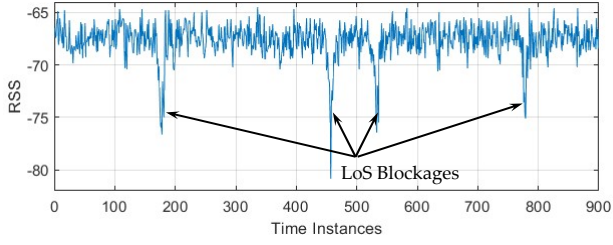
(a) Single Person.
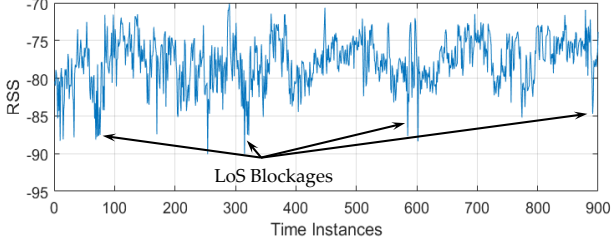
(b) Three Persons. For visual clarity, not all the blockages are indicated.

Fig. 4. The RSS measurements of a WiFi link while some persons are moving around. The time instances when a person crosses the LoS are highlighted.

The synthesized data allows the model to generalize better to noisy unseen data.

C. Online Phase: Human Counting

In this stage, the trained network is used to estimate the unknown crowd count. The WiFi link RSS stream is analyzed to extract the LoS blockage pattern of the current crowd. Feeding this blockage sequence to the trained LSTM activates a forward path to an output class which will be reported as the count estimate.

1) The RSS collector: The RSS reader is a lightweight agent running on the receiver to record the temporal change of the Received Signal Strength (RSS) from the transmitter along with its time stamp. Finally, a sequence of l timestamped readings, $\{(r_0, t_0), (r_1, t_1), ..., (r_l, t_l)\}$, is collected and sent to preceding modules. Where each reading is streamed as a pair of the RSS value and time (r, t).

2) LoS Blocking Detection: Fig. 4 shows a typical example of the RSS measurements of a deployed WiFi link while a human is moving around. The figure shows that the RSS measurements are significantly attenuated when someone crosses the LoS. The figure shows that there is a down pulse whenever the target person crosses the link LoS while the other fluctuations are due to the multipath effect [17]. From the above observations, the RSS fluctuations due to multipath are limited within a certain level around the mean value, while the LoS blocking has a higher down pulse. Accordingly, the *LoS Blockage Detector* of CROSSCOUNT captures an LoS blockage by a simple thresholding process on the RSS changes. Specifically, at any time instance if the RSS exceeds its mean value $\bar{r}$, by a certain threshold, τ, an LoS blocking event is declared at this instance. Finally, the module converts the detected LoS blockage events into a binary stream

TABLE II
DEFAULT SYSTEM PARAMETERS.

Parameter	Range	Default value (Testbed $1, 2$)
Blockage Detection Threshold (τ) **(dBm)**	0 - 10	5, 5.5
Estimation Window (w) **(min.)**	1 - 5	5
LSTM layer size (units)	10 - 100	100
Number of training epochs	$10 - 150$	120, 150
Training mini-batch size (samples)	$1 - 30$	15, 3

$B = \{b_0, b_1, ..., b_{w-1}\}$, where

$$b_i = \begin{cases} 1 & \text{if} \quad r_j \geq \bar{r} + \tau, \ T_i \leq t_j < T_{i+1} \\ 0 & \text{otherwise} \end{cases}, \quad (1)$$

$$\bar{r} = \frac{1}{l} \sum_{k=0}^{l-1} r_k \quad (2)$$

Where T_i is the ith time step.

3) Count Estimation: The crowd count in the current counting window is estimated by classifying its blockage pattern. The sequence B is fed forward to the trained LSTM, and eventually, CROSSCOUNT reports the output class as the human count estimate.

IV. EVALUATION

In this section, we evaluate the CROSSCOUNT performance. We start by describing the experimental testbeds and training process, followed by testing the effect of system parameters on counting accuracy. Finally, we compare CROSSCOUNT performance to the state-of-the-art RF human counting systems. Table II contains the default system parameters values used throughout the evaluation section.

