Article | Received 16 January 2026; Revised 4 March 2026; Accepted 12 March 2026; Published 8 April 2026
https://doi.org/10.55092/neuroelectronics20260006

Lossy compression with feature reduction preserves motor decoding for brain-computer interfaces

Byeongchan Jeong[1], Anh Tuan Nguyen[2], Tong Wu[3], Brian Z. H. Lim[1] and Zhi Yang[1,2,*]

[1] Department of Biomedical Engineering, University of Minnesota, Minneapolis, USA
[2] Fasikl Incorporated, Bloomington, USA
[3] IQVIA Holdings, Inc., Wayne, USA

* Correspondence author; E-mail: yang5029@umn.edu.

Highlights:

- Compression Based Feature Reduction (CBFR) is presented as a transform-based lossy compression and feature selection pipeline for BCI devices.
- For transform-based compression in CBFR, DWT (Sym4, Haar), DCT, and WHT are compared for compression performance, signal quality, and decoding performance.
- CBFR is validated on invasive and non-invasive wrist surface recordings.
- CBFR reaches 11.29× compression on invasive data while preserving or improving accuracy.
- CBFR reaches 21.08× compression on non-invasive data while preserving decoding accuracy.

Abstract: Implantable neural prosthetic systems must transmit multichannel peripheral nerve recordings under strict power and wireless bandwidth constraints. This study evaluates a compression based feature reduction (CBFR) pipeline that couples transform domain lossy compression with post-compression feature reduction to preserve motor decoding while reducing data rate. After preprocessing, signals are compressed using Sym4/Haar the discrete wavelet transform (DWT), the discrete cosine transform (DCT), or the Walsh–Hadamard transform (WHT) with coefficient soft-thresholding, reconstructed, and used to compute 14 time-domain features. CBFR then computes feature-wise normalized root mean square error (NRMSE) relative to the preprocessed baseline and discards features that are insufficiently preserved before training a GRU classifier. On invasive recordings, CBFR achieves up to 11.29× compression while keeping accuracy about 11% above baseline. On non-invasive recordings, compression ratios up to 21.08× are obtained while accuracy remains about 5% above baseline. DCT provides consistently strong balanced accuracy and compression results, whereas WHT produces higher compression with greater variability. All evaluations are performed in software on recorded datasets, and end-to-end on-device benchmarking and direct comparisons to learned compressors remain future work.

Keywords: brain-computer interface; neural prosthetics; neural signal compression; lossy compression;

discrete wavelet transform; discrete cosine transform; Walsh-Hadamard transform; feature extraction; feature reduction

1. Introduction

Brain-computer interfaces (BCIs) decode patterns of neural activity into commands that control external devices or software. Their clinical applications include restoring voluntary limb movements in people with motor impairments [1,2] and enabling communication for individuals with amyotrophic lateral sclerosis (ALS) [3]. Because neural signals are stochastic, distinguishing meaningful responses from background noise is challenging [4–8]. Consequently, high performance BCIs that support rapid, intention driven control often require many recording channels and higher sampling resolutions to accurately capture motor intention [9–11]. These choices generate large data volumes and high transmission demands, which are especially problematic for implantable systems. At the same time, implantable BCIs must operate under stringent constraints on power, device size, and wireless bandwidth [12], so balancing channel quantity and signal quality against real time transmission capacity remains a persistent design challenge.

Data compression is a widely used approach to reduce signal bandwidth. In general, data compression can be categorized into two approaches: lossy compression, which achieves a higher compression ratio by sacrificing some of the original data, and lossless compression, which preserves all original information but typically attains only a modest compression ratio [13,14]. In neural signal processing, both lossy and lossless compression have evolved over decades, with their use diverging by application. Lossy compression is commonly used to extract features or decode information directly from the signal, whereas lossless compression is typically employed to store the complete signal for future analysis or to identify new features. Present study focuses on lossy compression and seeks to extend our previous experimental work into a more practical implementation [15,16].

Early lossy compression methods, based on wavelet transforms that effectively represent nonstationary and oscillatory data [17,18], extracted spikes from neural signals and applied the discrete wavelet transform (DWT) with entropy coding such as run length encoding (RLE), achieving about 5$\times$ compression [19]. Later, DWT based image coding methods such as JPEG2000 and SPIHT yielded up to 8$\times$ compression on neural signals [20]. Recently, deep neural network (DNN) based compression methods, such as autoencoders, have achieved compression ratios of 20$\times$ or more [21–23]. The performance of these methods has typically been evaluated by measuring post compression signal quality using signal-to-noise ratio (SNR), percent root mean square difference (PRD), and normalized root mean square error (NRMSE) [24]. Spike extraction has been applied prior to the compression algorithm in the aforementioned studies, achieving significant reductions in data rate. However, this approach is limited for peripheral nerve recordings, whose amplitudes are significantly lower than those of central nervous system signals due to the mixture of afferent and efferent activity [25–27]. In addition, compound action potentials (CAPs) generated by hundreds of axons in peripheral nerves [28] pose a challenge for applying spike-detection methods [29,30] to fully capture intention-driven information for gesture classification. To the best of our knowledge, no studies have verified whether spikes extracted from the full neural signal still contain sufficient information for use in BCI systems. Therefore, an alternative perspective is necessary to validate the practicality

of lossy compression for motor decoding, not only to evaluate the compression ratio and signal quality but also to verify whether the compression algorithm contributes to degradation in gesture classification accuracy. Accordingly, this study will not only apply lossy compression techniques and evaluate their compression efficiency but also assess their practicality using a deep-learning (DL) classification model to examine the trade off between compression ratio and achievable accuracy.

Our team has previously controlled prosthetic devices through invasive recordings of upper limb peripheral nerves [16]. To capture richer information, we sampled 16 channels at 10 kHz per channel and extracted 14 feature types [31]. By training a DL classification model on these features, we decoded finger-movement intentions and demonstrated feasibility. However, its resulting data rate has exceeded the capacity of existing wireless links [32,33]. Consequently, experiments were conducted using a percutaneous wired connection, which may introduce risks of tissue inflammation and channel breakage, and prevents long-term implantation.

Therefore, in this study, we present compression based feature reduction (CBFR), which not only performs real-time lossy compression but also enhances the decoding performance of finger movements. The proposed pipeline extends our previous work [34], applies a transform-based compression algorithm to compress neural signals, and then selects undistorted features for further motor decoding. Specifically, we compare four orthonormal transforms: DWT using Symmlet 4 and Haar wavelets, the discrete cosine transform (DCT), and the Walsh–Hadamard transform (WHT). Compared to our prior conference report [34], this manuscript expands CBFR by providing (i) an end-to-end frame-based streaming transmitter/receiver pipeline with coefficient-domain quantization and entropy coding, (ii) a systematic benchmark across these transforms over multiple threshold scales, (iii) validation on both invasive intrafascicular recordings and non-invasive surface recordings, and (iv) embedded-oriented analyses including computational complexity and balanced operating-point selection. On invasively recorded signals, the proposed methods achieved compression ratios between $5.91\times$ and $11.29\times$, while decoding accuracy ranged from 72.36%–91.49%, corresponding to changes of roughly -8% to $+11\%$ relative to the filtered baseline of 80.88%. On non-invasively recorded neural signals, the same methods yielded higher compression ratios between $7.73\times$ and $21.08\times$, with decoding accuracy confined to a narrow band of 92.98%–94.91%, about 3%–5% above the 89.42% baseline. In this work, we test the practicality of lossy compression for motor decoding by introducing CBFR. We evaluate not only its compression efficiency but also its decoding performance on invasive and non-invasive neural recordings to determine the acceptable level of signal degradation introduced by compression.

2. Methods

2.1. Dataset

We evaluated CBFR on two datasets as summarized in Table 1.

Dataset 1: The invasive neural signals used in the current study were recorded in a prior study [16]. For each finger, the dataset includes 10 repetitions alternating relaxed and flexed states, with 2 second movement and rest intervals. A total of 16 recording channels, consisting of eight channels from the median nerve and eight channels from the ulnar nerve, were sampled at 10 kHz. A total of nine datasets

were used to train, validate, and test the DL classification model for decoding performance.

Dataset 2: Neural signals were recorded non-invasively from the wrist of a healthy participant using surface electrodes [15]. The dataset also includes 10 repetitions alternating relaxed and flexed states for each finger, with 2 second movement and rest intervals. Recordings were acquired from 16 channels (eight channels targeting the median nerve and eight channels targeting the ulnar nerve) sampled at 10 kHz, and a total of five datasets were used to train, validate, and test the DL classification model for decoding performance. For the invasive experiments, we used the Scorpius-1 system, and for the non-invasive experiments, we used the Scorpius-3 neural recorder. Both systems are equipped with the in house Neuronix neural recording chip.

Table 1. Summary of invasive and non-invasive neural recording datasets.

Data set	Electrode Type	Total Sessions	Sampling[1] rate	Recording time	ADC resolution	Data size[2] per session	Data rate[3] for streaming
1	Intrafascicular [16]	9	10 kHz	∼200 s	12 bits	256 MB	1.92 Mbps
2	Surface [15]	5	10 kHz	∼150 s	12 bits	212 MB	1.92 Mbps

[1] All data are downsampled by a factor of 2. Compression ratios are computed after downsampling.
[2] Data size refers to the raw (uncompressed) data. One sample corresponds to one time point across channels. Total raw data size equals (data size per session) × (number of sessions), *i.e.*, 256 MB × 9 ≈ 2.3 GB for Dataset 1 and 212 MB × 5 ≈ 1.1 GB for Dataset 2.
[3] Data rate is calculated as sampling rate × ADC resolution × number of channels.

2.2. Neural recorder (transmitter)

Figure 1 illustrates the CBFR pipeline. Figure 1a shows the transmitter, which will ideally be implanted in the subject and will transmit the compressed neural recordings. Figure 1b shows the receiver, which obtains the compressed signal from the transmitter and performs model training on the host computer. The entire process implements a frame-based streaming pipeline suitable for real-time lossy compression. After downsampling, the system processes 10 samples per frame across 16 channels, which corresponds to 320 bytes per uncompressed frame when packetized as 16-bit words (10 × 16 × 16/8), to meet the packet-size constraints of the Field-Programmable Gate Array (FPGA). Although the recorder digitizes signals with a 12-bit Analog-to-Digital Converter (ADC), the samples are stored and transmitted as 16-bit signed integers (int16) for memory alignment and packet formatting.

2.2.1. Preprocessing

Both datasets were processed through a uniform preprocessing pipeline that included an anti-aliasing filter, downsampling by a factor of two, notch filters at harmonic frequencies, a powerline noise removal filter, and a 25–600 Hz bandpass filter. The 25–600 Hz bandpass range was applied because it preserves the spectral content related to motor control, as reported in previous studies [16]. Each frame was filtered in real time while preserving the filter state, and then processed with running mean subtraction ($a = 0.1$). With downsampling by a factor of two, each frame in the streaming data contains 10 samples × 16 channels.

