Exploiting Temporal Priors for Efficient Real-time Compression and Feedback of Wireless Channels

Anonymous Author(s)
Affiliation
Address
email

Abstract

Machine learning based compression frameworks are rapidly gaining popularity as the demand for efficient storage, processing, and transmission of large-scale data continues to grow across diverse applications such as video streaming and IoT. Recently, such frameworks have also sparked significant interest in wireless communications and the task of ML based wireless channel compression is currently one of the use cases being explored by the international wireless standards body, 3GPP, for standardization. In wireless communication systems, each user device or user equipment (UE) typically estimates the wireless channel between the transmitting base station (BS) and itself and feeds back information related to the estimated channel state information (CSI) to the serving BS, which may then be utilized for downstream processing. While the current 5G communication stack employs a combination of matrix factorization and quantization approaches to compress the CSI, autoencoders (AE) have emerged as a viable option to compress the estimated spatial-frequency (SF) channel sample and send it back to the base station for reconstruction. Although the AE-based approaches have shown acceptable CSI reconstruction performance, there is still a large room for further improvement, both from an overhead reduction as well as reconstruction performance perspectives. This paper proposes a new AE framework that leverages the temporal correlation properties of the channel to enhance the compression process. In particular, we propose an AE framework that performs temporal-spatial-frequency (TSF) compression by utilizing priors based on historical CSI samples to efficiently compress the current estimated CSI sample. End-to-end simulation results on a realistic test bench demonstrate the superiority of the proposed TSF compression approach relative to the state-of-the-art methods.

1 Introduction

Compression tasks are critical across numerous domains, including but not limited to, telecommunications, healthcare, video streaming and IoT, where efficiently storing and transmitting large amounts of data is essential for maintaining performance and reducing costs. Compression techniques like compressive sensing, matrix and tensor decompositions have long been used to reduce data dimensionality and extract meaningful features; however, autoencoders (AE) offer a distinct advantage by learning non-linear representations directly from raw data. Autoencoders can preserve essential features while reducing redundancy, ultimately leading to more efficient, scalable and cost-effective solutions in domains such as video processing, wireless communications, e.g., sensor data analytics and wireless channels.

For time-series data compression, AEs can potentially be more efficient by leveraging Recurrent Neural Networks and Transformers to exploit the inherent temporal correlations within the data to

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

achieve superior compression. Depending on the application, the availability of data and the set of underlying constraints may differ. While some applications may have the flexibility to utilize multiple or all temporal data samples before compressing and transmitting the data, others may require real-time compression and transmission as soon as a new data sample arrives. The latter is indeed more challenging from a compression perspective. For example, video data inherently benefits from temporal correlations between consecutive samples (GOPs or group of pictures), where the content remains similar over short periods, allowing compression techniques to exploit these redundancies for efficient compression. In contrast, wireless data, e.g., sensor data or cellular channels, presents a different challenge, as it requires real-time processing and transmission of data arriving sequentially over time. In other words, while video has all or at least a few temporally-correlated samples available apriori for analysis and compression, wireless systems must compress and transmit any observed or estimated data without delay, making real-time compression more complex and time-sensitive.

In this work, we introduce a novel autoencoder framework that can efficiently handle real-time compression of time-series data. While the proposed framework can be widely applicable to several time-series data compression use-cases, e.g., wireless sensor networks, IoT, etc., we here adopt the wireless channel compression as an example use-case for real-time time-series data compression. The wireless channel data compression has recently gained popularity in the machine learning domain and it is one of the few ML use-cases that is currently considered as a study item in the 3GPP wireless standards. In the context of wireless channel compression, the processing and transmission of each sample has to be completed within 1-2 milliseconds (ms) while the time difference between two consecutive samples can range from 5 ms to 20 ms. Thus, each sample has to be compressed and transmitted before the next sample arrives. The wireless channel compression is a crucial task in massive multiple-input-multiple-output (MIMO) systems.

