CS230

Retinal Cell Identification from Compressed Signals

Maxwell Strange, Pumiao Yan, Andrew Wang
Department of Electrical Engineering
Stanford University
{mstrange, pumiaoy, zwang267}@stanford.edu

Abstract

The creation of brain-machine interfaces opens a gateway to a range of extraordinary applications, from treating crippling neurological diseases to expanding humankind's sensory perception. Traditional interfaces had limited electrodes of < 100, and present systems target significantly larger and denser arrays for single-cell specificity such as Neuralink. This trend is currently bottlenecked by the power budget and supportable data rate of a fully implanted device. Some proposed solutions use specialized circuitry to perform compression on signals from excited retinal neurons, but certain configurations of this compression, albeit power-efficient, are lossy. To that end, this work implements a 1-dimensional convolutional neural network to reconstruct raw neuronal spikes from their compressed counterparts. By offloading this reconstruction to simple inference on a device separate from the retinal implant, we can reduce the power needs of an implantable device. Compared to a baseline reconstruction model on 8-bit compressed waves (linear-interpolation), which only achieves a MSE of 42.1 and 81% cell identification accuracy on an array of 512 electrodes recording macaque monkey's retina, our network leads to reconstruction results that achieve a MSE of 24.0 that allows a cell mosaic recovery of over 98% accuracy for on and off parasol cells. This reconstruction network makes an extra 16x compression possible at the same level of recording performance.

1 Introduction

Multi-channel action potential recording systems are widely used in neuroscientific studies and emerging clinical applications collectively known as brain-machine interfaces. While first-generation interfaces had limited electrode counts of <100, present research systems target significantly larger and denser arrays for single-cell specificity. This trend is currently bottlenecked by the power budget and supportable data rate of a fully implanted device. We turned to focus on neural signal compression and reconstruction to break the power-data rate bottleneck. An on-going research of the Murmann Group works on novel architectures for the massively parallel digitization of neural action potentials to break this bottleneck. The scheme achieves simultaneous signal compression and channel multiplexing through wired-OR interactions within an array of single-slope A/D converters[1], shown as Fig. 1. While the xy-projection compressor architecture effectively retains critical samples belonging to spikes, under certain configurations the compression is lossy. To achieve good compression performance, reconstruction is crucial to recover cell information. In this project shown as Fig. 2, we apply deep learning algorithms to find a suitable approach to reconstruct neural spikes from the compressed outputs of the Wired-OR architecture to improve the compression rate under the same performance level. The input to our main network is a 71-sample time-series of a compressed neuronal spike. The range of this data is that of an 8b ADC → [-128 to 128]. We then use a 1-dimensional convolutional neural network to reconstruct the 71-sample raw waveform. The ultimate goal is to reconstruct a large set of raw spikes over a few minutes of a dataset and pass those through state-of-the-art neurology processing software Vision to identify the retinal cells

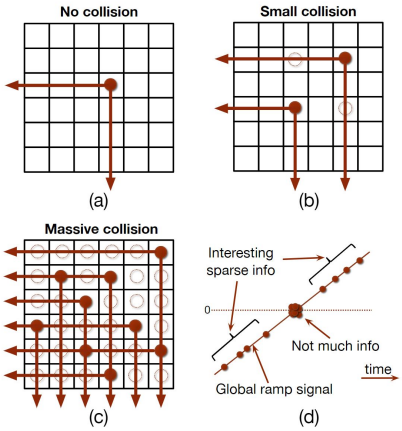


Figure 1: Compression Scheme [1]

CS230: Deep Learning, Autumn 2019, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