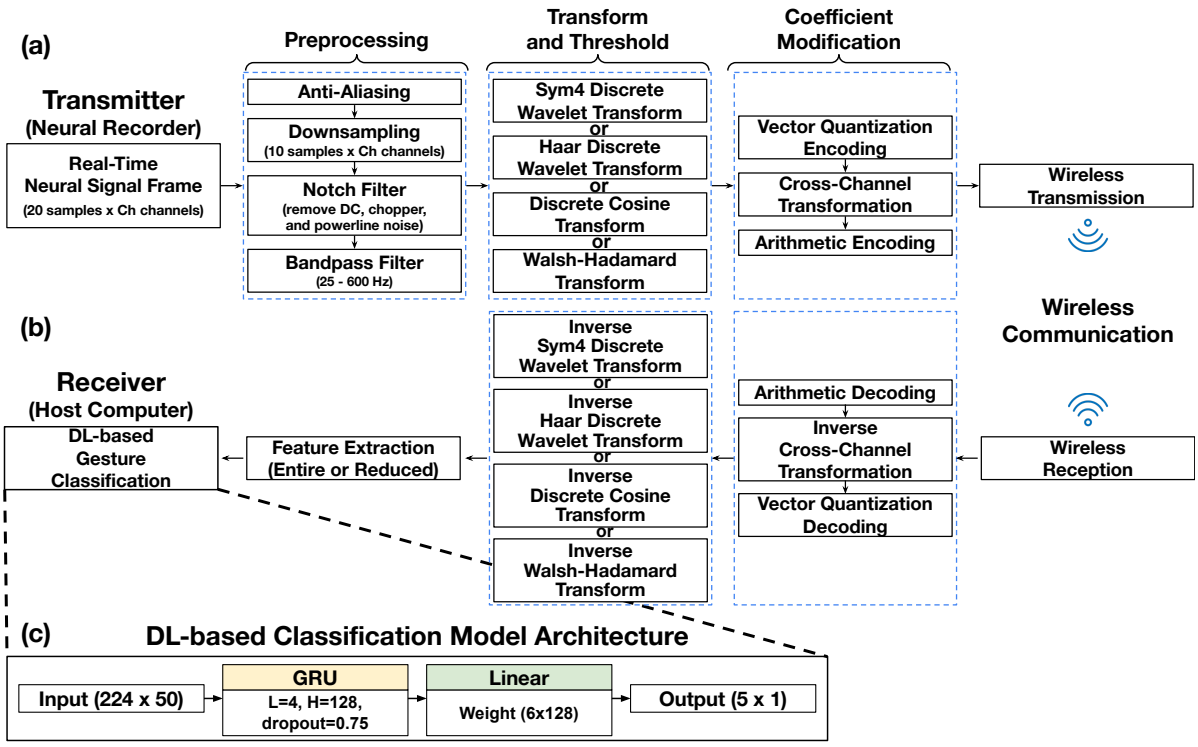

Figure 1. Compression based feature reduction pipeline for neural signal. **(a)** Transmitter side implementation intended for an implantable neural recorder; **(b)** Receiver side implementation on the host computer; **(c)** Deep-learning-based gesture classification model architecture.

2.2.2. Transform and threshold

We evaluated four transforms: Sym4 DWT, Haar DWT, DCT, and WHT, and compared their compression efficiency and decoding accuracy. DWT, DCT, and WHT are linear and invertible with orthonormal bases [35,36], so they preserve energy by Parseval's relation and allow exact reconstruction when no coefficients are discarded [37–39]. For many natural signals they compact energy and reduce inter sample correlation, producing sparse coefficient sets that work well with soft-thresholding and quantization [40–43]. They are efficient algorithms with complexity on the order of O(N log N) run on microcontrollers, which makes these methods practical for real time and embedded use [44]. Each method has also been applied in prior studies using neural signals recorded from the central nervous system [45–48]. All transforms were applied independently to each channel within each frame. After downsampling, each frame contained N = 10 samples per channel (16 channels), and a 1-D transform was computed along the time axis for each channel.

DWT decomposes time domain signals into multiple wavelet components that represent different frequency components of the original signal at varying levels of detail. For this study, we used the Sym4 and Haar wavelets at the first level ($j = 1$). In our implementation, the single-level DWT was computed using MATLAB R2023b (MathWorks, USA) wavedec, applied separately with the Sym4 wavelet and the Haar wavelet.

$$\begin{aligned} A_j[k] &= \sum h[n-2k] \cdot A_{j-1}[n], \\ D_j[k] &= \sum g[n-2k] \cdot A_{j-1}[n], \quad j = 1 \end{aligned} \tag{1}$$

Where A_j[k] and D_j[k] are the approximation and detail coefficients at level j and $h[n]$ and $g[n]$ denote the low-pass (scaling) and high-pass (wavelet) analysis filters. We take $A_0[n] = x[n]$ as the input signal.

DCT maps a finite sequence of samples to a sum of orthogonal cosine basis functions. In this study, the type II DCT (DCT-II), defined below, was applied for its energy compaction properties. In our implementation, the DCT coefficients were computed using MATLAB dct, which computes the unitary Type-II DCT by default.

$$X[k] \;=\; \alpha(k) \sum_{n=0}^{N-1} x[n] \cos\!\left(\frac{\pi}{N}\left(n+\tfrac{1}{2}\right)k\right), \qquad k = 0,\ldots,N-1, \tag{2}$$

where the normalization, $\alpha(k)$ is

$$\alpha(k) \;=\; \begin{cases} \sqrt{\frac{1}{N}}, & k = 0, \\ \sqrt{\frac{2}{N}}, & k = 1,\ldots,N-1. \end{cases} \tag{3}$$

WHT maps time domain signals onto an orthogonal basis of square wave functions, known as Walsh functions, which are characterized by values of +1 and -1. It is effective for piecewise constant signals in which energy concentrates in low-sequency components and is defined as

$$\mathrm{Y} = H_n \cdot \mathrm{X} \tag{4}$$

Where X is $2^n \times 1$ input vector and H_n is the Hadamard transform (HT), a $2^n \times 2^n$ matrix, defined as:

$$H_n = \frac{1}{\sqrt{2^n}} \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix} \tag{5}$$

In our implementation, the WHT was computed using MATLAB fwht. Since fwht operates on signals with length 2^n, each $N = 10$ sample segment was zero-padded to the next power-of-two length prior to the transform, and the inverse transform output was cropped back to N samples.

After each transform, we applied soft-thresholding to attenuate small-magnitude coefficients that are more likely dominated by noise, while retaining the dominant components that carry most of the signal energy for downstream decoding. The thresholding strategy and its progressive strength levels were chosen based on prior transform-domain compression and soft-thresholding approaches [17,49,50]. Thresholds were computed independently for each channel to account for channel-dependent amplitude and noise differences, and then applied using MATLAB wthresh with soft-thresholding.

To parameterize threshold strength, we used seven discrete levels indexed by scale $\in \{1,\ldots,7\}$ and defined an exponent $e = \text{scale} - 4 \in \{-3,\ldots,3\}$.

For the DWT (Sym4 and Haar) and DCT, the threshold for a given channel and frame was computed from the corresponding coefficient vector c as

$$T = \sqrt{\frac{\mathrm{MAD}(c)}{0.6745}}\,\sqrt{2^e \log(|c|)}, \tag{6}$$

where $|c|$ denotes the length of c and $\mathrm{MAD}(\cdot)$ denotes the median absolute deviation (computed in MATLAB using mad(c ,1)).

For the WHT, we used an Otsu histogram-based threshold computed by MATLAB graythresh from the coefficients and scaled it as

$$T = 2^{e}\,\text{graythresh}(c). \tag{7}$$

Figure 2 illustrates an example frame and the corresponding transform coefficients before and after thresholding for each method.

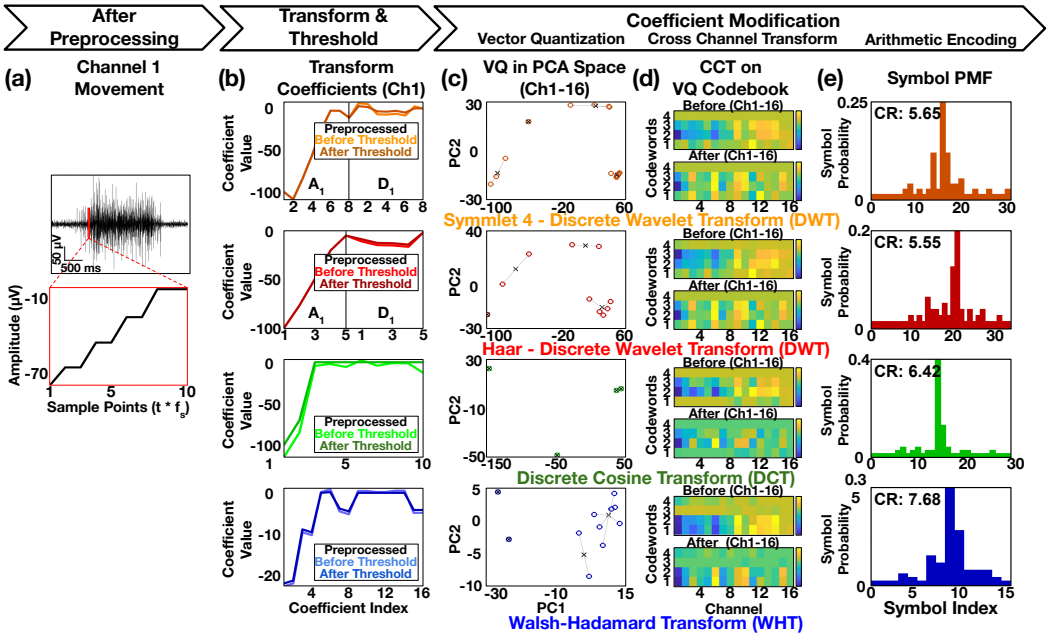

Figure 2. Example intermediate outputs of the transmitter side pipeline for one frame after preprocessing (10 samples × 16 channels). **(a)** Example preprocessed waveform segment with a 10 sample zoom; **(b)** Transform coefficients of Ch1 before and after thresholding for Sym4 DWT, Haar DWT, DCT, and WHT; **(c)** Vector quantization results in a 2D Principal Component Analysis (PCA) space for Ch1 to Ch16; **(d)** Cross channel transform of the VQ codebook across channels (Ch1 to Ch16); **(e)** Example symbol Probability Mass Function (PMF) used for arithmetic encoding, with the resulting compression ratio (CR) for one frame.

2.2.3. Coefficient modification

Figure 2 also shows the intermediate outputs of coefficient modification. We applied coefficient modification to the thresholded transform coefficients for efficient quantization. We first performed vector quantization (VQ) encoding using k-means with four codewords to learn a compact codebook from the surviving coefficient vectors, replaced each vector by the index of its nearest codeword, and retained the codebook for later reconstruction at receiver.

We then applied a cross-channel transformation to reduce inter-channel dependencies by isolating common components and simplifying the statistical structure [2]. This process concentrates shared noise and drift in the average component and normalizes the differential components to remain centered with reduced variability. Formally, for a subset of n channels with samples $\chi = [x_1, \ldots, x_n]^\top$, we compute the transformed vector $\chi' = [x'_1, \ldots, x'_n]^\top$ as

$$x'_1 = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad x'_k = \frac{1}{2}(x_{k-1} - x_k), \quad k = 2, \ldots, n \tag{8}$$

In this transform, x'_1 collects the shared component across the subset, while $x'_2, \ldots, x'_n$ describe adjacent differences that stay centered near zero with lower variance. We applied this transform to two subsets of eight channels each. For Dataset 1, the subsets correspond to channels recorded within the same implanted array/nerve context (shared reference and local proximity), whereas for Dataset 2, the subsets correspond to the median-targeted and ulnar-targeted surface electrode sets. We then concatenated the transformed codebook and the VQ index stream and flattened them into a 1D array for entropy encoding.