Massive MIMO is a leading technology that has the potential to meet the data rate requirements in the next-generation wireless communication systems [4]. The key advantage of employing large antenna arrays in massive MIMO systems is the capability of achieving striking performance gains in multiuser MIMO systems. However, achieving such gains is contingent on the availability of accurate channel state information (CSI) at the transmitter. This requires the receiver, e.g., user equipment, to send back the estimated CSI to the transmitter, e.g., base station. The CSI feedback process incurs additional system overhead that scales up with the number of transmit antennas, the number of receive antennas, and the number of allocated frequency resources, thereby resulting in a considerable uplink overhead that can impact the system performance.

Autoencoder (AE)-based CSI compression and feedback has gained significant popularity in the wireless domain [2, 3, 6]. The current 5G compliant communication systems under deployment across the world employ a combination of matrix factorization and quantization approaches to compress and feedback the CSI [1]. AE-based solutions have the potential to offer a favorable trade-off between the CSI feedback overhead and reconstruction performance. Existing AE-based approaches operate on solely on the spatial-frequency (SF) channel samples, where the spatial components represent the number of transmit side and receive side antennas while the frequency components refer to the number of orthogonal frequencies over which the data transmission happens. In such setups, the estimated CSI sample at each user is compressed using an encoder network, and then the compressed version is sent to the base station for reconstruction using a decoder network. This process is repeated for every CSI reporting instance, where each collected sample over time is compressed independently. While SF compression was shown to provide acceptable reconstruction quality, there is still large room for further improvement in the reconstruction performance or reduction in the signaling overhead.

One way to further improve the performance of the SF compression approach is to leverage the temporal correlation properties of the channel in the compression process. Exploiting the CSI temporal correlation on the top of SF compression has the potential to provide further i) improvement in the reconstruction performance for a given overhead, ii) reduction in the overhead for a given performance and/or ii) improvement in both performance and overhead relative to the SF compression.

To address the CSI feedback overhead reduction problem, we propose a new AE framework that seeks to efficiently compress the current spatial-frequency CSI sample by utilizing the temporal correlation of the channel in the compression process – this technique is referred to as Temporal-Spatial-Frequency (TSF) compression. Preliminary simulation results on a realistic 5G compliant test bench show that exploiting the past collected CSI samples in the compression task can result in considerable throughput gains relative to SF compression and state-of-the-art methods.

2 Problem Statement

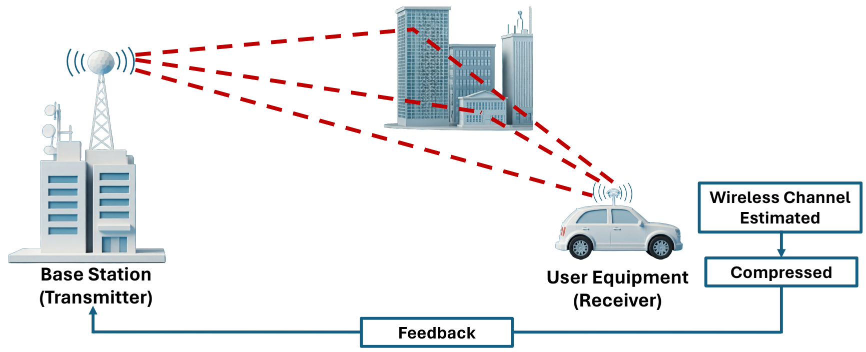


Figure 1: A representation of the downlink communication with base station (BS) as the transmitter and a self driving car as the receiver (user equipment(UE)). The transmitted signal from the BS, travels through multiple paths (red dotted lines) before being received at the UE. The wireless channel $\mathbf{H}_n$ represents the overall impulse response associated with the signal propagation and is estimated at the UE. The channel is then compressed and sent back to the BS for further downstream processing.

Consider a downlink data transmission setup where a single BS equipped with N_t transmit antennas is serving (or transmitting) to a single UE with N_r receive antennas. The UE receives its data over multiple frequency components, e.g., sub-carriers (orthogonal frequencies). The UE needs to estimate and transmit the channel tensor, $\mathbf{H}_n \in \mathbb{C}^{N_r \times N_t \times N_c}$, where N_c denotes the number of frequency components. The goal is then to compress the estimated channel $\mathbf{H}_n$ assuming that the user and base station may have access to upto N historical samples, i.e., $\mathbf{H}_{n-1}, \cdots, \mathbf{H}_{n-N}$.

