This paper has been accepted for publication at the 2023 IEEE Biomedical Circuits and Systems (BioCAS) Conference, Toronto, Canada

A 128-channel real-time VPDNN stimulation system for a visual cortical neuroprosthesis

Hasan Mohamed[1], Bogdan Raducanu[2], Ilya Kiselev[1], Zuowen Wang[1], Burcu Küçükoğlu[3], Bodo Rueckauer[3], Marcel van Gerven[3], Carolina Mora Lopez[2], Shih-Chii Liu[1]

[1] *Institute of Neuroinformatics*, University of Zurich and ETH Zurich, Zurich, Switzerland
[2]*imec*, Leuven, Belgium
[3]*Donders Institute for Brain, Cognition and Behaviour*, Radboud University, Nijmegen, the Netherlands

***Abstract*—With the recent progress in developing large-scale micro-electrodes, cortical neuroprotheses supporting hundreds of electrodes will be viable in the near future. We describe work in building a visual stimulation system that receives camera input images and produces stimulation patterns for driving a large set of electrodes. The system consists of a convolutional neural network FPGA accelerator and a recording and stimulation Application-Specific Integrated Circuit (ASIC) that produces the stimulation patterns. It is aimed at restoring visual perception in visually impaired subjects. The FPGA accelerator, VPDNN, runs a visual prosthesis network that generates an output used to create stimulation patterns, which are then converted by the ASIC into current pulses to drive a multi-electrode array. The accelerator exploits spatial sparsity and the use of reduced bit precision parameters for reduced computation, memory and power for portability. Experimental results from the VPDNN show that the 94.5K parameter 14-layer CNN receiving an input of 128 $\times$ 128 has an inference frame rate of 83 frames per sec (FPS) and uses only an incremental power of 0.1 W, which is at least 10$\times$ lower than that measured from a Jetson Nano. The ASIC adds a maximum delay of 2ms, however it does not impact the FPS thanks to double-buffered memory.**

***Index Terms*—Visual prosthesis, convolutional neural network, FPGA Accelerator, stimulation and recording ASIC**

I. Introduction

Visual prostheses such as retinal and visual neuroprostheses help to enable a rudimentary form of sight for the visually impaired [1], [2]. Recent developments in large-scale micro-electrode arrays such as the neural probe [3] show that these systems can support the stimulation of a few hundred electrodes to induce phosphene patterns, that is, point-like perception of light in response to stimulation of neuronal populations along the visual pathway (see Fig. 1). As the density of electrodes is expected to continue to grow, deep neural networks trained on tasks such as image and edge semantic segmentation [4], [5], may support the generation a more meaningful stimulation output. Convolutional neural networks (CNNs) are typically used for vision tasks and can be deployed on edge artificial intelligence (AI) platforms such as the Jetson Nano. Depending on the network size, the inference latency on these platforms can be low enough to allow the remaining inter-frame time to be used for stimulating a small set of electrodes [6]. Other CNN accelerator platforms such as FPGA, e.g. [7]–[10], and ASIC forms e.g. [11] are interesting for this application because they are usually more energy-efficient [12] and can incorporate energy-saving design strategies such as zero-skipping multiply operations of null activations [13]–[15] and the use of quantized parameters [16].

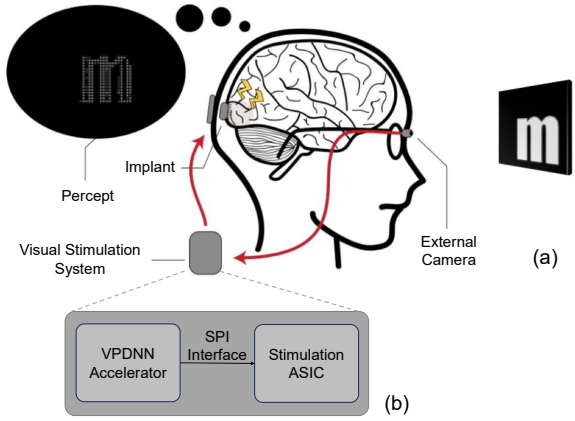

Fig. 1: (a) AI-driven visual prosthesis. Red arrows indicate flow of signals. Adapted from [17]. (b) Visual stimulation system comprising VPDNN and a stimulation ASIC whose outputs drive the implant.