Arithmetic coding was adopted for entropy encoding because the quantized coefficients in the dataset follow a heavy-tailed distribution rather than a uniform one, as our goal is to encode meaningful information related to movement intention within the dataset. Since the recorded dataset is event-driven, it exhibits greater variation than datasets such as the resting state [51]. By capturing subtle variations in symbol probabilities, arithmetic coding achieves finer granularity in compression compared to other entropy encoding methods such as Huffman coding or run-length encoding [52].

2.2.4. Complexity

The neural recorder is designed for implantation, so verifying its computational and memory complexity is essential for hardware implementation. Assuming the transmitter pipeline follows the order in Figure 1a, Table 2 reports the computed complexity for each stage.

Table 2. Computational and memory complexity of each stage in the transmitter pipeline.

Name	Computational Complexity	Memory Complexity
Preprocessing	$O(N_s N_{\mathrm{CH}} N_{f'})$	$O(N_s N_{\mathrm{CH}} N_{f'})$
Transform/Threshold (DWT)	$O(N_s N_{\mathrm{CH}})$	$O(N_s N_{\mathrm{CH}})$
Transform/Threshold (DCT & WHT)	$O\big((N_s N_{\mathrm{CH}}) \log(N_s N_{\mathrm{CH}})\big)$	$O(N_s N_{\mathrm{CH}})$
Vector Quantization	$O(N_s K N_{\mathrm{CH}})$	$O(N_s N_{\mathrm{CH}} + K N_{\mathrm{CH}})$
Cross–Channel Transformation	$O(N_s N_{\mathrm{CH}})$	$O(N_s N_{\mathrm{CH}})$
Arithmetic Encoding	$O(N_s N_{\mathrm{CH}})$	$O(\lvert A \rvert)$

N_s: number of samples, typical value 10 to 20.
N_{CH}: number of channels, typical value 16.
$N_{f'}$: filter order, typical value 4 to 8.
K: number of VQ codewords, typical value 4.
$|A|$: number of distinct quantized symbols.

To justify real time streaming feasibility, we convert the per frame computational cost from the Table 2 expressions into a concrete compute budget and compare it against a representative embedded recorder Microcontroller Unit (MCU). Using a representative setting ($N_s = 10$, $N_{CH} = 16$, $N'_f = 8$, $K = 4$), the Table 2 terms yield approximately 1280 (preprocessing) + 1170 (DCT or WHT) + 640 (VQ) + 160 (cross channel transform) + 160 (arithmetic coding), giving a total of about 3410 basic operations per frame. With a downsampled sampling rate f_s = 5 kHz, the frame rate is f_s/N_s, so the compute demand is $\approx 1.7 \times 10^6$ operations per second.

Although this study evaluates the pipeline in simulation, the above budget provides an order of magnitude feasibility check for the neural recorder used in our prior work [2], which assumes an

nRF52840 class MCU (Arm Cortex-M4F at 64 MHz) [53] At f_s= 5 kHz and $N_s = 10$, each frame spans 2 ms, corresponding to $64 \times 10^2 \times 2 \times 10^{-3} \approx 128,000$ Central Processing Unit (CPU) cycles available per frame on a 64 MHz core. Conservatively mapping each basic operation to 4 CPU to account for memory access and control overhead gives $\approx 3410 \times 4 \approx 13,640$ cycles per frame, which is about 11% of the per frame cycle budget, leaving substantial headroom for streaming tasks such as packetization and wireless stack execution. These results are analytical complexity estimates and are not end-to-end latency/power benchmarks. In addition, the Cortex-M4 DSP extensions include single cycle Multiply-Accumulate (MAC) style instructions, further supporting efficient execution of the transform and filtering style operations typical in this pipeline.

2.3. Host computer (receiver)

After the transmitter in Figure 1a sends the arithmetically encoded, thresholded transform coefficients to the host computer (receiver) in Figure 1b, the host computer reconstructs the neural signal and extracts features for DL classifier training. The receiver obtains a bitstream of arithmetically encoded coefficients together with the cumulative distribution function (CDF) required for decoding.

2.3.1. Inverse coefficient modification

At the receiver, inverse coefficient modification reverses the operations applied during coefficient modificationhi to recover the transform coefficients prior to entropy encoding. Once the host computer receives the bitstream of encoded symbols and the corresponding CDF, they are passed through arithmetic decoding to reconstruct the flattened 1D array. This will then reshaped back into a 2D structure that contains both the transformed codebook entries and associated VQ index stream.

While the VQ index stream has concatenated with transformed VQ codebook, we separate them and only apply the inverse cross-channel transformation to the codebook to restore the original VQ codebook vectors. Denoting the transformed vector by $\chi' = [x'_1, \ldots, x'_n]^\top$ and the recovered vector by $\chi = [x_1, \ldots, x_n]^\top$, the encoding stage computes $\chi' = T\chi$ as in Equation (8).

We therefore obtain

$$\chi = T^{-1}\chi'. \tag{9}$$

For the specific structure of Equation (8), this can be written explicitly as

$$x_1 = x'_1 + \frac{2}{n}\sum_{k=2}^{n}(n+1-k)\,x'_k, \qquad x_k = x_{k-1} - 2x'_k, \quad k = 2, \ldots, n, \tag{10}$$

which reconstructs the shared component x_1 and all adjacent-difference components $x_2, \ldots, x_n$ from the transformed coefficients. This inverse transform is applied separately to each subset of eight channels, mirroring the grouping used at the encoder.

Finally, VQ decoding restores the transform coefficient vectors using the VQ codebook and the VQ index stream. Each stored VQ index acts as a pointer to its corresponding codeword in the codebook, replacing that time step's quantized vector. Applying this lookup across all time indices reconstructs the sequence of modified transform coefficients over all 16 channels. By the end of this step, each frame contains 10 samples $\times$ 16 channels of transform coefficients.

2.3.2. Inverse transform

Because the transforms in Equations (1)–(5) are linear and orthonormal, the coefficients that remain after thresholding can be mapped back to the time domain without any additional loss. In what follows, we denote by $\tilde{x}[n]$ the reconstructed signal after thresholding and inverse transform.

For the 1st DWT in Equation (1), the original approximation sequence $A_{j-1}[n]$ is reconstructed from the approximation and detail coefficients $A_j[k]$ and $D_j[k]$ by

$$A_{j-1}[n] = \sum_k h[2k-n]A_j[k] + \sum_k g[2k-n]D_j[k], \quad j = 1. \tag{11}$$

The Type II DCT in Equation (2) is inverted using the same normalization $\alpha(k)$ in Equation (3):

$$x[n] = \sum_{k=0}^{N-1} \alpha(k) X[k] \cos\left(\frac{\pi}{N}\left(n+\tfrac{1}{2}\right)k\right), \qquad n = 0,\ldots,N-1. \tag{12}$$

For the WHT in Equation (4), the input vector X is recovered from the transform coefficients Y by

$$\mathrm{X} = H_n^{\mathsf{T}}\,\mathrm{Y}. \tag{13}$$

2.3.3. Feature extraction and feature reduction

Following prior peripheral nerve decoding work [31], we extracted 14 time-domain features that summarize amplitude, variability, and waveform complexity, providing a compact representation for motor decoding. These features were computed from the reconstructed time-domain signal using 100 ms windows with 80% overlap. Each frame consists of 10 samples across 16 channels (2 ms at 5 kHz after $2\times$ downsampling), so each 100 ms window spans 50 consecutive frames.

Feature reduction was motivated by the requirement that lossy compression should not substantially degrade the information captured by this baseline feature set. To evaluate the robustness of each feature to the transform and threshold pipeline, we computed the same 14 time domain features on the compressed and reconstructed signal and quantified, for each feature, the normalized reconstruction error across channels and time. Features whose NRMSE exceeded a predefined threshold $\tau_{NRMSE} = 0.5$ were considered insufficiently preserved and were excluded from the final feature set. In our implementation, for each feature k and channel c, we computed the root mean square error (RMSE) across time between the reconstructed feature sequence $\hat{f}_{k,c}(t)$ and the preprocessed baseline $f_{k,c}(t)$, and then aggregated across the $C = 16$ channels to obtain a single feature wise score:

$$\mathrm{NRMSE}_k = \frac{\frac{1}{C}\sum_{c=1}^{C} \mathrm{RMSE}\left(\hat{f}_{k,c}(t), f_{k,c}(t)\right)}{\frac{1}{C}\sum_{c=1}^{C}\left(\max_t f_{k,c}(t) - \min_t f_{k,c}(t)\right)}, \quad C = 16. \tag{14}$$

The resulting retained feature identities are explicitly recorded for each transform method and threshold scale (and for each dataset), the retained feature sets at the balanced operating points selected by the Euclidean distance metric (Equation (26)) are listed in Supplementary Table S1, together with the common reduced feature sets shared across methods. To perform feature reduction, we first need to compare the features extracted from the compressed signal with those extracted from the uncompressed signal. Because the set of features that are discarded depends on the threshold level, this comparison only

needs to be carried out once before model training in practical settings. After that, we can compute only the reduced feature set and use it to train the model. Since different transforms distort different subsets of features, the reduced feature set is generally method-dependent rather than shared across all methods. As mentioned earlier, the goal of feature extraction was to reduce data dimensionality while preserving essential information [31,54]. If we can further reduce the feature set after this extraction step, we obtain additional dimensionality reduction while maintaining the essential information carried by the signal, which can help lower the computational load during training and inference.

2.3.4. Deep-learning classification model

The purpose of the present study was to evaluate the compression efficiency, not only by achieving a high compression ratio but also by maintaining decoding accuracy for neural prosthetic control in this experimental setting after compressing neural signals that encode limb movements or their intentions. To ensure a fair comparison, we used the classification model inspired from the previous study [15] to evaluate the decoding performance under comparable conditions.

Following the previous study, Figure 1c illustrates the DL classifier model used in this work. We employed a gated recurrent unit (GRU) based recurrent neural network (RNN) with 4 layers, 128 hidden units per layer, and a dropout rate of 0.75 [55]. This relatively high dropout rate was used as a strong regularizer to reduce co-adaptation among recurrent units and to improve generalization stability when modeling temporally correlated feature sequences. The classifier configuration, including the dropout rate, was kept fixed across all compression and feature-reduction conditions to ensure a fair comparison, such that differences in decoding performance reflect the effects of compression and feature selection rather than hyperparameter tuning. After this selection, all classifier hyperparameters were held fixed to avoid confounding the effects of compression and feature reduction with classifier tuning. The classifier outputs five classes corresponding to thumb, index, middle, ring, and pinky. The feature data has a size of $16 * F \times T$, where 16 is the number of channels, F is the number of features, and T is the number of feature time frames. The number of features F varies depending on how many remain after feature reduction. We compare decoding performance between the full set of 14 features ($F = 14$) and the remaining "good features" after feature reduction. To reduce variance in the classification results, 5-fold cross-validation was performed at the session level, with entire recording sessions assigned to train, validation, and test folds without overlap. This session-level cross-validation evaluates within-subject generalization across sessions and does not assess inter-subject generalization. Accordingly, the decoder was trained and evaluated separately for each dataset/participant.

2.4. Evaluation

The metrics used to evaluate the compression efficiency of CBFR are divided into three categories: compression performance, signal quality, and decoding performance.

