Hardware-Efficient Compression of Neural Multi-Unit Activity Using Machine Learning Selected Static Huffman Encoders

Oscar W. Savolainen[1,3], **Zheng Zhang**[1,3], **Peilong Feng**[1], **Timothy G. Constandinou**[1,2]

[1] Department of Electrical and Electronic Engineering, Imperial College London, South Kensington Campus, London SW7 2AZ, UK

[2] UK Dementia Research Institute (UKDRI) Care Research & Technology (CR&T) Centre, based at Imperial College London and the University of Surrey, UK

[3] These authors contributed equally to this work.

E-mail: o.savolainen18@imperial.ac.uk

Abstract.

Objective: Recent advances in intracortical brain machine interfaces (iBMIs) have demonstrated the feasibility of using our thoughts; by sensing and decoding neural activity, for communication and cursor control tasks. It is essential that any invasive device is completely wireless so as to remove percutaneous connections and the associated infection risks. However, wireless communication consumes significant power and there are strict heating limits in cortical tissue. Most iBMIs use Multi Unit Activity (MUA) processing, however the required bandwidth can be excessive for large channel counts in mm or sub-mm scale implants. As such, some form of data compression for MUA iBMIs is desirable. *Approach:* We used a Machine Learning approach to select static Huffman encoders that worked together, and investigated a broad range of resulting compression systems. They were implemented in reconfigurable hardware and their power consumption, resource utilization and compression performance measured. *Main Results:* Our design results identified a specific system that provided top performance. We tested it on data from 3 datasets, and found that, with less than 1% behavioural decoding performance reduction from peak, the communication bandwidth was reduced from 1 kb/s/channel to approximately 27 bits/s/channel, using only a Look-Up Table and a 50 ms temporal resolution for threshold crossings. Relative to raw broadband data, this is a compression ratio of 1700-15,000 $\times$ and is over an order of magnitude higher than has achieved before. Assuming 20 nJ per communicated bit, the total compression and communication power was between 1.37 and 1.52 μW/channel, occupying 246 logic cells and 4 kbit RAM supporting up to 512-channels. *Significance:* We show that MUA data can be significantly compressed in a hardware efficient manner, 'out of the box' with no calibration necessary. This can significantly reduce on-implant power consumption and enable much larger channel counts in WI-BMIs. All results, code and hardware designs have been made publicly available.

Keywords: Brain Machine Interfaces, Multi-Unit Activity, Firing Rate Compression, Real-time Signal Processing, Embedded Systems, Reconfigurable Hardware

1 Introduction

1.1 Wireless Intracortical Brain-Machine Interfaces

Brain–machine interfaces (BMIs) are electronic devices that measure neural activity, extract features from that activity, and convert those features into outputs that replace, restore, enhance, supplement, or improve human functions. They can, for example, be used to treat paraplegia, quadriplegia, movement disorders, Locked-in syndrome and more [1]. Intracortical BMIs (iBMIs) are the most invasive form of BMI, where electrodes are placed into brain tissue [2]. They also provide the highest resolution of BMI data, capable of measuring the firing rates of individual neurons in the electrodes' vicinity.

For iBMIs, wireless communication and powering is becoming increasingly essential. This eliminates physical percutaneous connections, e.g. wires breaching the skin, and associated risks (infection, mechanical damage, etc). They also enable chronic powering without having to replace the implant, as battery life is limited. However, due to heating constraints in cortical tissue, power is strictly limited in Wireless Intracortical BMIs (WI-BMIs) to an approximately 1 C temperature increase or 1.6 mW/g of specific absorption rate (SAR) in tissue [2–4]. In the context of heating due to absorption of radio frequencies (RF), the IEEE standard C95.1-2019 gives limits for SAR heating depending on RF frequency [5]. However, it specifies that the understood SAR limits in brain tissue are generally derived from models and lack rigorous studies in live animals or humans, with significant variance between models [5]. FDA regulations further dictate that the local heat increase of brain tissue due to intracortical implants should be limited to only 0.5 C [6]. In muscle and lung tissue, it is understood that up to 40 mW/cm^2 heat flux can be allowed, however the limit is likely lower in cortical tissue [3]. For this work, we will assume a maximum heat flux limit of 10 mW/cm^2 to hopefully provide a reasonable safety margin, given the extreme paucity of data on the effects of chronic heat flux on cortical tissue [3,7]. Even with heating rates for non-cortical tissue, the heating limitation in WI-BMIs is so severe that methods to reduce on-implant power use are highly desirable.

The typical data flow and power-resource-performance optimisation problem for BMIs is shown in Fig. 1. In this work, we seek to reduce the on-implant power via data compression with minimal resources, while maximising the Behavioral Decoding Performance (BDP). The communication rate is equal to the temporal resolution of the neural and decoded behavioral data i.e. Behavioral Data Temporal Resolution (BDTP). As such, it should also be kept in a practical range (e.g. 1-100 ms). If it is too high, unnecessary detail is communicated; e.g. decoding the behavior at 1 ns intervals would be unnecessary. If this is however too low, the decoder will not output at a sufficient temporal resolution.

1.2 Communication Power

The wireless data communication channels for brain-machine interfaces can be described as uplink and downlink. The uplink refers to the data flow from implants to external devices, and the downlink is the opposite flow direction. Usually, the brain-machine interfaces

require a higher uplink data rate than that of downlink, as uplink needs to transmit a significant amount of recorded neural data. Due to the strict heat limits, a power-efficient data communication scheme is essential to brain-machine interfaces.

Recently, researchers have developed various communication microsystems for brain-machine interfaces. The common data communication schemes are implemented using different shift keying (amplitude, phase, on-off) [8–12]. They are the most power-efficient solutions for implants; however, their Bit Rates (BR), i.e. data rates, are below 20 Mbps. The ultra-wideband uplink proposed in [13] achieved 46 Mbps with 118.3 pJ/bit. The implantable microsystems that are designed and fabricated based on full-custom application specific integrated (ASIC) are more power-efficient than microcontroller and field-programmable gate array (FPGA) based solutions.

Table 1: Wireless data communication power consumption comparison

Publication	Downlink			Uplink		
	Modulation	Bit rate	Power	Modulation	Bit Rate	Power
[8]	ASK	15 kbps	-	Backscatter	-	-
[9]	OOK-PPM	50 kbps	4 nJ/bit	OOK	6.78 Mbps	1.34 nJ/bit
[10]	OOK	1 Mbps	13 pJ/bit	OOK	16 Mbps	50.4 pJ/bit
[11]	ASK	11 Kbps	1.25 nJ/bit	-	-	-
[13]	-	-	-	UWB	46 Mbps	118.3 pJ/bit
[12]	ASK-PWM	1 Mbps	-	BPSK	10 Mbps	-
[14]	-	-	-	Backscatter	0.5 Mbps	240 pJ/bit
[15]	Nordic - nRF24L01+	2 Mbps	16.95 nJ/bit	Nordic - nRF24L01+	2 Mbps	20 nJ/bit
[16]	-	-	-	Backscatter	1.25 Mbps	87 nJ/bit

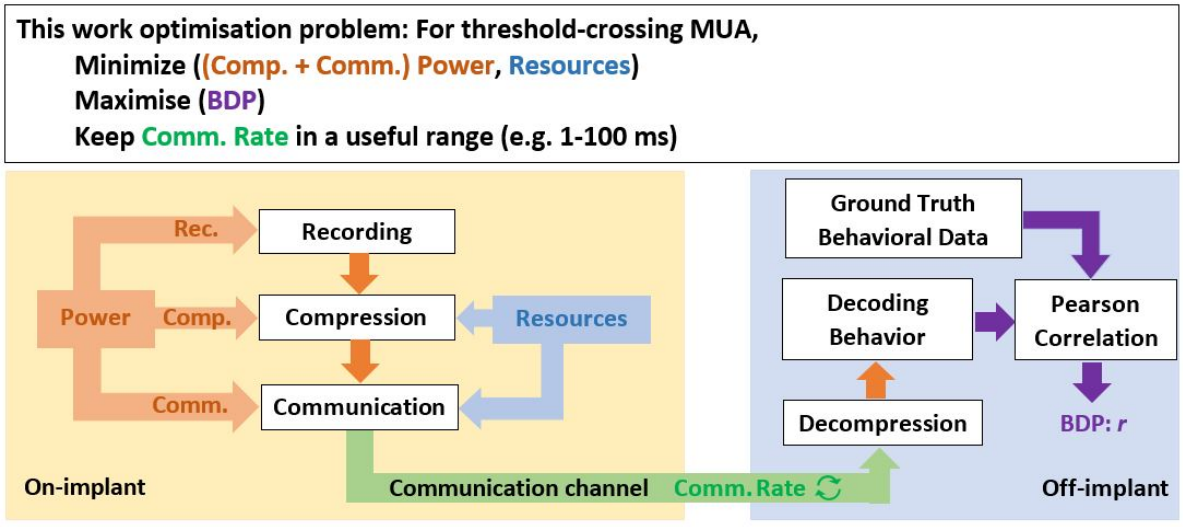


Figure 1: Data flow in behavior-decoding BMIs, and optimisation problem of on-implant power, resources and behavioral decoding performance (BDP). The communication rate, i.e. the temporal resolution of the neural and decoded behavioral data, should also be kept in a practical range based on the desired temporal resolution of the decoded behavioral data.

ASIC designs are optimal in terms of minimising power consumption and chip area.

However, the design process of FPGAs is significantly less expensive and easier compared to that of ASICs. FPGAs also benefit from increased flexibility for programming, which accommodates algorithmic changes better. As a result, it is typical for researchers to use FPGAs to validate the ASIC's performance before full development of the ASIC.

Therefore the system in this work is implemented in reconfigurable hardware (FPGA), and so the fairest comparison would be to the system in [15] which achieved a 20 nJ/bit uplink. As such, this work assumes a system communication energy to be 20 nJ/bit. The average communication power per channel can then be calculated from the BR, given in (bits/s/channel):

$$\text{Comm. power} = \text{BR} \times \text{Comm. energy per bit} \tag{1}$$

For a single channel, the BR is simply given by:

$$\text{BR} = \frac{avlen}{\text{BP}} \tag{2}$$

where $avlen$ is the average length of each symbol in a binning period (BP) in bits.

1.3 Data Compression

Raw broadband data is typically sampled at 25-30 kHz and 16 bits/sample [1,17–19], although resolutions as low as 7 kHz and 6 bits/sample are possible for applications such as spike detection of large spikes [20]. This creates very large communication bandwidths, ranging from 42-480 kb/s/channel. For 20 nJ/bit, this translates into a communication power demand of 0.84-9.6 mW/channel. Given heat flux limits, even for that in muscle tissue of 40 mW/cm^2, this severely limits the number of channels.

Millimeter or sub-mm scales are desired for WI-BMIs, mainly due to them reducing the likelihood of mechanical injury and foreign body response in the brain [2, 21–24]. As such, this work considered a 2.5 mm $\times$ 2.5 mm FPGA chip, discussed further in Section 2.4.

At a heat flux limit of 10 mW/cm^2, such an implant would have a maximum power budget of B = 625μW. As such, even a single broadband channel could not be communicated, as 42 kbits/s/channel $\times$ 20 nJ/bit = 840 μW/channel which exceeds the total implant power budget. As such, some form of data compression is necessary. There are two main methods for compressing data: lossless and lossy compression.

1.3.1 Lossless Compression

Lossless compression takes advantage of skewed and narrow histograms in the data to assign shorter codewords to more likely data values, and longer codewords to less likely data values [25]. On average, this reduces the length of the communicated data. Many different lossless algorithms have been proposed. These include Huffman encoding, Arithmetic encoding, Lempel-Ziv, etc.

Huffman compression is especially interesting for WI-BMIs. Firstly, this is because Huffman encoders can be static. Static encoders are pre-trained on representative data and do not adapt to the data they are compressing. As such, if the histogram of the to-be-compressed

Table 2: Huffman encoder example

Huffman Encoder		
Symbol	**Frequency**	**Codeword**
0	0.8	0
1	0.1	10
2	0.07	110
3	0.03	111

data can be accurately estimated *a priori*, a static encoder can be used. Relative to an adaptive encoder, this dramatically reduces the required operations and hardware resources because a static Huffman encoder can be implemented using only a few Look-Up Tables (LUTs). An example of Huffman encoder LUTs given in Table 2, where compression is achieved by giving shorter codewords to more likely symbols. Given the codeword lengths and frequencies, the average encoded length of a symbol will be $0.9 \times 1 + 0.1 \times 2 + 0.07 \times 3 + 0.03 \times 3 = 1.4$ bits, instead of the 2 bits that are normal for the binary representation of 4 values (e.g. codewords of 00, 01, 10, 11).