3 TSF Framework and Model Architecture

In this section, we introduce the framework designed to enable real-time compression and feedback of the current channel instance between a transmission and reception node, whilst leveraging historical channel information to improve reconstruction performance.

3.1 System Framework

Given a maximum look-back size of N historical samples, we consider a set of $N+1$ encoder-decoder pairs, one associated with each possible value of the available past samples $\{0, ..., N\}$. Thus for the channel sample at time n, i.e. $\mathbf{H}_n$, the k-th model is utilized wherein, $k = (n \bmod N+1)$. The k-th model utilizes k past channel samples for both, encoding and decoding. Thus, for a value of $k = 0$, no past information is utilized for the compression, and the channel $\mathbf{H}$ is compressed standalone.

A typical compression pipeline consisting of an encoder and a quantizer is used by the UE to obtain a compressed representation of the channel, $\mathbf{z}_n$ with dimensionality D_k. The UE feeds back the compressed representation, $\mathbf{z}_n$, to the base station which decompresses it using its decoder. The UE is further equipped with the same decoder model being utilized by the BS, this allows both UE and BS to have the same reference for the past samples. Post decoding, both BS and UE store the reconstructed channel in a buffer. The reconstructed channels stored in the UE-side buffer are utilized as priors for compressing the channel samples at the next time instance. Since the UE and BS utilize the same priors or past samples, the encoder and decoder remain synchronized (in the absence of packet loss and noise) allowing for better compression and reconstruction of the channel data. A diagrammatic representation of the compression framework can be seen in Fig. 2.

As mentioned earlier, the selection of the encoder-decoder pair for a specific n is governed by $k = (n \bmod N+1)$. This setup ensures any noise, error, or packet loss that may have been introduced during transmission (or feedback) is not accumulated for more than N samples. Further, having $N+1$ models, each dedicated to a specific look-back period, enables us to identify and analyze the maximum performance or improvement such a setup could achieve. However, this is achieved at the expense of having multiple models with potential redundancies across their learned layers/weights.

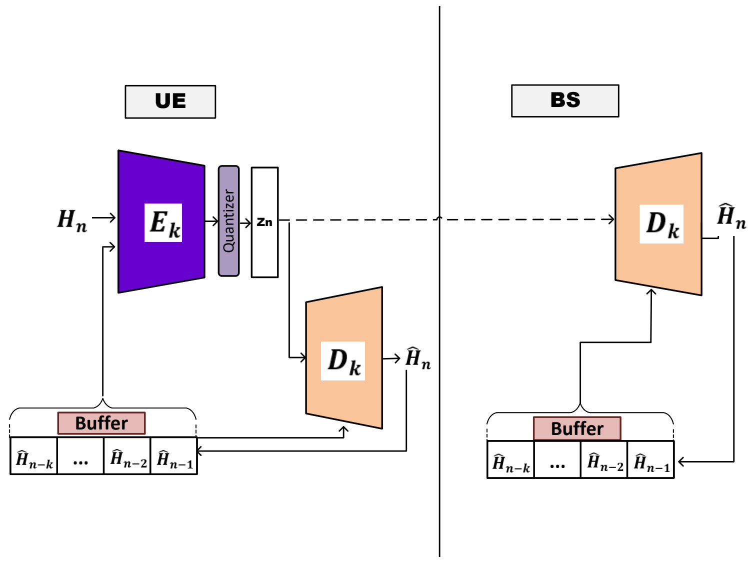


Figure 2: Proposed TSF Autoencoder framework. The UE utilizes an encoder model along with past reconstructed channel samples to compress the current channel sample. The NW and UE both utilize the same decoder network to reconstruct the channel sample with the synchronized information about the past channel data.

