ChannelMAE: Self-Supervised Learning Assisted Online Adaptation of Neural Channel Estimators

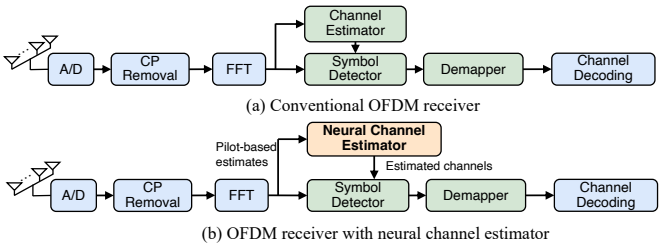
Tianxin Wang*, Yuanzhe Huang*, Xudong Wang†
*Shanghai Jiao Tong University, Shanghai, China
†Hong Kong University of Science and Technology Guangzhou, Guangzhou, China

***Abstract*—A pretrained neural channel estimator cannot generalize to all channel environments, necessitating online adaptation. Conventional methods demand ground-truth channel coefficients as supervised labels, but such labels are unavailable online. To this end, a self-supervised task is introduced on top of the original channel-estimation task to facilitate label-free adaptation of neural estimators. Specifically, this task randomly masks a fraction of resource elements in each received frame and reconstructs such masked parts. To enable effective reconstruction, the task input must incorporate two components: the unmasked parts and estimated data-symbols of masked parts. These estimated symbols are obtained via an online symbol-recovery mechanism, so no additional pilot overhead is incurred. To consolidate the self-supervised task with the original task, a two-branch masked auto-encoder (MAE) model called ChannelMAE is developed, with each branch dedicated to one task. The two branches share the same encoder but use separate decoders. During online adaptation, the encoder is updated by optimizing the self-supervised branch, which learns channel statistical features and shares them with the channel-estimation branch. Therefore, online channel-estimation accuracy is much improved. Extensive experiments show that ChannelMAE reduces channel-estimation error by up to 71.8% and 87.1% compared with the pretrained model and the state-of-the-art adaptation scheme, respectively.**

I. Introduction

Deep learning (DL)-based wireless physical layer design has attracted growing attention in recent years [1]. Particularly, the neural network (NN) based Orthogonal Frequency-Division Multiplexing (OFDM) channel estimator (i.e., *neural channel estimator*) is one of the most promising solutions for 6G intelligent receivers [2]. As shown in Fig. 1, a neural channel estimator only replaces the conventional channel estimator by an NN, which takes pilot-based channel estimates as inputs and predicts full-frame channel coefficients. It can significantly outperform traditional estimators in real-world scenarios without requiring explicit channel modeling or prior channel statistics [1], [3], [4]. As a common practice, a neural channel estimator is pretrained offline and then deployed online. However, it is impractical for an offline dataset to cover all possible channel conditions [5], so performance of the pretrained channel estimator can degrade in varying channel environments. Therefore, neural channel estimators must be adapted online.

Existing solutions to online adaptation of OFDM neural channel estimators [2], [6]–[8] still pose two challenges. First, conventional adaptation methods demand ground-truth channel coefficients as supervised labels, but such labels are unavailable online [6]. Some studies [2], [8] obtain approximated channel coefficients using prior channel statistics, but acquiring such prior knowledge is also unrealistic. Second, how to avoid additional pilot overhead is crucial. Existing work either doubles pilot overhead [2] or periodically populates an entire OFDM frame with pilots for online adaptation [9]–[11]. Such schemes also incur control signaling overhead, as signaling messages are required to trigger transmission of these additional pilots [2]. To the best of our knowledge, there still lacks an online learning framework that can fully resolve the above two challenges.



(a) Conventional OFDM receiver

(b) OFDM receiver with neural channel estimator

Fig. 1. Illustration of an OFDM neural channel estimator.

To this end, a *self-supervised task* that does not require true channel coefficients or additional pilots is introduced on top of the original channel-estimation task (i.e., *main task*). The key insight is that this new task must learn the latent features that benefit the main task. As a result, via online self-supervised learning (SSL) of this task alone, online features are captured and shared with the main task, thereby improving the main-task performance. This design rationale is inspired by test-time training (TTT) in machine learning [12]–[14]. While designing the channel-estimation task is straightforward, there still exist three challenges: 1) how to design an appropriate self-supervised task (also named SSL task) that aligns with the above design principles; 2) how to design a model architecture to consolidate the SSL task with the original task; 3) how to design an effective and efficient training strategy. They are addressed as follows.

First, the SSL task is designed as first randomly masking a large fraction of resource elements in each received frame and then reconstructing those masked parts, namely *masked reconstruction* [15]. The reason for this design lies in the fact that received signals inherently carry information of channel coefficients. Therefore, reconstructing different masked parts of received signals implicitly learns channel statistical fea-

Corresponding author: Xudong Wang (Email: wxudong@ieee.org).

tures in both time and frequency domains, and such features are also essential to the channel-estimation task. To enable effective reconstruction, this task must also incorporate the estimated data symbols as an input component besides the unmasked parts of the frame. These estimated symbols are obtained via an online symbol-recovery mechanism, where data-symbols are recovered through symbol detection online. Thus, no additional pilot overhead is incurred. Note that the recovered symbols cannot be used as pseudo-labels to back-propagate through the whole receiver for training, because the traditionally-designed detection modules can be non-differentiable [16]. In addition, the above task design results in high correlation between the SSL and main tasks, because channel estimation can be interpreted as a special case of masked reconstruction: it reconstructs channel coefficients from the noise-corrupted local observation at pre-known pilot positions. Such correlation leads to synergetic learning of the two tasks.

Second, to consolidate the SSL task with the original task, a two-branch masked auto-encoder (MAE) model architecture called *ChannelMAE* is developed, with each branch dedicated to one task. The two branches share a feature *encoder* but have their task-specific *decoders*. The encoder is designed as a lightweight attention-based transformer [15], which produces low-dimensional latent representations from the input. Via the self-attention mechanism, it implicitly captures channel statistical features with high computation efficiency [17]. Both decoders have fully-convolutional architectures but with different model sizes, performing reconstruction from the latent representations.

Third, a two-phase training strategy is designed for ChannelMAE. In the offline pretraining phase, all model parameters are pretrained by optimizing the main and SSL branches together. In the online adaptation phase, only the shared encoder is updated online by optimizing the SSL branch. The online-channel features learned by the encoder are also used in the main branch, thereby improving online channel-estimation accuracy. In addition, it is crucial to reduce memory footprint during adaptation. Thus, batch-wise online learning is enforced, where each adaptation step is conducted over a batch of online streaming samples that is immediately purged after use. To enrich data quantity and diversity, each online batch is further augmented by masking the same received frame with different random masks when optimizing the SSL task.

With the above three key designs, our designed two-task learning framework holds several distinct features compared with the existing TTT framework [12]: 1) TTT sets classification as its main task, whereas this paper is focused on a regression problem; 2) TTT constructs a single-input multi-task problem by letting two tasks process the same input, whereas in our case the two tasks can have diverse inputs, resulting in a multi-input multi-task problem. 3) TTT performs per-sample adaptation before inference, while ChannelMAE enforces per-batch adaptation on a relatively larger timescale after inference of this batch, which is better suited to wireless communication applications.

The main contributions of this paper are summarized as follows:

- An SSL task is introduced on top of the original channel-estimation task to facilitate label-free adaptation of neural channel estimators. It is designed as reconstructing the randomly-masked portions of received radio frames. An online symbol-recovery mechanism is designed to provide estimated data-symbols as the task input component, without incurring extra pilot overhead.
- A two-branch MAE model architecture named ChannelMAE is designed to consolidate the SSL task with the channel-estimation task. With each branch dedicated to one task, the two branches share a feature encoder but have their task-specific decoders.
- An offline-online two-phase training strategy is designed for ChannelMAE. Particularly in the online adaptation phase, the shared encoder is adapted by optimizing the SSL branch with batch-wise online learning. The adapted encoder improves online channel-estimation performance by implicitly capturing channel statistical features.
- Extensive experiments are carried out to validate key parameters and mechanisms of ChannelMAE. Performance results show that ChannelMAE significantly outperforms the state-of-the-art adaptation schemes.

The rest of this paper is organized as follows. Related work is presented in Section II, while the system model is provided in Section III. The overview of two-task framework is stated in Section IV, followed by the key designs elaborated in Section V. Performance evaluation is carried out in Section VI, and the paper is concluded in Section VII.

