TinyML-Empowered Indoor Positioning with Light: Model Optimization using Neural Architecture Search

Neel Lodha [1] Ran Zhu [1] Qing Wang [1]

[1]Department of EEMCS, Delft University of Technology

Introduction

Indoor positioning using a Visible Light Positioning System based on the Received Signal Strength (RSS) of light deployed on a Raspberry Pi Pico.

(a) Testbed (b) Illustration

Figure 1. DenseVLC testbed, a) actual testbed, b) testbed illustration.

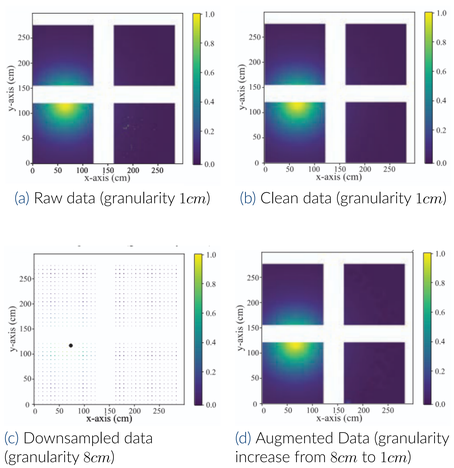
Research Questions

How can we improve the performance (accuracy, inference latency) of a VLP system running TinyML on resource-constrained devices?

- Which traditional neural network architectures (Convolutional Neural Networks, Multilayer Perceptrons) are most suitable for our RSS-based VLP system.
- How can we find architectures using Neural Architecture Search which satisfy the hardware constraints of the Raspberry Pi Pico.
- How does the density of the fingerprint dataset impact the model's performance?

Existing Data Preprocessing

$351,384$ samples of RSS values at different locations collected from the DenseVLC testbed as shown in Figure 1.



(a) Raw data (granularity $1cm$) (b) Clean data (granularity $1cm$)

(c) Downsampled data (granularity $8cm$) (d) Augmented Data (granularity increase from $8cm$ to $1cm$)

Figure 2. Data pre-processing proposed by Zhu et al. [1]

Methodology

Previous Research

Zhu et al. [1] used model architectures like Multi-Layer Perceptron (MLP), Random Forest, Support Vector Machine (SVM), for the VLP system. In our research, we will use their MLP model which 5 hidden layers consisting of 256, 512, 1024, 512 and 256 neurons respectively with ReLU activation function, as the baseline for comparison.

Neural Architecture Search

We explore more model architectures like Convolution Neural Networks (CNN) and use Neural Architecture Search (NAS) to find the best performing model.

With the help of NAS, we try to find different architectures focused on optimizing the following when deployed on a Raspberry Pi Pico which only has $264KB$ of SRAM and $2MB$ of flash memory.

- **Model accuracy**: Positioning error (mean squared error) on the test set.
- **Inference Latency**: How fast is the inference of the model when deployed on the Pico.

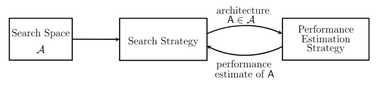


Figure 3. Abstract illustration of Neural Architecture Search methods [2]

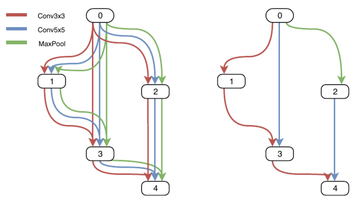


Figure 4. One shot strategy [2]

Method	Trial Type
Random	Multi-trial
GridSearch	Multi-trial
RegularizedEvolution	Multi-trial
DARTS	One-shot
ENAS	One-shot
GumbelDARTS	One-shot
RandomOneShot	One-shot
Proxyless	One-shot

Table 1. Various NAS search strategies [3]