A. Experimental Testbeds

We evaluated CROSSCOUNT in two testbeds. The first one is a 19.8 m^2 room in our lab as shown in Fig. 5a; it is a controlled environment with low multipath where the WiFi link is deployed right in the middle of the room. This is similar to the testbed used in [17]. The second testbed is more complex, as shown in Fig. 5b: it is a larger hall of 50 m^2, rich in multipath due to its furniture, and the WiFi link is not aligned with the hall nor in the middle. In both testbeds, the transmitter is mounted to the wall at height of 185 cm, the receiver is placed 75 cm from the floor, and the walls are made of bricks.

B. Training Data

The original training set is collected using a *single* person who walks in each testbed for 100 minutes generating a 100-minutes-long sequence. This sequence is divided into sub-sequences of w-length each. Following the data synthesis process described in Section III-A3 with the default counting window, 301 training sequences are generated per count class resulting in a total training set of $3.3k$ samples when testing using 10 persons. The details and types of the generated training sequences are listed in Table III.
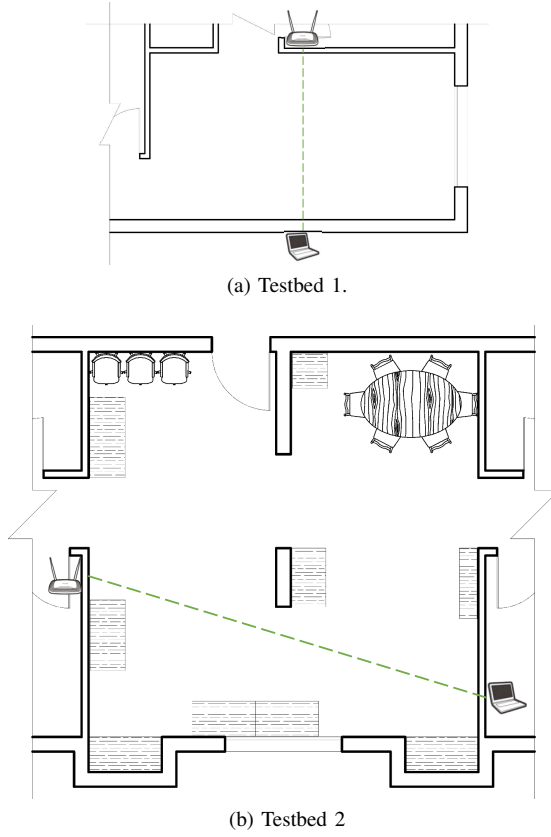
(a) Testbed 1.

(b) Testbed 2

Fig. 5. Evaluation Testbeds.

TABLE III
COLLECTED RAW DATA FOR TRAINING SEQUENCES

Count Class	0	1	2	≥ 3
Collected Sequences	1	20	0	0
Synthesized Sequences	0	0	190	301
Generated Noisy Sequences	300	281	111	0

The CROSSCOUNT model was trained using stochastic gradient descent with momentum (SGDM) optimizer, with 0.9 momentum and 0.01 initial learning rate. On average, the model training process takes us 5 hours to finish on a Dell XPS 8500 desktop with a core i7 processor and 12 GB memory.

C. Effect of System Parameters

In this section, we evaluate the effect of changing the system parameters on the counting performance as reported by the Absolute Counting error ε, where ε is the summation of the absolute difference between real and estimated counts in all cases.

$$\varepsilon = \sum |C_r - C_e| \qquad (3)$$

Where C_r, C_e are the real and estimated counts respectively.

1) LoS Blockage Detection Threshold: Fig. 6 shows the absolute counting error at different values of the LoS blockage detection threshold (τ) in both testbeds. For small thresholds, any minor fluctuation in RSS values is falsely detected as an LoS blockage; this increases the LoS blockage rate and reports extra persons in the testbeds. In contrast, high thresholds lead to missing real LoS blockage events, underestimating
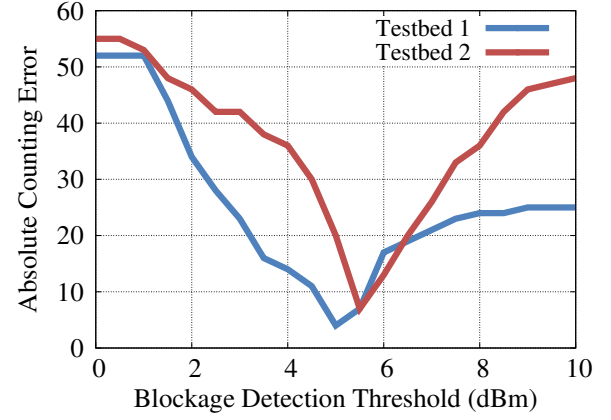