Secondly, each symbol is represented by a unique, fixed codeword. In particular, Huffman coding is a prefix coding. This means that no codeword is the prefix of another codeword. An advantage of this is that multiplexed channels can use different encoders. As long as the sequence of channel-encoder pairs is correctly known by the decoder, the sequence can be fully decoded. Considering that multi-channel neural recordings are often multiplexed and can have differently shaped histograms, this can be of great benefit. For example, each channel could use its own encoder. Alternatively, a handful of static Huffman encoders could be placed on-implant. The channels could then be assigned to different encoders based on which offered maximum compression.

As such, Huffman encoders are very simple encoders which offer a lot of flexibility, and are generally well-suited to the WI-BMI environment. An example scenario with two multiplexed Huffman encoders serving different channels is given in the Supplemental Material, Table 1.

1.3.2 Lossy Compression The second form of compression is lossy compression, i.e. feature extraction. It consists of compressing data by removing or degrading information that is assumed to not be of interest. For example, downsampling data is a form of lossy compression. Similarly, reducing the sampling resolution is a form of lossy compression. Dimensionality reduction, where one transforms the data so as to concentrate the useful information in a smaller number of channels/inputs and thereafter eliminate redundant observed inputs, is also a form of lossy compression, e.g. Principal Component Analysis thresholding.

In the case of broadband data, many informative lossy features have been determined. This is because the broadband signal can be split into two major components, the Local Field Potential (LFP) and the Extracellular Action Potential (EAP), both of which are sparse and

shown in Fig. 2. LFPs consist of the lowpassed broadband at 100-300 Hz, and are believed to result from the sum of extracellular currents and spike activity in the vicinity of the electrode [26, 27]. While low bandwidth, easy to measure and chronically available, it has not been shown to have as good decoding performance as higher-frequency features such as those derived from the EAP [28]. In [29], it was found that the LFP signal could be efficiently losslessly compressed using static Huffman encoders, especially if delta-sampling was used.

The EAP consists of the ∼300 Hz highpassed broadband. It is highly sparse, given the infrequency of neural spiking events which are believed to contain most of the interesting information in neural signals. Most of what makes up the EAP signal is spikes of varying amplitude combined with thermal and electronics noise [20, 30, 31]. To remove the uninteresting noise, many different EAP compressions have become common, e.g. Single Unit Activity (SUA), Multi-Unit Activity (MUA) and Entire Spiking Activity (ESA). These are shown in Fig. 2.

SUA consists of measuring the spike firing times of individual neurons in the vicinity of the electrode. It is obtained by identifying neurons firing in the EAP, detectable via a sudden spike in the signal. This is often done via setting a threshold in combination with a nonlinear energy operator, and if the threshold is exceeded then a spike is considered to have occurred. The shape of the spike is then clustered, where similarly shaped spikes are assumed to originate from the same putative neuron. SUA gives high decoding performance, however the spike sorting procedure is expensive to perform on-implant [32, 33]. As such, there has been work to extract and compress the spike shapes to send them off-implant for sorting [34, 35].

MUA is similar to SUA, however the spikes are not sorted, and one treats all spike events on the same electrode as originating from the same putative neuron. Although evidence is somewhat mixed, it is generally believed that MUA gives very similar decoding performance to SUA [28, 36–38]. By reducing the broadband to only the spikes in either SUA or MUA, significant *de facto* compression is achieved, along with power savings [30, 32].

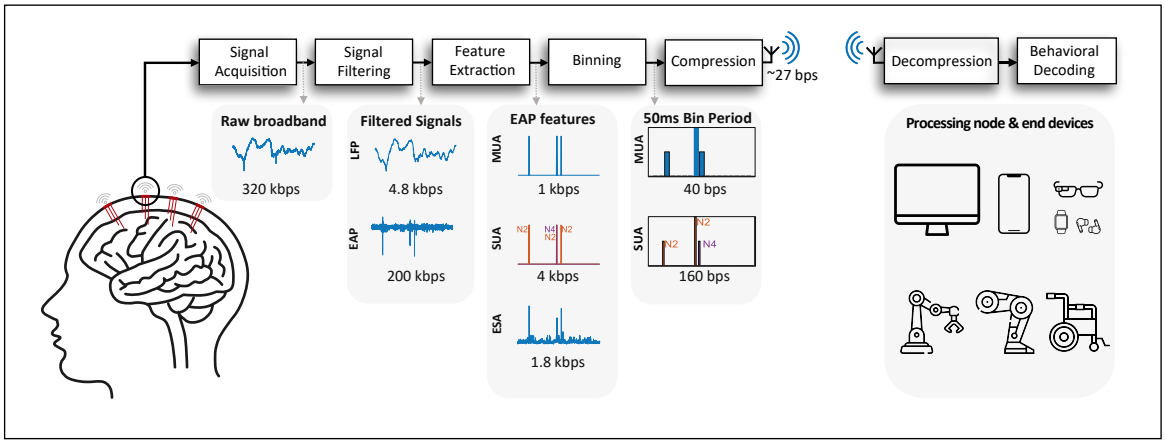


Figure 2: Typical BMI data processing and compression flow, with common extracted features / lossy compressions of intracortical broadband data. The numerical values beneath the signals give approximate BRs per channel for that signal.

ESA, sometimes also confusingly referred to as MUA, consists of rectifying the EAP and then lowpass filtering it at ∼50 Hz. This gives an envelope of unsorted spiking activity, and has been found to offer high decoding performance at low bandwidth [17, 28, 39, 40]. While SUA and MUA involve the detection of binary events, ESA is more analogue. It is underexplored as a signal, but sampling rates as low as ∼1000 Hz (with a Nyquist rate of 24 Hz) and sampling resolutions of 16 bits/sample offered exceptional decoding performance [28]. It may be that sampling rates as low as 20 Hz and resolutions as low as 5 bits/sample may offer good decoding performance, but this is speculation. If so, then the ESA signal could offer good decoding at even 100 bits/s/channel. Non-linear quantisation, and lossless compression oF ESA such as in [29, 41], may offer further reductions in BR.

While the ESA signal is of significant interest, and will be the subject of future work by the authors, most modern WI-BMI systems target the MUA signal [30, 38, 42, 43]. This is likely due to its well understood and sparse nature, ease of measurement, and high BDP with a standard BR of at most 1000 bits/s/channel [18, 30, 44, 45]. Recent work from the authors has also found that the MUA signal can also be reliably extracted using very few hardware resources and a very small power budget [46]. However, even such a BR is very limiting for mm or sub-mm scale implants. For the considered power budget $B = 625\mu$W, with a BR of 1000 bits/s/channel and 20nJ/bit, and with static FPGA power consumption of 162μW and a processing power for the 1 ms binner of 0.96 μW/channel, discussed later in Section 3.2.2, this translates into a maximum of 22 channels. It is assumed that the front-end amplifier and ADC has a separate footprint and power demand. While 22 channels may be acceptable for a 2.5 mm $\times$ 2.5 mm sized implant plus the front-end footprint, it would be useful to know if further data compression could enable increased channel counts or reduced implant size, and what the tradeoffs are.

1.4 Multi-Unit Activity

More specifically, MUA consists of measuring how many spikes occur in a time bin, of pre-determined duration, on each recording channel. Multi-channel MUA is generally represented by multiplexing the MUA signal from different channels together [30, 44]. E.g.

$$2,\ 3,\ 1$$

indicates that 2 events occurred on channel 1, 3 occurred on channel 2, and 1 occurred on channel 3 in the length of the given MUA time bin, i.e. the Binning Period (BP).

The BP commonly varies from 1-100 ms [18, 30, 42, 44, 45]. The practical lower limit of 1 ms is because spikes last approximately 2 ms. However, the effects of increasing BP beyond 1 ms is an area of active research [30, 47, 48].

The size of each multiplexed MUA communicated data block is $n \times m$ per BP. n is the number of MUA channels. m is the number of bits required to represent, in binary, the number of MUA events in a channel in the given BP. m is generally chosen *a priori* by researchers, based on their estimation of how many MUA events may occur in a bin.

By increasing BP, the MUA data is lossily compressed, saving on communication bandwidth [30]. This is because the BR required to communicate a channel of binned data is

equal to m/BP (bits/s/channel). There is an approximately linear relationship between the likely maximum number of MUA events in a bin and BP (Fig. 3 (a)). This produces a positive, approximately logarithmic effect on m from increasing BP (Fig. 3 (b). As such, merely increasing BP decreases the communication bandwidth, relative to a lower BP (Fig. 3 (c)). The cost is reduced temporal resolution of MUA firing times.

1.5 Prior Work in the Compression of MUA

The lossy compression of MUA was investigated in [47]. Specifically, the effect of varying MUA BP on BMI Behavioral Decoding Performance (BDP) was investigated. It was found that increasing BP had a slight but statistically significant negative effect on BDP: an absolute reduction of 0.85% per 10 ms BP increase in the mean Pearson correlation coefficient between the observed and predicted X and Y axes of a free hand movement, decoded with Long-Short Term Memory decoders. This is to be expected, as a lower temporal resolution intuitively translates into reduced BDP. However, the effect of BP on BDP likely varies by decoded action and decoding algorithm [30], and no effect of BP on BDP for SUA was found in [48].

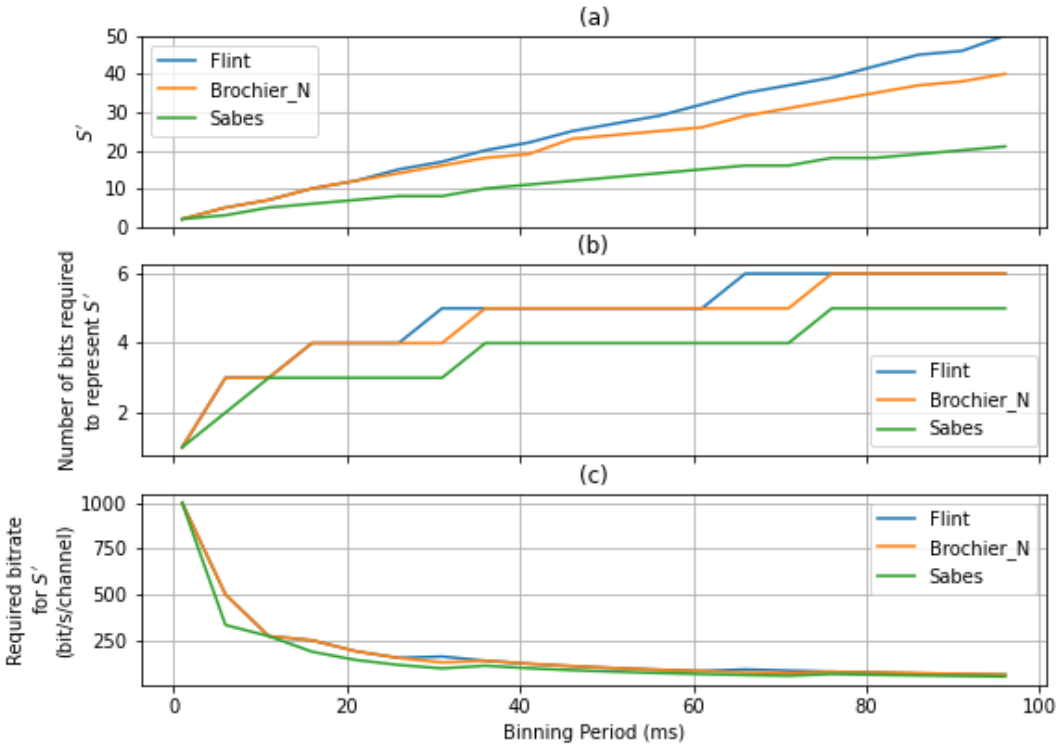


Figure 3: (a) A plot of S', the maximum range of MUA events measured in a bin, as a function of BP; $S' = max(x) + 1$, where the '+1' is because we need to encode the case of 0 MUA events occurring in a bin. (b) The number of bits $ceiling(log_2(S'))$ required to losslessly represent up to S' MUA events in a bin, as a function of BP. (c) The communication bitrate, in bits/s/channel, required to communicate S' events per bin, $ceiling(log_2(S'))/BP$. (a-c) The analysed MUA data, from which S' is measured, consists of the first 10 recordings in the Flint dataset [18], i.e. the recordings in files 'Flint_2012_e1', 'Flint_2012_e2', 'Flint_2012_e3' and 'Flint_2012_e4'. The recordings from the Brochier dataset [49], Subject N are also included. Finally, the recordings from Sabes lab [19], Subject 'Ìndy' are also used.

