Indoor Location Sensing Using Smartphone Acoustic System

What kind of deep models could be used for indoor location recognition?
How to deploy and evaluate the model on smartphones and make the inference run in real time?

1. Research Questions

1. Deep models suitable for location recognition
2. System app design deployment and evaluation
3. Interference run in real time

2. Methodology

1. Two architecture designs are presented in fig. 1:
   - client-server one - preprocessing, training and testing is happening on a remote server
   - front-end-only one - preprocessing and testing is happening on a smartphone
2. For both cases front-end responsible for:
   - emitting fixed number of 2 ms 20kHZ chirps every 100ms
   - collecting echo data from the chirps
3. Models:
   - 2 RNNs, 2 DNNs and 2 CNNs models with different hyper-parameters for the server
   - same models but compressed for the front-end
4. Tensorflow[1] APIs - model training, testing and compression
5. Metrics - FLOPS, training parameters and time to train
6. Data samples:
   - 3 different data sets
   - 5 different location points
   - place - 19th floor of EWI - sketch in fig. 2

1. https://www.tensorflow.org/

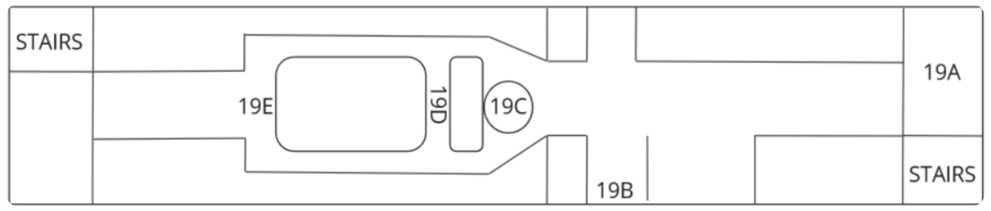

Figure 2: Plan of the 19th floor in EWI

Figure 3: example picture of gray-scale specrotgram

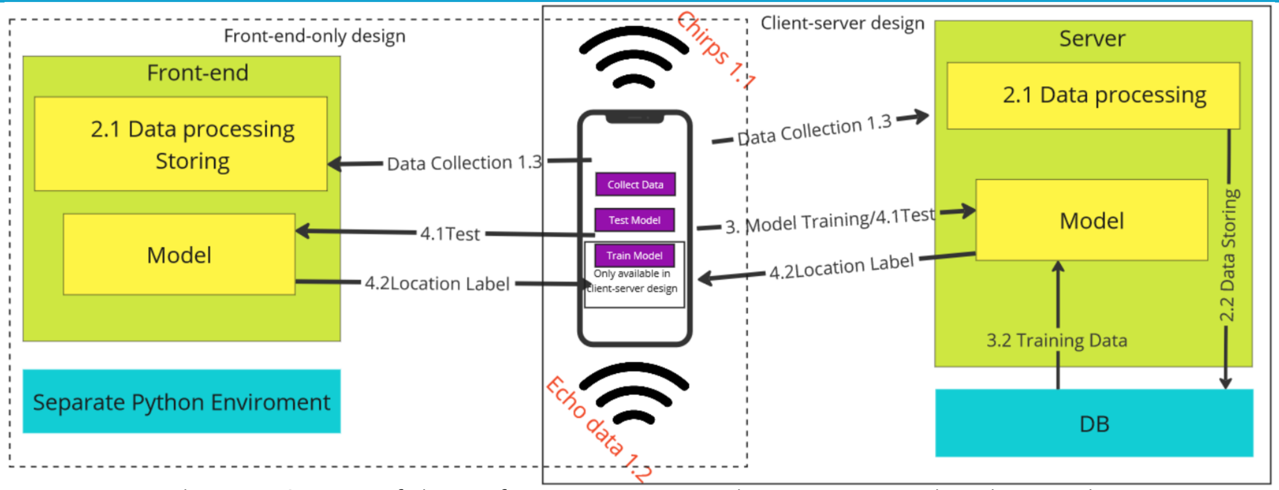

Figure 1: On the left is the front-end-only design and on the right is the client server one - both with arrows for process flow described in section 3.