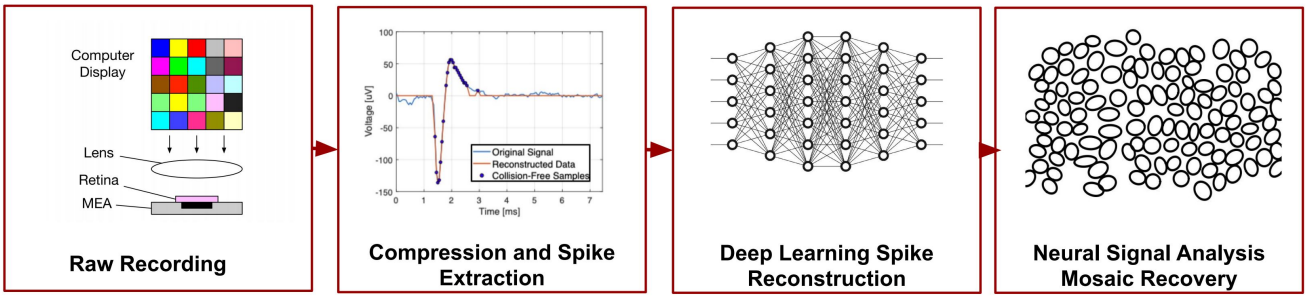


Figure 2: Project Structure

lying under the implanted electrode array. Doing this allows for lower required power and supported data rate for implantable devices, since the raw waveforms can simply be produced through inference on a device external to the implant.

2 Related work

Speech Waveform Reconstruction: Waveform reconstruction itself is not a novel idea, and has been explored in the audio domain. Some combination of residual and convolutional blocks are typically used to construct a 1D time-domain (TD) audio waveforms at the output. [6] shows that a high quality audio signal from a low-resolution, downsampled input signal. However, the dataset we are using is compressed in amplitutude, rather than compressed in time, which makes our reconstruction problem different. Convolutional neural networks have also been shown to be efficient at constructing audio, given mel-spectral features. [3] and [4] propose an interesting network for this problem using mean squared error as the loss, and will be explored more in section 4.

Bandwidth Expansion Bandwidth expansion (BWE) is the process of constructing a wideband signal, given a narrowband input signal. While BWE has been performed using both algorithmic [7] and deep neural network approaches [8], experiments have found that DNN-based bandwidth expansion can produce less distortion and a smoother frequency spectrum [8][9].

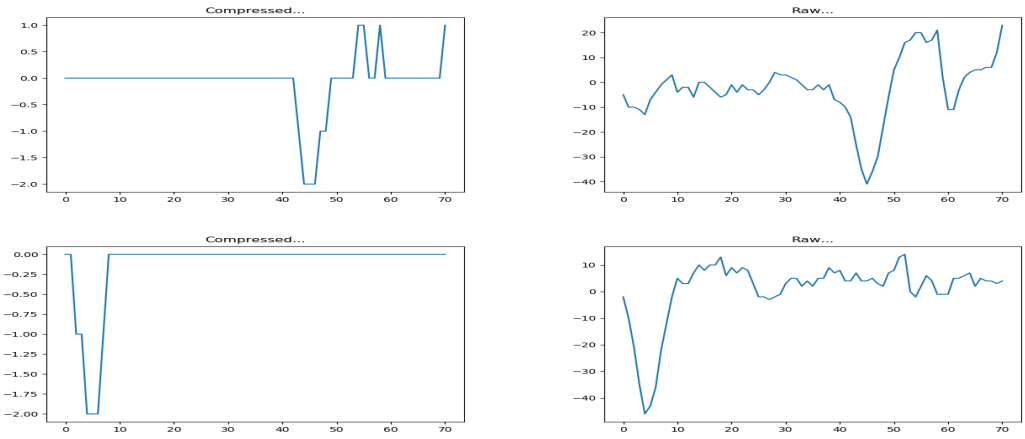
One shared characteristic among all these waveform reconstruction tasks is the use of a spectrogram. Spectrograms rely on the short-time Fourier transform (STFT), which takes a window of n time samples and produces $\frac{n}{2} + 1$ frequency bands for that window. Windows with 25ms length with a 10ms stride are often used to produce tens of frequency bands and give good resolution in the frequency domain. On the other hand, each of our neural signals are only 3.5ms in length, which significantly limits our spectrogram resolution and makes processing in the frequency domain less effective. As such, we take some basic network ideas from these projects in the audio domain as a starting point, but cannot directly copy existing approaches.

