Radio Tomography and Machine Learning for Person Identification

Robert Hulbert (rdhulber@calpoly.edu), Lucy Bowen (lrbowen@calpoly.edu),
Amanda Strand (astrand@calpoly.edu), Andreas Apitz aapitz@calpoly.edu)
California Polytechnic State University, San Luis Obispo

***Abstract*—Radio tomography is a rapidly developing field with numerous applications in home security, elder care, and other industries. This paper introduces an example of a radio tomography system which recognizes a target walking through a room and identifies them from a set of known test subjects. Utilizing the open source firmware patch, Nexmon, on a Nexus 5 smartphone, fluctuations in the 5 GHz Wifi-band channel state information (CSI) are recorded and used to train a neural network for reliable classification and noise suppression. The trained neural network could distinguish an individual from other individuals or empty room with 86% accuracy.**

***Index Terms*—Radio Tomography, Software Defined Radio, TensorFlow, BiDirectional Neural Network, Long Term Short Term Memory, Channel State Information, Nexmon**

I. Introduction

RADIO tomography allows for locational and displacement characterization of an object interfering with the communication channel. A wireless system constructed to identify people has two main applications: security and safety. However, wireless research on radio tomography in recent years has focused on attaining greater detail and accuracy of characterization, which led to utilizing non-commodity hardware. This specialized hardware makes reproducing results challenging and adds a high barrier to entry by the general public. This paper shows that radio tomographic technology can be implemented on commodity hardware using open source software to characterize interfering objects. With machine learning techniques, this system can rapidly and accurately identify multiple objects operating in the wireless zone. This non-invasive identification system can be utilized by a home owner to monitor the state of their house and prevent intrusion by unrecognized people, or utilized by the police (with a warrant) to plan a raid at minimal risk thanks to prior knowledge of the building's occupants. This commercial-minded system provides non-invasive identification of people in a given setting to improve security and safety.

This paper implements a commodity hardware, open source radio tomographic system with machine learning techniques to identify interfering objects. It uses a commodity router and phone to extract CSI data from the surroundings, which can then be analyzed by the neural network to identify people in the surroundings. People are identified based on their gait, ie. each person's way of moving is unique enough to identify the person from this data. This is useful in characterizing a mobile phone environment and, potentially, tracking the movements of people.

II. Background

Radio tomography utilizes the principles of reflection and scattering to characterize objects within the wireless zone. In the 802.11a/g/n/ac Wi-Fi standards, equipment utilizes the multiple paths between antennas to identify and correct for the attenuation due to scattering and reflection in the channel [1]. This information known as Channel State Information (CSI), by correcting the channel attenuation, characterizes any interfering object in the channel paths to invert the effect on the channel. CSI continuously updates to maintain the stability of the channel. In a static environment, the CSI will reach equilibrium, but any disturbance in the channel will generate a change in the CSI. Moving objects that interfere with the propagation of the transmission will produce a Micro-Doppler shift in the channel. A Micro-Doppler shift refers to the enormous scale difference of the velocity of the object in comparison to the velocity of the light transmitting the information and the minor shift in frequency that occurs [1] [2]. Micro-Doppler shifts will produce amplitude and phase shifts in the CSI that represent the unique movement of the object through the space. These shifts are measured, recorded, and analyzed to determine the nature of the movement.

Most commodity Wi-Fi chipsets do not support the extraction of CSI. However, Nexmon, an open source firmware patch for specified Broadcom chipsets such as those found on the Raspberry Pi 3B and Nexus 5, allows the monitoring of wireless network traffic [3]. The Nexmon group further augmented the Nexus 5 Broadcom chip, BCM4339, firmware to extract CSI. By copying the values out from the hardware registers and generating a UDP packet, a pcap file monitoring the modified wireless interface will capture the generated UDP packet containing the CSI [4]. The UDP packet data section contains 255 singed two-byte pairs representing the real and imaginary components of each of the 255 potential sub-carriers in the full 80 MHz bandwidth. After CSI extraction, the list of pairs can be analyzed as a spectrogram centered about the carrier frequency (Wi-Fi band) with a range covering the bandwidth generated by the router for that channel (20, 40, 80 MHz). This spectrogram displays the changes in the frequency spectrum over time. These spectrograms contain information about the motion of the interfering objects in the channel through variation in frequency response. This frequency response change indicates the doppler shift caused by the interfering object. A two dimensional grey-scale image stores the information from the CSI.

