
A Scalable, Causal, and Energy Efficient Framework for Neural Decoding with Spiking Neural Networks

Georgios Mentzelopoulos^{1,*}, Ioannis Asmanis^{1,*}, Konrad P. Kording¹, Eva L. Dyer¹, Kostas Daniilidis^{1,2,†}, and Flavia Vitale^{1,†}

¹University of Pennsylvania, ²Archimedes, Athena RC

Abstract

Brain-computer interfaces (BCIs) promise to enable vital functions, such as speech and prosthetic control, for individuals with neuromotor impairments. Central to their success are neural decoders, models that map neural activity to intended behavior. Current learning-based decoding approaches fall into two classes: simple, causal models that lack generalization, or complex, non-causal models that generalize and scale offline but struggle in real-time settings. Both face a common challenge, their reliance on power-hungry artificial neural network backbones, which makes integration into real-world, resource-limited systems difficult. Spiking neural networks (SNNs) offer a promising alternative. Because they operate causally (i.e. only on present and past inputs) these models are suitable for real-time use, and their low energy demands make them ideal for battery-constrained environments. To this end, we introduce *Spikachu: a scalable, causal, and energy-efficient neural decoding framework based on SNNs*. Our approach processes binned spikes directly by projecting them into a shared latent space, where spiking modules, adapted to the timing of the input, extract relevant features; these latent representations are then integrated and decoded to generate behavioral predictions. We evaluate our approach on 113 recording sessions from 6 non-human primates, totaling 43 hours of recordings. Our method outperforms causal baselines when trained on single sessions using between 2.26× and 418.81× less energy. Furthermore, we demonstrate that scaling up training to multiple sessions and subjects improves performance and enables few-shot transfer to unseen sessions, subjects, and tasks. Overall, Spikachu introduces a scalable, online-compatible neural decoding framework based on SNNs, whose performance is competitive relative to state-of-the-art models while consuming orders of magnitude less energy.

1 Introduction

Brain-computer interfaces (BCIs) are opening new frontiers in assistive technology, particularly for those affected by severe neuromotor disorders [1–10]. The implantation of miniaturized BCI devices in the brains of patients suffering from debilitating disorders, such as limb loss or ALS, enabled them to control computers and smartphones using only their thoughts, vastly improving their quality of life [11–13]. Latest advances have even restored speech-based communication in individuals who had lost the ability to speak by reconstructing their intended words directly from neural activity [14, 15].

*Equal contribution. The order of the co-first authors was determined by a coin flip and is fully interchangeable. Contact: {gment, asmanis}@seas.upenn.edu

†Equal contribution. Contact: {kostas@cis, vitalef@pennmedicine}.upenn.edu,
Project page and code: <https://spikachu-bci.github.io>

At the core of every BCI system lies a neural decoder, the software that maps neural signals to the user's intended actions, such as moving a cursor on a screen or controlling a prosthetic limb [16–19].

Deep learning has significantly advanced our ability to build powerful neural decoders [14, 15, 20, 21]. Unlike traditional approaches that rely on hand-crafted features, artificial neural networks (ANNs) can learn the mapping from neural activity to intended actions directly from data. Despite this progress, designing scalable, high-performance neural decoders suitable for integration into real-world BCI systems remains an open challenge with constraints along several axes. First, neural decoders need to be energy-efficient to operate within the tight power budgets of battery-constrained implantable devices [22–25]. Second, to enable online operation, models must be causal (i.e. rely only on present and past inputs) [10]. Third, models need to scale, since scaling up model complexity and training dataset size has been shown to boost performance in the neural decoding domain [20, 21, 26–28]. Finally, models should generalize to new subjects and tasks with minimal training examples to reduce the need for lengthy calibration sessions that hinder the practical deployment of BCIs [29].

While significant progress has been made along each of these axes, to our knowledge, no single framework excels across all of them simultaneously. Existing approaches tend to fall into two main categories, each with its own shortcomings. On one hand, simple models, often based on traditional architectures like multi-layer perceptrons (MLPs) or Gated Recurrent Units (GRUs), tend to perform well within individual experiments. These methods are typically causal but require homogeneous input structures, making them difficult to scale or generalize across subjects [30–34]. On the other hand, more sophisticated frameworks that can be trained across datasets have demonstrated strong performance and generalization, particularly at scale [20, 21, 26–28, 35]. However, their lack of causal processing and heavy computational demands challenges their applicability outside the research lab.

Spiking neural networks (SNNs) offer a promising alternative. Their inherent causality supports integration into online systems, and their low computational footprint makes them well-suited for battery-constrained environments. To this end, we introduce *Spikachu: a scalable, causal, and energy-efficient framework for multi-session, multi-subject neural decoding based on SNNs*. Our approach operates on binned spike trains, which are first projected into a latent space shared across sessions and subjects. Temporal features are then extracted from the latents using parallel spiking networks. To capture long-range dependencies, the extracted features are processed by spiking self-attention blocks. Finally, the enriched latents are integrated and mapped to behavioral predictions through another set of parallel spiking modules.

We evaluate our approach on 113 neural recording sessions from 6 non-human primates (NHPs) totaling more than 111M spikes and 43 hours of recordings [36, 37]. Our approach outperforms causal baselines while consuming between 2.26× and 418.81× less energy when trained on single sessions. We then build unified models trained on multi-subject data that outperform single-session models, and show they can be transferred to new sessions, subjects, and tasks very efficiently. Overall, this work introduces a scalable, online-compatible neural decoding framework based on SNNs, whose performance is competitive relative to state-of-the-art models while consuming orders of magnitude less energy. This combination of performance and efficiency makes it a promising foundation for BCIs designed for edge computing environments.

Our contributions can be summarized as:

- *A framework for multi-session, multi-subject neural decoding that is scalable, causal, and energy efficient.* By combining SNNs with transformers, our approach offers strong decoding performance while using minimal energy, positioning it as a compelling solution for real-time BCI applications.
- *A causal, architecture agnostic building block for multi-session, multi-subject model training.* We propose a novel formulation for mapping neural recordings from heterogeneous datasets into a shared latent space, enabling scalable training across sessions and subjects.
- *A fast, architecture agnostic building block for efficient processing across temporal resolutions.* We propose a novel module capable of extracting features at multiple, distinct temporal scales based on SNNs.
- *Pretrained models for neural decoding.* We trained a unified model on the combined data from 3 NHPs spanning 99 neural recording sessions, which is transferable to new sessions, subjects, and tasks. We will make the model and code publicly available.

2 Related Work

Neural decoding. Recent advances in deep learning have significantly improved neural decoding capabilities, with ANNs enabling impressive results across a range of applications, including cursor control, prosthetic control, typing, and speech decoding [10, 14, 15, 17–19, 34]. These works primarily relied on lightweight, causal architectures such as MLPs and RNNs, which performed well on single-session datasets [38]. Relatively more sophisticated approaches based on VAEs and self-supervision have also been used with success by Liu et al. [39] and Peterson et al. [40], respectively. However, their performance degrades on new sessions or subjects due to their reliance on homogeneous input structures and known electrode correspondences across sessions and subjects.

