Compact and Low-Power Neural Spike Compression Using Undercomplete Autoencoders

Jameson Thies and Amirhossein Alimohammad

***Abstract*—Implantable microsystems which collect and transmit neural data are becoming very useful entities in the field of neuroscience. Limited by high data rates, on-chip compression is often required to transmit the recorded data without causing power dissipation at levels that would damage sensitive brain tissue. This article presents a data compression system designed for brain computer interfaces (BCIs) based on undercomplete autoencoders. To the best of our knowledge, the proposed system is the first to achieve an average spike reconstruction quality of 14 dB signal-to-noise-and-distortion ratio (SNDR) at a 32× compression ratio (CR), 18 dB SNDR at a 16× CR, 22 dB SNDR at an 8× CR, and 35 dB SNDR at a 4× CR of neural spikes. The spike detection and autoencoder-based compression modules are designed and implemented in a standard 45-nm CMOS process. The post synthesis simulation results report that the compression module consumes between 1.4 μW and 222.5 μW of power per channel and takes between 0.018 mm^2 and 0.082 mm^2 of silicon area, depending on the desired CR and number of channels.**

I. Introduction

One of the primary motivation of brain-computer interfaces (BCIs) is to restore behavioral functions for patients who are unable to move or communicate through normal neural pathways caused by strokes or chronic deceases. BCIs capable of restoring a full range of movements to paralyzed individuals require simultaneous recordings from a relatively large population of brain neurons to accomplish acceptable prediction accuracy. However, wireless transmission of large amounts of neural data for off-line signal decoding when the number of microelectrodes approaches a few hundreds imposes serious limitations, such as the potential threat of heat-related tissue damage. For instance, the need to collect more data at higher resolutions has lead to neural recording systems with up to 1024 channels, 62.5 kHz sampling frequency, and 24-bit analog-to-digital converter (ADC) resolution [1]. Typically around 64 channels with 20 kHz sampling and 10 to 14 bit resolution is employed, [2]–[8], resulting in data rates in the Mbits/s range. At such high data rates, it is unlikely to transmit all collected data without dissipating excessive power, which would cause damage to sensitive brain tissue. To reduce the data rate of neural recording systems, many researchers have applied various compression techniques. Compressed sensing (CS) has proven to be a viable method among others [9], [10]. Most recently, it has been shown that CS in conjunction with dictionary learning is an effective and efficient approach for compressing neural signals [11], [12]. The trade-off present in all compression techniques for BCIs is power consumption for a desired signal reconstruction quality. Both the number of arithmetic operations of the underlying compression techniques and compression ratio (CR) have a direct impact on power dissipation. The compression systems proposed in previously published works aim to find a solution which produces the highest quality reconstruction while still consuming acceptable amounts of power.

The two prevailing philosophies in neural signal decoding are rate-based and spike-based decoding [13]. In rate-based decoding, information is communicated by the rate at which a spike fires. In contrast, spike-based decoding assumes that individual spikes matter, and more complicated temporal codes of spikes carry information. In either scenario, the meaning of a spike train can be decoded from a combination of knowing spike times and shapes. It follows that neural signal processing systems which detect and transmit only compressed versions of neural spikes could aid in the decoding of neural spike trains. This article proposes to utilize autoencoders for the compression of neural spikes. Autoencoders are a specific type of neural network in which the networks is trained to replicate its input [14]. Through the process of backpropagation, the autoencoder is trained to compress and reconstruct a given type of data based upon what optimal compression and expansion are learned from the training data. This network can then be mapped onto a BCI system such that the first part of the network is computed on an implantable device in-vivo, the compressed data at the bottleneck of the network is transmitted wirelessly, and an optimal reconstruction is performed off-chip where power dissipation is not a major concern.

The rest of this article is organized to progressively describe the application of autoencoders and neural networks in BCIs for the purpose of neural spike reconstruction. Section II briefly explains the mathematical basis of autoencoders and the parameters which define their training and compression performance. In Section III, we layout the structure of a BCI system and explain how the autoencoder would be incorporated for compression. Section IV presents the performance of autoencoder-based compression of neural data. Comparison with other compression methods in recently published work is discussed. Section V details a digital circuit designed to detect spikes, and compute the portion of autoencoders that would need to be implemented on a chip for brain implantation. The ASIC implementation results and comparison with other compression circuits are also presented. Finally, Section VI makes some concluding remarks.

II. Autoencoders

An autoencoder is a type of unsupervised neural network which is trained so that the network's output reproduces its input. In general, the input and output of the autoencoder must be the same size N. The two primary elements of an autoencoder are the encoder and decoder. The hidden layer

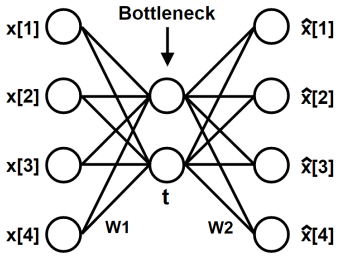

Fig. 1. An autoencoder with $N = 4$ and $M = 2$. The hidden layer between the input and output layers is considered the bottleneck, which is the point where the data is coded. Going from one layer to the next requires weight matrix multiplication, bias vector addition, and activation function computation.

of size M in an autoencoder, known as the bottleneck, is responsible for compression. Sending data through an M-sized layer gives the compression ratio $CR = N/M$. The network up to and including the M-sized layer is the encoder and the remainder of the network is the decoder. The layers in the encoder define how the input data is coded. The decoder then completes the neural network to reconstruct an approximation of the input. Significant features of the dataset can be learned while the network is trained. As with most feed-forward neural networks, each layer of the network can be represented as a matrix multiplication, followed by a vector addition and an activation function [14]. Common activation functions include a saturated linear function and a logistic sigmoid function [15].

Fig. 1 shows a basic undercomplete autoencoder. The first layer transforms a 4×1 input vector into a 2×1 vector. This part of the network is considered the bottleneck. At this point the data is both coded and compressed. The second layer of the network then decodes the 2×1 coded data vector to approximate the input. For simple autoencoders with a single hidden layer, the number of arithmetic operations of coding the data is directly proportional to the size of the code.

The mathematical representations of a simple autoencoder with one hidden layer can be given as:

$$t = f_1\left(\begin{bmatrix} W1_{1,1} & \dots & W1_{1,N} \\ \vdots & \ddots & \vdots \\ W1_{M,1} & \dots & W1_{M,N} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} + \begin{bmatrix} b1_1 \\ \vdots \\ b1_M \end{bmatrix}\right)$$

$$\hat{x} = f_2\left(\begin{bmatrix} W2_{1,1} & \dots & W2_{1,M} \\ \vdots & \ddots & \vdots \\ W2_{N,1} & \dots & W2_{N,M} \end{bmatrix} \begin{bmatrix} t_1 \\ \vdots \\ t_M \end{bmatrix} + \begin{bmatrix} b2_1 \\ \vdots \\ b2_N \end{bmatrix}\right)$$

where $\mathbf{x}$ denotes the system's input, $\mathbf{t}$ denotes the coded data, $\hat{\mathbf{x}}$ denotes the reconstructed approximation of the input, the $\mathbf{W}$ matrices denote the weights of each layer, the $\mathbf{b}$ vectors denote biases, and functions f_1 and f_2 are the activation functions of the encoder and decoder, respectively.