Machine Learning techniques rapidly classify enormous data sets after validated training. Convolutional Neural Networks (CNNs) constitute a machine learning technique adept at characterizing images. CNNs analyze images by convolving feature masks with the image and generating sub-images emphasizing certain specified characteristics. The next iteration pools the generated sub-images, often selecting a maximum, minimum, or average of the pixel values. These steps repeat until a singlular vector remains which feeds into a feed-forward neural network. The feed forward neural network utilizes multiple layers of neurons that trigger based on their weights, inputs, bias, and activation specification. The output generated by the CNN corresponds to the unique motion of the original object.

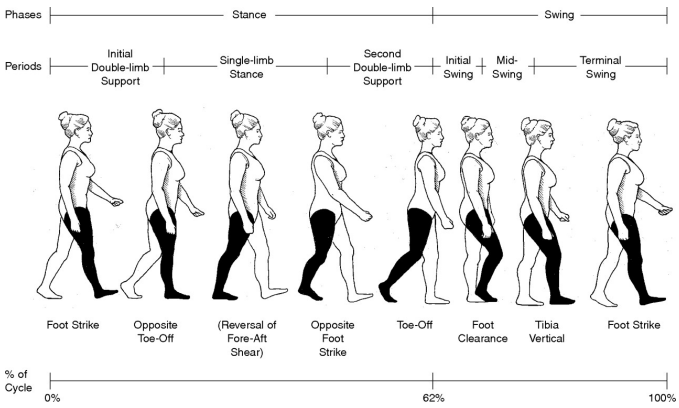


Fig. 1: Different Phases of a Human Gait

III. Experimental Design

A. CSI Capture Setup

The Nexus 5 with Nexmon installed served as the capture device for the CSI. This device would capture the 100 ms interval beacon frames generated from the Sagemcom F@st 5260 router. The router beaconed on Wifi Channel 44 with a bandwidth of 20 MHz. The Nexmon utility filter option allowed only the beacon frames on the channel and bandwidth specified to be processed. The Nexmon firmware would then produce UDP packets that were captured through the *tcpdump* commandline tool monitoring the wireless interface. The CSI data extraction script converted one packet in the pcap file to one row in the csv file. The signed short pairs for each sub-carrier frequency were converted into a signed short magnitude. Each row in the csv file would contain the time of the packet and the parsed 255 signed magnitudes resulting in a 1x256 row vector. All the packets in the file were iteratively converted and placed into the csv file in chronological order.

B. Preprocessing Method

The Nexmon supplies frequency information already in the form of CSI, where each packet contains 255 values showing the transfer function amplitude at frequencies about the carrier frequency, 5.22 GHz. A grey-scale image from the csv file must be produced for processing by the convolutional neural network. The csv file data can be converted to a gray-scale spectrogram which represents the change in magnitude of the frequencies through time. Each spectrogram must first be preprocessed by converting the frequency response into a two-dimensional grayscale image. This coded image will serve as training data for the neural network.

As long as information loss is minimized, it makes no difference how the data is encoded in the image, so long as all inputs are processed in the same way. Therefore, a procedure was selected which minimizes computational intensity for the preprocessing stage. In this case, the frequency response input is saved as a 2D array of signed short (2-byte) values. The desired 2D image is an array of uint8 values. A significant amount of data is lost through this truncation due to quantization error, but this conversion is necessary due to constraints from the Nexmon and the neural network. Quantization error is minimized by adding an offset to ensure that all values are positive, then scaling the matrix to cover a range from 0x00 to 0xFF. The matrix is then converted to uint8.

In addition to converting to a positive value, the output spectrograms were truncated to 255x56 images in order to remove the extraneous columns. These extra columns were present because they represented the other WiFi carriers left unused during the test. As they were unused, they also held no useful information regarding the subject's gait.