Other studies have focused on improving cross-session generalization. LFADS was among the first to model latent dynamics across sessions and subjects [35], with extensions for large-scale training [41]. CEBRA introduced a contrastive learning approach for learning neural representations that are shared across subjects [21], while POYO used transformers to achieve multi-session, multi-subject, and multi-task generalization [26]. POYO+ extended those results across distinct cell types and brain regions [20]. The MICrONS foundation model demonstrated that large-scale pretraining on the mouse visual cortex can yield neural representations that generalize across visual stimuli [28]. While these approaches perform impressively and generalize well, they often sacrifice causality and introduce significant computational overhead, which is unsuitable for online applications such as battery-constrained BCIs.

Spiking neural networks. SNNs are a class of neural networks that excel in processing event-driven data sequences with spikes [42]. Unlike conventional ANNs, which rely on static activations like ReLU, SNNs employ bio-inspired spiking mechanisms, with the Leaky Integrate-and-Fire (LIF) model being the most widely used (see App. A.1). Key to their success is their remarkably energy-efficient inference compared to ANNs, particularly when deployed on neuromorphic hardware [43–45]. Furthermore, their event-driven, online nature makes them well-suited for processing real-time or asynchronous data streams [46–49].

SNNs have been investigated for their potential in energy-efficient neural decoding [50–54]. However, prior efforts have primarily focused on shallow spiking MLPs, typically with four or fewer layers, trained and evaluated within the confines of single recording sessions. The main emphasis of these works has been on deployment feasibility on neuromorphic hardware. In contrast, our work introduces the first SNN-based neural decoding framework that not only leverages the inherent energy efficiency of spiking computation but also scales to large, multi-session, and multi-subject datasets, marking a significant step forward in the applicability of SNNs to real-world neural decoding challenges.

3 Methodology

SNNs offer remarkable energy efficiency across a variety of tasks in computer vision and natural language processing [42, 55, 56], particularly with asynchronous, event-based data [47]. Our work aims to leverage their advantages in the context of neural decoding.

3.1 Harmonizing the neural activity across sessions and subjects

BCI systems rely on microelectrode arrays (MEAs) to record neural activity at a high spatial and temporal resolution. Thanks to their small size and low impedance, individual MEA electrodes can detect extracellular action potentials from single neurons or small groups of neurons (referred to as “units”) in their immediate vicinity [57, 58]. These recordings are typically represented as spike trains, which abstract away the waveform of each action potential and encode only the precise times at which spikes occur.

The spikes recorded across the electrodes of a subject are, of course, not independent. Rather, they reflect the coordinated activity of neurons that are part of distributed brain networks [26]. A key challenge is interpreting these signals not as isolated events, but as components of a broader neural context [27]. This challenge becomes increasingly complex when integrating data across multiple sessions and subjects. Within a single subject, MEA recordings can drift over time [26, 32, 57, 59]. Across subjects, MEAs inevitably sample from distinct neurons, with no known correspondence between electrodes [60]. Consequently, our decoding approach must be capable of extracting

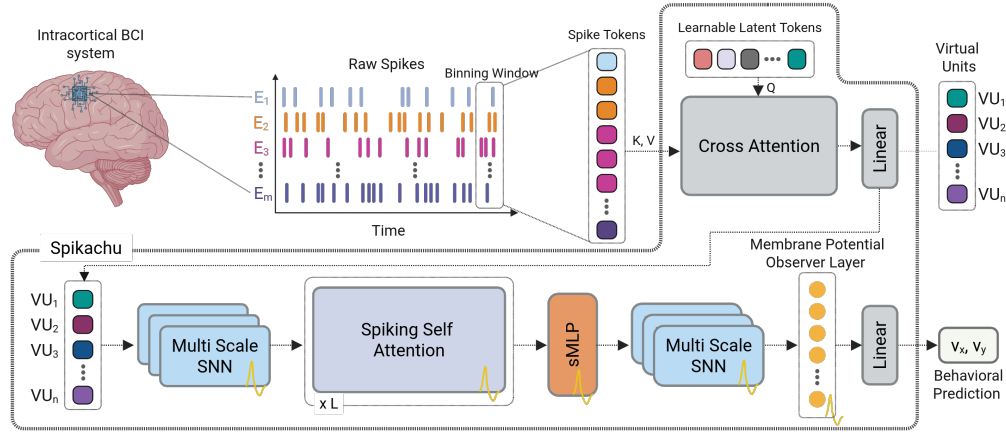


Figure 1: Overview of the Spikachu framework.

meaningful, generalizable representations despite differences in electrode placement, neural dynamics, and recorded populations across sessions and individuals.

Tokenization. Motivated by the heterogeneity of the neural recordings across sessions and subjects, we designed a tokenization scheme that enables multi-session training. At the same time, our approach operates within timepoints to enable online usability. Inspired by Azabou et al. [26], we represent each unit as a token via a learnable embedding in $\mathbb{R}^d$. Specifically, for a binned spike window, let $\mathcal{U} = \{u_1, u_2, u_3, \dots, u_{n_u}\}$ be the multiset of all units recorded during that window. Let $\text{UnitEmb}(\cdot)$ denote a lookup table that maps each unit to its embedding. We summarize the neural activity of a given time window as the sequence $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_{n_u}] \in \mathbb{R}^{n_u \times d}$, where $\mathbf{x}_i = \text{UnitEmb}(u_i) \forall i \in \{1, 2, \dots, n_u\}$. We note that if the i^{th} electrode recorded n_i units, n_i repeats of $\mathbf{x}_i = \text{UnitEmb}(u_i)$ would be present in the sequence $\mathbf{X}$ for that window. This sequence summarizes the cumulative neural activity across all electrodes of a subject for the given window.

Projecting units to a shared latent space. We then project the sequence $\mathbf{X}$ into a latent space that is shared across sessions and subjects using the Perceiver encoder [61, 62]. Specifically, let $\mathbf{Z}_0 = [\mathbf{z}_{0,1}, \mathbf{z}_{0,2}, \dots, \mathbf{z}_{0,n_0}] \in \mathbb{R}^{n_0 \times d_{z_0}}$ be a sequence of n_0 learnable latent tokens, where $\mathbf{z}_{0,i} \in \mathbb{R}^{d_{z_0}} \forall i \in \{1, 2, \dots, n_0\}$. We use cross-attention to project the input sequence $\mathbf{X}$ into the latent $\mathbf{Z}_1 \in \mathbb{R}^{n_o \times d}$ whose length n_o is independent of the length of the input sequence $\mathbf{X}$. To do so, we linearly project the latent and input sequences into Queries: $\mathbf{Q} = \mathbf{W}_Q \mathbf{Z}_0$, Keys: $\mathbf{K} = \mathbf{W}_K \mathbf{X}$, and Values: $\mathbf{V} = \mathbf{W}_V \mathbf{X}$, and compute,

$$\mathbf{Z}_1 \leftarrow \text{Cross-Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax} \left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}} \right) \mathbf{V}, \quad (1)$$

where d_k refers to the feature dimensionality of $\mathbf{K}$. For this operation, we use the standard transformer block preceded by layer-normalization and followed by a feed-forward network [63].