In preparation for an eventual visual cortical neuroprostheses, we present a 128-channel VPDNN visual stimulation system (Fig. 1 (b)) that combines an FPGA accelerator implementing a visual prosthesis deep neural network (VPDNN), and a high-channel-count recording and stimulation ASIC [18] that can directly interface with an implantable micro-electrode array. The system should enable real-time stimulation for a visual prosthesis. The accelerator design exploits spatial sparsity in both input and the layer feature maps for reduced computation and memory resources required on the platform. The accelerator can be interfaced to a camera. The stimulation ASIC can drive up to 128 electrodes with 8 independent, fully-programmable stimulation patterns. For demonstration purposes, the output of the ASIC was displayed using a matrix LED display in this work. Section II describes the phosphene stimulation network running on the VPDNN and the FPGA CNN accelerator design. Section III describes the dataset and procedure for training the phosphene network, and Section IV

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 899287.

describes the measurement results from the system.

II. Methods

A. Phosphene Network

Current approaches for prosthetic vision rely on deep neural networks to transform rich naturalistic visual input into low-dimensional but informative stimulation signals [19]–[21]. To explore the use of our hardware stimulation system for neural prostheses, we considered the CNN phosphene network developed by [17]. This model has the structure of an autoencoder, where the encoder generates a reduced representation of the visual input. When deployed in a prosthesis, the encoder output could be used to drive the electric stimulation of the implant. To visualize stimulus-induced prosthetic vision outside the clinical setting, one commonly employs a phosphene simulator [22]. The authors of [17] pass the encoder output through a phosphene simulator and use the resulting prosthetic image representation as input to a decoder CNN, which is tasked with reconstructing the original input image. The dissimilarity between original input and phosphene-based reconstruction is used as an objective function to train the encoder to generate meaningful stimulation patterns. The phosphene simulator and decoder are only needed during offline training since, in the deployed system, the trained encoder alone remains to generate stimulation patterns. Our experiments focus on accelerating inference in a modified variant of the encoder, also called *E2E-Phosphene DNN* in this work.

B. Stimulation ASIC

The stimulation ASIC [18] implements electrical stimulation of up to 128 electrodes simultaneously. Eight independent stimulation units generate customizable square current pulses with various settings such as amplitude, duration, frequency, polarity (anodic or cathodic first), and number of periods. These circuits provide flexibility, allowing the generation of different types of pulses like biphasic or monophasic, bipolar or monopolar. The stimulation current is created using a 7-bit digital-to-analog converter (IDAC) with a resolution of 1 μA per step. By using a switch matrix and a dedicated high-voltage output driver for each electrode, it is possible to stimulate any number and combination of electrodes. Thus, any 128-pixel image could be mapped to phosphenes. The ASIC accepts SPI commands and converts them into stimulation current waveforms destined for the brain, that will create phosphenes within the visual field. This conversion is performed by a digital control block optimized for low latency to enable at least 25 FPS video.

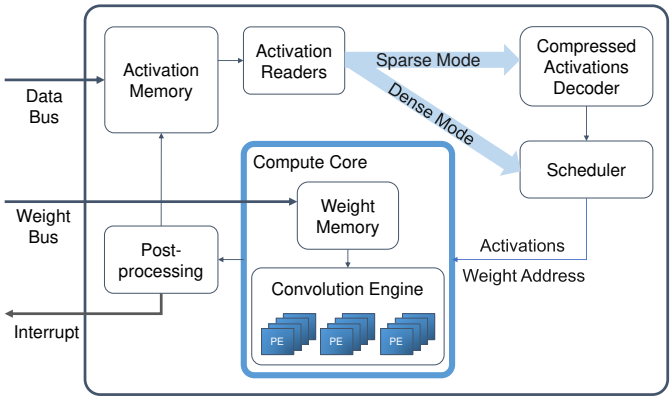

Fig. 2: VPDNN accelerator architecture.