In [29], the lossless compression of various intracortical neural signals with pre-trained static Huffman encoders was investigated. These included the EAP, ESA, and the LFP, all at various sampling resolutions.

1.6 This Work

To the best of the authors' knowledge, a holistic analysis of the compression of MUA signals has not been undertaken. As such, in this work we investigated the compression performance of static Huffman encoders on MUA data at different BPs. This involved binning and then using multiple pre-trained, static Huffman encoders to losslessly compress the binned data. Channels were assigned to the static encoders based on an estimate of which encoders would give the best compression performance, given an on-implant estimation of the channels' sample histograms. This was investigated as a potentially computationally efficient but well-performing alternative to having either a single static encoder for all channels or adaptive encoders for each channel. Additionally, channels' firing rates can change over time, making the semi-adaptivity valuable.

A basic schematic of the data flow is shown in Fig. 4. The analysis looked at different BPs, the number of encoders and the number of on-implant resource budgets for the sample histograms. After determining the effect of each parameter on BR, the systems were then implemented in FPGA and the hardware requirements and processing power consumption measured. From the communication energy per bit estimates in Section 1.2 and the observed BR, the communication power was obtained. The processing power and communication power were combined into an estimate of the total system power. The BDP of the data was then investigated using Wiener Cascaded Filters. As such, a holistic view of BDP, temporal resolution, hardware resources and on-implant power consumption is given for the different strategies.

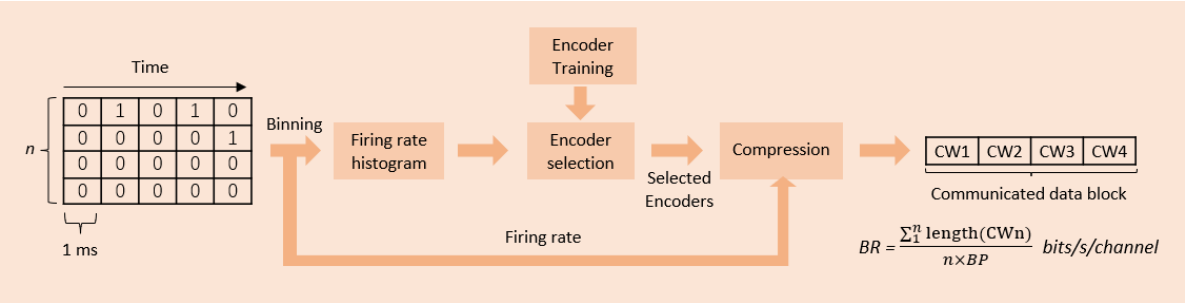


Figure 4: Data flow with binning and Huffman compression using multiple pre-trained encoders with assignment of channels to encoders. If only 1 encoder is used, no assignment of channels to encoders is necessary. CWx corresponds to the codeword given to the x^{th} channel by its assigned encoder.

2 Materials and Methods

The public datasets were loaded with Python 3.8 and MATLAB 2020a, the analysis was performed in Python 3.8, and the FPGA design in Modelsim Lattice Edition and iCEcude2

Table 3: Dataset summaries.

Dataset	Neural data type	Species, electrode type and brain region	Details	Behaviour
Flint [18]	SUA	Rhesus macaque monkey Utah array M1	One subject 12 recordings across 5 days 96 channels Recording lengths (quartiles, s): 597, 604, 630	Free-reaching hand task Continuous data stored
Sabes [19]	SUA	Rhesus macaque monkeys Utah array M1 and S1	Subjects Indy and Loco 37 recordings for Indy across 10 months 10 recordings for Loco across a month 96-192 channels Recording lengths (quartiles, s): Indy: 472, 524, 816 Loco: 1771, 1928, 2384	Free-reaching hand task Continuous data stored
Brochier [49]	SUA	Rhesus macaque monkeys Utah array M1 and PMv/PMd	Subjects N and L Single session recordings 96 channels Recording lengths (s): N: 1003 L: 709	Hand reaching task Target stored

2020. The analysis Python code and FPGA Verilog code and designs have all been made publicly available at [50]. The formatted data and results have been available at [51].

2.1 Analysed Datasets

To get a broad sample of MUA conditions, three publicly available datasets were used. These are summarised in Table. 3, and further details are given in Supplemental Material, Section 2.

In this work, the X and Y-axis cursor velocities in the hand reaching tasks were used as the observed behavioral data. The BDP is defined in this work as the across-axes average Pearson correlation coefficient r between the predicted and observed X and Y-axis velocities. In the Brochier et al. dataset, the behavioural data consisted of labelled actions. As these were not continuous measurements, the behavioral data from the Brochier et al. dataset was not analysed in this work so as to keep the BDP metric consistent.

The Flint dataset was split so that the first 4 days of recording sessions were included in the training data, referred to as set A. This corresponded to 10 out 12 recording sessions. The remaining 2, taking place over another day, were used as testing data and included in set B. The Brochier dataset was all included in the testing data B. Finally, the Sabes dataset was split so that data from subject Indy was included in A, and the data from subject Loco was included in B.

For each dataset, the SUA data was intra-channel collated to MUA, then binned to

the desired BP. The behavioral data was resampled to the same BP resolution using linear interpolation.

2.2 Determining the Impact of Compression on Behavioural Decoding Performance

The maximum number of MUA events in A was empirically measured as $S'-1$ (Fig. 3 (a)). S' is the resulting encoder length, i.e. the number of potential inputs to the Huffman encoder, as the case of 0 MUA events also needs to be represented. However, communicating up to S' possibilities per bin may be unnecessary. Saturating the range of measurable MUA events per bin at a lower value would save on hardware resources and reduce the bandwidth. This is because more possible symbols requires more, and longer, codewords and histogram bins. In effect, by saturating the data we are limiting the dynamic range. However, this is a form of lossy compression and may therefore reduce BDP. Therefore, using the A training data, it was tested whether saturating the range of measurable MUA events per bin at an integer value S, where $2 \leq S \leq S'$, had any negative consequences on BDP. In other words, every value in the signal above $S-1$ was set to $S-1$ and the resulting BDP observed.

A Wiener Cascaded Filter (WCF) was used for the decoder, with the binned MUA data as input and the X and Y-axis cursor velocities as the decoded output. WCFs have been found to have good decoding neural performance relative to other simple decoders, although they have generally found to not be as effective as deep learning methods [28, 48, 52]. However, their training times are significantly shorter [52]. As such we used them to investigate the relationship between S, BP and BDP. In this work, the WCF code from [28] was used. 5-fold cross-validation (hyper-)parameter optimisation was performed, and the details given in the Supplemental Material Section 3. Once the parameters were optimised for each S and BP, the BDP was calculated for each combination using separate testing data.

2.3 Impact of Parameters on Communication Power

In this section, we determined the effect of various compression architectures on BR and therefore communication power. The role of the number of encoders, S, BP, and histogram memory size, which is used in assignment of channels to encoders, were investigated. However, firstly, not all static Huffman encoders will be useful for compressing MUA data at a given BP. As such, to identify the subset of interesting Huffman encoders, a Machine-Learning (ML) strategy was adopted. This is because the subset of interesting Huffman encoders, to be implemented on-implant, should be found in a way that is compatible with real-life application. I.e., the choice of the subset of Huffman encoders implemented on-implant should not be influenced by the data to be measured on-implant, as this is not known prior to implantation.

An approach was taken where all possible static Huffman encoders of length S were considered. During each training-validation round, the set of Huffman encoders was reduced by removing the encoder that contributed the least to the compression. Eventually, only one encoder, i.e. the 'best encoder', was left, in which case no assignment or histogram

Table 4: Demonstrating a non-redundant representation of Huffman encoder codeword lengths, the SCLV. (a) All Huffman code combinations for S = 3; (b) Vectors of the lengths of the Huffman codes, with copies removed. (c) Sorted Codeword Length Vectors, with copies removed.

(a)

All Huffman Encoders of Length S = 3			
Huffman code combination	**1st CW***	**2nd CW**	**3rd CW**
1	0	10	11
2	0	11	10
3	00	1	01
4	01	00	1
5	11	0	10
6	00	01	1

***CW = Codeword**

(b)

Non-redundant Codeword Length Vectors, S = 3			
Huffman code length combination, key	**1st CW length**	**2nd CW length**	**3rd CW length**
1	1	2	2
2	2	1	2
3	2	2	1

(c)

Non-redundant Sorted Codeword Length Vector (SCLV)			
SCLV combination	**1st CW length**	**2nd CW length**	**3rd CW length**
1	1	2	2

were necessary and every channel used the same encoder. This is discussed further in Section 2.3.2.1.

As required for a ML approach, a training/validation split was used. The training data was used to reduce the set of Huffman encoders. The validation data was used to represent neural data recorded on-implant, on which the quality of the assignment and compression was measured.

The training and validation data, from A, were split by taking the full recording from a channel and assigning it to either training or validation. The training and validation channels were randomly selected with a 50/50 split. The channels from the Flint and Sabes datasets were split separately, so that both were represented 50/50 in training and validation. All 960 Flint channels in A were considered, and a random 2000 out of 4224 channels in the Sabes data in A were considered. The Sabes data was limited so that the contribution of the Flint data was not overshadowed, and so prevented overfit to the Sabes data. It warrants mentioning that this training-validation split is distinct from that in Section 2.2, that looked at the effect of BP and S on BDP, although both splits concerned data from A.

2.3.1 SCLV representation We wanted to determine how the best k ($k \in Z, 1 \leq k \leq h$) Huffman encoders, from the set of all possible h Huffman encoders with S fields, would perform in compressing MUA data from multiple channels where each channel was assigned to its ideal encoder. As such, all possible Huffman encoders of length S were produced with arbitrary keys. For example, for $S = 3$, all possible Huffman encoder codeword combination are given in Table 4 (a).

There are many redundant configurations of Huffman encoders, assuming the order of

Table 5: Taking the dot product between the SCLVs (a) and example Channel Sorted Histograms (b) to obtain the size of the compressed data blocks (c). From (c), we can determine the best SCLV-channel pairs. In (d), the channel-average length $avlen$ in bits of the encoded symbols after each channel is compressed using its ideal encoder is obtained, with $L = 1000$.

(a)

SCLVs					
SCLV \Symbol	**0**	**1**	**2**	**3**	**4**
1	1	2	3	4	4
2	2	2	2	3	3
3	1	3	3	3	3

(b)

Channel Sorted Histograms (SH)				
Symbol \Chan.	**1**	**2**	**3**	**4**
0	650	500	300	950
1	200	200	300	5
2	100	120	200	31
3	50	100	100	7
4	0	80	100	7

(c)

Dot Product of Channel SHs and SCLVs				
SCLV \Chan.	**1**	**2**	**3**	**4**
1	**1550**	**1980**	2300	1109
2	2050	2180	**2200**	2014
3	1700	2000	2400	**1100**
Best SCLV	**1**	**1**	**2**	**3**

(d)

$$avlen = \frac{1}{n}\Sigma_{i=1}^{n}\frac{min(dotprod[:,i])}{L}$$

$$avlen = \frac{1}{4}\frac{1550 + 1980 + 2200 + 1100}{1000} = 1.7075 \text{ bits}$$

the symbols in unimportant. The only constraint is that, in a Huffman encoder, the shortest codeword should represent the most likely symbol in the data, the 2nd shortest should match the 2nd most likely, etc. As such, the Huffman encoders were reduced down to a Sorted Codeword Length Vector (SCLV) representation, which consisted of the ascending sorted vector of codeword lengths. This process is shown in Table 4 (a-c). In the case of $S = 3$, it can also be observed in Table 4 (c) that all $h = 6$ encoders reduce down to a single SCLV. Taking the dot product of the SCLV and Sorted Histogram (SH), where the histogram is sorted in descending order, gives the length of the data, in bits, after compression by a Huffman encoder with the same SCLV as shown in Table 5 (c). Dividing the dot product by the number of samples gives $avlen$ from Equation 2, where dividing further by BP gives the BR. This enables us to assign encoders to channels based on which encoder gives the smallest BR (Table 5 (d)). It warrants mentioning that, to actually compress the data, a Huffman encoder is required. However the BR can be obtained from the SCLV via its dot product with the SH.