3 Dataset and Features

The dataset consists of about 60 million examples in total. We use around 38 million examples for training and use 468k examples for our dev set. Our test set consists of the remaining 20 million samples. The reasons for this unusual dataset split across training/validation/testing will become apparent in Section 5.2. These 60 million examples come from retinal response waveforms from a macaque monkey provided by Professor E.J. Chichilnisky at Stanford. A raw recording of 30 minutes from a 512-electrode array sampled at 20kHz is processed by a state-of-the-art neurology data processing tool called Vision to provide the timestamps for each spike in the recording.

The data are first converted from their native bin file format to a mat file for consumption in matlab. In their raw format, these waveforms have millivolt resolution, and in their compressed form, the waveforms have values from -128 to 128, based on an 8-bit compression scheme. Then, the waveforms are chopped into 71-sample pieces at the times corresponding to the spikes. We take these 71-sample spikes and pass them through a model of the compression inherent in the circuit configuration. After compression, we filter out those compressed results that become entirely 0. This is reasonable because there are long stretches of 0 in the other examples that can be used to learn a model on 0-valued data. Figure 3 below shows some examples of raw spikes and their compressed versions.

The last step in our data processing pipeline was to convert these (compressed, raw) spike pairs into TFRecord files. Since our dataset was too large to fit in the memory of any machines available to us, we needed to leverage the dynamic input pipelines offered by tensorflow, which unfortunately only accept these file types and not numpy arrays.



(a) Compressed spike examples (b) Raw spike examples

Figure 3: Dataset preview

4 Methods

We approached the problem of reconstructing raw spike waveforms with a few different methods, each of which attempts to exploit different features of a time series signal. We looked at implementing networks that leveraged one-dimensional convolution, the spectrogram of the time series, and 2D convolution on the spectrogram. With that being said, we found that the latter two methodologies had strange behavior when we implemented them, and we actually saw cost increase on these during training.

Our one-dimensional CNN has a structure that uses 3 main conv and max pooling layers to extract features in a set of channels, and then we push them through two large dense layers before using a 1D conv layer to collapse the channels and create the output layer. Figure 4 explicitly shows the final architecture of the 1D CNN. To train this network, we used Mini-batch gradient descent with an Adam Optimizer. The Adam optimizer leverages both momentum and RMS prop to prevent the loss from oscillating too much.

We chose mean squared error (MSE) for our loss function as we are looking for the reconstructed waveform to be similar to the raw waveform on a sample-by-sample basis. MSE is defined on a training example as:

$$MSE(\hat{y}, y) = \frac{1}{71}\sum_{i=1}^{71}(y(i) - \hat{y}(i))^2$$

where y is the raw signal and $\hat{y}$ is the reconstructed signal. By minimizing this, we hope to very closely reconstruct the noteworthy parts of the signal (where it is nonzero). Because MSE loss on a time signal tends to predict the average of the signal for stochastic portions of the waveform, we also attempted to use a combination of MSE loss in the time and frequency domains to better predict the stochastic effects [3].

