Online DNN for Massive MIMO Channel Estimation

Vincent LAU

Chair Professor

Department of Electronic and Computer Engineering

e Hong Kong University of Science and Technology

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

Outline

01 Massive MIMO Channel Estimation

02 Online DNN for Point-to-Point Massive MIMO Channel Estimation

03 Extension for MU Massive MIMO Channel Estimation and Limited Feedback

04 Conclusions

Massive MIMO Channel Estimation

Massive MIMO Signal Model

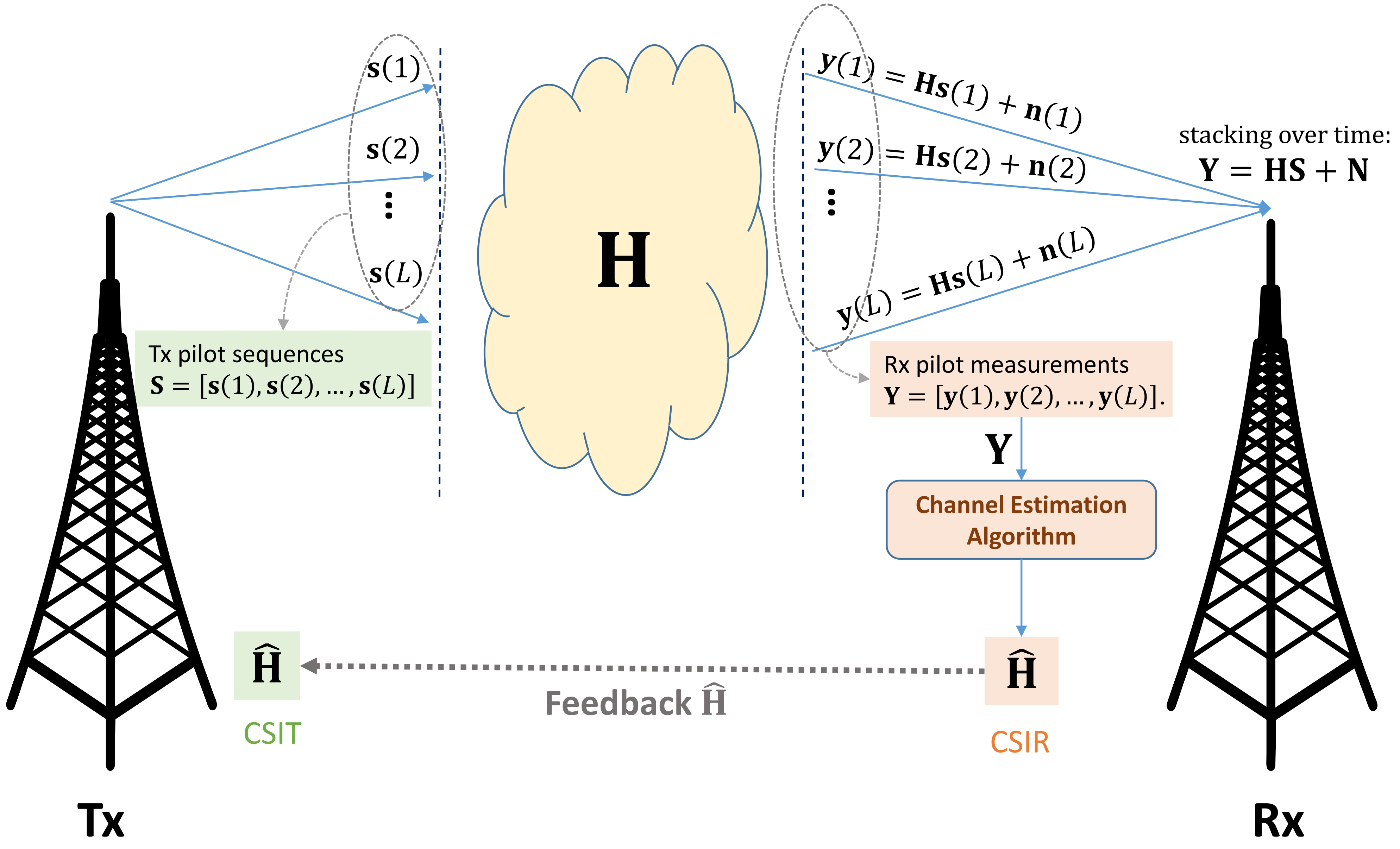


Fig 1: Point-to-Point MIMO with Explicit CSIT Feedback.

- Consider a massive MIMO system:
  - Rx has N antennas
  - Tx has M antennas
- For CE, the Tx transmits sequences of known pilot symbols $\mathbf{S} \in \mathbb{C}^{M \times L}$ of length L to the Rx, the received signal in matrix form is

  Pilot Observations Pilot Symbols

$$\mathbf{Y} = \mathbf{HS} + \mathbf{N},$$

- It is important to estimate the CSI $\mathbf{H} \in \mathbb{C}^{N \times M}$ to leverage the benefits of massive MIMO, and various MIMO techniques rely on accurate CE:
  - Design precoding & decoding matrices to exploit spatial multiplexing gain
  - Design equalization for data detection
  - Power & interference management
  - …
- CSIT will also be needed for enhanced performance (e.g., for precoding), in which case the CSI needs to be fed back to the Tx.

Traditional Channel Estimation

- If no prior information on the channel
  - **Least square (LS)** formulation

$$\hat{\mathbf{H}}_{\mathrm{LS}} = \arg\min_{\mathbf{H}} \|\mathbf{Y} - \mathbf{HS}\|_F^2$$

  - LS has closed-form expression

$$\hat{\mathbf{H}}_{\mathrm{LS}} = \mathbf{YS}^H(\mathbf{SS}^H)^{-1}$$

  - **Problem**: 1) pilot number L needs to be larger than channel dimension M, induces large pilot overhead for massive MIMO 2) high complexity due to matrix inversion
- If given statistics (covariance) of the channel, i.e., let $\mathbf{y} = \mathrm{vec}(\mathbf{y})$, $\mathbf{h} = \mathrm{vec}(\mathbf{h})$, and we know $\mathbf{R} = \mathbb{E}[\mathbf{hh}^H]$:
  - **Linear minimize mean square error (LMMSE)** can be formulated as

$$\hat{\mathbf{h}}_{\mathrm{MMSE}} = \arg\min_{\mathbf{W},\mathbf{b}} \mathbb{E}_{p(\mathbf{h}|\mathbf{y})}\left\{\left(\hat{\mathbf{h}} - \mathbf{h}\right)^H\left(\hat{\mathbf{h}} - \mathbf{h}\right)\right\}$$

$$\hat{\mathbf{h}} = \mathbf{Wy} + \mathbf{b}$$

  - LMMSE has closed-form expression [1] provided $\mathbf{S}$ satisfies $\mathbf{SS}^{\boldsymbol{H}} = \rho L\mathbf{I}_M$:

$$\hat{\mathbf{h}}_{\mathrm{MMSE}} = (\mathbf{R}^{-1}\sigma_n^2 + \rho M\mathbf{I}_{NM})^{-1}(\mathbf{S}^T \otimes \mathbf{I}_N)\mathbf{y}.$$

  - **Problems**: 1) hard to obtain accurate covariance and 2) high complexity due to large matrix inversion

To reduce pilot overheads, we must exploit the intrinsic structures of H

[1] A. Assalini, E. Dall'Anese and S. Pupolin, "Linear MMSE MIMO Channel Estimation with Imperfect Channel Covariance Information," 2009 IEEE International Conference on Communications, 2009.

Compressive Sensing-Based CE

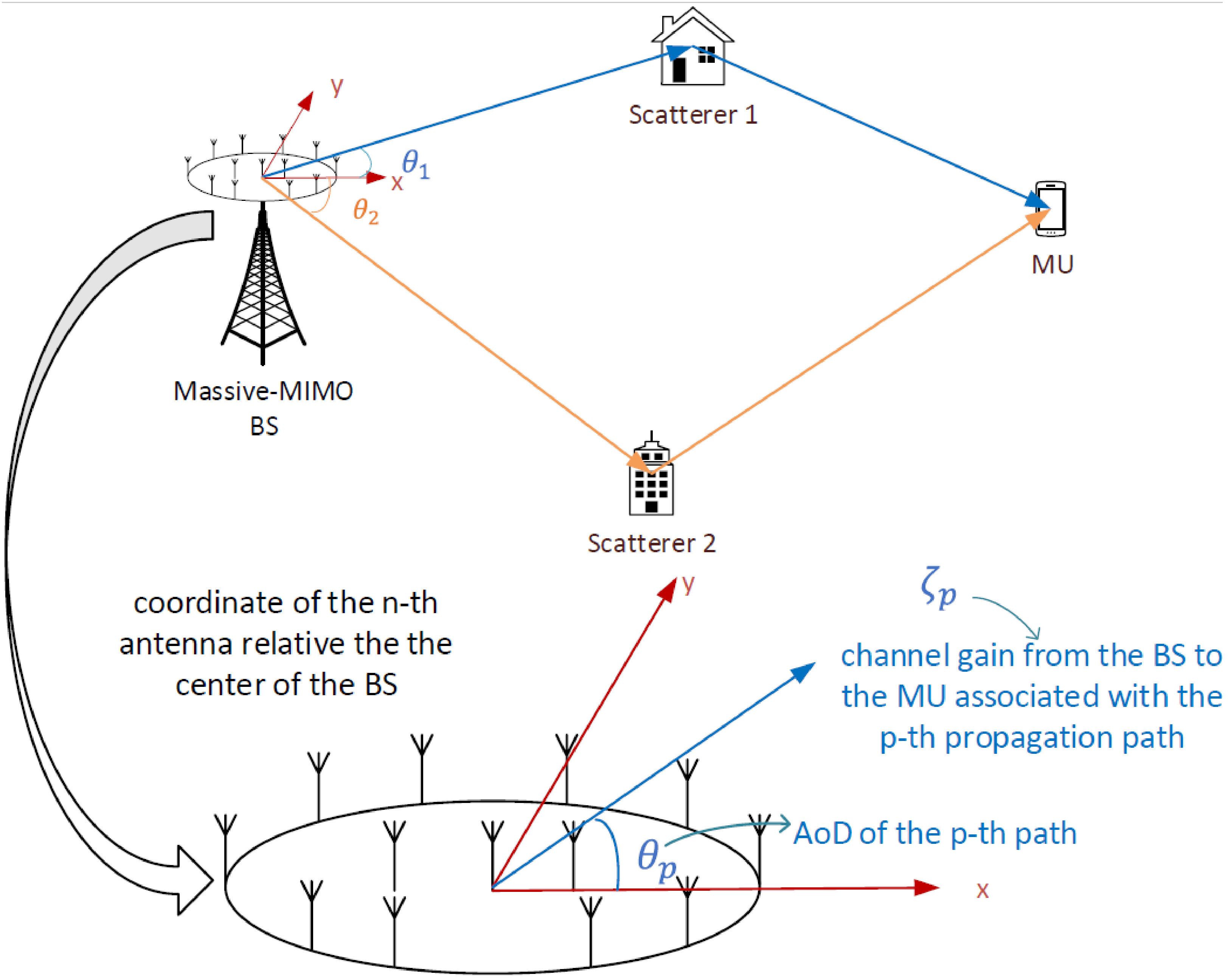


Fig. 2: Channel sparsity induced by limited scattering in the propagation environment.

- The channel is sparse under certain basis due to limited scattering in the propagation environment.
- By exploiting hidden sparsity structures in the MIMO channel, we can estimate $\mathbf{H}$ with reduced pilot overhead ($L < M$).
- For example, consider the channel vector $\mathbf{h} \in \mathbb{C}^N$ between one Tx antenna and the Rx antennas, then $\mathbf{h}$ has a sparse representation in the angular domain

$$\mathbf{h} = \mathbf{F}\mathbf{x}$$

  - $\mathbf{F} \in \mathbb{C}^{N \times N}$ is the steering matrix (determined by array geometry)
  - $\mathbf{x}$ is the sparse angular domain channel
- Different sparsity structures can be exploited to reduce pilot overhead

Different Sparsity Structures

Random Sparsity

- The channel is sparse due to limited propagation paths between Tx and Rx
- We just know $\mathbf{x}$ is sparse and the support is random without special structure

Clustered Sparsity

- The channel supports are clustered in subsets of overlapping candidate supports
- Induced by angular domain spreading of propagation paths

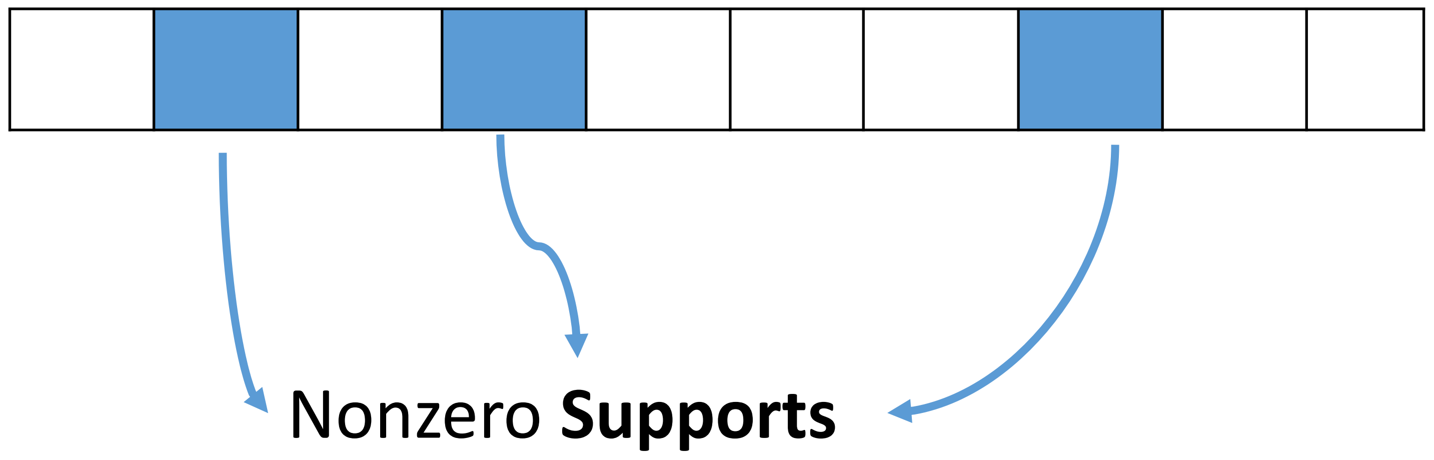


Fig 3: Random sparse channel $\mathbf{x}$.

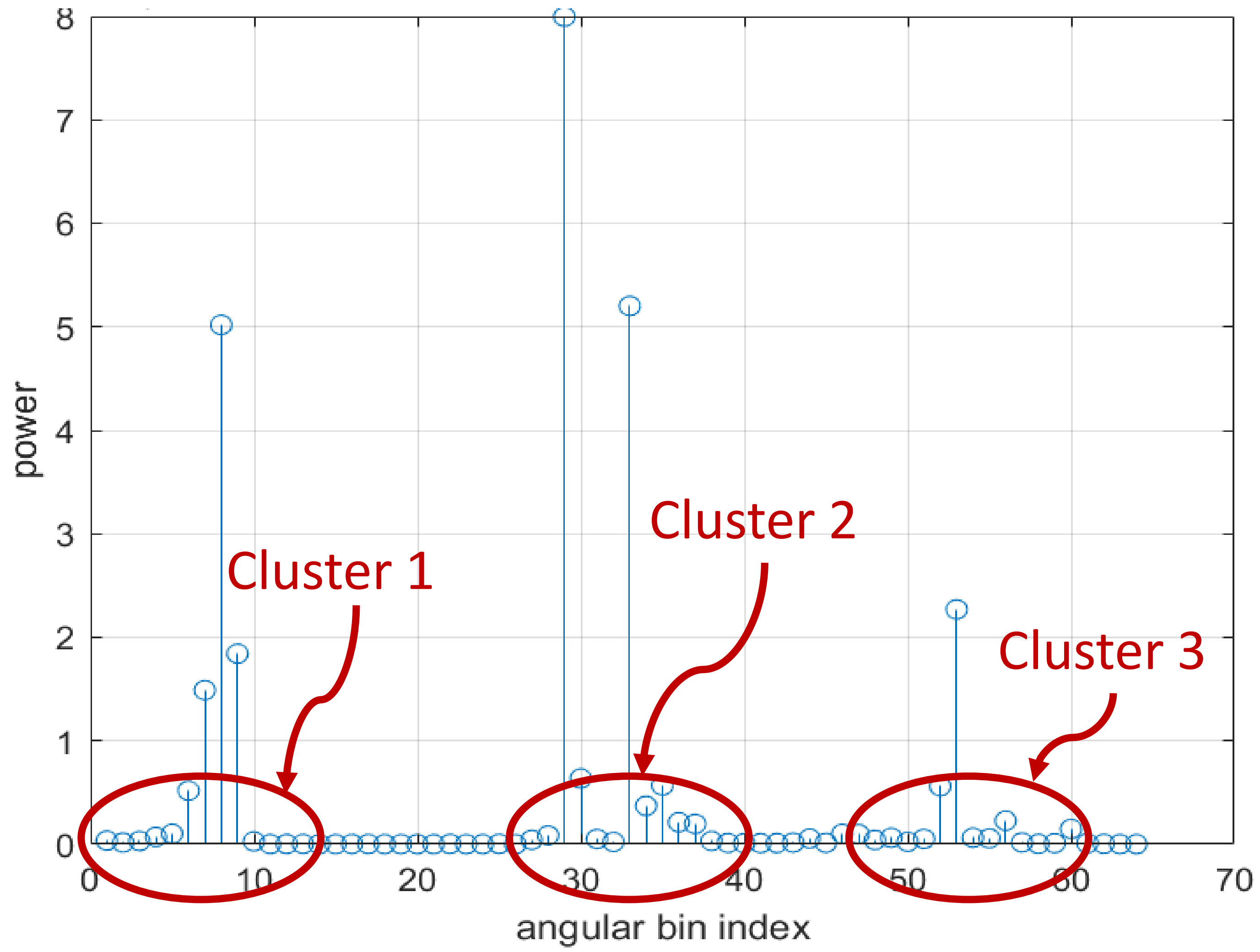
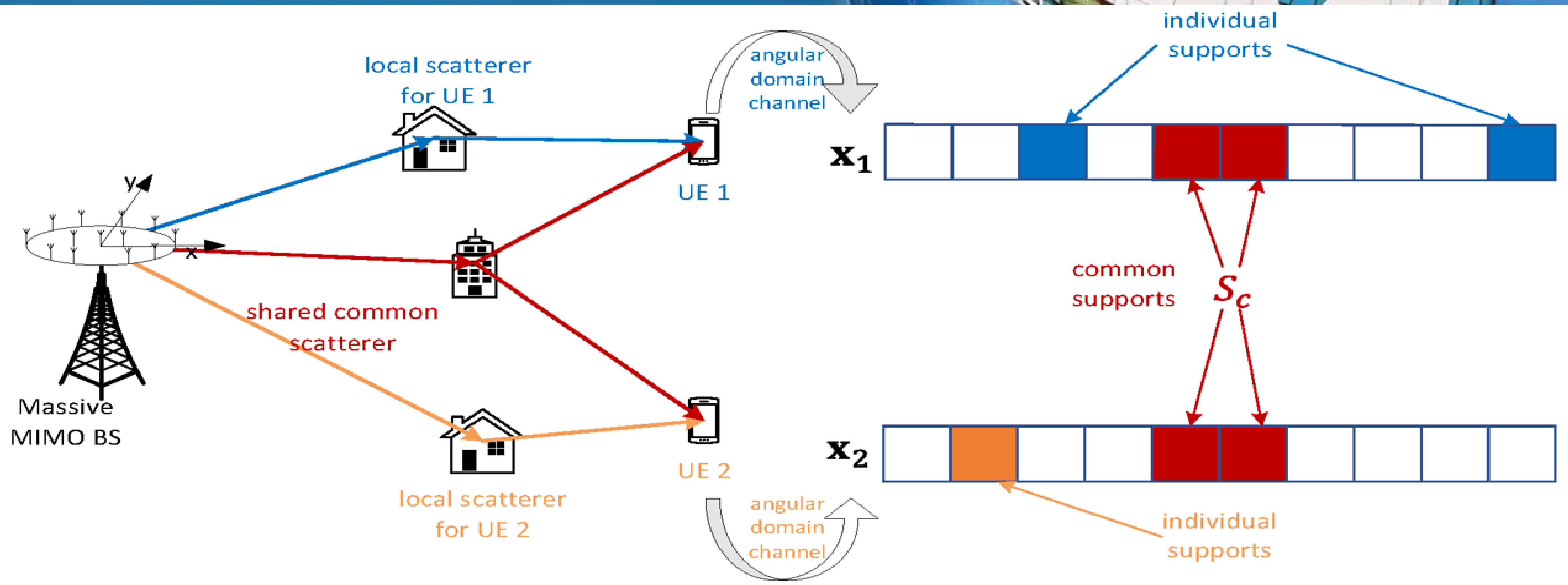


Fig 4: Clustered sparse channel $\mathbf{x}$ generated from the COST2100 channel model.