Based on the BDP vs. S results that will be shown in Section 3.1, and our knowledge of the hardware costs of large S values that will be shown in Section 3.2, we opted to examine S values between 2 and 9 for our set of BPs. This allowed relatively good BDP values, as discussed later, while minimising the resources. Therefore, for each integer value of S $\{S \in \mathbb{Z},\ 2 \leq S \leq 9\}$, the full set of SCLVs was produced. The details are given in the Supplemental Material, Section 4.

2.3.2 ML process

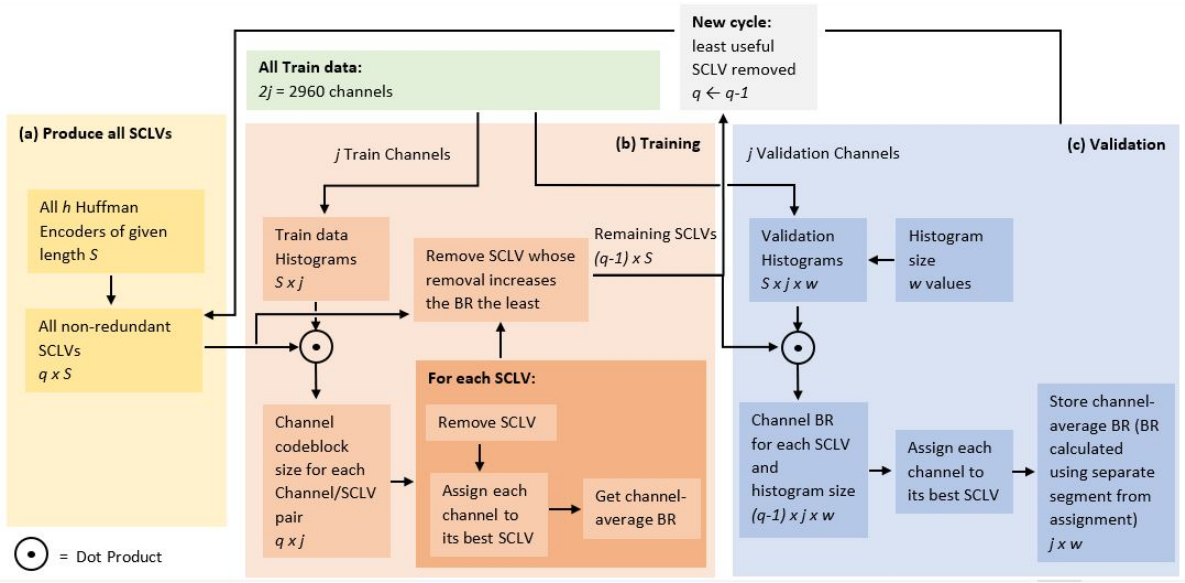


Figure 5: Training-validation process. Italics represent data dimensions. First, all possible SCLVs are produced (as shown in (a)). Then, each round selects an increasingly smaller subset of SCLVs based on which give the best compression in the training set (as represented in (b)). At each round, i.e. with different numbers of SCLVs, the compression is tested on the validation dataset for different on-implant histogram sizes (as represented in (c)).

2.3.2.1 Training

The system was trained by assigning the SCLVs to the training channels based on which SCLV gave the best BR for each channel. Multiple channels could be assigned the same SCLV, and all of the data in each channel was used.

To identify interesting SCLVs, i.e. train the system, the following strategy was used. Firstly, each training channel's SH was obtained. Secondly, one by one, with replacement, each SCLV was removed. The average BR across channels was then measured after each channel was assigned to its best remaining SCLV, e.g. as in Table 5. Channels could be assigned to any SCLV, except the missing one. After the total BR had been obtained for each missing SCLV, the SCLV that was found to be the least effective SCLV for compressing the training data, in concert with the other SCLVs, was removed. As such, at the end of the training round, one SCLV had been removed. This process is represented in Fig. 5 (b).

2.3.2.2 Validation

Between each training round, validation was performed. The purpose of validation was to measure the BR when each 'on-implant' channel was assigned to its ideal encoder from the selection of encoders available during the current training round. In other words, it was tested what the on-implant BR would be given the selection of encoders found during that training round. The method of assigning on-implant MUA channels to encoders should be realistic to implement in hardware. In this work, the assignment was done by using a segment from the beginning of each on-implant channel recording to produce a sample

histogram. Each channel was then assigned to an encoder based on which encoder gave its histogram the smallest BR. This is represented in Fig. 5 (c).

How much of the beginning of the 'on-implant' validation recording was used to calculate the sample histogram depended on the size of the histogram. The considered histogram sizes were $S \times 2^d - 1$ samples, where $d \in [2, 3, 4, ..., 9, 10]$. Each histogram bin, of which there were S, was given a size of $2^d - 1$ maximum samples, i.e. of d bits. Once $2^d - 1$ samples had been measured across all bins, the histogram growth ended, the histogram was sorted, and the channel was assigned to an encoder.

After assignment, the resulting BR was calculated. This was done for each channel by taking the rest of the validation data, that which had not been used for assignment, and calculating its BR after compression. In particular, only a segment of the remaining data was used, where the segment was always equal in length to half of the total recording length. This was so that the amount of compressed data was the same across all histogram sizes. This is shown in Fig. 6, where the beginning of the recording is used for assignment, and a segment of the rest for calculating the BR. The average BR across validation channels was then stored for each histogram size.

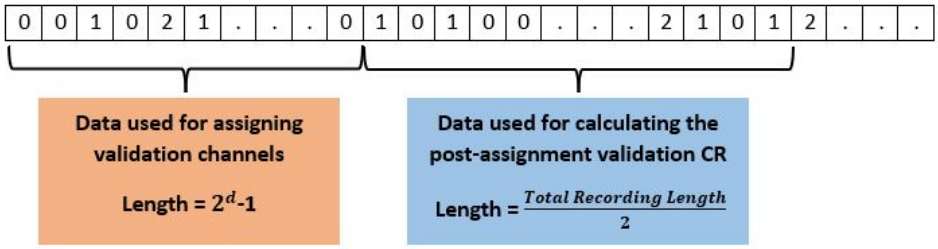


Figure 6: Single-channel split of assignment (for sample histogram) and to-be-compressed validation data, based on histogram memory size d and total recording length.

We then moved to the next training-validation round. At each round, the least useful SCLV was removed based on the training data. The validation channels were then assigned to the remaining SCLVs, and the average BR for each histogram memory size stored. The process continued until only 1 encoder (i.e. the best encoder for the training data) was left, where all validation channels were automatically assigned to the last encoder. As such, for each round (i.e. number of non-redundant SCLVs), the mean BR, ideal set of encoders, and effect of histogram size on BR were all determined. 30 different cross-validation iterations of this process were run in parallel, with different training-validation channel splits, and the results were combined.

It is important to mention that, for any channel, the compressed on-implant data should be sorted the same way its histogram was sorted. For example, if a firing rate of 3 MUA events per bin is found to be the most common firing rate in the sample histogram, the occurrence of 3 MUA events in a bin in the remaining to-be-compressed data should be given the shortest codeword during compression. This is because the sorted ordering in the compressed data has to be determined somehow, and so is approximated using the sample histogram for each channel. As such, the histogram serves to not only get an idea of the

Table 6: Required modules for each system configuration. The histogram is used for assignment of encoders to channels, and can be the subject of both assignment sorting and mapping, which is sorting but applied to the to-be-compressed data. The system configurations vary if assignment is required or not, e.g. if only 1 encoder is considered, or if the histogram is to be sorted or not. The configuration also varies if no Huffman compression is considered, in which case only a binner is required.

	With Huffman encoding				**Without Huffman encoding / Only Binning**	
	With Sort Logic		**Without Sort Logic**			
	#enc >1	**#enc = 1**	**#enc >1**	**#enc=1**		
Binner	True	True	True	True	**Binner**	True
Histogram	True	True	True	False	**Histogram**	False
Selector	True	False	True	False	**Selector**	False
Encoder(s)	True	True	True	True	**Encoder(s)**	False
Sorting and Mapping	True	True	False	False	**Sorting and Mapping**	False

shape of the channel's histogram, used to find the ideal encoder for compression, but also to find the sorting order of the to-be-compressed data, as the most common firing rates should be assigned the shortest codewords within each encoder. We refer to this assignment of to-be-compressed firing rates to encoder fields, based on the sample histogram, as mapping (e.g. mapping the symbols to the encoder fields). As such, the size of the histogram affects the quality of both the assignment and the mapping, both which impact the BR. The sorting and mapping implementation is discussed further in Section 2.4.

2.3.3 Different system configurations Finally, it was tested whether certain modules could be removed, saving on power and resources. Firstly, the effect of removing the sorting and mapping modules was investigated. This is because the histogram of MUA events in a BP $\leq$ 100 ms tends to follow a decaying exponential, where having fewer events is more common than having more. As such, sorting can be redundant, and in the case of small histogram sizes, counterproductive. This is because if the sampled histogram is unrepresentative of the rest of the compressed data, the n^{th} most common value in the to-be-compressed data may not be assigned to n^{th} shortest codeword. As such, if the data generally follows a decaying exponential, the compression would be undermined by the sorting with an inaccurate sample histogram, where the mapping would have been correct if not for our sorting and mapping. Sorting also requires extra FPGA resources and processing power.

Secondly, it was investigated whether only using binning, without any Huffman encoding, could give effective compression. Each symbol is given a codeword of the same length , i.e. ceiling($\log_2(S)$. No lossless compression is used in this configuration, giving a useful benchmark as to the effect of Huffman compression in our system. The modules included in each system configuration are given in Table 6.

2.4 FPGA realisation

We simulated the presented architectures on an FPGA target, Lattice ice40LP, in order to assess the overhead on power and resources brought by compression so as to guide our configuration selection. Such a low-power, high-performance FPGA with 40nm technology and small BGA package is ideal for the thinnest devices like implantable BMIs. All programs are written in Verilog, simulated on Mentor Modelsim Lattice Edition and synthesised with iCECube 2020.12.

Lattice ice40LP1K is an ultra-low-power FPGA board with 1280 logic cells and sixteen 4kbit memory blocks (bRAMs). The architecture of the FPGA implementation is shown in Fig. 7, including the Binner, Histogram counter, Sorter, Mapper, Encoder selector, Encoders and Memory. Referring to Table 6, different configurations can be achieved by bypassing

