Feature Learning for Neural-Network-Based Positioning with Channel State Information

Emre Gönültaş[1], Sueda Taner[1], Howard Huang[2], and Christoph Studer[3]

[1]*School of Electrical and Computer Engineering, Cornell University, Ithaca, NY*
[2]*Nokia Bell-Labs, Murray Hill, NJ*
[3]*Department of Information Technology and Electrical Engineering at ETH Zürich, Switzerland*

***Abstract*—Recent channel state information (CSI)-based positioning pipelines rely on deep neural networks (DNNs) in order to learn a mapping from estimated CSI to position. Since real-world communication transceivers suffer from hardware impairments, CSI-based positioning systems typically rely on features that are designed by hand. In this paper, we propose a CSI-based positioning pipeline that directly takes raw CSI measurements and learns features using a structured DNN in order to generate probability maps describing the likelihood of the transmitter being at pre-defined grid points. To further improve the positioning accuracy of moving user equipments, we propose to fuse a time-series of learned CSI features or a time-series of probability maps. To demonstrate the efficacy of our methods, we perform experiments with real-world indoor line-of-sight (LoS) and non-LoS channel measurements. We show that CSI feature learning and time-series fusion can reduce the mean distance error by up to 2.5× compared to the state-of-the-art.**

I. Introduction

The need for low-cost but accurate positioning systems is driven by recent trends in virtual reality, asset tracking, robotics, and industrial automation [2], [3]. Existing outdoor positioning solutions mostly rely on global navigation satellite systems (GNSS) that provide meter-level accuracy but require line-of-sight (LoS) satellite connectivity. High-precision indoor positioning solutions typically require specialized hardware that uses visible or infra-red light to localize objects with either active IR transmitting markers [4] or passive reflectors [1]. Such systems require unobstructed views, are affected by sunlight and reflective surfaces, and are costly.

A. CSI-Based Positioning with Neural Networks

Low-cost indoor positioning can be achieved with existing communication infrastructure that utilizes orthogonal frequency division multiplexing (OFDM) [5]. OFDM receivers must acquire channel state information (CSI) to suppress inter-symbol interference caused by multi-path propagation [5]. Since the measured CSI strongly depends on the environment between the transmitting user equipment (UE) and the receiver (access point or base station), it can be used for positioning purposes [6]–[17]. Such CSI-based positioning solutions either use geometrical models [18] or learn a function that maps CSI to position [11], [13], [14], [16], [19]. The latter approach often relies upon deep neural networks (DNNs), which has shown to enable accurate outdoor and indoor positioning accuracy [9], [12], [14]–[17].

DNN-based positioning pipelines that rely on *raw* CSI measurements enable sub-meter-level indoor positioning accuracy [15], but are unable to compete with approaches that use carefully-engineered CSI features [16], [17]. The main reason is as follows: Real-world CSI measurements are affected by small-scale fading, antenna orientation, and hardware impairments [14], [16], such as synchronization errors, residual timing and carrier frequency offsets, and phase noise, which necessitates features that are resilient to such effects. As a consequence, learning DNNs from raw CSI measurements would require a prohibitive amount of training data to learn from a sufficiently diverse training set that contains all possible combinations of real-world system and hardware nonidealities. Hence, existing systems almost exclusively rely on CSI features that are designed by hand and tailored to the communication standard and hardware equipment to be resilient to such impairments.

State-of-the-art CSI feature extraction pipelines often transform the frequency-domain CSI to the delay domain [11], [13], [14], [16]. Computing an autocorrelation instead of using the raw delay-domain information has been shown in [14] to further improve resilience to hardware impairments. Although such features combined with DNNs enable centimeter (cm)-level indoor positioning accuracy [16], they do not exploit the learning capabilities of DNNs. Furthermore, most CSI feature extraction pipelines compute a position estimate based on measurements acquired only during a single time instance, with the exception of the recent papers [15], [20]. However, both of these papers propose to fuse a time series of CSI measurements without providing any insight on the efficacy of fusing the *outputs* of the positioning DNN.

The work of EG and CS was supported in part by the US National Science Foundation (NSF) under grants CNS-1717559, CNS-1955997, and ECCS-1824379. The work of ST and CS was supported in part by ComSenTer, one of six centers in JUMP, a SRC program sponsored by DARPA. The work of CS was also supported by an ETH Research Grant.

Contact author: E. Gönültaş; e-mail: eg566@cornell.edu

We thank P. Huang and B. Rappaport for discussions on DNN-based positioning. We also thank Prof. K. Petersen for allowing us to use the VICON positioning system [1] and the lab space at Cornell University.

B. Contributions

In this paper, we propose a DNN-based positioning pipeline that takes in a time-series of *raw* CSI measurements in order to generate a probability map, which indicates the likelihood of the transmitting user equipment (UE) to be at a certain grid point. This probability map is then used to estimate the UE's position. In order to enable CSI feature learning, we build upon the structure of hand-designed feature extraction pipelines [14], [16] while learning its key parameters. We furthermore improve the positioning accuracy by (i) CSI feature fusion, which combines a time-series of raw-CSI measurements using a recursive neural network (RNN), and (ii) probability map fusion, which combines a time-series of the generated probability maps using an RNN or probability conflation. We systematically study the efficacy of our methods using real-world CSI measurements for LoS and non-LoS scenarios, and we show that our methods can improve positioning accuracy by up to $2.5\times$ compared to the state-of-the-art.