Obtaining virtual units. We then unroll the latent $\mathbf{Z}_1 \in \mathbb{R}^{n_o \times d} \rightarrow \mathbb{R}^{n_o \cdot d}$ and we project it to the low-dimensional latent $\mathbf{Z}_2 \in \mathbb{R}^{n_v}$ using a linear layer. We interpret $\mathbf{Z}_2$ as a new set of n_v “virtual” units that are shared across sessions and subjects. We note that the weights of this layer are shared across sessions and subjects—no session or subject specific adaptation is performed.

3.2 Efficient and stateful processing of the latents using spiking neural networks

Having projected the neural activity for each time bin into a new set of virtual units that is shared across sessions and subjects, we could in principle process the latent $\mathbf{Z}_2$ with any causal architecture commonly used for neural decoding such as MLP, GRU, or any model surveyed by Glaser et al. [38].

Rather than relying on such conventional power-hungry ANNs, we instead leverage SNNs which are considered the third generation of neural network models [64]. SNNs are theoretically as expressive

as ANNs and, due to their recurrent nature, are particularly well-suited for time-dependent tasks [65]. While their performance advantages remain an open area of research, their energy efficiency when deployed on neuromorphic hardware has been well demonstrated [44, 45]. This makes SNNs particularly attractive for BCI applications, where power constraints are critical due to the limited battery life of the miniaturized BCI implants.

Decomposing the neural activity across temporal streams. Given that the brain operates across multiple intrinsic timescales [66], we sought to process the data in a similarly multi-timescale fashion. To do this, we pass the latent $\mathbf{Z}_2$ through p parallel spiking feed forward networks (FFNs), each designed to operate at a distinct temporal resolution. Each network returns a latent $\mathbf{Z}_{s,i} \in \mathbb{R}^{d_s}$ by projecting $\mathbf{Z}_2$ through sequences of the following layers,

$$\mathbf{Z}_{l+1} = \text{BN}(\mathbf{W}_l \mathcal{SN}_l(\mathbf{Z}_l)), \quad (2)$$

where $\mathcal{SN}_l$ represents a spiking activation, $\mathbf{W}_l$ a linear projection, and BN the batch normalization operation [67]. Each network's spiking activation layers $\mathcal{SN}$ are initialized with independent and learnable decay constants (see App. A.1), effectively allowing each network to extract features that evolve as separate streams of information. We concatenate the latents from each stream into the sequence of latent tokens $\mathbf{Z}_{\text{ms}} = [\mathbf{Z}_{s,1}, \mathbf{Z}_{s,2}, \dots, \mathbf{Z}_{s,p}] \in \mathbb{R}^{p \times d_s}$.

Capturing long-range temporal dependencies. We then process the sequence $\mathbf{Z}_{\text{ms}}$ for informational dependencies across timescales using spiking-self attention [55]. Specifically, we project the sequence $\mathbf{Z}_{\text{ms}}$ into equally shaped Queries ($\mathbf{Q}$), Keys ($\mathbf{K}$), and Values ($\mathbf{V}$),

$$\mathbf{Q} = \mathcal{SN}_Q(\text{BN}(\mathbf{W}_Q \mathbf{Z}_{\text{ms}})), \quad \mathbf{K} = \mathcal{SN}_K(\text{BN}(\mathbf{W}_K \mathbf{Z}_{\text{ms}})), \quad \mathbf{V} = \mathcal{SN}_V(\text{BN}(\mathbf{W}_V \mathbf{Z}_{\text{ms}})), \quad (3)$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{d_{\text{ssa}}}$ and compute the spiking self attention matrix as,

$$\text{SSA}' = \mathcal{SN}(\mathbf{Q} \mathbf{K}^\top \mathbf{V} \cdot s), \quad (4)$$

where s is a scaling factor. We then compute $\mathbf{Z}_{\text{ssa}} \in \mathbb{R}^{p \times d_{\text{ssa}}}$ by projecting SSA' through the following spiking layers,

$$\mathbf{Z}_{\text{ssa}} \leftarrow \text{SSA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathcal{SN}(\text{BN}(\mathbf{W}_{\text{ssa}} \text{SSA}')). \quad (5)$$

Following Zhou et al. [55], we precede this operation using layer normalization and follow it with a spiking feed-forward network. For a primer on the main differences between spiking [55] and vanilla [63] self-attention, please see App. A.2.

Compressing the latents to a compact spatiotemporal representation. Having identified long-term dependencies across the p temporal streams, we unroll the latents $\mathbf{Z}_{\text{ssa}} \in \mathbb{R}^{p \times d_{\text{ssa}}} \rightarrow \mathbb{R}^{p \cdot d_{\text{ssa}}}$ and using a spiking MLP (whose layers follow Eq. 2), we project them to a low dimensional representation $\mathbf{Z}_{\text{mlp}} \in \mathbb{R}^{d_{\text{mlp}}}$ where $d_{\text{mlp}} \ll p \cdot d_{\text{ssa}}$. The latent $\mathbf{Z}_{\text{mlp}}$ is a compact representation of the neural activity that is informed by the evolution of the neural code along the spatial (electrode) and temporal axes.

Smoothing predictions using multiple timescales. Having extracted neural representations that capture long-range spatiotemporal dependencies, we process the latent $\mathbf{Z}_{\text{mlp}}$ with another set of p' parallel spiking MLPs that process the latents at different temporal scales (composed of layers described in Eq. 2). This time, we use the parallel MLPs to extract latents $\mathbf{Z}_{s',i} \in \mathbb{R}^{d_{s'}}$, each evolving at unique temporal scales based on the decay constant of the neurons in that network. We then concatenate the latents to obtain the sequence $\mathbf{Z}_{\text{sm}'} \in \mathbb{R}^{p' \times d_{s'}}$.

Tracking continuous variables through the membrane potential of spiking neurons. In many BCI applications, the behavioral variables of interest, such as the velocity of a computer cursor or the movement of a prosthetic limb, are continuous in nature. To accommodate this, our framework needs to map the spiking activity generated by the SNN layers to continuous-valued outputs. We achieve this using a membrane potential observer layer, denoted $\overline{\mathcal{SN}}_{\text{obs}}$, that never spikes and whose membrane potential fluctuates over time without resetting [52, 68, 69]. We first unroll the latents $\mathbf{Z}_{\text{sm}'} \in \mathbb{R}^{p' \times d_{s'}} \rightarrow \mathbb{R}^{p' \cdot d_{s'}}$ and then pass them through the observer layer,

$$\mathbf{M}_{\text{obs}} \leftarrow \overline{\mathcal{SN}}_{\text{obs}}(\mathbf{Z}_{\text{sm}'}), \quad (6)$$

where $\mathbf{M}_{\text{obs}} \in \mathbb{R}^{p' \cdot d_{s'}}$ denotes the membrane potential of the observer neurons. These accumulated latents are then linearly projected to the target variable's output space producing $\mathbf{Z}_{\text{out}} \in \mathbb{R}^{d_{\text{out}}}$, the final continuous behavioral prediction of the network.

4 Experiments

In this section, we validate the promise of our approach (see Fig. 1) for causal, energy-efficient and scalable neural decoding based on spiking neural networks.