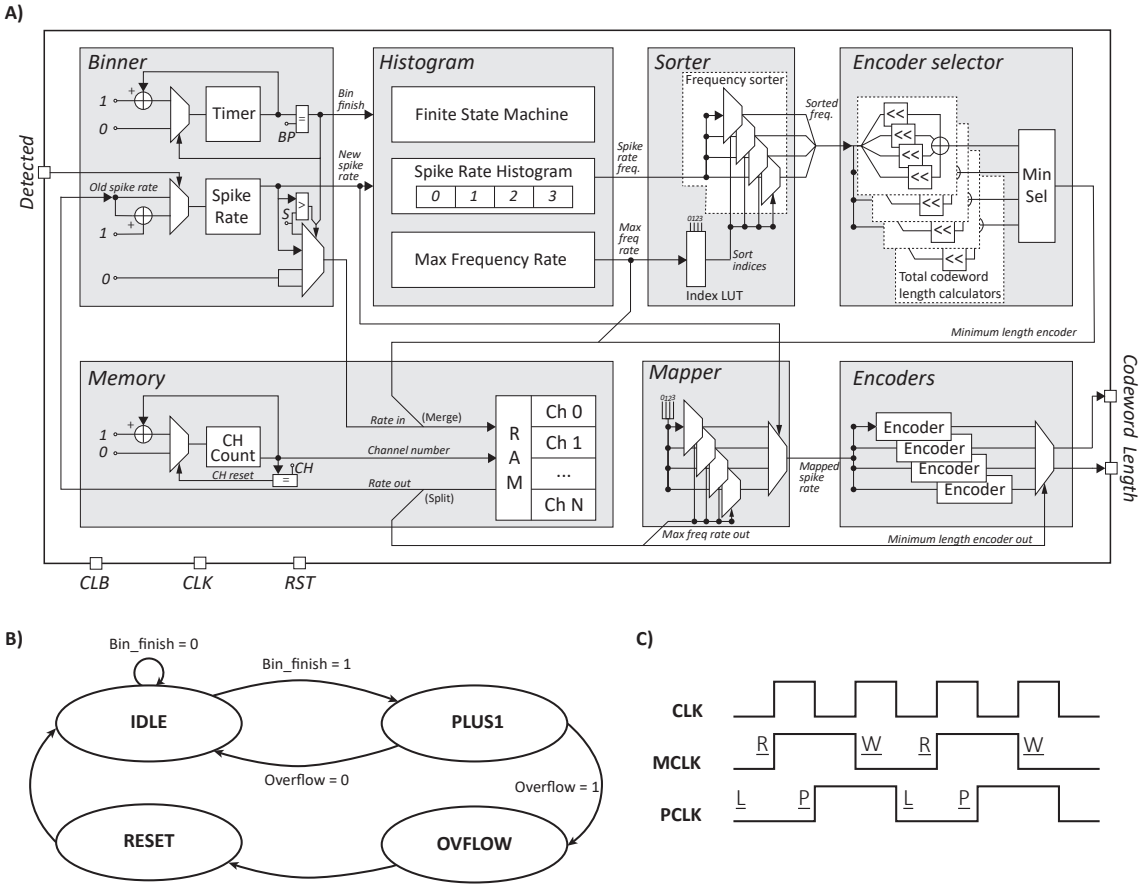


Figure 7: A). The FPGA implementation includes Binner, Histogram, Sorter, Mapper, Encoder selector, Encoders and Memory. For conciseness, several circuits are not shown: Clock generator for memory clock and processing clock, reset and re-calibration logic, clip for *New spike rate* and MUXs selecting *Minimum length encoder out*, *Sorted indices out* as the input to the RAM after calibration. B). State transition diagram of the finate state machine in Histogram module. C). Clock timing diagram of the system clock (CLK), Memory clock (MCLK) and Processing clock (PCLK). MCLK drives the RAM, it is read at the posedge (R) and written at the negedge (W). PCLK trigers the Binner and Histogram at its posedge (P). It also synchronise the detected signal at its negedge (L). The update of channel number is also happened at the negedge of PCLK (L).

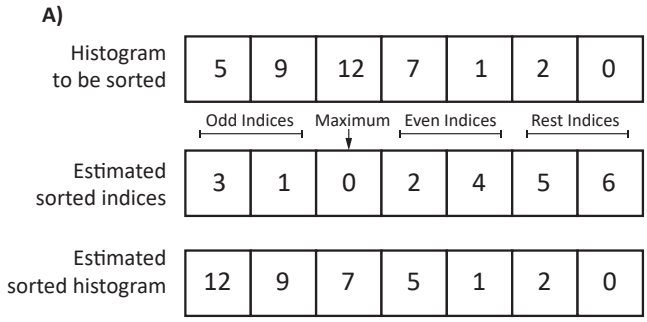
some of the components. In the 'one encoder' version, there is only one encoder in Encoders and therefore no need for the Encoder selector module. In the 'without sort' configuration, we assume the events in the histogram are already sorted and the sorter and mapper are bypassed. The *Spike rate freq* is connected with the *Sorted freq* in the encoder selector, and the *New spike rate* is connected with the *Mapped spike rate* in Encoders. Similarly, in the 'just binning' version, only the Binner module is implemented. A detailed breakdown of Fig. 7 is given below.

2.4.1 Binner A timer and a recurrent spike rate counter is used for the Binner, which counts the number of in-coming spike numbers in a given BP. The multiplexer (MUX) after the spike rate counter is used to clip the spike rate within the preset range and reset the spike rates stored in memory when a BP is over.

2.4.2 Histogram The Histogram is implemented using a Finite State Machine. The state transition diagram is shown in Fig. 7 (B). It accumulates the number of MUA events according to the *Bin finished* and new spike rate signals. When the histogram overflows, a finish signal is issued and the histogram is emptied for the next channel. As the maximum spike rate is clipped at $S-1$, the number of registers required for storing the different MUA frequencies is highly reduced. The index of the maximum value in the MUA histogram is recorded in the histogram counter, to be used in the Sorter.

2.4.3 Sorter and mapper Sorting the spike rate frequency in order can be resource-hungry or time-consuming in hardware. The resources used for implementing a sorting algorithm such as merge sort or quick sort can overwhelm the whole system. For sorting MUA histograms, we can take advantage of the fact that the MUA histogram, even if it does not follow a decaying exponential, is almost always expected to follow a unimodal peak distribution. As such, the Sorted Histogram (SH) can be easily estimated by setting the index 0 at the index of the histogram maximum, and the index number will increment by iterating on both sides of the histogram peak. For example, values to the left of the peak will take odd index numbers of 1, 3, 5, etc., while values to the right of the peak will take even numbers. When indices can no longer be assigned on one side, the rest are assigned serially to the other side. An example is illustrated in Fig. 8 A. Using such an estimation, we can reduce both the sorting space and time complexity to $O(n)$.

From a hardware perspective, this estimated sort algorithm can be implemented with a finite state machine (FSM). However, in Fig. 7, we show a combinatorial implementation. As the estimated sort order is only affected by the most frequent spike rate index, we can easily create a LUT that defines the sorting order based on the measured maximum index. The spike rate Mapper can also be implemented using an identical LUT. A demo of the sorted indices LUT when $S=5$ is given in Fig. 8. We also implemented two other sort algorithms: a swapping sort and estimation sort with sequential logic, which are given in Supplemental Material Section 5.



B)

Max/Index	0	1	2	3	4
0	0	1	2	3	4
1	1	0	2	3	4
2	3	1	0	2	4
3	4	3	1	0	2
4	4	3	2	1	0

Figure 8: A). A demo for the estimated sort algorithm. Index 0 of the sorted histogram will be the maximum of the unsorted histogram. Odd indices will be placed on the left while the even indices will be placed on the right. The rest of the indices will be placed serially on the unsorted side. B). A demo of the sorted indices when $S = 5$. Max indicates the index of the peak value in the histogram, and Index is the histogram field. Based on the peak value, the indices in the table are given to the histogram fields.

2.4.4 Encoder Selector The encoder selector assigns encoders to channels. For the given channel, it selects the encoder that gives the minimum encoded data length, determined by taking the dot product of the SH with the SCLVs. The SCLVs are stored on-implant for each encoder. Since multiplications are resource-hungry in hardware, we replace the multiplications with bit shifts. The number of bits to shift for different encoders and codewords are stored on board.

2.4.5 Encoders A Huffman encoder, in hardware, is a big LUT. Multiple encoders have been implemented on board. Different encoders are selected according to the encoder selector. The selected set of Huffman codewords and codeword length are then set as the outputs of this channel.

2.4.6 Memory unit The Memory unit consists firstly of a channel counter that counts the processed channels to schedule the data flow among channels. Secondly it consists of RAM that stores each channels' current number of MUA events, their sample histogram largest value's index during calibration (used for mapping during regular encoding operation) and their assigned encoders. Instead of using registers for each channel, utilising RAM increases the scalability, making it possible to upscale to thousands of channels within only hundreds of logic cells. However, the clock speed needs to be doubled to maintain the same data throughput. As the resources are highly constrained in lattice iCE40LP, the RAM implementation is preferred over using registers.

Fig. 7 (c) shows the timing of the clocks. CLK is the system clock and a clock generator is used to generate MCLK for memory and PCLK for different processing units. The detection signal will be loaded at the negedge of the PCLK. The RAM will provide the stored parameters for different processing units at MCLK posedge. All processing happens on PCLK posedge and the results are stored back at the MCLK negedge. These modules work together to compress the MUA data of each channel. The histogram counter, sorter and encoder selector are used at the start of implant operation for encoder selection as a calibration process.

During the calibration, for each channel, the sample histogram largest value's index and encoder assignment will be determined and stored into RAM. One can also periodically recalibrate each channel when the brain environment changes, or at some defined interval. After the calibration, the spike detection signal of different channels will flow through the binner, mapper, encoder and RAM interchangeably for compression.

3 Results

3.1 Impact of compression on Behavioural Decoding Performance

To determine the effect of S on each BP, as far as BDP is concerned, the procedure from Section 2.2 was carried out. Fig. 9 (a) shows the results averaged across the Flint and Sabes datasets, averaged because we wanted the results to be robust across different recording conditions. Examples of observed and predicted behavioral data are shown in Fig. 9 (b-d).

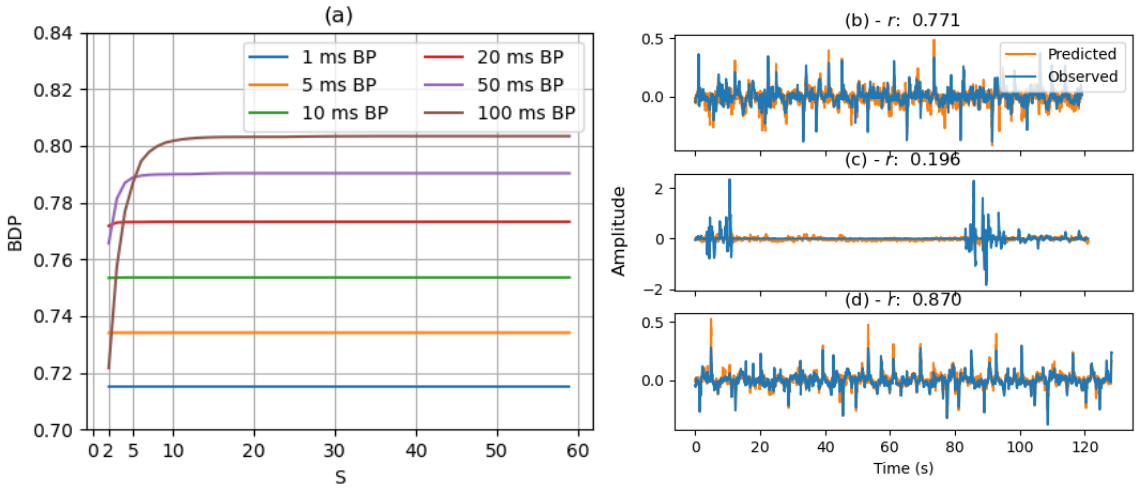


Figure 9: (a) Behavioral decoding performance (BDP) as a function of BP and S. Each S/BP combination was parameter optimised on 5-fold CV, with the results averaged from the Sabes lab and Flint datasets. (b-d) Example observed vs. predicted X-axis velocities from 5-fold CV, with corresponding BDP (r) for random Flint recording and parameter combinations during parameter optimisation at a BP of 5 ms.

We observe that BDP increases as a function of BP, which is somewhat contrary to other studies [30, 47]. However, it is also expected that results vary by decoded behavior and decoding algorithm [30]. As such, the results in Fig. 9 are not unexpected. It is also positive that BPs as high as 100 ms can achieve relatively high BDPs (e.g. ∼80%) using simple WCF decoders. Further discussion and more detailed results, e.g. boxplots, are given in the Supplemental Material, Section 6.

It warrants mentioning that BDTP = BP in our system. This is because the temporal resolution of the decoded behavior is equal to the binning period of the neural data, as all communication and other processing is relatively instantaneous.

Table 7: Logic cells were used in two different settings for three sort implementations.

	Swapping sort	Estimation sort - Seq	Estimation sort - Comb
S = 5, hist size = 4	480	239	75
S = 9, hist size = 6	682	355	260

3.2 FPGA resources and processing power

To assess how different configurations affect the processing resources and power consumption, we implemented all different settings of the compression algorithm with Verilog. The resource occupation can be obtained from the Placing Summary of ICEcube2, which is indicated as the number of LUTs and Flip-Flops (FF) used. To get the power consumption, it is not practical, given the massive parameter space, to download all the different configurations to the FPGA board and measure their power consumption individually. We used the iCEcube2 power estimator to estimate the processing power to reduce the assessment load. Although the estimation is not the real power consumption, it is accurate enough for estimating the effect of different configurations and guiding the selection process.