C. The Neural Network

Through the use of Keras, a python wrapper for TensorFlow, a convolutional neural net (CNN) with a BiDirectional Long Short Term Memory (LSTM) hidden layer was implemented to classify and identify the individual who generated the corresponding spectrograms. Through extensive iteration and hyper parameter optimization, the optimal solution consisted of two convolutional layers, a bidirectional LSTM layer, and a dense layer detailed in Fig. 2. Pooling layers after each of the convolutional layers reduced the parameter size. To decrease overfitting and improve validation accuracy, dropout layers randomly removed outputs after each convolutional layer as well. The LSTM hidden layer provides a memory to the system that increases the likelihood of recognizing past patterns. Finally, a fully connected layer activated by softmax produces a 1x5 matrix representing the probability of the output from the NN being one of the individuals or empty room.

Notable observations: Initially the NN was trained with the Adam optimizer, which allowed for an accuracy of 80%, but switching to the rmsprop optimizer allowed for an additional 6% gain. It was also observed that having a kernel of 3x3 instead of a kernel of 5x5 for the pooling layers in any circumstance would cause the network to be unable to rise above 20% training accuracy. Finally, if the second max pooling layer is removed before it is fed into the LSTM layer, there is a drop of 20% accuracy.

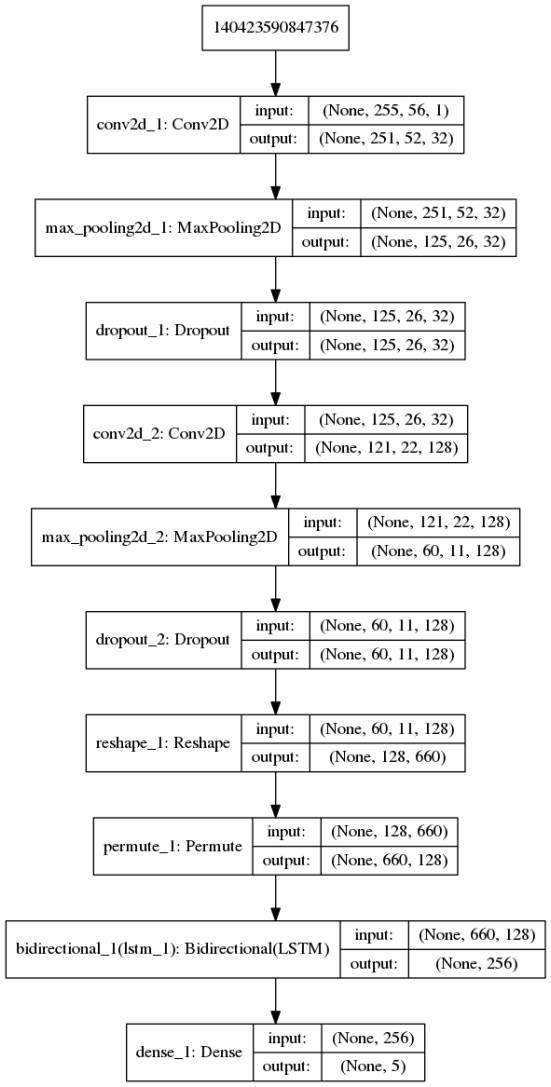


Fig. 2: Neural Network Architecture

D. Training the Neural Network

The Nexmon firmware patch allows a device containing a Broadcom chipset to monitor CSI and output complex values of each sub-carrier affected by a person walking through the room. The test set up for collecting data requires each test subject to walk across the testing room in order to collect enough packets of information to train the NN for each person. This data is then converted into smaller sets of images for training the NN, per the preprocessing method discussed earlier.

Some people move through a space slowly and others move quickly, resulting not only in differences in personal gait, but also differences in the periodicity of data collected for each person. In general, about 50 minutes worth of data are obtained for each person with one state for the empty room state. For training data, the desired outcome is represented by a label in Tensorflow, which tells the Neural Network which state the data represents.