4.1 Experimental setup

Dataset. We utilized two publicly available electrophysiology datasets summarized in Tab. 1. Altogether, the data span 6 NHPs engaged in 4 distinct behavioral tasks, encompassing a total of 113 recording sessions [36, 37, 70–72]. In terms of scale, *the combined dataset contains over 43 hours of recordings, 10,000 units, 110 million spikes, and 20 million behavioral timepoints*, providing a rich and diverse foundation for evaluating the scalability of our approach.

Table 1: *Datasets used in this work. CO: Center-Out, RT: Random Target.*

Study	Regions	# Indiv	# Sess	# Units	# In	# Out	Tasks
Perich et al. [36]	M1, PMd	4	111	10,410	111.39M	20M	CO, RT
Pei et al. [37]	M1	2	2	312	5M	9M	RT, Maze

Behavioral tasks. Neural recordings were collected from NHPs that performed motor tasks of various complexities (see Fig. 2A and App. B). In the CO task, the animal executes a relatively structured sequence: after receiving a go cue, it reaches toward one of eight predefined targets before returning to the center. The RT task presents additional complexity. The animal engages in continuous, self-paced movements, with new targets appearing unpredictably across the workspace. *Our goal was to decode the velocity of the cursor controlled by the animal from their neural activity.*

Design choices. Throughout all experiments, we bin spikes using a 0.01 sec sliding window on 1 sec segments of data. We do not use the trial structure during model training. We report the decoding performance for CO tasks during reaching movements only, as established in Azabou et al. [26]. Spikachu’s implementation and training details are described in App. C and D.

Energy estimation. All energy-related results in this work are derived from the number of floating-point operations (FLOPs) required for model inference, which we convert to energy consumption estimates following well-established procedures described in Bal and Sengupta [56] and Zhu et al. [49]. *Because FLOPs are hardware-agnostic, our analyses do not depend on any specific hardware implementation.* All details of the energy calculations are provided in App. E.2.

4.2 Performance on single sessions

We began by evaluating the performance of our modeling approach in a single-session setting. Specifically, we trained individual models on 99 recording sessions from three animals (NHPs C, J, and M) from the Perich et al. [36] dataset. Each session ranged from 10 to 106 minutes in duration, comprising 78 sessions of the CO task and 21 sessions of the RT task. These models achieved an *average per-session R^2 of 0.84 and 0.68 for the CO and RT tasks*, respectively (see Fig. 2B). We also estimated the mean energy required per inference across the single-session models (see App. E.2 for details) and found it to be $5.14\mu J$ and $5.13\mu J$ for the CO task and RT tasks, respectively.

Baseline comparisons. We then benchmarked the single session models against other architectures commonly used for neural decoding, such as MLP, GRU, and POYO [26, 38]. Our approach outperformed all causal baselines (see Fig. 2C and Tab. 2), while narrowing the performance gap with non-causal models. *Notably, Spikachu was the most energy-efficient model, requiring $2.26\times$ less energy per inference when compared to GRU (the second most efficient model) and $418.81\times$ less when*

Table 2: *Model performance for Spikachu and baselines. Best performing model is in bold and second best model is underlined.*

Model	Decod. Perf. (R^2) $\uparrow$		Energy (μJ) $\downarrow$	
	CO	RT	CO	RT
LSTM	0.4935	0.4214	15.08	14.94
MLP	0.7424	0.5724	12.18	12.06
POYO-causal	0.7961	0.5629	2151.65	2136.82
GRU	0.8336	0.6681	<u>11.65</u>	<u>11.54</u>
POYO	0.8937	0.6785	2151.65	2136.82
Spikachu	<u>0.8398</u>	<u>0.6761</u>	5.14	5.13

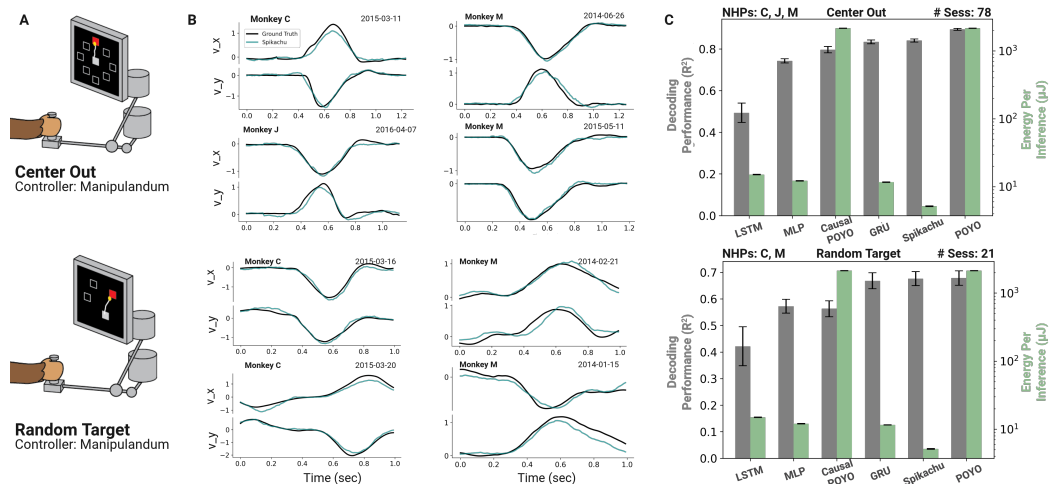


Figure 2: *Model performance on single sessions.* (A) Schematic of CO and RT tasks (reproduced from Azabou et al. [26], with permission). (B) Examples of true and predicted behavior (x, y velocities) for the CO (top 4 subplots) and RT (bottom 4 subplots) tasks. (C) Mean ($\pm$ SEM) decoding performance and energy consumption for Spikachu and baselines on sessions from monkeys C, J, and M from the Perich et al. [36] dataset.

compared to POYO, which was the best performing model. Importantly, the same trend was observed when comparing the number of FLOPs required for inference for each model (see Tab. 5, App. F.5). These findings highlight the dual benefits of our approach: high-performance neural decoding and improved energy efficiency, underscoring its suitability for power-constrained applications such as implantable BCIs.

We also benchmarked our model against baselines in terms of memory access costs (where Spikachu outperformed all baseline models). We refer the reader to App. F.5 for this analysis. Detailed implementation descriptions of the baseline models and the corresponding energy calculations are provided in App. C.2 and App. E.1, respectively.

4.3 Spikachu-mp: Pretraining on large amounts of data

After demonstrating Spikachu's strong performance when trained on individual sessions, we were interested in investigating whether training on more data, despite the heterogeneity, could further enhance Spikachu's performance, a strategy that has shown benefits in prior works [21, 26–28]. To this end, we developed Spikachu-mp: a multi-session, multi-subject model trained on the complete set of 99 recording sessions from monkeys C, J, and M from the Perich et al. [36] dataset. *Notably, this model was trained on over 40 hours of neural recordings and more than 110 million spikes.* In contrast to previous works (see Azabou et al. [26]), we did not increase our model's size during multi-subject training. This design choice ensured that our model would remain energy-efficient even when scaling to larger datasets, a critical consideration for resource-constrained implantable BCIs. Spikachu-mp achieved a per-session test set R^2 of 0.80 for CO and 0.57 for the RT task, indicating Spikachu's ability to learn generalized neural representations shared across sessions and subjects.