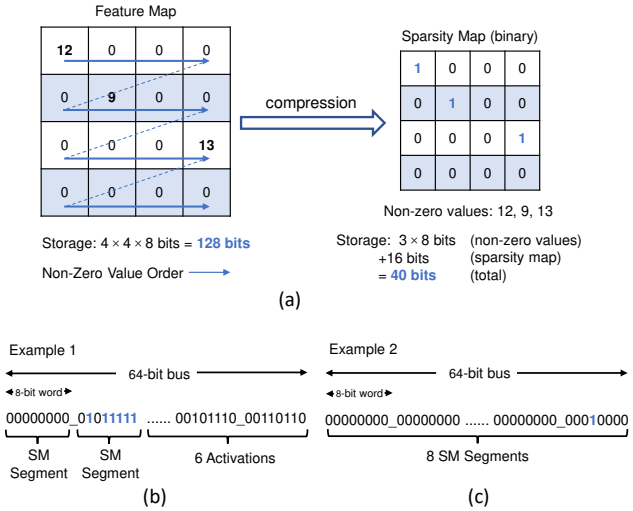

Fig. 3: (a) Compression scheme for a single feature map. A sparsity map (SM) encodes indices of non-zero entries in the feature map. (b) In case of a zero SM segment, no activation values are sent. The next non-zero SM segment has six '1' entries, thereby 6 activation values follow thereafter. (c) No activations are sent until there is first a non-zero SM segment.

C. Neural Network Accelerator Design

Figure 2 shows the block diagram of the VPDNN hardware accelerator implemented on the FPGA Zybo Z7-7020. The accelerator communicates with the host processor using three interfaces: the data bus, the weight bus and the interrupt bus. The weight bus is used to transfer weights between the host processor and the weight memory while the data bus is used to transfer feature map activations between the host processor and the activation memory. Both memory types are implemented using BRAM. The accelerator acts as a slave, such that the host processor initiates the write and read transactions. On the FPGA Zybo, the data bus and the weight bus follow the AXI4 specification[1], each with a bus of 64 bits.

Activations can be written or read in a compressed or uncompressed format. Activation compression is used to reduce data transfers on the data bus. The compression scheme takes advantage of the presence of zero-valued activations, usually introduced as a result of using the ReLU activation function. The uncompressed format is used when the layer to be processed is not preceded by a ReLU activation. The compression scheme shown in Fig. 3 is based on a previous scheme [14]. We optimize the sparsity map encoding for use

[1]https://documentation-service.arm.com/static/5f915b62f86e16515cdc3b1c

with multiple feature maps. The first step is to create two data structures; the sparsity map (SM) and the non-zero value list (NZVL). Each byte of the SM encodes the sparsity value of up to 8 layers of feature maps. A value of '1' represents a non-zero activation in the corresponding pixel of every feature map (see Fig. 3a). The NZVL is a list of only non-zero values in a specific preset order. The used order is shown in Fig. 3 for a one-channel activation. If the number of channels exceeds one, the channel dimension is traversed first.

The next step is to interleave SM segments and non-zero activation values in the corresponding activation segment, such that the SM encodes the number and location of non-zero activation values for the possible 8 feature maps. The compressed activation assumes the following order: The first word is an SM. If there are '1' values in the SM, the following word(s) represent the non-zero activations in the encoded segment. Then the next word is an SM, and so on. If the SM segment is all zeros, then the encoded segment does not have any non-zero activations, and the next word is the next SM. See the two examples in Fig. 3b&c. The activation memory consists of a number of line buffers, each line buffer has a dedicated activation reader for each row of the image. An activation reader controls the reading process of activations.

The accelerator has two operating modes: the *sparse* mode and *dense* mode. In the *Sparse* mode, compressed activations are sent to the accelerator. They are first decoded, sent to the scheduler and then to the compute core. In this mode, one saves computations and eliminates storage of zero activations. In the *Dense* mode, uncompressed activations are sent to the accelerator and the decoder is bypassed.