The relation between the size of the code and the size of the input defines how compression is achieved. The size of the coded data is either greater than (overcomplete), equal to (complete), or smaller than (undercomplete) the size of the input. In undercomplete autoencoders, the code is by definition a compressed representation of the input. Training a network to recreate data from a smaller representation forces the network to learn the most significant features of the training data. There are multiple applications of sparse data representations. For example, image compression often relies on sparse representation of images in the frequency domain.

Training an autoencoder is most often done by feeding the network a set of sample data and using a training algorithm to minimize the loss function, which measures the difference between the original and reconstructed data. Given a code size M, autoencoders learn efficient encoder and decoder weights for a given dataset. Additional parameters can be added to the loss function, which further affect the properties of the code. A general formulation for the loss function used to train undercomplete, complete, and overcomplete one-dimensional autoencoders is given as [16]:

$$L = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}[n] - \hat{\mathbf{x}}[n])^2 + c_1 r(\mathbf{W1}, \mathbf{W2}) + c_2 r(\mathbf{x}, \hat{\mathbf{x}}), \quad (1)$$

where the first term in (1) denotes the mean squared error (MSE) between input and reconstructed output, the second term denotes a weight regularization function, and the third term denotes a sparsity regularization function. Weight and sparsity regularization can be as simple as the $l1$ and $l0$ norms, respectively. The scaling factors c_1 and c_2 are used to adjust how much each term is weighted in the loss function. Regularization functions are added to the primary cost function to give the network some desired effect. For example, a sparsity regularization function adds a penalty if the results of training are not sparse. This causes the network to optimize itself to produce sparse outputs. A weight regularization penalizes the network when the weights are larger than desired, and in turn optimizes the network to perform well with small weights. While equation (1) gives a general outline of the loss function, a more specific loss function is typically determined based on the underlying application.

Neural spikes are often quite similar in their shape and take on only a few significant variations based on their meaning. In other words, the inherent dimension of neural spikes needed for classification or approximate reconstruction is less than the dimension of neural spikes in the time domain. This suggests that primary features of a neural spike can be represented by fewer values than those required in the time domain. In general, undercomplete autoencoders perform well by determining a relatively small number of features common to multiple items within a dataset, which makes them suitable for compressing predictable data. They perform especially well in situations where the expected data falls constantly into a few classes. These features of neural spikes make an undercomplete autoencoder a suitable model for compression.

Principal component analysis (PCA) [17] is commonly used to efficiently represent data and is most similar to autoencoders. PCA reduces the dimensionality of data by finding a set of mutually orthogonal vectors. These vectors can create a lower-dimensional subspace in which the most pertinent portions of data are present. An autoencoder with linear activation functions is identical to PCA. However, the linear

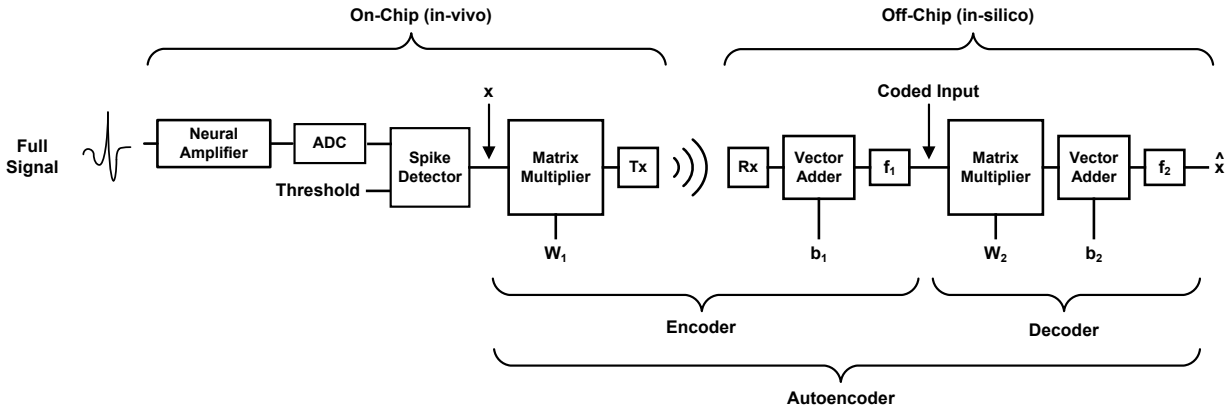

Fig. 2. System-level block diagram of a single channel BCI utilizing an autoencoder for in-vivo neural signal compression. Bias vector additions and activation functions are performed off-chip (in-silico) because neither compress data. In multi-channel systems, the elements after the neural amplifier would be multiplexed.

restrictions of PCA can limit its performance relative to a non-linear autoencoder. Primarily, autoencoders are less sensitive to variations within neural spike classes and random noise. These two advantages of autoencoders have been exploited in comparisons between autoencoders and PCA for face recognition [18]. The key advantage of PCA over autoencoders is that a less complex training process is required. However, in low-power systems where training is performed off-chip, this is not a concern.

Two other popular techniques used for low-power data compression are CS [19] and dictionary learning [11]. CS allows for sub-Nyquist rate sampling of a signal given that the signal is sparse in some domain. Similar to autoencoders, the CS sampling process requires matrix multiplication of an N-length signal by a $M \times N$ measurement matrix. After compression, a sparse solution can be found using a matching pursuit algorithm and an approximation of the original signal can be reconstructed. The efficiency of CS compression is a function of the measurement matrix. The size of the measurement matrix determines the compression ratio and the elements of the compression matrix determine the arithmetic complexity. Dictionary learning is the process of finding the dictionary capable of creating the most efficient sparse representations of a given dataset. Systems which utilize CS in conjunction with dictionary learning have shown to be very efficient at compressing full neural signals as opposed to only spikes [10], [11], [20].

III. System Framework

The system-level block diagram of our autoencoder-based compression system for BCIs is shown in Fig. 2. The part of the BCI system, that is implanted on the patient's brain (in-vivo) is responsible for reading the data, compressing it, and then sending it off-chip to be reconstructed or classified (in-silico). Since neural data between spikes is primarily noise and does not represent meaningful information in rate-based and spike-based decoding schemes [13], it is feasible to only process and transmit neural spikes for BCIs [21], [22]. By discarding the noise between neural spikes, the amount of data that must be processed by the on-chip compression module and transmitted wirelessly is greatly reduced, which also reduces the overall energy dissipation of the brain-implantable chip. Furthermore, having predictable input data allows the compression to be optimized based on the application. All BCI system frameworks share a number of characteristics including sensing elements, neural amplifiers, ADCs, and transmitters [9], [23]–[26]. The system framework shown in Fig. 2 is for a single channel system. In a multichannel system, it is common for an analog multiplexer to feed the ADC such that only one data stream arrives at the digital signal processing element and transmitter [23], [26].