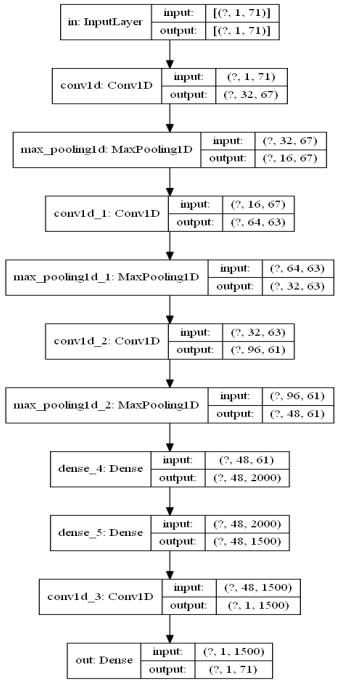


Figure 4: 1D CNN

Two main network architectures were tested using the combined loss. The first network uses a 1D CNN as previously described to output a 1D TD signal, but also adds some layers in series at the output to compute the spectrogram of the predicted TD signal. The second network computes the spectrogram of the input signal and uses a 2D CNN on the spectrogram, while a parallel branch uses a 1D CNN on the input TD signal. The inverse short-time Fourier transform is taken at the output of the spectrogram CNN to recover a TD signal, which is then added to the output of the 1D CNN to produce the final prediction. In each of these networks, the MSE loss is computed both between the TD prediction and raw signal waveform, and between the spectrogram prediction and the spectrogram of the raw signal. The relative weighting of the two losses can be tuned as hyperparameters, and the error signal is backpropagated through the STFT layers to train the rest of the network. However, training on the spectrogram results in large errors, as shown in Figure 5. Residual layers may have to be added or adjusted to improve the spectrogram loss, but time constraints prevented us from any further tuning.

5 Experiments/Results/Discussion

5.1 Hyperparameter Search

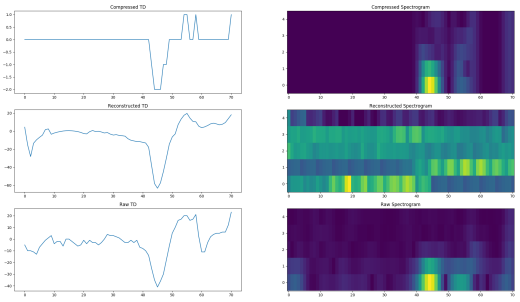


Figure 5: Spectrogram Training Waveforms

We experimented with multiple architectures, but we chose a 1D CNN architecture for the final network which we implemented in Keras [13][14]. Due to limited time constraints, we were only able to do a modest hyperparameter search on the network. We chose to search the space of the learning rate (α), the momentum parameter (β_1), and the mini-batch size. We experimented with the learning rate from .001 to .01, although choosing values from .001 to about .008 gave marginally different values in both training loss and validation loss. An interesting thing to note was that pushing the learning rate to .01 and above caused the training loss to increase rapidly between and within epochs. The actual weights are likely small values in the trained network, so it is very possible that the glorot initialization on all layers are already reasonable values, and pushing them by even .01 causes the learning algorithm to leave the optimal region of the cost space. We found that .002 gave the best training loss and validation loss in our experiments. We also experimented with changing the momentum parameter, but we only ran experiments with .85 and .9 due to limited time. We found that $\beta_1 = .85$ gave better loss without introducing variance. Finally, we tried mini-batches of size 64, 256, and 512. While epochs trained faster when using size 64, which is likely due to the underlying implementation and architecture of the GTX 1080 used for training, the variance between the training loss and dev loss decreased as we increased the batch size. Thus, the batch size we used for training was 512.

5.2 Neural Signal Analysis

The ultimate goal of neural interfaces is to record "just enough" information from neural tissues that allow clear understanding of the cell types in the case of vision restoration. As mentioned in Section 3, our train/dev/test split was atypical. According to empirical evidence from Murmann [1], for reasonable confidence in cell type classification, a recording should surpass six minutes in length. To that end, we left the first ten minutes of our raw dataset (totalling about 20 million examples) to be used for this final evaluation. Therefore, to effectively evaluate our signal reconstruction performance, we form a 10 minute recording from reconstructed spikes and perform neural signal analysis with the-state-of-the-art neurology processing software Vision to identify cells. The analysis performs clustering to categorize different neural cell types and forms a dictionary of neurons, shown as Fig. 6.