The scheduler prepares the received activations for processing. It also generates the corresponding weight memory addresses for the correct computation sequence. The compute unit consists of three processing element (PE) arrays where the data flows from one PE array to the next in systolic steps.

The outputs generated from the compute core are forwarded to the post-processing module which performs the following operations: Element-wise add (optional), quantization, and ReLU activation compression (optional). Output activations can either be compressed or uncompressed before they are stored in the activation memory.

III. Experimental Setup

A. Dataset

The *E2E-Phosphene DNN* is trained on a dataset of basic black and white images with randomly positioned lowercase alphabetic characters [23]. Each image has a resolution of 128×128, and contains a randomly selected letter from the alphabet displayed in one of 47 font types (38 font types for the training dataset and 9 font types for the validation dataset) - see Fig. 4, 1st column. There are 988 images in the training set and 234 images in the validation dataset.

The network is trained to output an embedding layer (2nd and 3rd columns of Fig. 4) that drives a decoder which outputs a reconstructed image of resolution 32×32 (4th and 5th) of the input image (1st column).

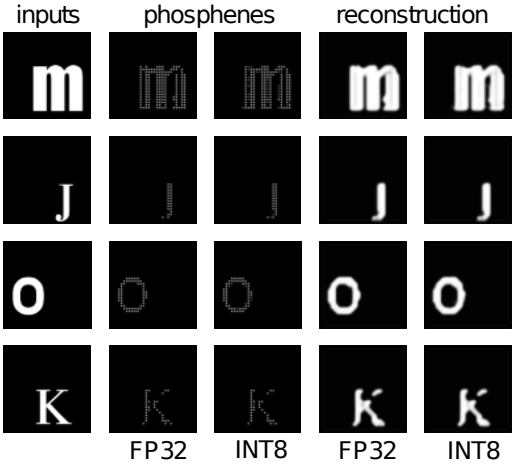

Fig. 4: Illustration of phosphene encoding and decoding results. 1st column: Five sample input images from the validation dataset. 2nd and 3rd columns: Output phosphene images of the trained model with 32-bit floating point (FP32) and 8-bit integer (INT8) respectively. 4th and 5th columns: Reconstructed images. Note that the phosphene output (32×32) is upscaled to the input resolution (128×128) in these columns.

B. Network Architecture and Modifications

The *E2E-Phosphene DNN* contains in total 14 convolutional layers. The layers produce 8, 16, 10×32, 16 and 1 output feature maps, respectively. The network uses only ReLU activation functions and convolutions with a stride of 2 for downsampling the feature maps in layers 1 and 3. All other convolutional layers use a stride of 1. The number of parameters is 94.5K. In addition, the network uses 8-bit precision weights and activations to match the bit precision on the FPGA while still maintaining comparable network accuracy. Post-Training Quantization (PTQ) was performed to reduce precision of the parameters from 32-bit floating point precision (FP32) to 8-bit integer (INT8). This bit precision reduction results in memory, area and latency reduction.

C. Training Procedure

All simulations were carried out using PyTorch 1.7.1 on a GTX1080 GPU. The encoder and decoder were trained end-to-end with batch size 16. The training was stopped early at epoch 14 after the validation loss had not improved for 50 consecutive training steps. The Adam [24] optimizer was used with learning rate of 0.001. The loss consisted of a term for the mean squared error (MSE) between original input and reconstruction, and an L_1 sparsity term to reduce the number of electrodes being activated. For the quantization experiments,

TABLE I: Evaluation of FP32 and INT8 quantized models.