A. Neural recording

To record and process neural data, one or more electrodes must be used to read the voltage levels at certain regions of the brain. For the small voltages to be used in digital circuits, they must first be amplified. Low-power neural amplifiers typically consume around 7.6 to 9.8 μW/Channel to amplify the neural signals with over 55 dB gain [3], [19]. In neural recording systems with multiple channels, the neural amplifiers are often the portion of the circuit which consumes the most power. In single channel neural recording systems, the neural amplifier leads directly to the ADC and then to the signal processing part of the circuit, as shown in Fig. 2.

B. Spike detector

The purpose of the spike detector is to detect the spikes and pass them to the compression module in an organized and predictable way. This requires a definition of a spike which can be defined in mathematical terms, based on spike window size, spike peak threshold, and spike sensitivity. The spike detector used in our system operates by finding the spike's peak values, which is defined as a value which is greater a variable threshold and is also greater than a fixed number of values surrounding it. Multiple of the most recent data points are stored and once a spike is detected, the values are sent to the compression module along with the data which comes after the spike peak for some fixed number of samples. Because the computational complexity of spike compression is

directly proportional to the spike window size N (the number of samples per spike), adjusting the amount of data which is sent to the compression module when a spike is detected directly impacts the number of arithmetic operations required for compressing a spike.

We use the peak value detection technique [27] due to its hardware simplicity. Other common spike detection approaches, including the non-linear energy operator (NEO) [28], and wavelet based detection [29], use multiplications. Additionally, the NEO method requires another module for spike alignment. By using a relatively simple detection method, which requires no multiplication nor an additional alignment module, the number of arithmetic operations and power consumption are both reduced.

C. In-vivo compression

As explained in the previous section, if the autoencoder has only one hidden layer, then data is entirely compressed by the first matrix multiplication. For this reason, the only element of the on-chip compression is a matrix multiplier. For compressing a spike with N data points into an M-length code, it must multiply an $M \times N$ weight matrix with an $N \times 1$ data vector representing the spike. The resulting $M \times 1$ length code is then transmitted to the off-chip module. The weight matrix $\mathbf{W_1}$ is constant and is stored on-chip. The matrix-vector multiplication requires $M \times N$ multiplications and $M \times (N-1)$ additions.

D. Transmitter

Classification of neural spikes is a computationally-intensive process and it is infeasible to do the vast majority of required computations within the strict power constraints of neural implants. Therefore, the data is generally transmitted off-chip for further processing. Along with the neural amplification, wireless transmission of neural signals off-chip is often the element of the BCI which consumes the most power. Wireless transmission typically consumes between 30 μW and 6 mW [30]–[34]. How much power the transmitter consumes is largely a function of data rate. Large-scale transmitters for medical implants transmit data in the Mbps range and consume power at the mW level [31]–[33]. In contrast, small-scale transmitters for medical implants transmit data in the Kbps range and consume power in the tens of μW range. For example, the transmitter proposed in [34] transmits 10 Kbps and uses 3 nJ per bit. By comparison, digital compression circuits often only consume 0.5-40 μW power [9], [23]–[26]. It follows that in small-scale BCIs with relatively low transmission data rates, the reconstruction quality becomes the key decision factor in compression circuit as opposed to power consumption.

E. In-silico reconstruction

The off-chip receiver receives partially coded neural signals from the implantable device. The data is completely coded off-chip and then decoded to attain an approximation of the original signal. Because there are no strict restrictions on the size and power consumption of the off-chip reconstruction, it is hence done by software in floating-point arithmetic. As shown in Fig. 2, the off-chip reconstruction requires two vector additions, two activation functions computations, and a matrix multiplication. It is not necessary to use an activation function in the decoder.

IV. System Validation

A. Validation using simulated data

For the primary evaluation of the autoencoder-based compression of neural spikes, we use the publicly available Leicester Database [35]. The database includes 20 different simulated neural waveforms, each of which contain three different classes of neural spikes. The simulated waveforms were created by randomly selecting spikes from a database of 594 spike shapes from neural recordings, then placed with random amplitudes and times in each waveform to resemble spikes from other neurons. Then, a preselected spike train with predefined types of spikes was imposed on the random signal. Noise with a standard deviation between 0.05 and 0.4 relative to the magnitude of the spikes was added to the simulated waveform. The data is first simulated at 96 kHz, then down sampled to 24 kHz. Within the database, datasets further differ based on the ease of spike classification. The database includes only two levels of classification difficulty (Easy and Difficult). For our primary evaluation, we used the Leicester dataset with Easy classification difficulty and noise with 0.05 standard deviation, which presents a -30.5019 dB signal-to-noise-ratio (SNR). Because the systems are being tested for their ability to reconstruct neural spikes, as apposed to classify them, we found that spike classification difficulty had little to no effect on compression and reconstruction performance. For a fair comparison, we have filtered out overlapping spikes [11]. Since the dataset we used for training the networks includes simulated spikes which fire from far off neurons, distant neuron firings are accounted for in the training and testing of the autoencoders. Fig. 3 shows the spikes used for our initial evaluation, which are captured within a window of 64 data points, and separated by the class of spike. Our spike detection module detects 99.08% of spikes in our primary dataset. While our spike detection performs well, noisier environments may warrant the use of more accurate and computationally complex spike detection techniques, such as NEO.

The designed autoencoders are trained using the MATLAB Deep Learning Toolbox for multiple datasets at multiple compression ratios with variations on the parameters, such as window size and the activation function. We used scaled conjugate gradient backpropagation with an MSE-based loss function, which includes a relatively small L_2 weight regularization. Our utilized cost function is given as:

$$C = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}(x_{kn} - \hat{x}_{kn})^2 + 0.0001\Omega,$$

where x denotes the training data, N denotes the length of each spike in samples, K denotes the number of training samples, and Ω denotes an L_2 weight regularization. We found

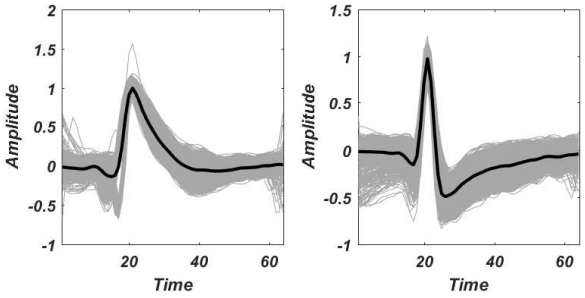

Fig. 3. Simulated neural spikes from the Leicester database [35]. Data on the left from C_Easy2_noise005.mat have Easy classification difficulty and data on the right from C_Difficult1_noise005.mat have difficult classification difficulty. The thick black line is the mean. As shown above, 64 samples are sufficient to capture the spikes.