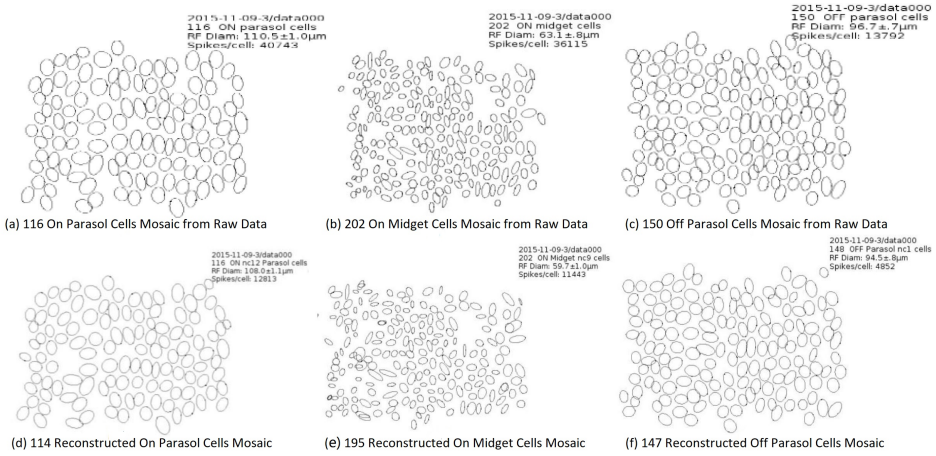


Figure 6: Mosaic Recovery Result.

5.3 Results

The primary metric we used in evaluating our network was MSE. The training and dev loss that we achieved on the final choice of hyperparameters were 23.99 and 24.88, respectively. Comparing to the baseline reported from [1], our reconstructed waveform has a lower MSE than the 8-bit resolution readout with no compression,

	12 bit	10 bit	8 bit
Conventional	0	5.8	27.9
1 wire	23.5	32.5	42.1
	74x	198x	559x
2 wire	18.7	27.5	37.5
	31x	81x	228x
4 wire	13.8	22.4	32.5
	13x	35x	96x
8 wire	7.9	16.1	28.2
	6x	15x	42x

Figure 7: MSE and Compression Rate vs. Bit Resolution and Wiring Scheme [1]

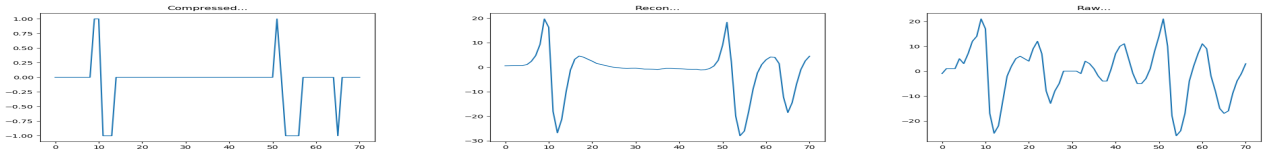
Table 1: Neural Signal Analysis Result

Mosaic Recovery	ON Parasol	ON Midget	OFF Parasol
Ground Truth / [cells]	116	202	150
Baseline /[cells]	96 (83%)	Not reported	121 (81%)
This Work	114 (98.27%)	195 (96.50%)	147 (98%)

shown as Fig.7, which means we effectively achieved more resolution. At the same time, our MSE is on par with the 4-wire-10-bit configuration, which means we effectively achieved an extra 16x compression at the same level of recording performance. After being confident that the MSE was low enough and that we were not overfitting too much to the training set, the final step in evaluating our system was to use the network to reconstruct raw signals for cell type classification. Thus, our final metric is cell type classification accuracy. We find that with our reconstructed spikes, Table 1 shows that we are able to correctly recover 98.2% of ON Parasol Cells, 98% of OFF Parasol Cells, and 96.50% of ON Midget cells. Qualitatively, we can see from the mosaic recovery results that the cells we find are very similar, as shown in Fig. 6 .

5.4 Discussion

One of our biggest concerns was overfitting our training set. The low variance makes us confident that we have not overfit, further shown by one of these training examples in Figure 8.



(a) Compressed spike example (b) Reconstructed spike example (c) Raw spike example

Figure 8: Output of network

The portions of the signal that correspond to smaller peaks and troughs in the raw signal are compressed away to 0, and the reconstructed signal doesn't show too much of that small noise. One thing we noticed was that when we tried training on smaller portions of the training set was that these zero-valued portions in the compressed form looked much more akin to the raw portion, which we took to be some form of overfitting. When training on the whole training set, however, we saw these zero-valued portions smooth out. This supports not only our claim that we have not overfit the training data, but the claim that these zero-valued portions in the compressed waves correspond to actual random noise in the raw waveform. If there was a non-random trend in these areas shared between all training examples, then we would see that trend show up in the reconstructed forms. If this portion becomes smooth and zero-valued after averaging over all 38 million examples, then it seems that it is truly near random. In this way, the dataset itself has a regularizing effect during the training of the network.