Compression performance: The compression performance examines how efficiently the compressed bitstream represents the original multichannel neural signal, using the compression ratio and the sample-wise Shannon entropy of the reconstructed time-domain signal.

The compression ratio is defined as

$$\mathrm{CR} = \frac{S_{\mathrm{orig}}}{S_{\mathrm{comp}}} = \frac{N \times C \times L_z}{L_{\mathrm{enc}}}, \tag{15}$$

where N is the number of samples, C is the number of channels (16 channels), L_z is the bit depth of the original signal (12 bits), and L_{enc} is the total bit length of the arithmetic coded output. Here, $L_z = 12$ reflects the recorder's effective acquisition resolution (12-bit ADC) and is used throughout this work when computing the compression ratio. In the streaming implementation, samples are stored and packetized as 16-bit transport words for memory alignment and packet formatting. If CR is expressed with respect to the 16-bit transport word length, it scales as $CR_{16} = CR_{12} \times \frac{16}{12}$. This does not change the CR definition in (Equation (15)), which is referenced to the acquisition resolution. The compression ratio was computed after the preprocessing stage, including downsampling and bandpass filtering.

The entropy of the reconstructed signal is computed as

$$H = -\sum_{i=1}^{M} p_i \log_2 p_i, \tag{16}$$

where M is the number of unique quantized amplitude levels $\{s_i\}_{i=1}^{M}$ in the reconstructed time-domain signal and p_i is the empirical probability of level s_i,

$$p_i = \frac{\#\{n : \mathrm{signal}[n] = s_i\}}{N}. \tag{17}$$

This sample-wise Shannon entropy represents the average information per sample (in bits) and serves as a proxy for information content and signal complexity: a higher entropy indicates a more variable, less redundant signal with a richer set of amplitudes, whereas a lower entropy indicates a more concentrated amplitude distribution with reduced information content and increased redundancy.

Signal quality: Signal quality quantifies the distortion introduced by compression in the reconstructed time-domain signal $\hat{x}[n]$ with respect to the preprocessed signal $x[n]$. In Equations (18) and (19), the denominator term P_n represents the compression-induced reconstruction error power $\left(x[n] - \hat{x}[n]\right)$, rather than measurement noise during acquisition.

The SNR is

$$\mathrm{SNR} = 10 \log_{10} \left(\frac{P_s}{P_n} \right) \ [\mathrm{dB}], \tag{18}$$

where the signal power P_s and the reconstruction error power P_n are

$$P_s = \frac{1}{N} \sum_{n=1}^{N} x[n]^2, \qquad P_n = \frac{1}{N} \sum_{n=1}^{N} (x[n] - \hat{x}[n])^2. \tag{19}$$

The peak signal-to-noise ratio (PSNR) is defined as

$$\mathrm{PSNR} = 10 \log_{10} \left(\frac{\mathrm{Peak}^2}{\mathrm{MSE}} \right) \ [\mathrm{dB}], \qquad \mathrm{MSE} = \frac{1}{N} \sum_{n=1}^{N} (x[n] - \hat{x}[n])^2, \tag{20}$$

where Peak is the maximum possible amplitude of $x[n]$.

The NRMSE is

$$\mathrm{NRMSE} = \frac{\sqrt{\frac{1}{N}\sum_{n=1}^{N}(x[n]-\hat{x}[n])^2}}{\max_n x[n] - \min_n x[n]} \tag{21}$$

The PRD is

$$\mathrm{PRD} = 100\sqrt{\frac{\sum_{n=1}^{N}(x[n]-\hat{x}[n])^2}{\sum_{n=1}^{N}x[n]^2}}\ [\%] \tag{22}$$

Decoding performance: Decoding performance measures how well the features from compressed data and preprocessed data support downstream classification for neural prosthetic control. We report accuracy and the F1 score, which are defined as follows.

$$\text{Balanced Accuracy} = \frac{\text{Sensitivity} + \text{Specificity}}{2}, \tag{23}$$

$$\text{Sensitivity} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}, \qquad \text{Specificity} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}, \tag{24}$$

where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively. Balanced accuracy has stated as classification accuracy in current study.

$$\text{Precision} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}, \qquad \mathrm{F1} = \frac{2\cdot\text{Sensitivity}\cdot\text{Precision}}{\text{Sensitivity}+\text{Precision}}. \tag{25}$$

For the present study, classification accuracy, sensitivity, specificity, precision, and F1 are computed for each class and then averaged across classes. These metrics are computed independently for each of the five cross-validation folds using the held-out test sessions, and are reported as mean $\pm$ standard deviation across folds unless otherwise specified.

While decoding performance varies with compression ratio due to the threshold scale, different settings often trade accuracy and compression in opposite ways. Some configurations produce high compression with poor classification accuracy, whereas others maintain high accuracy with poor compression. To evaluate CBFR fairly across transform methods, we quantified this tradeoff using a weighted Euclidean distance metric, applied after normalizing accuracy and compression ratio within each method. This allowed us to identify an operating point that balances both objectives on a comparable scale across all methods.

$$d = \sqrt{w_{\mathrm{acc}}(1-Acc_n)^2 + w_{\mathrm{CR}}(1-CR_n)^2}, \tag{26}$$

where Acc_n is the normalized classification accuracy, CR_n is the normalized compression ratio, w_{acc} is the weight applied to the accuracy term (set to 10), and w_{CR} is the weight applied to the compression ratio term (set to 1). These weights were chosen to reflect that, in this application, even small drops in classification accuracy can noticeably impair motor intent decoding, while the compression methods already achieve high compression levels, making further gains in compression ratio less important than maintaining accurate decoding.

3. Results

Figure 3 compares the preprocessed signal obtained after the preprocessing stage with the reconstructed signals produced by each transform method. Figure 3a shows a two-second window of a single-finger movement in one channel, and Figure 3b presents a zoomed-in view of the same interval. The maximum threshold scale

(scale = 7) was applied to highlight the substantial differences across transform methods. Although each transform was applied frame-by-frame (10 samples × 16 channels), the time-domain traces alone do not clearly indicate which method preserves the movement characteristics more effectively. Therefore, compression performance, signal quality, and decoding performance were evaluated for a more informative comparison.

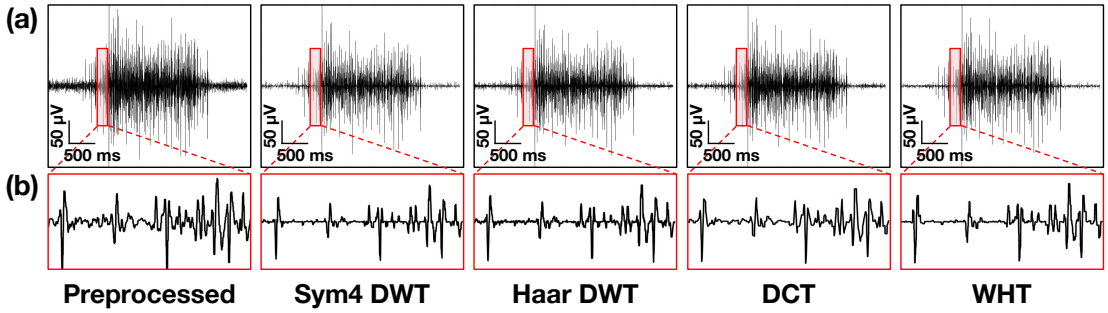

Figure 3. Example of single channel time domain signal before and after transform-based lossy compression: **(a)** Around 2 seconds segment during a finger movement in the preprocessed signal; **(b)** Zoomed view of the same interval after applying Sym4 DWT, Haar DWT, DCT, and WHT at the highest threshold scale.

3.1. Compression performance

Figure 4 shows the compression performance of each transform method across threshold scales 1 to 7. Because the threshold is applied only to the transform coefficients, the preprocessed signal is unaffected by thresholding and therefore maintains the same compression ratio across all scales in Figure 4a and the same entropy in Figure 4b, shown in black. In Figure 4a, the other four transform methods exhibit gradual increases in compression ratio as the threshold scale increases, and WHT achieves the highest compression ratio while maintaining entropy comparable to the other methods, as shown in Figure 4b. This trend appears consistently for both Dataset 1 (top) and Dataset 2 (bottom). The transform methods, including the compression ratio of the preprocessed signal, produced higher compression in Dataset 2 (Figure 4a, bottom) compared to Dataset 1 (Figure 4a, top). In Figure 4b, the preprocessed signal of Dataset 2 shows the highest entropy among all methods, whereas the preprocessed signal of Dataset 1 shows the lowest. This indicates that Dataset 2 contains less noise and provides a clearer representation of finger movements than Dataset 1.

As a result, applying the transform methods to Dataset 1 likely removed a greater amount of noise, which may have caused the remaining signal distribution to become more informative for representing movement related patterns. This is reflected in the noticeable increase in entropy from 3.85 to 6.46 even at first scale, corresponding to a 67% rise. In contrast, applying the same transform methods to Dataset 2 reduced entropy in a way that may also have removed useful information, given that its preprocessed signal already exhibited relatively high entropy. Therefore, although the compression performance in Figure 4 suggests that WHT achieves the best compression ratio while maintaining similar entropy, its impact on signal quality and decoding performance still needs to be examined.

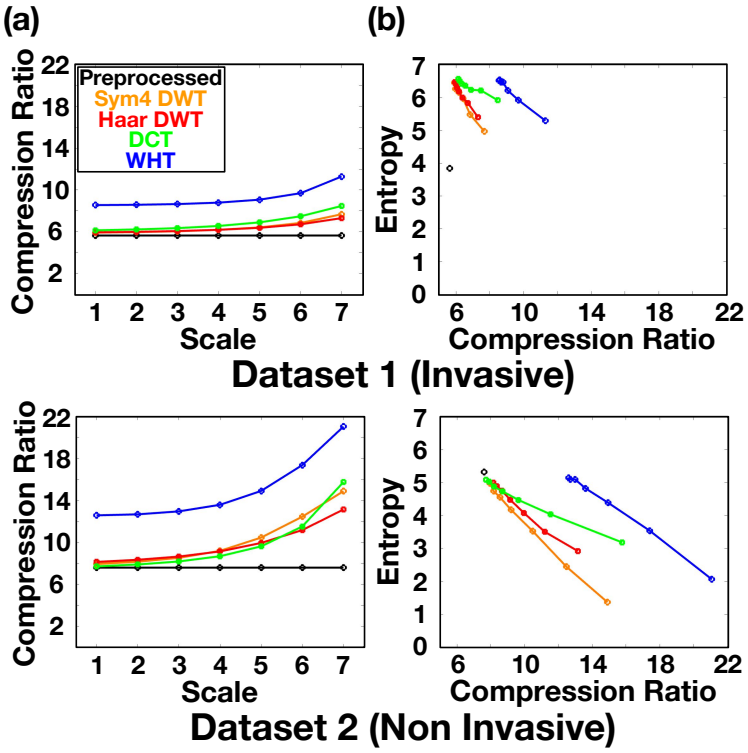

Figure 4. Compression efficiency for transform-based compression across threshold scales. **(a)** Compression ratio over threshold scale; **(b)** Sample-wise Shannon entropy over compression ratio. The top row corresponds to Dataset 1 (Invasive), and the bottom row corresponds to Dataset 2 (Non-Invasive).