II. Related Work

Neural channel estimators play a critical role in future wireless communications. Its objective is to learn the non-linear mapping from channel estimates at pilot positions to channel coefficients of an entire frame based on channel datasets. Such a data-driven channel estimator is first proposed in [4], [18], where ChannelNet consisting of the super-resolution and restoration convolutional neural networks (CNN) is designed to interpolate and denoise channel estimates. Deep residual learning is introduced to OFDM channel estimation in [19]. To improve model generalization ability and computational efficiency, CNN is further replaced by the attention-based transformer in OFDM channel estimation [2], [17], [20], [21]. An end-to-end transformer model is designed in [21], but it incurs high computation cost by applying attention on the entire frame. To reduce computation cost, a hybrid architecture is proposed in [2], [20], consisting of the transformer, the fully-connected up-sampling layer, and the CNN. Distinct from [2], [20], we design an MAE for the channel-estimation task and it features a transformer-based encoder and a ResNet-based decoder, which significantly improves parameter efficiency by eliminating fully-connected layers.

The above research is focused on training offline neural channel estimators with ground-truth channel coefficients as

supervised labels. Online adaptation without such labels attracts growing attention. In [2], a new type of pilot signals, termed label pilot, is designed to assist online adaptation, which incurs substantial extra pilot overhead. Instead of predicting the full-frame channel coefficients, the model HA03 inputs channel estimates at standard-pilot positions and outputs channel coefficients at label-pilot positions, and the results are then bilinearly interpolated to the full-frame channel responses. To acquire accurate coefficients at label-pilot positions as training labels, either transmitted power of label pilot signals is increased or prior channel statistics are assumed. The approach in [8] adopts a similar method. Beyond these studies, a denoising network (DnCNN) is trained in [7], using signals with additional synthetic noise as inputs and original signals as outputs. The trained model is directly used for channel estimation. However, the method is limited to narrow-band channels, and applying it to OFDM channels requires bilinear interpolation of pilot estimates as the first step.

Compared with existing methods, our scheme introduces a two-task adaptation framework and designs a two-branch MAE architecture, which is the first to realize online adaptation without ground-truth channel coefficients, prior channel statistics, or extra pilot overhead.

III. System Model

The Single-Input Single-Output (SISO) OFDM communication is considered with one receive antenna and one transmit antenna. One OFDM frame spans a transmission time interval (TTI) and includes N_{s} OFDM symbols and N_{f} subcarriers. Each resource element (RE) has one symbol time and one subcarrier. Let $\mathbf{Y}, \mathbf{H}, \mathbf{N} \in \mathbb{C}^{N_{\mathrm{s}} \times N_{\mathrm{f}}}$ be the received signals, the channel coefficients, the zero-mean additive white Gaussian noise of one OFDM frame, respectively. The transmit signal matrix is $\mathbf{X} \in \mathbb{C}^{N_{\mathrm{s}} \times N_{\mathrm{f}}}$. The frequency-domain received signal at the receive antenna during one TTI is $\mathbf{Y} = \mathbf{H} \odot \mathbf{X} + \mathbf{N}$, where $\mathbf{N} \sim \mathcal{CN}(0, \sigma_n^2)$ and σ_n is the standard deviation of Gaussian noise. Let N_{sp} be the number of OFDM symbols carrying pilots (i.e., pilot symbol time) within one frame. At each pilot symbol time, N_{fp} subcarriers carry pilots (i.e., pilot subcarriers). Thus, in total $N_{\mathrm{sp}} N_{\mathrm{fp}}$ REs are occupied with pilot symbols. With transmitted pilots $\mathbf{X}_{\mathrm{p}} \in \mathbb{C}^{N_{\mathrm{sp}} \times N_{\mathrm{fp}}}$, received pilots $\mathbf{Y}_{\mathrm{p}} \in \mathbb{C}^{N_{\mathrm{sp}} \times N_{\mathrm{fp}}}$ are obtained. Pilot-based Least-Squares (LS) estimate $\hat{\mathbf{H}}_{\mathrm{p}} \in \mathbb{C}^{N_{\mathrm{sp}} \times N_{\mathrm{fp}}}$ is computed as:

$$\hat{\mathbf{H}}_{\mathrm{p}} = \underset{H_{\mathrm{p}} \in \mathbb{C}^{N_{\mathrm{sp}} \times N_{\mathrm{fp}}}}{\arg\min} \|\mathbf{Y}_{\mathrm{p}} - \mathbf{H}_{\mathrm{p}} \odot \mathbf{X}_{\mathrm{p}}\|^2 = \frac{\mathbf{Y}_{\mathrm{p}}}{\mathbf{X}_{\mathrm{p}}}, \tag{1}$$

where the division between $\mathbf{Y}_{\mathrm{p}}$ and $\mathbf{X}_{\mathrm{p}}$ is element-wise.

Among conventional methods, linear Minimum Mean-Squared Error (LMMSE) channel estimator is widely acknowledged as a strong baseline, which estimates channel coefficients by linearly filtering $\hat{\mathbf{H}}_{\mathrm{p}}$ with channel cross- and auto-correlation matrices [22]. But it is extremely challenging to obtain accurate channel correlations in practice. By contrast, neural channel estimators aim to learn a non-linear mapping from $\hat{\mathbf{H}}_{\mathrm{p}}$ to $\mathbf{H}$ using NNs in a purely data-driven fashion, without requiring prior channel statistics. This resolves the challenge of LMMSE estimators.

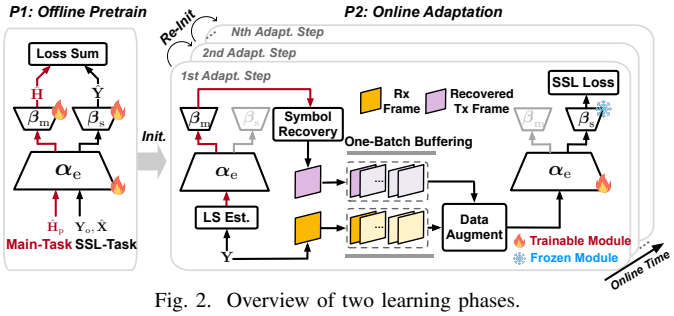


Fig. 2. Overview of two learning phases.

IV. Overview of Two-Task Learning Framework

A. Task Formulation

As depicted in Fig. 2, the SSL and channel-estimation tasks (i.e., main task) are consolidated to a two-branch MAE model ChannelMAE, wherein they: 1) share one common feature encoder denoted by $\boldsymbol{\alpha}_{\mathrm{e}}$ and have their task-specific decoders (main-task decoder $\boldsymbol{\beta}_{\mathrm{m}}$ and SSL-task decoder $\boldsymbol{\beta}_{\mathrm{s}}$) and 2) they have different input-output pairs. We denote the main branch as $\boldsymbol{\theta}_{\mathrm{m}} = \{\boldsymbol{\alpha}_{\mathrm{e}}, \boldsymbol{\beta}_{\mathrm{m}}\}$ and the SSL branch as $\boldsymbol{\theta}_{\mathrm{s}} = \{\boldsymbol{\alpha}_{\mathrm{e}}, \boldsymbol{\beta}_{\mathrm{s}}\}$.

1) Main-Task Formulation: The main branch takes pilot-based LS estimates $\hat{\mathbf{H}}_{\mathrm{p}}$ as the input and outputs full-frame estimated channel coefficients $\hat{\mathbf{H}}$. By denoting the mapping function parameterized by $\boldsymbol{\theta}_{\mathrm{m}}$ as $\varphi_{\boldsymbol{\theta}_{\mathrm{m}}}(\cdot)$, the main task is formulated as

$$\hat{\mathbf{H}} = \varphi_{\boldsymbol{\theta}_{\mathrm{m}}}(\hat{\mathbf{H}}_{\mathrm{p}}). \tag{2}$$

2) SSL-Task Formulation: The SSL task is formulated as reconstructing received frame $\mathbf{Y}$ from its randomly-masked version. Due to the randomness of transmitted data-symbols $\mathbf{X}$, directly reconstructing $\mathbf{Y}$ is infeasible. Thus, the estimated data-symbols $\hat{\mathbf{X}}$ must be incorporated into the task input as prior information to enable effective reconstruction. $\hat{\mathbf{X}}$ is obtained via an online symbol-recovery mechanism, as elaborated in Section V-C.