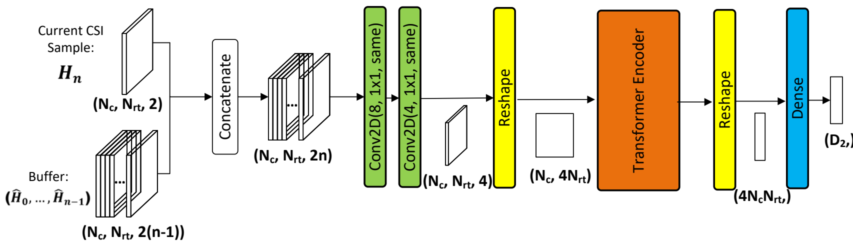
3.2 TSF Model Architecture

The full architecture of the encoder and decoder blocks are detailed in Fig. 3. The previously decoded samples $(\widehat{\mathbf{H}}_{n-1}, \widehat{\mathbf{H}}_{n-2}, ..., \widehat{\mathbf{H}}_{n-k})$ are combined with the current channel sample $\mathbf{H}_n$ at the input of the encoder. The input tensor has shape $\mathbf{H}_{in} \in \mathbb{R}^{N_c \times N_{rt} \times 2(k+1)}$, where $(k+1)$ represents the current and the past k channel samples, 2 represents the real and imaginary parts of the complex channels and $N_{rt} = N_r * N_t$. We compute 2D convolutions with a 1x1 kernel, so that each element in the output is derived from a combination of elements at the same position in the input, across all the current and past samples. This representation is reshaped appropriately for input to the transformer block shown in Fig. 3c. The multi-head self-attention block extracts pairwise similarities between frequency sub-bands, N_c resulting in an $(N_c \times N_c)$ attention matrix that is used to weight the full input tensor. Channel samples typically exhibit high correlation across sub-bands therefore the attention mechanism can be viewed as removing redundancy by focusing on the most relevant sub-bands. This representation is passed to a position-wise feed-forward layer that transforms the features of each sub-band independently. The output of the transformer block is reshaped and passed through a final dense layer and discretized using 2-bit scalar quantization, giving $\mathbf{z}_n$.

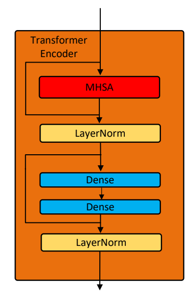
At the decoder (Fig. 3b) the prior samples are introduced into the model post the transformer. The objective being, that the earlier layers in the encoder learn to filter out the redundant or correlated information across samples and the transformer part of the encoder and decoder models only focus on compressing and reconstructing the non-redundant information. The redundant information from past samples can then be reintroduced into the data at the final stages of the decoding utilizing the past samples and 2D convolution layers.

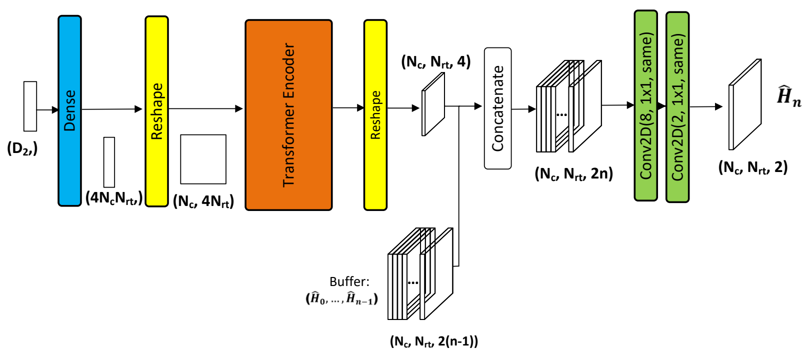
3.3 Training

As mentioned earlier, we train a total of $N+1$ encoder-decoder pairs $\{(E_0, D_0), ..., (E_{N,}, D_{N,})\}$ models. These models are trained serially. Given a set of sequential channel samples, $\mathbf{S} = \{\mathbf{H_1}, \mathbf{H_2}, ..., \mathbf{H_i}, ...\mathbf{H}_{|\mathbf{S}|}\}$, where $|\mathbf{S}|$ represents the cardinality of the set $\mathbf{S}$. The first model is trained such that E_1 and D_1 minimize the reconstruction loss $\|\mathbf{H}_i - D_1(E_1(\mathbf{H}_i))\|_F^2 \ \forall i \in \mathbf{S}$. For the second encoder-decoder pair, we seek to minimise the reconstruction loss: $||\mathbf{H}_i - D_2(E_2(\mathbf{H}_i, \hat{\mathbf{H}}_{i-1}))||_F^2 \ \forall i \in$