3.2.1 Resource usage Resources usage reflects the area occupation of the implementation. The full results are given in Supplemental Material 2 as an excel spreadsheet, and the same is available on the Github at [50]. The histogram and encoder selector are two resource-hungry modules. The amount of resources required by the histogram is mostly dependent on increased histogram size, but the S values also have a significant but smaller impact. The resource usage of the encoder selector can exceed that of the histogram when S is larger than 7 because of the dot product (implemented with bit shifting) between two vectors with length S. Alternatively, all possible multiplication results could be pre-calculated and stored in a RAM. The selector logic would be simplified, however this would sacrifice the processing speed as the results would need to be fetched from RAM one by one and summed together. As a result, the calibration time would be increased. Such an alternative approach could be useful if the limitation of the resources is extreme or the configuration requires a large S, histogram size or number of encoders. It would be less effective when these values are small. Aiming at finding the most compact compression scheme, we opted to use the bit-shift implementation discussed in Section 2.4.

For the sorter, we have compared three different approaches: Swapping sort, and estimation sort with sequential and combinatorial implementations. A summary of the resources needed for the three implementations is given in Table 7.

One can notice that compared to the Swapping sort, the sequential estimation sort reduced the required resources by half, which makes it possible to do on-implant sorting with limited resources available. More noticeable, when S = 5, histogram size = 4, the resources of the combinatorial implementation is only one-third of the sequential implementation, making the sorting no longer the bottleneck for resource usage. This advantage is lessened

when the settings are extreme. However, even when S = 9 and histogram size = 6, in which case the whole system can require too many logic cells to be implemented within our resources budget, the combinatorial one still uses fewer logic cells than the sequential version.

The remaining two modules, i.e. the Binner and Encoders, use few resources. These are normally below 100 LUTs+FFs each.

To guide the configuration selection, using only one encoder without sorting would be preferred because it can get rid of the histogram, sorter, mapper and encoder selector. If one has a histogram, increasing S would be preferred over increasing histogram size because increased histogram size has a large effect on both the selector and histogram counter, which are the two resource-dominating modules. These findings are purely from the resource perspective, the selection should also be guided jointly on the resultant total power and BDP.

3.2.2 Power consumption Power consumption is another aspect of concern. We should guarantee that the added processing power does not exceed the reduced communication power. However, the estimated power indicates that the processing power consumption per channel is consistent among different configurations. As the histogram counter, sorter and encoder selector are only used during calibration, the Binner, Mapper, RAM and Encoder continuously consume energy. The binner and RAM tick at the processing clock/memory clock speed, but the input of the encoders only changes at $\frac{1}{BP}$ Hz, which is much lower than the clock speed. Therefore the power of the Binner and RAM dominate the FPGA dynamic power, which is around 0.96 μW per channel. The power of the encoder is negligible at 1 to 20 nW, the binner consumes about 0.46 μW per channel and RAM shares the remaining 0.5 μW per channel. For the remainder of this work, the combined compression/processing and communication power is referred to as the dynamic power. The board static power is 162 μW.

As all configurations use the binner and RAM, the processing power of different architectures is similar whether we encode the firing rate or not. Therefore the total power reduction we gain from the compression is proportional to the BR reduction.

Bases on the exploration of resources and power, we can conclude that it is the resources that constrain the algorithm complexity for the on-implant Huffman encoding.

3.3 System configuration selection and considerations

In this work, we had 7 inputs into our system: S, BP, histogram size, number of encoders, the communication energy per bit, whether to sort the sample histograms or not in case of assignment, and whether to use a binning-only architecture without Huffman encoding. These influenced the 4 outputs: the (communication + processing) power, the required resources, the BDP and the temporal resolution of the decoded behavioral data (BDTP, equal to BP).

In Section 3.1, we obtained the BDP and BDTP as a function of S and BP, which are the

Table 8: Considered subset of analysed parameter space.

Subset of parameter space	
BP (ms)	1, 5, 10, 20, 50, 100
S	3, 5, 7, 9
Hist size (bits per bin)	2, 4, 6
#Enc	1 ,2, 3, 5, 7, 10, 15, 20

only impactors. In Section 2.3, we measured the communication power as a function of S, BP, number of encoders, histogram memory size, communication energy per bit, whether to sort the histogram and whether to use a binning-only architecture. In Section 3.2 we measured the processing power and resources as a function of the same system parameters. As such, we can now make a complete comparison of power, resources, BDP and BDTP as a function of the system parameters and transmission energy per bit. The integrated results are shown in Fig. 10, where each analysed combination of inputs is plotted in terms of its produced outputs. The hardware static power of 162 μW is not included, as it is independent of channel count. Due to the non-automatability of the hardware optimisation for different parameter combinations, only a subset of the parameter space was sampled, given in Table 8.

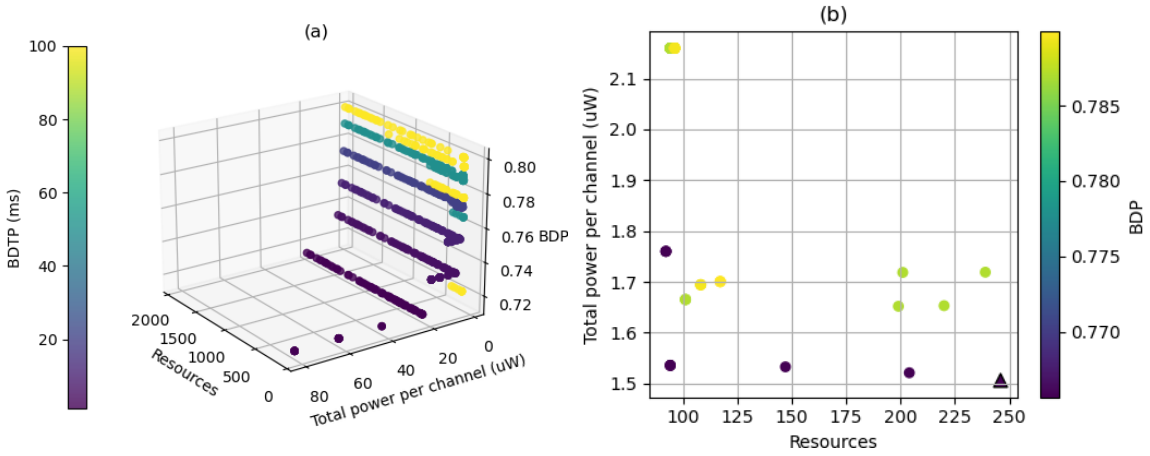


Figure 10: (a) Integrated results: BDP and BDTP values for different resources and power consumption levels. (b) Sample of integrated results, with BP/BDTP = 50 ms, resources $<$ 260, dynamic power $<$ 2.2 μW/chan for our 128 channel system. The outlined triangle represents the chosen system configuration for our tested system. Note that the color bars for (a) and (b) are distinct. Total power indicates total dynamic power per channel.

Ideally, one would consume few resources, little power, have a low BDTP and have a high BDP. However, it can be seen in Fig. 10 (a) that there is a negative relationship between the dynamic power and BDTP. This makes sense, as a lower BDTP means data is being sent out more often which translates into a higher data rate as given by Fig. 3, and therefore higher communication and dynamic power. As such, there is an unfortunate tradeoff between behavioral data temporal resolution and dynamic power consumption on-implant: a better temporal resolution means more power.

What is more positive is that the total consumed resources has close to no effect on either BDP, BDTP, or power consumption. Based on our results, this is for multiple reasons.

Firstly, having more of fewer encoders generally has little impact on BR, but has a large effect on the required resources. Often, having a single encoder is sufficient to dramatically reduce the BR. This is because the MUA histograms consistently follow a decaying exponential, especially if they have been sorted. Therefore, the same encoder that gave shorter codewords to smaller firing rates often gave the best compression from amongst all encoders. The addition of other encoders significantly increased the required resources while only marginally decreasing the BR. Secondly, increasing S significantly increases resources but only marginally increases BDP as seen in Fig. 9, and increases BR and so the dynamic power. Thirdly, it was observed that increasing the histogram size did generally decrease the BR. This makes sense, as a larger sample histogram means the sorting and mapping is likely to be more representative of the observed data, offering better assignment and mapping, both positively impacting the BR. However the effect was small, e.g. ≤ 1 bit/s/channel when going from 2 to 6 bits/bin. Generally, increasing the amount of resources (e.g. S, number of encoders, histogram size) had little noticeable effect on both the dynamic power and BDP, and could not have an impact on BDTP. As such, the considered low-resource architectures performed largely as well as the high-resource architectures.

However, there is an exception. There was a noticeable effect of having very few resources on the dynamic power. This is because various architectures were considered, i.e. the 'full', 'just-binning', and 'no-sort' architectures as in Table 6, the full results for which are given as an excel spreadsheet along with the analysis code at [50]. The 'just-binning' architecture used very few resources, but had a significantly higher communication power due to an increased BR, since no lossless compression was integrated into the system. The 'just-binning' architectures are noticeable as outliers with high power and low resources in Fig. 10 (a), where they are like the bottom segment of an L shape in the overall data.

Ultimately, to select our parameters, we chose to concentrate on the 'upper right' of Fig. 10 (a), in the zone of low resources and power, and high BDP. In [53], the human reaction speed for visual and auditory stimulus was tested in 120 healthy medical students between 18–20 years old. It was found that the mean reaction time was above 220 ms for both stimuli. As such, a BP of $\sim$50 ms BP is unlikely to cause any significant delay in user experience, as the communication and any additional data processing is practically instantaneous relative to human reaction time. This is supported by the fact that it is common for researchers to use a BP of 100 ms for decoding motor actions [18, 45, 48]. 50 ms BP also had the second highest BDP amongst all BPs (Fig. 9 (a)) and very low communication power. As such, we opted for a BP of 50 ms for our tested system. Fig. 10 (b) presents a subset of the results, with low resources, power, and a BP of 50 ms. After some consideration, for the testing of our system on the testing data B, we opted for the combination shown by the triangular marker in Fig. 10 (b). The results and system parameters are given in Table 9.

This configuration was principally because it scales well with channel count. The amount of required FPGA resources, excepting the RAM, does not increase much with channel count, as the data is multiplexed. The dynamic power of the implant increases

Table 9: Chosen system parameters and encoder for testing and associated training (A) and testing (B) results. For the training and testing results, all data in A and B were respectively used.

Chosen parameters	
Architecture	Full system
BP (ms)	50
S	3
#Enc	1
Histogram size	6 bits/bin

	Training Results (Flint, Sabes)	**Testing Results (Flint, Sabes, Brochier)**
Resources	246	246
BR (bits/s/chan)	26, 28	27, 28, 21
Dynamic Power (uW/chan.)	1.49, 1.52	1.49, 1.52, 1.37
BDP	78%, 78% (1% and 0% reductions from S = max(x))	72%, 62%, null (1% and 0% reductions from S = max(x))

Encoder (Symbol → Codeword)
0 → 0; 1 → 01; 2 → 11

roughly linearly, as more channels mean more communication power. Additionally, the processing power increases as the clock cycle needs to be increased. Finally, the BDP probably increases logarithmically, as the inclusion of more channels in decoding has been generally shown to do via neuron dropping curves [54, 55]. The BDTP is unaffected by channel count. As such, when increasing channel count, it is perhaps best to prioritise low power architectures, as power is the value that scales the least well with increased channel count. We therefore opted for the lowest power configuration for 50 ms BP, which is that given in Table 9 and indicated with the triangular marker in Fig. 10 (b).

It was verified, in both Python and Verilog, that inputting samples of the analysed neural data resulted in the desired encoded output and BR, with the correct selection of encoders, sorting, mapping, etc. As such, the system operated as expected.

3.4 Testing data results

Next, we looked at testing our chosen system on data it had not seen yet. Given our chosen system parameters, summarised in Table 9, we determined the BR and BDP on the testing data B, also shown in Table 9.

For the Flint, Sabes and Brochier data in B, the across-channel-and-recording average BRs were 26.5, 27.8, and 20.6 bits/s/channel respectively, corresponding to dynamic power/chan values of 1.49, 1.52 and 1.37 μW respectively. These are highly similar to those in the training data, where the averages for the Flint and Sabes data were 26.4 and 27.8 bits/s/chan and 1.49 and 1.52 μW respectively.