Data was obtained in the living room at 650 Graves St. #3. To prepare the room for testing, all windows were closed, all blinds were drawn, and all furniture was removed. The only furniture left in the room is the couch where the router was placed, broadcasting the beacon frame every 0.1 seconds. The doors were closed during all data collection phases. Fig. 3 shows the floor plan of this room.

(a) View of Back Corner (b) View of Front Corner

(c) Computer Setup

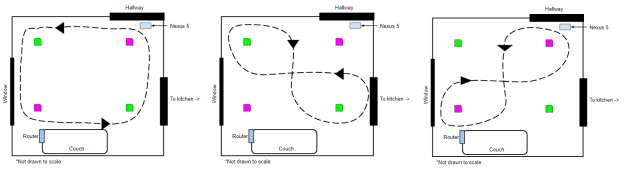
Fig. 3: Room Layout for Data Collection

The Nexus 5 phone installed with Nexmon records the CSI of the test room when unoccupied to define the equilibrium state of the room. Spectrogram samples from the empty room are included with training and testing data. Then, each subject walks through the room along three predetermined paths (Fig. **??**) for 8 to 15 minutes each, until 10,200 packets have been recorded. The 10,200 packets represent enough data to create 20 images for each run, providing sufficient training data for the NN. Initially five tests were performed along the three routes, described in Table I. Four sticky tabs, represented by the pink and green squares, were placed on the corners of the route as reference points to mark off the test routes. This ensures all subjects walk along the same route and maintain consistent data with only the gait differentiating the subjects.

The runs with added weight and shoes removed were performed to evaluate the effects of different conditions on a person's gait and whether the gait is still unique enough to identify the person. All exported data is processed in Matlab to create spectrogram images. Some examples of spectrograms made with collected data are shown in Fig. 5. Due to the selected 20 MHz bandwidth, the router will utilize the 52 sub-carriers to transmit beacon frame. As a result, the first two columns of 26 pixel width represent the attenuated data transmitted by the router. The bands in between the columns represent the zero-crossing of the carrier frequency. The bands to the right of these columns that mimic bar-code streaks do not represent transmitted data and were removed. The resulting images measure 255x52 in size and were processed by the NN. The spectrograms for each person is split into training data (60%), validation data (20%), and testing data (20%).

TABLE I: Test Conditions

Test #	Route #	Shoes worn	Added Weight
1	1	yes	no
2	2	yes	no
3	3	yes	no
4	2	yes	yes
5	2	no	no



(a) Route 1 (b) Route 2 (c) Route 3

Fig. 4: Walking Routes for Training and Testing the NN

The project originally aimed to train the NN on data produced by instruments, but this would not work because of the type of NN used for processing the data. The bidirectional recurrent NN requires consecutive data such as data which sequentially changes over time. As a result the data collected from subjects walking through the area of interest is required in order to train the NN. This used the recorded and processed data taken from each of the walking route runs and input it into a CNN for training, validation, and testing. Once the NN completed training, it was given new data of one of the subjects walking and left to identify the subject who was walking.