C. Notation

Lowercase and uppercase boldface letters denote column vectors and matrices, respectively. For a matrix $\mathbf{A}$, we denote the transpose by $\mathbf{A}^T$, Hermitian transpose by $\mathbf{A}^H$, entry in the ith row and jth column by $A_{i,j}$, ith column by $\boldsymbol{a}_i$, and real and imaginary parts by $\Re(\mathbf{A})$ and $\Im(\mathbf{A})$, respectively. The column-wise vectorization of $\mathbf{A}$ is denoted by $\mathrm{vec}(\mathbf{A})$. For a vector $\mathbf{a}$, the kth entry is a_k and the ℓ^2-norm is $\|\mathbf{a}\|_2$. The operator $|\cdot|^{\circ 2}$ is the element-wise absolute value squared.

II. System Model

We start by outlining the operation principle of the proposed CSI-based positioning pipeline. We then describe state-of-the-art CSI-based feature extraction and DNN pipelines.

A. System Model

We consider a single-input multiple-output (SIMO) OFDM communication system building on the IEEE 802.11ac standard [21] with U single-antenna mobile UEs, an access point (AP) with M_{R} antennas, and OFDM transmission with W occupied subcarriers. We assume that the uth UE at time instant t is at position $\mathbf{x}^{(u_t)} \in \mathbb{R}^D$, where D is either two or three. The UE transmits a pre-defined pilot sequence, which is used by the AP to estimate the CSI on the occupied OFDM subcarriers. Since the system is assumed to operate in time-division duplexing mode, the other UEs are inactive. The estimated M_{R}-dimensional SIMO channel vector associated with UE u at time t and subcarrier $w = 1, \ldots, W$ is denoted by $\mathbf{h}_w^{(u_t)} \in \mathbb{C}^{M_{\mathrm{R}}}$. We call the collection $\mathbf{H}^{(u_t)} = [\mathbf{h}_1^{(u_t)}, \ldots, \mathbf{h}_W^{(u_t)}]$ of these channel vectors for all subcarriers the *estimated CSI* of UE u at time t. In what follows, we assume that the estimated CSI is affected by real-world system and hardware impairments, such as noise, synchronization errors, mismatches in carrier frequency and sampling rates, and phase noise.

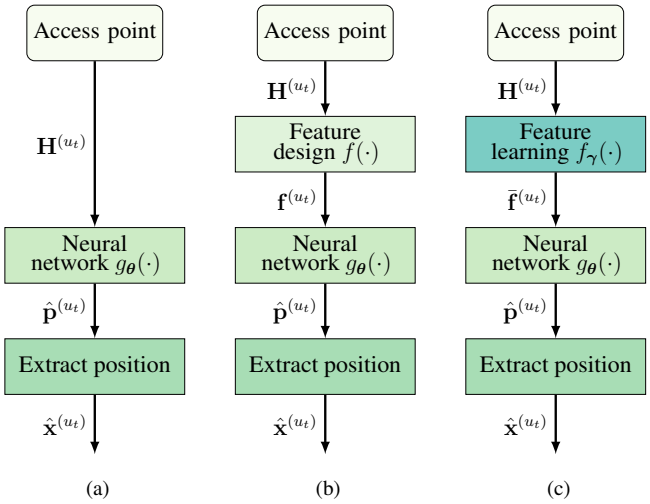

Fig. 1. Overview of three different CSI-based positioning pipelines: (a) Estimated CSI is directly fed into a DNN to produce a probability map; (b) estimated CSI is first transformed into CSI features using a pre-designed feature extraction function $f(\cdot)$; (c) proposed architecture that uses a learned feature extraction function $f_{\boldsymbol{\gamma}}(\cdot)$, which is implemented as a structured DNN.

B. CSI-Based Positioning Using DNNs

Multiple architectures are possible to perform CSI-based positioning with DNNs. Fig. 1 illustrates three different alternatives, which we detail next.

The first architecture is shown in Fig. 1a. The estimated CSI $\mathbf{H}^{(u_t)}$ of UE u at time t is fed directly to a DNN that computes a probability map $\hat{\mathbf{p}}^{(u_t)} \in \mathbb{R}^K$ according to $\hat{\mathbf{p}}^{(u_t)} = g_{\boldsymbol{\theta}}(\mathbf{H}^{(u_t)})$. Here, K is the number of grid points and $g_{\boldsymbol{\theta}}$ is the function that maps estimated CSI to a probability map [16]. Each grid point $\mathbf{g}_k \in \mathbb{R}^D$, $k = 1, \ldots, K$, corresponds to a fixed location in the area of interest, and each entry $\hat{p}_k^{(u_t)} \in [0, 1]$, $k = 1, \ldots, K$, of the estimated probability map $\hat{\mathbf{p}}^{(u_t)}$ indicates the likelihood of the UE u being at the associated grid point at time t. By computing $\hat{\mathbf{x}}^{(u_t)} = \sum_{k=1}^K \mathbf{g}_k \hat{p}_k^{(u_t)}$, one then generates an estimate of the UE's position.[1] We emphasize that directly feeding CSI estimates into the DNN does, in general, not perform well (see our results in Section V-B), mainly because it would require a prohibitive amount of training data to capture a representative set of CSI measurements for all possible system and hardware impairment realizations.