Different Sparsity Structures

Common Sparsity for MU-MIMO

- The channels of different users are usually correlated as they tend to share some common local scatterers at the BS
- Channels of different users usually share some partial common supports.



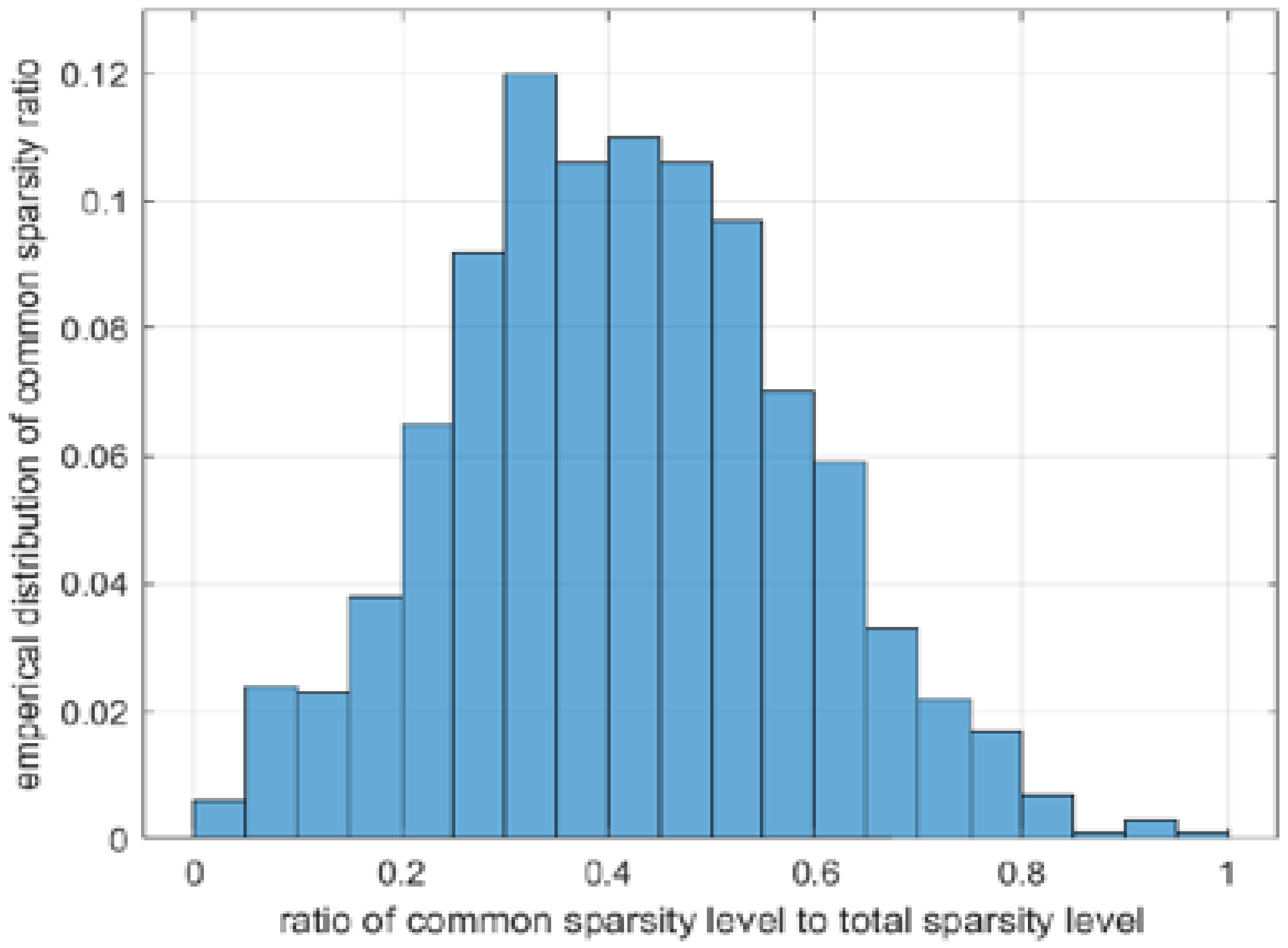


Fig 5: Histogram of ratio of common sparsity level .

The histogram in Fig. 5 is generated for COST2100 channel model

- 2.6-GHz NLoS semi-urban environments with closely spaced users
- 20 users randomly clustered in a 5m x 5m square
- A common support is identified when it is a support for all the 20 users

Common sparsity exists in the MU-MIMO channel, and typical ratio of common sparsity level ranges in 30% ~ 50%

Exploiting Different Sparsity Structures

Optimization-Based Approach

- *Random Sparsity*: L1-norm regularized LS (standard lasso)

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2+\lambda\|\mathbf{x}\|_1$$

  with $\mathbf{A}=\mathbf{S}\mathbf{F}$ being the overall measurement matrix

- *Clustered Sparsity*: Use block sparse lifting transform $\mathbf{x}=\mathbf{L}\mathbf{z}$ and $l_{2,1}$-norm regularization for clustered sparsity [2]:

$$\min_{\mathbf{z}} \frac{1}{2}\|\mathbf{y}-\mathbf{A}\mathbf{L}\mathbf{z}\|_2^2+\lambda\|\mathbf{z}\|_{2,1}$$

- *Common Sparsity*: given statistical common support information in the MU channels (e.g., probability of being a common support), use weighted L1-norm regularization [3]:

$$\min_{x} \frac{1}{2}\|y-Ax\|_2^2+\sum_{n=1}^{N} w_n|x_n|$$

  smaller probability → larger w_n → larger sparsifying penalty

Bayesian Approach

- *Random Sparsity*: impose an i.i.d. Laplacian Prior on the random sparse channel:

$$p(\mathbf{x})=\prod_{n=1}^{N}\frac{1}{2\lambda}\exp\left(-\frac{|x_n|}{\lambda}\right)$$

- *Clustered Sparsity*: Impose an HMM prior on the clustered sparse channel $\mathbf{x}$ [4]:

$$p(\mathbf{s})=p(s_1)\prod_{n=1}^{N-1}p(s_{n+1}|s_n)$$

$$p(\mathbf{x}|\mathbf{s})=\prod_{n=1}^{N}\left(s_n\cdot\mathcal{CN}(x_n;0,\sigma_n^2)+(1-s_n)\delta(x_n)\right)$$

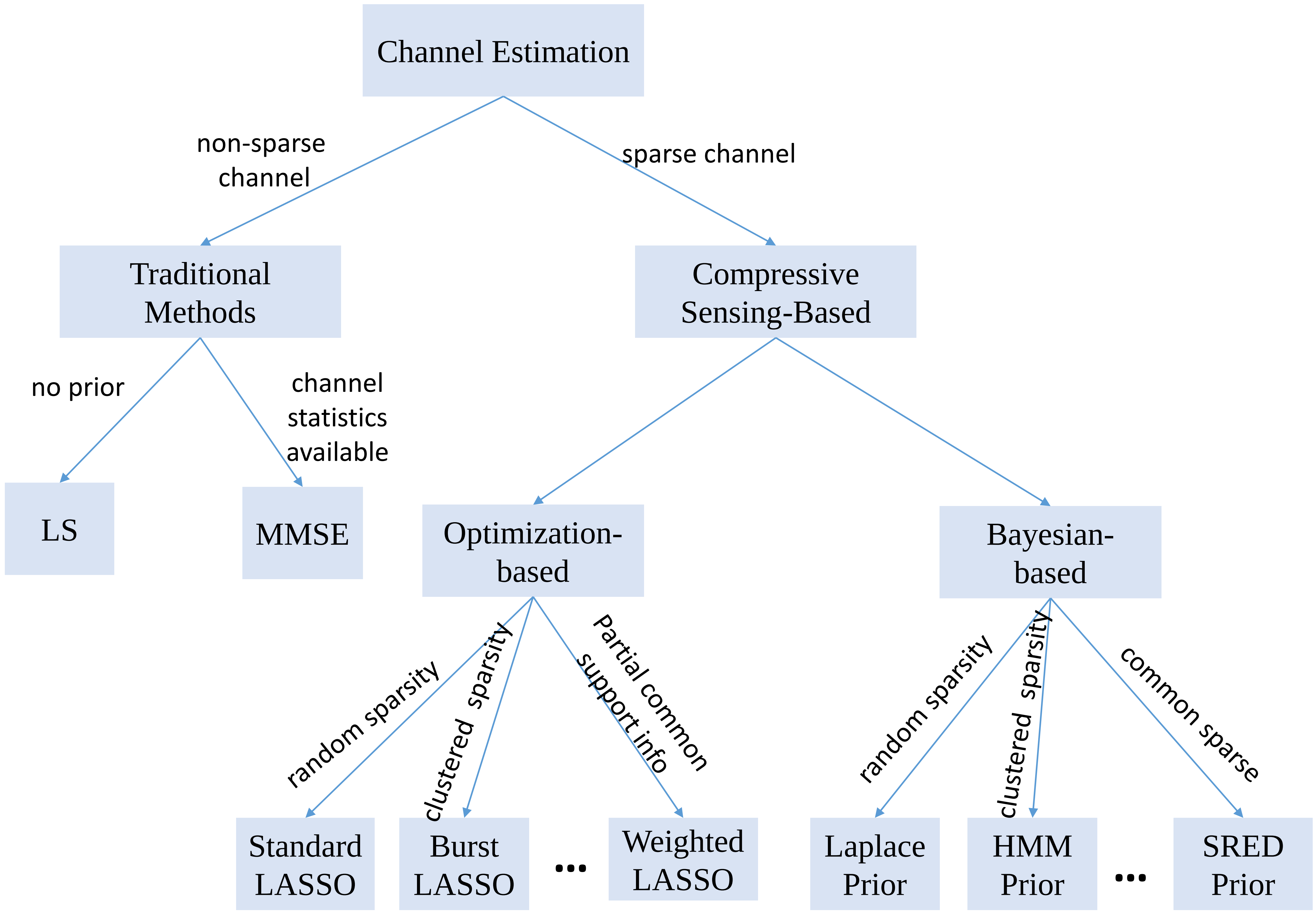
- *Common Sparsity*: impose a spherically-contoured radial exponential distribution (SRED) on the common-sparse $\mathbf{x}$ [5]:

$$p(\mathbf{x})\sim\prod_{g=1}^{G}\lambda^B\exp\left(-\lambda\|\mathbf{x}_g\|_2\right)$$

  with $\mathbf{x}_g$ being the g-th group of common sparse $\mathbf{x}$

[2] A. Liu, V. K. N. Lau and W. Dai, "Exploiting Burst-Sparsity in Massive MIMO With Partial Channel Support Information," in IEEE Transactions on Wireless Communications, Nov. 2016.
[3] L. Lian, A. Liu and V. K. N. Lau, "Weighted LASSO for Sparse Recovery With Statistical Prior Support Information," in IEEE Transactions on Signal Processing, 15 March15, 2018.
[4] A. Liu, L. Lian, V. K. N. Lau and X. Yuan, "Downlink Channel Estimation in Multiuser Massive MIMO With Hidden Markovian Sparsity," in IEEE Transactions on Signal Processing, 15 Sept.15, 2018.
[5] X. Zheng, A. Liu and V. Lau, "Joint Channel and Location Estimation of Massive MIMO System With Phase Noise," in IEEE Transactions on Signal Processing, 2020.

Exploiting Different Sparsity Structures



For CS-base solution, the pilot overhead can be significantly reduced by exploiting sparsity structure.

For example, standard LASSO algorithm only requires the pilot length grows with [6]

$$L \sim p \cdot log(N) \ll N$$

i.e., L grows linearly with sparsity level, and only logarithmically with channel dimension.

Problems with CS-based solution:

1. No closed-form solution
2. Requires an iterative algorithm to find solution
3. Iterative algorithms usually have high computational complexity, thus difficult to be applied for massive MIMO CE in real-time (within a TTI < 1ms for 5G).

[6] Wright, John, and Yi Ma. *High-dimensional data analysis with low-dimensional models: Principles, computation, and applications*. Cambridge University Press, 2022.

Compressive Sensing v.s. Deep Learning

Signal Model	Inverse Model
$\mathbf{Y} = \mathbf{HS} + \mathbf{N}$	$\hat{\mathbf{H}}_\Omega = f_\Omega(\mathbf{Y})$

CS-based CE: the OMP algorithm

$\mathbf{Y} \rightarrow$

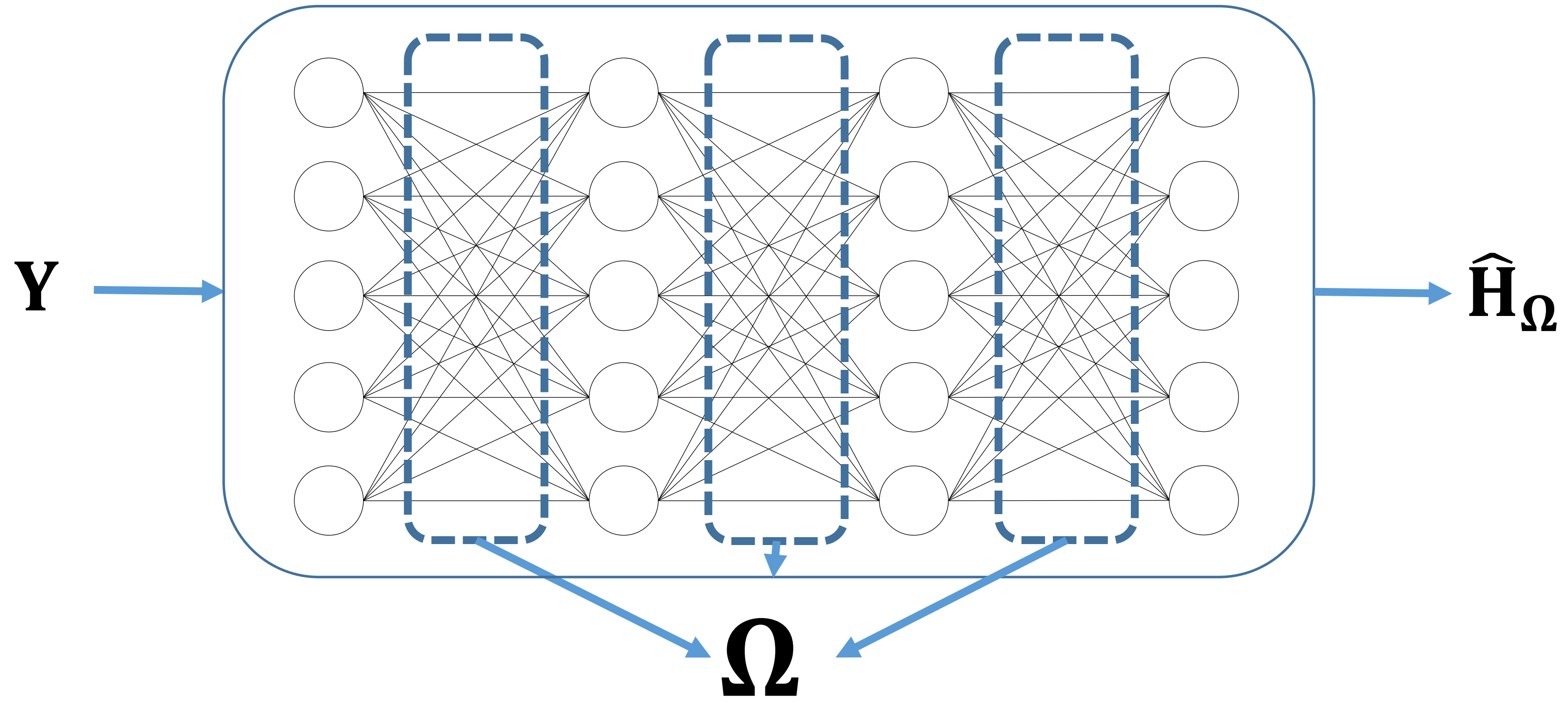
Initialize $\mathbf{r}_0 = \mathbf{y}$, $\Lambda_0 = \emptyset$.
for $t = 1,2,\ldots$ **do**
  $\lambda_t = \mathrm{argmax}_{k \notin \Lambda_{t-1}} \left|\boldsymbol{\xi}_k^H \mathbf{r}_{t-1}\right|$
  $\Lambda_t = \Lambda_{t-1} \cup \{\lambda_t\}$
  $\mathbf{x}_t = \mathrm{argmin}_{\mathbf{x}} \left\|\boldsymbol{\Xi}_{\Lambda_t}\mathbf{x} - \mathbf{y}\right\|_2$
  $\mathbf{r}_t = \mathbf{y} - \boldsymbol{\Xi}\mathbf{x}_t$
end
$\hat{\mathbf{H}} = \mathbf{F}\mathbf{X}_t$

$\rightarrow \hat{\mathbf{H}}$

vectorized signal $\mathbf{y} = \boldsymbol{\Xi}\mathbf{x} + \mathbf{n}$ with $\mathbf{y}, \mathbf{x} = \mathrm{vec}(\mathbf{Y}), \mathrm{vec}(\mathbf{X})$, $\boldsymbol{\Xi} = \mathbf{S}^H \otimes \mathbf{F}$

Replaced by DNN

DNN-Based Channel Estimator



Fast inferencing: forward propagation of a trained DNN is very fast, suitable for real-time channel inferencing

Table 6: Estimation and Computational Performance of DNN Compared with Iterative OMP and PGD Algorithms for a 64 x 64 MIMO.