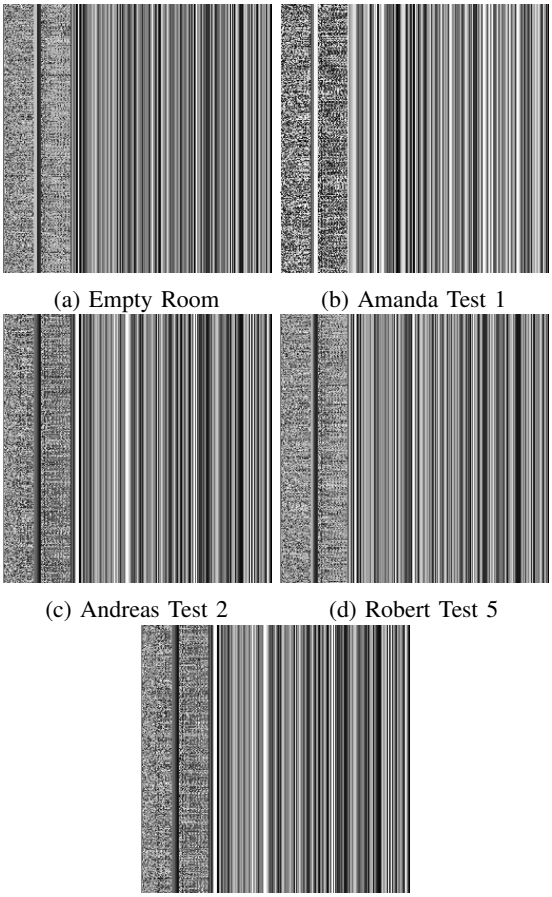
TABLE II: Subject Body Characteristics

Subject Name	Age	Height	Weight	Leg Length	Arm Length
Andreas	21	6'0"	205	38.5"	29"
Lucy	26	5'4"	145	35"	27"
Robert	23	5'11"	180	35.5"	30"
Amanda	23	5'5"	155	32"	26"

IV. Results

After collecting sufficient amounts of data, the NN was trained. The NN was developed using the Keras library to define the CNN and set the number and types of layers. The NN was able to achieve an 86% validation accuracy after training on 60% of the available data. The remaining 40% was used for validation. Fig. 7a shows the accuracy of the NN over training epochs. A Keras API was used to save the best checkpoint based on accuracy. As seen by Fig. 7b, there is still additional work that could be done by optimizing with loss.

The confusion matrix, Table III shows which individuals were misidentified. The NN was always able to correctly identify an empty room from an individual. This is promising for future work on more advanced person detection. Additionally, only Andreas and Lucy were significantly confused as each other. While not physically similar, it was noted that they used a very similar gait during all the tests, which shows that gait recognition may not be uniquely identifiable, but retains a high level of accuracy.



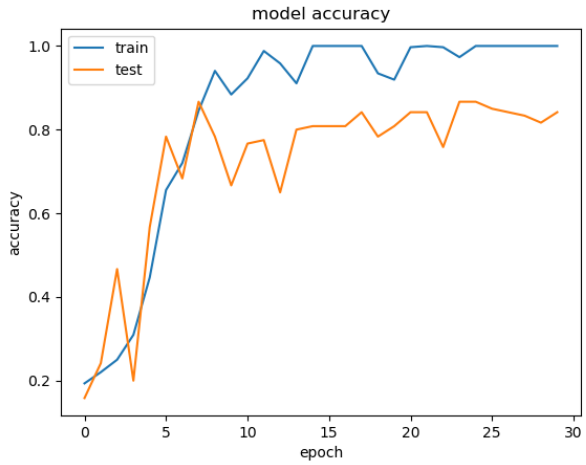
(a) Empty Room (b) Amanda Test 1

(c) Andreas Test 2 (d) Robert Test 5

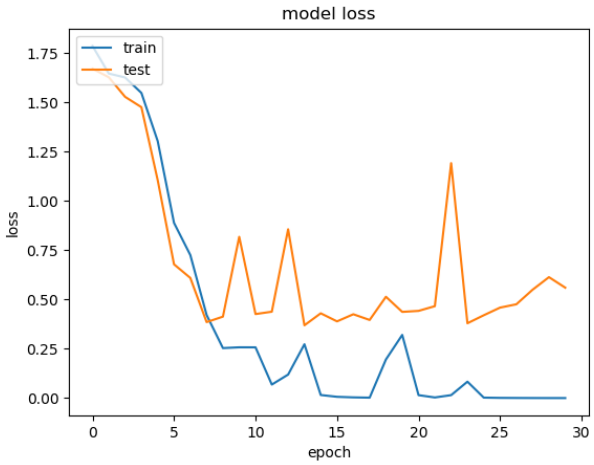
(e) Lucy Test 2

Fig. 5: Original Spectrograms from Data Collected at Robert's Apartment

Fig. 6: Modified Spectrogram Example (Empty Room) from Data Collected at Robert's Apartment



(a) Neural Network Accuracy



(b) Neural Network Loss

TABLE III: Confusion Matrix

Actual\Predicted	Empty Room	Lucy	Andreas	Robert	Amanda
Empty Room	29	0	0	0	0
Lucy	0	21	8	0	0
Andreas	0	11	16	0	0
Robert	0	0	0	21	0
Amanda	0	0	0	1	37

V. Related Work

The authors of Nexmon implemented further firmware modifications in micro-code and template-ram of the Nexus 5 smart phone to extract CSI extraction for practical covert channel implementation. This article's focus was on using CSI for channel security, not for spectrogram generation and object characterization [4]. Furthermore, this work did not utilize power or speed as evaluation parameters of their system.