(a) TSF Encoder Architecture





(c) Transformer Encoder

(b) TSF Decoder Architecture

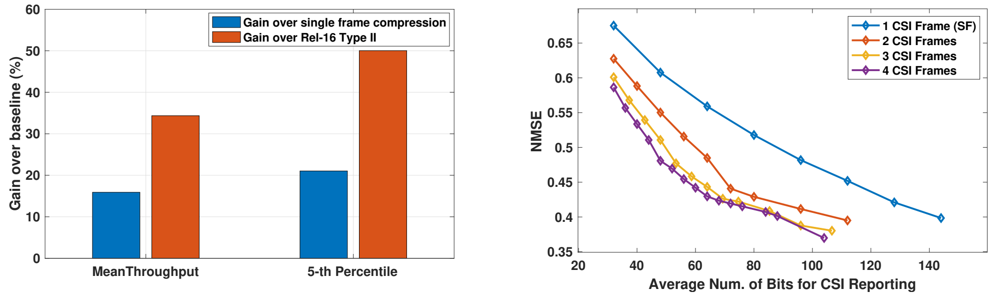
Figure 3: TSF Model Architecture.

$\{2, ..., |\mathbf{S}|\}$, where $\hat{\mathbf{H}}_{i-1} = D_1(E_1(\mathbf{H}_{i-1}))$. Generalising this to the k^{th} encoder-decoder pair, we minimise $\|\mathbf{H}_i - D_k(E_k(\mathbf{H}_i, \hat{\mathbf{H}}_{i-1}, ..., \hat{\mathbf{H}}_{i-k}))\|_F^2 \ \forall i \in \{k+1, ..., |\mathbf{S}|\}$, where, $\hat{\mathbf{H}}_{i-j} = D_j(E_j(\mathbf{H}_{i-j})) \ \forall j \in \{1, ..., k\}$. We train the encoder-decoder pairs serially so that at the end of each training cycle, we can run inference using the trained model to generate the priors to train the next model.

We train our models using Adam and use a learning rate scheduler that reduces the learning rate by 10% every 5 epochs, with a starting rate of 0.01. We use a batch size of 128 and train each model for 100 epochs.

4 Results

In this section, we provide some preliminary results on the performance of the proposed TSF approach relative to two baselines; SF compression using an auto-encoder with the same architecture as the proposed TSF model but without using any past information and a 3GPP code-book based baseline, referred to as Rel-16 Type II codebook, which is part of the existing wireless standards. For the simulation setup, we consider the urban macro (UMa) channel scenario [5] at 4 GHz carrier frequency. The channels are collected from multiple BSs and multiple users moving at 10 km/hr speed, where each BS has 16 transmit antennas while each user has two receive antennas. We consider a bandwidth of 26 frequency components, i.e. $N_c = 26$. This makes each channel sample a complex tensor of dimensions $2 \times 16 \times 26$. For the TSF approach, we assume that the value of N is set to 3, so each user is utilizing up to 3 past channel samples in the compression process of the current sample.



(a) (b)

Figure 4: (a)Throughput results of the proposed TSF approach relative to Rel-16 Type II baseline and SF compression baseline. (b)NMSE performance comparison between SF and TSF compression methods with different number of feedback bits.