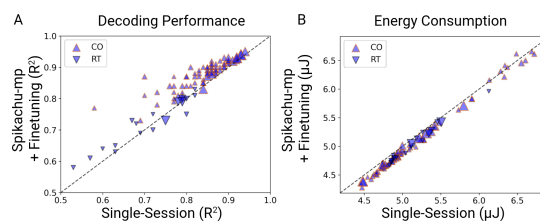


Figure 3: *Head-to-head comparison between Spikachu-mp + finetuning vs Spikachu trained on single-sessions.* (A) Decoding Performance, (B) Energy consumption per inference. Marker size encodes the number of trials available for each session.

To evaluate the utility of the learned representations, we then finetuned Spikachu-mp on individual sessions. The finetuned models achieved an average test set R^2 of 0.88 and 0.69 for the CO and RT tasks, respectively, outperforming models trained from scratch on single sessions ($\Delta R^2_{\text{CO}} = 3.75\%$ and $\Delta R^2_{\text{RT}} = 1.71\%$; Fig. 3A). The finetuned models also showed increased energy efficiency compared to single-session models trained from scratch, consuming $5.04\mu J$ and $5.03\mu J$ per inference for the CO and RT tasks, respectively (see Fig. 3B). *This is equivalent to saving 1.91% and 2.01% energy per inference for the CO and RT tasks, respectively.* Together, these results suggest that leveraging cross-session representations not only improves decoding accuracy but also enhances energy efficiency compared to training single-session models from scratch.

4.4 Transferring Spikachu-mp to new subjects

Having established that pretraining Spikachu on large amounts of data enhances decoding performance and energy efficiency, we sought to determine whether the pretrained model could be effectively transferred to entirely new sessions from a previously unseen subject. To test this, we trained single-session models from scratch on 12 held-out recording sessions (6 CO and 6 RT) from a new animal (monkey T) drawn from Perich et al. [36]. These models achieved a mean test set R^2 of 0.76 and 0.66 for the CO and RT tasks, while consuming $4.97\mu J$ and $5.14\mu J$ per inference, respectively.

We then used Spikachu-mp to transfer the learned representations to these new sessions. The transferred models yielded improved decoding performance, with average per-session test set R^2 of 0.78 and 0.68 for the CO and RT tasks, respectively (Fig. 4A). In addition to higher decoding accuracy, the transferred models were also more energy-efficient, consuming $4.79\mu J$ and $4.95\mu J$ per inference (Fig. 4B). *This represents a reduction in energy consumption of 3.71% and 3.63% for the CO and RT tasks, respectively, relative to models trained from scratch.* Finally, we analyzed the training dynamics and observed that the transferred models converged (reached 90% of their maximum attained R^2) on average 3× and 4× faster than their scratch-trained counterparts for the CO and RT tasks, respectively (see Fig. 4C, D). These findings suggest that Spikachu-mp learned neural representations that generalize to new subjects, providing a powerful foundation for transfer learning in BCI. In practical terms, this means that new users could benefit from robust BCI performance with minimal calibration, reducing the burden of subject-specific training and enabling faster deployment in real-world clinical or assistive settings.

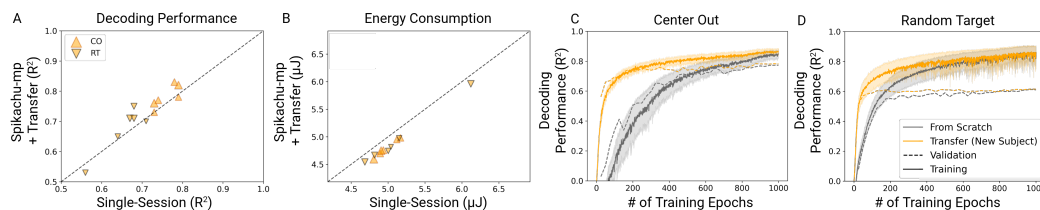


Figure 4: Performance comparison between single-session models trained from scratch vs single-session model transferred from Spikachu-mp. (A) Decoding Performance, (B) Energy consumption per inference, (C, D) Learning dynamics for the CO and RT tasks.

4.5 Scaling laws of multi-session, multi-subject training

Having shown that pretraining Spikachu learns robust neural representations that generalize across sessions and subjects, we next investigated its scaling behavior, specifically, how (1) decoding performance, and (2) energy efficiency change as a function of the amount of data used for pretraining. In addition to Spikachu-mp, we trained three multi-session, multi-subject models using 20, 49, and 75 sessions drawn from animals C, J, and M in Perich et al. [36], the same subjects used to train Spikachu-mp. To probe generalization and transfer capabilities, we used each pretrained model as the initialization for three finetuning conditions: (1) Seen sessions: finetuning on the same sessions used during pretraining, (2) New sessions: transferring to unseen sessions from the same subjects (NHPs C, J, M), (3) New subject: transferring to unseen sessions from a new subject (monkey T).

Decoding performance and energy consumption results for the CO and RT tasks are shown in Fig. 5A, B and Fig. 5C, D, respectively. For comparison, we overlaid the performance of single-session models trained from scratch (gray bars). Across all conditions, seen sessions, new sessions, and new

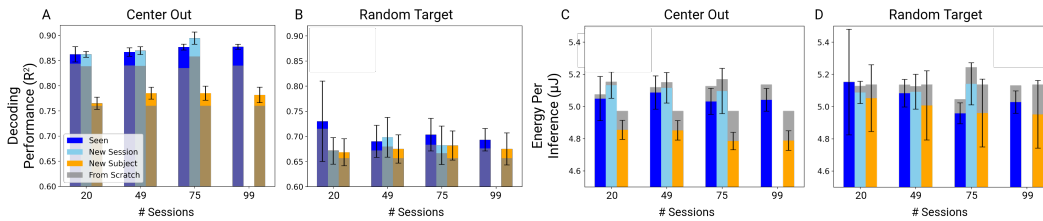


Figure 5: *Benefits of scaling up the training dataset size.* (A, B) Decoding performance of finetuned and transferred models (performance of scratch-trained single-session models overlaid in gray) as a function of the number of sessions used for model pretraining for the (A) CO, and (B) RT tasks. Panels (C, D) show the energy consumption per inference for finetuned and transferred models (performance of scratch-trained single-session models overlaid in gray) as a function of the number of sessions used for model pretraining for the (C) CO, and (D) RT tasks.

subjects, the pretrained models consistently outperformed their from-scratch counterparts. Moreover, *the performance gains from pretraining scaled positively with the number of sessions used during model pretraining* (as seen by the growing gap between colored and gray bars in Fig. 5A, B, and Fig. 8A, B). We observed a similar trend for energy consumption per inference, reported in Fig. 5C, D. Pretrained models required less energy across all transfer settings when compared to training from scratch, and *energy savings improved with the size of the pretraining dataset* (Fig. 8C, D). As a bonus, *we observed that all pretrained models converged much faster than models trained from scratch* (see App. F.1). Overall, these results show that scaling up pretraining yielded consistent improvements in both decoding accuracy and energy efficiency, with larger datasets providing greater gains and faster convergence.

For direct, head-to-head comparisons of decoding performance, energy efficiency, and convergence speed between pretrained and from-scratch models, refer to App. F.1 (Fig. 9 and 10).