3.2. Signal quality

Figure 5 compares the signal quality of each transform method across their corresponding compression ratios. While PSNR, NRMSE, and PRD were computed with respect to the original signal $x[n]$ as defined in Equations (18)–(22), SNR was computed using the preprocessed signal, indicated by the black circle in Figure 5a. Overall, the SNR increases after applying the transform methods for both Dataset 1 (top) and Dataset 2 (bottom), indicating that the transforms generally suppress noise relative to the preprocessed signal. Although Dataset 2 shows a reduction in entropy in Figure 4b, the SNR trend suggests that the transform methods still remove noise components effectively. PSNR in Figure 5b decreases consistently as the compression ratio increases across all transform methods. This follows the expected behavior, since higher compression leads to larger reconstruction errors relative to the preprocessed signal. Accordingly, a lower PSNR indicates greater deviation from the baseline signal at higher compression ratios. NRMSE and PRD in Figure 5c,d show the same trend, increasing as the compression ratio increases, which reflects the accumulation of reconstruction error when stronger thresholding is applied.

WHT achieves the highest compression ratio with relatively small changes in entropy in Figure 4. However, its PSNR in Figure 5b is the lowest among the methods, and its NRMSE and PRD values in Figure 5c,d is relatively higher, indicating greater reconstruction distortion despite its strong compression performance. In contrast, DCT maintains PSNR near the upper range and generally lies near the lower boundary for NRMSE and PRD across the threshold scales, suggesting that it introduces less distortion overall. Although DCT does not always produce the highest SNR, it achieves a higher compression ratio than DWT and preserves more

stable reconstruction quality. Taken together, these signal quality metrics suggest that DCT offers a more balanced tradeoff than WHT by providing sufficiently high compression with noticeably lower distortion.

Taken together, increasing the threshold scale improves noise suppression (Figure 5a, SNR), but also increases the deviation from the preprocessed signal (Figure 5b–d, PSNR, NRMSE, PRD). Because these metrics use the preprocessed signal as the baseline, they quantify how much distortion is introduced by compression, but they do not directly indicate whether the preserved information is still sufficient for capturing meaningful movement-related neural activity.

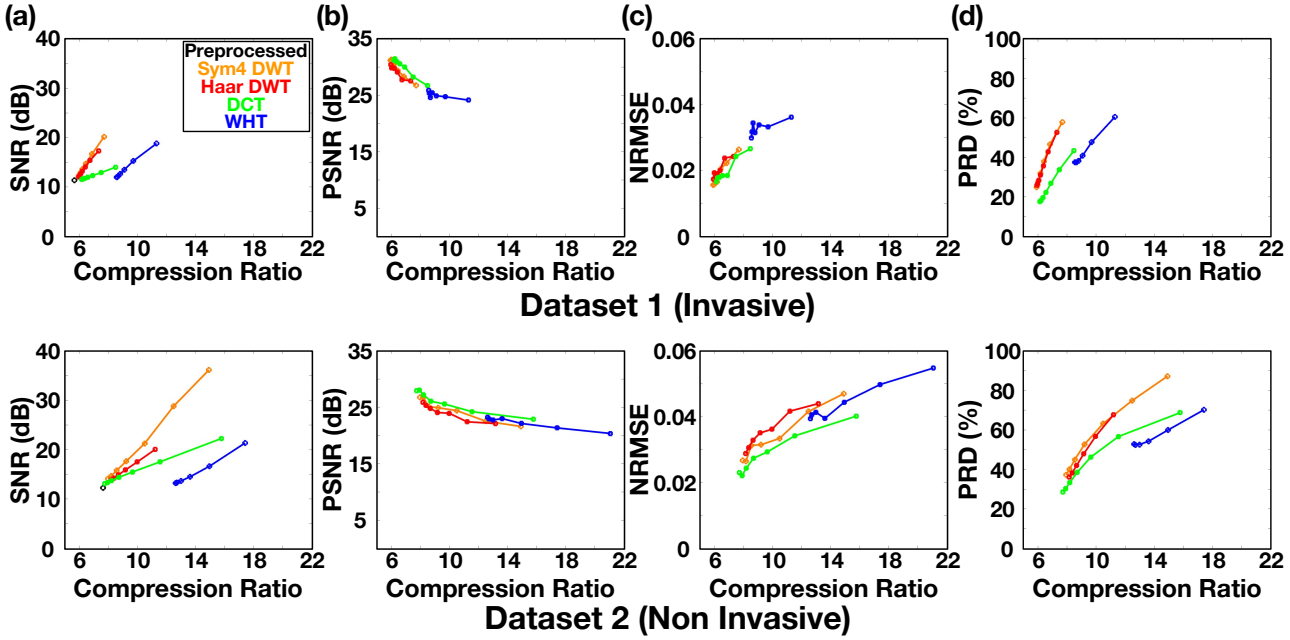

Figure 5. Signal quality metrics for transform-based compression across compression ratios. **(a)** Signal-to-noise ratio; **(b)** PSNR relative to the original signal; **(c)** Normalized root mean square error; **(d)** Percent root mean square difference.

3.3. Decoding performance

Figure 6 summarizes the decoding performance by comparing average (mean ± standard deviation across the five cross-validation folds) gesture classification accuracy across transform methods using either all 14 features or the reduced feature set. The number of retained features for each condition is shown in Figure 7. Two main observations can be made from Figure 6.

First, classification accuracy did not consistently decrease as the compression ratio increased. This agrees with our previous offline findings [34], where we observed that moderate coefficient thresholding can act as a denoising step and does not necessarily degrade decoding when the classifier operates on feature representations. Prior work has also shown that time-domain feature extraction can preserve discriminative structure for gesture decoding while substantially reducing input dimensionality [31,56–58]. Because transform-domain soft-thresholding attenuates low-magnitude coefficients that are more likely dominated by noise or non-task-related variability, moderate compression can reduce feature-level fluctuations and act as an implicit regularizer, occasionally improving decoding accuracy.

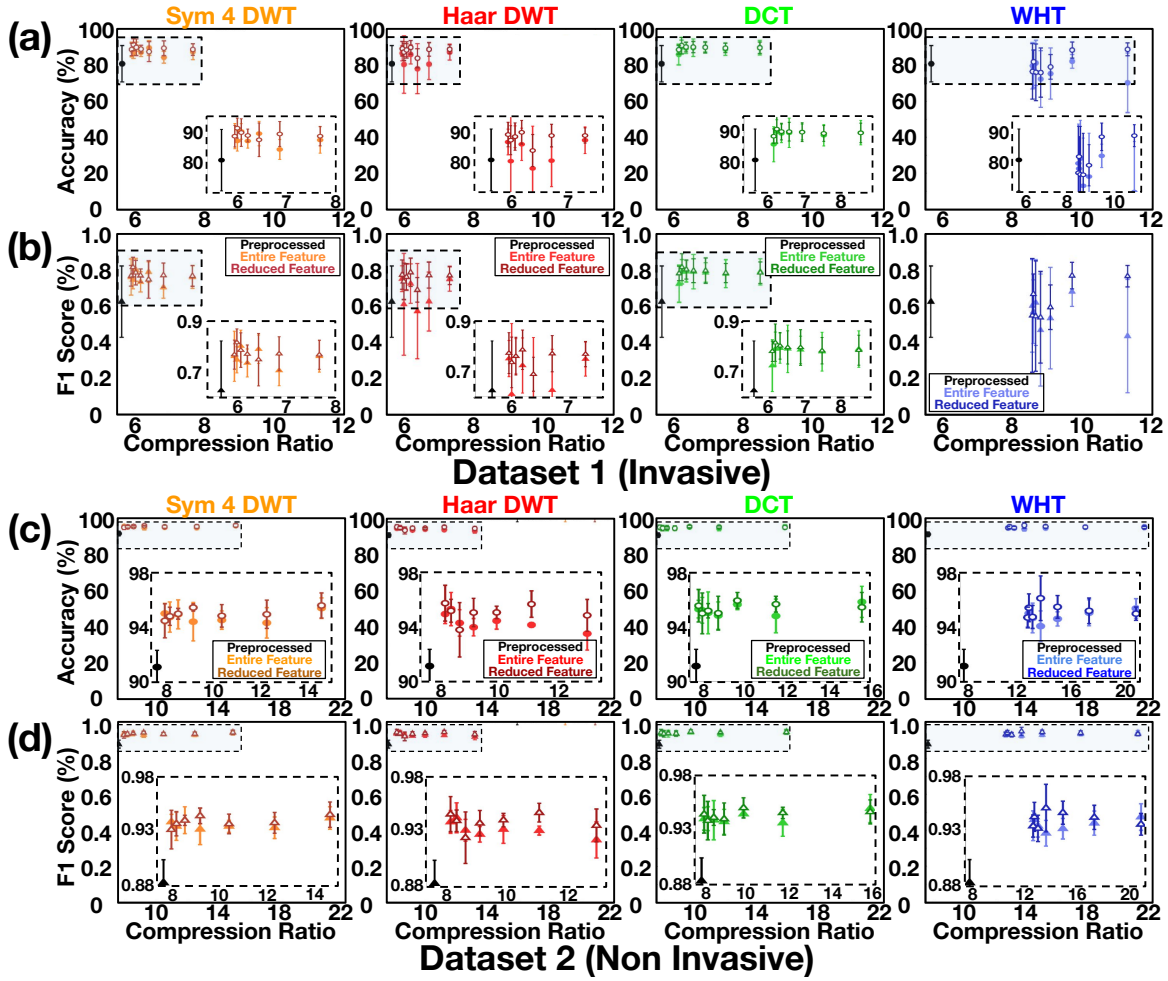

Figure 6. Classification performance for transform-based compression with the entire and reduced feature sets: **(a)** Average classification accuracy for Dataset 1; **(b)** Average F1 score for Dataset 1; **(c)** Average classification accuracy for Dataset 2; **(d)** Average F1 score for Dataset 2 Error bars indicate ± one standard deviation across the five cross-validation folds.

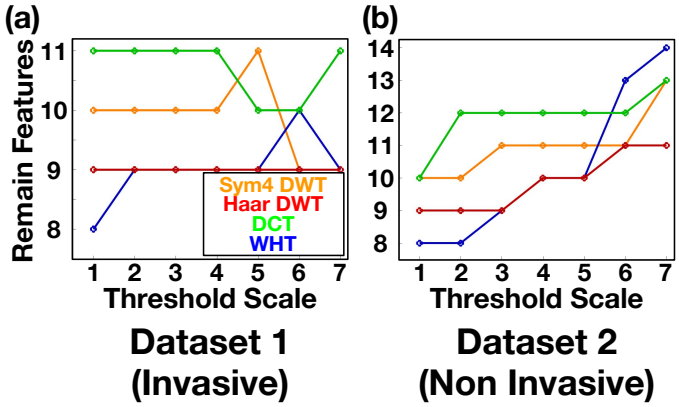

Figure 7. Number of retained features after feature reduction across threshold scales.

Second, using the reduced feature set resulted in similar or, higher classification accuracy than using all features. Notably, Haar DWT and WHT on Dataset 1 exhibited improved accuracy after feature reduction (Figure 6a), while the remaining cases, including Dataset 2, showed a similar classification performance. These findings suggest that feature reduction helps remove distortion-sensitive or unstable

features, allowing the model to train on a more consistent input representation and maintain higher F1 scores at larger compression ratios. In several cases, the transform methods achieved higher compression and also showed accuracy levels that were slightly above those of the preprocessed signal, a result that is not readily inferred from the compression and signal quality trends in Figure 4 and Figure 5. However, because accuracy does not consistently increase or decrease with compression ratio, obtaining a balanced result requires a further comprehensive evaluation. To this end, we applied the Euclidean distance metric defined in Equation (26) to jointly consider accuracy and compression ratio, and the balanced results are summarized in Figure 8 and Table 3.