A Wifi CSI extraction system was implemented utilizing the Wireless NIC 5300. Through community support, this network card in combination with open source software tools allows the extraction of CSI information [1]. The paper then compares the usefulness of the STFT and Discrete Wavelet Transform (DWT). The paper implements the STFT for their Random Forest and Hidden Markov models. For analysis, the group evaluated three different techniques: Random Forest, Hidden Markov, and LSTM RNN. These systems evaluated seven unique actions of the same individual with the LSTM RNN proving the most accurate model. This paper, however, did not analyze these models on mobile environments nor did they evaluate the performance based on power consumption or processing time.

A radar system operating at 7.25 GHz utilized the same principle of Micro-Doppler shifts to characterize the movements of four different animals and seven different actions of the same person [2]. They utilized a STFT to generate spectrograms to feed their Convolutional Neural Network and showed a ninety percent accuracy in classification. However, this system utilized active sensing to generate the data to feed to their neural network whereas Wifi CSI can be extracted from a wireless network passively. Furthermore, the proliferation of 802.11a/n/ac networks allow for the scalability of this system in urban areas.

Another paper used wall mounted access-points to enable device free human tracking via radio tomography using a hybrid neural network [5]. This was a hierarchical neural network hidden Markov model that performed maximum likelihood sequence estimation by combining the probability of standing vs. walking with a probabilistic transition model of a person moving from one pre-set region of the apartment to another. This paper was more concerned with tracking where the subjects were moving in a room then who they were, thus the design of the neural network is completely different than the neural network implemented in this paper.

VI. Ethical Concerns

As with any new technology intended for public use, it can be used for benevolent and malicious acts. Technology which utilizes radio tomography is no exception. These applications consist of identifying people based on their gait when walking and can be extended to identify the gesture or movement performed by a person. Due to the use of ISM band for channel state identification and the wireless nature of radio tomography, the observer does not need to be in the same room, or even building, as the subject. The remoteness allows the observer to invade the subject's privacy without the subject's knowledge.

The level of accuracy achievable by radio tomography technology for identifying a subject's location along with identifying who the subject is presents the possibility of use for law enforcement and security systems, businesses, and criminals. Law enforcement may use the technology to determine if a person of interest is on the premises or identify any potential threats when securing a building, though they must obtain a warrant prior to using this technology. Due to the precedent of similar technology used for the same purpose, use of radio tomography would not be a problem when presented to a judge at the time of requesting a warrant.

Businesses could enter into the gray area of ethics in their use of radio tomographic technology because they could perform

analysis of what part of the home in which residents spend the most time. This information could then be used to determine marketing strategies focused on the local population. This strategy would be an invasion of privacy for homeowners, which creates greater profit for businesses.

Criminals could wrongly use the technology to identify who is in a building and where in the building the people are located in order to determine whether or not the area is occupied. If the criminal has a specific target in mind, the target could be identified and located more easily than without radio tomography. The technology could also be used by burglars to determine whether a residence is clear of occupants when choosing a target to burgle.

The additional work done in this project to perform radio tomography on commodity hardware gives it greater potential for misuse as it is more easily accessible to the general public. The paper also covers improvements to radio tomographic analysis by utilizing machine learning. Machine learning in the form of a trained convolutional neural network has the potential to improve speed and accuracy of the analysis of channel state information. Through these improvements, access to this technology will improve making the situations above occur much more quickly than intended and by a larger base of users who are not technologically capable. If unregulated, home privacy could become an antiquity of this era.