4.6 Transferring Spikachu-mp to a new animal, setup, and task

In Sec. 4.4 and 4.5, we demonstrated Spikachu's ability to generalize across sessions and animals not encountered during pretraining. To further assess Spikachu's generalization capabilities, we evaluated the model on entirely novel conditions: neural recordings from a new animal performing a novel task under experimental settings that substantially differed from those described in Perich et al. [36]. Specifically, we applied our framework to the held-out MC-RTT and MC-Maze datasets from the Neural Latents Benchmark [37] (see Fig. 6A, C for schematics and App. B for task details).

We first trained single-session models from scratch. We used a batch size of 128 for the MC-RTT dataset and a batch size of 512 for the MC-Maze dataset (as this dataset is much larger). Spikachu achieved a test set R^2 and 0.57 on the RTT task and 0.79 on the Maze task, matching the performance of other state-of-the-art models (see Azabou et al. [26] for a comparison). Those models consumed 5.12 and 4.86 μJ per inference for the MC-RTT and MC-Maze tasks, respectively. We then used the pretrained weights of Spikachu-mp as a basis for transferring to the datasets. The transferred model achieved R^2 values of 0.56 for RTT and 0.78 for the Maze task while consuming 4.96 and 4.75 μJ per inference, respectively.

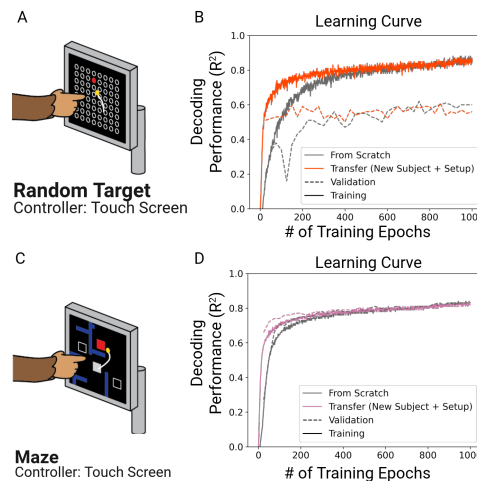


Figure 6: *Generalizing to new animals, setup, and tasks* (A, C) Schematic (reproduced from Azabou et al. [26], with permission) and (B, D) learning dynamics for the MC-RTT and MC-Maze tasks, respectively.

While this approach did not yield a significant performance gain, it led to markedly faster convergence ($2.33\times$ speedup for the MC-RTT and $2.66\times$ for MC-Maze; see Fig. 6B, D) and reduced energy consumption by 3.03% and 2.25% for the MC-RTT and MC-Maze tasks, respectively. This result is particularly noteworthy given the substantial difference in experimental conditions between the MC-RTT and MC-Maze datasets and those used in pretraining Spikachu-mp. Despite these domain shifts, the neural representations learned through pretraining proved transferable, highlighting Spikachu's ability to generalize across animals, recording setups, and tasks.

5 Discussion

In this work, we introduced Spikachu: a causal, scalable, and energy-efficient framework for multi-session, multi-subject neural decoding based on SNNs. Unlike existing approaches, it combines the expressivity of transformers with the low-power advantages of SNNs, resulting in a system that is both high-performing and energy-conscious. Our experiments demonstrate that Spikachu performs comparably to state-of-the-art neural decoding models while offering large energy savings. We also demonstrate that pretraining our model on a large corpus of heterogeneous neural data benefits neural decoding. To our knowledge, this is the first demonstration that unified models trained on multi-subject, multi-task neural data can yield improved performance in the spiking domain. Moreover, we find that pretraining encourages sparsity in network activity, further enhancing energy efficiency, a critical factor for scaling up training across heterogeneous datasets.

This work provides strong experimental evidence that SNNs are a practical and energy-efficient alternative to ANNs for neural decoding. While the majority of our architecture leverages SNNs to achieve substantial energy savings, it still relies on an ANN-based harmonization module to support large-scale training across datasets and subjects. Although this module represents a small fraction of the total model parameters (less than 15% of total parameters), it accounts for a disproportionately high share of the model's energy. Replacing it with a fully spiking alternative could yield additional energy savings. We believe that adaptations of spiking attention, such as those introduced by Zhou et al. [73] or Li et al. [74], offer promising pathways to develop a fully spiking, end-to-end architecture.

Although our results provide strong evidence for spiking-based neural decoding, real-world validation remains a critical next step. Implementing Spikachu on neuromorphic hardware and using it in online experiments would ultimately test its usability in the real world. Prior works have demonstrated the feasibility of simple online SNN decoders [50, 52–54] but scaling to more complex architectures remains a formidable challenge. Encouragingly, Spikachu's resource requirements in this work are modest (fewer than 4M synapses and roughly 10K neurons) and well within the capacity of modern neuromorphic chips such as the Loihi 2 [43] and Darwin 3 [75]. Importantly, the compact form factor of these chips is also well within the physical constraints of fully implantable BCI systems, which typically fit within the area of a U.S. quarter. We hope our work inspires collaboration between neuroscience and hardware communities to address implementation barriers.

Overall, this work represents a meaningful step forward in neuroscience by introducing an energy efficient neural decoding framework based on SNNs. We demonstrate that our framework delivers strong decoding performance that scales when trained on large, heterogeneous, multi-session datasets. This challenges the conventional notion that spiking networks must sacrifice performance for energy savings and highlights their potential for broader generalization [73]. Given that early demonstrations of BCI therapeutic interventions have already been shown [14, 15, 17–19], our work paves the way towards energy efficient neural decoding therapeutic interventions, bringing them closer to clinical translation.

Acknowledgments and Disclosure of Funding

We would like to thank the following funding sources for their generous support: National Institutes of Health grant R01NS121219 (FV); National Science Foundation CAREER award IIS-214607 (ELD); Onassis Foundation graduate student scholarship and A. G. Leventis Foundation graduate student scholarship (GM); Bodossaki Foundation graduate student scholarship (IA).

Authors declare no competing interests.

