Indoor Millimeter Wave Localization using Multiple Self-Supervised Tiny Neural Networks

Anish Shastri, Andres Garcia-Saavedra, Paolo Casari

***Abstract*—We consider the localization of a mobile millimeter-wave client in a large indoor environment using multilayer perceptron neural networks (NNs). Instead of training and deploying a single deep model, we proceed by choosing among multiple tiny NNs trained in a self-supervised manner. The main challenge is then to determine and switch to the best NN among the available ones, as an incorrect NN will fail to localize the client. In order to upkeep the localization accuracy, we propose two switching schemes: one based on the innovation measured by a Kalman filter, and one based on the statistical distribution of the training data. We analyze the proposed schemes via simulations, showing that our approach outperforms both geometric localization schemes and the use of a single NN.**

I. Introduction

MILLIMETER wave (mmWave) technologies have been instrumental for designing high-accuracy location systems [1]. Although mmWaves propagate quasi-optically, they are short-ranged due to high pathloss and atmospheric attenuation (especially around the 60 GHz band), and are easily blocked by obstacles. Hence, it is common to consider dense deployments of mmWave access points (APs) [2]. In such scenarios, location information becomes a vital tool to optimize the performance of mmWave networks [3].

To localize a mmWave device, existing localization schemes employ the geometric properties of mmWave signals such as the angle of arrival (AoA), angle of departure (AoD), and time of flight (ToF). However, they require knowledge of the indoor area, e.g., the locations of the APs and corresponding anchors, the geometry of the room, the device orientation, etc. [1]. In practice, maintaining and distributing this information is not always feasible. Machine learning techniques have been explored for precise indoor localization [4], [5]. However, they rely on collecting large training datasets, which is often burdensome, and the resulting models are often computationally complex for resource-constrained devices. In our previous work [6], we proposed a shallow NN model to map angle difference-of-arrival (ADoA) measurements to client location coordinates. These NNs relieve the training data collection process by exploiting the location labels obtained from a bootstrapping localization algorithm (the data annotation technique that automatically labels the training data). However, large irregular-shaped indoor environments exhibit more complex relationships between ADoAs and the client locations. This issue typically translates into larger training datasets.In this paper, we advocate that subdividing a room into sections (smaller regular-shaped overlapping or non-overlapping subareas) and using multiple tiny NN models to cover each section is a better strategy than training and covering the whole space with a single model. This makes the approach scalable even in rooms of large size. These NNs, one for each given section of the indoor space, are trained using location labels obtained from a geometric bootstrapping localization algorithm. Training multiple NNs offers two key advantages: (i) the models require less training data to accurately localize a client, resulting in lower computational overhead on the bootstrapping localization algorithm, and expediting the NNs training; (ii) the resulting localization scheme is device-centric (i.e., location estimation is carried out at the client) and can be easily scaled up: once the models are trained at a central server, they can be independently downloaded and run by multiple clients. In this work, we obtain the training labels in a self-supervised manner by resorting to JADE [7] as the bootstrapping algorithm [6]. JADE requires zero knowledge of the indoor environment, and employs ADoAs as input (like our tiny NNs), thus making the localization problem invariant to client orientation. To select the best NN to localize the client, we propose two schemes: one based on the innovation measured by a Kalman filter (KF), by exploiting track information of the client, and another based on out-of-distribution detection (ODD), that exploits the statistical distribution of the training labels.

The key propositions of our work are: *(i)* to train multiple tiny NNs in a self-supervised fashion, by exploiting the training labels obtained from a bootstrapping localization algorithm; *(ii)* two NN switching schemes: a Kalman filter tracking-based and an out-of-distribution detection (ODD)-based scheme, to choose the best NN to infer the client location; *(iii)* performance evaluation via a simulation campaign.

The outline of the paper is as follows: Section II presents a brief review of the existing localization schemes; Section III elaborates on our proposed localization schemes; Section IV presents the results of our simulation campaign; finally, we draw our conclusions in Section V.