Metric	**FP32**	**INT8**	Δ
MSE ($\times 10^{-6}$)	340	288	52
SSIM ($\times 10^{-3}$)	676.68	675.80	0.88
PSNR (dB)	20.09	20.12	-0.03

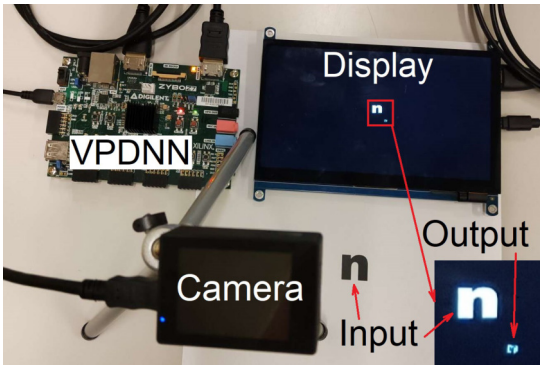

Fig. 5: HDMI camera attached to VPDNN accelerator captures the black character, 'n'. The inset at the bottom-right shows a zoomed-in view of the network input and output on the display.

activations were quantized to 8 bits in the range $[0, 255]$. and weights were quantized to 8 bits in the range $[-128, 127]$. Reconstruction performance was evaluated using MSE, the structural similarity index (SSIM) [25], and the peak signal-to-noise ratio (PSNR) between the reconstructed output and the input image. Table I summarizes the results.

D. FPGA Setup for Phosphene Stimulation Network

The VPDNN CNN accelerator was implemented on a Zybo Z7 board which features the Xilinx Zynq XC7Z020-1CLG400C device. The board has dedicated HDMI in and out ports. The design ran on a clock frequency of 100 MHz. The synthesized design uses 48 multipliers (16 PEs per array). The network output was downsampled to 32×32. Further FPGA utilization details are presented in Table II.

IV. Experimental Results

Fig. 5 shows a camera attached to the VPDNN accelerator whose network output is sent to a display monitor. The latencies for the different parts of the processing pipeline were profiled. The captured camera frames are first reformatted into the input data format required by the accelerator (latency=2.3 ms). The network inference took 12.0 ms. The network output is then reformatted for the display (latency=0.4 ms). The total latency is 14.7 ms.

The inference latency and power numbers of the VPDNN were compared against the numbers on the Jetson Nano as listed in Table III. The results show that the incremental power of running the network on Jetson Nano is at least $10\times$ higher than that of the VPDNN FPGA.

The stimulation patterns generated from the VPDNN network output are sent to the stimulation ASIC over SPI. The ASIC then converts them into output stimulation waveforms with low latency.

TABLE II: VPDNN FPGA resource utilization

Resource	Utilization	Available	Percentage
LUT	9,229	53,200	17.3%
FF	8,777	106,400	8.2%
BRAM	34.5	140	24.6%
DSP	56	220	25.4%

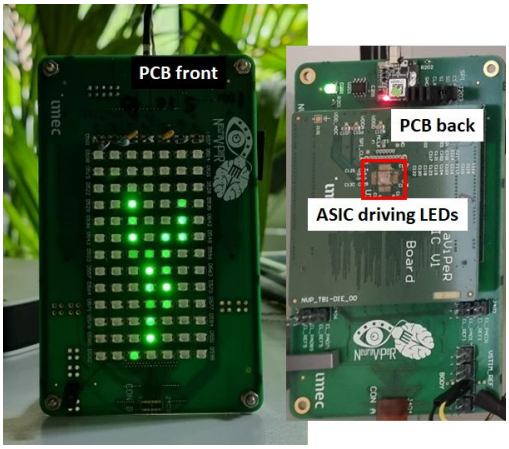

Fig. 6: Demonstration with the CNN driving the ASIC which drives a high efficiency LED matrix in place of electrodes.

TABLE III: Hardware inference latency and power numbers.

Platform	**Precision**	**Latency (ms)**	**FPS**	**Δ P (W)**
VPDNN	INT8	12.0	83.3	0.2
Nano-GPU	FP32	13.9	71.9	1.6
Nano-GPU	INT8	11.1	90.5	2.1

To show this functionality in a demonstrator, the electrodes are replaced with high-efficiency LEDs which are directly driven by the stimulation ASIC using the stimulation current, as shown in Fig. 6, where the letter Y is shown. The LEDs are placed in a 7×13 matrix and can produce an image similar to the one resulting from induced phosphenes, where the electrodes would be distributed across the visual cortex of the brain. The variable activation currents required by real electrodes are visible on the LED array as different intensities, even though the source image from the CNN is 2-color.