that a weight regularization coefficient of 0.0001 was small enough for the network to have near optimal performance, but also large to keep the weights small enough that only a few integer bits are required for the fixed-point digital logic implementation of the cost function. Rather than training over a set number of epochs, each autoencoder is trained until the performance gradient fell below 0.000001. To measure the reconstruction quality, we use the signal-to-noise-and-distortion-ratio (SNDR) defined as $20\ log_{10}\frac{||\mathbf{x}||_2}{||\mathbf{x}-\hat{\mathbf{x}}||_2}$, where $\mathbf{x}$ and $\hat{\mathbf{x}}$ denote the original and reconstructed signal, respectively. In addition to providing a reliable metric for gaging accuracy, SNDR allows us to compare the reconstruction quality of various compression methods.

Among alternative activation functions for the encoder, including a saturated linear function and hyperbolic tangent, we found that the logistic sigmoid consistently outperformed the other two. Because the slope of the logistic around zero is lower than the slope of the hyperbolic tangent, there is more resolution in the result after rounding errors introduced by a fixed-point implementation. Therefore, for the activation function in the encoder, we used the logistic sigmoid function $f(z) = 1/(1 + e^{-z})$. For the decoder, we used a linear activation function $f(z) = z$ to avoid imposing limitations on the output. Because the activation function in the decoder is the last step in the autoencoder, the range of the activation function is the range of the output. Since other common activation functions have an output value between -1 and 1, the incoming data would have to be scaled for logistic sigmoid, saturated linear, and hyperbolic tangent functions. While scaling would allow using a more complex activation function, it would increase the likelihood of underflow in fixed-point implementations.

To determine how the spikes within the dataset should be divided into training and testing spikes, autoencoders with multiple sizes are trained with a varying amount of data used for training. Then each autoencoder is tested with the remaining spikes in the dataset. Note that these networks were trained using spikes detected in our primary dataset. After detecting spikes and removing overlapping spikes, there were 2,789 spikes in the dataset to be used for testing and training. The mean SNDR of the reconstructed test spikes are given in Table I. From Table I, it can be seen that the autoencoders perform best when 70% of each dataset is used for training, but they perform nearly as well when only 30% of each dataset is used for training.

TABLE I
COMPARISON OF MEAN SNDR VALUES OF SPIKES RECONSTRUCTIONS AS A FUNCTION OF PERCENTAGE OF DATA USED FOR TRAINING

Percent of data used for training	*Mean SNDR* (dB) *CR = 16*	*Mean SNDR* (dB) *CR = 8*	*Mean SNDR* (dB) *CR = 5.33*	*Mean SNDR* (dB) *CR = 4*
5%	18.68	23.08	27.87	32.68
10%	18.92	23.49	28.73	33.56
20%	18.98	23.61	29.62	33.71
30%	19.00	23.70	29.50	34.06
40%	18.99	23.64	29.61	34.19
50%	19.02	23.66	29.63	34.13
60%	19.00	23.61	29.66	34.04
70%	19.04	23.60	29.68	34.17

For our primary evaluation, we continued to use the Leicester Dataset with easy classification difficulty, and added noise with 0.05 standard deviation, and overlapping spikes removed. We trained the autoencoders for spikes with a window size of both 64 and 128 for comparison, but we found that 64 samples are sufficient to capture a spike and lead to a more efficient compression system with reduced number of arithmetic operations. For the remaining autoencoders in this section, 837 spikes (30%) were used for training and 1,952 spikes (70%) were used for testing.

After training autoencoders for all combinations of $N \in \{64, 128\}$ and $M \in \{4, 8, 12, 16, 20, 24, 28, 32\}$, we used them to compress and reconstruct all test spikes and calculated the SNDR for each test. Then, we averaged the reconstructed spikes' SNDR for all 1,952 test spikes to find the mean SNDR in dB of a given autoencoder. Table II gives the resulting mean SNDR, SNDR standard deviation, minimum SNDR, and number of training epochs for all autoencoders trained and tested using the data shown in Fig. 3. We repeated this process using Leicester Difficult data with 0.05 standard deviation noise. The difficult data had 3,007 spikes, which were separated into 902 training spikes and 2,105 test spikes. After training 16 autoencoders with varying input sizes and compression ratios, we tested all autoencoders on all test spikes. the performance of autoencoder compression on Difficult Leicester data is given in Table III. Autoencoders perform nearly as well on the Difficult dataset as they perform on the Easy dataset at high compression ratios. At lower compression ratios, autoencoders perform better on the Difficult dataset than the Easy data. Autoencoders perform well when compressing nearly uniform data. The property of the Difficult dataset which makes it difficult to classify is that the spike classes are similar. While this property makes classification more challenging, it actually alleviates compression's complexity.

The result of training the networks is finding efficient values for the weights and biases, given the parameters of the network. Not only do the weights provide information about the important features of neural spikes, but also partially establish the requirements of digital circuit implementations. We found that the weights in trained autoencoders mirror the spikes themselves. This allows us to conclude that the

TABLE II
ACCURACY OF THE AUTOENCODERS TRAINED AND TESTED USING SIMULATED NEURAL SPIKES (EASY DATASET)

N	CR	*Mean SNDR* (dB)	*SNDR Standard Deviation* (dB)	*Worst Case SNDR* (dB)	Epochs
64	16.00	19.00	2.71	7.13	1317
64	8.00	23.70	2.91	11.22	1694
64	5.33	29.50	3.00	12.72	1617
64	4.00	34.06	2.62	15.65	3619
64	3.20	35.07	2.46	16.64	2648
64	2.67	35.15	2.45	16.66	3009
64	2.29	35.20	2.44	16.74	2299
64	2.00	35.23	2.44	16.81	3523
128	32.00	13.45	3.17	2.21	1725
128	16.00	15.82	3.34	2.90	1744
128	10.67	18.27	3.49	3.71	2980
128	8.00	20.80	3.61	5.00	2076
128	6.40	23.34	3.80	5.72	3698
128	5.33	25.80	3.95	6.42	4202
128	4.57	28.38	4.13	8.04	3187
128	4.00	30.35	4.26	8.53	3942

TABLE III
ACCURACY OF THE AUTOENCODERS TRAINED AND TESTED USING SIMULATED NEURAL SPIKES (DIFFICULT DATASET)

N	CR	*Mean SNDR* (dB)	*SNDR Standard Deviation* (dB)	*Worst Case SNDR* (dB)	Epochs
64	16.00	17.90	2.79	9.80	2354
64	8.00	22.98	2.99	13.53	3303
64	5.33	28.53	3.11	17.11	3388
64	4.00	34.27	2.84	19.59	2131
64	3.20	38.62	2.68	25.16	2649
64	2.67	41.75	2.90	27.20	3165
64	2.29	43.67	3.08	28.01	3680
64	2.00	49.03	3.23	31.69	4196
128	32.00	11.71	3.81	1.76	1162
128	16.00	14.25	4.32	2.61	2190
128	10.67	15.76	4.55	2.83	3218
128	8.00	17.93	5.04	3.06	4245
128	6.40	19.76	5.41	3.31	5273
128	5.33	22.23	5.85	3.59	6299
128	4.57	23.95	6.05	4.28	7328
128	4.00	25.89	6.19	5.00	8355

autoencoders learned that a spike's peak is the most significant portion of the spike and have the largest influence on the coded values. As the compression ratio lowers, the network transmits more data per spike. This allows more possible compressed representations of spikes and hence, more accurate reconstruction.