The second architecture is shown in Fig. 1b and overcomes the main drawbacks of the first architecture. The difference to Fig. 1a is that the CSI estimate $\mathbf{H}^{(u_t)} \in \mathbb{C}^{M_{\mathrm{R}} \times W}$ is first transformed into a CSI feature vector $\mathbf{f}^{(u_t)} \in \mathbb{R}^S$ containing S features and is computed according to $\mathbf{f}^{(u_t)} = f(\mathbf{H}^{(u_t)})$, where the function f represents the feature extraction stage. The feature extraction function f is typically designed by hand (hence dubbed "feature design" in Fig. 1b) and tailored to the specifics of the communication system and designed to cope with small-scale fading, antenna orientation, and hardware impairments. Feature extraction also serves the purpose of preparing the data for subsequent DNN processing [14]. While

[1] Alternative approaches that directly generate a position estimate within the DNN have been shown in [16] to result in inferior accuracy.

such architectures are widely used in state-of-the-art CSI-based positioning pipelines [8], [13], [14], [16], [17], [22]–[25], such hard-coded CSI feature extraction stages are unable to exploit the learning capabilities of DNNs.

The third architecture is shown in Fig. 1c. The key difference to the previous architectures is the fact that we directly *learn* a suitable CSI feature extraction stage. Specifically, we learn a function $f_{\boldsymbol{\gamma}}$ (hence dubbed "feature learning" in Fig. 1c) with parameters $\boldsymbol{\gamma}$ that maps CSI estimates to CSI features according to $\bar{\mathbf{f}}^{(u_t)} = f_{\boldsymbol{\gamma}}(\mathbf{H}^{(u_t)})$. By using a *structured* DNN to implement the function $f_{\boldsymbol{\gamma}}$, this architecture is able to fully benefit from the power of DNNs, which results in improved position accuracy (see our results in Section V-B).

C. Existing CSI Feature Extraction Pipelines

As explained above, CSI-based positioning pipelines often rely on hand-crafted CSI features that are extracted from estimated CSI (see Fig. 1b). The CSI features must not only be resilient to small-scale fading, antenna orientation, and hardware impairments, but they should also be different for different transmit positions. We now summarize the key steps of the CSI feature extraction pipeline from [16]. The main insights will then be used for CSI feature learning in Section III.

1) Delay-Domain Transform: Since the propagation channel is described in a compact manner in the time domain, it is often beneficial to convert the frequency-domain CSI into the delay domain according to [11], [13], [14], [16]

$$\hat{\mathbf{H}}^{(u_t)} = \mathbf{H}^{(u_t)}\mathbf{F}_W^H. \tag{1}$$

Here, $\mathbf{F}_W$ is the W-dimensional unitary discrete Fourier transform (DFT) matrix[2] and $\hat{\mathbf{H}}^{(u_t)} \in \mathbb{C}^{M_\mathrm{R}\times W}$ is the delay-domain matrix, which contains the C taps of the wireless channel between UE u and one AP antenna in each row.

2) Autocorrelation: Computing the autocorrelation [14] or "instantaneous" autocorrelation [16] has shown to improve resilience of the CSI features to synchronization errors and residual phase errors (caused, e.g., by carrier frequency offset or phase noise). Following [16], let $\hat{H}_{m,k}^{(u_t)}$ be the kth delay-domain sample measured at the mth receive antenna of UE u at time t. Then, we calculate the 2-dimensional convolution over the delay and antenna domains as

$$R^{(u_t)}[\kappa,\tau] = \sum_{m=1}^{M_\mathrm{R}}\sum_{k=1}^{W} \hat{H}_{m,k}^{(u_t)}\hat{H}_{m+\kappa-M_\mathrm{R},k+\tau-W}^{(u_t)^*}, \tag{2}$$

where $\kappa = 1,2,\ldots,2M_\mathrm{R}$ and $\tau = 1,2,\ldots,2W$, and the resulting matrix is denoted by $\mathbf{R}^{(u_t)}$.

3) Real-Valued Decomposition and CSI Feature Normalization: Since some deep learning frameworks are unable to deal with complex numbers, we first vectorize the autocorrelation matrix in (2) via $\mathbf{r}^{(u_t)} = \mathrm{vec}(\mathbf{R}^{(u_t)})$, and then convert the $2M_\mathrm{R}2W$-dimensional vector into the real domain as follows:

$$\hat{\mathbf{f}}^{(u_t)} = \left[\Re\{\mathbf{r}^{(u_t)}\}^T, \Im\{\mathbf{r}^{(u_t)}\}^T\right]^T. \tag{3}$$

[2]Performing an inverse DFT over only the used subcarriers works well in practice and requires lower complexity than first extrapolating the channel coefficients to all subcarriers, e.g., using the method from [26].

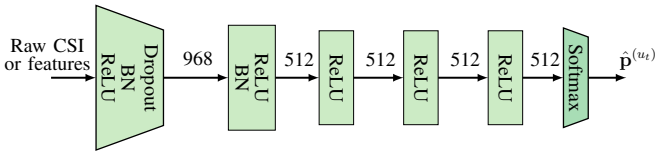

Fig. 2. DNN topology used to generate probability maps from raw estimated CSI $\mathbf{H}^{(u_t)}$, designed CSI features $\mathbf{f}^{(u_t)}$, or learned CSI features $\bar{\mathbf{f}}^{(u_t)}$.