The BDP was measured for the Sabes and Flint datasets using the testing data in B. Each channel was split 90-10% into training and testing sets. S and the BP were fixed at 3 at 50 ms respectively, and as in Section 2.2 the WCF hyper-parameters and pre-processing parameters were 5-fold cross-validated on the training set. The best parameters for each BP/S combination were taken, and the BDP measured on the testing set. The average BDP for the Flint dataset was 0.724, and for the Sabes data it was 0.616. While these are relatively low,

it warrants mentioning that the best performance in the datasets for a BP of 50 ms were 0.731 and 0.619 respectively, which means that reducing S to 3 had an absolute negative impact of 0.07% and 0.03% respectively, which we deemed acceptable in exchange for significant power and resource reductions.

These BDP values are however lower than in the training data, seemingly because the recordings had less behavioral information in them. This seems apparent from comparison of Fig. 3 and 6 in the Supplemental Material. It is important to note that the results from the training data are not overfitted relative to the testing results: each BP/S combination was cross-validated and then tested on separate data within A and B individually. The A/B split was mainly to test the compression, where overfit would have been a real concern if only data in A had been considered. The main takeaway for the BDP results is that reducing S to 3 had no significant negative effect on BDP. It is expected that if the recording quality is similar to that in the training data, higher BDPs will result. It is also likely that using more advanced deep learning decoders would result in higher BDPs [28].

4 Discussion

As discussed in Section 1.3, wirelessly transmitting even a single channel of broadband is infeasible for a 2.5 mm $\times$ 2.5 mm scale FPGA implant. Communication broadband consumes a minimum of 42 kbit/s/channel. At a 20 nJ/bit communication energy, this consumes 840 μW/channel, exceeding the power budget of $B = 625\mu$W. Transmitting uncompressed MUA signals is better: 1 kbit/s/channel consumes 20 μW/channel. With a static FPGA power of 162 μW, negligible spike detection power [46] and a processing power for the 1 ms binner of 0.96 μW/channel, a maximum of 22 channels could be measured on-implant.

However, our system's power consumption depends on variable-length codewords. The system is non-adaptive for single-encoder configurations, and only semi-adaptive when multiple encoders are used. Therefore, there is a risk that the BRs will be higher than the expected $\sim$27 bits/s/channel as given in Table 9. For example, one might measure a handful of particularly active channels, and this may increase the BR. If one chooses the number of channels on-implant so as to be close to the permitted power budget, and the channels are more active than expected, this may produce more heat than desired. As such, it warrants choosing the number of channels based on a statistical understanding of the worst case scenarios.

As such, we took random samples of the channels. For sample size $z \in \mathbb{Z}^+$, we took 100,000 samplings of z random channels. For each random sampling Y, from the channels' summed BRs we obtained the resulting total system power P using the estimates

in Equation 3:

$$\begin{aligned} P_Y &= \sum_{i=1}^{z}(\mathrm{BR_i}) \times \text{Comm. Energy} + z \times \text{Processing Power} + \text{Static FPGA Power} \\ P_Y &= \sum_{i=1}^{z}(\mathrm{BR_i}) \times 20\,\mathrm{nJ/bit} + z \times 0.96\,\mu\mathrm{W/channel} + 162\,\mu\mathrm{W} \end{aligned} \tag{3}$$

where $\mathrm{BR_i}$ is the BR of the $\mathrm{i^{th}}$ channel in sampling Y, where $1 \leq i \leq z, i \in \mathbb{Z}^+$.

It was then determined, for each number of channels z, what percentage of random channel combinations exceeded the desired power budget B:

$$p(z) = \frac{1}{10^5}\sum_{Y=1}^{10^5}(P_Y > B) \tag{4}$$

where $(P_Y > B)$ is a boolean value equal to 1 if $P_Y > B$ and 0 otherwise. As such, $p(z)$ gave a permutation derived p-value for each number of channels z not exceeding the power budget.

Using our chosen architecture and power budget of B = 625μW and averaged across our 30 CV runs, it was found that from the training data results that having up to 304 channels never exceeded the power budget. Having 305 channels had a p-value of $\sim$5e-4 of not exceeding the power budget with $p(305) = 55/10^5$, and 306 channels or higher had significant chances of exceeding the power budget of $p(z > 305) > 0.05$. As such, we think having approximately 300 channels or fewer for our FPGA hardware is ideal assuming the given power estimates hold true, while staying within a conservative heating safety margin. As such, by compressing the MUA data one can send out over 13 times as many channels as when sending out the raw MUA data for a similarly sized FPGA device. In ASIC, this difference would likely be far more pronounced, given the reductions in dynamic and static power. However, the contribution of the front-end amplifier and ADC would need to be included as they would likely be integrated. Given that ADCs with power consumption as low as 0.87μW/channel have been achieved [56], there is reason to believe that impressive channel counts could be obtained at mm-scale in ASIC.

The benefit is derived from the simplicity of our tested system: a binner at 50 ms BP, a dynamic range limited to 3 possible values, and losslessly compressing the resulting signal with a pre-trained static Huffman encoder. Our results across 2 datasets and 3 subjects suggest that the lossy aspects of the compression do not present significant obstacles for behavioral decoding, and in the case of the BP may even improve the results over using smaller BPs, although low-BP data could of course be further binned off-implant prior to decoding. All results and hardware designs are made publicly available, and researchers are free to select from them for their own system designs.

It warrants mentioning that 100 ms BPs for behavioral decoding are common in the literature [18,30,45]. Therefore, a $>$ 50 ms BP system could also be of interest. Additionally, if increased BDP is an absolute priority, then a configuration with a higher S may be

appropriate. However, one should consider that if increased power is required as a result, this can reduce the amount of allowable channels for an implant of the same size, perhaps reducing the final BDP.

4.1 Robustness of Different Architectures

It warrants mentioning that, in theory, the 'no sort' configuration is more vulnerable than the others to unusual histograms, e.g. if there is consistent pathological activity. The 'just binning' architecture is the most robust, and the 'full' system is robust if the data used for assignment is well-representative of the compressed data, e.g. if the histogram size is sufficient.

To give some integrated context, the 'no sort' architecture is based on the assumption that zero is the most frequent MUA firing rate. Our sorting procedure's purpose is to shift the distribution to whatever the peak is, always assuming a unimodal peak. Each encoder represents a different sorted MUA distribution. The closer the distribution of the MUA data is to that of the encoder, the better the compression will be. For the multiple encoder version, there are multiple possible distributions to be selected from, making it more adaptive. The training process consists of pruning distributions in the encoders that are unlikely to occur in real MUA data.

4.2 Fixed Length vs. Variable Length Codewords and Bit-Flip Errors

It warrants mentioning that lossless compression works by giving variable length codewords to symbols. Due to the multiplexed encoding of MUA, this makes losslessly compressed MUA data more vulnerable to bit flip errors making the multiplexed communicated data block undecodable. As such, some noisy channel encoding or decreasing the BP may be necessary, assuming the bit flip error rate is sufficient to warrant it. This would increase the BR marginally, and is discussed further in the Supplemental Material, Section 7.

4.3 Future Work

Future work will consider compressing the entire MUA signal across channels, whereas this work looked at compressing intra-channel MUA. It may be that some dimensionality reduction is possible, or that correlations between adjacent channels can be taken advantage of as in [57] to further compress the data without reducing BDP or other metric of interest.

On-going work is looking at methods to compress the MUA in 'asynchronous' architectures, where the MUA firing rate is only communicated for a channel if it is larger than 0 for the given time period. Preliminary results show that, for BPs higher than or equal to 10 ms, the methods in this paper are superior. For BPs lower than 10 ms, asynchronous methods seems to perform best even for larger numbers of channels, e.g. 10,000.

5 Conclusion

In conclusion, our objective was to reduce MUA-based WI-BMI power consumption so as to prevent tissue heating and damage. To do so, we explored the use of ML-selected static Huffman encoders for the compression of MUA signals in a hardware and power efficient way. We developed a wide range of FPGA designs, made them publicly available, and tested our chosen configuration on 3 datasets where it consumed exceptionally little power and few resources. Main techniques involved the use of saturation of measurable MUA events at a maximum S value, varying the binning period, the use of hardware efficient techniques such as estimation sorting and bit shifting, and the coordination of multiple encoders via ML-selection. We find that MUA data can be reliably and significantly compressed, giving significant power-saving opportunities for wireless intracortical BMIs. For example, with a BP of 50 ms, $S = 3$ and the resulting bitrate of 27 bits/s/channel, a 1 Mbps wireless communication channel could support up to 37 thousand MUA channels.

Acknowledgement

O.W.S. is supported through an Physical Sciences Research Council (EPSRC) Doctoral Training Partnership (DTP) award (EP/N509486/1). P.F. was partly supported by the Engineering and EPSRC grant (EP/M020975/1, EP/R024642/1). This work was supported by the UK Dementia Research Institute which receives its funding from DRI Ltd, funded by the UK Medical Research Council, Alzheimer's Society and Alzheimer's Research UK.

O.W.S. and Z.Z. contributed equally to the study design and considered compression schemes and algorithms. Compression and behavioral decoding work was carried out by O.W.S., and hardware design and optimisation work was done by Z.Z. P.F. contributed the communication energy review. T.G.C. supervised the work, and directed and edited the manuscript.

Conflict of interest

We declare that we do not have any commercial or associative interests representing a conflict of interest in connection with the work submitted.

References

[1] Chethan Pandarinath, Paul Nuyujukian, Christine H Blabe, Brittany L Sorice, Jad Saab, Francis R Willett, Leigh R Hochberg, Krishna V Shenoy, and Jaimie M Henderson. High performance communication by people with paralysis using an intracortical brain-computer interface. *Elife*, 6:e18554, 2017.

[2] Adrien B Rapeaux and Timothy G Constandinou. Implantable brain machine interfaces: first-in-human studies, technology challenges and trends. *Current Opinion in Biotechnology*, 72:102–111, 2021. Tissue, Cell and Pathway Engineering.

[3] Patrick D Wolf and WM Reichert. Thermal considerations for the design of an implanted cortical brain–machine interface (BMI). *Indwelling Neural Implants: Strategies for Contending with the In Vivo Environment*, pages 33–38, 2008.

[4] Joseph C LaManna, Kimberly A McCracken, Madhavi Patil, and Otto J Prohaska. Stimulus-activated changes in brain tissue temperature in the anesthetized rat. *Metabolic brain disease*, 4(4):225–237, 1989.

[5] IEEE C95. 1. Ieee standard for safety levels with respect to human exposure to electric, magnetic, and electromagnetic fields, 0 hz to 300 ghz. The Institute of Electrical and Electronics Engineers New York, NY, 2019.

[6] Arto Nurmikko. Challenges for large-scale cortical interfaces. *Neuron*, 108(2):259–269, 2020.

[7] Pavel S Yarmolenko, Eui Jung Moon, Chelsea Landon, Ashley Manzoor, Daryl W Hochman, Benjamin L Viglianti, and Mark W Dewhirst. Thresholds for thermal damage to normal tissues: an update. *International Journal of Hyperthermia*, 27(4):320–343, 2011.

[8] Chul Kim, Jiwoong Park, Sohmyung Ha, Abraham Akinin, Rajkumar Kubendran, Patrick P. Mercier, and Gert Cauwenberghs. A 3 mm × 3 mm fully integrated wireless power receiver and neural interface system-on-chip. *IEEE Transactions on Biomedical Circuits and Systems*, 13(6):1736–1746, 2019.

[9] Yaoyao Jia, Ulkuhan Guler, Yen-Pang Lai, Yan Gong, Arthur Weber, Wen Li, and Maysam Ghovanloo. A trimodal wireless implantable neural interface system-on-chip. *IEEE Transactions on Biomedical Circuits and Systems*, 14(6):1207–1217, 2020.

[10] Shuenn-Yuh Lee, Chieh Tsou, Peng-Wei Huang, Po-Hao Cheng, Chi-Chung Liao, Zhan-Xien Liao, Hao-Yun Lee, Chou-Ching Lin, and Chia-Hsiang Hsieh. 22.7 a programmable wireless eeg monitoring soc with open/closed-loop optogenetic and electrical stimulation for epilepsy control. In *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, pages 372–374, 2019.