B. Validation using in-vivo data

We also tested our design with a dataset of a single cell neural waveform recordings from epileptic patients [36]. Compared to the Leicester database, this dataset is not as widely used as a benchmark for neural data compression. However, it would serve to illustrate how effective autoencoders are for neural data compression of nonuniform spikes. Fig. 4 shows the complete dataset of human neural waveforms. It includes 9,195 spikes, which we scaled by a factor of $1/256$, and divided into 2,758 (30%) training spikes and 6,437 (70%) testing spikes. As with the simulated dataset, we trained several autoencoders for different compression ratios. Once again, we used a window size of $N = 64$, a logistic sigmoid activation function in the encoder, and no activation function in the decoder. As with the previously trained autoencoders, we used an MSE based cost function. In contrast with the autoencoders trained on simulated data, we did not use an L_2 weight regularization for training networks on this dataset. Table IV gives the mean reconstruction accuracy of the autoencoders trained and tested using the recorded neural spikes with $N = 64$.

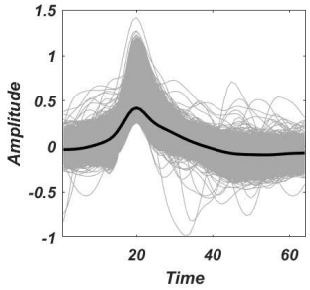

Fig. 4. Recorded neural data from epileptic patients [36]. The thick black line shows the mean.

TABLE IV
ACURACCY OF THE AUTOENCODERS TRAINED AND TESTED USING RECORDED NEURAL SPIKES

CR	*Mean SNDR* (dB)	*SNDR Standard Deviation* (dB)	*Worst Case SNDR* (dB)	Epochs
16.00	12.41	3.55	1.88	1746
8.00	15.74	3.76	5.00	1110
5.33	19.45	3.90	8.03	1615
4.00	24.75	3.98	11.77	4261
3.20	30.82	4.02	13.59	2648
2.67	37.44	4.06	12.87	3164
2.29	42.85	4.06	15.22	3681
2.00	45.09	4.08	10.45	5783

In comparison to simulated data, the in-vivo data is less uniform. This lowers the reconstruction accuracy, especially at high compression ratios. Comparing Tables II and III to Table IV shows that autoencoders perform around 5 dB better on simulated data when the compression ratio is greater than 3. Moreover, the decompressed signals reconstruction SNDR have about 1 dB greater standard deviation. At lower compression ratios, autoencoders perform comparable on simulated and in-vivo datasets. Fig. 5 illustrates this trend.

C. Comparison to other spike compression methods

Various published work suggest alternative techniques for compressing neural signals with varying degrees of accuracy. Common compression techniques include the Discrete Wavelet Transform (DWT) [37] and dictionary learning [11], [12], [38]. The DWT transforms data by passing it through a series of lowpass and highpass filters to define it in terms of both frequency and time. While the DWT has the advantage of not needing a training phase, the convolution operations

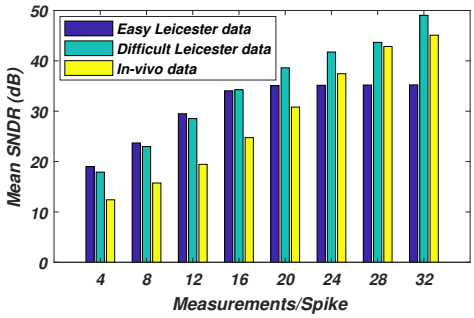

Fig. 5. Autoencoder compression on multiple datasets. $N = 64$

.

make hardware implementation power hungry. Furthermore, the DWT underperforms compared to autoencoders, as shown in Fig. 6. Note that the DWT compression shown in Fig. 6 uses Daubechies wavelets *db7* and *db17* for $N = 64$ and $N = 128$, respectively. We tested DWT compression of neural spikes in the Leicester dataset using Haar wavelets, Daubechies wavelets, coiflets, biorthogonal wavelets, Fejer-Korovkin filters, and reverse biorthogonal wavelets and found Daubechies wavelets performed the best on this data. Recently, sparse representations [39] have been studied as a way to efficiently interpret data. Dictionary learning is a more specific form of compression that uses sparse representations. It relies on multiplying an N-length input vector by an $M \times N$ sensing matrix which satisfies the random isometry property (RIP) to compress the data into an M-length vector. Common matrices which satisfy the RIP are Gaussian matrices and Bernoulli matrices [11]. If the data is sparse in relation to some dictionary D, then it can be partly recovered after compression using a matching pursuit algorithm. As with the proposed autoencoders, the number of arithmetic operations required in the dictionary learning based systems is $N \times M$ multiplications and $M \times (N-1)$ additions. Extensive recent research prove the viability of using dictionary learning for the compression of neural data [11], [12], [40]. These published work found success using Bernoulli matrices as the sensing matrix. Because the sensing matrix in dictionary learning with Bernoulli matrices only include 1 and -1 entries, the number of arithmetic operations is effectively lowered to $M \times (N-1)$ additions. More recently, dictionary learning in conjunction with on-line spike sorting and compressed sensing [38] has been shown to be an efficient method providing a mean SNDR of 11.60 dB and 10.21 dB at CR = 10 on Easy and Difficult Leicester data, respectively. On the more narrowly defined task of compressing neural spikes as opposed to the full neural signal, autoencoders outperform dictionary learning reconstruction at a higher compression ratio of CR = 16. The increased amount of data processing and transmission required to compress and reconstruct a full neural signal is unnecessarily complex in systems which only use the neural spike shape and time for decoding.