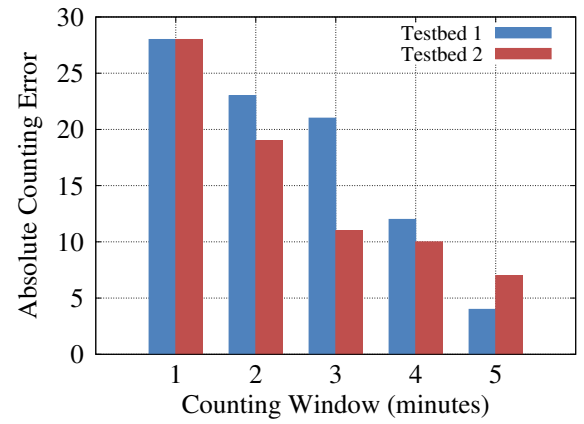Fig. 6. Evaluation of blockage detection threshold.

Fig. 7. Evaluation of the counting window length.

the actual crowd number and increasing the absolute error. Throughout the rest of the evaluation section, we set $\tau = 5$ and 5.5 as the default blockage detection thresholds for testbeds 1 and 2 respectively as they lead to the best performance.

2) Counting window length: Figure 7 shows the effect of various time windows w on the counting performance. For short windows, the information reflected by the blockages sequence encountered within the window is not enough for providing accurate count estimates. Increasing the window size to 5 minutes, as used in this paper, gives the best counting accuracy. Note that this is the same estimation delay used in literature [12], [13], [17]. The system designer needs to tune this parameter to trade-off latency and accuracy of estimation based on her specific application need.
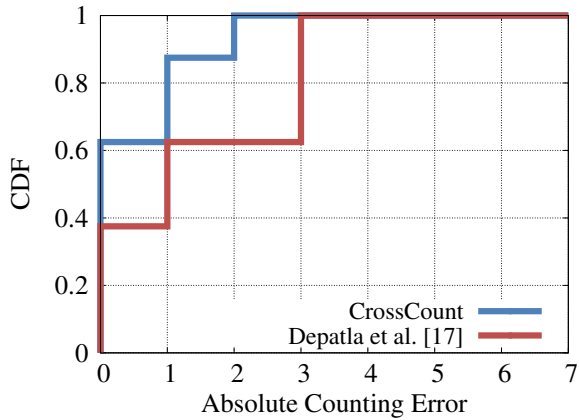
D. Comparisons with other systems

In this section, we compare the CROSSCOUNT performance with Depatla et al. [17] as the most recent related work. Moreover, the functionality of the two systems is based on the LoS blockage of a single WiFi link using RSS. For this comparison, 10 volunteers walked through the testbeds. Due to the limited area of testbed 1, only 7 persons participated. The distribution of the absolute counting error ε for the two testbeds is reported in Fig. 8 and detailed in Tables IV,V. In Testbed 1, CROSSCOUNT achieve 63% exact count accuracy

TABLE IV
COUNTING RESULTS OF TESTBED 1. LIGHTER COLORS REFER TO BETTER PERFORMANCE.
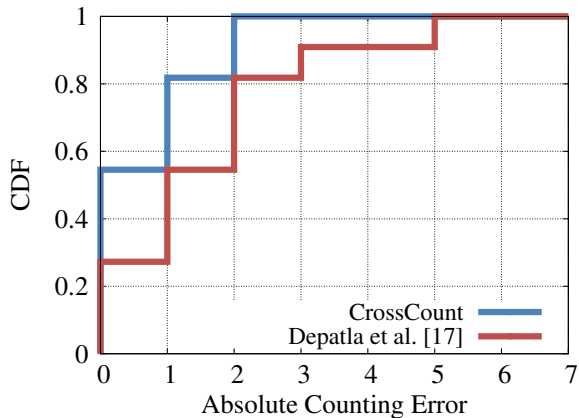
Real Count	CrossCount		Depatal et al. [17]	
	Estimation	Error	Estimation	Error
0	0	0	0	0
1	1	0	1	0
2	2	0	2	0
3	3	0	2	−1
4	4	0	3	−1
5	4	−1	2	−3
6	4	−2	3	−3
7	8	+1	4	−3

TABLE V
COUNTING RESULTS OF TESTBED 2. LIGHTER COLORS REFER TO BETTER PERFORMANCE.