In practice, the UE's transmit power and the gain of the low-noise amplifier at the AP are typically set independently. In addition, the path loss can vary vastly for different propagation scenarios. Thus, the received power is, in general, not a reliable quantity for CSI-based indoor positioning applications. To this end, we normalize the CSI features in (3) according to $\mathbf{f}^{(u_t)} = \hat{\mathbf{f}}^{(u_t)}/\|\hat{\mathbf{f}}^{(u_t)}\|_2$ prior to feeding them into the DNN [13], [16], [24], [27]–[29]. This normalization step also improves convergence of stochastic gradient descent.

D. DNN Structure and Training

For all of the positioning pipelines shown in Fig. 1, we use the same DNN structure from [16], as illustrated in Fig. 2. For the architecture depicted in Fig. 1a, the positioning DNN takes in raw CSI measurements. For the architecture depicted in Fig. 1b, the positioning DNN takes in hand-designed features. For the architecture depicted in Fig. 1c, the positioning DNN additionally includes trainable input layers that implement the feature learning function f_γ which processes raw CSI measurements; the rest of the positioning DNN is the same as for the two other architectures. The positioning DNN $g_{\boldsymbol{\theta}}$, where the vector $\boldsymbol{\theta}$ contains all weights and biases, consists of six hidden layers and the numbers in Fig. 2 indicate the number of activations per layer. All layers use rectified linear unit (ReLU) activation functions, except the last layer is using a softmax activation to generate the estimated probability map $\hat{\mathbf{p}}^{(u_t)}$ for the uth UE at time t. The first layer uses batch normalization (BN) and dropout; the second layer only uses BN.

We assume that all of the DNNs are trained off-line from a dataset containing CSI-location pairs. While self-supervised methods, e.g., channel charting [22], could avoid the acquisition of large data sets, we leave an investigation of such methods for future work. During training, we initialize the weights using Glorot [30] and use the binary cross entropy (BCE) as the loss. To generate the training set, we compute a reference probability map $\mathbf{p}^{(u_t)}$ using the approach described in [16] for each reference position $\mathbf{x}^{(u_t)}$. Concretely, we identify the nearest four grid points to the reference position and assign the associated four probability values so that the expected position corresponds to the reference position.

III. CSI Feature Learning

We now propose our approach to learn the CSI feature extraction stage as shown in Fig. 1c instead of working with hard-coded CSI features. We note that learning an unstructured DNN directly from the raw CSI features would suffer from the same limitations as the architecture in Fig. 1a. Thus, we use a simple but effective alternative: We first build a structured DNN

that models the key steps of the feature extraction pipeline detailed in Section II-C. We then learn the key parameters of this structured DNN together with the DNN that generates the probability maps. These steps are detailed next.

A. Delay-Domain Transform

We apply the transform in (1) with a trainable weight matrix $\mathbf{W}_1 \in \mathbb{C}^{W\times W}$ instead of the inverse DFT (IDFT) matrix $\mathbf{F}_W^H$. To this end, we initialize the trainable weight matrix with $\mathbf{F}_W^H$. We then compute $\hat{\mathbf{H}}^{(u_t)} = \mathbf{H}^{(u_t)}\mathbf{W}_1$ using the real-valued decomposition. We note that after learning the weight matrix $\mathbf{W}_1$, the transformed matrix $\hat{\mathbf{H}}^{(u_t)}$ contains CSI features that are not necessarily in the delay domain anymore.

B. DFT-Based Autocorrelation

In order to calculate the instantaneous 2D autocorrelation in (2), we use the Wiener–Khinchin theorem. Specifically, we calculate the autocorrelation matrix with the aid of DFT and IDFT matrices as follows:

$$\hat{\mathbf{R}}^{(u_t)} = \mathbf{F}_{2M_\mathrm{R}}^H |\mathbf{F}_{2M_\mathrm{R}} \mathbf{H}_z^{(u_t)} \mathbf{F}_{2W}|^{\circ 2}\, \mathbf{F}_{2W}^H. \tag{4}$$

Here, $\hat{\mathbf{R}}^{(u_t)} \in \mathbb{C}^{2M_\mathrm{R}\times 2W}$ is the autocorrelation of the zero-padded delay-domain CSI matrix $\mathbf{H}_z^{(u_t)} \in \mathbb{C}^{2M_\mathrm{R}\times 2W}$, $\mathbf{F}_{2M_\mathrm{R}}$ and $\mathbf{F}_{2W}$ are the $2M_\mathrm{R}$-dimensional and $2W$-dimensional DFT matrices respectively. Zero padding is achieved by appending zeros to the columns and rows of $\hat{\mathbf{H}}^{(u_t)}$ to get $\mathbf{H}_z^{(u_t)}$, which doubles the number of rows and columns.