Fig. 6 shows the SNDRs for various compression techniques and those of our proposed autoencoder-based compression. Partial discrete cosine transformation (DCT) is a compression

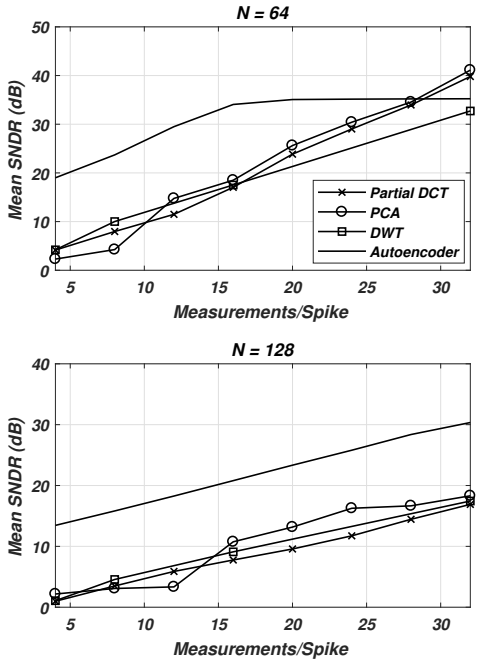

Fig. 6. Reconstruction accuracies of various compression methods. All methods in this comparison used the Easy Leicester data [35].

method in which only the DCT coefficients representing the frequencies most present in a given dataset are computed. When the signal is reconstructed, all other coefficients are assumed to be zero. PCA reduces the spike into its most significant components based upon a set of test spikes. Our proposed method reconstructs the spikes with a greater SNDR at $M = 4$ (CR = 32) than any other methods with $M = 12$ (CR = 10.6667).

V. Hardware Architecture of the Compression Module

We have designed the spike detection unit and the compression module. The circuit would accept a neural waveform at a constant data rate at the input, and its purpose is to efficiently parse the neural signals for spikes and then compress each spike. The compression circuits of the published BCI systems take up to 0.4 mm^2 of area, as given in Table VI. Using a current CMOS process, a great deal of digital logic can fit within the reasonable size of a compression circuit for BCIs. Moreover, due to the relation between system frequency and power dissipation, we optimized the circuit to operate at relatively low frequencies. The number of electrodes can exceed 1024 [1], but 64 is widely used [4], [6], [8]. To process the incoming data, the system operates with a frequency equal to the data rate. Each channel typically samples at around 20 kHz and hence, the spike detector and compression circuit will likely have to process between 20,000 and 1,280,000 samples per second.

Because the identifying portion of the spike reaches the detector after the beginning of the spike, the data which is part of the spike and arrives before the peak must be stored. We use

TABLE V
CHARACTERISTICS AND IMPLEMENTATION RESULTS OF THE AUTENCODERS WITH SPIKE DETECTOR IN STANDARD 45-NM CMOS PROCESS.

M	*WL*	*Freq. (MHz)*	*Internal Power (μW)*	*Leakage Power (μW)*	*Switching Power (μW)*	*Total Power (μW)*	*Area (mm^2)*
4	14	0.02	0.38	0.23	52.05	52.65	0.018
4	14	1.28	24.11	14.26	52.05	90.42	0.018
8	14	0.02	0.68	0.40	87.73	88.81	0.031
8	14	1.28	43.40	26.17	87.78	157.4	0.031
12	16	0.02	1.30	0.84	170.8	172.9	0.064
12	16	1.28	83.42	53.63	168.7	305.8	0.064
16	16	0.02	1.88	1.20	219.5	222.5	0.082
16	16	1.28	120.5	76.38	219.5	416.4	0.082

an input buffer of 22 registers. At the output of the last buffer, a multiplexer either permits data to be sent to the compression module or discards it. At each clock cycle, the data in the second input register is compared to the data in the first input register, third input register, and a threshold value. If the value in the second register is greater than all three other values, then its value is the peak of a spike. For the next 64 clock cycles, the multiplexer at the output allows data to be sent to the compression unit. The size of the buffer determines where in the spike's window the peak occurs. Having a buffer size of 22 causes the peak to appear at the 20-th index of the spike. The data is sent from the spike detector to the compression unit, one value at a time, in the burst of 64 data points for each spike. The compression unit is then responsible for multiplying the 64 inputs by the weight matrix.

Fig. 7 shows the block diagram of the designed compression unit. An array of M multiply-accumulate (MAC) units accept the inputs in parallel and multiply each of the 64 inputs by a weight corresponding to the input's index within the spike. To feed the weights to the MACs, a series of M read-only memories (ROMs) are used. All ROMs are addressed with one counter in the control unit. To reduce the power consumption and the number of arithmetic operations of the proposed design, all operations are computed in fixed-point arithmetic. While using fixed-point arithmetic is more power and area-efficient than floating-point computation, the quantization error must be controlled. Fig. 8 shows the SNDR for each autoencoder as a function of word lengths (WL). For a relatively small M, 4 integer bits and 10 fraction bits ($WL = 14$) are sufficient to keep SNDR within 0.1 — 0.6 dB of the floating-point's SNDR. At larger values of M, 4 integer bits and 12 fraction bits ($WL = 16$) are required to reconstruct the spikes within 0.1 — 0.6 dB of the floating-point's SNDR. For the weights, the same number of fraction bits as the rest of the system is used, but only two integer bits are required due to the small magnitude of the weights. Off-chip operations use floating-point arithmetic.

We designed and synthesized eight autoencoders using Synopsis DC Compiler in a standard 45-nm CMOS technology using the OSU-SoC standard cell kit from FreePDK [41]. After netlist compilation, Cadence SOC Encounter was used for place and route. After place and route, the netlists of each design was exported from Cadence SOC Encounter. We then used Cadence NCLaunch to compile the netlist with the standard cell library. A testbench for each compiled design was developed. Our testbenches took 4,000 samples from the Easy Leicester data, converted them to fixed-point, then repeatedly

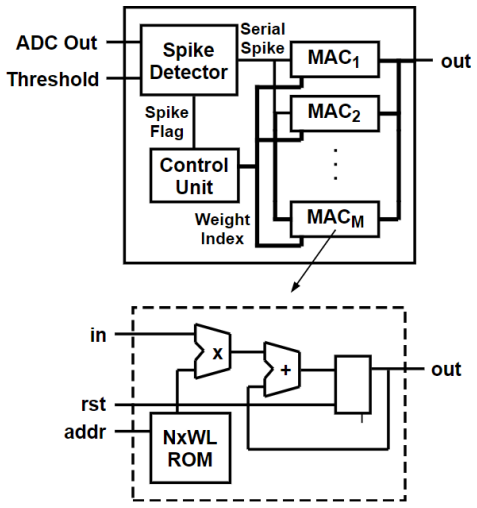

Fig. 7. Block diagram of the compression unit.