3. Process flow

1.1 A fixed number of chirps emitted by a smartphone
1.2 The same device is used for collecting the echo reflections from them
1.3 For the front-end only application the data is preprocessed and converted to spectrograms as in fig. 3. Data remains on the smartphone. For the client-server approach it is sent to the server
2.1 On the server the data is preprocessed and converted to spectrograms and stored in a DB
3.1 In the next step is happening the model training on the server or in a separate Python environment and models compression
4.1 When models are created, they are used for interference runs with data collected by the front-end and preprocessed depending where the models are
4.2 The models return localization label of the place

4. Results

	Model	Val Acc	DS1	DS2	DS3	Train time	Params	FLOPS	Pred. time
CNN	256C 128C 1024D	94%	98% 75%	89% 64%	88% 68%	36m	1500K	1186M	60ms 1ms
DNN	256D 512D 256D	94%	98% 23%	89% 22%	89% 22%	2m 20s	290k	19.5M	60ms 1ms
RNN	128LSTM 128D	90%	88% 37%	88% 38%	86% 38%	3m 48s	330k	1.5K	80ms 1ms

Table 1 : Best performing CNN, DNN and RNN models. C - convolutional filters, D - dense units, LSTM - LSTM units, DS - data set, Params - parameters. For the DS1, 2 and 3 columns, the first number is for the server models and the second one for the compressed models

	Model	Val Acc	DS1	DS2	DS3	Train time	Params	FLOPS	Pred. time
CNN	16C 32C 1024D	94%	97% 64%	88% 58%	82% 59%	4m 12s	280k	37M	60ms 1ms
DNN	512D 128D 2048D 512D	94%	98% 23%	89% 21%	86% 22%	9m	1500k	93M	70ms 1ms
RNN	256LSTM 128LSTM 256D	72%	73% 36%	67% 37%	64% 46%	20m 30s	1500k	1.5K	97ms 1ms

Table 2: Best and worst performing CNN, DNN and RNN models, follows the same notation as in table 1

5. Discussion

1. Test Results:
   - server models - best CNN and DNN - confusion matrix in fig. 4 for the CNN
   - compressed models - best CNN
2. Training results:
   - number of parameters correlates with time to train and FLOPS but not accuracy
3. Interference runs time:
   - on server between - 60ms to 100ms
   - on front-end 1-2ms
4. Designs comparison - client-server has better test data results despite taking more time for classification

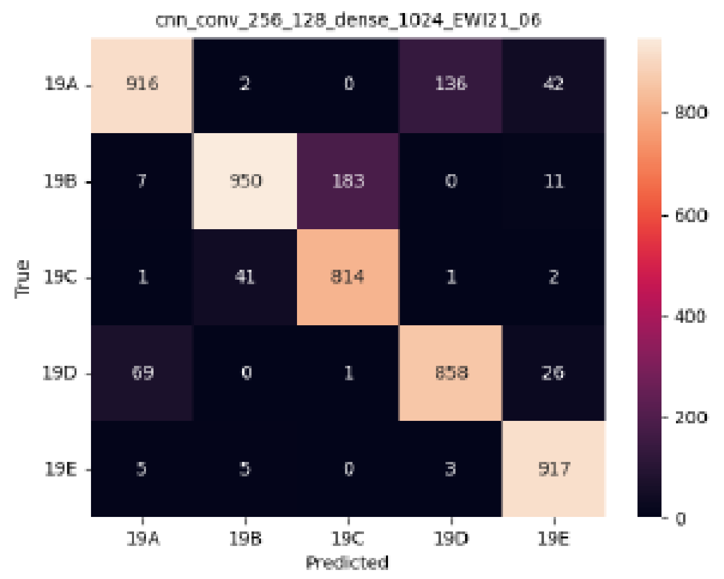

Figure 4: Confusion matrix of the best performing CNN

6. Conclusion and Future work

- DNN and CNN better than RNN
- Server-client architecture has better results
- Test on data collected by different devices
- Impact of second 2ms 20kHZ chirp played at the same time with the initial one

Author
Radoslav Sozonov
radoslav.sozonov@gmail.com

Professor/Supervisor
Qun Song