Our main idea for this stage is to calculate (4), where we replace the DFT matrix $\mathbf{F}_{2M_\mathrm{R}}$ by a trainable weight matrix $\mathbf{W}_2 \in \mathbb{C}^{2M_\mathrm{R}\times 2M_\mathrm{R}}$, which is initialized by $\mathbf{F}_{2M_\mathrm{R}}$, and $\mathbf{F}_{2M_\mathrm{R}}^H$ by $\mathbf{W}_2^H$. Analogously, we replace the DFT matrix $\mathbf{F}_{2W}$ by a trainable weight matrix $\mathbf{W}_3 \in \mathbb{C}^{2W\times 2W}$, which is initialized by $\mathbf{F}_{2W}$, and $\mathbf{F}_{2W}^H$ by $\mathbf{W}_3^H$. We then compute

$$\hat{\mathbf{R}}^{(u_t)} = \mathbf{W}_2^H |\mathbf{W}_2 \mathbf{H}_z^{(u_t)} \mathbf{W}_3|^{\circ 2}\, \mathbf{W}_3^H. \tag{5}$$

We note that after learning the two weight matrices $\mathbf{W}_2$ and $\mathbf{W}_3$, the matrix $\hat{\mathbf{R}}^{(u_t)}$ in (5) does not necessarily correspond to the autocorrelation anymore. Once again, we carry out the above steps using the real-valued decomposition.

C. CSI Feature Normalization

Finally, we vectorize the matrix $\hat{\mathbf{R}}^{(u_t)}$ from (5) via $\hat{\mathbf{r}}^{(u_t)} = \mathrm{vec}(\hat{\mathbf{R}}^{(u_t)})$ and separate the real and imaginary parts as in (3), resulting in the vector $\hat{\mathbf{f}}^{(u_t)}$. As a last step, we normalize the result as $\bar{\mathbf{f}}^{(u_t)} = \hat{\mathbf{f}}^{(u_t)}/\|\hat{\mathbf{f}}^{(u_t)}\|_2$, which is the CSI feature vector that is fed in the positioning DNN (cf. Fig. 1c).

IV. Time-Fusion of CSI-Features and Probability Maps

We now propose three methods that take into account the facts that (i) CSI estimates are generated at fast rates and (ii) multiple consecutive CSI features and/or probability maps can be fused to improve positioning accuracy of moving UEs.

A. Fusion of CSI Features

In practical situations, CSI estimates are generated as a time series and consecutive CSI features will exhibit some degree of similarity due to the relatively slow motion of the UEs with respect to the rate of CSI acquisition. Our idea is illustrated in Fig. 3a and fuses $\delta+1$ consecutive CSI estimates $\{\mathbf{H}^{(u_n)}\}_{n=t-\delta}^{t}$ using a recurrent neural network (RNN). For each time step, we first apply our trainable CSI feature extraction layer to each CSI estimate, followed by a CSI feature fusion layer that is implemented using $\delta+1$ gated-recurrent-units (GRUs) [31] with linear activation functions. The GRUs produce a single CSI feature vector $\underline{\mathbf{f}}^{(u_t)}$ that is passed to the neural network $g_{\boldsymbol{\theta}}$ to produce a probability map.

B. Probability Map Fusion

An alternative to fusing time series of CSI features is to fuse $\tau+1$ consecutive probability maps $\{\hat{\mathbf{p}}^{u_n}\}_{n=t-\tau}^{t}$. The basic idea is illustrated in Fig. 3b and we propose two different fusion methods. The first fusion approach uses a simple four hidden-layer RNN with an initial layer consisting of $\tau+1$ GRUs [31] with K linear activation functions, which takes in the time series of probability maps and computes a fused probability map $\underline{\mathbf{p}}^{u_t}$. The hidden layers have K ReLU activations, and the output layer has a softmax activation with K outputs. The second fusion approach uses Gaussian conflation [32], which has been used in [16] to fuse probability maps from multiple APs and multiple transmit antennas to improve positioning accuracy. Here, we first compute the location estimates $\{\hat{\mathbf{x}}^{(u_n)}\}_{n=t-\tau}^{t}$ and associated covariance matrices $\mathbf{K}^{(u_n)} = \sum_{k=1}^{K} p_k^{(u_n)} (\mathbf{g}_k - \hat{\mathbf{x}}^{(u_n)})(\mathbf{g}_k - \hat{\mathbf{x}}^{(u_n)})^T$ for $n = t-\tau, \ldots, t$ from the $\tau+1$ probability maps $\{\hat{\mathbf{p}}^{u_n}\}_{n=t-\tau}^{t}$. We then perform Gaussian conflation [16]

$$\hat{x}_d^{(u_t)} = \frac{\sum_{n=t-\tau}^{t} {K_{d,d}^{(u_n)}}^{-1} \hat{x}_d^{(u_n)}}{\sum_{n=t-\tau}^{t} {K_{d,d}^{(u_n)}}^{-1}}, \tag{6}$$

where $d = 1, \ldots, D$ with $D = 2$. The intuition behind (6) is that location estimates with large variance (which are less reliable) are deweighted in the averaging process.

C. CSI Feature and Probability Map Fusion

In order to further improve the positioning accuracy, we can combine CSI feature fusion with probability map fusion. The idea is illustrated in Fig. 3c and takes in $\tau+1$ separate groups of raw CSI estimates. Each group of $\delta+1$ consecutive raw CSI estimates are first passed through our learned CSI feature extraction pipeline and feature fusion RNN. The features are then passed through DNNs that produce $\tau+1$ probability maps, which are fused using an RNN or Gaussian conflation in order to produce a fused probability map. Finally, this probability map is used to compute the position estimate.