References

- [1] Henri Lorach, Andrea Galvez, Valeria Spagnolo, Felix Martel, Serpil Karakas, Nadine Interling, Molywan Vat, Olivier Faivre, Cathal Harte, Salif Komi, Jimmy Ravier, Thibault Collin, Laure Coquoz, Icare Sakr, Edeny Baaklini, Sergio Daniel Hernandez-Charpak, Gregory Dumont, Rik Buschman, Nicholas Buse, Tim Denison, Ilse van Nes, Leonie Asboth, Anne Watrin, Lucas Struber, Fabien Sauter-Starace, Lilia Langar, Vincent Auboiroux, Stefano Carda, Stephan Chabardes, Tetiana Aksenova, Robin Demesmaeker, Guillaume Charvet, Jocelyne Bloch, and Grégoire Courtine. Walking naturally after spinal cord injury using a brain–spine interface. *Nature*, 618(7963):126–133, May 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06094-5. URL <http://dx.doi.org/10.1038/s41586-023-06094-5>.
- [2] Eleanor J. Cole, Katy H. Stimpson, Brandon S. Bentzley, Merve Gulser, Kirsten Cherian, Claudia Tischler, Romina Nejad, Heather Pankow, Elizabeth Choi, Haley Aaron, Flint M. Espil, Jaspreet Pannu, Xiaoqian Xiao, Dalton Duvio, Hugh B. Solvason, Jessica Hawkins, Austin Guerra, Booil Jo, Kristin S. Raj, Angela L. Phillips, Fahim Barmak, James H. Bishop, John P. Coetzee, Charles DeBattista, Jennifer Keller, Alan F. Schatzberg, Keith D. Sudheimer, and Nolan R. Williams. Stanford accelerated intelligent neuromodulation therapy for treatment-resistant depression. *American Journal of Psychiatry*, 177(8):716–726, August 2020. doi: 10.1176/appi.ajp.2019.19070720. URL <https://doi.org/10.1176/appi.ajp.2019.19070720>.
- [3] Tyler Singer-Clark, Xianda Hou, Nicholas S Card, Maitreyee Wairagkar, Carrina Iacobacci, Hamza Peracha, Leigh R Hochberg, Sergey D Stavisky, and David M Brandman. Speech motor cortex enables bci cursor control and click. *Journal of Neural Engineering*, 22(3):036015, May 2025. ISSN 1741-2552. doi: 10.1088/1741-2552/add0e5. URL <http://dx.doi.org/10.1088/1741-2552/add0e5>.
- [4] Jiajie Jessica Xu, Lauren L. Zimmerman, Vanessa H. Soriano, Georgios Mentzelopoulos, Eric Kennedy, Elizabeth C. Bottorff, Chris Stephan, Kenneth Kozloff, Maureen J. Devlin, and Tim M. Bruns. Tibial nerve stimulation increases vaginal blood perfusion and bone mineral density and yield load in ovariectomized rat menopause model. *International Urogynecology Journal*, 33(12):3543–3553, March 2022. ISSN 1433-3023. doi: 10.1007/s00192-022-05125-5. URL <http://dx.doi.org/10.1007/s00192-022-05125-5>.
- [5] Lauren L. Zimmerman, Georgios Mentzelopoulos, Hannah Parrish, Vlad I. Marcu, Brandon D. Luma, Jill B. Becker, and Tim M. Bruns. Immediate and long-term effects of tibial nerve stimulation on the sexual behavior of female rats. *Neuromodulation: Technology at the Neural Interface*, 27(2):343–352, February 2024. ISSN 1094-7159. doi: 10.1016/j.neurom.2022.11.008. URL <http://dx.doi.org/10.1016/j.neurom.2022.11.008>.
- [6] Brian Kim, Brian A. Erickson, Guadalupe Fernandez-Nunez, Ryan Rich, Georgios Mentzelopoulos, Flavia Vitale, and John D. Medaglia. Eeg phase can be predicted with similar accuracy across cognitive states after accounting for power and signal-to-noise ratio. *eneuro*, 10(9):ENEURO.0050–23.2023, August 2023. ISSN 2373-2822. doi: 10.1523/eneuro.0050-23.2023. URL <http://dx.doi.org/10.1523/ENEURO.0050-23.2023>.
- [7] Brian Kim, Brian A Erickson, Guadalupe Fernandez-Nuñez, John D Medaglia, Georgios Mentzelopoulos, Ryan R Rich, and Flavia Vitale. A - 175 eeg phase can be predicted with similar accuracy across cognitive states after accounting for power and snr. *Archives of Clinical Neuropsychology*, 38(7):1347–1348, October 2023. ISSN 1873-5843. doi: 10.1093/arclin/acad067.192. URL <http://dx.doi.org/10.1093/arclin/acad067.192>.
- [8] Georgios Mentzelopoulos, Nicolette Driscoll, Sneha Shankar, Brian Kim, Ryan Rich, Guadalupe Fernandez-Nunez, Harrison Stoll, Brian Erickson, John Dominic Medaglia, and Flavia Vitale. Alerting attention is sufficient to induce a phase-dependent behavior that can be predicted by frontal eeg. *Frontiers in Behavioral Neuroscience*, 17, May 2023. ISSN 1662-5153. doi: 10.3389/fnbeh.2023.1176865. URL <http://dx.doi.org/10.3389/fnbeh.2023.1176865>.
- [9] Brian Erickson, Brian Kim, Philip Sabes, Ryan Rich, Abigail Hatcher, Guadalupe Fernandez-Nuñez, Georgios Mentzelopoulos, Flavia Vitale, and John Medaglia. Tms-induced phase resets depend on tms intensity and eeg phase. *Journal of Neural Engineering*, 21(5):056035, October 2024. ISSN 1741-2552. doi: 10.1088/1741-2552/ad7f87. URL <http://dx.doi.org/10.1088/1741-2552/ad7f87>.