The ASIC supports a high-speed SPI interface which allows a full memory write in < 2 ms. However, because the ASIC includes double buffered memory, new parameters can be written while a stimulation event is ongoing. Therefore, while the ASIC adds to the total latency between image and stimulation, it does not produce a reduction in the overall frame rate.

V. Discussion and Conclusion

This work presents results from a real-time 128-channel visual stimulation system intended for a brain prosthesis. The VPDNN accelerator runs a phosphene stimulation network receiving input camera images. The combined VPDNN and camera system runs at approximately 71 FPS and only uses an incremental power of 0.2 W, $10\times$ lower than running the same setup on Jetson Nano. The VPDNN outputs are used by the stimulation ASIC to produce the currents needed by a demonstrator. Future work is planned for the use of this system in driving implantable electrodes [26] and using simultaneous recorded responses as feedback [27], [28].

REFERENCES

[1] M. S. Beauchamp, D. Oswalt, P. Sun, B. L. Foster, J. F. Magnotti, S. Niketeghad, N. Pouratian, W. H. Bosking, and D. Yoshor, "Dynamic stimulation of visual cortex produces form vision in sighted and blind humans," *Cell*, vol. 181, no. 4, pp. 774–783, 2020.

[2] E. Fernández, A. Alfaro, C. Soto-Sánchez, P. Gonzalez-Lopez, A. M. Lozano, S. Peña, M. D. Grima, A. Rodil, B. Gómez, X. Chen, P. R. Roelfsema, J. D. Rolston, T. S. Davis, and R. A. Normann, "Visual percepts evoked with an intracortical 96-channel microelectrode array inserted in human occipital cortex," *The Journal of Clinical Investigation*, vol. 131, 2021.

[3] C. M. Lopez, J. Putzeys, B. C. Raducanu, M. Ballini, S. Wang, A. Andrei, V. Rochus, R. Vandebriel, S. Severi, C. Van Hoof *et al.*, "A neural probe with up to 966 electrodes and up to 384 configurable channels in 0.13 μm SOI CMOS," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 3, pp. 510–522, 2017.

[4] M. Sanchez-Garcia, R. Martinez-Cantin, and J. J. Guerrero, "Semantic and structural image segmentation for prosthetic vision," *PloS ONE*, vol. 15, no. 1, p. e0227677, 2020.

[5] H. Wang, H. Mohamed, Z. Wang, B. Rueckauer, and S.-C. Liu, "LiteEdge: Lightweight semantic edge detection network," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, October 2021, pp. 2657–2666.

[6] A. Lozano, J. S. Suárez, C. Soto-Sánchez, J. Garrigós, J. J. Martínez-Alvarez, J. M. Ferrández, and E. Fernández, "Neurolight: A deep learning neural interface for cortical visual prostheses," *International Journal of Neural Systems*, vol. 30, no. 09, p. 2050045, 2020.

[7] L. Bai, Y. Zhao, and X. Huang, "A CNN accelerator on FPGA using depthwise separable convolution," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 10, pp. 1415–1419, 2018.

[8] X. Wei, C. H. Yu, P. Zhang, Y. Chen, Y. Wang, H. Hu, Y. Liang, and J. Cong, "Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs," in *Proceedings of the 54th Annual Design Automation Conference 2017*, 2017, pp. 1–6.

[9] R. Ghosh, A. Mishra, G. Orchard, and N. V. Thakor, "Real-time object recognition and orientation estimation using an event-based camera and CNN," in *2014 IEEE Biomedical Circuits and Systems Conference (BioCAS) Proceedings*, 2014, pp. 544–547.

[10] C. Farabet, C. Poulet, J. Y. Han, and Y. LeCun, "CNP: An FPGA-based processor for convolutional networks," in *2009 International Conference on Field Programmable Logic and Applications*, 2009, pp. 32–37.