Let $\mathbf{M}_{\mathrm{r}} \in \mathbb{R}^{N_{\mathrm{s}} \times N_{\mathrm{f}}}$ denote a random binary mask, where 1 indicates an unmasked position and 0 indicates a masked position. Following the masked-reconstruction paradigm, let $\mathbf{Y}_{\mathrm{o}} \in \mathbb{C}^{N_{\mathrm{s}} \times N_{\mathrm{f}}}$ be the masked frame after applying $\mathbf{M}_{\mathrm{r}}$, i.e., $\mathbf{Y}_{\mathrm{o}} = \mathbf{Y} \odot \mathbf{M}_{\mathrm{r}}$. The SSL branch takes $\mathbf{Y}_{\mathrm{o}}$ and $\hat{\mathbf{X}}$ as two raw inputs and reconstructs received frame $\hat{\mathbf{Y}}$ as the output. Letting $\phi_{\boldsymbol{\theta}_{\mathrm{s}}}(\cdot)$ be the mapping parameterized by $\boldsymbol{\theta}_{\mathrm{s}}$, the SSL task is expressed by

$$\hat{\mathbf{Y}} = \phi_{\boldsymbol{\theta}_{\mathrm{s}}}\left(\mathbf{Y}_{\mathrm{o}}, \hat{\mathbf{X}}\right). \tag{3}$$

B. Overview of Two Learning Phases

There exist two learning phases: offline pretraining and online adaptation. ChannelMAE is pretrained offline by jointly optimizing $\{\boldsymbol{\alpha}_{\mathrm{e}}, \boldsymbol{\beta}_{\mathrm{m}}, \boldsymbol{\beta}_{\mathrm{s}}\}$ on labeled data for both the main and SSL tasks. During online adaptation, we update only the shared encoder $\boldsymbol{\alpha}_{\mathrm{e}}$ using the SSL loss and keep both decoders frozen, since true channel coefficients are unavailable.

As illustrated in Fig. 2, Online adaptation is conducted step-by-step, with each step spanning ΔN TTIs. At each TTI, the following procedures are performed: 1) the receiver obtains received frame $\mathbf{Y}$ and performs LS estimation to obtain $\hat{\mathbf{H}}_{\mathrm{p}}$ based on Eq. (1), and feeds $\hat{\mathbf{H}}_{\mathrm{p}}$ into the main branch to output $\hat{\mathbf{H}}$; 2) the receiver then recovers transmitted data-symbols $\hat{\mathbf{X}}$; 3) $\{\mathbf{Y}, \hat{\mathbf{X}}\}$ of the current TTI is stored in an online buffer. As ΔN TTIs of data are buffered, they are augmented into one online batch, over which the SSL-branch loss is computed and used to update $\boldsymbol{\alpha}_{\mathrm{e}}$. The buffer is then emptied, and the next adaptation step begins with $\boldsymbol{\alpha}_{\mathrm{e}}$ re-initialized to its updated parameters. Through this *batch-wise online learning* pattern, ChannelMAE is gradually adapted to new channel distributions over time with a low memory footprint in terms of data storage.

V. Key Designs of ChannelMAE

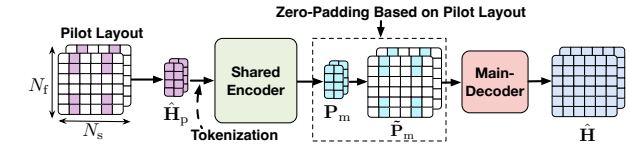
A. Input Pre-Processing

The shared feature encoder is realized by a transformer encoder. Thus, before entering the encoder, the inputs of the SSL task must be pre-processed and tokenized, while the input of the main task must be tokenized. Note that the complex tensors are transformed to real-valued ones by stacking their real and imaginary parts, which gives $\hat{\mathbf{H}}_{\mathrm{p}} \in \mathbb{R}^{N_{\mathrm{sp}} \times N_{\mathrm{fp}} \times 2}$ and $\mathbf{Y}, \mathbf{H}, \mathbf{X} \in \mathbb{R}^{N_{\mathrm{s}} \times N_{\mathrm{f}} \times 2}$. The last dimension of real tensors is termed 'channel' in this paper.

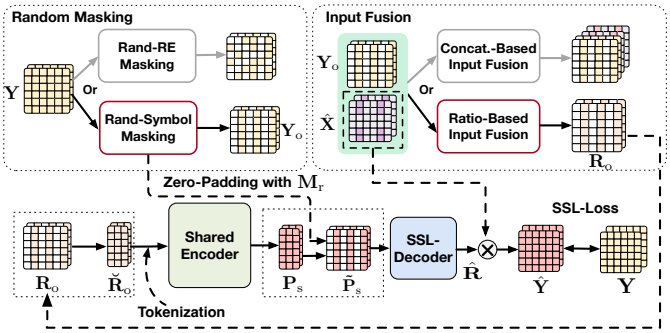
1) SSL-Task Pre-processing (Random Masking and Input Fusion): Before converting the input of the SSL task into a sequence of tokens, both random masking and input fusion must be carefully designed. First, the random masking scheme, i.e., the procedure for generating $\mathbf{M}_{\mathrm{r}}$, is developed. As illustrated in Fig. 3(b), two variants are investigated: random-symbol masking and random-RE masking. In random-symbol masking, N_{rm} OFDM symbols in each frame are masked, leaving an unmasked part with dimensions $(N_{\mathrm{s}} - N_{\mathrm{rm}}) \times N_{\mathrm{f}}$. By contrast, the random-RE masking scheme randomly masks N_{re} REs in a frame. Random-symbol masking better preserves inter-symbol temporal coherence and thus results in a higher reconstruction accuracy, so it is employed in ChannelMAE.

Second, as the model simultaneously ingests $\mathbf{Y}_{\mathrm{o}}$ and $\hat{\mathbf{X}}$ as defined in Eq. (3), an input fusion mechanism is required to transform these raw inputs into a single composite input. We compare two fusion schemes as in Fig. 3(b). The first scheme, termed concatenation-based fusion, concatenates the two inputs channel-wise, leading to $[\mathbf{Y}_{\mathrm{o}}; \hat{\mathbf{X}}]$. The second scheme computes an element-wise Hadamard quotient, $\mathbf{R}_{\mathrm{o}} = \mathbf{Y}_{\mathrm{o}} \odot \hat{\mathbf{X}}^{-1}$, which is called termed ratio-based fusion. In practice, the ratio-based fusion reduces input dimensionality and computational cost while delivering comparable performance to the other fusion scheme, so it is adopted in ChannelMAE.

As depicted in Fig. 3(b), given the random-symbol masking and the ratio-based fusion schemes, the resulting composite input $\mathbf{R}_{\mathrm{o}} \in \mathbb{R}^{N_{\mathrm{s}} \times N_{\mathrm{f}} \times 2}$ is then tokenized, which will be elaborated later. Note that only its non-zero part denoted by $\breve{\mathbf{R}}_{\mathrm{o}} \in \mathbb{R}^{(N_{\mathrm{s}} - N_{\mathrm{rm}}) \times N_{\mathrm{f}} \times 2}$ is tokenized and sent to the encoder, thereby reducing the shared encoder's computation cost.



(a) Detailed design of the main branch



(b) Detailed design of the SSL branch

Fig. 3. Detailed designs of two model branches.

2) Input Tokenization for Both Tasks: Main-task input $\hat{\mathbf{H}}_{\mathrm{p}} \in \mathbb{R}^{N_{\mathrm{sp}} \times N_{\mathrm{fp}} \times 2}$ is converted into a sequence of tokens $\{\mathbf{T}_{\mathrm{m}}\}$, letting $\mathbf{T}_{\mathrm{m}}$ be a single token of main task. Two tokenization schemes are compared. The first one is symbol-wise tokenization: it treats each pilot symbol time as one token, i.e., $\mathbf{T}_{\mathrm{m}} \in \mathbb{R}^{2N_{\mathrm{fp}}}$, with N_{sp} tokens in total. The other one is channel-wise tokenization that aggregates all values belonging to the same complex channel into a single token, producing only two tokens—one for the real part and one for the imaginary part—each of size $\mathbf{T}_{\mathrm{m}} \in \mathbb{R}^{N_{\mathrm{sp}} N_{\mathrm{fp}}}$. This tokenization allows the attention layer to capture time–frequency patterns over the entire frame more effectively, and thus the model performance outperforms the former one. Ablation studies further confirm that the channel-wise scheme consistently outperforms the symbol-wise alternative. Therefore, ChannelMAE uses channel-wise tokenization throughout its operation. Similar to the main-task tokenization, $\tilde{\mathbf{R}}_{\mathrm{o}}$ is converted into a sequence of two tokens, each of size $\mathbf{T}_{\mathrm{s}} \in \mathbb{R}^{(N_{\mathrm{s}} - N_{\mathrm{rm}}) N_{\mathrm{f}}}$, where $\mathbf{T}_{\mathrm{s}}$ represents a single SSL token.