V. Results

We now provide experimental results for the CSI feature learning and time fusion methods proposed in Section III and Section IV, respectively. We start by describing the measurement setup and then show our results.

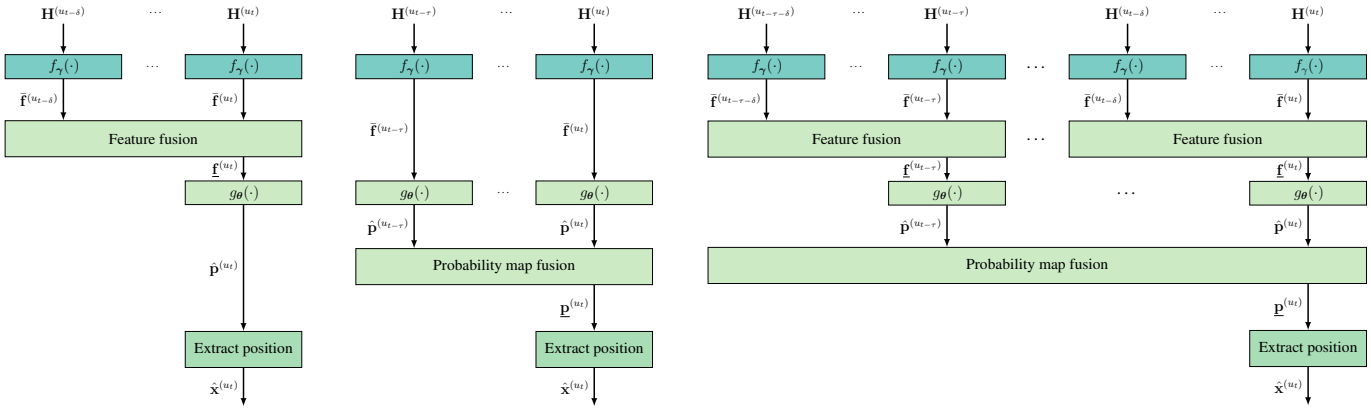

Fig. 3. CSI feature fusion and probability map fusion: (a) CSI feature fusion in which an RNN combines a time-series of CSI estimates to produce a CSI feature; (b) probability map fusion in which an RNN or Gaussian conflation combines a time-series of probability maps; (c) CSI feature fusion and probability map fusion that combines both methods to compute a combined probability map.

A. Measurement Setup

For our experiments, we use two datasets from [16] containing real-world CSI measurements recorded in a LoS lab scenario (called *LoS Lab v1* with 154k training samples and 3800 test samples) and a non-LoS home scenario (called *NLoS Home* with 96k training samples and 2200 test samples). In addition, we also acquired a new dataset (called *LoS Lab Data v2* with 202k training samples and 50k test samples), which has been recorded in a lab space at Cornell University using the same measurement setup described in [16]. The dataset is generated with a robot that follows a random path in a $3.2\times4.2\,\mathrm{m}^2$ area under LoS conditions, where we first record the training set and then a test set that have different UE locations. This new dataset is more challenging than the two other datasets from [16] as the test set contains many locations that were not previously fingerprinted in the training set. We obtain CSI measurements for $W = 234$ subcarriers and at $M_{\mathrm{R}} = 4$ receive antennas, and we use a VICON [1] precision positioning system to collect the ground-truth location information. To construct the probability maps, we use a 22×22 uniformly-spaced grid which corresponds to $K = 484$ grid-point probabilities. In what follows, we provide the mean distance error (MDE) in meters evaluated on the test sets.

B. Experimental Results

We now show our experimental results for the proposed CSI feature learning and time fusion methods.

1) Impact of CSI Feature Learning: Table I shows MDE results for the following methods: (i) Fig. 1a, in which the DNN $g_{\boldsymbol{\theta}}$ takes in *raw* CSI measurements and generates probability maps as described in Section III; (ii) Fig. 1b, in which the DNN $g_{\boldsymbol{\theta}}$ uses *designed* CSI features as described in Section II-C; and (iii) Fig. 1c, in which the neural network $g_{\boldsymbol{\theta}}$ takes in the output of the proposed CSI *feature learning* approach as described in Section III. For the *LoS Lab v1* and *NLoS Home* scenarios, the MDE for the designed CSI feature pipeline matches those reported in [16]. As mentioned earlier, the use of raw CSI estimates does not perform well. The proposed CSI feature learning approach, however, consistently outperforms the two other methods in terms of the MDE.

TABLE I
IMPACT OF CSI FEATURE LEARNING ON MDE.

Features	**Raw CSI**	**Designed**	**Learned**
Figure	1a	1b	1c
LoS Lab v1	0.36	0.05	**0.04**
LoS Lab v2	0.50	0.30	**0.25**
NLoS Home	0.52	0.28	**0.19**

TABLE II
IMPACT OF CSI FEATURE TIME-FUSION ON MDE.