	NMSE			CPU time (in sec.)		
# of samples	OMP	PGD	DNN	OMP	PGD	DNN
3200	0.4706	0.0027	0.1524	193.0590	6.1419	0.2809
1600	0.5802	0.0082	0.1903	67.7541	5.5186	0.1532
800	0.6162	0.1608	0.3112	38.7693	4.9726	0.1209
400	0.6852	1.1680	0.4221	26.1790	2.4516	0.0920

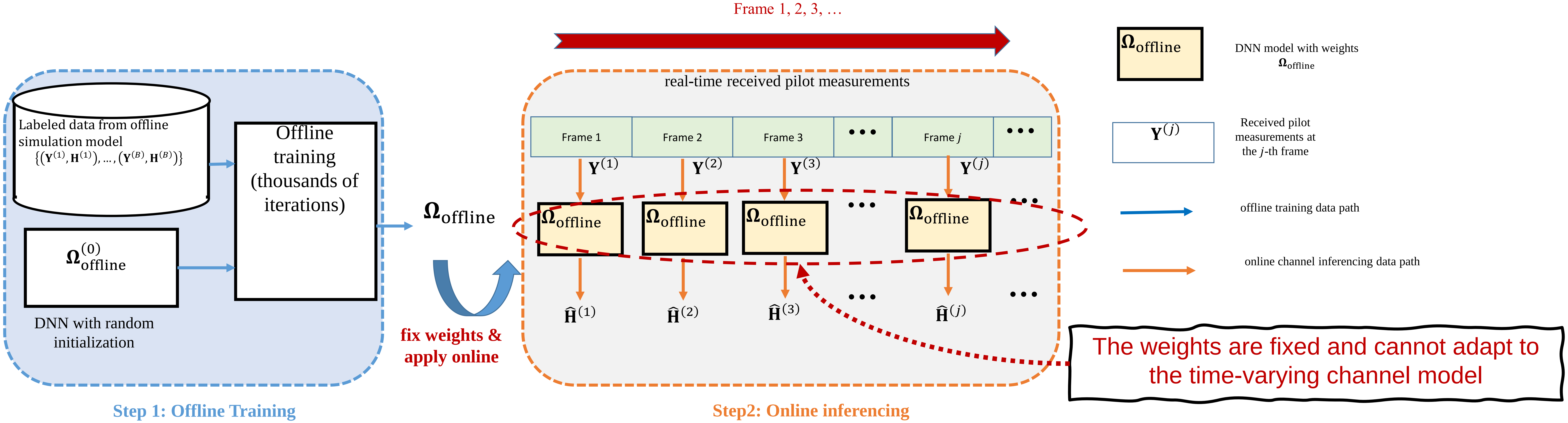
300x ~ 700x speed up than the iterative OMP algorithm.

Traditional Offline DNN-Based CE

- Traditional DL-based channel estimators are trained offline based on MSE loss, training is modeled as an optimization problem over all the trainable weights

$$\arg\min_{\mathbf{\Omega}} \mathcal{L}_{\mathrm{MSE}}\left(\hat{\mathbf{H}}_{\mathbf{\Omega}}; \mathbf{H}\right) = \arg\min_{\mathbf{\Omega}} \left\| f_{\mathbf{\Omega}}\left(\mathbf{Y}\right) - \mathbf{H} \right\|_F^2$$

- The MSE loss needs truth **H** in labeled data pairs $(\mathbf{Y}, \mathbf{H})$ for supervised training. But in practice, true **H** are difficult to obtain, and are generated offline according to certain channel model. The training and CE are divided into two stages:
  - ***Offline Training Stage***: tune the DNN weights offline based on MSE loss and channel labels
  - ***Online Inferencing Stage***: after offline training, fix the weights for CE for online deployment
- **Problem**: the offline trained DNN **cannot adapt its weights to the channel model in actual scenario**.



Desire for Online DNN

Can we have an online training scheme that

1. learns the channel model online, while
2. enjoying fast channel inferencing

at the same time?

Online DNN for Point-to-Point Massive MIMO Channel Estimation

X. Zheng and V. K. N. Lau, "Online Deep Neural Networks for MmWave Massive MIMO Channel Estimation With Arbitrary Array Geometry," in *IEEE Transactions on Signal Processing*, vol. 69, pp. 2010-2025, 2021.

X. Zheng and V. K. N. Lau, "Simultaneous Learning and Inferencing of DNN-Based mmWave Massive MIMO Channel Estimation in IoT Systems With Unknown Nonlinear Distortion," in IEEE Internet of Things Journal, vol. 9, no. 1, pp. 783-799, 1 Jan.1, 2022.

Online DNN-Based CE

- In order to enable online training, we need an online loss function that (i) does not need true **H**, (ii) channel model free and (iii) measures the error with the true **H**. Formally, we define an online loss function as the following:

Definition 1. An online loss function for channel estimation

$$\mathcal{L}\left(\hat{\mathbf{H}};\theta\right) \qquad \{\mathbf{Y},\mathbf{S}\} = \boldsymbol{\theta}$$

should satisfy the following four requirements:

1) [Online Requirement] $\mathcal{L}(\cdot)$ is a function of observed measurements $\mathbf{Y}$, deep neural network output $\hat{\mathbf{H}}$ (without requiring true channel $\mathbf{H}$ nor the underlying channel model).
2) [Regularity Requirement] $\mathcal{L}\left(\hat{\mathbf{H}};\theta\right)$ is continuous in $\hat{\mathbf{H}}$ and θ.
3) [Consistency Requirement] There exists a constant C, such that

$$\left\|\hat{\mathbf{H}}^{*} - \mathbf{H}\right\|_F^2 \leq \frac{C}{\rho/\sigma_n^2}$$

where $\hat{\mathbf{H}}^{*}$ is the minimizer of the loss function, i.e.,

$$\hat{\mathbf{H}}^{*} = \arg\min_{\hat{\mathbf{H}}} \mathcal{L}\left(\hat{\mathbf{H}};\theta\right).$$

1) exempts the training from the need for labeled channel data ad the underlying channel model, making online training possible

2) guarantees that the minimizer, as an inverse mapping of the loss function, can be approximated by the DNN

3) guarantees that when the DNN is trained to minimize the loss function, the output will be driven close to the true channel for large signal-to-noise ratio (SNR)

Online Training Formulation

- Define the mapping $\boldsymbol{\eta}: \mathbf{X} \to \tilde{\mathbf{x}} = \mathrm{vec}\left(\left[\ \mathrm{Re}\left(\mathbf{X}\right) \quad \mathrm{Im}\left(\mathbf{X}\right)\ \right]\right)$
- The input to the DNN is $\tilde{\mathbf{y}} = \boldsymbol{\eta}\left(\mathbf{Y}\right)$. The output is denoted by $\tilde{\mathbf{h}} = \boldsymbol{\eta}\left(\mathbf{H}\right)$
- $\boldsymbol{\Omega} = \left\{\mathbf{W}^{[l]}, \mathbf{b}^{[l]}\right\}_{l=1}^{L}$ are the learnable weights in an L-layer DNN
- $\hat{\mathbf{H}}_{\mathrm{DNN}}\left(\tilde{\mathbf{y}};\boldsymbol{\Omega}\right) = \boldsymbol{\eta}^{-1}\left(\tilde{\mathbf{h}}\right)$ is the estimated channel matrix
- Training of the DNN can be modeled as an **optimization problem**

$$\boxed{\boldsymbol{\Omega}^{*} = \arg\min_{\boldsymbol{\Omega}} \mathrm{E}\left\{\mathcal{L}\left(\hat{\mathbf{H}}_{\mathrm{DNN}}\left(\tilde{\mathbf{y}};\boldsymbol{\Omega}\right);\boldsymbol{\theta}\right)\right\}}$$

Online Loss vs. Offline Loss

Online Loss Function

$$\mathcal{L}_{\text{online}}\left(\widehat{\mathbf{H}_{\Omega}};\boldsymbol{\theta}\right),\ \ \{\mathbf{Y},\mathbf{S}\}=\boldsymbol{\theta}$$

1. Does not need true channel labels $\mathbf{H}$, nor any prior knowledge about the channel model or antenna geometry.
2. Training is based on real-time received measurements $\mathbf{Y}$ only, which contains information about the actual channel model.
3. Training can be implemented online where the training data comes in a streaming mode. The weights can adapt to the time-varying channel model while generating channel estimation.

Offline Loss Function

$$\mathcal{L}_{\text{offline}}=\left\|\widehat{\mathbf{H}_{\Omega}}-\mathbf{H}\right\|_F^2$$

1. Need paired-labels $(\mathbf{Y},\mathbf{H})$ for supervised training.
2. Labels generated offline according to some channel model & antenna geometry, which may differ from the actual scenario.
3. Training is implemented offline. Then the weights are fixed for online inferencing, during which the DNN cannot tracking the changing channel model.

Example Online Loss Functions

E.g. 1: LS Online Loss

For general "non-sparse" channel matrix $\mathbf{H}$, we propose the least square online loss function:

$$\mathcal{L}_{\text{LS}}\left(\hat{\mathbf{H}};\boldsymbol{\theta}\right)=\left\|\mathbf{Y}-\hat{\mathbf{H}}\mathbf{S}\right\|_F^2,\quad \text{where } \boldsymbol{\theta}=\{\mathbf{Y},\mathbf{S}\}.$$

Clearly, it satisfies the

1. **online requirement**: it does not need true $\mathbf{H}$ as training labels and does not depend on any channel model nor antenna geometry
2. **regularity requirement**: it is continuous in $\mathbf{Y}$ and $\mathbf{S}$.
3. The **consistency requirement** is satisfied by the lemma on the right side.

Lemma 3: [Consistency of the Online LS Loss Function] If the transmitted pilot matrix $\mathbf{S}$ is row orthogonal such that $\mathbf{S}\mathbf{S}^H=\frac{\rho L_s}{K}\mathbf{I}_K$ with pilot length $L_s\geq K$, and let $\hat{\mathbf{H}}^*$ be the minimizer of the LS loss function (16) given by

$$\hat{\mathbf{H}}^*=\arg\min_{\hat{\mathbf{H}}}\left\|\mathbf{Y}-\hat{\mathbf{H}}\mathbf{S}\right\|_F^2. \tag{17}$$

Then, $\hat{\mathbf{H}}^*$ has an error in the Frobenius norm given by

$$\left\|\hat{\mathbf{H}}^*-\mathbf{H}\right\|_F^2\leq\frac{\alpha^2N_r}{L_s\left(\rho/\sigma_n^2\right)}. \tag{18}$$

with probability at least $1-\exp(-\frac{1}{2}N_r(\alpha-1)^2)$ for any $\alpha>1$.

Least Square online loss function still requires pilot length > channel dimension. We can exploit channel sparsity in the online loss design for reduced pilot overhead.

Example Online Loss Functions

E.g. 2: Nuclear-Norm Based Online Loss

For sparse channels, we propose the nuclear-norm regularized online loss function:

$$\mathcal{L}_{\text{Nuclear}}\left(\hat{\mathbf{H}};\boldsymbol{\theta}\right)=\frac{1}{2}\left\|\mathbf{r}-\mathcal{A}\left(\hat{\mathbf{H}}\right)\right\|_2^2+\gamma\left\|\hat{\mathbf{H}}\right\|_*,\{\gamma,\mathbf{Y},\mathbf{S}\}\in\boldsymbol{\theta}, \quad (1)$$

- $\mathbf{r}$ is some M-dim linear & noisy measurement of H under a linear mapping $A(\mathbf{H})$
- $\|\mathbf{H}\|_*$ is the nuclear norm, well-known for imposing rank-sparsity.

Clearly, the 1)**online requirement** and the 2)**regularity requirement** are satisfied.

To satisfy the 3) **consistency requirement**, linear mapping A should satisfy rank-RIP, i.e., $\forall\mathbf{H}$ with rank($\mathbf{H}$)≤d, $\exists\delta_d$ with $0<\delta_d<0$, s.t.

$$(1-\delta_d)\|\mathbf{H}\|_F^2\leq\|\mathcal{A}(\mathbf{H})\|_2^2\leq(1+\delta_d)\|\mathbf{H}\|_F^2 .$$

The rank-RIP can be satisfied (w.h.p.) by subspace sampling on $\mathbf{y}(t)$, i.e., at channel use t:

$$\mathbf{r}(t)=\mathbf{F}^H(t)\mathbf{H}\mathbf{s}(t)+\mathbf{F}^H(t)\mathbf{n}(t),t=1,2,...,L.$$

The combining matrices $\mathbf{F}(t)\in\mathbb{C}^{N\times M/L}$ and pilots $\mathbf{s}(t)$ are generated randomly [7] according to

$$[\mathbf{F}(t)]_{n,m}=\frac{\sqrt{K}}{\sqrt{NL}}\exp\left(j\eta_{n,m}^{(t)}\right),\quad[\mathbf{s}(t)]_k=\frac{\sqrt{\rho}}{\sqrt{K}}\exp\left(j\xi_k^{(t)}\right),$$

where $\eta_{n,m}^{(t)}$ and $\xi_k^{(t)}$ are uniformly distributed in $[0,2\pi)$.

With rank-RIP satisfied, we have the following Theorem for its consistency requirement.

Theorem

(Consistency of the Online Loss Function) Given $\mathcal{A}$ satisfies the rank-RIP with constant δ_{4P}, and let $\hat{\mathbf{H}}^*$ be the minimizer of the online loss function (1), then with probability at least 1-2exp$(-qM)$, $\hat{\mathbf{H}}^*$ has an error in Frobenius norm given by

$$\left\|\hat{\mathbf{H}}^*-\mathbf{H}\right\|_F^2\leq\frac{C_1\max(K,N)P}{M(\rho/\sigma_n^2)}$$

with $\gamma=16\sqrt{\frac{\sigma^2}{\rho}}\cdot\max(K,N)$ provided $M\geq 8P(N_r+K+1)$ for some constant $C_1,q>0$.

[7] W. Zhang, et al. "Leveraging the Restricted Isometry Property: Improved Low-Rank Subspace Decomposition for Hybrid Millimeter-Wave Systems," in IEEE Transactions on Communications, Nov. 2018.

Online Training Algorithm

Training can be implemented on-the-fly based on real-time received pilot measurements, where the training data comes in a streaming mode.

Algorithm 1: Online Training Algorithm.

Input: Pilot $\mathbf{S}$, received pilot measurements $\mathbf{Y}^{(j)}$, $j = 1, 2, 3, \ldots$, online loss function $\mathcal{L}(\cdot)$.

Output: $\hat{\mathbf{H}}_{\mathrm{DNN}}^{(j)}$, $j = 1, 2, 3, \ldots$.

Initialize: $j = 0$, DNN weights $\boldsymbol{\Omega} = \boldsymbol{\Omega}^{(0)}$.

while Online training mode is on **do**

- Obtain received sample $\mathbf{Y}^{(j)}$ at the j-th frame.
- [FP] Compute $\mathbf{z}^{[l]}$, $\mathbf{a}^{[l]}$, $l = 1, \ldots, L$ by (10) for $\mathbf{Y}^{(j)}$.
- [BP] Compute $d\mathbf{a}^{[L]}$ by (15) and use BP to compute $d\mathbf{W}^{[l]}$, $d\mathbf{b}^{[l]}$, $\forall l$ to obtain $d\boldsymbol{\Omega}^{(j)}$ for $\mathbf{Y}^{(j)}$.
- Update the weights $\boldsymbol{\Omega}^{(j)}$ based on the first and second moments of $d\boldsymbol{\Omega}^{(j)}$.
- Output CE for $\mathbf{Y}^{(j)}$ by $\hat{\mathbf{H}}_{\mathrm{DNN}}^{(j)} = \boldsymbol{\eta}^{-1}(\mathrm{DNN}_{\boldsymbol{\Omega}^{(j)}}(\tilde{\mathbf{u}}^{(j)}))$.
- $j \leftarrow j + 1$.

end while

Online Learning: The online training algorithm will update the DNN weights on-the-fly whenever a pilot measurement is received in a frame

Simultaneous Inferencing: the DNN will output the CE based on its current weights simultaneously

ϵ-Analysis of Online DNN-Based CE

Theorem

[ϵ-Analysis for online DNN] Given a legitimate online loss function $\mathcal{L}\left(\hat{\mathrm{H}}_{\Omega};\boldsymbol{\theta}\right)$ satisfying the properties in Definition 1, for any given $\epsilon > 0$, there exists a deep neural network with width M and at most $L = 2\left(\lfloor \log_2 2KN \rfloor + 2\right)$ layers, such that the output $\tilde{\mathrm{h}} = \mathrm{DNN}\left(\tilde{\mathrm{y}}\right)$ satisfies

$$\sup_{\mathrm{Y}\in\mathcal{Y}} \left\|\boldsymbol{\eta}^{-1}\left(\tilde{\mathrm{h}}\right) - \mathrm{H}\right\|_F^2 \leq \frac{C}{\rho/\sigma_n^2} + \epsilon$$

for some constant C, with probability at least $1 - e^{-rNL/5}$ in the compact set $\mathcal{Y} = \left\{\mathrm{Y} : \tilde{\mathrm{y}}^T\tilde{\Sigma}^{-1}\tilde{\mathrm{y}} \leq 4NLr\right\}$ for $r \geq 1$ and some positive definite matrix $\tilde{\Sigma}$.

Sketch of proof:

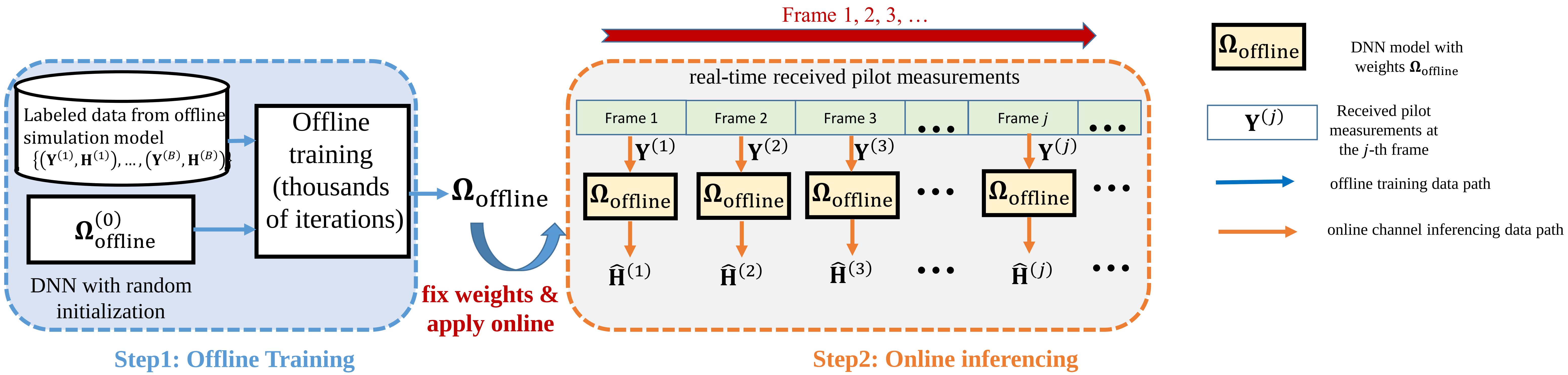
1. Constrain the DNN input **Y** in some compact set $\mathcal{Y}$ w.h.p.
2. With *Regularity Reqrm.*, prove that the inverse mapping $\hat{\mathbf{H}}^* = \min_{\hat{\mathbf{H}}} \mathcal{L}\left(\hat{\mathbf{H}};\boldsymbol{\theta}\right)$ is a continuous mapping on $\mathcal{Y}$. [The Maximum Theory].
3. Bound the error between DNN output and the inverse mapping by a small approximation error $\boldsymbol{\epsilon}$. [Universal Approximation Theorem]
4. With *Consistency Reqrm.*, bound the error between the DNN output and the true channel using the triangle inequality.