II. Related work

Geometry-based schemes exploit angle and range information to localize mmWave devices [1], e.g., ADoAs are used to triangulate a client's location in [8]. Blanco *et al.* exploit AoA and ToF measurements from 60-GHz 802.11ad-based and sub-6 GHz routers to trilaterate the client location [9]. The authors of [10] process the channel impulse response (CIR) measurements from an FPGA-based 802.11ad implementation in order to estimate ToF and AoDs, and compute the client locations. Then a Kalman filter smooths out the trajectory.

Deep learning methods have also been explored to localize a mmWave device. For example, RSSI and SNR beam indices

This work received support from the European Commission's Horizon 2020 Framework Programme under the Marie Skłodowska-Curie Action MINTS (GA no. 861222).

A. Shastri and P. Casari are with the Department of Information Engineering and Computer Science, University of Trento, Italy. E-mails: {anish.shastri, paolo.casari}@unitn.it

A. Garcia-Saavedra is with NEC Laboratories Europe, Heidelberg, Germany. E-mail: andres.garcia.saavedra@neclab.eu

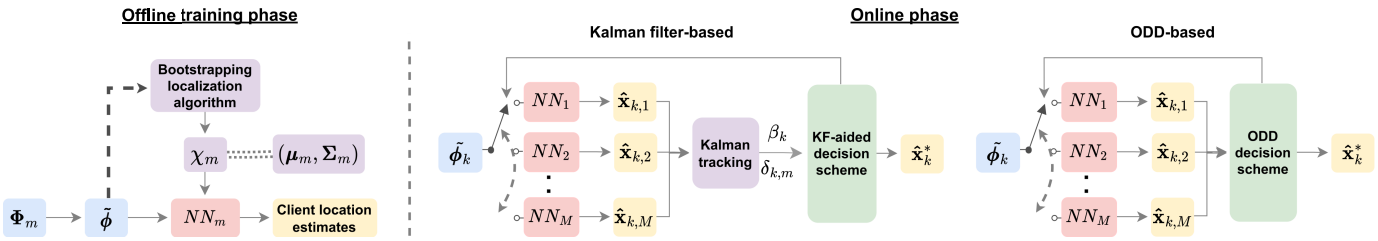

Fig. 1. The workflow of our proposed approaches to select the best NN model for accurate indoor localization.

from 802.11ad-based mmWave routers help learn a multi-task model for location and orientation estimation [4], [11]. The authors of [12] designed a dual-decoder neural dynamic learning framework to sequentially reconstruct the intermittent beam-training measurements, and thus estimate a client's trajectory. signal-to-noise ratio (SNR) fingerprints were used to train a multilayer perceptron regression model and the coarse estimates were filtered using a Kalman filter [13].

III. Proposed Localization Scheme

A. Problem statement and main idea

The main objective of this work is to employ multiple self-supervised tiny NNs to localize a client moving in a large indoor environment in a distributed fashion. A key aspect then is to decide when to switch to the right NN model, given that prior information about the ground truth locations and the map of the indoor environment is not available.

Fig. 1 illustrates the workflow of our proposed localization scheme, which consists of two phases. In the offline training phase, NN m (corresponding to section m of the indoor space) is trained using the ADoA values $\tilde{\boldsymbol{\phi}}$ computed from the AoA measurements $\boldsymbol{\Phi}_m$ (obtained by processing channel state information (CSI) at the client as in [9]). These are fed to the bootstrapping localization algorithm to obtain sets of training labels χ_m. These sets of labels form a distribution with the corresponding mean and covariance matrix paired as $(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$. These labels are used to train NN m. Note that the input to both our NN model and the bootstrapping localization algorithm are the same, i.e, $\tilde{\boldsymbol{\phi}}$.

For the online training phase, we propose two schemes. The first scheme exploits the location estimates from the NN m to track the evolution of the client's state. We exploit the state innovation $\mathbf{y}_t$ and the normalized innovation squared (NIS) metric β of the predicted state, which measures how accurately the Kalman filter predicted the measurements, to choose the best NN. The second scheme exploits the statistical parameters of the training labels to compute the distance between the NN estimates from m different training label distributions. The idea is to compute the distance between the NNs' estimates and the distribution of the labels (in this work, we resort to the Mahalanobis distance). As the wrongly estimated location will be far from the distribution corresponding to the true label, we refer to this scheme as *out-of-distribution detection*.