Features	**Non-fusion, learned**	**Designed**	**Learned**
Figure	1c	3a and 1b	3a and 1c
LoS Lab v1	0.04	0.04	**0.03**
LoS Lab v2	0.25	**0.24**	**0.24**
NLoS Home	0.19	0.21	**0.17**

TABLE III
IMPACT OF PROBABILITY MAP FUSION ON MDE.

	Non-fusion	**RNN**		**Gaussian conflation**	
Features	Learned	Designed	Learned	Designed	Learned
Figure	1	3b and 1b	3b and 1c	3b and 1b	3b and 1c
LoS Lab v1	0.04	**0.03**	**0.03**	**0.03**	**0.03**
LoS Lab v2	0.25	0.23	**0.19**	0.23	0.20
NLoS Home	0.19	0.22	**0.16**	0.23	0.17

TABLE IV
IMPACT OF CSI FEATURE AND PROBABILITY MAP FUSION ON MDE.

	Non-fusion	**RNN**		**Gaussian conflation**	
Features	Learned	Designed	Learned	Designed	Learned
Figure	1c	3c and 1b	3c and 1c	3c and 1b	3c and 1c
LoS Lab v1	0.04	0.04	**0.02**	0.04	**0.02**
LoS Lab v2	0.25	0.22	**0.21**	**0.21**	**0.21**
NLoS Home	0.19	0.20	**0.15**	0.19	**0.15**

2) Impact of CSI Feature Time-Fusion: Table II shows MDE results for CSI feature time fusion as detailed in Section IV-A. Here, the RNN in Fig. 3a takes in $\delta + 1 = 3$ subsequent CSI features to generate a combined CSI feature that is used to generate probability maps (we have observed that $\delta + 1 = 3$ resulted in the lowest MDE). Compared to the non-fusion case, we observe an additional MDE decrease for all datasets.

3) Impact of Probability Map Time-Fusion: Table III shows MDE results for probability map time fusion as detailed in

Section IV-B. Here, an RNN or Gaussian conflation is used to combine $\tau + 1 = 3$ subsequent probability maps to generate a combined probability map (we have observed that $\tau + 1 = 3$ resulted in the lowest MDE). Compared to the non-fusion case, we observe an additional MDE decrease for all datasets, where the MDE is slightly lower for the RNN-based approach (compared to Gaussian conflation) for the *LoS Lab v2* and *NLoS Home* datasets.

4) Impact of CSI Feature and Probability Map Time-Fusion: Table IV shows MDE results for the combination of CSI feature and probability map fusion as detailed in Section IV-C. Here, we take groups of $\tau + 1 = 3$ probability maps, each of which are generated by fusing groups of $\delta + 1 = 3$ CSI features, and outputs a combined probability map as shown in Fig. 3c. Compared to CSI feature learning alone (no time fusion) and hand-crafted CSI features, we observe a $2\times$ and a $2.5\times$ MDE reduction, respectively. Fusing both the CSI features and the probability maps results in the lowest MDE.

VI. Conclusions

We have proposed a DNN-based positioning pipeline that takes in raw CSI estimates and learns CSI features using a structured positioning DNN. We have also proposed two time-fusion methods, which combine a time series of CSI features and probability maps in order to further improve positioning accuracy. Our experiments with real-world indoor CSI measurements show a reduction in mean distance error of up to $2.5\times$ compared to state-of-the-art methods that build on hand-crafted CSI features. Our next step is to apply our methods to self-supervised channel charting [22] and semi-supervised positioning methods [13], [24] in order to improve their localization capabilities.

References

[1] VICON. Vero family motion capture system. [Online]. Available: https://www.vicon.com/hardware/cameras/vero/

[2] S. Han, Z. Gong, W. Meng, C. Li, and X. Gu, "Future alternative positioning, navigation, and timing techniques: A survey," *IEEE Wireless Commun.*, vol. 23, no. 6, pp. 154–160, Oct. 2016.

[3] N. Fallah, I. Apostolopoulos, K. Bekris, and E. Folmer, "Indoor human navigation systems: A survey," *Interacting with Computers*, vol. 25, no. 1, pp. 21–33, Jan. 2013.

[4] WorldViz. Precision position tracking (PPT). [Online]. Available: https://www.worldviz.com/virtual-reality-motion-tracking

[5] R. W. Chang, "Synthesis of band-limited orthogonal signals for multi-channel data transmission," *The Bell System Technical Journal*, vol. 45, no. 10, pp. 1775–1796, Dec. 1966.

[6] K. Wu, J. Xiao, Y. Yi, D. Chen, X. Luo, and L. M. Ni, "CSI-based indoor localization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1300–1309, Jul. 2013.

[7] Z. Yang, Z. Zhou, and Y. Liu, "From RSSI to CSI: Indoor localization via channel response," *ACM Comput. Surveys*, vol. 46, no. 2, pp. 1–32, Nov. 2013.

[8] V. Savic and E. G. Larsson, "Fingerprinting-based positioning in distributed massive MIMO systems," in *Proc. IEEE 82nd Vehic. Tech. Conf.*, Sep. 2015, pp. 1–5.

[9] X. Wang, L. Gao, S. Mao, and S. Pandey, "DeepFi: Deep learning for indoor fingerprinting using channel state information," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Mar. 2015, pp. 1666–1671.