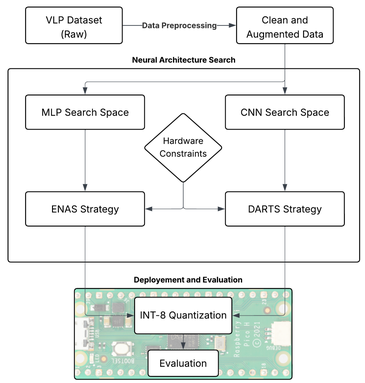


Figure 5. Methodology workflow

Layer Type	Options
FC Layer	32, 64, 128, 256, 512, 1024, 2048
Activation Function	ReLU, LeakyReLU, ELU, Hardswish, Tanh, Sigmoid

Table 2. MLP Search Space

Layer Type	Options
2D Convolution	Channels: 16,32,64,128,256 Kernel Size: 3, 5 Stride: 0, 1 Padding: 0, 1
2D Batch Normalization	Channels: 16,32,64,128,256
2D Pooling	Type: Average, Maximum
Full Connected Layer	32, 64, 128, 256, 512, 1024
Activation Function	ReLU, LeakyReLU, ELU, Hardswish, TanH, Sigmoid

Table 3. CNN Search Space

Results

Model	Model Size (KB)	Model Size PQ (KB)	Position Accuracy (mm)			Position Accuracy PQ (mm)			Inference Latency (ms)
			Raw	Clean	8cm	Raw	Clean	8cm	
Baseline	5168	1303	20.7	14.2	17.6	21.7	16.2	19.0	283
MLP_SRAM	553	**114**	10.0	7.4	15.8	20.1	16.9	18.9	**18**
CNN_Flash	5689	1446	6.4	**4.1**	6.0	19.6	**12.1**	13.6	334

Table 4. Comparison of top models found by NAS. PQ stands for Post Quantization.

MLP_SRAM

Linear(36, 256) → Linear(256, 256) → Hardswish → Linear(256, 256) → Hardswish → Linear(256, 256) → Tanh → Linear(256, 2) → Sigmoid

CNN_Flash:

Conv2d(1, 32, kernel=1, padding=1) → BatchNorm2d(32) → LeakyReLU → DepthwiseSeparableConv2d(32, 64, kernel=3) → BatchNorm2d(64) → LeakyReLU → DepthwiseSeparableConv2d(64, 128, kernel=3) → BatchNorm2d(128) → Tanh → MaxPool2d(2) → Flatten → Linear(512, 1024) → Tanh → Linear(1024, 512) → ReLU → Linear(512, 512) → ReLU → Linear(512, 256) → Hardswish → Linear(256, 2) → Sigmoid

Discussion

- NAS found performant models under the hardware constraints with good accuracy and inference latency.
- Inconsistent results from quantization.
- Room for improvement using more advanced quantization techniques like quantization aware training, pruning and knowledge distillation.
- Data augmentation reduces the labour intensive task of data collection, while maintaining reasonably good position accuracy.
- Some inconsistencies found where augmented dataset would perform worse which might indicate over-fitting or due to the randomness involved in the model's training process.

Conclusion

In our research, we used Neural Architecture Search (NAS) to find efficient MLP and CNN architectures for Visible Light Positioning using Received Signal Strength data. Our models target the Raspberry Pi Pico and improve positioning accuracy by 50% compared to prior work by Zhu et al. [1], while achieving a low inference latency under $100ms$ on the Pico.
We also showed that our models maintain good performance when trained with augmented data, helping reduce the manual effort needed for data collection.

References

[1] R. Zhu, M. Van den Abeele, J. Beysens, J. Yang, and Q. Wang, "Centimeter-level indoor visible light positioning," *IEEE Communications Magazine*, vol. 62, no. 3, pp. 48–53, 2024.

[2] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," 2019. [Online]. Available: https://arxiv.org/abs/1808.05377

[3] Microsoft, "Neural Network Intelligence," 1 2021. [Online]. Available: https://github.com/microsoft/nni