B. Kalman filter (KF)-based decision scheme

The KF-based scheme involves two stages: the trajectory tracking phase and the decision phase.

Kalman tracking phase. In this phase, we track the evolution of the client trajectory, as estimated by our trained NNs, using a Kalman filter [14]. Let the state of the client at time t be $\mathbf{s}_t = [x_t, \dot{x}_t, y_t, \dot{y}_t]^T$, where x and y are the 2-D coordinates of the client, and $\dot{x}$ and $\dot{y}$ are the x and y component of the velocity. The state $\mathbf{s}$ evolves in time following a constant-velocity (CV) model. The evolution of the state $\mathbf{s}$ at t is given as $\mathbf{s}_t = \mathbf{F}_t\mathbf{s}_{t-1} + \mathbf{w}_t$, where $\mathbf{F}_t = \boldsymbol{I}_2 \otimes \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$ is the 4×4 state transition matrix that transforms the state of the client at time step $t-1$ to t, and $\mathbf{w}_t \sim \mathcal{N}(0, \mathbf{Q}_t)$ represents the zero-mean Gaussian distributed process noise with covariance matrix $\mathbf{Q}_t$. Here, $\Delta t = 1$ s, $\otimes$ is the Kronecker product, and $\boldsymbol{I}_2$ is the 2×2 identity matrix. The predicted state representing the 2D location of the client is $\mathbf{H}\mathbf{s}_t + \mathbf{r}_t$, where $\mathbf{H}$ is the observation matrix given by $\text{diag}(1, 0, 1, 0)$ and $\mathbf{r}_t \sim \mathcal{N}(0, \mathbf{V}_t)$ is the zero mean Gaussian observation noise, with the observation noise covariance matrix $\mathbf{V}_t$.

The Kalman filter performs two steps: *prediction* and *model update*. The *prediction* step estimates the current *a priori* state of the client $\mathbf{s}_{t|t-1}$ based on the previous *a posteriori* estimate $\mathbf{s}_{t-1|t-1}$, i.e., $\mathbf{s}_{t|t-1} = \mathbf{F}_t\mathbf{s}_{t-1|t-1} + \mathbf{w}_t$, and also computes the *a priori* state covariance matrix $\mathbf{P}_{t|t-1} = \mathbf{F}_t\mathbf{P}_{t-1|t-1}\mathbf{F}_t^T$. The *model update* equations correct the existing state predictions and the covariance matrix using the measurement vector $\hat{\mathbf{x}}_t$ and the updated Kalman gain. The prediction error $\hat{\mathbf{y}}_t$ is the innovation of the Kalman filter and is given as

$$\hat{\mathbf{y}}_t = \hat{\mathbf{x}}_t - \mathbf{H}\mathbf{s}_{t|t-1}. \tag{1}$$

This is used to correct the predicted state of the client as $\mathbf{s}_{t|t} = \mathbf{s}_{t|t-1} + \mathbf{K}_t\hat{\mathbf{y}}_t$, where $\mathbf{K}_t$ is the Kalman gain.

We exploit the innovation along with the innovation covariance $\mathbf{G}_t = \mathbf{H}\mathbf{P}_{t|t-1}\mathbf{H}^T + \mathbf{r}_t$, to compute

$$\beta_t = \mathbf{y}_t^T\mathbf{G}_t^{-1}\mathbf{y}_t. \tag{2}$$

We also compute the Euclidean distance δ_t between the predicted state $\mathbf{H}\mathbf{s}_{t|t-1}$ and the current measurement $\hat{\mathbf{x}}_t$, given as $\delta_t = ||\mathbf{H}\mathbf{s}_{t|t-1} - \hat{\mathbf{x}}_t||_2$. We use $\hat{\mathbf{y}}_t$ and β_t to implement the decision scheme to switch among the NNs.