- The theorem states that given a legitimate online loss function, there exists a "large enough" DNN, such that its output will approximate the true channel with arbitrary accuracy.
- The $\boldsymbol{\epsilon}$ -analysis above gives an error bound between the DNN output (with bounded layer) and the truth channel
  - $\boxed{\frac{C}{\rho/\sigma_n^2}}$ is the error induced by system noise, which can be arbitrarily small for large enough SNR
  - $\boxed{\boldsymbol{\epsilon}}$ is the DNN approximation error, which can be arbitrarily small for DNN with large enough width

Online Training v.s Offline Training

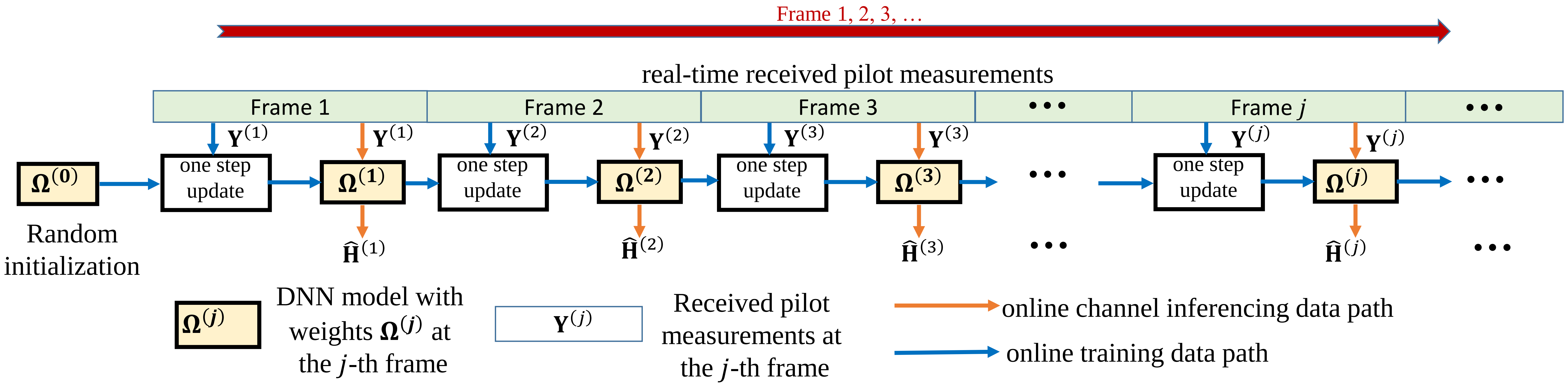
Offline DNN:

Separate offline training and online training stage, with weights fixed during online inferencing.



Online DNN:

Simultaneous online training and online inferencing. The DNN weights are continuously updated on-the-fly to track the time-varying channel model.



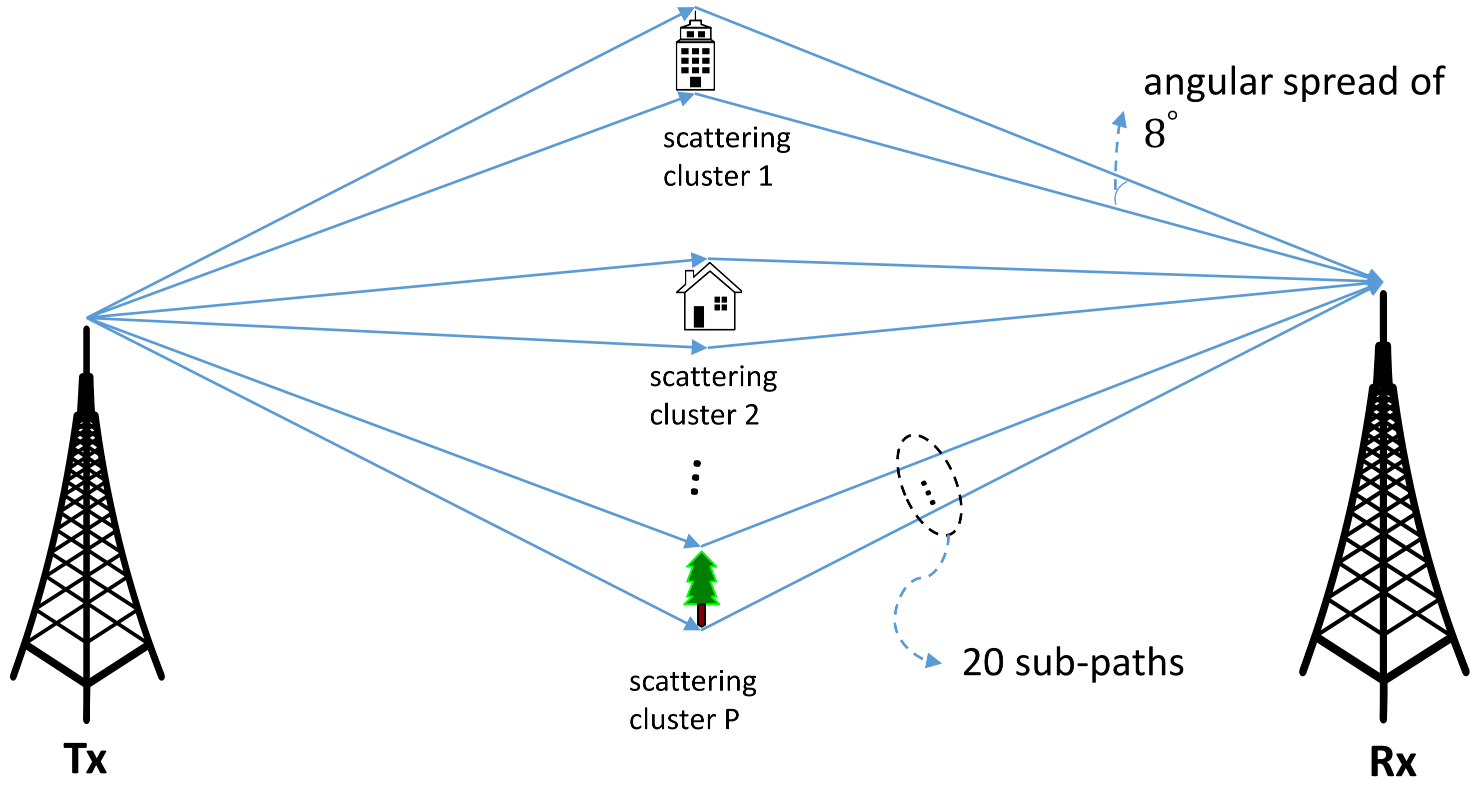
Computational Complexity

Table 7: Complexity and Computational Performance of DNN Compared with Iterative Algorithms for a 64 x 10 MIMO.

Algorithm	Complexity per	Overall complexity	CPU time of different algorithms	
			Algorithm	CPU time (in ms)
Online DNN Inferencing	$\Theta(N_r^4)$	$\Theta(LN_r^4)$	One step channel inferencing	**0.299**
Burst LASSO	$\Theta(D^3 N_r^6)$	$\Theta(\kappa D^3 N_r^6)$	Burst LASSO on one sample	**1248**
GOMP	$\Theta(L_s N_r^4)$	$\Theta(\kappa L_s N_r^4)$	GOMP on one sample	**16.323**
MMSE	-	$\Theta(N_r^6)$	MMSE on one sample	**12.561**

- For clarity, the computational complexity is computed for a MIMO with $N_r \times N_r$ MIMO. L is the number of layers in the DNN and each layer has $\Theta(N_r^2)$ neurons. κ is the number of iterations for iterative algorithms.
- For online DNN channel inferencing, the main computational burden is just matrix-vector multiplication during forward propagation, which has very low complexity.
- One step of channel inferencing is more than 1000x faster than Burst LASSO algorithm, and 50x faster than OMP-based algorithm.
- The fast inferencing enables real-time CE for massive MIMO.

Simulation: Comparison to Baselines



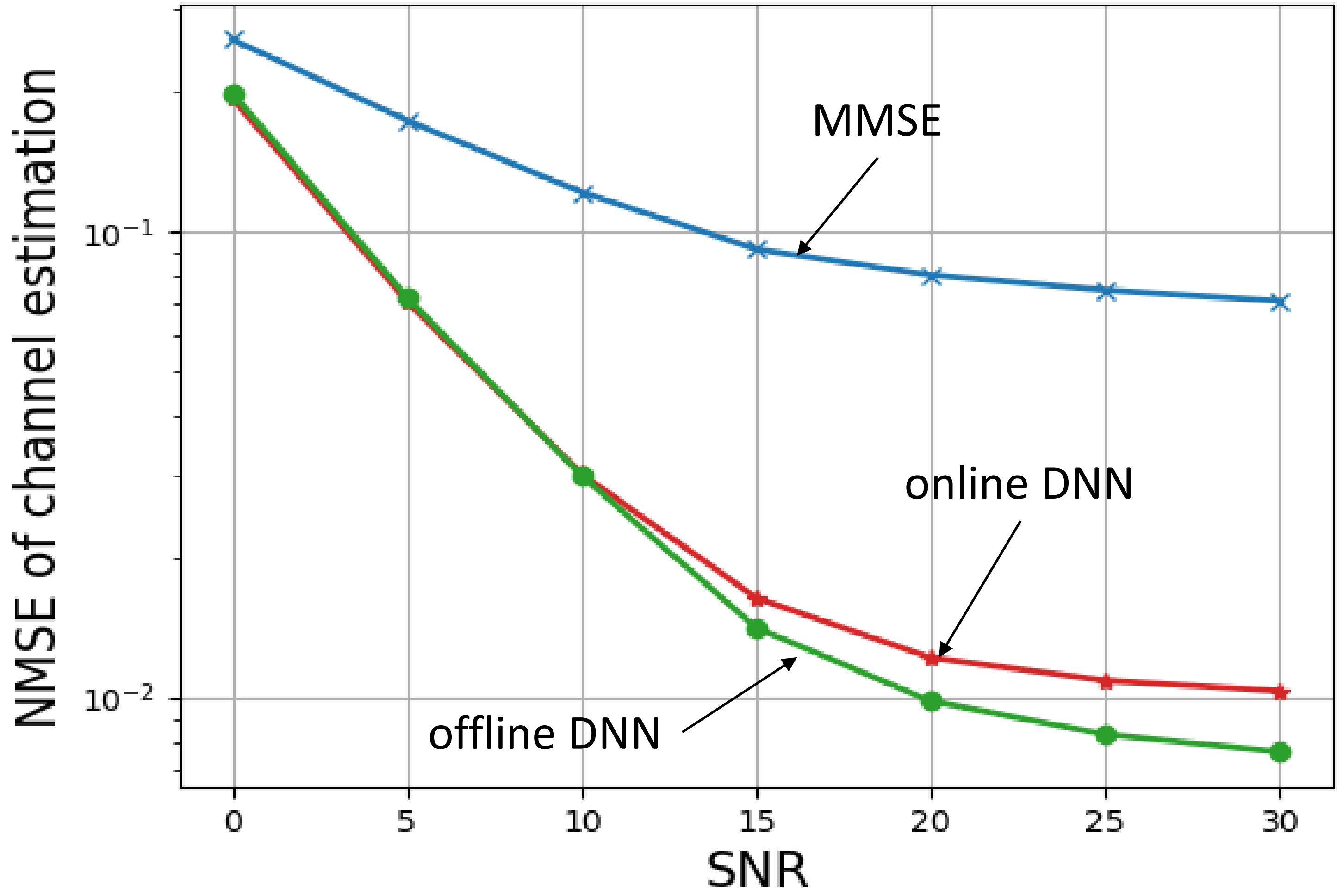


Fig. 8: CE NMSE versus SNR compared with baselines evaluated on 3GPP TR 25.996 SCM channel model.

Default Simulation Setup:

- N=64, M=10 antennas, pilot length = 8, transmit power ρ=1 with varying noise power.
- Channel model uses 3GPP SCM TR 25.996 for an urban macro propagation environment with P=2 clusters of scatters, each cluster produces 20 significant sub-paths with an angular spread of 8°.
- DNN structure: fully connected NN with 2 hidden layers, each with 240 neurons
- Online training scheme: 8000 steps of weight updates (i.e., 8000 pilot measurements) at SNR = 30 dB .

➢ MMSE does not perform well due to insufficient pilots

➢ Online DNN NMSE reaches that of offline DNN at high SNR when there is no model mismatch

➢ We will see the offline DNN will be prone to various model mismatch problems.

Robustness to Channel Model Mismatches

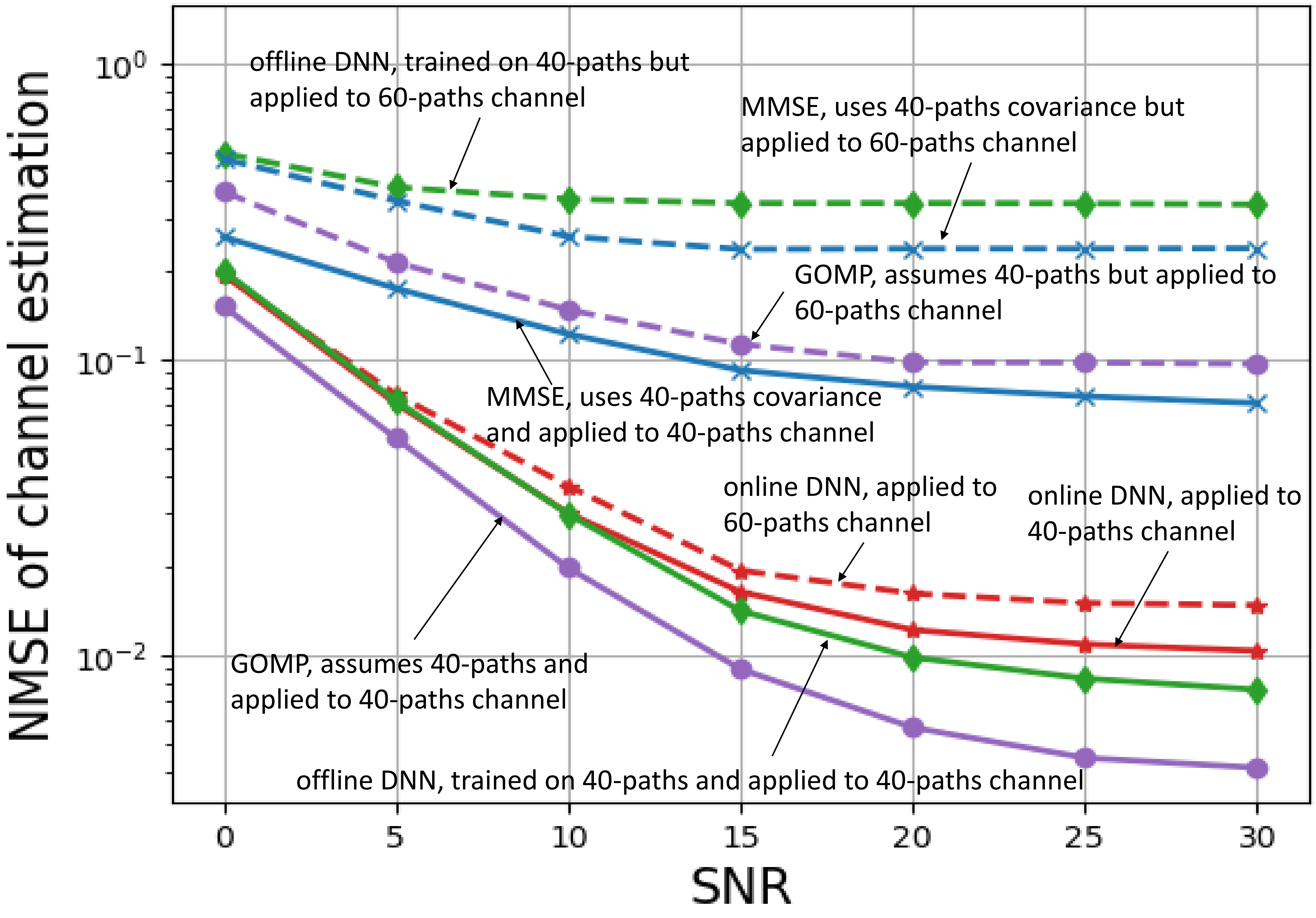


Fig. 9: CE NMSE versus SNR for **different propagation environments.**

- Baselines:
  - ***NMSE*** uses 40-paths covariance
  - ***GOMP*** assumes 40-paths channel model
  - ***Offline DNN*** trained on 40-paths channel data
- NMSE of the baselines degrades significantly as the propagation environment switches from 40-paths to 60-paths case.
- ***Online DNN*** can adapt to such change of propagation environment.