VII. Conclusion

CSI corrects for the attenuation present in the channel due to static and dynamic objects. Static objects can be accounted for by measuring the equilibrium state. The remaining deviation will characterize the dynamic objects within the wireless area. The Nexus 5 Nexmon CSI firmware patch extracted the CSI from the beacon frame of a Sagecom F@st router (trademark name) in the 5 GHz band. The project group performed extensive research into the operation and configuration of the Nexmon utility occurred to specify CSI extraction and discovered that the CSI patch solely operates for the Nexus 5. The CSI magnitude for each sub-carrier over time produced the gray-scale spectrogram images which were processed by the NN. These images needed to specify the useful image data as the image includes the entire 80MHz bandwidth while the router was broadcasting with a 20MHz bandwidth. By utilizing Keras to implement a CNN with a hidden LSTM layer, an accuracy of 86% was achieved. Multiple combinations of layers were tested to optimize the performance of the NN.

VIII. Future Work

One of the original goals of this paper included testing the trained NN on multiple platforms to compare power consumption and compute time. The purpose of this analysis focused on demonstrating the viability of a commercial low-power device to perform real time identification utilizing any wireless system. The testing platforms included the Raspberry Pi 3B, Jetson TX2, Typical Desktop, and High Performance Desktop from Cal Poly San Luis Obispo Bld. 20 Room 127. Furthermore, testing each of these platforms with and without the Intel Movidius Stick to evaluate the improvement in inference computation time during operation (not training).

If the above goal completed successfully on the smaller platforms like the Raspberry 3B or Jetson TX2 and depending on the size of the resulting NN, training in real time would be the next step in commercializing this system. Training represents a much more significant computation requirement and may not be a viable option for the smaller platforms.

The final step would be to implement the CSI extraction patch for the Raspberry Pi 3B or Jetson TX2 to produce a fully enclosed system that could extract the CSI, train the NN, and identify people in real time. The Nexmon group implemented wireless monitoring for multiple chipsets including the Nexus 5 and Raspberry Pi 3B, but only implemented the CSI extraction patch for the Nexus 5. Due to time constraints and extensive research in micro-code of these platforms, CSI implementation was not pursued by the project group.

Acknowledgment

The authors would like to thank Dr. Bruce Debruhl for his assistance with the project and for introducing interesting topics in the field of software defined radio, and Dr. Helen Yu for her expertise in neural networks and computational intelligence.

References

[1] S. Yousefi, H. Narui, S. Dayal, S. Ermon, and S. Valaee, "A survey on behavior recognition using wifi channel state information," *IEEE Communications Magazine*, vol. 55, no. 10, pp. 98–104, Oct. 2017, ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1700082.

[2] Y. Kim and T. Moon, "Human detection and activity classification based on micro-doppler signatures using deep convolutional neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 1, pp. 8–12, Jan. 2016, ISSN: 1545-598X. DOI: 10.1109/LGRS.2015.2491329.

[3] M. Schulz, D. Wegemer, and M. Hollick. (2017). Nexmon: The c-based firmware patching framework, [Online]. Available: https://nexmon.org.

[4] M. Schulz, J. Link, F. Gringoli, and M. Hollick, "Shadow wi-fi: Teaching smart- phones to transmit raw signals and to extract channel state information to implement practical covert channels over wi-fi," *Proceedings of the 16th ACM International Conference on Mobile Systems, Applications, and Services*, Jun. 2018.

[5] A. S. Paul, E. A. Wan, F. Adenwala, E. Schafermeyer, N. Preiser, J. Kaye, and P. G. Jacobs, "Mobilerf: A robust device-free tracking system based on a hybrid neural network hmm classifier," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '14, Seattle, Washington: ACM, 2014, pp. 159–170, ISBN: 978-1-4503-2968-2. DOI: 10.1145/2632048.2632097. [Online]. Available: http://doi.acm.org/10.1145/2632048.2632097.