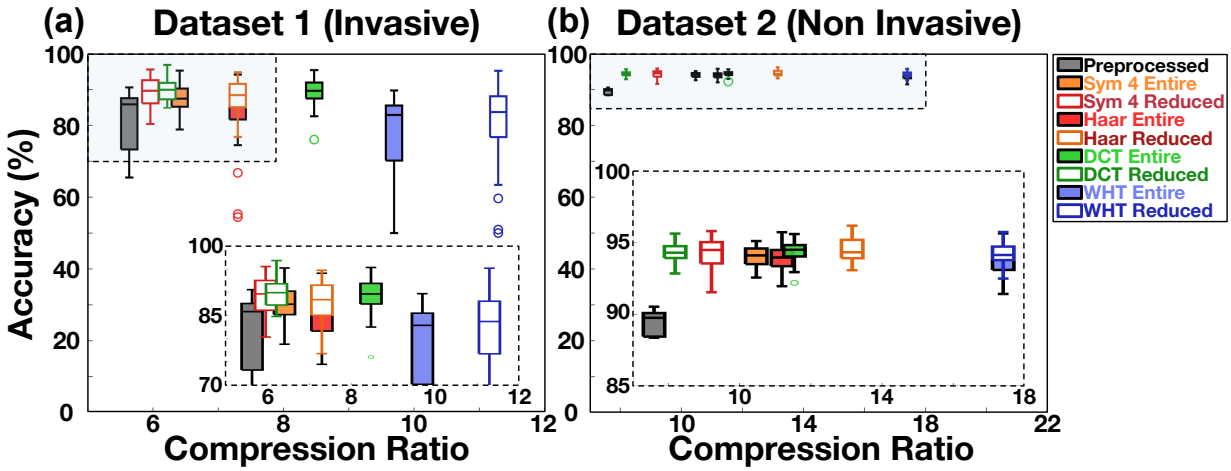

Figure 8. Balanced compression and accuracy results selected by the Euclidean distance metric: **(a)** Dataset 1; **(b)** Dataset 2. Boxplots summarize fold-wise results across the five cross-validation folds (center line: median; box: interquartile range; whiskers: 1.5× interquartile range).

Table 3. Balanced compression ratio and classification accuracy across transform methods.

Dataset	Feature Set	Preprocessed	Sym4 DWT	Haar DWT	DCT	WHT
Dataset 1	Entire[1]	CR: 5.64 Acc(%): 80.88 ± 10.19	CR: 6.41 Acc(%): 87.91 ± 3.70	CR: 7.30 Acc(%): 83.99 ± 9.26	CR: 8.47 Acc(%): 89.23 ± 3.95	CR: 9.69 Acc(%): 77.06 ± 12.51
	Reduced[2]	-	CR: 5.96 Acc(%): 89.34 ± 3.73	CR: 7.30 Acc(%): 88.14 ± 4.23	CR: 6.22 Acc(%): 89.94 ± 2.96	CR: 11.29 Acc(%): 81.03 ± 11.04
	Remaining[3]	14	10	9	11	9
Dataset 2	Entire[1]	CR: 7.60 Acc(%): 89.42 ± 0.95	CR: 10.47 Acc(%): 94.07 ± 0.67	CR: 11.19 Acc(%): 93.97 ± 0.83	CR: 11.54 Acc(%): 94.34 ± 0.77	CR: 17.40 Acc(%): 93.92 ± 0.97
	Reduced[2]	-	CR: 9.21 Acc(%): 94.28 ± 0.99	CR: 13.16 Acc(%): 94.48 ± 0.77	CR: 8.10 Acc(%): 94.37 ± 0.63	CR: 17.40 Acc(%): 94.18 ± 0.76
	Remaining[3]	14	11	11	12	13

[1]Entire = all 14 features, [2]Reduced = reduced feature set, and [3]Remaining = number of retained features. Values are reported as mean ± standard deviation across the five cross-validation folds at the balanced operating point selected by the Euclidean distance metric.

3.4. Balanced result with euclidean distance

Figure 8 applies the Euclidean distance metric to the accuracy and compression ratio shown in Figure 6a,c, and selects only the balanced results for each method and scale. In this way, Figure 8 and Table 3 together describe how each transform behaves at a practically relevant operating point where compression and decoding performance are jointly considered, rather than at extremes that favor only one objective. Confusion matrices (counts and normalized) and per-class F1 scores at the balanced operating points are provided in Supplementary Figures S1–S2.

WHT consistently provides the highest compression ratios across the four methods in both datasets, whereas DCT maintains strong decoding accuracy across conditions and remains among the best performing methods in the balanced results for both datasets. Sym4 DWT and Haar DWT show reasonable performance, but their advantages are less pronounced when accuracy and compression ratio are considered together compared with DCT and WHT.

When moving from the entire feature set to the reduced feature set, Sym4 DWT shows a small decrease in compression ratio accompanied by a modest increase in accuracy. Haar DWT and WHT generally maintain a similar or slightly higher compression ratio while improving accuracy. DCT behaves somewhat differently. With feature reduction, the upper bound of its accuracy distribution increases and the median shifts upward, but this improvement comes at the cost of a lower compression ratio. Even so, DCT still occupies a favorable region in the accuracy versus compression space, with relatively high accuracy, reasonable compression, and a comparatively tight boxplot that implies stable performance across folds. WHT, on the other hand, starts as the lowest accuracy method among the four transforms when all features are used, but after feature reduction it achieves both high compression and substantially improved accuracy. The WHT boxplots remain more dispersed than those of DCT, which suggests larger fold to fold variability and potentially lower stability in practical deployment, but they demonstrate that an aggressive compression setting can still deliver competitive decoding performance once distortion-sensitive features are removed.

Building on the balanced results in Figure 8, Figure 9 illustrates the gesture specific classification results together with a t-distributed Stochastic Neighbor Embedding (SNE) [59] embedding of the 128 dimensional hidden representation produced by the DL classifier at the final time step before the classification layer. Both the classification results and the t-SNE embeddings were computed from the best performing fold out of the five folds in Figure 8, and the preprocessed input (Figure 9a), Balanced WHT (Figure 9b), and Balanced WHT with reduced features (Figure 9c) are shown as representative examples. Since Figure 8a and Table 3 indicate that WHT resulted the lowest average accuracy among the four transform methods with Dataset 1, the highest fold within WHT was selected for visualization in Figure 9 to highlight how its feature characteristics behave at the qualitative level.

In the preprocessed case, the classifier already tracks most gesture transitions with high confidence, although occasional fluctuations appear around boundaries between gestures and some overlap remains between clusters in the t-SNE space. Using Balanced WHT features improves the separability of several gestures, particularly Thumb and Middle, but produces less stable predictions and a more dispersed Ring and Pinky cluster, consistent with the lower average accuracy in Table 4. In particular, Ring accuracy drops substantially under Balanced WHT (81.42% in Table 4), suggesting that this WHT setting

disproportionately perturbs a subset of features that are informative for Ring discrimination. After feature reduction, temporal predictions become more stable and the t-SNE clusters appear tighter and more clearly separated, especially for Ring, Pinky and Index. This recovery is consistent with the NRMSE-based feature reduction removing distortion-sensitive features and retaining a more robust feature subset, which restores Ring decoding performance (97.22% in Table 4). These qualitative changes align with the quantitative results in Table 4, where Balanced WHT with reduced features achieves the highest overall accuracy (94.88%). It consistently improves every gesture relative to the unreduced Balanced WHT, reflecting the benefit of discarding distortion-sensitive features. Therefore, Figure 9 further supports the reliability of the balanced results reported in Figure 8 by demonstrating that the decoding performance can be maintained while achieving higher compression ratio than the preprocessed baseline.

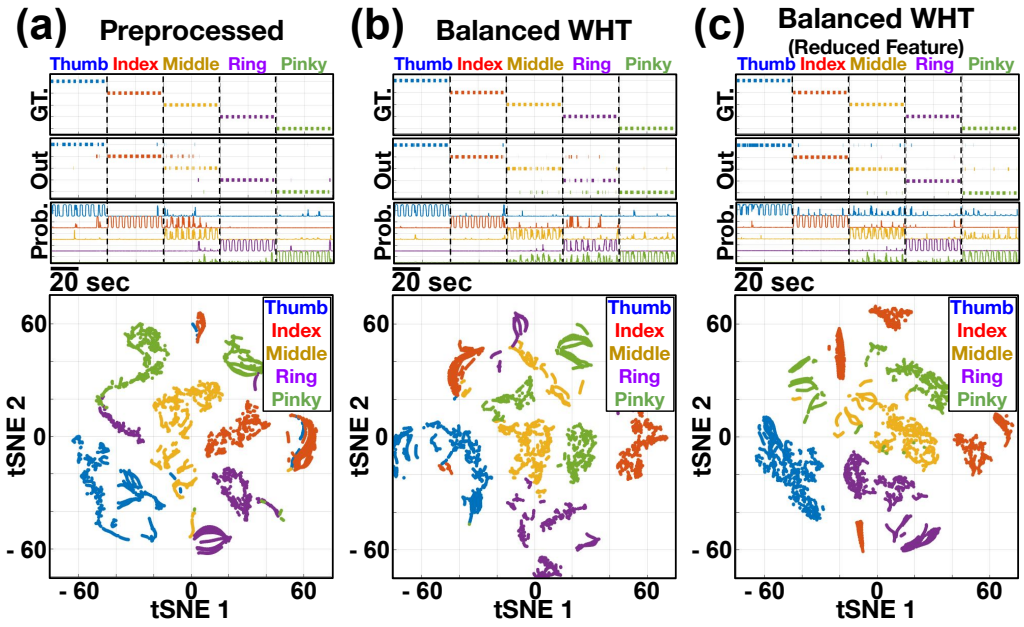

Figure 9. Gesture specific decoding and t SNE embeddings at representative balanced conditions. Temporal gesture predictions (top) and 2D t-SNE embeddings of the 128 dimensional hidden representation (bottom) for **(a)** preprocessed baseline, **(b)** Balanced WHT, **(c)** Balanced WHT with reduced features in Dataset 1. Results are shown for a representative fold selected for qualitative visualization; fold-wise distributions are summarized in Figure 8.

Table 4. Gesture wise classification accuracy for representative balanced conditions.

Method	Thumb	Index	Middle	Ring	Pinky	Average
Preprocessed	89.73%	96.72%	83.12%	95.70%	95.76%	92.21%
Balanced WHT	96.00%	92.52%	86.79%	81.42%	94.45%	90.23%
Balanced WHT (Reduced feature)	96.40%	96.46%	92.67%	97.22%	91.65%	94.88%

Values correspond to the representative fold used for Figure 9 and are not averaged across folds.