[11] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2016.

[12] N. M. Philip and N. M. Sivamangai, "Review of FPGA-based accelerators of deep convolutional neural networks," in *2022 6th International Conference on Devices, Circuits and Systems (ICDCS)*, 2022, pp. 183–189.

[13] A. Ardakani, C. Condo, and W. J. Gross, "Fast and efficient convolutional accelerator for edge computing," *IEEE Transactions on Computers*, vol. 69, no. 1, pp. 138–152, 2020.

[14] A. Aimar, H. Mostafa, E. Calabrese, A. Rios-Navarro, R. Tapiador-Morales, I.-A. Lungu, M. B. Milde, F. Corradi, A. Linares-Barranco, S.-C. Liu *et al.*, "Nullhop: A flexible convolutional neural network accelerator based on sparse representations of feature maps," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 3, pp. 644–656, 2018.

[15] M. Liu, Y. He, and H. Jiao, "Efficient zero-activation-skipping for on-chip low-energy CNN acceleration," in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2021, pp. 1–4.

[16] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[17] J. de Ruyter van Steveninck, U. Güçlü, R. van Wezel, and M. van Gerven, "End-to-end optimization of prosthetic vision," *Journal of Vision*, vol. 22, no. 2, pp. 20–20, Feb. 2022.

[18] B. C. Raducanu, J. Aymerich, W.-Y. Hsu, P. Hendrickx, and C. M. Lopez, "A 128-channel neural stimulation and recording ASIC for scalable cortical visual prosthesis," in *Proceedings of the IEEE 49th European Solid-State Circuits Conference*, Sep 2023.

[19] N. Han, S. Srivastava, A. Xu, D. Klein, and M. Beyeler, "Deep learning–based scene simplification for bionic vision," in *AHs'21*. New York, NY, USA: Association for Computing Machinery, Feb. 2021, pp. 45–54.

[20] J. Granley, L. Relic, and M. Beyeler, "Hybrid neural autoencoders for stimulus encoding in visual and other sensory neuroprostheses," in *Advances in Neural Information Processing Systems*, Oct. 2022.

[21] B. Küçükoğlu, B. Rueckauer, N. Ahmad, J. d. R. van Steveninck, U. Güçlü, and M. van Gerven, "Optimization of neuroprosthetic vision via end-to-end deep reinforcement learning," *Int. J. Neural Syst.*, vol. 32, no. 11, p. 2250052, Nov. 2022.

[22] M. van der Grinten, J. d. R. van Steveninck, A. Lozano, L. Pijnacker, B. Rückauer, P. Roelfsema, M. van Gerven, R. van Wezel, U. Güçlü, and Y. Güçlütürk, "Biologically plausible phosphene simulation for the differentiable optimization of visual cortical prostheses," p. 2022.12.23.521749, Dec. 2022.

[23] J. de Ruyter van Steveninck, U. Güçlü, R. van Wezel, and M. van Gerven, "End-to-end optimization of prosthetic vision," *Journal of Vision*, vol. 22, no. 2, pp. 20–20, 2022.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[25] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[26] F. Grani, C. Soto-Sanchez, F. D. Farfan, A. Alfaro, M. D. Grima, A. R. Doblado, and E. Fernández, "Time stability and connectivity analysis with an intracortical 96-channel microelectrode array inserted in human visual cortex," *Journal of Neural Engineering*, vol. 19, no. 4, p. 045001, 2022.

[27] J. Hadorn, Z. Wang, B. Rueckauer, X. Chen, P. R. Roelfsema, and S.-C. Liu, "Fast temporal decoding from large-scale neural recordings in monkey visual cortex," in *SVRHM 2022 Workshop @ NeurIPS*, 2022. [Online]. Available: https://openreview.net/forum?id=fV8CfTjnXgH

[28] B. Rueckauer and M. van Gerven, "An in-silico framework for modeling optimal control of neural systems," *Frontiers in Neuroscience*, vol. 17, p. 1141884, 2023.