B. Model Architecture: Two-Branch MAE

Both the main and SSL branches of ChannelMAE adopt an MAE architecture, as shown in Fig. 3. A transformer encoder is shared between two branches, and each branch ends in a ResNet-based decoder. Leveraging self-attention, the transformer encoder can learn global latent representation that CNN-based encoders often overlook [17]. Moreover, the encoder in ChannelMAE only processes the non-zero parts of each input. Then, through zero-padding, the positional information of the pilot layout and the random mask is re-inserted into the main and SSL branches before entering the decoder, respectively, as shown in Fig. 3(a) and (b). Insertion of such information explicitly informs the model which parts are missing, enabling accurate reconstruction. The model components are detailed in the following.

1) Shared Encoder: As shown in Fig. 4(a), the encoder $\boldsymbol{\alpha}_{\mathrm{e}}$ comprises N_{e} transformer layers [17], each containing

a multi-head self-attention block and a two-layer multi-layer perception (MLP) network. Residual connections and layer normalization (LN) follow both blocks. Within each layer, self-attention captures correlations among input tokens. Letting $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ be queries, keys, and values, we have $\mathbf{Q} = \mathbf{K} = \mathbf{V}$. Throughout the encoder, their dimensions are kept as $\mathbb{R}^{N_{\rm seq} \times N_{\rm m}}$, where $N_{\rm seq}$ is the token count in the input sequence and $N_{\rm m}$ is the embedding dimension. Let $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{R}^{N_{\rm m} \times d_k}$ be the learnable parameters of each attention head i associated with queries, keys, and values, respectively, where d_k is the key dimension (also the query and value dimension). The output projection matrix $\mathbf{W}^O \in \mathbb{R}^{hd_k \times N_{\rm m}}$, where h is the number of attention heads. To this end, multi-head self-attention layer $\mathrm{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ [17] is computed as $\mathrm{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Concat}(\mathrm{head}_1, ..., \mathrm{head}_h)\mathbf{W}^O$, $\mathrm{head}_i = \mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{W}_i^Q(\mathbf{K}\mathbf{W}_i^K)^{\rm T}}{\sqrt{d_k}}\right)\mathbf{V}\mathbf{W}_i^V$, where $\mathrm{Softmax}(\cdot)$ denotes SoftMax function. The subsequent MLP has hidden size $N_{\rm hid}$ and output size $N_{\rm m}$, with GeLu activation applied only after the first layer.

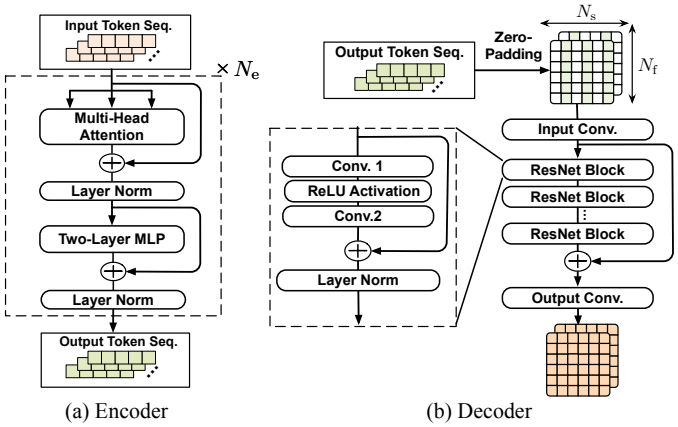
Note that tokens $\{\mathbf{T}_{\rm m}\}$ or $\{\mathbf{T}_{\rm s}\}$ stated in Section V-A are linearly projected to $\mathbb{R}^{N_{\rm seq} \times N_{\rm m}}$ and serve as the initial $\mathbf{Q}, \mathbf{K}, \mathbf{V}$. Since channel-wise tokenization is employed, $N_{\rm seq} = 2$ for both tasks.

2) Main-Task Decoder: The output sequence of the encoder denoted by $\mathbf{P}_{\rm m} \in \mathbb{R}^{2 \times N_{\rm sp} N_{\rm fp}}$ is reshaped to $\mathbb{R}^{N_{\rm sp} \times N_{\rm fp} \times 2}$. Using the pre-known pilot pattern, non-pilot positions are padded with zeros to obtain a full-size representation $\tilde{\mathbf{P}}_{\rm m} \in \mathbb{R}^{N_{\rm s} \times N_{\rm f} \times 2}$, as shown in Fig. 3(a). This tensor passes through the main-task decoder that consists of an input convolutional layer, $N_{\rm dm}$ ResNet blocks [23], and an output convolutional layer, resulting in estimated channel coefficients $\hat{\mathbf{H}} \in \mathbb{R}^{N_{\rm s} \times N_{\rm f} \times 2}$.

3) SSL-Task Decoder: The encoder output for the SSL branch, denoted as $\mathbf{P}_{\rm s} \in \mathbb{R}^{2 \times (N_{\rm s} - N_{\rm rm}) N_{\rm f}}$, is reshaped to $\mathbb{R}^{(N_{\rm s} - N_{\rm rm}) \times N_{\rm f} \times 2}$. Then random-symbol mask $\mathbf{M}_{\rm r}$ is inserted back to this representation, which gives $\tilde{\mathbf{P}}_{\rm s} \in \mathbb{R}^{N_{\rm s} \times N_{\rm f} \times 2}$. The SSL-task decoder follows the same design as that of main-task decoder, except that it has a different number of ResNet blocks denoted by $N_{\rm ds}$. Since only the SSL task is trained online to update the shared encoder, a more lightweight SSL-decoder is designed to reduce the back-propagation cost online (i.e., $N_{\rm ds} < N_{\rm dm}$). At last, as the ratio-based fusion is applied during pre-processing, the output of SSL-decoder denoted by $\hat{\mathbf{R}}$ is further post-processed to obtain reconstructed frame $\hat{\mathbf{Y}}$, i.e., $\hat{\mathbf{Y}} = \hat{\mathbf{R}} \odot \hat{\mathbf{X}}$, as depicted in Fig. 3(b).

C. Two-Phase Training Procedures

ChannelMAE is trained in two phases: offline pretraining and online adaptation. Let $\ell_{\rm m}(\cdot)$ and $\ell_{\rm s}(\cdot)$ represent per-sample loss functions of the main and SSL tasks, respectively. We denote $\mathbf{Y}'$ as the masked parts of $\mathbf{Y}$ and $\hat{\mathbf{Y}}'$ as the reconstructed masked parts extracted from $\hat{\mathbf{Y}}$. The SSL-task loss computes the squared error *only* on the masked parts of the frame, which is $\ell_{\rm s}(\mathbf{Y}_{\rm o}, \hat{\mathbf{X}}, \mathbf{Y}') = \left\|\mathbf{Y}' - \phi'_{\boldsymbol{\theta}_{\rm s}}(\mathbf{Y}_{\rm o}, \hat{\mathbf{X}})\right\|_F^2$, where $\phi'_{\boldsymbol{\theta}_{\rm s}}(\cdot)$ further applies the extraction of the masked parts upon $\phi_{\boldsymbol{\theta}_{\rm s}}(\cdot)$, i.e., $\hat{\mathbf{Y}}' = \phi'_{\boldsymbol{\theta}_{\rm s}}(\mathbf{Y}_{\rm o}, \hat{\mathbf{X}})$, and $\|\cdot\|_F$ denotes the Frobenius norm. The main-task loss is also the squared error, i.e., $\ell_{\rm m}(\hat{\mathbf{H}}_p, \mathbf{H}) = \left\|\mathbf{H} - \varphi_{\boldsymbol{\theta}_{\rm m}}(\hat{\mathbf{H}}_p)\right\|_F^2$.



(a) Encoder (b) Decoder

Fig. 4. Model architecture of ChannelMAE.

1) Offline Pretraining: ChannelMAE is first optimized offline, where the trainable parameters are $\boldsymbol{\theta} = \{\boldsymbol{\alpha}_{\rm e}, \boldsymbol{\beta}_{\rm m}, \boldsymbol{\beta}_{\rm s}\}$. Both the ground-truth channel coefficients and transmit symbols are available in offline simulations, so $\mathbf{H}$ and $\mathbf{X}$ are known. Assuming N offline samples, the mean-squared error (MSE) loss is

$$\mathcal{L}_{\rm off}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\Big[\ell_{\rm m}\big(\hat{\mathbf{H}}_p^{(i)}, \mathbf{H}^{(i)}\big) + \ell_{\rm s}\big(\mathbf{Y}_{\rm o}^{(i)}, \mathbf{X}^{(i)}, \mathbf{Y}'^{(i)}\big)\Big], \tag{4}$$

where i denotes the index of data sample. This loss is then minimized via multi-epoch training with respect to $\boldsymbol{\theta}$.