KF-based decision scheme. The measurements used by the Kalman filter are the location estimates obtained from NN m, trained in section m of the indoor space. NN m will be able to estimate the client location accurately as long as the client is moving within section m. Thus, the Kalman filter will be able to predict the location of the client with a low δ_t. At any time step k when the client moves into a different section, the estimates of NN m become more inaccurate. However, the Kalman filter would predict the location estimate based on the

model learned up to time step $k-1$, yielding an expected client location that closely adheres to the past trajectory. Conversely, an inaccurate NN estimate would result in a large δ_k and an even larger β_k. When β_k exceeds a user-defined threshold η, we feed the corresponding ADoAs $\tilde{\boldsymbol{\phi}}_k$ to the M trained NN models to compute the location estimate $\hat{\mathbf{x}}_{k,m}$, where $m = 1, \ldots, M$. We select the NN m^* that minimizes the Euclidean distance between $\hat{\mathbf{x}}_{k,m}$ and the Kalman-predicted location $\mathbf{H}_k \mathbf{s}_{k|k-1}$, i.e., $m^* = \arg\min_m \delta_{k,m}$, where $\delta_{k,m} = ||\mathbf{H}_k \mathbf{s}_{k|k-1} - \hat{\mathbf{x}}_{k,m}||_2$, with the best estimate of the client being $\hat{\mathbf{x}}_k^* = \hat{\mathbf{x}}_{k,m^*}$. The subsequent set of measurements for the Kalman filter would be the locations estimated by NN m^*. We repeat this procedure to switch to the right NN whenever the β metric exceeds a threshold η. In our evaluation, we set $\eta = 2$, as theoretically, the expectation of β should equal the number of degrees of freedom of the Kalman filter (i.e., the 2D coordinates of the client) [15].

C. Out-of-distribution detection (ODD) switching scheme

In this approach, we exploit the statistical properties of the location labels used to train the NNs of each section of the indoor space. Specifically, we compute the mean and the covariance of the training labels distribution χ_m corresponding to section m, characterized by its mean-covariance pair $(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$. In the online testing phase, at any time instance k, if the distance between the current location estimate $\hat{\mathbf{x}}_k$ and the previous estimate $\hat{\mathbf{x}}_{k-1}$ exceeds a user-defined threshold ζ, i.e., $||\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_{k-1}||_2 > \zeta$, where ζ is the distance threshold (1 m in our case), the ADoA values $\tilde{\boldsymbol{\phi}}_k$ at test location k are fed to M NNs. This results in M location estimates $\hat{\mathbf{x}}_{k,m}$. We then compute the Mahalanobis distance $\rho_{k,m}$ of $\hat{\mathbf{x}}_{k,m}$ from each of the distributions χ_m as

$$\rho_{k,m} = \sqrt{(\hat{\mathbf{x}}_{k,m} - \boldsymbol{\mu}_m)^T \boldsymbol{\Sigma}_m^{-1} (\hat{\mathbf{x}}_{k,m} - \boldsymbol{\mu}_m)}\,. \tag{3}$$

The Mahalanobis distance measures the distance of a point from a distribution of data samples, where, the smaller the distance, the closer the point to the distribution. We finally choose the best NN as $m^* = \arg\min_m \rho_{k,m}$, and its corresponding estimate $\hat{\mathbf{x}}_{k,m^*}$ as the best location estimate. Whenever the location estimated by NN m^* is far from the previous estimate, i.e., the Euclidean distance between the current estimate and the previous estimate exceeds the distance threshold ζ (chosen after exhaustive search), we repeat the above procedure and switch to another NN model.

D. Tiny neural network architecture

We resort to a 4-layer tiny NN model with $(N_i,\, N_{h_1},\, N_{h_2},\, N_{h_3},\, 2)$ neurons in each layer. Here, N_i is greater than or equal to the number of ADoAs from visible anchors [6], $N_{h_1} = \lceil \kappa N_i \rceil$, $N_{h_2} = N_{h_1}$, $N_{h_3} = \lceil N_{h_2}/2 \rceil$, and $\lceil \cdot \rceil$ represents the ceiling function. The NN outputs 2D coordinates of the client by learning a non-linear regression function $\mathcal{F}(\tilde{\boldsymbol{\phi}}_t)$ between the ADoAs $\tilde{\boldsymbol{\phi}}_t$ and the client location $\hat{\mathbf{x}}_t$. The regression problem is minimizes the mean-square error (MSE) between the self-supervised training labels and the locations estimates obtained from the NN. The NN employs the rectified linear unit (ReLU) activation function and Adam optimizer. The hyperparameters used for tuning the NN are the factor κ, dropout rate p, learning rate r, and the training batch size b.