4. Discussion

In prior work, high compression ratios were often achieved by discarding data through spike extraction or heavy sparsification, and compression quality was mainly evaluated using reconstruction based signal metrics, leaving the practical impact on downstream BCI decoding uncertain. Table 5 provides context by summarizing representative prior studies, including the recording setup and reported compression ratios, which vary substantially depending on the study objective and the amount of signal content preserved. Comparisons of signal quality metrics are excluded because they are not consistently reported or defined across studies. In addition, the very high compression ratios reported by learned compressors (e.g., autoencoders [21–23]) are often obtained under different recording and encoding paradigms (e.g., spike waveform compression after spike extraction, or highly sparse activity), and are therefore not directly comparable to the continuous, task-driven peripheral nerve recordings used for feature-based motor decoding in this study. In this study, we address this gap by evaluating transform-based lossy compression directly in the context of a decoding model, so that practicality is assessed by whether motor intention can still be decoded from the compressed representation. Accordingly, we focus on consistent within-dataset comparisons (Figures 4–9), while direct benchmarking against learned compressors under the same dataset and task setting remains future work. We refer to this pipeline as CBFR, where lossy compression is followed by feature extraction and feature reduction, and then evaluated with a DL classifier. With this perspective, we organize the discussion around three questions: (i) how varying thresholds of signal compression affects compression efficiency and signal distortion, (ii) when decoding performance remains stable under compression and feature reduction, and (iii) which transform and threshold scale provide balanced results where compression and accuracy are jointly maintained.

Table 5. Literature meta analysis of datasets and reported compression ratios.

Subject	Data type	Sampling rate	Number of channel	Recording task	Spike extraction	Compression method	CR	Reference
Guinea pig	Intracortical recording (Auditory cortex)	20 kHz	64	NR	No	Sym4 DWT	1.78×	[45]
					No	Haar DWT	1.78×	
					Yes	Sym4 DWT	108×	
					Yes	Haar DWT	112×	
Guinea pig	Intracortical recording (Auditory cortex)	25 kHz	128	NR	No (hard thresholding; near spikes only)	WHT	63×	[47]
						DCT	69×	[48]
Rat	Hippocampal CA1 recording	10 kHz	4	NR	Yes	CNN autoencoder	20–500×	[21]
Epileptic patient	Single cell waveform recording	NR	NR	NR	Yes	Autoencoder	2–16×	[22]
Healthy human	**Non invasive neural signal**	**10 kHz**	**16**	**Finger movement**	**No**	**DCT**	**7.73–15.8×**	**Our work**
						WHT	**12.60–21.08×**	

CR denotes compression ratio.
NR indicates not reported.

4.1. Compression and distortion

We first examine how threshold scale affects compression efficiency and entropy, and why these trends alone are insufficient to select an operating point for decoding. Figure 4 shows that increasing the threshold scale increases the compression ratio. However, entropy and reconstruction based signal quality metrics in Figure 5 are not sufficient to fully evaluate the true effectiveness of each compression method from a task relevant perspective. For example, WHT produces the highest compression ratio with relatively small changes in entropy, yet its PSNR, NRMSE, and PRD indicate larger distortion than DCT. These observations motivate an evaluation that directly reflects decoding utility rather than relying only on reconstruction based metrics.

4.2. Decoding and feature reduction

To evaluate compression in a task relevant manner, Figure 6 applies feature extraction and trains a DL classifier on the compressed signals. Two findings emerge. First, transform-based compression can still achieve similar or higher decoding performance than the uncompressed preprocessed condition in several cases, indicating that higher compression does not necessarily translate to poorer decoding performance when the model operates on feature representations. Second, when feature reduction is applied after feature extraction, decoding performance often remains similar or improves for certain methods. Together, these outcomes suggest that compression combined with feature reduction can remove distortion-sensitive or unstable features and provide a more consistent feature representation to the classifier. A plausible explanation is that transform-domain thresholding suppresses low-energy components that contribute non-informative variability, while the subsequent NRMSE-based feature reduction removes features whose values are disproportionately perturbed by reconstruction error, yielding a regularized and more stable input to the classifier. To assess whether method rankings depend on using method-specific reduced feature sets, we additionally evaluated a common reduced feature set shared across methods defined in Supplementary Table S1 and evaluated in Supplementary Table S2.

While Figure 6 demonstrates that decoding can be preserved under compression, it does not by itself specify which method and threshold scale should be selected in a practical setting. To enable systematic selection, we apply the Euclidean distance defined in Equation (26) to identify a balanced result that reflects both high compression ratio and high classification accuracy. Figure 8 and Table 3 visualize these balanced results across methods and scales. In addition, Figure 9 and Table 4 present the gesture specific classification performance and t-SNE embeddings of the hidden state before the classification layer for representative balanced conditions. These visualizations support that the balanced points correspond to conditions where high compression and high decoding performance can be jointly maintained while producing well separated gesture representations.

Based on the balanced results, DCT based CBFR can be considered a strong candidate among the evaluated methods because it maintains high compression ratio and high decoding performance for both invasive and non-invasive datasets while exhibiting relatively lower distortion in the signal quality metrics. At the same time, Haar DWT or WHT can also be effective choices depending on the constraints and objectives of the experimental setting, such as when a hardware friendly transform implementation and a training free, low complexity pipeline (Table 2) or when a larger compression ratio is prioritized,

since both Haar DWT and WHT have been widely used in hardware oriented realizations with efficient FPGA or Very Large-Scale Integration (VLSI) architectures [60,61]. Because this study is evaluated in software, we do not report direct on-device latency or power measurements for CBFR or learned compressor baselines. Haar DWT provides stable decoding performance in both the entire feature and reduced feature conditions, while WHT offers the highest compression ratio but with larger distortion and greater variability in decoding outcomes, highlighting a complementary tradeoff between robustness and maximum compression.

4.3. Key contributions

Compared to previous approaches that often extract only spike based responses and evaluate compression quality mainly using signal based metrics such as SNR, our study takes a step further by applying lossy compression to neural signals that contain real movement intention and by evaluating not only compression efficiency and signal quality but also decoding performance. To our knowledge, transform-based lossy compression combined with DL decoding has been applied only in limited cases to peripheral neural signals, and our work provides one of the first examples that systematically examines how transform-based compression affects both the stochastic characteristics of the signal and the decoding accuracy for movement intention.

4.4. Limitation and future work

In the present study, the CBFR pipeline was implemented in a real time capable software setting using frame based streaming of recorded datasets. In future work, this pipeline can be deployed on a microcontroller for the compression stage, while the trained DL classifier can be deployed on an inference-capable embedded GPU such as Nvidia Jetson, consistent with our prior portable peripheral nerve decoding demonstration [15]. In addition, multiple prior studies have reported hardware-feasible implementations of transform-domain neural signal processing and compression, including low-complexity and low-power architectures based on Haar DWT, WHT, and DCT [17,18,46–48]. However, because this study evaluates the pipeline in software using recorded datasets, end-to-end latency, power, and robustness under wireless constraints remain to be validated on device. Therefore, the next step is to implement CBFR on actual hardware and evaluate its performance in real world settings. To place these results in context and outline the remaining steps toward deployment, we summarize the key limitations and future work below.

The current evaluation is conducted in software using session-level cross-validation, which primarily assesses within-subject generalization across sessions. While each dataset contains multiple sessions and a substantial total volume of raw recordings (Table 1), the limited number of participants does not capture inter-subject variability, and we do not claim cross-subject generalization with a fixed CBFR configuration. In practice, decoder training and CBFR parameter selection may require subject-specific personalization, and end-to-end validation on embedded hardware remains necessary to quantify latency and power under wireless constraints. Future work will therefore focus on (i) hardware implementation and on-device evaluation, including end-to-end latency and power benchmarking, (ii) cohort-level validation across participants, and (iii) efficient personalization/calibration strategies to support practical deployment, as well as direct comparisons to learned compressors under the same peripheral-nerve decoding task.

5. Conclusion

This study presented CBFR, a transform-based lossy compression and feature selection pipeline for BCI devices, and evaluated its practicality using both invasive and non-invasive datasets during finger movement decoding. Across both datasets, CBFR substantially reduced the data rate while maintaining decoding performance in this experimental setting, achieving up to 11.29× compression on invasive recordings and about 21.08× on non-invasive recordings, while accuracy was preserved or improved relative to the preprocessed baseline. Feature reduction applied after feature extraction indicated that removing distortion-sensitive or unstable features can yield a more consistent representation for deep-learning-based decoding. Among the evaluated transforms, the DCT provided consistently strong balanced results, whereas the WHT offered the highest compression ratios with greater variability, indicating complementary choices depending on whether robustness or maximum compression is prioritized. Overall, these results support transform-based lossy compression combined with feature reduction as a viable approach to reduce wireless bandwidth demands for peripheral nerve interfaces, with future work focused on validating end-to-end performance on embedded hardware. Inter-subject validation across a larger cohort and evaluation of subject-specific personalization remain important next steps toward practical deployment.

Supplementary data

The supplementary tables and figures are available in the supplementary materials associated with this paper. The Supplementary Material provides (i) the identities of the retained feature sets after NRMSE-based feature reduction, (ii) the evaluation using common reduced feature sets shared across methods, and (iii) confusion matrices and per-class F1 scores for the balanced operating points reported in the main manuscript.

Data availability statement

The datasets referred to in the Methods Section—namely, invasive data (Dataset 1) and non-invasive data (Dataset 2)—are not publicly available due to proprietary and authorization restrictions.

Declaration of generative AI and AI-assisted technologies

During the preparation of this manuscript, the authors used generative AI tools only to improve language and readability. The authors take full responsibility for the content of the manuscript.

Acknowledgments

This work was supported by Fasikl, Inc.

Authors' contribution

Byeongchan Jeong: conceptualization, literature investigation, methodology, software, formal analysis, writing—original draft preparation; Anh Tuan Nguyen: data curation, software, methodology, writing—review and editing; Tong Wu: conceptualization, methodology; Brian Lim: methodology; Zhi Yang: funding acquisition, conceptualization, supervision, writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Conflicts of interest

Zhi Yang is co-founder of, and holds equity in, Fasikl Incorporated, a sponsor of this project. This interest has been reviewed and managed by the University of Minnesota in accordance with its Conflict of Interest policy.

References

[1] Pandarinath C, Nuyujukian P, Blabe CH, Sorice BL, Saab J, *et al.* High performance communication by people with paralysis using an intracortical brain-computer interface. *eLife* 2017, 6:e18554.

[2] Nguyen AT, Jeong B, Lim BZH, Drealan MW, Su K, *et al.* Lossless compression methods for wireless, real-time streaming of neural data from neural interface devices. *IEEE Sens. J.* 2025, 26(2):2511–2523.

[3] Wolpaw JR. Brain-computer interfaces (BCIs) for communication and control. In *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility*, Tempe, USA, October 15–17, 2007, pp. 1–2.

[4] Navarro X, Krueger TB, Lago N, Micera S, Stieglitz T, *et al.* A critical review of interfaces with the peripheral nervous system for the control of neuroprostheses and hybrid bionic systems. *J. Peripher. Nerv. Syst.* 2005, 10(3):229–258.

[5] Goris RL, Movshon JA, Simoncelli EP. Partitioning neuronal variability. *Nat. Neurosci.* 2014, 17(6):858–865.

[6] Marder E. Variability, compensation, and modulation in neurons and circuits. *Proc. Natl. Acad. Sci.* 2011, 108(supplement_3):15542–15548.

[7] Fee MS, Mitra PP, Kleinfeld D. Variability of extracellular spike waveforms of cortical neurons. *J. Neurophysiol.* 1996, 76(6):3823–3833.