2) Online adaptation: During online deployment, only the SSL task is learned to update the shared encoder. In each online adaptation step mentioned in Section IV, estimated data-symbols $\hat{\mathbf{X}}$ must be recovered after main-task inference. Let T be the index of online adaptation step and t be the index of TTI within each step. Thus $\boldsymbol{\alpha}_{\rm e}^{(T)}$ represents the encoder used in online adaptation step T, and $\boldsymbol{\theta}_{\rm m}^{(T)}, \boldsymbol{\theta}_{\rm s}^{(T)}$ denote the main and SSL branch in the current step, respectively.

The online symbol-recovery mechanism is stated as follows. As shown in Fig. 5, two optional schemes are provided for recovering $\hat{\mathbf{X}}$, which are the uncoded and channel-coded recovery loops. First, in the uncoded recovery loop, the data symbols are detected by the conventional LMMSE symbol detector, and the output is directly fed back as one of the input components of the SSL branch. This scheme conforms to the normal symbol-detection procedure and thus does not incur extra computation cost to the detection pipeline. Specifically, at TTI t, the main-task inference is performed through $\hat{\mathbf{H}}^{(t)} = \varphi_{\boldsymbol{\theta}_{\rm m}^{(T)}}(\hat{\mathbf{H}}_{\rm p}^{(t)})$, where $\hat{\mathbf{H}}^{(t)}, \hat{\mathbf{H}}_{\rm p}^{(t)}$ are the main-task input and output at TTI t. Given $\hat{\mathbf{H}}^{(t)}$, the LMMSE symbol detection is conducted via

$$\hat{\mathbf{X}}^{(t)} = f_{\rm det}(\hat{\mathbf{H}}^{(t)}, \mathbf{Y}^{(t)}), \tag{5}$$

where $f_{\rm det}(\cdot)$ characterizes the symbol detection process.

Second, in the channel-coded recovery loop, $\hat{\mathbf{X}}$ is recovered by incorporating channel decoding/encoding processes. In this

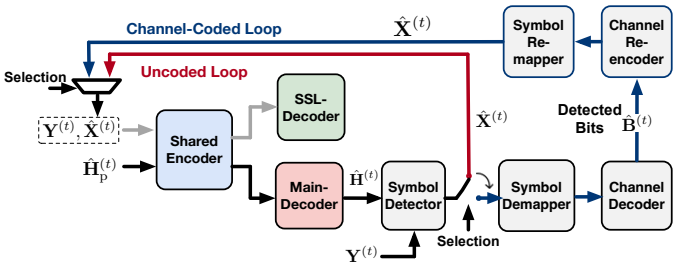


Fig. 5. Uncoded and channel-coded online symbol-recovery schemes.

scheme, we denote the detected bits of TTI t output by the channel decoder as $\hat{\mathbf{B}}^{(t)}$. Then the process of acquiring estimated symbols $\hat{\mathbf{X}}$ is:

$$\hat{\mathbf{B}}^{(t)} = g_{\text{dec}}(\hat{\mathbf{H}}^{(t)}, \mathbf{Y}^{(t)}), \quad \hat{\mathbf{X}}^{(t)} = h_{\text{enc}}(\hat{\mathbf{B}}^{(t)}), \tag{6}$$

where $g_{\text{dec}}(\cdot)$ abstracts the receiving process of symbol detection, symbol de-mapping, and channel decoding; $h_{\text{enc}}(\cdot)$ abstracts the additional process of channel re-encoding and symbol remapping. It is evident that the channel-coded recovery loop incurs extra high computation cost due to $h_{\text{enc}}(\cdot)$.

In both schemes, symbol errors in $\hat{\mathbf{X}}$ can corrupt SSL learning and misdirect encoder updates. To limit this effect, we use a threshold-based filter: online symbol recovery is performed only on received frames with SNR above 5 dB, and only these frames are used to train the encoder. Section VI-B further shows that the SSL task tolerates a moderate error level. Since the two schemes achieve similar adaptation performance, we adopt the uncoded loop for online symbol recovery due to its much lower computational cost.

At each TTI t, $\{\mathbf{Y}^{(t)}, \hat{\mathbf{X}}^{(t)}\}$ is pushed in the online buffer. During one online adaptation step, a batch of online samples $\{\mathbf{Y}^{(t)}, \hat{\mathbf{X}}^{(t)}\}_{t=1}^{\Delta N}$ with size ΔN is buffered. By the end of each step, this batch is popped and enlarged by an *augmentation factor* A. Specifically, for each $\mathbf{Y}^{(t)}$, A independent random masks are drawn and applied to generate A masked frames. Each original pair $\{\mathbf{Y}^{(t)}, \hat{\mathbf{X}}^{(t)}\}$ appears exactly A times in the augmented batch denoted by $\mathcal{D}_{\text{aug}}$, yielding a total of $A\Delta N$ augmented training samples. $\mathcal{D}_{\text{aug}}$ is the one-batch data for learning the SSL task online.

During online adaptation, the SSL decoder $\boldsymbol{\beta}_{\text{s}}$ is frozen and only the shared encoder $\boldsymbol{\alpha}_{\text{e}}$ is updated. Over the online augmented-batch data, the MSE loss is

$$\mathcal{L}_{\text{on}}(\boldsymbol{\alpha}_{\text{e}}) = \frac{1}{|\mathcal{D}_{\text{aug}}|} \sum_{i=1}^{|\mathcal{D}_{\text{aug}}|} \ell_{\text{s}}\big(\mathbf{Y}_{\text{o}}^{(i)}, \hat{\mathbf{X}}^{(i)}, \mathbf{Y}'^{(i)}\big). \tag{7}$$

The encoder parameters in the current adaptation step T are updated via one-step gradient descent to obtain new encoder parameters $\boldsymbol{\alpha}_{\text{e}}^{(T+1)}$, i.e., $\boldsymbol{\alpha}_{\text{e}}^{(T+1)} = \boldsymbol{\alpha}_{\text{e}}^{(T)} - \mu \nabla_{\boldsymbol{\alpha}_{\text{e}}} \mathcal{L}_{\text{on}}\big(\boldsymbol{\alpha}_{\text{e}}^{(T)}\big)$, with learning rate μ.

VI. Performance Evaluation

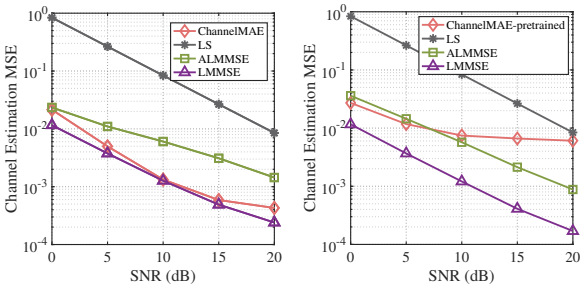
A. Simulation Setup

1) Wireless System and Channel Datasets: Considering an SISO-OFDM uplink communication model, the system setup is as follows. The carrier frequency is 3GHz with subcarrier frequency 30kHz. Each OFDM frame consists of 14 symbols and 72 subcarriers, i.e, $N_{\text{s}} = 14, N_{\text{f}} = 72$. Within each frame, a 3GPP-aligned pilot pattern [24] is used: two OFDM symbol times (the 3rd and 10th symbols) are selected as pilots, giving $N_{\text{sp}} = 2, N_{\text{fp}} = 72$. The modulation scheme is fixed as 4-QAM. LDPC channel coding employs code rate $658/1024$.

Two categories of channel datasets are considered. First, 3GPP standard channel models, urban macro (UMa) and urban micro (UMi) [25], are simulated. To demonstrate offline-online channel distribution shifts, UMa with low mobility (0–5 m/s) provides the offline distribution, whereas UMi with high mobility (25–30 m/s) serves online. Second, ray-traced (RT) real-world channels are generated with Sionna ray tracing tools [26], [27]. RT-based channel datasets with mobility 0–10 m/s are prepared using five real-world city scenes [28], namely generic street canyon, Paris, Florence, Munich, and San Francisco, and they are labeled as City1–City5 for clarity. For a single distribution shift, we specifically employ City5 as the offline channel environment and City3 as the online one, and we also evaluate multiple online environment shifts.

2) Training Setup: During offline pretraining, 40,000 TTIs in the SNR range of 10–20 dB are generated for each offline channel distribution, with the training/validation/test split as 0.8:0.1:0.1. After 80 training epochs using the Adam optimizer (with learning rate 0.001 and batch size 64). During online adaptation, the adaptation step contains 32 TTIs, and thus the original online batch size is also 32. The batch-wise online learning as stated in Section V-C is performed with learning rate 0.0005. In total 10,000 TTIs within SNR range 10–15 dB are simulated online. With the ultimate online-adapted model, we evaluate the online channel estimation performance with another separate test dataset. Unless otherwise specified, the key hyper-parameters are specified as follows: ChannelMAE uses one encoder layer ($N_{\text{e}} = 1$); $N_{\text{dm}} = 4$ and $N_{\text{ds}} = 2$ ResNet blocks in the main and SSL decoders, respectively; $N_{\text{rm}} = 12$ masked symbols in the SSL branch; data augmentation $A = 5$ times; MLP hidden dimension $N_{\text{hid}} = 16$; and encoder embedding dimension $N_{\text{m}} = 144$. Both decoders employ a kernel size of 5. All simulations are conducted with Sionna 1.0.1 [29] and TensorFlow 2.15.0 on an NVIDIA RTX 4090 and an Apple M4 CPU.

3) Baselines: Among conventional methods [30], we include LS, ideal approximate-LMMSE (i.e., ALMMSE), and ideal LMMSE (i.e., LMMSE). The last two are both ideal baselines, as ALMMSE estimates channel statistics from true channel coefficients and LMMSE uses noiseless pilot estimates and perfect channel statistics for its implementation. For DL-based methods we compare ChannelMAE with a CNN-based benchmark ChannelNet [4], the state-of-the-art transformer-based HA02 [2], [20], the self-supervised denoiser DnCNN [7], the state-of-the-art model that supports online adaptation HA03 [2], [8]. The details of these models are stated in Section II. During online adaptation, only the last two models are considered as they can be adapted online without true channel coefficients. We also consider using ground-truth channel coefficients to train the main branch of ChannelMAE



(a) Offline RT channel (b) Online RT channel

Fig. 6. Performance degradation of the pretrained model encountering the offline-online distribution shift.

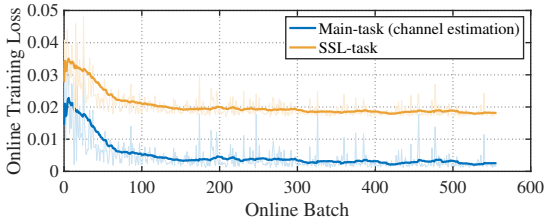


Fig. 7. Online losses of ChannelMAE versus online batch index (both losses are running-averaged for 30 batches).

in a *supervised* manner, which serves as the upper bound for ChannelMAE if no data augmentation is applied.

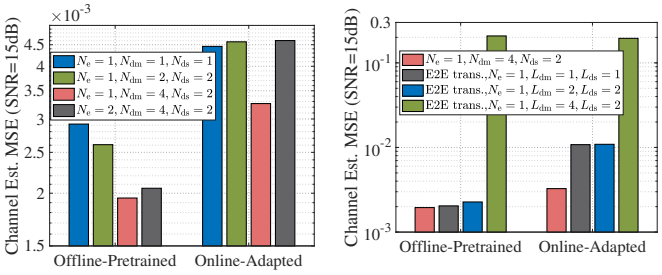
To evaluate channel estimators, standard Monte-Carlo simulations are conducted at the selected SNR points, computing the MSE between estimated and ground-truth channel coefficients. The MSE gain (in dB) of scheme 2 over scheme 1 is defined as $10\log_{10}(\mathrm{MSE}_1/\mathrm{MSE}_2)$, where MSE_1 and MSE_2 are MSEs of two schemes, respectively. Computation cost is measured by TensorFlow Profiler in terms of floating-point operation in millions (MFLOP).

B. Ablation Studies

Validation of Online Performance Degradation and Online Model Convergence. ChannelMAE is pretrained in the offline RT channel and then evaluated in both the offline and online RT channels. As shown in Fig. 6(a), the pretrained model outperforms ALMMSE across 0–20 dB SNR and even achieves performance comparable to LMMSE. However, it witnesses a drastic performance degradation online, as shown in Fig. 6(b), which validates the necessity of online adaptation. The convergence behavior of online adaptation is depicted in Fig. 7, where the SSL task alone is trained to update the shared encoder. As more online batches are observed, the SSL-task loss decreases to a plateau, driving down the main-task loss as well. This indicates a clear synergy between the learning processes of the two tasks.

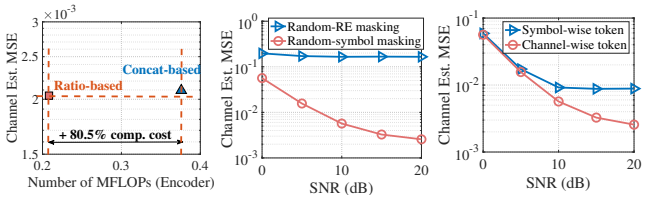
The following ablation studies are conducted using 3GPP channels and the determined designs are also applied to RT channels.

1) Model Architecture: The impact of various model architectures on ChannelMAE is studied in two aspects. First, the number of encoder layers (i.e., N_{e}) and the numbers of ResNet blocks in the main-decoder and SSL-decoder (i.e., N_{dm}, N_{ds}) are varied. The offline-pretrained and online-adapted models with various model structures are evaluated at an SNR of



(a) Various numbers of layers (b) CNN vs. transformer decoder

Fig. 8. Various model architectures of ChannelMAE.



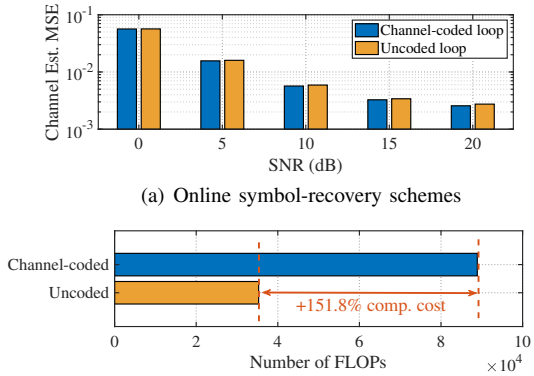
(a) Input fusion (b) Masking scheme (c) Tokenization

Fig. 9. Ablation studies of input pre-processing.

15 dB, respectively. As shown in Fig. 8(a), the model with $N_{\mathrm{e}}=1, N_{\mathrm{dm}}=4, N_{\mathrm{ds}}=2$ outperforms all others in both the offline and online phases. A relatively smaller SSL-decoder is designed to reduce online backpropagation cost through the SSL branch. Second, an end-to-end transformer-based MAE is compared with ChannelMAE. Specifically, the main and SSL decoders in ChannelMAE are replaced by transformer decoders [15] with L_{dm} and L_{ds} layers, respectively, while retaining the original encoder. This variant achieves a similar channel-estimation MSE to ChannelMAE when $L_{\mathrm{dm}}=L_{\mathrm{ds}}=1$, but its performance degrades significantly during online adaptation as in Fig. 8(b). Based on these two analyses, the two-branch MAE is determined with $N_{\mathrm{e}}=1$, $N_{\mathrm{dm}}=4$, and $N_{\mathrm{ds}}=2$.

2) Input Pre-Processing: As stated in Section V-A, the following design aspects must be studied: a) the random masking scheme and the input fusion scheme for the SSL-task; b) the tokenization scheme for both tasks. We first compare two fusion schemes, ratio-based and concatenation-based fusion, in the pretraining phase. As shown in Fig. 9(a), the ratio-based fusion scheme achieves a slightly lower MSE than the concatenation-based scheme in the high-SNR region (i.e., 20 dB). Meanwhile, the latter one increases computation cost by around 80.5% in terms of MFLOPs. Therefore, the ratio-based fusion scheme is adopted. Next, the random-symbol and random-RE masking schemes are compared. The latter fails to converge online, producing a flat MSE curve, as seen in Fig. 9(b). Thus, random-symbol masking must be set for the SSL task. We also vary the number of masked symbols N_{rm} but it does not have any noticeable impact on the results. Last, two tokenization methods are compared in Fig. 9(c). The online-adapted model using channel-wise tokenization achieves an MSE gain of around 5.4 dB than the symbol-wise approach at 20 dB SNR, so the channel-wise tokenization is adopted.

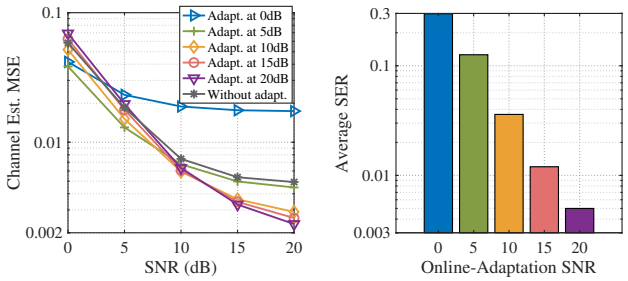
3) Online Symbol-Recovery Mechanism: Two schemes for obtaining estimated data-symbols stated in Section V-C are



(a) Online symbol-recovery schemes

(b) Per-sample FLOP counts

Fig. 10. Comparison of two online symbol-recovery schemes in terms of channel estimation performance and computation cost.



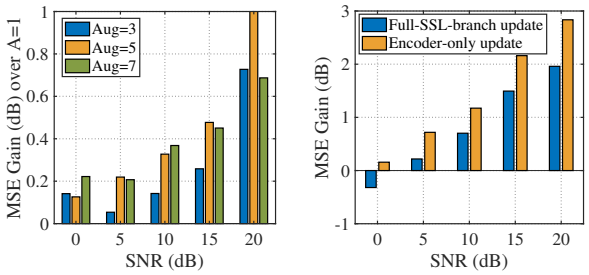
(a) Evaluation of adapted models

(b) Average SER in $\hat{\mathbf{X}}$

Fig. 11. Comparison of various online-adaptation SNR values in terms of channel estimation performance and average SER in online symbol recovery.

compared, depending on whether channel coding is included in the loop. As shown in Fig. 10(a), the channel-coded and uncoded loops achieve nearly identical performance over 0–20 dB SNR, indicating that channel coding offers little benefit for SSL-task learning. During online adaptation, even if the average symbol-error rate (SER) of $\hat{\mathbf{X}}$ increases from 0.00298 to 0.0241 without channel coding, the SSL reconstruction still remains effective. This indicates that the SSL task is robust to SER at this level and thus both schemes can perform similarly. Moreover, using the channel-coded loop increases the number of FLOPs per online sample by 151.8% in Fig. 10(b). Therefore, throughout the online adaptation process, we select the uncoded loop.

4) Impact of Online-Adaptation SNR: The impact of online-adaptation SNR (i.e., SNR condition of online batches) on model performance is studied as in Fig. 11(a). When the online-adaptation SNR drops to 0 dB, adaptation proves ineffective and can even degrade performance below the pretrained model. This is mainly due to two factors: a higher SER in $\hat{\mathbf{X}}$ and stronger noise in $\hat{\mathbf{Y}}$, both of which corrupt SSL learning. Fig. 11(b) reports the average SER of the uncoded symbol-recovery loop. At lower online-adaptation SNRs ($\leq$5 dB), the average SER exceeds 0.1, where SSL reconstruction breaks down and provides little to no benefit to the main task. At higher online-adaptation SNRs, the SER stays low enough for SSL to remain effective, and the adapted model achieves low channel-estimation MSE in high-SNR regions, while still lagging in low-SNR regions as seen in Fig. 11(a). This motivates the threshold-based filtering in Section V-C: we



(a) Data augmentation

(b) Online model updating

Fig. 12. Comparison of various data augmentation times and online model updating schemes.

TABLE I
COMPARISON OF MODEL PARAMETER COUNTS (IN MILLIONS).

Model	Total params(M)	Online-trainable params(M)
HA03	**0.147**	0.147
ChannelMAE	0.206	**0.126**
DnCNN	0.228	0.227
HA02	0.272	N/A
ChannelNet	0.686	N/A

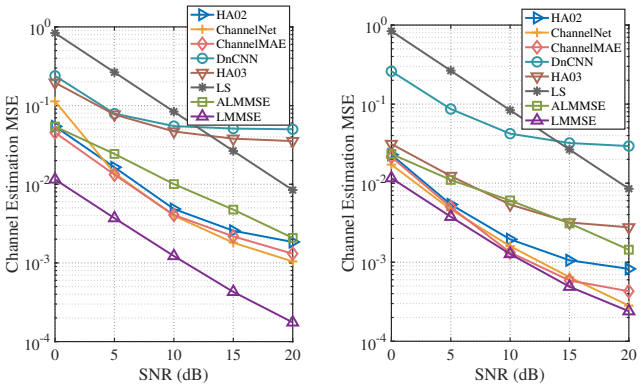
perform online symbol recovery only on frames with SNR above 5 dB and use only these frames for SSL.

5) Online Data Augmentation and Model Updating: Online data augmentation is also studied as in Fig. 12(a). Compared with the no-augmentation case (A=1), the five-fold augmentation scheme (A=5) achieves the overall largest performance gain, drastically outperforming the 7-fold one (A=7) at 20 dB. Thus, we adopt A=5. Furthermore, two update schemes are compared: updating the full SSL branch versus updating only the shared encoder. As shown in Fig. 12(b), we compute the MSE gain of the adapted model relative to the pretrained baseline. The encoder-only update achieves a larger MSE improvement than updating the full branch, which justifies our design of only adapting the shared encoder online. This is because updating the full branch is more prone to overfitting the encoder to the SSL objective, which in turn harms the main task. Restricting updates to the encoder mitigates this overfitting and better preserves the features that are shared across the two tasks. Meanwhile, as in Table I, the encoder contains 0.126M parameters while the full SSL branch has 0.153M parameters, and thus the encoder-only update scheme reduces model updating cost by roughly 18%.

C. Comparison with Baselines

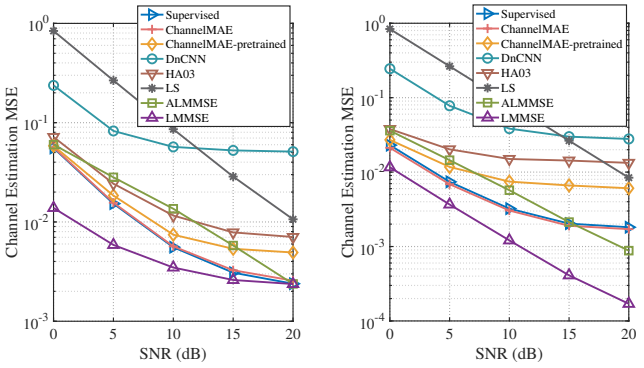
Table I compares the parameter counts of each model. Memory consumption scales with parameter count under a given floating-point quantization. Among the three architectures that support online adaptation, HA03 has the smallest overall footprint (0.147M parameters), whereas ChannelMAE requires the fewest trainable parameters during adaptation (0.126M). In the following, we elaborate the performance of both offline-pretrained and online-adapted models in 3GPP-aligned and RT channel scenarios.

1) Offline-Pretrained Model Performance: In offline 3GPP channels shown in Fig. 13(a), ChannelMAE reduces the channel-estimation MSE by 15.2–29.0% compared with HA02 across the whole SNR range. Also, its MSE is up to 95.1%



(a) Offline 3GPP channel (b) Offline RT channel

Fig. 13. Evaluation of offline-pretrained models and conventional baselines in offline channel environments.



(a) Online 3GPP channel (b) Online RT channel

Fig. 14. Evaluation of online-adapted models and conventional baselines in online channel environments.

lower than LS and up to 59.5% lower than ALMMSE. Although ChannelNet attains a slightly lower MSE at high SNR values of 15–20 dB than ChannelMAE, its performance significantly drops at 0 dB SNR. In offline RT channels shown in Fig. 13(b), ChannelMAE nearly matches the performance of LMMSE and has a lower MSE than HA02 by 6.52–47.9%. Despite ChannelNet yields comparable results to ChannelMAE, its model size is over three times larger than ChannelMAE. These results demonstrate that ChannelMAE achieves the state-of-the-art offline channel-estimation performance, while maintaining a smaller model size than its competing counterpart.

2) Online-Adapted Model Performance: For online adaptation, models pretrained in offline 3GPP channels undergo adaptation in online 3GPP channels, while those pretrained in offline RT channels are adapted in online RT channels. These adapted models are evaluated online as seen in Fig. 14. The online-adapted ChannelMAE significantly achieves lower MSEs by 3.55–47.9% and 21.7–71.8% than the pretrained one in 3GPP and RT channels, respectively. In both channel environments, it reaches close to the performance of the supervised scheme using ground-truth channel coefficients, and even slightly outperforms the supervised scheme in RT channels thanks to online data augmentation. Compared with the state-of-the-art online model HA03, ChannelMAE reduces MSEs by 21.4–63.7% and 44.0–87.1% in 3GPP and RT channels, respectively.

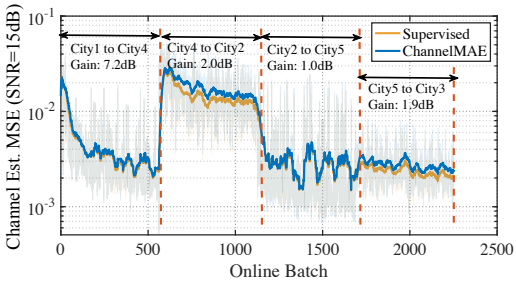


Fig. 15. Continual online adaptation in multiple channel environment changes.

3) Continual Adaptation Under Environmental Change: A continual online adaptation process with four environmental shifts is studied, where ChannelMAE is pretrained in City1 and then adapted online throughout City4, City2, City5, and City3 sequentially. As shown in Fig. 15, for the first two shifts, ChannelMAE witnesses significant MSE gains evaluated at 15 dB SNR, while the model maintains a low MSE for the last two shifts as channel knowledge accumulates. Throughout the adaptation process, ChannelMAE reaches close to the supervised-learning scheme.

One may ask whether ChannelMAE can retain long-term knowledge so that further retraining can be avoided. Our focus, however, is effective and efficient online adaptation in a self-supervised manner promptly whenever needed. We do not aim to preserve long-term knowledge or explicitly address catastrophic forgetting. Continual-learning techniques such as experience replay could be incorporated to reduce the adaptation frequency, but this is beyond the scope of this paper.

VII. Conclusion

To enable label-free online adaptation of pretrained neural channel estimators, an SSL task was designed on top of the original channel-estimation task. It effectively reconstructed the masked parts of randomly-masked received frames. Both tasks were then consolidated into a two-branch MAE model ChannelMAE, where each branch was dedicated to one task and two branches shared the same encoder but used separate decoders. By online adapting this shared encoder through optimizing the SSL-task branch, online channel-estimation performance was significantly improved over baselines. ChannelMAE is the first approach that realizes online adaptation of neural channel estimators without ground-truth channel coefficients, prior channel statistics, or additional pilot overhead. The designed two-task framework shows great promise for label-free adaptation across a broad range of wireless applications, such as neural receivers, interference detection, and anomaly detection.

For future work, the model architecture demands more studies to support Multiple-Input Multiple-Output (MIMO). A formal theoretical analysis will be developed to quantify the synergy between the SSL objective and channel estimation. Hardware prototyping is also planned to assess computational cost in real-world deployments. The authors have provided public access to their code and data at https://github.com/tesiawang/ChannelMAE.

REFERENCES

[1] M. Honkala, D. Korpi, and J. M. Huttunen, "DeepRx: Fully convolutional deep learning receiver," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 6, pp. 3925–3940, 2021.

[2] D. Luan and J. S. Thompson, "Channelformer: Attention based neural solution for wireless channel estimation and effective online training," *IEEE Trans. Wireless Commun.*, vol. 22, no. 10, pp. 6562–6577, 2023.

[3] M. Belgiovine, K. Sankhe, C. Bocanegra, D. Roy, and K. R. Chowdhury, "Deep learning at the edge for channel estimation in beyond-5G massive mimo," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 19–25, 2021.

[4] M. Soltani, V. Pourahmadi, A. Mirzaei, and H. Sheikhzadeh, "Deep learning-based channel estimation," *IEEE Commun. Lett.*, vol. 23, no. 4, pp. 652–655, 2019.

[5] M. B. Fischer, S. Dörner, S. Cammerer, T. Shimizu, H. Lu, and S. Ten Brink, "Adaptive neural network-based OFDM receivers," in *Proc. IEEE Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, 2022, pp. 1–5.

[6] O. Wang, J. Gao, and G. Y. Li, "Learn to adapt to new environments from past experience and few pilot blocks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 9, no. 2, pp. 373–385, 2022.

[7] Z. Zhang, T. Ji, H. Shi, C. Li, Y. Huang, and L. Yang, "A self-supervised learning-based channel estimation for IRS-aided communication without ground truth," *IEEE Trans. Wireless Commun.*, vol. 22, no. 8, pp. 5446–5460, 2023.

[8] L. Kong, X. Liu, X. Zhang, J. Xiong, H. Zhao, and J. Wei, "Representation-based continual learning for channel estimation in dynamic wireless environments," *IEEE Trans. Wireless Commun.*, vol. 24, no. 8, pp. 6382–6396, 2025.

[9] T. Wang, S. Wang, Y. G. Li, and X. Wang, "Collaborative learning for less online retraining of neural receivers," in *Proc. IEEE Int. Workshop Machine Learn. Signal Process. (MLSP)*, 2024, pp. 1–6.

[10] T. Wang, Y. G. Li, and X. Wang, "GraphRx: Graph-based collaborative learning among multiple cells for uplink neural receivers," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2025, pp. 1–10.

[11] S. Wang, T. Wang, Y. G. Li, and X. Wang, "FedPDA: Collaborative learning for reducing online-adaptation frequency of neural receivers," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2025, pp. 1–10.

[12] Y. Sun, X. Wang, Z. Liu, J. Miller, A. Efros, and M. Hardt, "Test-time training with self-supervision for generalization under distribution shifts," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 9229–9248.

[13] Y. Gandelsman, Y. Sun, X. Chen, and A. Efros, "Test-time training with masked autoencoders," in *Proc. Adv. Neural Inf. Process. Syst.(NeurIPS)*, vol. 35, 2022, pp. 29 374–29 385.

[14] Y. Liu, P. Kothari, B. van Delft, B. Bellot-Gurlet, T. Mordan, and A. Alahi, "TTT++: When does self-supervised test-time training fail or thrive?" in *Proc. Adv. Neural Inf. Process. Syst.(NeurIPS)*, vol. 34, 2021, pp. 21 808–21 820.

[15] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recogn.(CVPR)*, 2022, pp. 16 000–16 009.

[16] T. Raviv, S. Park, O. Simeone, Y. C. Eldar, and N. Shlezinger, "Online meta-learning for hybrid model-based deep receivers," *IEEE Trans. Wirel. Commun.*, vol. 22, no. 10, pp. 6415–6431, 2023.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017, p. 6000–6010.

[18] M. Soltani, V. Pourahmadi, and H. Sheikhzadeh, "Pilot pattern design for deep learning-based channel estimation in OFDM systems," *IEEE Wireless Commun. Lett.*, vol. 9, no. 12, pp. 2173–2176, 2020.

[19] L. Li, H. Chen, H.-H. Chang, and L. Liu, "Deep residual learning meets OFDM channel estimation," *IEEE Wireless Commun. Lett.*, vol. 9, no. 5, pp. 615–618, 2020.

[20] D. Luan and J. Thompson, "Attention based neural networks for wireless channel estimation," in *Proc. IEEE Vehic. Technol. Conf. (VTC-Spring)*. IEEE, 2022, pp. 1–5.

[21] Z. Chen, F. Gu, and R. Jiang, "Channel estimation method based on transformer in high dynamic environment," in *Proc. IEEE Int. Conf. Wireless Commun. Signal Process. (WCSP)*, 2020, pp. 817–822.

[22] Y. Liu, Z. Tan, H. Hu, L. J. Cimini, and G. Y. Li, "Channel estimation for OFDM," *IEEE Commun. Surv. Tutor.*, vol. 16, no. 4, pp. 1891–1908, 2014.

[23] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recogn. Workshops*, 2017, pp. 136–144.

[24] 3rd Generation Partnership Project (3GPP), "NR;physical channels and modulation," 3GPP, TS 38.211, 2024, v18.3.0.

[25] ——, "Study on channel model for frequencies from 0.5 to 100 GHz," 3GPP, TS 38.901, 2024, v17.1.0.

[26] NVIDIA, "SionnaRT: A lightning-fast stand-alone ray tracer for radio propagation modeling," https://nvlabs.github.io/sionna/, 2025.

[27] J. Hoydis *et al.*, "Sionna RT: Differentiable Ray Tracing for Radio Propagation Modeling," *arXiv preprint arXiv:2303.11103*, 2023.

[28] OpenStreetMap contributors, "Openstreetmap building footprints," https://openstreetmap.org, 2025.

[29] J. Hoydis, S. Cammerer, F. A. Aoudia, A. Vem, N. Binder, G. Marcus, and A. Keller, "Sionna: An open-source library for next-generation physical layer research," *arXiv preprint arXiv:2203.11854*, 2022.

[30] A. Feriani, D. Wu, S. Liu, and G. Dudek, "Cebed: A benchmark for deep data-driven OFDM channel estimation," *arXiv preprint arXiv:2306.13761*, 2023.