Robustness to Antenna Array Geometry

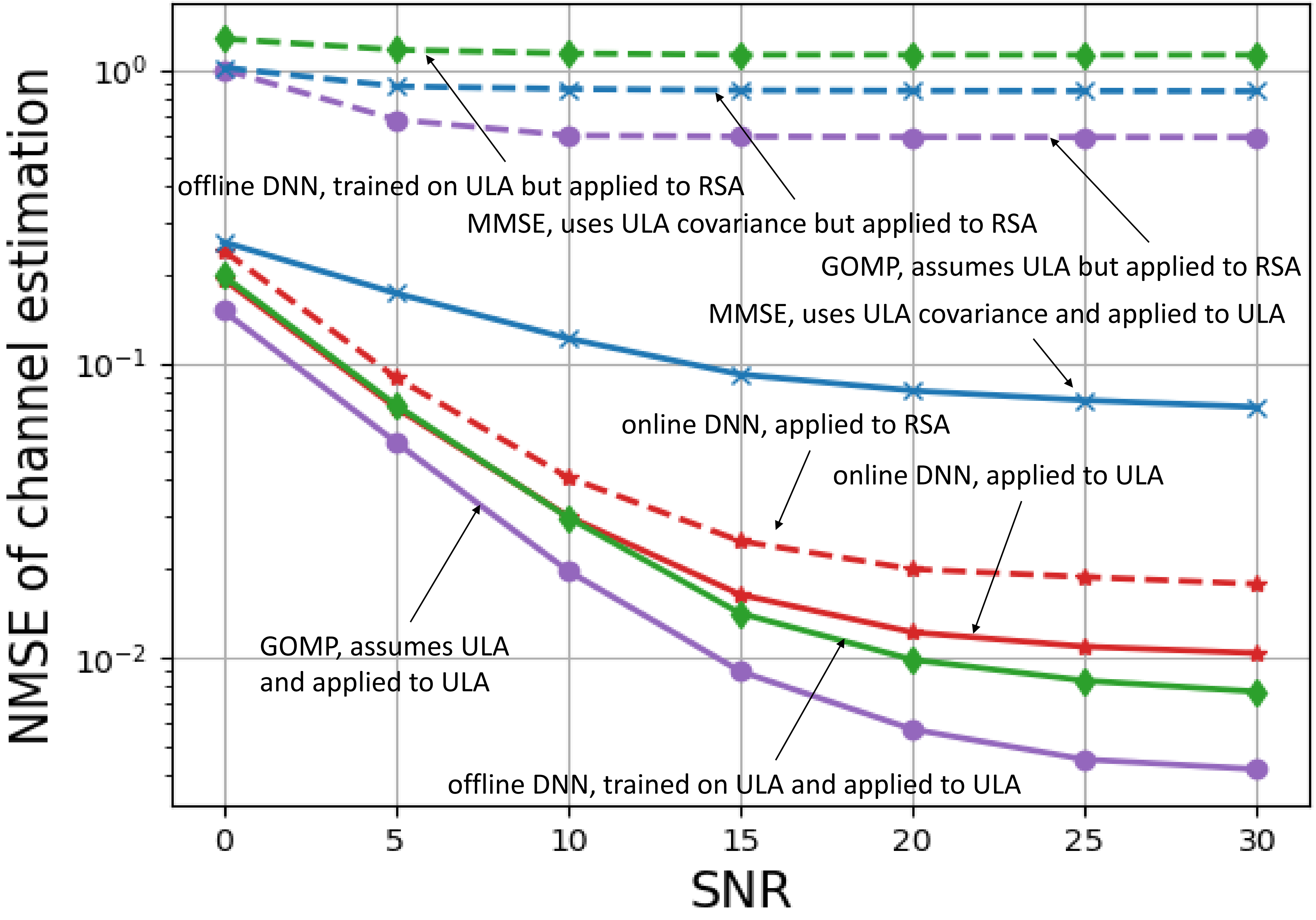
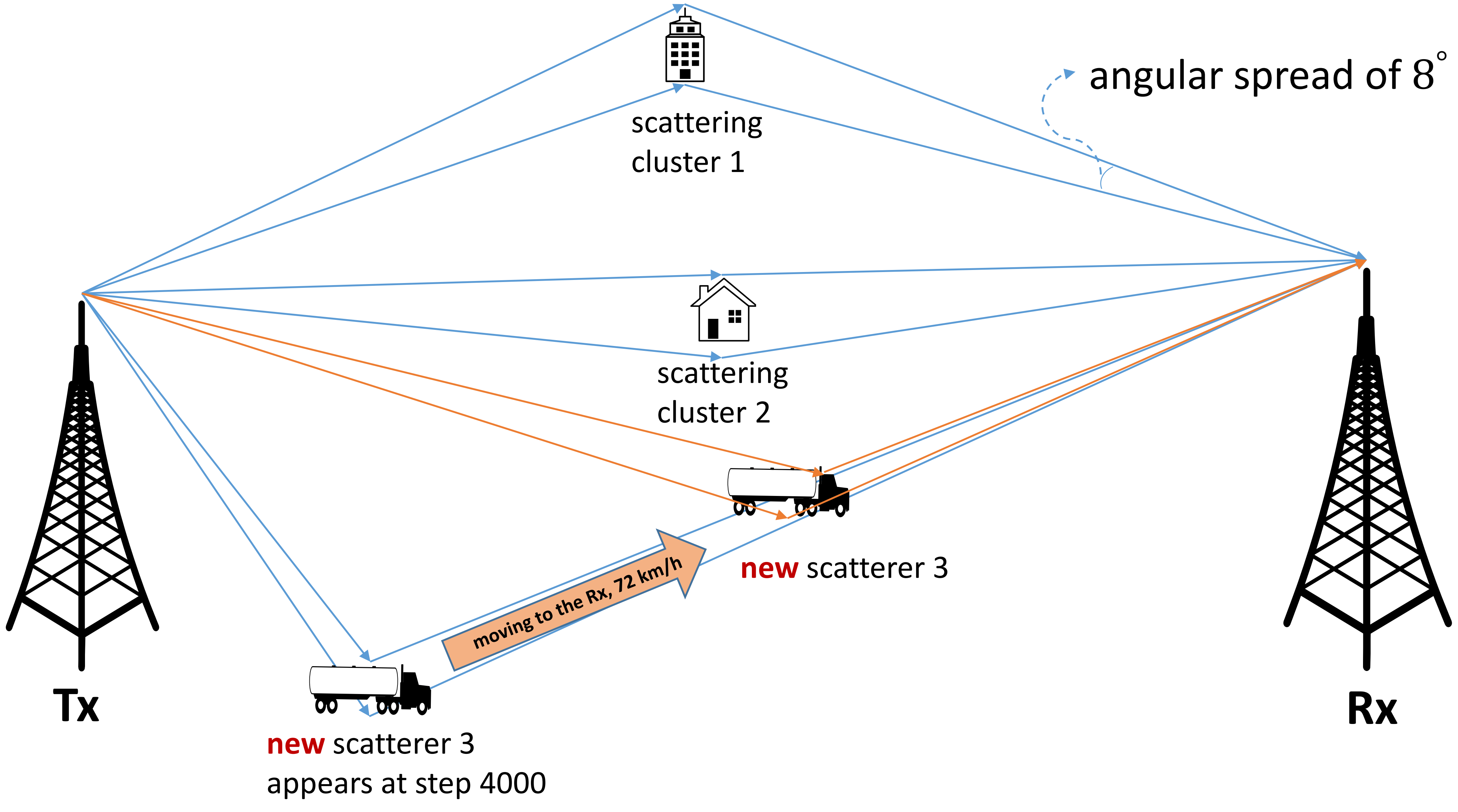


Fig. 10: CE NMSE versus SNR for **different antenna geometry**.

- Baselines:
  - ***NMSE*** uses channel covariance for MIMO equipped with ULA
  - ***GOMP*** assumes ULA antenna geometry
  - ***Offline DNN*** trained on channel data generated for ULA system
- NMSE of the baselines degrades significantly as the antenna geometry switches from ULA to RSA.
- ***Online DNN*** can learn the underlying antenna array geometry based on received pilot measurements

Tracking Ability



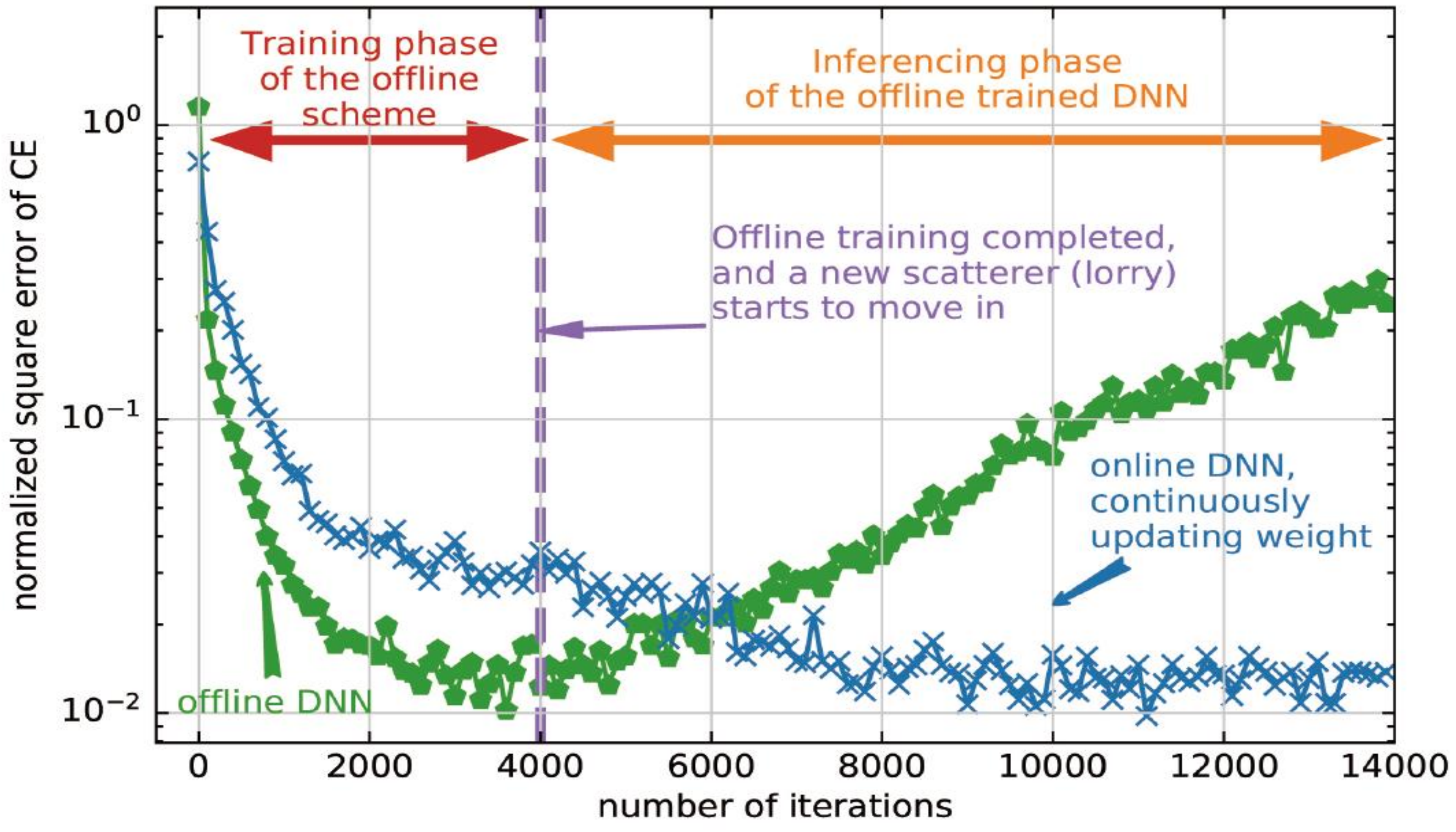


Fig. 11: Sample path of NMSE of CE v.s. weight update step number for online DNN and offline DNN.

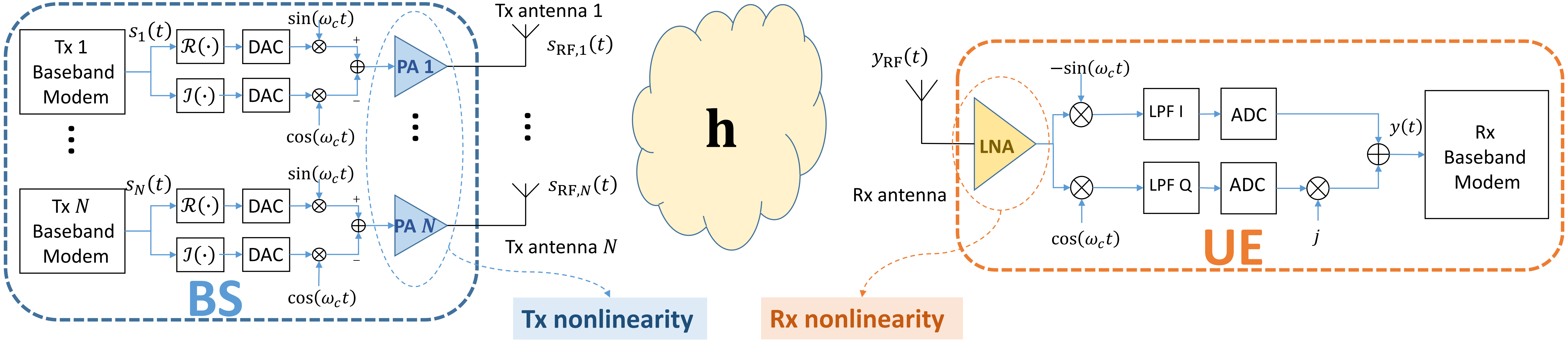
➢The environment is static for the first 4000 steps. A new scatterer (lorry) starts moving towards the Rx at the 4000-th step.

➢The offline DNN weights are fixed after completion of the training phase at the 4000-th step. But the online DNN is continuously updating the weight.

➢ *Offline DNN* CE error starts to increase after 4000-th step as the weights cannot adapt to the change of model induced by the new scatterer

➢ *Online DNN* can keep track of the channel model on-the-fly by adjusting the DNN weights, it still maintains good CE accuracy despite a changing propagation environment.

Extension to MIMO with Nonlinearity



Massive MIMO system with Nonlinearity:

- One BS with N antennas, one single-antenna user
- For downlink CE, the BS broadcasts pilot sequences $\mathbf{S} \in \mathcal{C}^{M \times N}$ of length M to the UE
- The nonlinear distorted, noise corrupted measurements at the UE is

$$\mathbf{y} = f_{rx}\left(f_{tx}\left(\mathbf{S}\right)\mathbf{h}\right) + \mathbf{z},$$

  - $\mathbf{h} \in \mathcal{C}^N$ is the spatial channel to be estimated
  - $f_{rx}(\cdot)$ and $f_{tx}(\cdot)$ are the transfer functions of the PA at the BS and LNA the UE, applied elementwisely to the envelope of symbols
  - The nonlinear transfer functions are, modeled as a Rapp model with clipping voltage V and smoothness factor p.

$$f\left(r; V, p\right) = rG\left(|r|\right) \quad \text{with } G\left(|r|\right) = \left(1 + \left(\frac{|r|}{V}\right)^{2p}\right)^{-1/2p},$$

Similarly, for online training, we need an online loss function that satisfies the three axioms. But we also need to incorporates the nonlinearity.

$$\mathcal{L}\left(\hat{\mathbf{x}}; \boldsymbol{\theta}, f_{\mathrm{tx}}, f_{\mathrm{rx}}\right), \quad \{\mathbf{y}, \mathbf{S}\} \in \boldsymbol{\theta},$$

Incorporates the nonlinearity

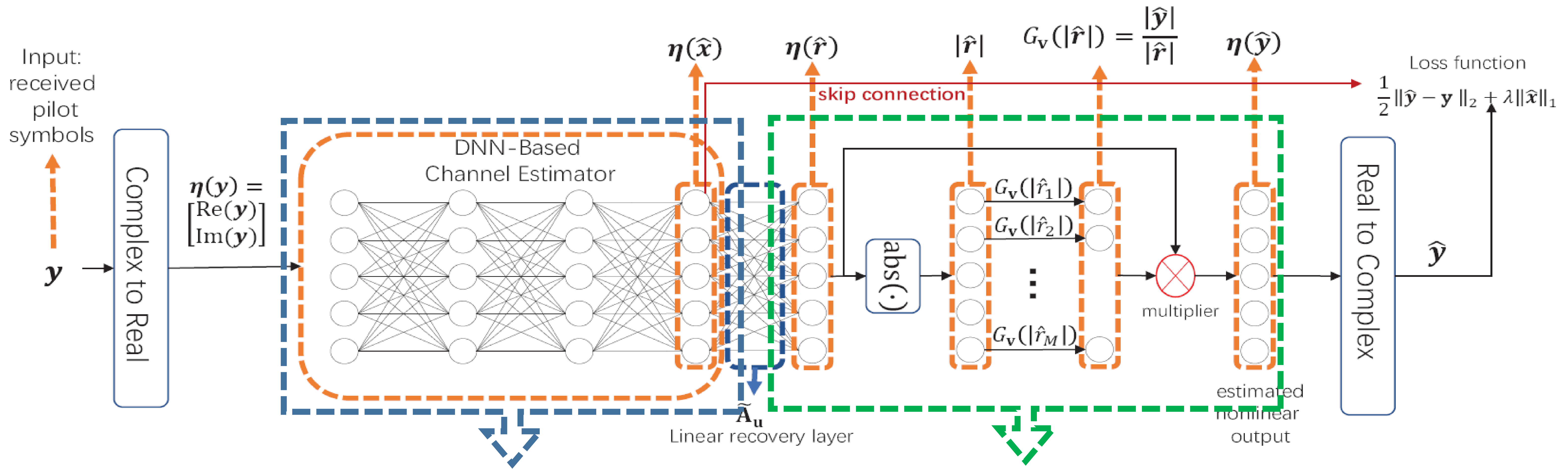
Two-Stage DNN Structure with Unknown Nonlinearity

- We introduce parameterized nonlinear modules $f_{\mathbf{u}}(\cdot)$ and $f_{v}(\cdot)$ to approximate $f_{tx}(\cdot)$ and $f_{rx}(\cdot)$
  - find model parameters u and v such that $f_{\mathbf{u}} \approx f_{tx}(\cdot)$ and $f_{v} \approx f_{rx}(\cdot)$
  - choice of nonlinearity approximation modules: polynomial model with limited number of odd orders
- The training should also update the nonlinear module parameters $\boldsymbol{\omega} = [\boldsymbol{u}, \boldsymbol{v}]$

$$\boldsymbol{\Omega}^{*}, \boldsymbol{\omega}^{*} = \arg\min_{\boldsymbol{\Omega}, \boldsymbol{\omega}} E\left\{\mathcal{L}_{\text{unk}}\left(\hat{\mathbf{x}}\left(\tilde{\mathbf{y}}; \boldsymbol{\Omega}\right); \boldsymbol{\theta}, f_{\mathbf{u}}, f_{\mathbf{v}}\right)\right\}$$

$$\text{with } \mathcal{L}_{\text{unk}}\left(\hat{\mathbf{x}}; \boldsymbol{\theta}, f_{\mathbf{u}}, f_{\mathbf{v}}\right) = \frac{1}{2}\left\|\mathbf{y} - f_{\mathbf{v}}\left(f_{\mathbf{u}}\left(\mathbf{S}\right)\mathbf{F}\hat{\mathbf{x}}\right)\right\|_2^2 + \lambda\left\|\hat{\mathbf{x}}\right\|_1,$$

  - Online training is still applicable since the loss function satisfies the three axioms.
- Based on the loss, we designed a two-stage DNN structure for joint CE training and nonlinearity approximation



Stage I: DNN-Based Channel Estimator   Stage II: Nonlinearity Approximation Module

Simulation for Unknown Nonlinearity

Default Simulation Setup:

- N=64 antennas, pilot length M=20, $V_{tx} = V_{rx} = V = 1.5$, $p_{tx} = p_{rx} = 1$
- fully connected NN with 2 hidden layers, each with 640 neurons
- Nonlinearity module is odd order polynomial raised to the 5-th power
- Online training scheme: 10000 steps of weight updates (i.e., 10000 pilot measurements) at SNR = 30 dB.