[10] X. Wang, L. Gao, S. Mao, and S. Pandey, "CSI-based fingerprinting for indoor localization: A deep learning approach," *IEEE Trans. Veh. Technol.*, vol. 66, no. 1, pp. 763–776, Jan. 2017.

[11] J. Vieira, E. Leitinger, M. Sarajlic, X. Li, and F. Tufvesson, "Deep convolutional neural networks for massive MIMO fingerprint-based positioning," in *Proc. IEEE Intl. Symp. Personal, Indoor, Mobile Radio Commun.*, Oct. 2017, pp. 1–6.

[12] M. Arnold, J. Hoydis, and S. ten Brink, "Novel massive MIMO channel sounding data applied to deep learning-based indoor positioning," in *Intl. ITG Conf. Systems, Commun., Coding*, Feb. 2019, pp. 1–6.

[13] E. Lei, O. Castañeda, O. Tirkkonen, T. Goldstein, and C. Studer, "Siamese neural networks for wireless positioning and channel charting," in *Proc. 57th Ann. Allerton Conf. Commun., Control, and Comput.*, Sep. 2019, pp. 200–207.

[14] P. Ferrand, A. Decurninge, and M. Guillaud, "DNN-based localization from channel estimates: Feature design and experimental results," *ArXiv preprint: 2004.00363*, Apr. 2020.

[15] P. Li, H. Cui, A. Khan, U. Raza, R. Piechocki, A. Doufexi, and T. Farnham, "Wireless localisation in WiFi using novel deep architectures," *ArXiv preprint: 2010.08658*, Oct. 2020.

[16] E. Gönültaş, E. Lei, J. Langerman, H. Huang, and C. Studer, "CSI-based multi-antenna and multi-point indoor positioning using probability fusion," *ArXiv preprint: 2009.02798*, Sep. 2020.

[17] A. Foliadis, M. H. C. Garcia, R. A. Stirling-Gallacher, and R. S. Thomä, "CSI-based localization with CNNs exploiting phase information," *ArXiv preprint: 2101.08983*, Jan. 2021.

[18] F. Wen, H. Wymeersch, B. Peng, W. P. Tay, H. C. So, and D. Yang, "A survey on 5G massive MIMO localization," *Digital Signal Process.*, vol. 94, pp. 21–28, Nov. 2019.

[19] H. Chen, Y. Zhang, W. Li, X. Tao, and P. Zhang, "ConFi: Convolutional neural networks based indoor Wi-Fi localization using channel state information," *IEEE Access*, vol. 5, pp. 18 066–18 074, Sep. 2017.

[20] M. T. Hoang, B. Yuen, K. Ren, X. Dong, T. Lu, R. Westendorp, and K. Reddy, "A CNN-LSTM quantifier for single access point CSI indoor localization," *ArXiv preprint: 2005.06394*, May 2020.

[21] IEEE 802.11ac-2013, "Draft standard for information technology — telecommunications and information exchange between systems — local and metropolitan area networks — specific requirements — part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," IEEE, Tech. Rep., 2013.

[22] C. Studer, S. Medjkouh, E. Gönültaş, T. Goldstein, and O. Tirkkonen, "Channel charting: Locating users within the radio environment using channel state information," *IEEE Access*, vol. 6, pp. 47 682–47 698, Aug. 2018.

[23] S. Medjkouh, E. Gönültaş, T. Goldstein, O. Tirkkonen, and C. Studer, "Unsupervised charting of wireless channels," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–7.

[24] P. Huang, O. Castañeda, E. Gönültaş, S. Medjkouh, O. Tirkkonen, T. Goldstein, and C. Studer, "Improving channel charting with representation-constrained autoencoders," in *Proc. IEEE Int. Workshop Signal Process. Advances Wireless Commun. (SPAWC)*, Aug. 2019, pp. 1–5.

[25] P. Ferrand, A. Decurninge, L. G. Ordoñez, and M. Guillaud, "Triplet-based wireless channel charting," *ArXiv preprint: 2005.12242*, May 2020.

[26] S. Haene, A. Burg, P. Luethi, N. Felber, and W. Fichtner, "FFT processor for OFDM channel estimation," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2007, pp. 1417–1420.

[27] J. Deng, S. Medjkouh, N. Malm, O. Tirkkonen, and C. Studer, "Multipoint channel charting for wireless networks," in *Proc. IEEE Conf. Rec. Asilomar Conf. Signals, Sys., and Comp.*, Feb. 2018, pp. 286–290.

[28] C. Geng, H. Huang, and J. Langerman, "Multipoint channel charting with multiple-input multiple-output convolutional autoencoder," in *Proc. IEEE/ION Position, Location Navigation Symp. (PLANS)*, Apr. 2020, pp. 1022–1028.

[29] P. Agostini, Z. Utkovski, and S. Stańczak, "Channel charting: An Euclidean distance matrix completion perspective," in *Proc. IEEE Intl. Conf. Acoustics, Speech and Signal Proces.*, May 2020, pp. 5010–5014.

[30] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Thirteenth Int. Conf. Artificial Intelligence Stat.*, vol. 9, May 2010, pp. 249–256.

[31] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *ArXiv preprint: 1412.3555*, Dec. 2014.

[32] T. P. Hill, "Conflations of probability distributions," *Trans. Amer Math. Soc.*, vol. 363, no. 6, pp. 3351–3372, Jun. 2011.