IV. Simulation Results

A. Simulation environment

We validate the performance of our proposed schemes via a simulation campaign. We simulate human motion trajectories in an irregular U-shaped room (chosen to mimic realistic indoor scenarios) as shown in Fig. 2, comprising two vertical rectangular sections (S-1 and S-3) of size 5 m $\times$ 18 m and 6 m $\times$ 18 m, connected by a 20 m wide horizontal section (S-2). We deploy seven mmWave APs to cover the indoor space. We collect AoAs from all the APs and their corresponding virtual anchors (VAs) (mirror images of the APs with respect to each wall of the room) at each client location along a trajectory using a ray tracer (considering line-of-sight (LoS) and first-order reflections only, which dominate mmWave propagation [1]). To simulate realistic noisy measurements, we perturb the AoAs with zero-mean Gaussian noise of standard deviation $\sigma = 5^\circ$.

We train each model with $\approx$ 800 locations within each boxed section (S-1, S-2, S-3) in Fig. 2. The trained NNs have the following architecture: NN-1 and NN-2 have $(53, 48, 48, 24, 2)$ neurons in each layer with hyperparameters tuple (κ, p, r, b) as $(0.9, 0.1, 0.003, 50\%)$, and $(0.9, 0.05, 0.004, 50\%)$ respectively, while NN-3 has $(53, 43, 43, 22, 2)$ neurons with $(0.8, 0.1, 0.003, 50\%)$. We test the trained NNs on a trajectory comprising 388 client locations across the three sections of the room (grey line).

B. Analysis of the NN switching schemes

We first analyze the performance of the KF-based switching scheme. Fig. 3 illustrates the metric β computed while using NN-1 to track the client along the test trajectory, and the localization error. Initially, when NN-1 estimates the client locations in S-1, β is close to zero. As the client crosses the border between S-1 and S-2 (around location 140), the errors of NN-1's estimates increase. These estimates significantly deviate from the client's state as predicted by the Kalman filter, resulting in large δ_t and hence even larger β values, leading to the appearance of peaks. Conversely, with every subsequent wrong estimates by NN-1, the Kalman filter keeps predicting the client's location to be around the wrong estimates, resulting again in lower values of β. This trend can be observed when the client moves within S-2 and S-3.

We now analyze the ODD scheme. Fig. 4 illustrates the Mahalanobis distance between the location estimates obtained from different NNs and the distribution of the training data χ_m corresponding to each section of the room, as a function of the client location. The locations estimated by the NNs and lying within the distribution of the training data consistently yield $\rho \leq 2$. We also observe that a large set of locations in the common sections of the room, i.e., C-1 (client locations from 90 to 140) and C-2 (from 235 to 270), were localized accurately by the NNs sharing the overlapped sections, as they compute the ADoAs from the same set of visible anchors. Thus, these estimates have similar ρ values, as they are part of two χs. In such situations, we can exploit the Euclidean

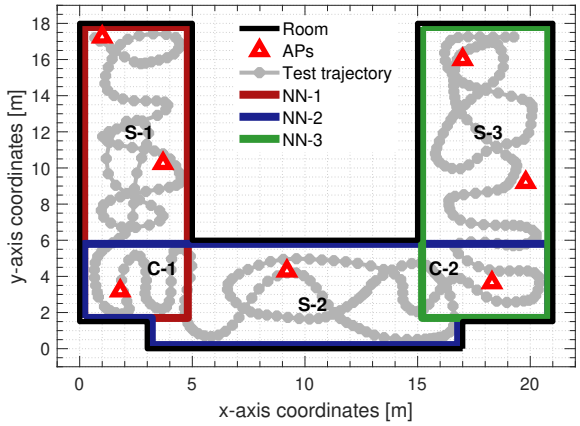

Fig. 2. U-shaped room used for simulations. The colored boxes represent the three sections of the room; the grey line shows the test trajectory.

distance between the current location estimate and the previous location estimate to choose the right NN.