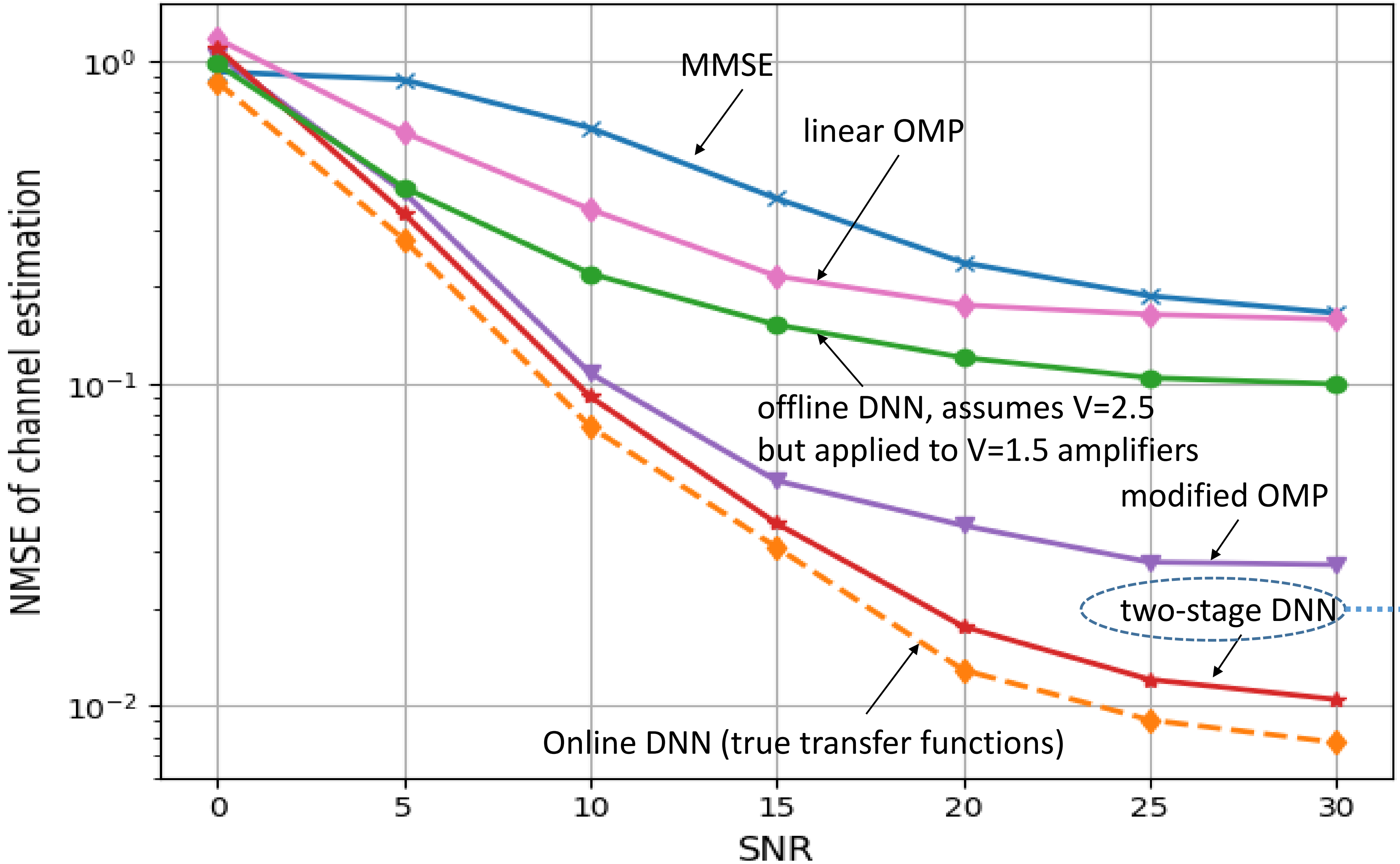
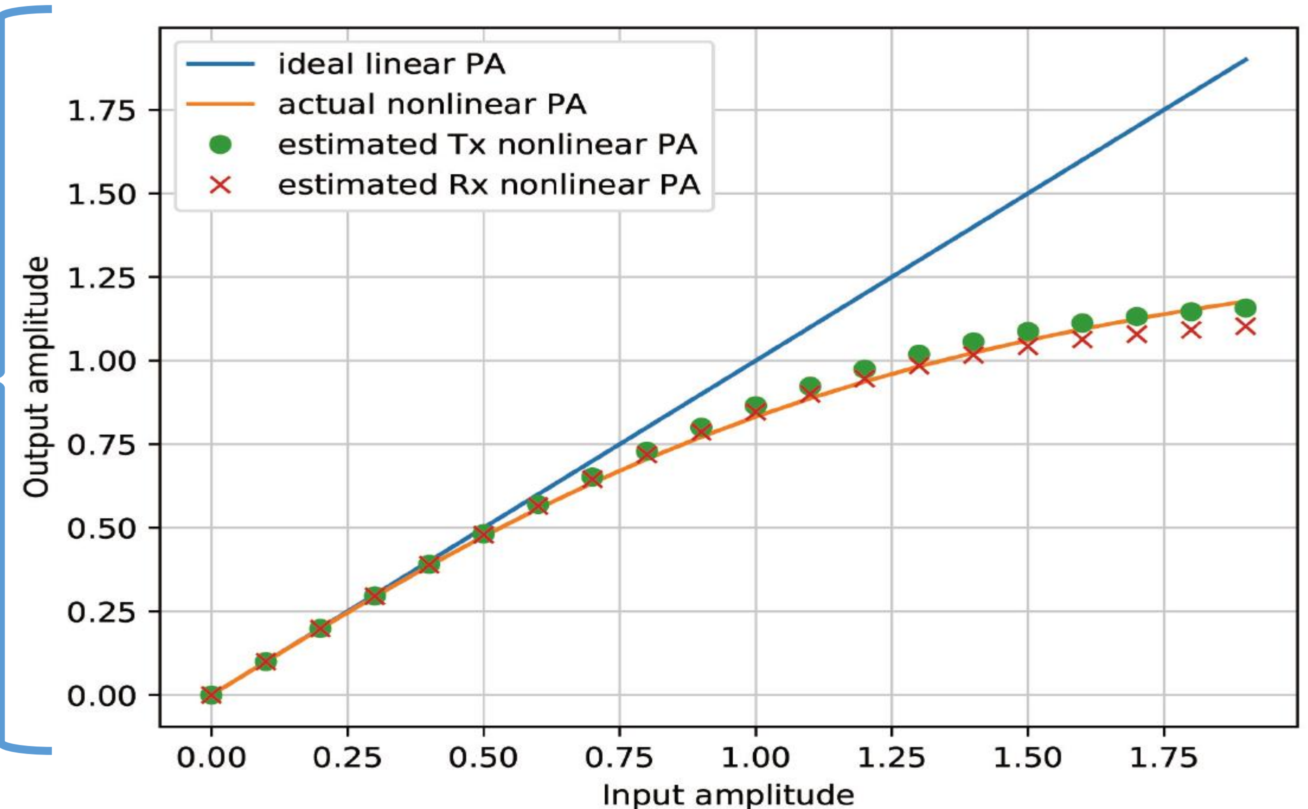


Fig. 12: CE NMSE versus SNR compared with baselines

- **Linear OMP** performs badly as it ignores nonlinearity
- **Modified OMP** compensates the nonlinear distortion with true transfer functions. Promisingly the **two-stage DNN** outperforms **modified OMP.**
- The **online DNN** (with known transfers) achieves slightly better performance than **two-stage DNN**, meaning that the nonlinear transfer functions are accurately approximated by the two-stage structure
- **Offline DNN** is trained on “pilot-channel” labels generated from the $V = 2.5$ amplifiers but applied to the system with $V = 1.5$ nonlinearity. The offline-DNN is prone to nonlinear model mismatch problem as well.

Verify the approximated nonlinear transfers with the actual ones



Extension to Limited Feedback MU-MIMO Systems

X. Zheng and V. Lau, "Federated Online Deep Learning for CSIT and CSIR Estimation of FDD Multi-User Massive MIMO Systems," in IEEE Transactions on Signal Processing, vol. 70, pp. 2253-2266, 2022.

Multi-User Massive MIMO

Massive MU-MIMO system:

- One BS with N antennas, K single-antenna UEs
- The received signal at the k-th user: $\mathbf{y}_k = \mathbf{S}\mathbf{h}_k + \mathbf{n}_k$
- To leverage multiplexing gain of MIMO,
  - UEs need to know the CSIR
  - BS needs to know the CSIT
- The CSITs for the BS need to be feedback by the UEs, which induces large feedback overhead in massive MIMO

Conventional CS-based solution

- $\mathbf{h}_k$ are estimated at the K UEs and are fed back to the BS.
- This approach will not be able to explore the **common sparsity structure** in MU channels, because user k only have pilot measurement for $\mathbf{h}_k$

We should explore the common sparsity structure in the MU-MIMO channels to reduce the pilot & feedback overhead.

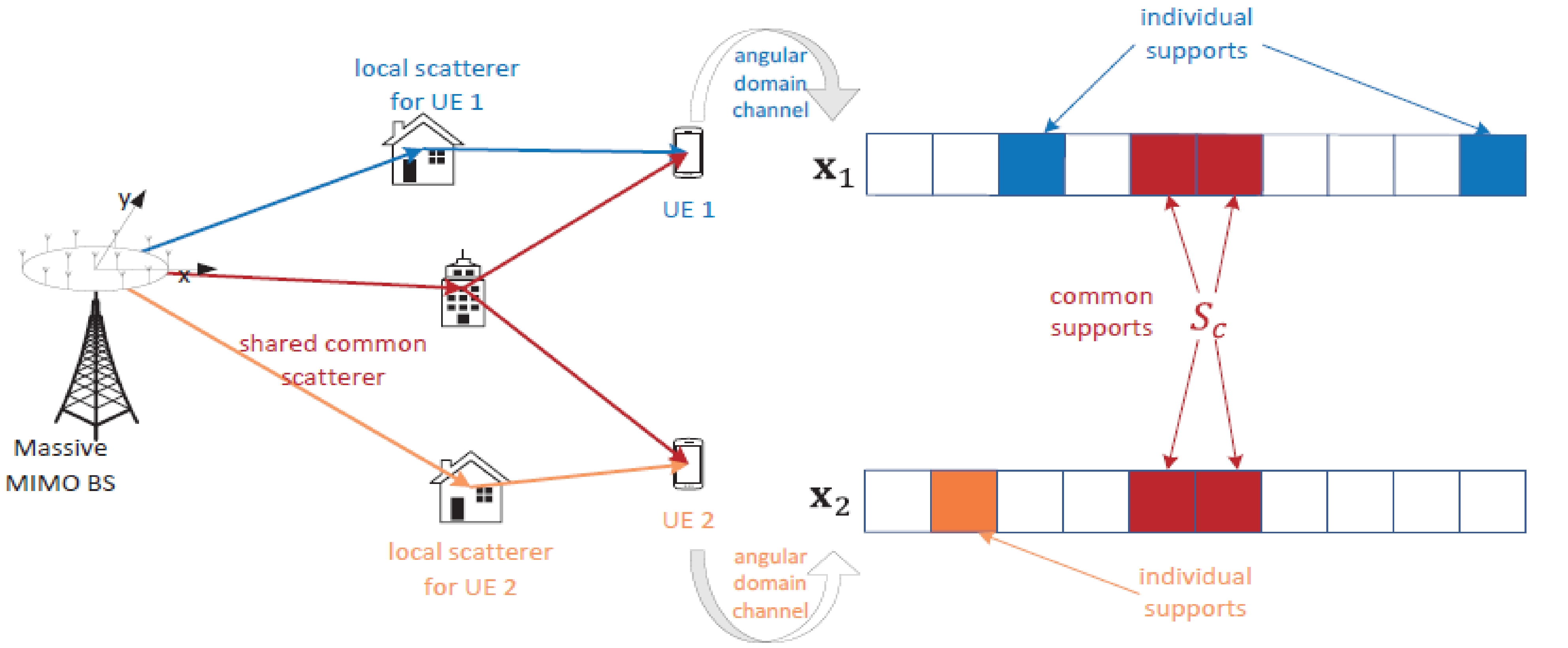


Fig. 14: Massive MU-MIMO system with structural common sparsity

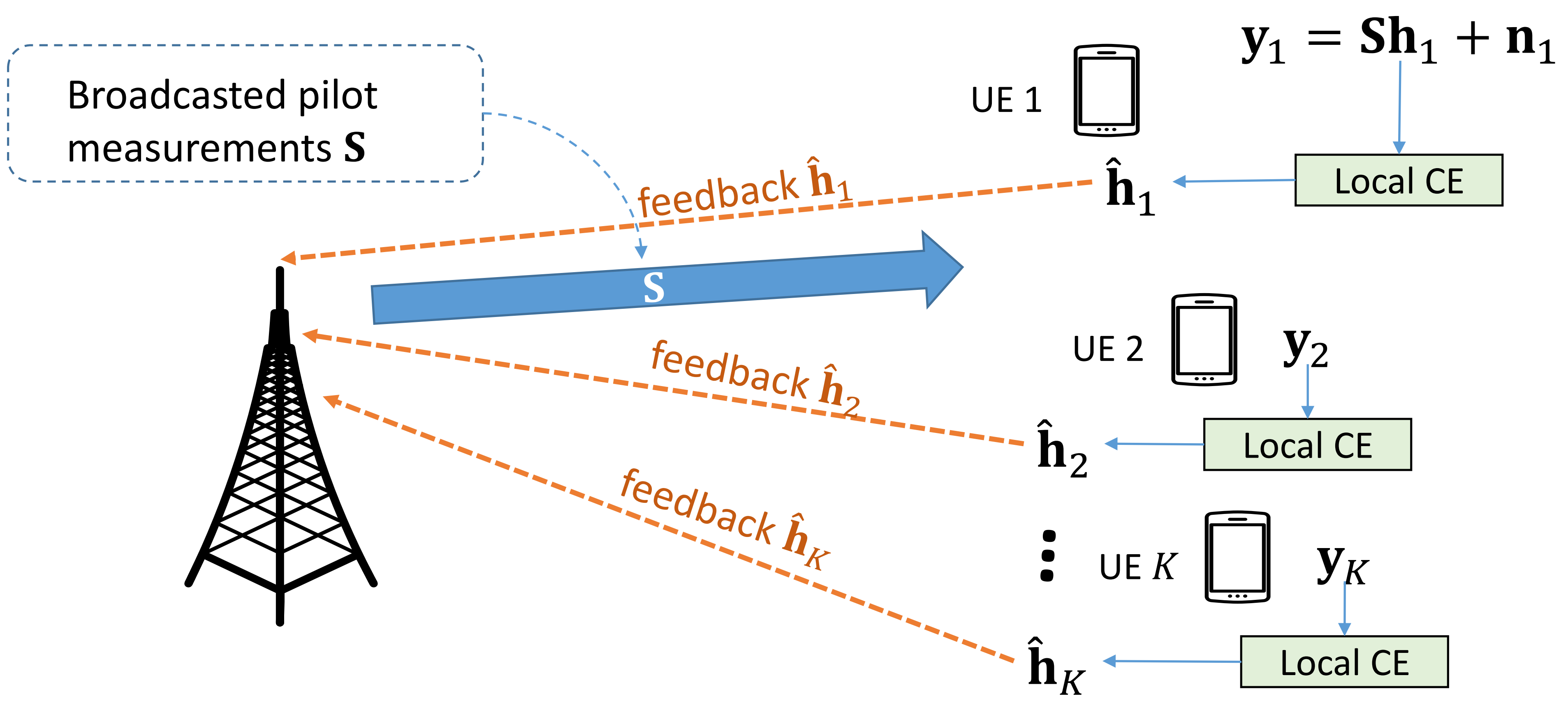


Fig. 15: Conventional channel estimation and CSIT feedback in MU-MIMO system.

Common Sparsity in MU-MIMO

Common Sparsity in MU-MIMO Channels:

- *<u>Individual Sparsity</u>*: the channel of each UE has a sparse representation in angular domain: $\mathbf{h}_k = \mathbf{F}\mathbf{x}_k$, each $\mathbf{x}_k$ is sparse.
- *<u>Partial Common Sparsity</u>* among all users: The UEs share some common scatterers, thus the channels of different UEs share a partial common support set.

Verification of Common Sparsity via COST2100 channel model

- 2.6-GHz NLoS semi-urban environments with closely spaced users randomly clustered in a 50 m x 50 m target area
- 20 users randomly clustered in a 5 m x 5 m square
- A support is identified with significant raise-over-thermal (20 dB)
- Common support is identified if it is a support for all users

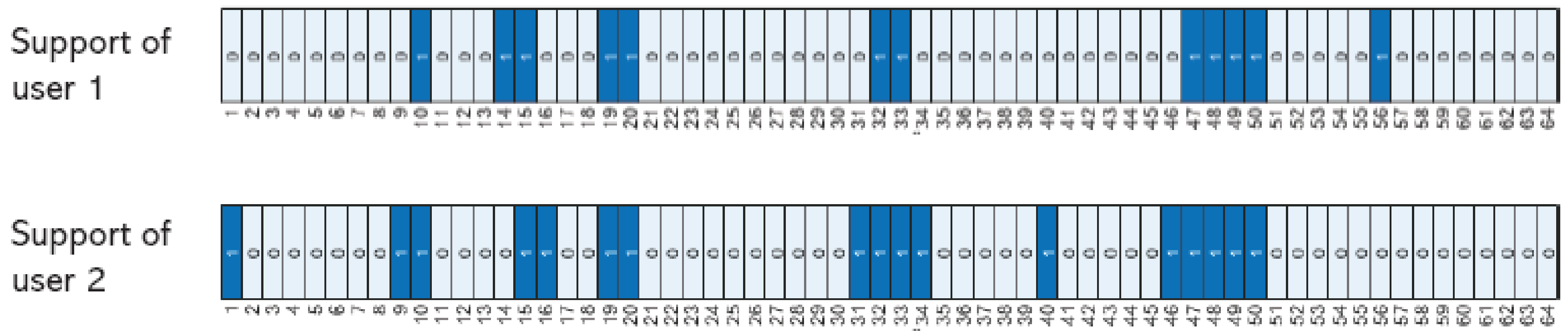


Fig. 16: Channel supports of two random users. We can see there is large portion of overlap in the supports of the two users.