Real Count	CrossCount		Depatal et al. [17]	
	Estimation	Error	Estimation	Error
0	0	0	0	0
1	2	+1	3	+2
2	4	+2	4	+2
3	3	0	5	+2
4	4	0	5	+1
5	6	+1	5	0
6	6	0	6	0
7	7	0	6	−1
8	9	+1	7	−1
9	7	−2	6	−3
10	10	0	5	−5

(a) Testbed 1

(b) Testbed 2

Fig. 8. CDF of absolute counting error

with only a maximum of 2-count-difference all the time, while Depatla et al. [17] delivers only 39% exact count accuracy. Depatla et al. [17] do not take advantage of the arrival order of blockage events when processing the given counting window unlike CROSSCOUNT which consider the whole context information including the arrival order leading to improved performance. The counting estimates are less accurate in Testbed 2 for both the two systems due to the environment complexity. CROSSCOUNT could achieve lower exact accuracy of 55% while maintaining a maximum error of 2 count difference, while Depatla et al. [17] degraded to 27% counting accuracy and in some cases it reports 5 count difference out of 10 persons. This can be explained by noting that the mathematical model in Depatla et al. [17] is tailored for special cases of testbeds where the WiFi link is in the middle and aligned with the area of interest. These assumptions hold in Testbed 1, but is not true in Testbed 2, leading to increasing the accuracy improvement for sake of CROSSCOUNT over Depatla et al. [17] in the second testbed.

V. DISCUSSION

In this paper, we proposed a novel idea for counting people by classifying their blockage pattern on WiFi links using deep learning. We introduced a data synthesizing technique that augmented the training set for robust training and provided a lightweight calibration phase. We conduced many experiments that proved the idea and showed that the proposed data generation technique could capture the reality to a reasonable extend. However, the presented system has some limitation that are encouraged to be addressed in the future extensions.

First, CROSSCOUNT generated the multiple-person training data by superposing the blockage pattern of a single person. However, further improvement in system performance is expected when simulating the multi-person data as the convolution of RSS signals of single-person. Moreover, the superposition technique could be enhanced by applying left\right translation on blockage sequences before being superposed. Second, CROSSCOUNT implemented a higher level data noising approach where the blockage events are noised to simulate the RSS changes due to wireless characteristics. However, a lower noising level could be considered by simulating the signal amplitude changes. Finally, CROSSCOUNT assumed a casual human motion with non-zero speed while collecting the training and testing sequences. Nonetheless, more sophisticated walking patterns could be investigated, besides, some special scenarios of human movement might be handled, such as when a human stands at the LoS generating a contiguous blocking sequence, among others.

VI. CONCLUSION

In this paper, we presented the design, implementation, and evaluation of CROSSCOUNT: an accurate human counting system based on Recurrent Neural Networks. The system provides different techniques for handling a number of challenges found in the literature such as through-wall signal weakness, labor-intensive data collection, imbalanced training data, high training overhead, high number of data links, and

unavailability of CSI data in commodity devices. The main idea is to process the WiFi link blockage inter-arrivals rather than depending on the statistical features extracted from RSS values. By classifying the blockage pattern using an LSTM network, CROSSCOUNT achieved superior counting accuracy than the current state-of-art single-link RF-based counting systems.

Currently, we are extending CROSSCOUNT in different directions including extending the system to work with multiple links and leveraging CSI information when available.

REFERENCES

[1] O. Perdikaki, S. Kesavan, and J. M. Swaminathan, "Effect of traffic on sales and conversion rates of retail stores," *Manufacturing & Service Operations Management*, vol. 14, no. 1, pp. 145–162, 2012.

[2] L. Boominathan, S. S. S. Kruthiventi, and R. V. Babu, "CrowdNet: A deep convolutional network for dense crowd counting," in *Proceedings of the 2016 ACM on Multimedia Conference*, ser. MM '16. New York, NY, USA: ACM, 2016, pp. 640–644.