[11] Jayant Charthad, Ting Chia Chang, Zhaokai Liu, Ahmed Sawaby, Marcus J. Weber, Sam Baker, Felicity Gore, Stephen A. Felt, and Amin Arbabian. A mm-sized wireless implantable device for electrical stimulation of peripheral nerves. *IEEE Transactions on Biomedical Circuits and Systems*, 12(2):257–270, 2018.

[12] Jihun Lee, Vincent Leung, Ah-Hyoung Lee, Jiannan Huang, Peter Asbeck, Patrick P. Mercier, Stephen Shellhammer, Lawrence Larson, Farah Laiwalla, and Arto Nurmikko. Neural recording and stimulation using wireless networks of microimplants. *Nature Electronics*, 4(8):604–614, Aug 2021.

[13] Hossein Kassiri, Muhammad Tariqus Salam, Mohammad Reza Pazhouhandeh, Nima Soltani, Jose Luis Perez Velazquez, Peter Carlen, and Roman Genov. Rail-to-rail-input dual-radio 64-channel closed-loop neurostimulator. *IEEE Journal of Solid-State Circuits*, 52(11):2793–2810, 2017.

[14] Dongjin Seo, Ryan M. Neely, Konlin Shen, Utkarsh Singhal, Elad Alon, Jan M. Rabaey, Jose M. Carmena, and Michel M. Maharbiz. Wireless recording in the peripheral nervous system with ultrasonic neural dust. *Neuron*, 91(3):529–539, 2016.

[15] Gabriel Gagnon-Turcotte, Mehdi Noormohammadi Khiarak, Christian Ethier, Yves De Koninck, and Benoit Gosselin. A 0.13-μm cmos soc for simultaneous multichannel optogenetics and neural recording. *IEEE Journal of Solid-State Circuits*, 53(11):3087–3100, 2018.

[16] Peilong Feng, Michal Maslik, and Timothy G. Constandinou. Em-lens enhanced power transfer and multi-node data transmission for implantable medical devices. In *2019 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 1–4, 2019.

[17] Eran Stark and Moshe Abeles. Predicting movement from multiunit activity. *Journal of Neuroscience*, 27(31):8387–8394, 2007.

[18] Robert D Flint, Eric W Lindberg, Luke R Jordan, Lee E Miller, and Marc W Slutzky. Accurate decoding of reaching movements from field potentials in the absence of spikes. *Journal of neural engineering*, 9(4):046006, 2012.

[19] Joseph E. O'Doherty, Mariana M. B. Cardoso, Joseph G. Makin, and Philip N. Sabes. Nonhuman primate reaching with multichannel sensorimotor cortex electrophysiology, May 2017.

[20] Joaquin Navajas, Deren Y Barsakcioglu, Amir Eftekhar, Andrew Jackson, Timothy G Constandinou, and Rodrigo Quian Quiroga. Minimum requirements for accurate and efficient real-time on-chip spike sorting. *Journal of neuroscience methods*, 230:51–64, 2014.

[21] Dongjin Seo, Ryan M Neely, Konlin Shen, Utkarsh Singhal, Elad Alon, Jan M Rabaey, Jose M Carmena, and Michel M Maharbiz. Wireless recording in the peripheral nervous system with ultrasonic neural dust. *Neuron*, 91(3):529–539, 2016.

[22] Pyungwoo Yeon, Muhannad S Bakir, and Maysam Ghovanloo. Towards a 1.1 mm 2 free-floating wireless implantable neural recording soc. In *2018 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–4. IEEE, 2018.

[23] Nur Ahmadi, Matthew L Cavuto, Peilong Feng, Lieuwe B Leene, Michal Maslik, Federico Mazza, Oscar Savolainen, Katarzyna M Szostak, Christos-Savvas Bouganis, Jinendra Ekanayake, et al. Towards a distributed, chronically-implantable neural interface. In *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 719–724. IEEE, 2019.

[24] Jihun Lee, Vincent Leung, Ah-Hyoung Lee, Jiannan Huang, Peter Asbeck, Patrick P Mercier, Stephen Shellhammer, Lawrence Larson, Farah Laiwalla, and Arto Nurmikko. Wireless ensembles of sub-mm microimplants communicating as a network near 1 ghz in a neural application. *bioRxiv*, 2020.

[25] Claude Elwood Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(3):379–423, 1948.

[26] György Buzsáki, Costas A Anastassiou, and Christof Koch. The origin of extracellular fields and currents—eeg, ecog, lfp and spikes. *Nature reviews neuroscience*, 13(6):407–420, 2012.

[27] Oscar Herreras. Local field potentials: myths and misunderstandings. *Frontiers in neural circuits*, 10:101, 2016.

[28] Nur Ahmadi, Timothy G Constandinou, and Christos-Savvas Bouganis. Robust and accurate decoding of hand kinematics from entire spiking activity using deep learning. *Journal of Neural Engineering*, 18(2):026011, 2021.

[29] Oscar W Savolainen and Timothy G Constandinou. Lossless compression of intracortical extracellular neural recordings using non-adaptive huffman encoding. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 4318–4321. IEEE, 2020.

[30] Nir Even-Chen, Dante G Muratore, Sergey D Stavisky, Leigh R Hochberg, Jaimie M Henderson, Boris Murmann, and Krishna V Shenoy. Power-saving design opportunities for wireless intracortical brain–computer interfaces. *Nature Biomedical Engineering*, pages 1–13, 2020.

[31] Deren Y Barsakcioglu, Yan Liu, Pooja Bhunjun, Joaquin Navajas, Amir Eftekhar, Andrew Jackson, Rodrigo Quian Quiroga, and Timothy G Constandinou. An analogue front-end model for developing neural spike sorting systems. *IEEE transactions on biomedical circuits and systems*, 8(2):216–227, 2014.

[32] Song Luan, Ian Williams, Michal Maslik, Yan Liu, Felipe De Carvalho, Andrew Jackson, Rodrigo Quian Quiroga, and Timothy G Constandinou. Compact standalone platform for neural recording with real-time spike sorting and data logging. *Journal of neural engineering*, 15(4):046014, 2018.

[33] Lan Luan, Jacob T Robinson, Behnaam Aazhang, Taiyun Chi, Kaiyuan Yang, Xue Li, Haad Rathore, Amanda Singer, Sudha Yellapantula, Yingying Fan, et al. Recent advances in electrical neural interface engineering: minimal invasiveness, longevity, and scalability. *Neuron*, 108(2):302–321, 2020.

[34] Jameson Thies and Amirhossein Alimohammad. Compact and low-power neural spike compression using undercomplete autoencoders. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(8):1529–1538, 2019.

[35] Tong Wu, Wenfeng Zhao, Edward Keefer, and Zhi Yang. Deep compressive autoencoder for action potential compression in large-scale neural recording. *Journal of neural engineering*, 15(6):066019, 2018.

[36] Sonia Todorova, Patrick Sadtler, Aaron Batista, Steven Chase, and Valérie Ventura. To sort or not to sort: the impact of spike-sorting on neural decoding performance. *Journal of neural engineering*, 11(5):056005, 2014.

[37] Eric M Trautmann, Sergey D Stavisky, Subhaneil Lahiri, Katherine C Ames, Matthew T Kaufman, Daniel J O'Shea, Saurabh Vyas, Xulu Sun, Stephen I Ryu, Surya Ganguli, et al. Accurate estimation of neural population dynamics without spike sorting. *Neuron*, 103(2):292–308, 2019.

[38] George W Fraser, Steven M Chase, Andrew Whitford, and Andrew B Schwartz. Control of a brain–computer interface without spike sorting. *Journal of neural engineering*, 6(5):055004, 2009.

[39] Wasim Q Malik, John P Donoghue, and Leigh R Hochberg. Real-time control of a clinical brain-computer interface using continuous wideband multiunit activity.

[40] Eric Drebitz, Bastian Schledde, Andreas K Kreiter, and Detlef Wegener. Optimizing the yield of multi-unit activity by including the entire spiking activity. *Frontiers in neuroscience*, 13:83, 2019.

[41] Matteo Pagin and Maurits Ortmanns. A neural data lossless compression scheme based on spatial and temporal prediction. In *2017 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 1–4. IEEE, 2017.

[42] David M Brandman, Tommy Hosman, Jad Saab, Michael C Burkhart, Benjamin E Shanahan, John G Ciancibello, Anish A Sarma, Daniel J Milstein, Carlos E Vargas-Irwin, Brian Franco, et al. Rapid calibration of an intracortical brain–computer interface for people with tetraplegia. *Journal of neural engineering*, 15(2):026007, 2018.

[43] Elon Musk et al. An integrated brain-machine interface platform with thousands of channels. *Journal of medical Internet research*, 21(10):e16194, 2019.

[44] Reid Harrison, Paul Watkins, Ryan Kier, Robert Lovejoy, Daniel Black, Richard Normann, and Florian Solzbacher. A low-power integrated circuit for a wireless 100-electrode neural recording system. In *2006 IEEE International Solid State Circuits Conference-Digest of Technical Papers*, pages 2258–2267. IEEE, 2006.

[45] Beata Jarosiewicz, Anish A Sarma, Daniel Bacher, Nicolas Y Masse, John D Simeral, Brittany Sorice, Erin M Oakley, Christine Blabe, Chethan Pandarinath, Vikash Gilja, et al. Virtual typing by people with tetraplegia using a self-calibrating intracortical brain-computer interface. *Science translational medicine*, 7(313):313ra179–313ra179, 2015.

[46] Zheng Zhang and Timothy G Constandinou. Adaptive spike detection and hardware optimization towards autonomous, high-channel-count bmis. *Journal of Neuroscience Methods*, 354:109103, 2021.

[47] Oscar W Savolainen and Timothy G Constandinou. Investigating the effects of macaque primary motor cortex multi-unit activity binning period on behavioural decoding performance. *bioRxiv*, 2020.

[48] Joshua I Glaser, Ari S Benjamin, Raeed H Chowdhury, Matthew G Perich, Lee E Miller, and Konrad P Kording. Machine learning for neural decoding. *Eneuro*, 7(4), 2020.

[49] Thomas Brochier, Lyuba Zehl, Yaoyao Hao, Margaux Duret, Julia Sprenger, Michael Denker, Sonja Grün, and Alexa Riehle. Massively parallel recordings in macaque motor cortex during an instructed delayed reach-to-grasp task. *Scientific data*, 5(1):1–23, 2018.

[50] Oscar Savolainen, Zheng Zhang, Peilong Feng, and Timothy Constandinou. https://github.com/next-generation-neural-interfaces/hardware-efficient-mua-compression: Code, results and hardware designs for hardware efficient compression of threshold-crossing neural mua signals via machine-learning selected static huffman encoders, Feb 2022.

[51] Oscar Savolainen, Zheng Zhang, Peilong Feng, and Timothy Constandinou. Zenodo: Hardware-efficient compression of neural multi-unit activity using machine learning selected static huffman encoders - data and results; doi: 10.5281/zenodo.6265023, 2022.

[52] Zishen Xu, Wei Wu, Shawn S Winter, Max L Mehlman, William N Butler, Christine M Simmons, Ryan E Harvey, Laura E Berkowitz, Yang Chen, Jeffrey S Taube, et al. A comparison of neural decoding methods and population coding across thalamo-cortical head direction cells. *Frontiers in neural circuits*, 13:75, 2019.

[53] Aditya Jain, Ramta Bansal, Avnish Kumar, and KD Singh. A comparative study of visual and auditory reaction times on the basis of gender and physical activity levels of medical first year students. *International Journal of Applied and Basic Medical Research*, 5(2):124, 2015.

[54] Mikhail A Lebedev and Miguel AL Nicolelis. Brain-machine interfaces: From basic science to neuroprostheses and neurorehabilitation. *Physiological reviews*, 97(2):767–837, 2017.

[55] Mikhail A Lebedev. How to read neuron-dropping curves? *Frontiers in systems neuroscience*, 8:102, 2014.

[56] Dong Han, Yuanjin Zheng, Ramamoorthy Rajkumar, Gavin Stewart Dawe, and Minkyu Je. A 0.45 v 100-channel neural-recording ic with sub-μw/channel consumption in 0.18μm cmos. *IEEE transactions on biomedical circuits and systems*, 7(6):735–746, 2013.

[57] Karim G Oweiss. A systems approach for data compression and latency reduction in cortically controlled brain machine interfaces. *IEEE Transactions on Biomedical Engineering*, 53(7):1364–1377, 2006.