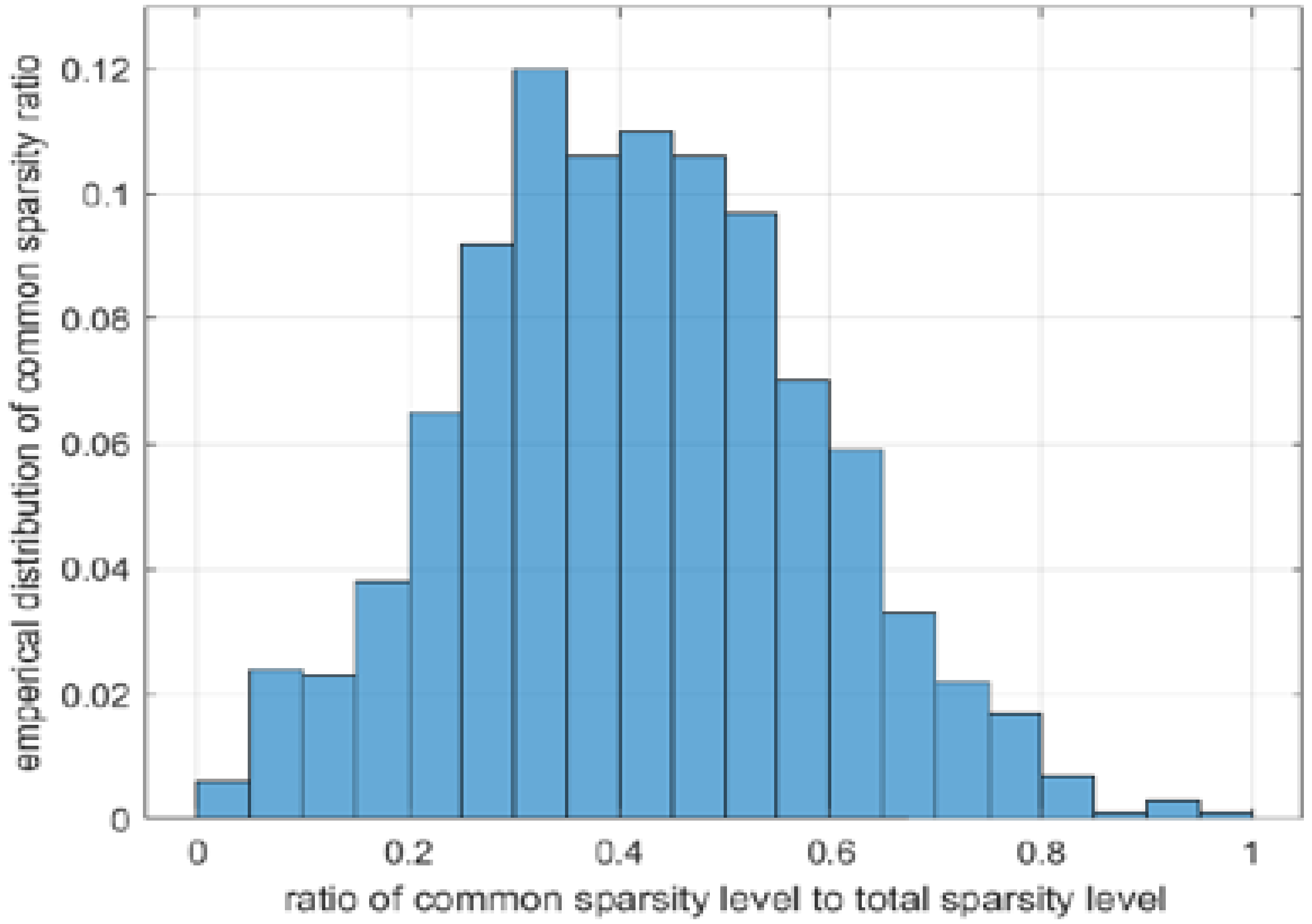
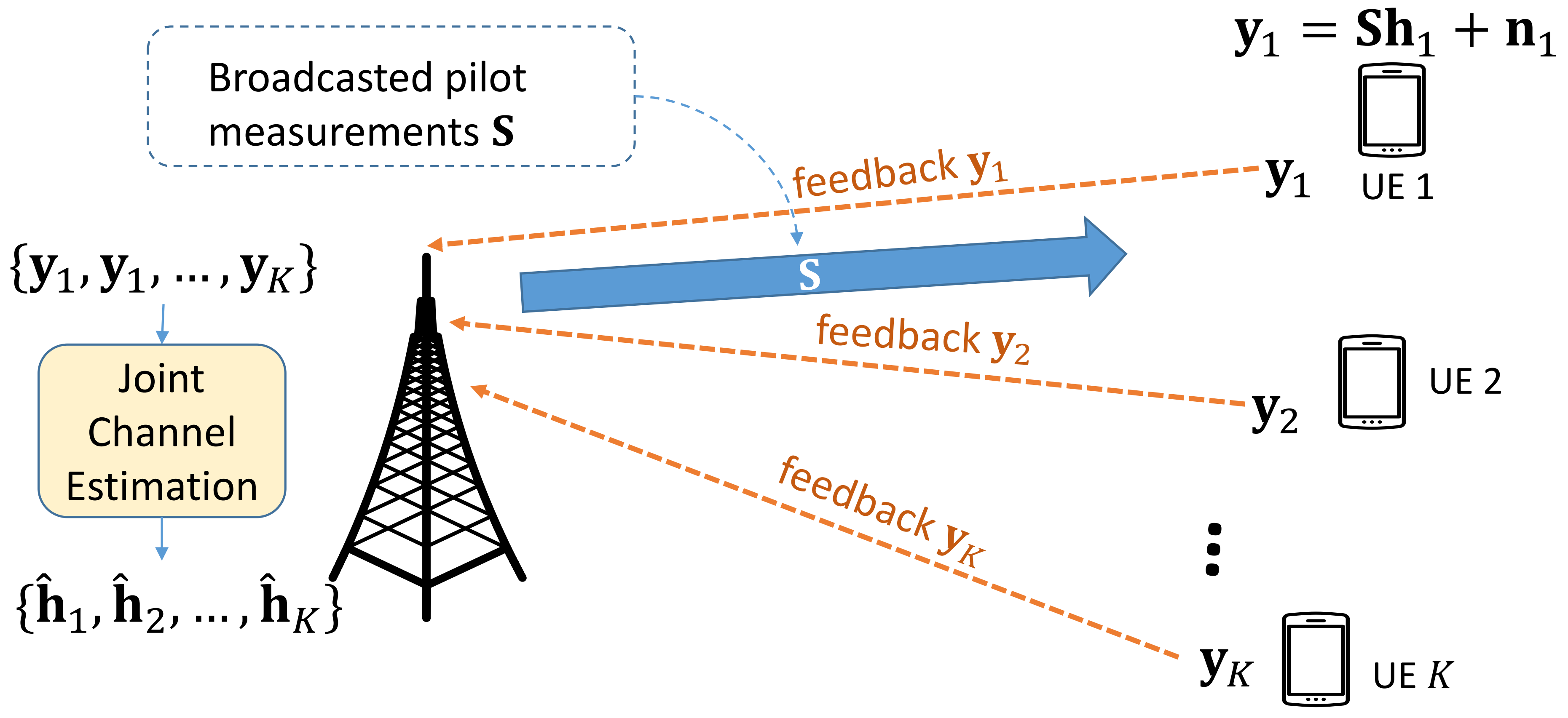


Fig 17: Histogram of ratio of common sparsity level. Typical common sparsity ratio lies in the range of 30% ~ 50%.

Exploiting Common Sparsity

Distributed MU-MIMO Channel Estimation Scheme [8]:

1. The BS broadcast compressed training pilots to UEs
2. The UEs feedback compressed pilot measurements to the BS
3. The BS recovers the channels via joint-orthogonal matching pursuit algorithm exploiting common sparsity



[8] X. Rao and V. K. N. Lau, "Distributed Compressive CSIT Estimation and Feedback for FDD Multi-User Massive MIMO Systems," in IEEE Transactions on Signal Processing, 2014.

This approach can facilitate CSIT at the BS because the CSITs are jointly recovered at the BS.
Problems:

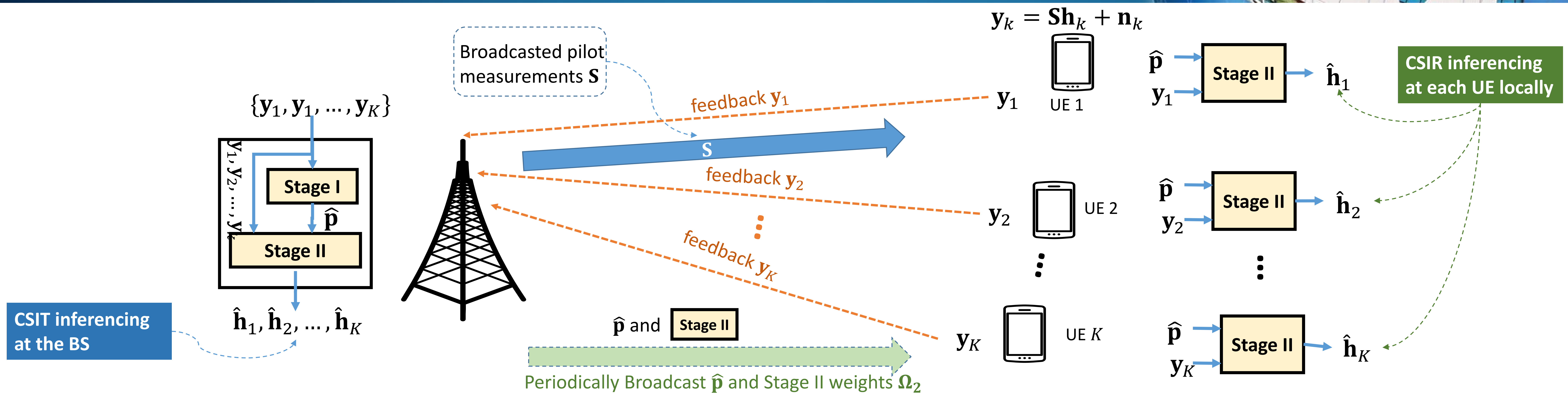
1. The **CSIR estimation** of the UE cannot be achieved because the compressed pilot measurement observed locally will NOT be sufficient for individual UE to estimate the per-link $\mathbf{h}_k$
2. The J-OMP algorithm is iterative and has **high computational complexity** → channel estimation cannot be done in real time for massive MIMO

Desire for Online DNN of the MU Case

We also need an online DNN-based solution for MU-case such that

- Channel inferencing is fast
- Online training that utilizes MU structural sparsity
- Facilitate both CSIT and CSIR estimation at the same time

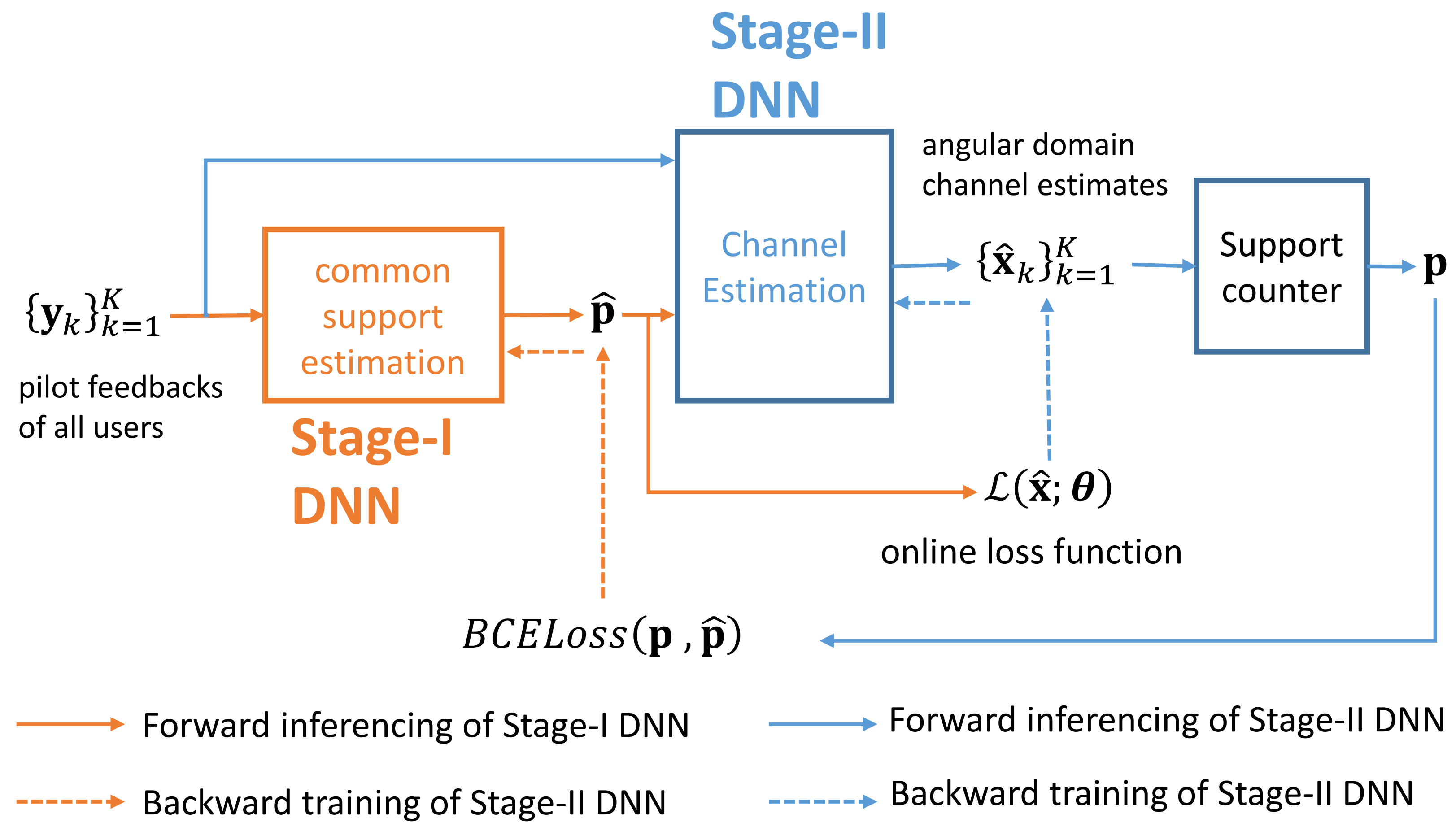
Enabling CSIT and CSIR Estimation



To achieve the purpose of CSIT and CSIR estimation:

- **We propose a two-tier DNN structure**
  - Stage-I jointly estimate the common supports $\hat{\mathbf{p}}$ from pilot feedbacks $\{\mathbf{y}_1, \mathbf{y}_1, \dots, \mathbf{y}_K\}$ of all UEs
  - Stage-II estimates the channel $\hat{\mathbf{h}}_k$ for each user utilizing $\hat{\mathbf{p}}$ and $\mathbf{y}_k$ (all users share the same Stage-II weights)
- **For CSIT estimation:**
  - The BS implements forward propagation of Stage-I and Stage-II to estimate the CSIT of all users
- **For CSIR estimation (federated learning):**
  - The BS will periodically broadcast $\hat{\mathbf{p}}$ and Stage-II DNN weights to the UEs.
  - The UEs perform CSIR inferencing locally based on downloaded Stage-II weights and its own pilot measurements.

Detailed Two-Tier DNN Structure



Challenge: Design of loss

- Enable online training
- Leverages individual channel sparsity
- Leverages the partial common sparse structure among users

Stage-I : DNN-based common support estimator (with weigths Ω_1)

- Input: pilot feedbacks from all users $\{\mathbf{y}_k\}_{k=1}^K$
- Output: common support $\hat{\boldsymbol{p}} \in [0,1]^N$
- The activation function of the output layer is the sigmoid function, that outputs a soft probability.
- Training of Stage-I is a multi-class multi-label classification problem using the BCE loss

$$\mathrm{BCE}\left(\hat{\mathbf{p}},\mathbf{p}\right)=\frac{1}{N}\sum_{n=1}^{N}\left(p_n\log\hat{p}_n+(1-p_n)\log\left(1-\hat{p}_n\right)\right)$$

Stage-II : DNN-based Channel Estimator (with weights Ω_2)

- Input: pilot feedbacks and $\hat{\boldsymbol{p}}$ estimated by Stage-I
- Output: channel of all users $\{\hat{\mathbf{x}}_k\}_{k=1}^K$
- A common support counter calculates p based on channel estimates and use it as training label for Stage-I training
- Stage-II DNN training is trying to find a set of weights that minimizes some loss function for CE

$$\mathcal{L}\left(\mathrm{DNN}_{\boldsymbol{\Omega}_2}\left(\left[\tilde{\mathbf{y}}_k;\hat{\mathbf{p}}\right]\right);\boldsymbol{\theta}\right)$$

Online Loss Design Exploiting Common Sparsity

- Similarly, for online training of the channel estimator, we need an online loss function that (i) does not need true channel, (ii) leverages the common support p and (iii) measures the error with the true channel.
- We proposed a weighted l1-norm regularized loss function utilizing the common support estimate $\hat{\mathbf{p}}$.

$$\|\mathbf{x}\|_{\mathbf{w},1} = \sum_{n=1}^{N} w_n |x_n|$$

$$\mathcal{L}(\hat{\mathbf{x}};\boldsymbol{\theta}) = \frac{1}{2}\|\mathbf{y}-\mathbf{A}\hat{\mathbf{x}}\| + \lambda \|\hat{\mathbf{x}}\|_{\mathbf{w},1}$$

$$\text{with } w_n = 1-\hat{p}_n, \quad \boldsymbol{\theta} = \{\mathbf{y}, \hat{\mathbf{p}}, \mathbf{S}, \mathbf{F}\}$$

- The main idea: set w_n smaller if x_n is more likely to be large, such that x_n is less penalized.

Lemma

(Consistency of the Weighted l_1-Norm Regularized Loss Function) Assume the angular domain channel x *is s-sparse. Suppose the overall measurement matrix* $\mathrm{A} = \mathrm{SF}$ *satisfies the RIP property and the RIP constant* δ_{ts} *satisfies*

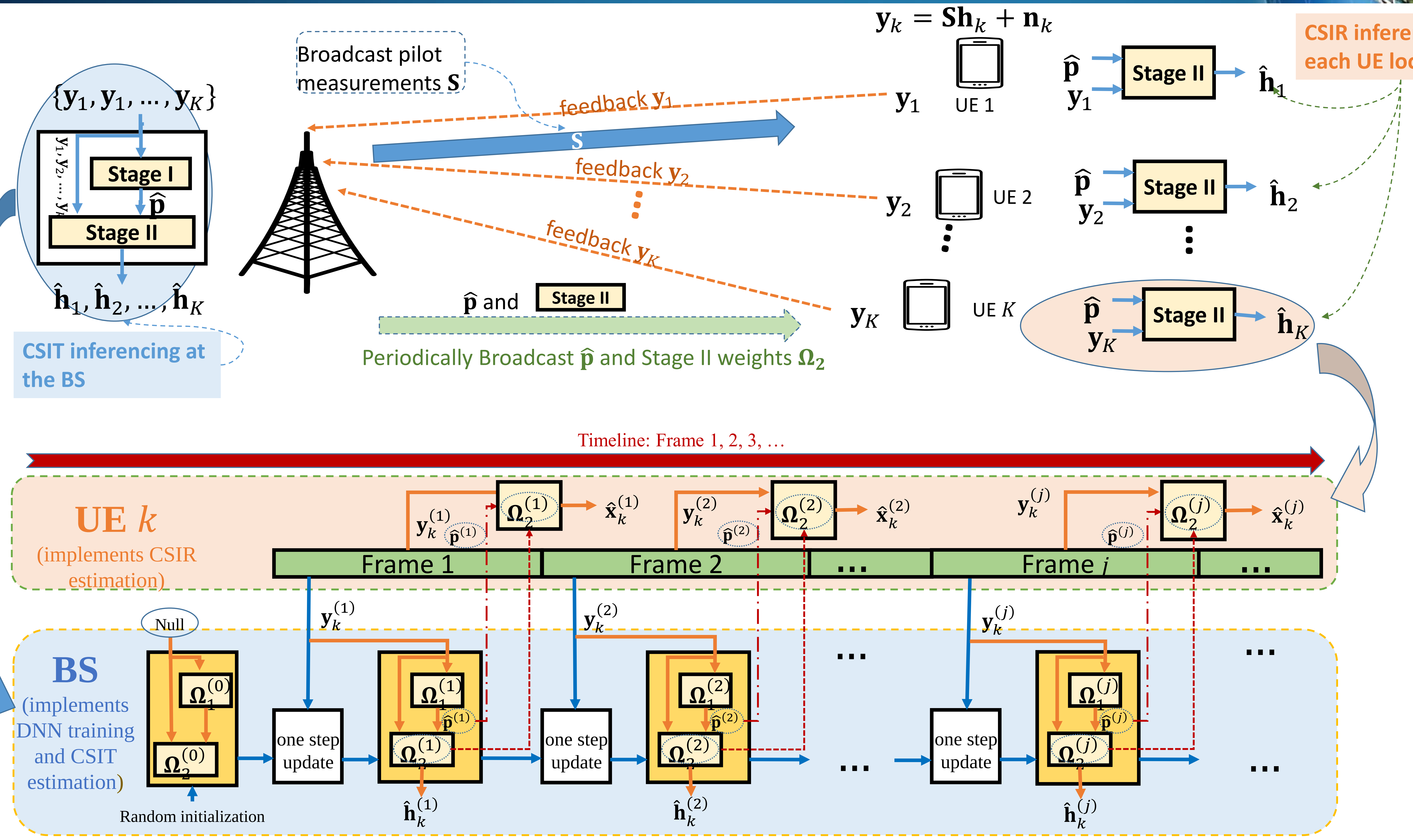
$$\delta_{ts} < \sqrt{\frac{t-d}{t-d+\theta^2}} \tag{6}$$

for some $t > d$*. Let the regularizer be chosen as* $\lambda = \sigma_n/\sqrt{s}$*. Then, the minimizer* x^* *of the loss function, satisfies*

$$\|\mathrm{x}^* - \mathrm{x}\|_2 \leq \sigma_n \epsilon(\hat{\mathrm{p}}) = 2\sigma_n \left(\frac{\beta_1 u (1+\theta s) + (r\beta_2 + u\sqrt{rs})}{(u-\theta\beta_2)(\theta\beta_1+\sqrt{r})}^{-1} \right) \tag{7}$$

w.h.p. at least $1-e^{-r/10}$ *with* $r \geq M$ *and* $u = \sqrt{t-d}$*, where* θ *and* d *are constants depending on accuracy of* $\hat{\mathrm{p}}$*, while* β_1 *and* β_2 *are some constants depending on* δ_{ts}*.*