While the values of ρ for NN-1 and NN-3 are low in their respective sections and distinguishable thanks to ADoAs from a disjoint set of APs, the ρ values for location estimates from NN-2 are low for the entire trajectory. This is because a vast majority of the locations in S-2 can compute the ADoA values from multipath components (MPCs) arriving from all the APs. Thus, NN-2 estimates the client locations closer to all the three χs. However, the localization error (yellow dashed line) is still large, especially for the locations in the non-overlapping sections of S-1 and S-3. This implies that NN-2's location estimates, even though highly erroneous, lie within the other two distributions. Thus, we remark that the ODD technique is more appropriate in environments where each section is illuminated by a disjoint set of APs.

The switching points obtained from the KF-based (triangles) and ODD-based (squares) switching schemes are depicted in Fig. 5. The dotted and shaded sections are the borders and the common areas C-1 and C-2, between S-1 and S-2, and S-2 and S-3 respectively. The ideal switching points are within the boundaries of C-1 and C-2 (see Fig. 2), from location index 90 to 140 (C-1) and from 235 to 270 (C-2). We observe that both schemes decide to switch NN models within the ideal switching area. However, unlike the ODD scheme, the KF-based scheme tends to be more robust, as it uses the motion model of the client to predict its future location.

We now compare the performance of our proposed scheme with a single NN (SNN) model having up to as many neurons as the three NNs together. The total number of neurons N_T in the 3 NNs is given by $\sum_{c=1}^{3}\sum_{d=1}^{3} N_{c,h_d}$, where d indexes hidden layers in our tiny NNs. Fig. 6 presents the statistics of localization errors when using our KF-based scheme (NN-KF) and the SNN with varying number of neurons in its hidden layers. We observe that our KF-based switching scheme outperforms all the SNN models. Also, decreasing the size of the SNN (from 100% to 33% of N_T) causes the whiskers to span larger intervals, as an SNN with fewer neurons will underfit the mapping between the ADoAs and the location labels, especially in a complex rooms.

We also performed a preliminary experiment to evaluate

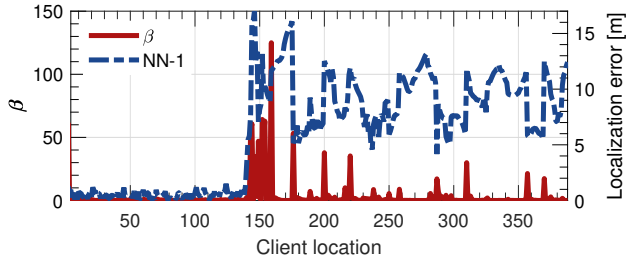

Fig. 3. Values of β and localization error obtained when using NN-1.

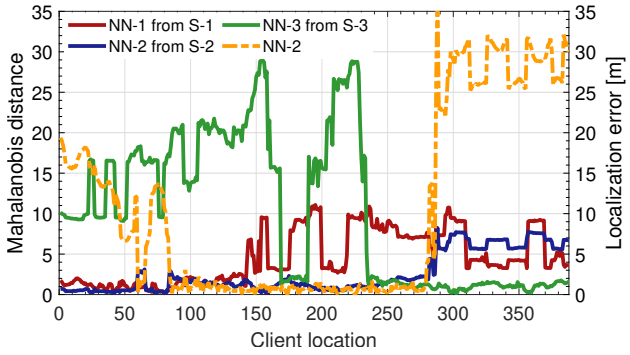

Fig. 4. Mahalanobis distance between the estimated client locations and χ_m for each NN. The dashed line represents the localization error of the test trajectory using NN-2.