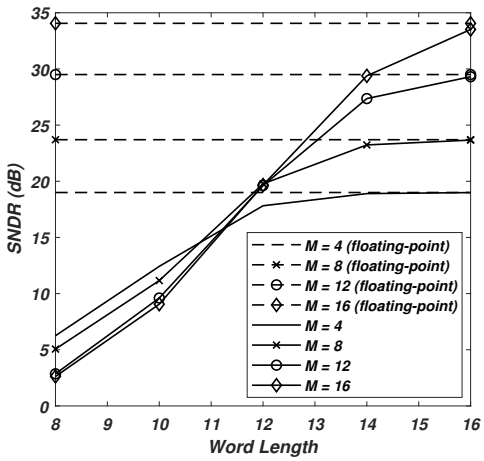

Fig. 8. Reconstruction accuracy for various word lengths. These were obtained using the autoencoders and test spikes from Table II.

sent the 4,000 samples as input to the circuits under test. Each circuit was tested using Cadence SimVision for 100 ms and the netlist activities were saved as a Value Change Dump (VCD) file. The contents of the VCD files were then used as the activity profile of the circuits. We used Cadence Encounter to estimate the power consumption of the placed and routed circuits using the switching activity specified in the VCD file at a 1.1 V operating voltage. Table V gives the estimated area and power consumption of various autoencoders. The

TABLE VI
CHARACTERISTICS AND ASIC IMPLEMENTATION RESULTS OF THE AUTOENCODERS COMPARED TO OTHER BCI COMPRESSION CIRCUITS

Work	*Method*	*CR*	*Process*	*Channels*	*Power/Ch. (μW)*	*Area/Electrode (mm^2)*
Proposed, $M = 4$	Detector + Autoencoder	16	45-nm	1	52.65	0.018
Proposed, $M = 4$	Detector + Autoencoder	16	45-nm	4	13.61	0.018
Proposed, $M = 4$	Detector + Autoencoder	16	45-nm	16	3.85	0.018
Proposed, $M = 4$	Detector + Autoencoder	16	45-nm	64	1.41	0.018
[9]	CS	8	180-nm	16	4.8	0.19[1]
[19]	CS	2.05	90-nm	1	1.9	0.09
[23]	CS	Up to 16	180-nm	16	9.4 [3]	0.0625
[24]	CS	6	130-nm	64	0.028	-
[26]	Slope Prediction	2.55	350-nm	4	0.54 [3]	0.4
[40]	CS + Dictionary Learning	10.6	180-nm	4	0.83	0.11

[1]Estimate from Figure 9 in [9].
[2]CR = 8.3 from spike detection. The detected spike is not compressed.
[3]Includes ADC power.
[4]Power of proposed circuits is based on simulation.

number of data points N per spike is 64 for all systems listed in Table V. The frequencies of 20 kHz and 1.28 MHz represent the 1-channel and 64-channel systems, respectively. In addition to the total power consumption, power dissipated per channel is an important factor to gage how efficiently data is compressed. For the systems with 0.02 MHz frequency, the power consumption per channel is equal to the total estimated power and for the systems operating at 1.28 MHz, the power per channel is the total estimated power divided by 64. This is because when N, M, and WL do not change, the circuit is the same design. To increase the number of channels the circuit can process, the frequency is proportionally increased. One can see from the results in Table V that increasing the number of channels increases the power efficiency by amortizing leakage power across channels. Utilizing standard cell libraries that allow for a higher threshold voltage could also increase the power efficiency of circuits operating at relatively low frequencies by reducing overall leakage power. Note that the designs were not fabricated.

Table VI gives the characteristics and implementation results of various published compression circuits along with ours. The method given in [40] compresses spikes with CS and dictionary learning, then uses spike clustering in principle component subspace for classification. The designs in [40] and [19] consume power and area comparable to our designs. Using a sensing matrix and a dictionary D, CS compression allows for neural signals to be sampled at sub-Nyquist rates. While our proposed designs have comparable area and power consumption, our designed system offers the highest compression ratio, which leads to minimal data being sent, which reduces overall on-chip power consumption. Our proposed designs are able to reconstruct the transmitted neural spikes more efficiently than the systems listed in Table VI because they focus on full neural signal transmission as opposed to neural spike transmission.

While neural spikes can be efficiently compressed and reconstructed using autoencoders, an autoencoder-based system for the compression of neural spikes is not without concern. Spike morphology can change over time and a new spike class may be introduced into the neural recordings. While training on a diverse dataset can hedge against unforeseen occurrences,

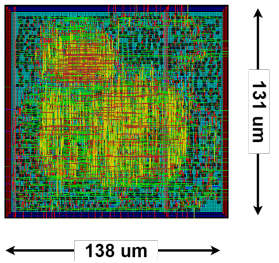

Fig. 9. Layout of the designed compression circuit for M = 4 at 1.28 MHz to support 64 channels in a standard 45-nm CMOS technology.

it will cause the autoencoder to begin performing at a lower level. In this case, retraining and re-uploading of the autoencoder weights to the implementable device may be required. Techniques for increasing the robustness of autoencoder-based compression circuits for signals which may change over time is an open concern.

VI. CONCLUSION

This article presented systems which use trained autoencoders for compression and reconstruction of neural signals. Among various techniques for compressing neural data, our proposed architectures provide high reconstruction quality of spikes when only 4 to 32 samples are sent per spike. Using an autoencoder to compress N data points into an M-length code requires $M \times N$ multiplications and $M \times (N - 1)$ additions, which makes it as computationally-efficient as most other previously published methods for the compression of neural data. We then synthesized the autoencoders using the standard 45-nm CMOS process to confirm that the use of autoencoders are reasonable in terms of both power consumption and silicon area. While this article presented the advantages of using trained compression for neural data, it can also be used for high-performance processing of specific data across other biomedical engineering applications.

ACKNOWLEDGMENT

This work was supported by the Center for Neurotechnology (CNT), a National Science Foundation Engineering Research Center (EEC-1028725).

REFERENCES

[1] K. A. White et al., "Single-cell recording of vesicle release from human neuroblastoma cells using 1024-ch monolithic CMOS bioelectronics," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 6, pp. 1345–1355, 2018.

[2] M. Ballini et al., "A 1024-channel CMOS microelectrode array with 26,400 electrodes for recording and stimulation of electrogenic cells in vitro," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 11, pp. 2705–2719, 2014.

[3] Y. C. Huang et al., "Ultrahigh-density 256-channel neural sensing microsystem using TSV-embedded neural probes," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 5, pp. 1013–1025, 2017.

[4] C. S. Mestais, G. Charvet, F. Sauter-Starace, M. Foerster, D. Ratel, and A. L. Benabid, "Wimagine: Wireless 64-channel ECoG recording implant for long term clinical applications," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 23, no. 1, pp. 10–21, 2015.

[5] F. Shahrokhi, K. Abdelhalim, D. Serletis, P. L. Carlen, and R. Genov, "The 128-channel fully differential digital integrated neural recording and stimulation interface," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 4, no. 3, pp. 149–161, 2010.

[6] R. Muller et al., "A minimally invasive 64-channel wireless uECoG implant," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 1, pp. 344–359, 2015.