Federated Online Training



At the UEs:

1. Feedback received pilot measurements
2. Periodically download $\hat{\mathbf{p}}$ and Stage-II weights
3. CSIR inferencing

At the BS:

1. Broadcasts pilots
2. Update Stage-I and Stage-II weights online
3. CSIT inferencing
4. Periodically broadcast $\hat{\mathbf{p}}$ and Stage-II weights

Simulation

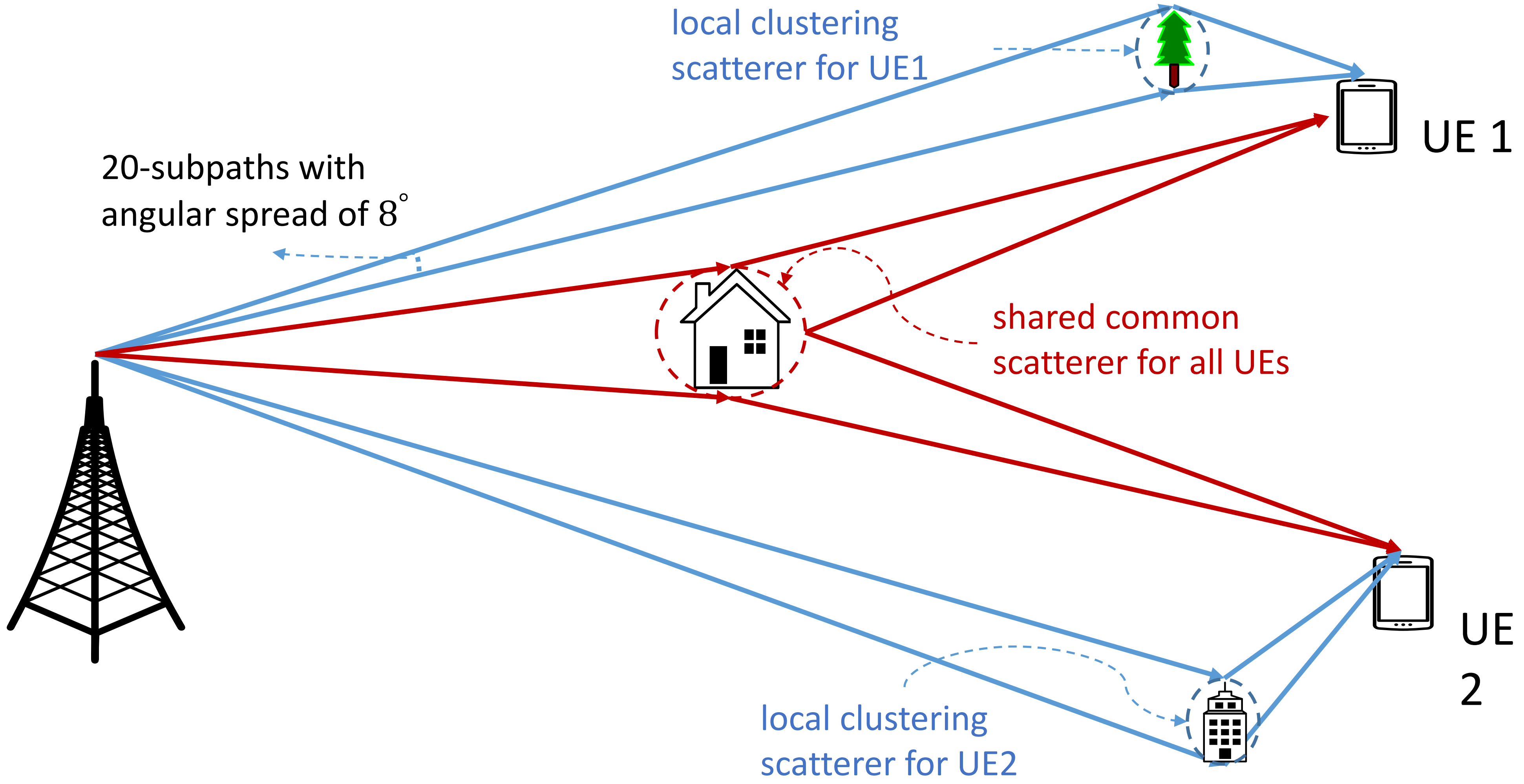


Fig. 18: MU-MIMO channel for a $K = 2$ user case. There is a common clustering scatterer for all users and one local scatterer for each user.

Default Setting:

- BS equipped with ULA of $N = 64$ antennas, $K = 20$ UEs with single antenna, pilot length $M = 40$.
- 3GPP SCM channel model: 28 GHz carrier frequency system in an urban macro propagation environment. $C = 2$ clusters of scatterers, each has 20 significant paths with angular spread of 8°. One of the cluster is a common scatterer.
- DNN structure: both stages has 2 hidden layers with 100~300 neurons each.

Computation Time Saving

number of received pilot feedbacks per time slot	20	40	100
Tow-tier DNN one step of CE inferencing	0.1206 ms	0.2137 ms	0.5161 ms
JOMP	122.994 ms	236.92 ms	654.08 ms
MMSE	46.236 ms	94.83 ms	234.47 ms

Fig. 19: CPU time of proposed solution compared with various baselines under different numbers of pilot feedbacks in each time slot.

- ➢ One step of channel inferencing is about **1000x** faster than JOMP [8] algorithm, and **300x** faster than MMSE algorithm.
- ➢ The fast inferencing enables real-time CE for MU-massive MIMO.

Gain from Common Sparsity

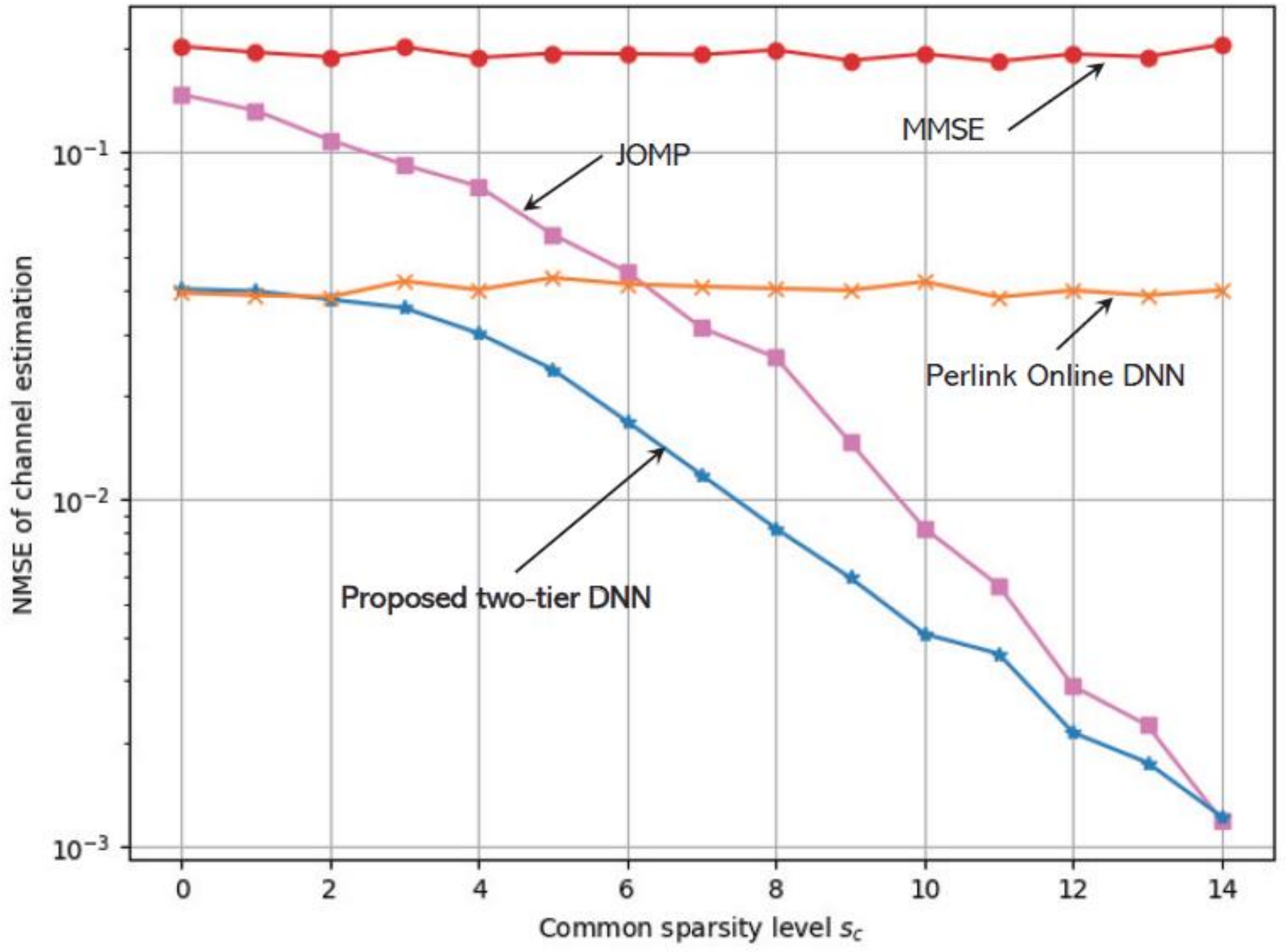


Fig. 20: CE NMSE v.s. common sparsity level at SNR = 30 dB and total sparsity = 14.

- In this experiment, the channel of each user has 14 support, and we vary the number of common support of each user
- The CE quality of the two-tier DNN gets better as the number of common supports increases, illustrating the benefits of more common support (i.e., hidden correlation among all users)
- In the case of no common support ($s_c = 0$), the two-tier DNN has similar performance with the per-link online DNN

CSIT & CSIR Tracking

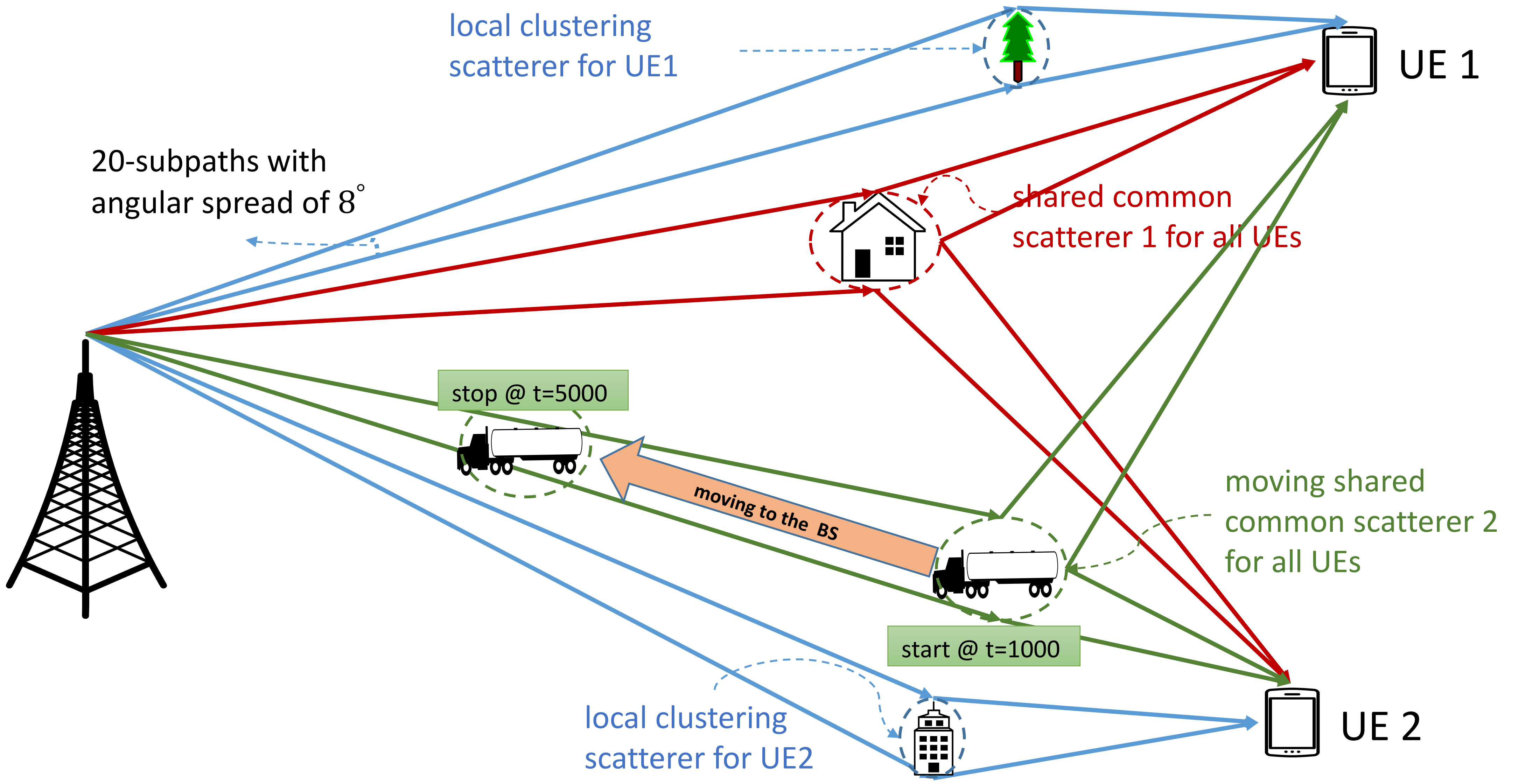


Fig 21. Time-varying propagation environment for a $K = 2$ system.

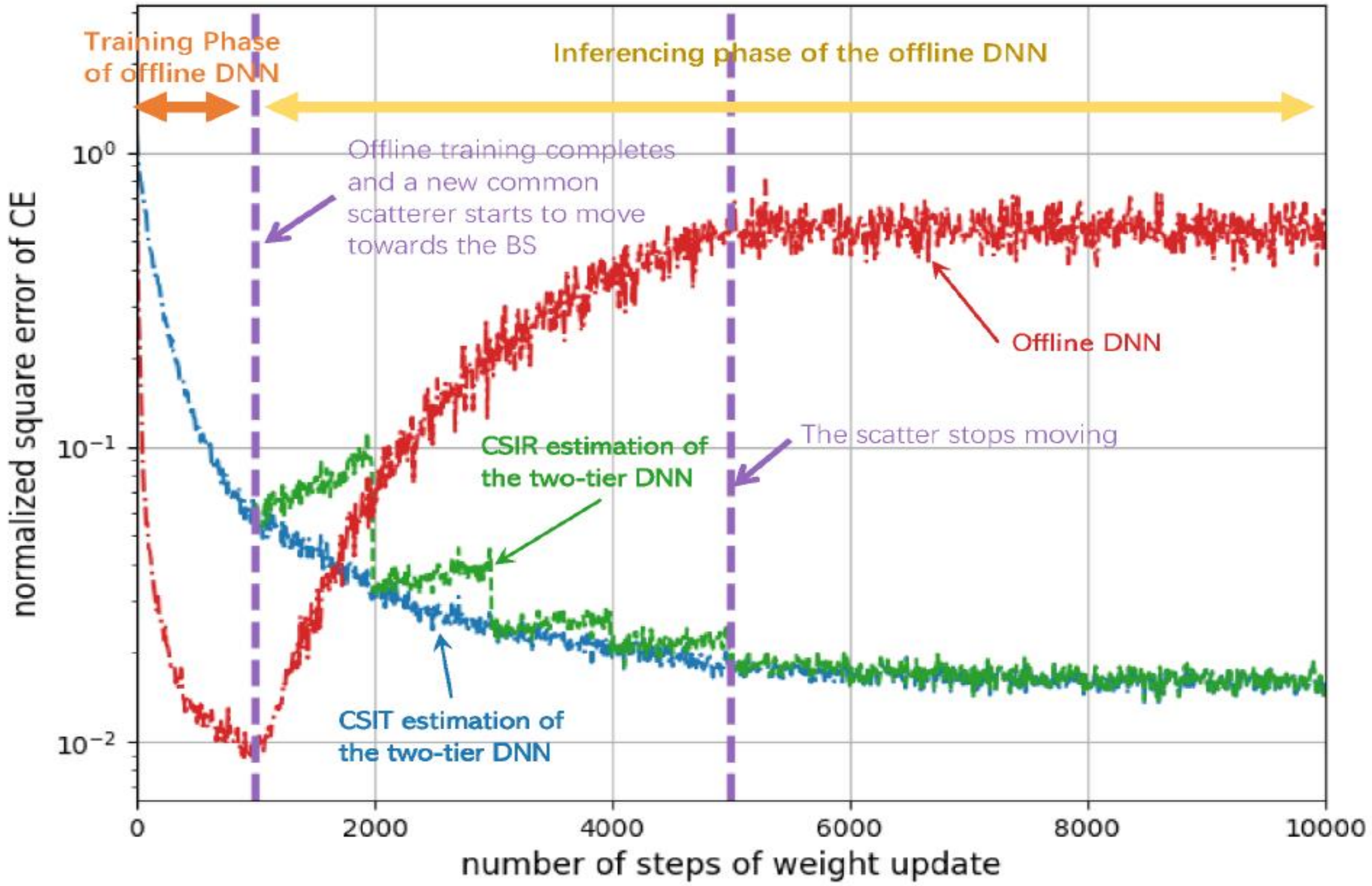


Fig 22. CSIT & CSIR Tracking of the two tire DNN compared with offline DNN.

- Propagation environment is static for the first 1000 steps. Then, a new common scatterer starts moving towards the BS and stops at 5000-th step.
- Offline DNN fixed weights after offline training & apply online, while online DNN continuously updates its weights based on real-time measurements.
- BS performs DNN training & CSIT estimation. It also broadcast Stage-II DNN weights to the UEs every T=1000 steps, so that the UEs can perform CSIR estimation

- ➢ Offline DNN performance degrade with time as its weights cannot adapt to the propagation environment
- ➢ Online DNN at the BS can keep track of CSIT channel model.
- ➢ At the UE, the CSIR estimation error converges to that of the CSIT estimation after about 5 Stage-II weights broadcasting periods.

Conclusions

Conclusions

Online Training Framework for massive MIMO CE:

- [**Online Loss Function**] propose 3 axioms for a legitimate online loss function - the channel estimator can be trained online based on real-time pilot measurements without need for channel labels.
  - example online loss functions that satisfy the 3 axioms and
  - exploiting channel sparsity for reduced pilot overhead
- [**Online Training Algorithm**] enables simultaneous training and inferencing.
  - the DNN weights can adapt to the time-varying channel model online (tracking ability)
  - robust to various model mismatches (channel model & nonlinear model & array geometry)
  - enjoying faster channel inferencing
  - ϵ-analysis of online training algorithm
- [**Extension to MU-MIMO**] enables CSIT & CSIR estimation with reduced pilot feedbacks
  - two-tier DNN structure exploiting partial common sparsity
  - federated online training enables UEs to utilize the common sparsity structure learned at the BS for CSIR estimation.

Thank You

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY