MambaCSP: Hybrid-Attention State Space Models for Hardware-Efficient Channel State Prediction

Aladin Djuhera [1] **Haris Gacanin** [2] **Holger Boche** [1]

Abstract

Recent works have demonstrated that attention-based transformer and large language model (LLM) architectures can achieve strong channel state prediction (CSP) performance by capturing long-range temporal dependencies across channel state information (CSI) sequences. However, these models suffer from quadratic scaling in sequence length, leading to substantial computational cost, memory consumption, and inference latency, which limits their applicability in real-time and resource-constrained wireless deployments. In this paper, we investigate whether selective state space models (SSMs) can serve as a hardware-efficient alternative for CSI prediction. We propose MambaCSP, a hybrid-attention SSM architecture that replaces LLM-based prediction backbones with a linear-time Mamba model. To overcome the local-only dependencies of pure SSMs, we introduce lightweight patch-mixer attention layers that periodically inject cross-token attentions, helping with long-context CSI prediction. Extensive MISO-OFDM simulations show that MambaCSP improves prediction accuracy over LLM-based approaches by 9–12%, while delivering up to 3.0x higher throughput, 2.6x lower VRAM usage, and 2.9x faster inference. Our results demonstrate that hybrid state space architectures provide a promising direction for scalable and hardware-efficient AI-native CSI prediction.

1. Introduction and Motivation

Large language models (LLMs) have gained strong capabilities in coding, math, and complex reasoning, demonstrating their transformative potential across diverse domains. In future 6G networks, they can serve as foundational building blocks for embedding agentic capabilities across communication, connectivity, and artificial intelligence (AI) layers, thereby replacing or augmenting existing workflows (Jiang et al., 2026; Djuhera et al., 2026).

However, not all communication layers can readily benefit from such AI-native integration. Wireless signal processing systems operate under strict real-time constraints, which have traditionally limited the adoption of AI due to high computational and latency requirements (Guo et al., 2026). LLMs, in particular, contain billions of parameters and require substantial compute, making them unsuitable for real-time deployment on resource-constrained devices and in latency-sensitive applications where time to first token (TTFT) is critical (Dao et al., 2022). Overcoming these constraints is essential for enabling practical AI integration.

Among such latency-sensitive applications is channel state prediction (CSP), which forecasts future channel state information (CSI) from historical observations, thereby reducing overall pilot overhead. However, acquiring CSI via conventional channel estimation methods is challenging, particularly in high-mobility scenarios where shortened channel coherence times significantly increase the estimation overhead. Recent works (Liu et al., 2024; Fan et al., 2025; Cui et al., 2025) have explored LLM-based architectures for CSP, leveraging attention mechanisms to effectively capture long-range, non-local dependencies across time and frequency. Despite their strong performance, transformer-based models exhibit quadratic scaling in sequence length, resulting in substantial computational and memory overhead, as well as increased inference latency due to key-value (KV) caching. This makes them impractical for long-context CSI prediction and real-time deployment in wireless systems.

Selective state space models (SSMs), such as Mamba (Gu & Dao, 2024), offer a hardware-efficient alternative to LLMs by modeling sequences with linear complexity without KV caching. Specifically, state transitions are input-dependent and include a continuous-time inductive bias, making SSMs better suited for modeling wireless channel variations. However, pure SSMs process tokens recursively, which can limit their ability to capture non-local CSI dependencies.

In this paper, we address these limitations and propose **MambaCSP**, a hybrid-attention state space architecture for

[1]Technical University Munich [2]RWTH Aachen University. Correspondence to: Aladin Djuhera <aladin.djuhera@tum.de>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

hardware-efficient CSP. Our approach employs a Mamba-based backbone with lightweight patch-mixer attention layers that periodically inject cross-token attention, enabling modeling of long-range dependencies across CSI sequences while preserving the efficiency of SSMs. To this end, we generalize LLM-based CSP pipelines into a unified framework that enables efficient processing of CSI sequences across both model architectures. Our key findings are:

- **Performance**: MambaCSP consistently outperforms LLM-based and other neural baselines across diverse mobility scenarios in both TDD and FDD settings.
- **Efficiency**: MambaCSP achieves up to 3.0× higher throughput, 2.6× lower memory usage, and 2.9× faster inference compared to LLM-based prediction.
- **Hybrid-Attention Design**: MambaCSP's patch-mixer attention is particularly beneficial for FDD, where non-local, cross-frequency dependencies require token interactions beyond recursive state-space modeling.

2. System Model and CSI Prediction Task

In this section, we formally define the wireless system model and the corresponding channel state prediction task.

2.1. Wireless System Model

We consider a single-cell MISO-OFDM link, where a base station (BS) equipped with a uniform planar array (UPA) with N_t antennas serves U single-antenna user equipments (UEs). We follow a 5G NR FR1 OFDM numerology with carrier frequency $f_c = 2.4$ GHz, subcarrier spacing $\Delta f = 15$ kHz, and $N_{\text{sc}} = 12$ subcarriers per resource block (RB), such that each RB has a bandwidth of $B_{\text{RB}} = N_{\text{sc}}\Delta f = 180$ kHz. We consider two contiguous frequency bands of equal bandwidth for uplink (UL) and downlink (DL), each consisting of $K_{\text{UL}} = K_{\text{DL}} = 48$ RBs, resulting in a per-band bandwidth of 8.64 MHz. Time is organized in slots of duration $T_{\text{slot}} = 1$ ms and users move with velocities in the range of $v \in [10, 100]$ km/h. For each UE, we consider a sequence of $N_{\text{slot}} = P + L$ consecutive slots, where the first P slots form a *history window* and the remaining L slots form a *prediction window*. For the UL band, let $k \in \{1, \ldots, K_{\text{UL}}\}$ denote the RB index and $s \in \{1, \ldots, N_{\text{slot}}\}$ the slot index. The corresponding UL and DL narrowband MISO channel vectors on RB k and slot s are denoted as $\mathbf{h}_{k,s}^{\text{UL}}$ and $\mathbf{h}_{k,s}^{\text{DL}} \in \mathbb{C}^{N_t}$, respectively. The corresponding parametric representation of the frequency-selective and time-varying UL channel is given by

$$\mathbf{h}_{k,s}^{\text{UL}} = \sum_{\ell=1}^{L_p} \alpha_\ell \mathbf{a}_{\text{BS}}(\theta_\ell, \phi_\ell) e^{-j2\pi f_k \tau_\ell} e^{j2\pi f_{D,\ell} t_s}, \quad (1)$$

where α_ℓ, τ_ℓ, and (θ_ℓ, ϕ_ℓ) are the complex gain, delay, and angles of departure of path ℓ, $\mathbf{a}_{\text{BS}}(\cdot)$ is the BS response, and $f_{D,\ell}$ is the Doppler shift. We consider both time-division (TDD) and frequency-division (FDD) duplex modes.

2.2. Channel State Prediction Task

Given the UL history window over P slots, we aim to predict the future DL CSI over the next L slots across all RBs. To this end, let the UL history be defined as

$$\mathcal{H}_{\text{his}}^{\text{UL}} = \{\tilde{\mathbf{h}}_{k,s}^{\text{UL}} \mid k \in \mathcal{K}_{\text{CSI}}, s \in \mathcal{S}_{\text{CSI}}\} \quad (2)$$

with corresponding DL prediction target

$$\mathcal{H}_{\text{pre}}^{\text{DL}} = \{\mathbf{h}_{k,s}^{\text{DL}} \mid k \in \{1, \ldots, K_{\text{DL}}\}, s \in \{P+1, \ldots, P+L\}\}, \quad (3)$$

This defines a sequence-to-sequence prediction of the form $\hat{\mathcal{H}}_{\text{pre}}^{\text{DL}} = f_\Theta(\mathcal{H}_{\text{his}}^{\text{UL}})$, where f_Θ denotes a parameterized model with learnable parameters Θ, whose performance is evaluated via normalized mean square error (NMSE), i.e.,

$$\text{NMSE} = \mathbb{E}\left[\frac{\sum_{k,s} \|\hat{\mathbf{h}}_{k,s}^{\text{DL}} - \mathbf{h}_{k,s}^{\text{DL}}\|_2^2}{\sum_{k,s} \|\mathbf{h}_{k,s}^{\text{DL}}\|_2^2}\right]. \quad (4)$$

3. MambaCSP: Hybrid-Attention State Space Models for CSI Prediction

We first introduce a unified framework for CSI prediction and then present our proposed hybrid MambaCSP architecture, which combines the efficiency of SSMs with the expressivity of attention mechanisms.

3.1. Unified CSI Prediction Framework

Building on prior works (Liu et al., 2024; Fan et al., 2025), we generalize attention-based CSP pipelines into a unified framework that supports arbitrary sequence modeling architectures, including SSMs (see Fig. 1). The key components are outlined as follows.

3.1.1. Data Preprocessing

CSI data is complex-valued and needs to be converted into real-valued embeddings first. To this end, we extract frequency- and delay-domain representations, normalize the data, and partition it into temporal patches. We denote the historical UL CSI over P slots and K RBs as $\mathbf{H}_f \in \mathbb{C}^{K \times P}$ and its delay-domain representation as $\mathbf{H}_\tau = \mathbf{F}_K^H \mathbf{H}_f$, where $\mathbf{F}_K$ is the K-point DFT matrix. We then separate real and imaginary components and obtain $\mathbf{X}_f, \mathbf{X}_\tau \in \mathbb{R}^{2 \times K \times P}$. To ensure stable training across varying SNR conditions, we normalize each representation and reshape the tensors by merging feature dimensions, yielding $\tilde{\mathbf{X}}_f, \tilde{\mathbf{X}}_\tau \in \mathbb{R}^{2K \times P}$. To further reduce sequence length and compute, we partition the temporal dimension into non-overlapping patches

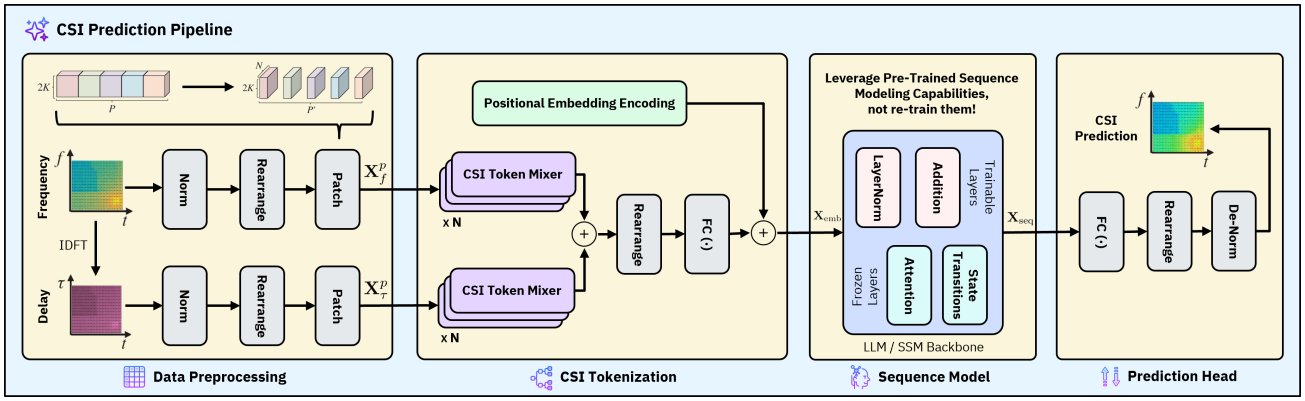

Figure 1. Unified CSI prediction pipeline: Historical UL CSI is converted into frequency/delay components, normalized, rearranged, and partitioned into temporal patches. CSI token mixer blocks then produce token embeddings for the sequence model. A final prediction head maps the outputs back to the complex CSI domain.

of size N (see Fig. 1a), resulting in

$$\mathbf{X}_f^p, \mathbf{X}_\tau^p \in \mathbb{R}^{2K \times N \times P'}, \tag{5}$$

where $P' = \lceil P/N \rceil$ denotes the number of patches.

3.1.2. CSI Tokenization

Preprocessed CSI patches need to be further tokenized into embeddings that are compatible with the input format of language models. To this end, each patch is processed by a *CSI Token Mixer* (see Fig. 2), which adaptively reweights features to emphasize informative multipath components while attenuating noisy ones. Let $\mathbf{X}_i$ denote the input tensor of a token mixer block. First, local feature extraction via convolutional $\mathrm{Conv}(\cdot)$ and $\mathrm{ReLU}(\cdot)$ layers is performed to capture joint temporal and frequency dependencies, i.e.,

$$\mathbf{X}_{\text{feat}} = \mathrm{Conv}(\mathrm{ReLU}(\mathrm{Conv}(\mathbf{X}_i))) \,. \tag{6}$$

To model the relative importance of each token, channel-wise gating with sigmoid activation $\sigma(\cdot)$ is applied

$$\mathbf{X}_{\text{gate}} = \sigma(\mathrm{FC}(\mathrm{ReLU}(\mathrm{FC}(\mathrm{GAP}(\mathbf{X}_{\text{feat}}))))) \,, \tag{7}$$

where $\mathrm{GAP}(\cdot)$ and $\mathrm{FC}(\cdot)$ denote global average pooling and fully connected layers. The resulting weights $\mathbf{X}_{\text{gate}} \in \mathbb{R}^{1 \times 1 \times P'}$ are then used to reweight each token, i.e.,

$$\mathbf{X}_{\text{sca}}[:,:,i] = \mathbf{X}_{\text{gate}}[i] \cdot \mathbf{X}_{\text{feat}}[:,:,i] \,, \tag{8}$$

before applying a residual connection, which preserves the original features and stabilizes gradient flow during training, i.e., $\mathbf{X}_o = \mathbf{X}_{\text{sca}} + \mathbf{X}_i$. Next, to jointly capture spectral correlations and multipath structure, we cascade multiple CSI Token Mixer blocks over the frequency/delay domains:

$$\mathbf{X}_{\text{tok}} = \mathrm{TM}^{(N)}(\mathbf{X}_f^p) + \mathrm{TM}^{(N)}(\mathbf{X}_\tau^p) \,, \tag{9}$$

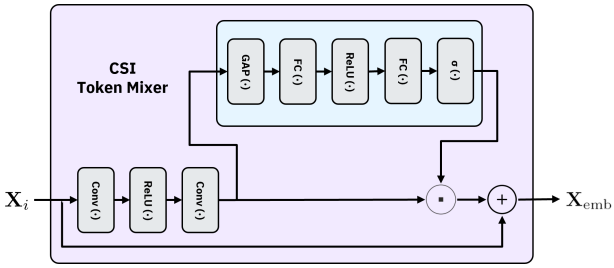

Figure 2. CSI token mixer module.

where $\mathrm{TM}^{(N)}(\cdot)$ denotes N cascaded token mixer blocks. In practice, we employ $N \in [2, 4]$ to balance modeling capacity and computational efficiency. The resulting tensor is rearranged into $\tilde{\mathbf{X}}_{\text{tok}} \in \mathbb{R}^{2KN \times P'}$ and projected into the model embedding dimension, yielding $\bar{\mathbf{X}}_{\text{tok}} \in \mathbb{R}^{F \times P'}$.

To preserve the temporal ordering, we follow (Liu et al., 2024) and add sinusoidal positional encodings defined as

$$\mathbf{X}_{\text{PE}}(i,j) = \begin{cases} \sin\left(\frac{j}{10000^{i/F}}\right) \,, & i \text{ even} \,, \\ \cos\left(\frac{j}{10000^{(i-1)/F}}\right) \,, & i \text{ odd} \,, \end{cases} \tag{10}$$

where i indexes the feature dimension and j the token position. The final CSI embeddings are then obtained as

$$\mathbf{X}_{\text{emb}} = \bar{\mathbf{X}}_{\text{tok}} + \mathbf{X}_{\text{PE}} \,. \tag{11}$$

3.1.3. Sequence Model Backbone

To model spatio-temporal dependencies across CSI token sequences, we leverage the pre-trained backbone's capacity to attend to complex multi-scale interactions rather than its learned semantic world knowledge. During training, we therefore keep the backbone's core layers frozen while only adapting lightweight task-specific components. In LLMs, this corresponds to freezing the multi-head attention and feed-forward blocks, while only fine-tuning LayerNorm and Addition layers (Liu et al., 2024; Fan et al., 2025). This

significantly reduces the overall training overhead and yields contextualized token representations given by

$$\mathbf{X}_{\text{seq}} = f_{\Theta_{\text{bb}}}(\mathbf{X}_{\text{emb}}) \in \mathbb{R}^{F \times P'} , \tag{12}$$

where $f_{\Theta_{\text{bb}}}(\cdot)$ denotes the sequence model backbone.

3.1.4. Prediction Head

In the final step, $\mathbf{X}_{\text{seq}}$ is mapped back to its original domain. We first apply a linear projection to obtain $\hat{\mathbf{X}} \in \mathbb{R}^{2K \times L}$ and reshape the predicted tensor into real/imaginary components of the form $\hat{\mathbf{X}} \in \mathbb{R}^{2 \times K \times L}$. We then de-normalize to recover the original CSI scale, i.e., $\hat{\mathbf{X}}_{\text{de}} = \sigma\, \hat{\mathbf{X}} + \mu$, where μ and σ denote the mean and standard deviation used during preprocessing. Finally, the predicted CSI is obtained as

$$\hat{\mathbf{H}}^{\text{DL}} = \hat{\mathbf{X}}_{\text{de}}[1, :, :] + j\, \hat{\mathbf{X}}_{\text{de}}[2, :, :] . \tag{13}$$

3.2. Hybrid Mamba Architecture

Mamba (Gu & Dao, 2024) extends classical SSMs by introducing input-dependent state transitions. Instead of fixed matrices $\mathbf{A}$ (state transition), $\mathbf{B}$ (input projection), and $\mathbf{C}$ (output projection), Mamba parameterizes these operators as functions of the input:

$$\mathbf{h}_t = \mathbf{A}(\mathbf{x}_t)\mathbf{h}_{t-1} + \mathbf{B}(\mathbf{x}_t)\mathbf{x}_t , \quad \mathbf{y}_t = \mathbf{C}(\mathbf{x}_t)\mathbf{h}_t . \tag{14}$$

This allows to dynamically control how information is propagated and updated at each time step, improving its ability to model non-stationary sequences. In addition, Mamba incorporates a continuous-time inductive bias via learned step sizes Δt, enabling adaptive temporal scaling of the state updates. This is particularly well-suited for Doppler-induced channel variations, which evolve continuously over time.

However, Mamba propagates information recursively through the hidden state. As a result, information from distant tokens must be compressed into $\mathbf{h}_t$, which can lead to degradation over long horizons. This can be limiting for CSI prediction, particularly in *a) high-mobility scenarios*, where rapid channel variations introduce non-local dependencies across time and frequency, and in *b) FDD settings*, where predicting DL CSI from UL CSI requires modeling complex cross-frequency relationships over distant tokens.

To address these limitations, we propose **MambaCSP**, a *hybrid-attention* extension of Mamba (see Fig. 3). Specifically, we introduce sparse *patch-mixer attention layers*, which operate on the tokenized CSI patches every k blocks and perform lightweight cross-token interactions using a small number of attention heads H (typically 2–4). Given $\mathbf{X}_{\text{emb}} \in \mathbb{R}^{F \times P'}$, patch-mixer attention is computed as

$$\text{Attn}(\mathbf{X}_{\text{emb}}) = \text{softmax}\left(\frac{\mathbf{Q}^\top \mathbf{K}}{\sqrt{d_h}}\right)\mathbf{V}^\top , \tag{15}$$

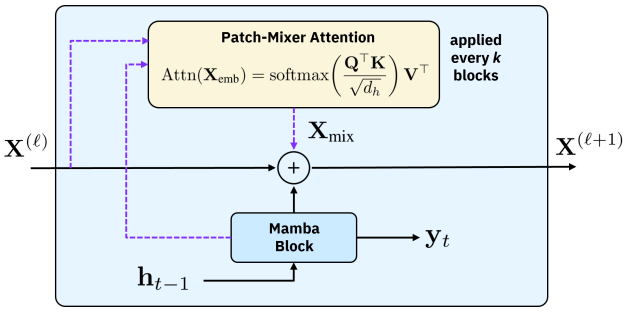

Figure 3. Hybrid-attention MambaCSP architecture at layer l.

where d_h denotes the head dimension and $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are the query, key, and value projections obtained from $\mathbf{X}_{\text{emb}}$ via learned linear mappings. The resulting representations are concatenated and projected back via $\mathbf{W}_O \in \mathbb{R}^{F \times F}$, i.e.,

$$\mathbf{X}_{\text{mix}} = \text{Concat}\big(\text{Attn}_1, \ldots, \text{Attn}_H\big)\mathbf{W}_O . \tag{16}$$

The attention output is then inserted sparsely every k Mamba blocks through a residual update, i.e.,

$$\mathbf{X}^{(\ell+1)} = \mathbf{X}^{(\ell)} + \mathbf{X}^{(\ell)}_{\text{mix}} , \qquad \ell \in \{k, 2k, \ldots\} . \tag{17}$$

This restores global context modeling while maintaining near-linear complexity. During training, patch-mixer attentions are learned jointly with projection and LayerNorm layers, while the majority of the backbone remains frozen.

3.3. Computational Complexity and Efficiency

Plain Mamba's state-space updates process sequences one step at a time with recurrence. Thus, for a token sequence of length P' and hidden dimension F, compute scales linearly as $\mathcal{O}(P'F)$. Since Mamba does not construct a full pairwise attention matrix and does not require KV caching, its memory usage also scales linearly with sequence length. For our hybrid MambaCSP, patch-mixer attentions are inserted only intermittently and use a small number of heads. Thus, for L_M Mamba blocks and $\lfloor L_M/k \rfloor$ patch-mixer layers, where in practice $\lfloor L_M/k \rfloor \ll L_M$, the overall complexity is

$$\mathcal{O}(L_M P'F) + \mathcal{O}\left(\left\lfloor \tfrac{L_M}{k} \right\rfloor P'^2 F\right) \;\approx\; \mathcal{O}(L_M P'F) , \tag{18}$$

such that MambaCSP retains near-linear scaling overall. In contrast, LLM backbones require self-attention in *every layer*. As a result, compute cost scales quadratically as $\mathcal{O}(P'^2 F)$, while memory is dominated by the attention matrix of size $P' \times P'$, i.e., $\mathcal{O}(P'^2)$. Although freezing transformer layers reduces the number of trainable parameters, it does not remove the quadratic self-attention computation in the forward pass. Therefore, MambaCSP provides a more favorable complexity–efficiency trade-off for long CSI histories and low-latency deployment.

4. Experimental Setup

To evaluate MambaCSP, we generate 8000 training, 2000 validation, and 10000 test samples for both TDD and FDD scenarios using the QuaDRiGa (Jaeckel et al., 2014) simulation engine. We benchmark our hybrid MambaCSP against plain Mamba, LLM-based prediction (Liu et al., 2024), and classical neural baselines including CNN (Safari et al., 2019), RNN (Jiang & Schotten, 2019), and LSTM (Jiang & Schotten, 2020). To ensure fairness, we employ a 130M parameter size Mamba 2 model (Dao & Gu, 2024), whose scale is comparable to the 125M parameter size GPT-2 (Radford et al., 2019) model used in (Liu et al., 2024). We further note that increasing the backbone model size or changing the architecture, e.g., to GPT-OSS (OpenAI, 2025) or other larger models, yields at most marginal gains for CSP, since the task does not primarily rely on semantic world knowledge or reasoning abilities (see App. C). We train each model for 10 epochs using Adam with $\beta = (0.9, 0.999)$, batch size 256, and learning rate 10^{-3}. We refer to App. B for more details on training and data generation.

5. Results and Discussion

We evaluate MambaCSP in terms of prediction accuracy and hardware efficiency, and perform additional ablations.

5.1. Prediction Accuracy

Fig. 4 and Fig. 5 compare the NMSE of all considered baselines, showing that MambaCSP outperforms LLM-based CSP in both duplex modes. For TDD, MambaCSP achieves 5% lower NMSE than plain Mamba and 9% lower NMSE than the LLM, indicating that state-space modeling already captures most relevant temporal structures in TDD, while patch-mixer attention provides a moderate but non-negligible gain. For FDD, the relative benefit of MambaCSP becomes more pronounced. It consistently outperforms plain Mamba by 17% and the LLM by 12% across the full velocity range, showing that patch-mixer attention is particularly beneficial in FDD when non-local cross-frequency dependencies arise due to the UL-to-DL prediction gap, which causes plain Mamba to underperform, particularly at higher velocities. Furthermore, MambaCSP, plain Mamba, and the LLM significantly outperform CNN, RNN, and LSTM baselines, suggesting that sequence models are better at predicting spatio-temporal CSI dependencies.

5.2. Throughput

Fig. 6 shows the training throughput as the sequence length P' increases. As expected, throughput decreases noticeably as the input history becomes longer. Overall, LLM-based CSP degrades much faster than Mamba-based variants. In particular, MambaCSP achieves $2.7\times$, $2.9\times$, and $3.0\times$ hi-

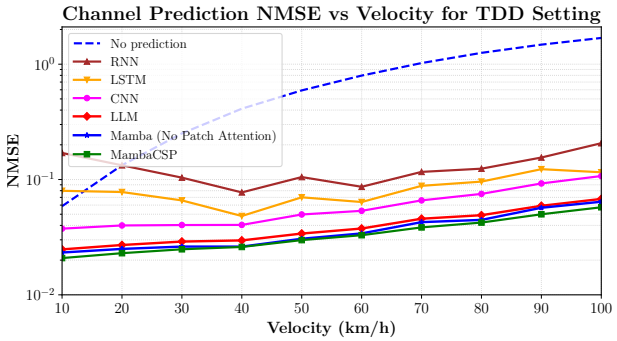

Figure 4. NMSE results for TDD. MambaCSP outperforms plain Mamba and the LLM. NMSE increases with increasing velocity.

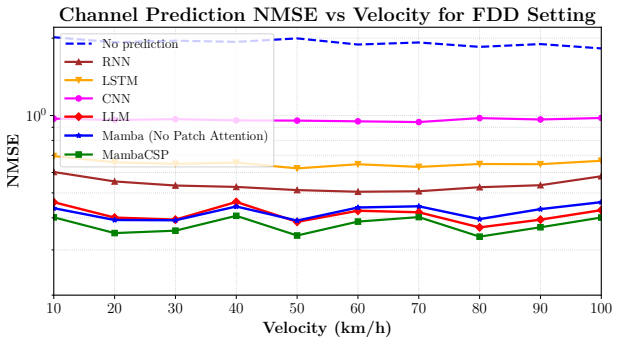

Figure 5. NMSE results for FDD. MambaCSP outperforms plain Mamba and the LLM. NMSE remains approximately flat.

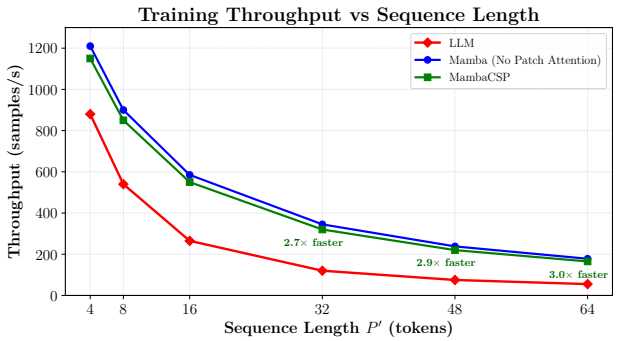

Figure 6. Throughput for different sequence lengths. MambaCSP achieves up to $3\times$ more samples/s for longer CSI patches.

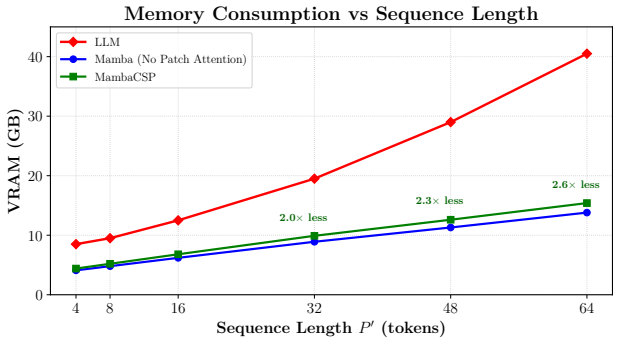

Figure 7. Memory consumption for different sequence lengths. LLM grows faster with sequence length due to KV caching.

gher throughput than the LLM at $P' \in \{32, 48, 64\}$, respectively. This shows that the proposed patch-level attention introduces only a minor throughput overhead.

5.3. Memory Consumption

Fig. 7 reports the peak memory usage. In line with our theoretical observations, the LLM exhibits a much steeper increase in memory consumption, driven by the quadratic attention matrix and KV caching. For short input histories up to $P' = 8$, the memory gap between LLM and Mamba-based models remains approximately constant, but it widens steadily as P' grows. Specifically, the LLM requires $2.0\times$, $2.3\times$, and $2.6\times$ more VRAM than the Mamba variants for $P' \in \{32, 48, 64\}$. In contrast, plain Mamba and MambaCSP remain close and scale linearly in memory, substantiating our analysis that patch-level attention incurs only a negligible memory overhead.

5.4. Inference Latency

Fig. 8 compares the inference latency per forward pass across sequence lengths. Similar to memory usage, latency increases with P' for all models. MambaCSP is about $2.0\times$, $2.6\times$, and $2.9\times$ faster than the LLM at $P' \in \{32, 48, 64\}$, respectively, while remaining close to plain Mamba, thereby preserving the latency benefits of the state-space backbone.

5.5. NMSE vs Input Sequence Length

Table 1 reports the FDD NMSE for MambaCSP and LLM backbones as P' increases. Overall, a longer CSI history improves prediction (with diminishing returns at $P' = 32$) as additional past observations expose more spatio-temporal structure that is useful for inferring future DL CSI. This supports our motivation for increasing P', especially in FDD, which benefits from longer-range CSI dependencies.

5.6. Ablations on Patch-Mixer Attention Layers

Table 2 ablates the impact of the patch-mixer attention layers in MambaCSP. Overall, injecting cross-attention with $H = 2$ heads every $k = 4$ Mamba blocks is sufficient to restore non-local sequence modeling capabilities for complex CSI patches, while providing a favorable trade-off between prediction accuracy and efficiency. Further increasing the number of heads or inserting attention more frequently yields only marginal gains at cost of increased complexity.

In summary, our results demonstrate that MambaCSP achieves a substantially better accuracy–efficiency trade-off than LLM-based CSP, particularly for longer CSI histories. Its lower latency and memory footprint make it especially attractive for real-time deployment on low-resource devices. Further reducing memory and compute through quantization and model pruning is an interesting future work.

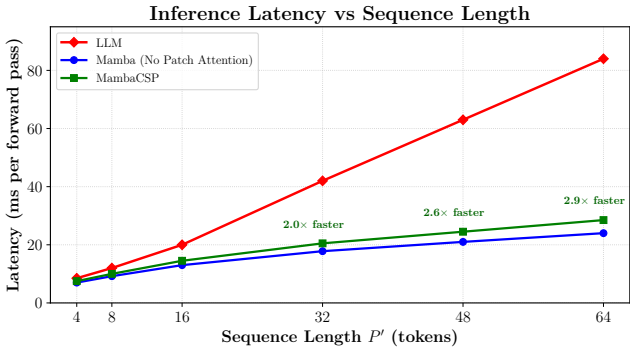

Figure 8. Latency for different sequence lengths. Mamba backbones scale linearly compared to the LLM (up to $2.9\times$ slower).

Table 1. NMSE versus sequence length P' for LLM and MambaCSP (FDD). Increasing P' improves prediction with diminishing returns at larger contexts.

Seq. Length P'	LLM NMSE	MambaCSP NMSE	Relative Gain
4	0.503	0.468	7.0%
8	0.462	0.422	8.7%
16	0.425	0.366	13.9%
32	0.392	0.331	15.6%
48	0.381	0.318	16.5%

Table 2. Ablation of the patch-mixer attention layers in MambaCSP. Sparse attention improves FDD more than TDD. In practice, we use $k = 4$ and $H = 2$ as a favorable accuracy-efficiency trade-off.

Patch-Mixer	Interval k	Heads H	TDD NMSE ↓	FDD NMSE ↓
None	–	–	0.0374	0.4222
Yes	4	2	0.0346	0.3750
Yes	4	4	**0.0345**	**0.3736**
Yes	2	4	0.0345	0.3731

6. Conclusion

In this paper, we proposed MambaCSP, a hybrid-attention state space model architecture as a hardware-efficient alternative to LLM-based CSI prediction. MambaCSP replaces the LLM backbone in existing CSP pipelines with a Mamba-based sequence model and augments it with sparse patch-mixer attention layers to better capture long-range CSI token dependencies while preserving the linear-time efficiency of state space models. Our experiments show that MambaCSP achieves better accuracy than LLM-based prediction in both TDD and FDD settings, while providing substantially higher throughput, lower memory consumption, and lower inference latency. Our findings thus point to a key takeaway: LLMs are not the final architectural choice for long-context CSI prediction, and combining efficient state space models with lightweight attention is a promising path toward scalable and real-time AI-native CSI prediction in future wireless systems.

Acknowledgments

This work was supported in part by the German Federal Ministry of Research, Technology and Space (BMFTR) within the research hubs 6G-life (Grant 16KISK002) and 6GEM+ (Grant 16KIS2409K) through the projects GEM-X (Grant 16KISS004K), VICTOR6G (Grant 16KIS2547), QD-CamNetz (Grant 16KISQ077), and AISAC (Grant 16KIS2462). This work was also supported by the Bavarian Ministry of Science and the Arts through the project Next Generation AI Computing (gAIn), by the Bavarian Ministry of Economic Affairs, Regional Development and Energy through the project 6G Future Lab Bavaria, and by the German Research Foundation (DFG) within the Centre for Tactile Internet with Human-in-the-Loop (CeTI) as part of Germany's Cluster of Excellence (EXC 2050/2, ID 390696704).

Impact Statement

This paper presents work whose goal is to advance the field of machine learning and wireless communications. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Boateng, G. O., Sami, H., Alagha, A., Elmekki, H., Hammoud, A., Mizouni, R., Mourad, A., Otrok, H., Bentahar, J., Muhaidat, S., et al. A Survey on Large Language Models for Communication, Network, and Service Management: Application Insights, Challenges, and Future Directions. *IEEE Communications Surveys & Tutorials*, 2025.

Cui, Y., Guo, J., Wen, C.-K., Jin, S., and Tong, E. Exploring the Potential of Large Language Models for Massive MIMO CSI Feedback. *arXiv preprint arXiv:2501.10630*, 2025.

Dao, T. and Gu, A. Mamba-2 130M Model Weights. `https://huggingface.co/state-spaces/mamba2-130m`, 2024.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness, 2022. URL `https://arxiv.org/abs/2205.14135`.

Djuhera, A., Koch, F., and Binotto, A. Joint Partitioning and Placement of Foundation Models for Real-Time Edge AI. *IEEE ICNC*, 2026.

Fan, S., Liu, Z., Gu, X., and Li, H. CSI-LLM: A novel Downlink Channel Prediction Method Aligned with LLM Pre-Training. In *IEEE Wireless Communications and Networking Conference*, 2025.

Gu, A. and Dao, T. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. In *COLM*, 2024.

Guo, J., Cui, Y., Jin, S., and Zhang, J. Large AI Models for Wireless Physical Layer. *IEEE Communications Magazine*, 2026.

Jaeckel, S., Raschkowski, L., Börner, K., and Thiele, L. QuaDRiGa: A 3-D Multi-Cell Channel Model with Time Evolution for Enabling Virtual Field Trials. *IEEE Transactions on Antennas and Propagation*, 2014.

Jiang, F., Pan, C., Wang, K., Michiardi, P., Dobre, O. A., and Debbah, M. From Large AI Models to Agentic AI: A Tutorial on Future Intelligent Communications. *IEEE JSAC*, 2026.

Jiang, W. and Schotten, H. D. Neural Network-Based Fading Channel Prediction: A Comprehensive Overview. *IEEE Access*, 2019.

Jiang, W. and Schotten, H. D. Deep Learning for Fading Channel Prediction. *IEEE Open Journal of the Communications Society*, 2020.

Liu, B., Liu, X., Gao, S., Cheng, X., and Yang, L. LLM4CP: Adapting Large Language Models for Channel Prediction. *Journal of Communications and Information Networks*, 9 (2):113–125, 2024.

OpenAI. GPT-OSS-120B & GPT-OSS-20B Model Card, 2025. URL `https://arxiv.org/abs/2508.10925`.

Radford, A. et al. Language Models are Unsupervised Multitask Learners. *OpenAI Blog*, 2019.

Safari, M. S., Pourahmadi, V., and Sodagari, S. Deep UL2DL: Data-Driven Channel Knowledge Transfer from Uplink to Downlink. *IEEE Open Journal of Vehicular Technology*, 2019.

Zhou, H., Hu, C., Yuan, Y., Cui, Y., Jin, Y., Chen, C., Wu, H., Yuan, D., Jiang, L., Wu, D., et al. Large Language Model (LLM) for Telecommunications: A Comprehensive Survey on Principles, Key Techniques, and Opportunities. *IEEE Communications Surveys & Tutorials*, 27(3):1955–2005, 2024.

A. Related Work

LLMs have recently received growing attention in wireless communications. Beyond their success in natural language processing, they have been explored for semantic communications, network management, edge intelligence, CSI feedback, and channel prediction (Jiang et al., 2026; Zhou et al., 2024; Boateng et al., 2025). A key motivation is that large

pre-trained models provide strong sequence modeling and transfer learning capabilities, which may be useful for complex wireless data beyond text.

A standard application in wireless communications is CSI prediction, which aims to forecast future channel realizations from historical observations in order to reduce estimation overhead and mitigate channel aging. Recent works have shown that transformer- and LLM-based models can outperform classical neural baselines by better capturing long-range temporal and cross-frequency dependencies.

Among the closest works, Liu et al. (2024) proposed LLM4CP, one of the first methods to adapt a pre-trained GPT-2 backbone to channel prediction. Their approach predicts future downlink CSI from historical uplink CSI and introduces dedicated preprocessing, embedding, and output modules to bridge the gap between CSI and the LLM feature space. It further combines frequency- and delay-domain processing and shows that most GPT-2 parameters can be frozen during fine-tuning.

A related approach is CSI-LLM (Fan et al., 2025), which aligns channel prediction more explicitly with the autoregressive next-token objective used in causal LLM pre-training. Rather than relying on fixed-length input-output mappings, CSI-LLM tokenizes CSI sequences at the time-step level and supports variable-length historical inputs at inference time. This design is motivated by better alignment with the pre-training paradigm of language models and is shown to improve robustness in continuous multi-step prediction.

A third related direction applies LLMs to CSI processing beyond prediction. For example, Cui et al. (2025) study pre-trained architectures for massive MIMO CSI feedback and interpret CSI reconstruction through an analogy to language error correction. Their framework likewise relies on customized preprocessing, embedding, and post-processing modules around a mostly frozen GPT-2 backbone. Although the task is CSI feedback rather than prediction, it provides further evidence that pre-trained language models can transfer useful sequence modeling capabilities to wireless signal processing tasks.

Despite these advances, existing LLM-based wireless methods still retain important limitations. Most notably, GPT- and transformer-based backbones preserve the quadratic complexity of self-attention, which increases memory usage and latency with sequence length and limits their practicality for long CSI histories and real-time deployment. Moreover, while these methods demonstrate the usefulness of large pre-trained sequence models, they leave open the question of whether comparable or better CSI prediction performance can be achieved with architectures that are more hardware-efficient by design.

Our work addresses this gap by replacing the transformer-style backbone with a selective state space model and introducing sparse cross-attention to recover non-local interactions when needed. In this way, MambaCSP retains the benefits of strong sequence modeling highlighted by prior LLM-based CSI prediction works, while achieving a more favorable accuracy–efficiency trade-off for long-context CSI prediction. More broadly, our results suggest that the main benefit in CSI prediction stems less from semantic world knowledge and more from having an expressive sequence model with the right inductive bias for temporal and cross-frequency channel structure. This further motivates future work on quantization, pruning, and other compression strategies to improve the practicality of AI-native CSI prediction in resource-constrained wireless systems.

B. Extended Details on Experimental Setup

To evaluate MambaCSP, we generate a comprehensive CSI dataset with QuaDRiGa (Jaeckel et al., 2014) for the 3GPP TR 38.901 UMa NLOS channel. The OFDM numerology and all remaining parameters follow our system model in Sec. 2. The BS is placed at $(0, 0, 30)$ m and is equipped with a 4×4 dual-polarized UPA. For each UE, we simulate a frame of $N_{\text{slot}} = 20$ slots, where the first $P = 16$ slots form the history window and the remaining $L = 4$ slots form the prediction window. For each speed realization, we randomly place 10 UEs in an annulus between 20 m and 50 m around a cluster center at $(200, 0, 1.5)$ m and generate linear user tracks.

In 5G NR, channel estimation follows the *Demodulation Reference Signal* (DMRS) pilot pattern, where pilots are transmitted on a sparse subset of RBs and symbols. Formally, let $\mathcal{K}_{\text{CSI}} \subseteq \{1, \ldots, K_{\text{UL}}\}$ denote the set of RBs carrying pilot signals and let $\mathcal{S}_{\text{CSI}} \subseteq \{1, \ldots, P\}$ denote the corresponding time indices. The observed CSI is given by

$$\tilde{\mathbf{h}}_{k,s}^{\text{UL}} = \mathbf{h}_{k,s}^{\text{UL}} + \mathbf{n}_{k,s} \ , \quad (k, s) \in \mathcal{K}_{\text{CSI}} \times \mathcal{S}_{\text{CSI}} \ , \qquad (19)$$

where $\mathbf{n}_{k,s} \sim \mathcal{CN}(0, \sigma_n^2 \mathbf{I})$ models estimation noise. In our experiments, we employ a frequency-selective pilot pattern with temporal spacing $\Delta t_{\text{DMRS}} = 2\Delta t$ and frequency spacing $\Delta k = 1$ RB. Note that, depending on the scenario, different DMRS pilot patterns can be configured, which control the corresponding pilot density. However, our initial results showed that both LLM and Mamba architectures are robust for any such configuration. Nevertheless, DMRS-based pilot optimization is an interesting future work.

In total, we generate 8000 training, 2000 validation, and 10000 test samples for both TDD and FDD. We plan on releasing both our training code and data generation pipeline upon acceptance.

Table 3. Comparison of different backbones. Larger models provide only marginal gains for CSP.

Backbone	Params	VRAM (GB)	TDD NMSE ↓	FDD NMSE ↓
GPT-2 Small	124M	6.8	0.050	0.425
GPT-OSS	1.3B	18.9	0.049	0.423
Mamba2-130M	130M	5.1	0.049	0.422
Mamba2-780M	780M	11.8	0.048	0.420

C. Scaling Backbone Architectures

Table 3 shows that scaling the backbone architecture and parameter count yields only marginal improvements for the CSI prediction task.

For transformer-based backbones, changing the architecture and increasing the model size, e.g., from GPT-2 Small (124M) to GPT-OSS 1.3B, improves NMSE only slightly, namely from 0.050 to 0.049 in TDD and from 0.425 to 0.423 in FDD, despite a nearly threefold increase in VRAM usage. A similar trend is observed for state space models, where scaling Mamba2 from 130M to 780M reduces NMSE only from 0.049 to 0.048 in TDD and from 0.422 to 0.420 in FDD, while more than doubling memory consumption.

These results suggest that the key sequence modeling capabilities required to capture the relevant temporal and cross-frequency channel structure are already sufficiently available in comparatively small backbones. In other words, once the backbone provides adequate long-range sequence modeling capacity, further gains from increasing parameter count or changing the architecture family appear to be limited.

We further hypothesize that these results indicate considerable room for model compression, and that smaller or quantized backbones may remain competitive for CSP while further improving the overall accuracy–efficiency trade-off in resource-constrained deployments.