APPENDIX A
CSI EXTRACION, SPECTROGRAM CONSTRUCTION, AND NN CODE

```
import dpkt
import pandas as pd
import sys
import socket
import struct
import operator
import numpy as np
import cmath

def grab_csi_data(packet):
        eth = dpkt.ethernet.Ethernet(packet)
        ip = eth.data
        udp = ip.data
        return udp.data

def grab_udp_condition(packet):
        capture = False
        eth = dpkt.ethernet.Ethernet(packet)
        ip = eth.data
        if type(ip) == dpkt.ip.IP:
                udp = ip.data
                if type(udp) == dpkt.udp.UDP:
                        capture = True

        return capture

def grab_udp(packets):
        return [(time, grab_csi_data(packet))
                                for (time, packet) in packets if grab_udp_condition(packet)]

def order_packets(packets):
        packets.sort(key=operator.itemgetter(0))

def parse_binary(csi):
        channel = np.empty([255], dtype='complex')
        csi = csi[14:]
        for index in range(0, len(csi), 4):
                real = struct.unpack('>h', csi[index:index + 2])[0]
                imag = struct.unpack('>h', csi[index + 2:index + 4])[0]
                val = complex(real, imag)
                channel[index/4] = abs(val)
        return channel


def print_csv(packets):
        df = pd.DataFrame()
        csi = pd.DataFrame()
        df = df.append([time for time, packet in packets])
        packets = [packet for time, packet in packets]
        for packet in packets:
                channel = parse_binary(packet)
                csi = csi.append(pd.DataFrame(channel).T)

        csi.index = range(len(csi.index))
        csi.columns = [index for index in range(1, len(csi.columns)+1, 1)]
```

```
        df = df.join(csi)
        df.to_csv(sys.argv[1].replace(".pcap", ".csv"))

def main(pcap_file):
        pfile = open(pcap_file)
        pcap = dpkt.pcap.Reader(pfile)
        packets = pcap.readpkts()
        udp_packets = grab_udp(packets)
        order_packets(udp_packets)
        print_csv(udp_packets)

if __name__ == "__main__":
        main(sys.argv[1])

import numpy as np
from keras import backend as K
from keras.models import Sequential
from keras.layers.core import Flatten, Reshape, Permute
from keras.layers import TimeDistributed, Conv2D, Dense, Dropout, Activation, LSTM, MaxPooling2
from keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import classification_report, confusion_matrix
from keras.layers.normalization import BatchNormalization
from keras.callbacks import ModelCheckpoint
from keras.utils.vis_utils import plot_model
import matplotlib.pyplot as plt
from keras.layers.normalization import BatchNormalization

#Start
train_data_path = './data/train'
test_data_path = './data/validation'
img_rows = 255
img_cols = 56
epochs = 30
batch_size = 30
num_of_train_samples = 366
num_of_test_samples = 144
convFilter1 = 128

input_shape = (img_rows, img_cols, 1)

#Image Generator
train_datagen = ImageDataGenerator()

test_datagen = ImageDataGenerator()

train_generator = train_datagen.flow_from_directory(train_data_path,
                                                    color_mode='grayscale',
                                                    target_size=(img_rows, img_cols),
                                                    batch_size=batch_size,
                                                    class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(test_data_path,
                                                        color_mode='grayscale',
                                                        target_size=(img_rows, img_cols),
                                                        batch_size=batch_size,
                                                        class_mode='categorical')

# Build model
```

```
model = Sequential()

model.add(Conv2D(filters=32,
                 kernel_size=(5,5),
                 input_shape=input_shape,
                 padding='valid',
                 activation='tanh',
                 strides=1))

model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(.15))

model.add(Conv2D(filters=convFilter1,
                 kernel_size=(5,5),
                 padding='valid',
                 activation='tanh',
                 strides=1))

model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(.10))

model.add(Reshape((convFilter1,-1)))
model.add(Permute((2,1)))
model.add(Bidirectional(LSTM(128)))
model.add(Dense(5, activation='softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])

filepath="weights.best.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_acc', verbose=1, save_best_only=True, mode=
callbacks_list=[checkpoint]

plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)

#Train
history = model.fit_generator(train_generator,
                    steps_per_epoch=num_of_train_samples // batch_size,
                    epochs=epochs,
                    callbacks=callbacks_list,
                    validation_data=validation_generator,
                    validation_steps=num_of_test_samples // batch_size)

#History for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

#History for accuracy
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```

```
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

-