[7] R. Shulyzki et al., "320-channel active probe for high-resolution neuromonitoring and responsive neurostimulation," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 9, no. 1, pp. 34–49, 2015.

[8] S. Yoshimoto et al., "Implantable wireless 64-channel system with flexible ECoG electrode and optogenetics probe," in *IEEE Biomedical Circuits and Systems Conference*, 2016, pp. 476–479.

[9] X. Liu et al., "A fully integrated wireless compressed sensing neural signal acquisition system for chronic recording and brain machine interface," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 4, pp. 874–883, 2016.

[10] W. Zhao, B. Sun, T. Wu, and Z. Yang, "On-chip neural data compression based on compressed sensing with sparse sensing matrices," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 1, pp. 242–254, 2018.

[11] Y. Suo, J. Zhang, T. Xiong, P. S. Chin, R. Etienne-Cummings, and T. D. Tran, "Energy-efficient multi-mode compressed sensing system for implantable neural recordings," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 5, pp. 0–0, 2014.

[12] J. Zhang et al., "Communication channel analysis and real time compressed sensing for high density neural recording devices," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 5, pp. 599–608, 2016.

[13] R. Brette, "Philosophy of the spike: Rate-based vs. spike-based theories of the brain," pp. 1–14, 2015.

[14] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[15] R. G. F. Soares and E. J. S. Pereira, "On the performance of pairings of activation and loss functions in neural networks," in *International Joint Conference on Neural Networks*, 2016, pp. 326–333.

[16] A. Sangari and W. Sethares, "Convergence analysis of two loss functions in soft-max regression," *IEEE Transactions on Signal Processing*, vol. 64, no. 5, pp. 1280–1288, 2016.

[17] I. T. Jolliffe, *Principal Component Analysis*. New York: Springer-Verlag, 2002.

[18] K. Siwek and S. Osowski, "Autoencoder versus PCA in face recognition," in *International Conference on Computational Problems of Electrical Engineeringf*, 2017, pp. 1–4.

[19] F. Chen, A. P. Chandrakasan, and V. M. Stojanovic, "Design and analysis of a hardware-efficient compressed sensing architecture for data compression in wireless sensors," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 3, pp. 744–756, 2012.

[20] J. Zhang, Y. Suo, S. Mitra, S. . Chin, S. Hsiao, R. F. Yazicioglu, T. D. Tran, and R. Etienne-Cummings, "An efficient and compact compressed sensing microsystem for implantable neural recordings," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 4, pp. 485–496, 2014.

[21] Y. Liu, S. Luan, I. Williams, A. Rapeaux, and T. G. Constandinou, "A 64-channel versatile neural recording SoC with activity-dependent data throughput," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 6, pp. 1344–1355, 2017.

[22] B. Senevirathna, A. Castro, T. Datta-Chaudhuri, E. Smela, and P. Abshire, "Characterization of an active micro-electrode array with spike detection and asynchronous readout," in *IEEE International Midwest Symposium on Circuits and Systems*, 2017, pp. 627–630.

[23] M. Shoaran, M. H. Kamal, C. Pollo, P. Vandergheynst, and A. Schmid, "Compact low-power cortical recording architecture for compressive multichannel data acquisition," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 6, pp. 857–870, 2014.

[24] D. Gangopadhyay, E. G. Allstot, A. M. R. Dixon, K. Natarajan, S. Gupta, and D. J. Allstot, "Compressed sensing analog front-end for bio-sensor applications," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 2, pp. 426–438, 2014.

[25] W. Biederman et al., "A 4.78 mm2 fully-integrated neuromodulation SoC combining 64 acquisition channels with digital compression and simultaneous dual stimulation," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 4, pp. 1038–1047, 2015.

[26] C. J. Deepu, X. Zhang, W. S. Liew, D. L. T. Wong, and Y. Lian, "An ECG-on-Chip with 535 nw/channel integrated lossless data compressor for wireless sensors," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 11, pp. 2435–2448, 2014.

[27] I. Obeid and P. D. Wolf, "Evaluation of spike-detection algorithms for a brain-machine interface application," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 905–911, 2004.

[28] J. F. Kaiser, "On a simple algorithm to calculate the 'energy' of a signal," in *International Conference on Acoustics, Speech, and Signal Processing*, 1990, pp. 381–384 vol.1.

[29] K. H. Kim and S. J. Kim, "A wavelet-based method for action potential detection from extracellular neural signal recording with low signal-to-noise ratio," *IEEE Transactions on Biomedical Engineering*, vol. 50, no. 8, pp. 999–1011, 2003.

[30] A. Hennessy and A. Alimohammad, "Design and implementation of a digital secure code-shifted reference UWB transmitter and receiver," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 7, pp. 1927–1936, 2017.

[31] H. Bahrami, S. A. Mirbozorgi, L. A. Rusch, and B. Gosselin, "Integrated UWB transmitter and antenna design for interfacing high-density brain microprobes," in *IEEE International Conference on Ubiquitous Wireless Broadband*, 2015, pp. 1–5.

[32] H. Miranda and T. H. Meng, "A programmable pulse UWB transmitter with 34% energy efficiency for multichannel neuro-recording systems," in *IEEE Custom Integrated Circuits Conference 2010*, 2010, pp. 1–4.

[33] A. Ebrazeh and P. Mohseni, "30 pj/b, 67 Mbps, centimeter-to-meter range data telemetry with an IR-UWB wireless link," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 9, no. 3, pp. 362–369, 2015.

[34] J. L. Bohorquez, A. P. Chandrakasan, and J. L. Dawson, "A 350μw CMOS MSK transmitter and 400μw OOK super-regenerative receiver for medical implant communications," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 4, pp. 1248–1259, April 2009.

[35] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering," *Neural Computation*, vol. 16, no. 8, pp. 1661–1687, 2004.

[36] I. Fried, K. A. MacDonald, and C. L. Wilson, "Single neuron activity in human hippocampus and amygdala during recognition of faces and objects," *Neuron*, no. 5, pp. 753–765, 1997.

[37] P. Turcza, "A wavelet-based compression for neural recording system," in *International Conference on Signals and Electronic Systems*, 2016, pp. 33–36.

[38] T. Xiong et al., "An unsupervised compressed sensing algorithm for multi-channel neural recording and spike sorting," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 26, no. 6, pp. 1121–1130, 2018.

[39] A. Akbari, M. Trocan, and B. Granado, "Image compression using adaptive sparse representations over trained dictionaries," in *IEEE International Workshop on Multimedia Signal Processing*, 2016, pp. 1–6.

[40] J. Zhang et al., "A closed-loop compressive-sensing-based neural recording system," *Journal of Neural Engineering*, vol. 12, no. 23, 2015.

[41] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal, "Freepdk: An open-source variation-aware design kit," in *2007 IEEE International Conference on Microelectronic Systems Education (MSE'07)*, June 2007, pp. 173–174.