Fig. 4a shows the overall throughput performance gain of the proposed TSF approach over the SF baseline (blue bar) and the Rel-16 Type II 3GPP baseline (orange bar). To evaluate the throughput, we plug the 3 different models in a 3GPP-compliant wireless communication simulation pipeline and observe the impact of these methods on the overall data throughput or data rate achieved. For TSF compression, the first SF model used to compress the first sample has an overhead of 128 bits while the TSF model for the second, third and fourth sample has an overhead of 64 bits. This brings the average overhead of the TSF approach to 80 bits per reporting instance. The standalone SF compression model has an overhead of 80 bits for every reporting instance and likewise Rel-16 Type II. It can be seen that in terms of mean throughput, the proposed approach achieves 34% and 16% gain over the Rel-16 Type II and the SF approach, respectively. In addition, in terms of 5-th percentile, i.e., cell-edge (users with the worst channel conditions), throughput, the proposed approach achieves quite promising gains of 50% and 21% over the Rel-16 Type II and the SF approach, respectively.

We further showcase the performance of the TSF method by comparing the number of bits utilized to compress a CSI sample vs the achieved normalized mean squared error (NMSE) associated with CSI reconstruction. The rate-distortion-styled curve has been evaluated for the SF baseline (which assumes access to the current CSI sample only) and the TSF scheme with access to the current as well as up to 3 past CSI samples. Fig. 4b shows the NMSE performance against the average overhead associated with the CSI reporting. It can be seen that the proposed TSF approach with access to just 2 samples (current and 1 past sample) considerably outperforms SF compression. Further, as the number of past samples available for compression is increased, the performance improves more, while almost saturating when 3 past samples (a total of 4 CSI samples) are used for compression.

To achieve a reconstruction error benchmark of 0.4 NMSE, the SF scheme uses 144 bits on average, while the 4-sample TSF approach is able to achieve similar performance with just 88 bits. That's an overhead reduction of almost 39%.

5 Conclusion and Future Work

In this work, we introduce the interesting data compression paradigm associated with the problem of real-time channel state information (CSI) compression in wireless communication systems. To address the challenge, we propose to use the knowledge of past samples to better compress and reconstruct the channel data. We further propose a transformer-based compression model that effectively outperforms the single-sample methods and the existing methods currently utilized as part of the 5G standard. As future work, we plan to explore improved model architectures to better leverage information contained in past samples and remove the dependency associated with having the decoder as part of the encoding process.

References

[1] 3GPP TS 38.214. NR; Physical layer procedures for data. 3rd Generation Partnership Project; Technical Specification Group Radio Access Network.

[2] Teng-Hui Huang, Akshay Malhotra, and Shahab Hamidi-Rad. A deep learning method for joint compression and unsupervised denoising of csi feedback. In *ICC 2023 - IEEE International Conference on Communications*, pages 4150–4156, 2023. doi: 10.1109/ICC45041.2023.10279775.

[3] Wendong Liu, Wenqiang Tian, Han Xiao, Shi Jin, Xiaofeng Liu, and Jia Shen. Evcsinet: Eigenvector-based CSI feedback under 3GPP link-level channels. *IEEE Wireless Communications Letters*, 10(12):2688–2692, Sept 2021. doi: 10.1109/LWC.2021.3112747.

[4] Lu Lu, Geoffrey Ye Li, A Lee Swindlehurst, Alexei Ashikhmin, and Rui Zhang. An overview of massive MIMO: Benefits and challenges. *IEEE Journal of Selected Topics in Signal Processing*, 8(5):742–758, Apr. 2014.

[5] Shu Sun, Theodore S Rappaport, Sundeep Rangan, Timothy A Thomas, Amitava Ghosh, Istvan Z Kovacs, Ignacio Rodriguez, Ozge Koymen, Andrzej Partyka, and Jan Jarvelainen. Propagation path loss models for 5G urban micro-and macro-cellular scenarios. In *IEEE 83rd Vehicular Technology Conference (VTC Spring)*, pages 1–6, Nanjing, China, May 2016.

[6] Chao-Kai Wen, Wan-Ting Shih, and Shi Jin. Deep learning for massive MIMO CSI feedback. *IEEE Wireless Communications Letters*, 7(5):748–751, Mar. 2018.