6 Conclusion/Future Work

To alleviate the requirements in power and transmission bandwidth of retinal implants for accurate retinal characterization, we developed a deep learning model to reconstruct raw signals from compressed retinal neuron responses. We showed that we are able to achieve 98% cell-type classification accuracy, which allows an extra 16x compression at the same performance level as a 12/10 bit compression strategy. Our highest performing algorithm was a 1D-CNN architecture that was trained with an Adam optimizer on mini-batch gradient descent. This algorithm likely performed better than others because of the simplicity of the compressed waveforms. The network only needed to learn simple waveform shapes that are salient to the vision software, and so it makes sense that our simplest approach gave the best results. If we had more time, we would likely do even more training (our capacity to train ended up being limited by time constraints and our large dataset) and push the size of our network down to see the smallest network that can achieve the same level of performance. Finding a smaller network would lower the overall energy of the system from end-to-end and even allow for quick inference on potentially less powerful devices.

7 Contributions

Max wrote the code for automating the data processing pipeline and formatting for training. He also wrote the main 1D CNN model as well as all other training infrastructure. Andrew worked on different models that leveraged different aspects of the spectrogram. Pumiao was the domain expert and helped with developing the dataset and guiding the results.

8 Acknowledgement

We would like to thank Prof. E.J Chichilnisky and his students for providing us the dataset and generous help on guiding us to perform neural signal analysis. We would also like to thank Dante Muratore and Pulkit Tandon for building the compression model.

References

[1] D.G. Muratore, et al. "A Data-Compressive Wired-OR Readout for Massively Parallel Neural Recording," *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, doi:10.1109/iscas.2019.8702387.

[2] C.M. Hyun, et al. "Deep learning for undersampled MRI reconstruction," *Physics in Medicine and Biology*, 63 (2018) 135007 (15pp) doi:10.1088/1361-6560/aac71a

[3] O. Watts, C. Valentini-Botinhao, and S. King, "Speech Waveform Reconstruction Using Convolutional Neural Networks with Noise and Periodic Inputs," *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, United Kingdom, 2019, pp. 7045-7049.

[4] O. Watts, C. Valentini-Botinhao, and S. King, "Fast and efficiently trainable neural speech waveform generator," 2019. [Online]. Available: https://github.com/CSTR-Edinburgh/waffler

[5] K.W. Choi, et al. "Kapre: Keras audio preprocessors," 2018. [Online]. Available: https://github.com/keunwoochoi/kapre

[6] V. Kuleshov, S.Z. Enam, and S. Ermon. "Audio super resolution using neural networks," *Workshops at International Conference on Learning Representations (ICLR)*, 2017.

[7] K.Y. Park and H.S. Kim, "Narrowband to wideband conversion of speech using GMM based transformation," *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*, Istanbul, Turkey, 2000, pp. 1843-1846 vol.3.

[8] K. Li, Z. Huang, Y. Xu, and C.H. Lee. "DNN-based speech bandwidth expansion and its application to adding high-frequency missing features for automatic speech recognition of narrowband speech," *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[9] Y. Nakatoh, M. Tsushima, and T. Norimatsu, "Generation of broadband speech from narrowband speech using piecewise linear mapping," *Proceedings of EUROSPEECH*, Vol. 3, pp. 1643-1646, 1997.

[10] J.A. Alexander and M.C. Mozer, "Template-based algorithms for connectionist rule extraction," G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press, 1995.

[11] J.M. Bower and D. Beeman, *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag, 1995.

[12] M.E. Hasselmo, E. Schnell, and E. Barkai, "Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3," *Journal of Neuroscience*, 1995, 15:5249-5262.

[13] F. Chollet, et al., "Keras," 2015

[14] M. Abadi, P. Barham, J. Chen, Z. Chen, et al. "Tensorflow: A system for large-scale machine learning," *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.