[8] Smith CE, Evans NK. Stochastic processes in the neurosciences. 1986. Available: https://repository.lib.ncsu.edu/server/api/core/bitstreams/98380c00-f258-44d8-b4e2-f512f9089cca/content (accessed on 25 March 2026).

[9] Del Valle J, Navarro X. Interfaces with the peripheral nerve for the control of neuroprostheses. *Int. Rev. Neurobiol.* 2013, 109:63–83.

[10] Maynard EM, Nordhausen CT, Normann RA. The Utah intracortical electrode array: a recording structure for potential brain-computer interfaces. *Electroencephalogr. Clin. Neurophysiol.* 1997, 102(3):228–239.

[11] Chapin JK, Moxon KA, Markowitz RS, Nicolelis MA. Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex. *Nat. Neurosci.* 1999, 2(7):664–670.

[12] Larson CE, Meng E. A review for the peripheral nerve interface designer. *J. Neurosci. Methods* 2020, 332:108523.

[13] Salomon D. Data compression. In *Handbook of massive data sets*, New York: Springer 2002. pp. 245–309.

[14] Elakkiya S, Thivya K. Comprehensive review on lossy and lossless compression techniques. *J. Inst. Eng. India Ser. B* 2022, 103(3):1003–1012.

[15] Nguyen AT, Drealan MW, Luu DK, Jiang M, Xu J, *et al.* A portable, self-contained neuroprosthetic hand with deep learning-based finger control. *J. Neural Eng.* 2021, 18(5):056051.

[16] Luu DK, Nguyen AT, Jiang M, Drealan MW, Xu J, *et al.* Artificial intelligence enables real-time and intuitive control of prostheses via nerve interface. *IEEE Trans. Biomed. Eng.* 2022, 69(10):3051–3063.

[17] Oweiss KG. A systems approach for data compression and latency reduction in cortically controlled brain machine interfaces. *IEEE Trans. Biomed. Eng.* 2006, 53(7):1364–1377.

[18] Oweiss KG, Mason A, Suhail Y, Kamboh AM, Thomson KE. A scalable wavelet transform VLSI architecture for real-time signal processing in high-density intra-cortical implants. *IEEE Trans. Circuits Syst. I Regul. Pap.* 2007, 54(6):1266–1278.

[19] Kamboh AM, Oweiss KG, Mason AJ. Resource constrained VLSI architecture for implantable neural data compression systems. In *Proceedings of 2009 IEEE International Symposium on Circuits and Systems*, Taipei, China, May 24–27, 2009, pp. 1481–1484.

[20] Daou H, Labeau F. Pre-processing of multi-channel EEG for improved compression performance using SPIHT. In *Proceedings of 2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, San Diego, USA, August 28–September 1, 2012, pp. 2232–2235.

[21] Wu T, Zhao W, Keefer E, Yang Z. Deep compressive autoencoder for action potential compression in large-scale neural recording. *J. Neural Eng.* 2018, 15(6):066019.

[22] Thies J, Alimohammad A. Compact and low-power neural spike compression using undercomplete autoencoders. *IEEE Trans. Neural Syst. Rehabil. Eng.* 2019, 27(8):1529–1538.

[23] Valencia D, Mercier PP, Alimohammad A. Efficient *in vivo* neural signal compression using an autoencoder-based neural network. *IEEE Trans. Biomed. Circuits Syst.* 2024, 18(3):691–701.

[24] Sriraam N. A high-performance lossless compression scheme for EEG signals using wavelet transform and neural network predictors. *Int. J. Telemed. Appl.* 2012, 2012(1):302581.

[25] Jiman AA, Ratze DC, Welle EJ, Patel PR, Richie JM, *et al.* Multi-channel intraneural vagus nerve recordings with a novel high-density carbon fiber microelectrode array. *Sci. Rep.* 2020, 10(1):15501.

[26] Sinkjær T, Yoshida K, Jensen W, Schnabel V. Electroneurography. In *Encyclopedia Of Medical Devices And Instrumentation*, 2nd ed. Hoboken: Wiley Online Library, 2006.

[27] Carboni C, Bisoni L, Carta N, Puddu R, Raspopovic S, *et al.* An integrated interface for peripheral neural system recording and stimulation: system design, electrical tests and in-vivo results. *Biomed. Microdevices* 2016, 18(2):35.

[28] Donohoe M, Jennings B, Balasubramaniam S. Capacity analysis of a peripheral nerve using modulated compound action potential pulses. *IEEE Trans. Commun.* 2018, 67(1):154–164.

[29] Citi L, Carpaneto J, Yoshida K, Hoffmann KP, Koch KP, *et al.* On the use of wavelet denoising and spike sorting techniques to process electroneurographic signals recorded using intraneural electrodes. *J. Neurosci. Methods* 2008, 172(2):294–302.

[30] Tarotin I, Aristovich K, Holder D. Effect of dispersion in nerve on compound action potential and impedance change: a modelling study. *Physiol. Meas.* 2019, 40(3):034001.

[31] Luu DK, Nguyen AT, Jiang M, Xu J, Drealan MW, *et al.* Deep learning-based approaches for decoding motor intent from peripheral nerve signals. *Front. Neurosci.* 2021, 15:667907.

[32] Dian FJ, Yousefi A, Lim S. A practical study on bluetooth low energy (BLE) throughput. In *Proceedings of 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Vancouver, Canada, November 1–3, 2018, pp. 768–771.

[33] Woolley M. Bluetooth core 5.0 feature enhancements. 2025. Available: chromeextension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.bluetooth.com/wp-content/uploads/2019/03/Bluetooth_5-FINAL.pdf (accessed on 25 March 2026).

[34] Jeong B, Wu T, Nguyen AT, Xu J, Cheng J, *et al.* Compression-based feature reduction for upper limb motor decoding from peripheral neural signal. In *Proceedings of 2024 IEEE 20th International Conference on Body Sensor Networks (BSN)*, Salt Lake City, USA, October 15–17, 2024, pp. 1–4.

[35] Ahmed N, Rao KR. *Orthogonal transforms for digital signal processing*, 1st ed. Berlin: Springer Science & Business Media, 2012.

[36] Getz NH. A fast discrete periodic wavelet transform. 1992. Available: https://digicoll.lib.berkeley.edu/record/135028?v=pdf&ln=en (accessed on 25 March 2026).

[37] Debnath L. Fourier transforms and their applications. In *Wavelet Transforms and Their Applications*, 1st ed. Boston: Springer, 2002. pp. 143–256.

[38] Alessio SM. *Digital Signal Processing And Spectral Analysis For Scientists: Concepts And Applications*, 1st ed. Switzerland: Springer, 2015.

[39] Širca S, Horvat M. Transformations of functions and signals. In *Computational Methods in Physics: Compendium for Students*, 3rd ed. Switzerland: Springer, 2025. pp. 243–312.

[40] Natarajan T, Ahmed N. Performance evaluation for transform coding using a nonseparable covariance model. *IEEE Trans. Commun.* 2003, 26(2):310–312.

[41] Cardoso G, Saniie J. Performance evaluation of DWT, DCT, and WHT for compression of ultrasonic signals. In *Proceedings of IEEE Ultrasonics Symposium, 2004*, Montreal, Canada, August 23–27, 2004, pp. 2314–2317.

[42] Vellaiappan E, Angam P. Comparison of dct and wavelets in image coding. In *IEEE International Conference on Computer Communication and Informatics*, Tamil Nadu, India, January 10–12, 2012, pp. 136–139.

[43] Wang Y, Mukherjee D. The discrete cosine transform and its impact on visual compression: fifty years from its invention [perspectives]. *IEEE Signal Process. Mag.* 2023, 40(6):14–17.

[44] Rodriguez VH, Medrano C, Plaza I. A real-time QRS complex detector based on discrete wavelet transform and adaptive threshold as standalone application on ARM microcontrollers. In *Proceedings of 2018 International Conference on Biomedical Engineering and Applications (ICBEA)*, Funchal, Portugal, July 9–12, 2018, pp. 1–6.

[45] Shaeri MA, Sodagar AM, Abrishami-Moghaddam H. A 64-channel neural signal processor/compressor based on Haar wavelet transform. In *Proceedings of 2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Boston, USA, August 30–September 3, 2011, pp. 6409–6412.

[46] Shaeri MA, Sodagar AM. A method for compression of intra-cortically-recorded neural signals dedicated to implantable brain–machine interfaces. *IEEE Trans. Neural Syst. Rehabil. Eng.* 2014, 23(3):485–497.

[47] Hosseini-Nejad H, Jannesari A, Sodagar AM. Data compression in brain-machine/computer interfaces based on the Walsh–Hadamard transform. *IEEE Trans. Biomed. Circuits Syst.* 2013, 8(1):129–137.

[48] Hosseini-Nejad H, Jannesari A, Sodagar AM, Rodrigues JN. A 128-channel discrete cosine transform-based neural signal processor for implantable neural recording microsystems. *Int. J. Circuit Theory Appl.* 2015, 43(4):489–501.

[49] Donoho DL. De-noising by soft-thresholding. *IEEE Trans. Inf. Theory* 2002, 41(3):613–627.

[50] Aloui N, Bousselmi S, Cherif A. Speech compression based on discrete Walsh Hadamard transform. *Int. J. Inf. Eng. Electron. Bus.* 2013, 3:59–65.

[51] Neymotin SA, Tal I, Barczak A, O'Connell MN, McGinnis T, *et al.* Detecting spontaneous neural oscillation events in primate auditory cortex. *Eneuro* 2022, 9(4).

[52] Witten IH, Neal RM, Cleary JG. Arithmetic coding for data compression. *Commun. ACM* 1987, 30(6):520–540.

[53] Nordic Semiconductor. nRF52840 Product Specification. 2024. Available: https://docs.nordicsemi.com/bundle/ps_nrf52840/page/keyfeatures_html5.html (accessed on 25 March 2026).

[54] Phinyomark A, Quaine F, Charbonnier S, Serviere C, Tarpin-Bernard F, *et al.* EMG feature evaluation for improving myoelectric pattern recognition robustness. *Expert Syst. Appl.* 2013, 40(12):4832–4840.

[55] Chung J, Gulcehre C, Cho K, Bengio Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* 2014, arXiv:1412.3555.

[56] Zardoshti-Kermani M, Wheeler BC, Badie K, Hashemi RM. EMG feature evaluation for movement control of upper extremity prostheses. *IEEE Trans. Rehabil. Eng.* 1995, 3(4):324–333.

[57] Phinyomark A, Nuidod A, Phukpattaranont P, Limsakul C. Feature extraction and reduction of wavelet transform coefficients for EMG pattern classification. *Elektron. ir Elektrotechn.* 2012, 122(6):27–32.

[58] Rafiee J, Rafiee M, Yavari F, Schoen MP. Feature extraction of forearm EMG signals for prosthetics. *Expert Syst. Appl.* 2011, 38(4):4058–4067.

[59] Van der Maaten L, Hinton G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* 2008, 9(11).

[60] Fakruddin DB, Parthasarathy K. Simplified algorithms based on Haar transforms for signal recognition in protective relays. *Proc. IEEE* 2005, 73(5):940–942.

[61] Amira A, Bouridane A, Milligan P, Roula M. An FPGA implementation of Walsh-Hadamard transforms for signal processing. In *Proceedings of 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)*, Salt Lake City, USA, May 7–11, 2001, pp. 1105–1108.