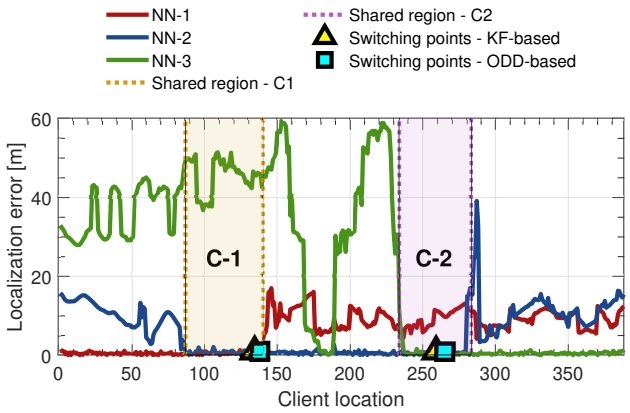

Fig. 5. Localization error using the three NNs and switching points elected by the KF-based (triangles) and ODD-based (squares) switching schemes. The shaded sections represent the overlap sections C-1 and C-2.

training and inference times. For this, we used a Thinkpad E14 laptop, with an AMD Ryzen 7 4000 CPU and 24 GByte of RAM, running in battery-saver mode. Having multiple tiny NNs reduces location estimation from 1.8 ms for the SNN model down to 1.2 ms (or 33% less) for each of the tiny NNs. The model training time also decreases by a factor of 4 (from about 400 s to about 100 s).

Fig. 7 illustrates the localization errors at each location along the test track, where we consider the NN switching points obtained from the KF-based scheme, and concatenate the location estimates of each NN. Here, blue hues represent errors ≤ 1 m whereas green to red hues correspond to higher errors. We observe sub-meter errors at most locations, with slightly larger errors near the bottom left and right corners,

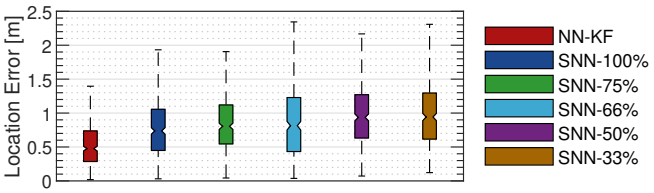

Fig. 6. Statistical dispersion of the localization errors for varying number of neurons (N_T) in the single NN (SNN). SNN-K% denotes the NN with K% of N_T neurons, where $N_T = 384$.

and at the switching points, where the NN being used cannot accurately localize the client. Fig. 8 presents the cumulative distribution function (CDF) of the localization errors of the reconstructed trajectory, and of the trajectory estimated using JADE, a single self-supervised NN, and the geometric ADoA algorithm [8]. We observe that JADE and our self-supervised NNs outperform the ADoA scheme [6]. While JADE achieves sub-meter localization errors in 80% of the cases (mean error: 0.74 m), a single NN achieves sub-meter errors in about 70% of the cases (mean error: 0.82 m). Instead, choosing the right NN via the KF- or ODD-based schemes achieves sub-meter errors in about 90% of the cases. We note that running JADE on the entire trajectory results in large errors, especially when moving closer to the bottom-left and -right corners of the room. This is because noisy ADoA values largely offset the estimates of the associated anchors. Moreover, using a single NN instead of multiple NNs results in large training and bootstrapping algorithm complexity, and a single NN typically needs more parameters to accurately localize a client in complex-shaped environments. Thus, training multiple, smaller NNs and employing our proposed switching schemes can help accurately localize a client in large or irregular rooms.

V. Conclusions

We proposed to employ multiple self-supervised tiny NNs to accurately localize a client as it moves across a large indoor environment. We presented two schemes based on Kalman filters and on the statistical distribution of training labels, in order to dynamically choose the best NN model. Results show that the trajectory reconstructed upon switching to the right NNs via the proposed schemes achieves $\leq$1 m localization errors in 90% of the cases. Upon evaluation, we ascertain that the KF-based scheme is more robust, owing to its ability to track the client's motion and predict the trajectory from NN location estimates. On the other hand, the ODD scheme is a viable choice in large distributed environments, where different sections of the space are illuminated by a disjoint set of APs. Future work will explore schemes that automatically segment the indoor space into sections, as larger environments will warrant a subdivision into larger number of sections.