[3] L. Zeng, X. Xu, B. Cai, S. Qiu, and T. Zhang, "Multi-scale convolutional neural networks for crowd counting," *CoRR*, vol. abs/1702.02359, 2017. [Online]. Available: http://arxiv.org/abs/1702.02359

[4] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 833–841.

[5] Z. Li, L. Zhang, Y. Fang, J. Wang, H. Xu, B. Yin, and H. Lu, "Deep people counting with faster R-CNN and correlation tracking," in *Proceedings of the International Conference on Internet Multimedia Computing and Service*, ser. ICIMCS'16. New York, NY, USA: ACM, 2016, pp. 57–60.

[6] A. Hunt, C. Tillery, and N. Wild, "Through-the-wall surveillance technologies," *Corrections Today*, vol. 63, no. 4, p. 132, 2001.

[7] H. Li, E. C. L. Chan, X. Guo, J. Xiao, K. Wu, and L. M. Ni, "Wi-Counter: Smartphone-based people counter using crowdsourced Wi-Fi signal data," *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 4, pp. 442–452, Aug 2015.

[8] J. Weppner and P. Lukowicz, "Bluetooth based collaborative crowd density estimation with mobile phones," in *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, March 2013, pp. 193–200.

[9] M. Youssef, M. Mah, and A. Agrawala, "Challenges: Device-free passive localization for wireless environments," in *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*, ser. MobiCom '07. New York, NY, USA: ACM, 2007, pp. 222–229.

[10] M. Nakatsuka, H. Iwatani, and J. Katto, "A study on passive crowd density estimation using wireless sensors," in *The 4th Intl. Conf. on Mobile Computing and Ubiquitous Networking (ICMU 2008)*. Citeseer, 2008.

[11] M. Seifeldin, A. Saeed, A. E. Kosba, A. El-Keyi, and M. Youssef, "Nuzzer: A large-scale device-free passive localization system for wireless environments," *IEEE Transactions on Mobile Computing*, vol. 12, no. 7, pp. 1321–1334, July 2013.

[12] T. Yoshida and Y. Taniguchi, "Estimating the number of people using existing WiFi access point in indoor environment," in *Proceedings of the 6th European Conference of Computer Science (ECCS15)*, 2015, pp. 46–53.

[13] S. Depatla, A. Muralidharan, and Y. Mostofi, "Occupancy estimation using only WiFi power measurements," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 7, pp. 1381–1393, July 2015.

[14] W. Xi, J. Zhao, X. Y. Li, K. Zhao, S. Tang, X. Liu, and Z. Jiang, "Electronic frog eye: Counting crowd using WiFi," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, April 2014, pp. 361–369.

[15] S. Di Domenico, M. De Sanctis, E. Cianca, and G. Bianchi, "A trained-once crowd counting method using differential WiFi channel state information," in *Proceedings of the 3rd International on Workshop on Physical Analytics*, ser. WPA '16. New York, NY, USA: ACM, 2016, pp. 37–42.

[16] Y. K. Cheng and R. Y. Chang, "Device-free indoor people counting using Wi-Fi channel state information for internet of things," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–6.

[17] S. Depatla and Y. Mostofi, "Crowd counting through walls using WiFi," in *Proc. IEEE International Conference on Pervasive Computing and Communications (PerCom)*, March 2018.

[18] O. T. Ibrahim, W. Gomaa, and M. Youssef, "Zero-calibration device-free localization for the iot based on participatory sensing," in *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec 2018, pp. 1–7.

[19] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, Sept 2009.

[20] H. Rizk, A. Shokry, and M. Youssef, "Effectiveness of data augmentation in cellular-based localization using deep learning," in *Proceedings of the International Conference on Wireless Communications and Networking (WCNC). IEEE*, 2019.

[21] D. Mease, A. J. Wyner, and A. Buja, "Boosted classification trees and class probability/quantile estimation," *J. Mach. Learn. Res.*, vol. 8, pp. 409–439, May 2007.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[23] A. Graves, A. r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 6645–6649.

[24] F. Raue, W. Byeon, T. M. Breuel, and M. Liwicki, "Parallel sequence classification using recurrent neural networks and alignment," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, Aug 2015, pp. 581–585.

[25] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, ser. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2012.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.