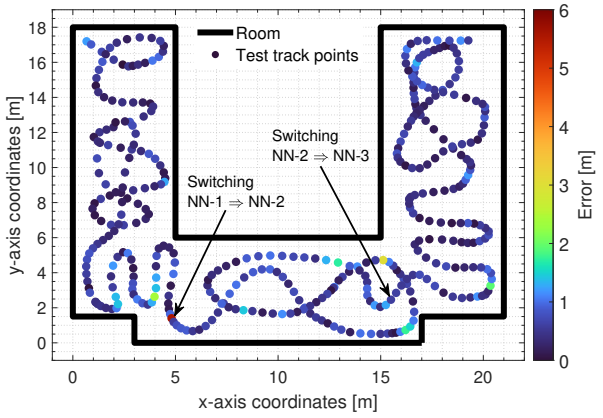

Fig. 7. Location-wise estimation error after reconstructing the trajectory and using the KF-based switching scheme to choose the best NN model out of the three available ones.

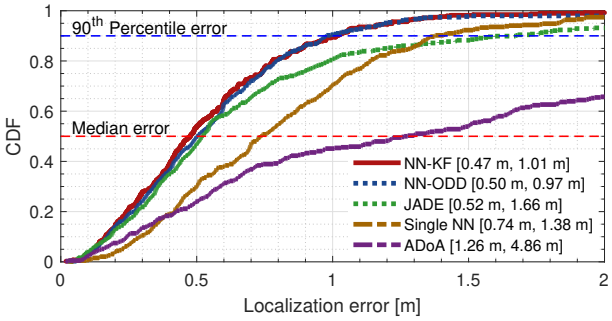

Fig. 8. CDF of the trajectory reconstructed using our proposed switching schemes against the state-of-the-art algorithms. Brackets in the key summarize the median and 90th percentiles, respectively.

References

[1] A. Shastri *et al.*, "A review of millimeter wave device-based localization and device-free sensing technologies and applications," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 3, 2022.
[2] I. A. Hemadeh *et al.*, "Millimeter-wave communications: Physical channel models, design considerations, antenna constructions, and link-budget," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, 2018.
[3] C. Fiandrino *et al.*, "Scaling millimeter-wave networks to dense deployments and dynamic environments," *Proc. IEEE*, vol. 107, no. 4, 2019.
[4] T. Koike-Akino *et al.*, "Fingerprinting-based indoor localization with commercial mmWave WiFi: A deep learning approach," *IEEE Access*, vol. 8, 2020.
[5] S. R. Jondhale *et al.*, "Comparison of neural network training functions for RSSI based indoor localization problem in WSN," *Handbook of Wireless Sens. Netw.*, 2020.
[6] A. Shastri *et al.*, "Millimeter wave localization with imperfect training data using shallow neural networks," in *Proc. IEEE WCNC*, 2022.
[7] J. Palacios *et al.*, "JADE: Zero-knowledge device localization and environment mapping for millimeter wave systems," in *Proc. IEEE INFOCOM*, 2017.
[8] ——, "Single-and multiple-access point indoor localization for millimeter-wave networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 3, 2019.
[9] A. Blanco *et al.*, "Augmenting mmWave localization accuracy through sub-6 GHz on off-the-shelf devices," in *Proc. ACM Mobisys*, 2022.
[10] D. Garcia *et al.*, "POLAR: Passive object localization with IEEE 802.11ad using phased antenna arrays," in *Proc. IEEE INFOCOM*, 2020.
[11] P. Wang *et al.*, "Fingerprinting-based indoor localization with commercial mmWave WiFi - Part II: Spatial beam SNRs," in *Proc. IEEE GLOBECOM*, 2019.
[12] C. J. Vaca-Rubio *et al.*, "mmWave Wi-Fi trajectory estimation with continuous-time neural dynamic learning," in *Proc. IEEE ICASSP*, 2023.
[13] A. Vashist *et al.*, "KF-Loc: A Kalman filter and machine learning integrated localization system using consumer-grade millimeter-wave hardware," *IEEE Consum. Electron. Mag.*, vol. 11, 2022.
[14] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.
[15] I. Reid and H. Term, "Estimation II," *University of Oxford, Lecture Notes*, 2001.