- [10] Francis R. Willett, Donald T. Avansino, Leigh R. Hochberg, Jaimie M. Henderson, and Krishna V. Shenoy. High-performance brain-to-text communication via handwriting. *Nature*, 593(7858):249–254, May 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-03506-2. URL <http://dx.doi.org/10.1038/s41586-021-03506-2>.
- [11] Paul Nuyujukian, Jose Albites Sanabria, Jad Saab, Chethan Pandarinath, Beata Jarosiewicz, Christine H Blabe, Brian Franco, Stephen T Mernoff, Emad N Eskandar, John D Simeral, et al. Cortical control of a tablet computer by people with paralysis. *PloS one*, 13(11):e0204566, 2018.
- [12] Mark Hettick, Elton Ho, Adam J. Poole, Manuel Monge, Demetrios Papageorgiou, Kazutaka Takahashi, Morgan LaMarca, Daniel Trietsch, Kyle Reed, Mark Murphy, Stephanie Rider, Kate R. Gelman, Yoon Woo Byun, Timothy Hanson, Vanessa Tolosa, Sang-Ho Lee, Sanjay Bhatia, Peter E. Konrad, Michael Mager, Craig H. Mermel, and Benjamin I. Rapoport. The layer 7 cortical interface: A scalable and minimally invasive brain–computer interface platform. January 2022. doi: 10.1101/2022.01.02.474656. URL <http://dx.doi.org/10.1101/2022.01.02.474656>.
- [13] Thomas Oxley. Long-term safety of a fully implanted endovascular brain-computer interface for severe paralysis. *Archives of Physical Medicine and Rehabilitation*, 103(12):e53, December 2022. ISSN 0003-9993. doi: 10.1016/j.apmr.2022.08.562. URL <http://dx.doi.org/10.1016/j.apmr.2022.08.562>.
- [14] Francis R. Willett, Erin M. Kunz, Chaofei Fan, Donald T. Avansino, Guy H. Wilson, Eun Young Choi, Foram Kamdar, Matthew F. Glasser, Leigh R. Hochberg, Shaul Druckmann, Krishna V. Shenoy, and Jaimie M. Henderson. A high-performance speech neuroprosthesis. *Nature*, 620(7976):1031–1036, August 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06377-x. URL <http://dx.doi.org/10.1038/s41586-023-06377-x>.
- [15] Sean L. Metzger, Kaylo T. Littlejohn, Alexander B. Silva, David A. Moses, Margaret P. Seaton, Ran Wang, Maximilian E. Dougherty, Jessie R. Liu, Peter Wu, Michael A. Berger, Inga Zhuravleva, Adelyn Tu-Chan, Karunesh Ganguly, Gopala K. Anumanchipalli, and Edward F. Chang. A high-performance neuroprosthesis for speech decoding and avatar control. *Nature*, 620(7976):1037–1046, August 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06443-4. URL <http://dx.doi.org/10.1038/s41586-023-06443-4>.
- [16] Paul Sajda, Klaus-robert Muller, and Krishna Shenoy. Brain-computer interfaces [from the guest editors]. *IEEE Signal Processing Magazine*, 25(1):16–17, 2008. ISSN 1053-5888. doi: 10.1109/msp.2008.4408438. URL <http://dx.doi.org/10.1109/MSP.2008.4408438>.
- [17] Jennifer L Collinger, Brian Wodlinger, John E Downey, Wei Wang, Elizabeth C Tyler-Kabara, Douglas J Weber, Angus JC McMorland, Meel Velliste, Michael L Boninger, and Andrew B Schwartz. High-performance neuroprosthetic control by an individual with tetraplegia. *The Lancet*, 381(9866):557–564, February 2013. ISSN 0140-6736. doi: 10.1016/s0140-6736(12)61816-9. URL [http://dx.doi.org/10.1016/s0140-6736\(12\)61816-9](http://dx.doi.org/10.1016/s0140-6736(12)61816-9).
- [18] Sharlene N. Flesher, John E. Downey, Jeffrey M. Weiss, Christopher L. Hughes, Angelica J. Herrera, Elizabeth C. Tyler-Kabara, Michael L. Boninger, Jennifer L. Collinger, and Robert A. Gaunt. A brain-computer interface that evokes tactile sensations improves robotic arm control. *Science*, 372(6544):831–836, May 2021. ISSN 1095-9203. doi: 10.1126/science.abd0380. URL <http://dx.doi.org/10.1126/science.abd0380>.
- [19] Matthew S. Willsey, Nishal P. Shah, Donald T. Avansino, Nick V. Hahn, Ryan M. Jamiolkowski, Foram B. Kamdar, Leigh R. Hochberg, Francis R. Willett, and Jaimie M. Henderson. A high-performance brain–computer interface for finger decoding and quadcopter game control in an individual with paralysis. *Nature Medicine*, 31(1):96–104, January 2025. ISSN 1546-170X. doi: 10.1038/s41591-024-03341-8. URL <http://dx.doi.org/10.1038/s41591-024-03341-8>.
- [20] Mehdi Azabou, Krystal Xuejing Pan, Vinam Arora, Ian Jarratt Knight, Eva L Dyer, and Blake Aaron Richards. Multi-session, multi-task neural decoding from distinct cell-types and brain regions. In *The Thirteenth International Conference on Learning Representations*, 2025.

- [21] Steffen Schneider, Jin Hwa Lee, and Mackenzie Weygandt Mathis. Learnable latent embeddings for joint behavioural and neural analysis. *Nature*, 617(7960):360–368, May 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06031-6. URL <http://dx.doi.org/10.1038/s41586-023-06031-6>.
- [22] Atharva Sahasrabudhe, Laura E. Rupprecht, Sirma Orguc, Tural Khudiyev, Tomo Tanaka, Joanna Sands, Weikun Zhu, Anthony Tabet, Marie Manthey, Harrison Allen, Gabriel Loke, Marc-Joseph Antonini, Dekel Rosenfeld, Jimin Park, Indie C. Garwood, Wei Yan, Farnaz Niroui, Yoel Fink, Anantha Chandrakasan, Diego V. Bohórquez, and Polina Anikeeva. Multifunctional microelectronic fibers enable wireless modulation of gut and brain neural circuits. *Nature Biotechnology*, 42(6):892–904, June 2023. ISSN 1546-1696. doi: 10.1038/s41587-023-01833-5. URL <http://dx.doi.org/10.1038/s41587-023-01833-5>.
- [23] Nicolette Driscoll, Marc-Joseph Antonini, Taylor M. Cannon, Pema Maretich, Greatness Olaitan, Valerie Doan Phi Van, Keisuke Nagao, Atharva Sahasrabudhe, Emmanuel Vargas Paniagua, Ethan J. Frey, Ye Ji Kim, Sydney Hunt, Melissa Hummel, Sanju Mupparaju, Alan Jasanoff, B. Jill Venton, and Polina Anikeeva. Multifunctional neural probes enable bidirectional electrical, optical, and chemical recording and stimulation in vivo. *Advanced Materials*, November 2024. ISSN 1521-4095. doi: 10.1002/adma.202408154. URL <http://dx.doi.org/10.1002/adma.202408154>.
- [24] Brian Erickson, Ryan Rich, Sneha Shankar, Brian Kim, Nicolette Driscoll, Georgios Mentzelopoulos, Guadalupe Fernandez-Núñez, Flavia Vitale, and John D Medaglia. Evaluating and benchmarking the eeg signal quality of high-density, dry mxene-based electrode arrays against gelled ag/agcl electrodes. *Journal of Neural Engineering*, 21(1):016005, January 2024. ISSN 1741-2552. doi: 10.1088/1741-2552/ad141e. URL <http://dx.doi.org/10.1088/1741-2552/ad141e>.
- [25] Peter Mitchell, Sarah C. M. Lee, Peter E. Yoo, Andrew Morokoff, Rahul P. Sharma, Daryl L. Williams, Christopher MacIsaac, Mark E. Howard, Lou Irving, Ivan Vrljic, Cameron Williams, Steven Bush, Anna H. Balabanski, Katharine J. Drummond, Patricia Desmond, Douglas Weber, Timothy Denison, Susan Mathers, Terence J. O’Brien, J. Mocco, David B. Grayden, David S. Liebeskind, Nicholas L. Opie, Thomas J. Oxley, and Bruce C. V. Campbell. Assessment of safety of a fully implanted endovascular brain-computer interface for severe paralysis in 4 patients: The stentrod with thought-controlled digital switch (switch) study. *JAMA Neurology*, 80(3):270, March 2023. ISSN 2168-6149. doi: 10.1001/jamaneurol.2022.4847. URL <http://dx.doi.org/10.1001/jamaneurol.2022.4847>.
- [26] Mehdi Azabou, Vinam Arora, Venkataramana Ganesh, Ximeng Mao, Santosh Nachimuthu, Michael Mendelson, Blake Richards, Matthew Perich, Guillaume Lajoie, and Eva L. Dyer. A Unified, Scalable Framework for Neural Population Decoding. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- [27] Georgios Mentzelopoulos, Evangelos Chatzipantazis, Ashwin Ramayya, Michelle Hedlund, Vivek Buch, Kostas Daniilidis, Konrad Kording, and Flavia Vitale. Neural decoding from stereotactic eeg: accounting for electrode variability across subjects. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 108600–108624. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/c473b9c8897f50203fa23570687c6b30-Paper-Conference.pdf.
- [28] Eric Y. Wang, Paul G. Fahey, Zhuokun Ding, Stelios Papadopoulos, Kayla Ponder, Marissa A. Weis, Andersen Chang, Taliah Muhammad, Saumil Patel, Zhiwei Ding, Dat Tran, Jiakun Fu, Casey M. Schneider-Mizell, Nuno Maçarico da Costa, R. Clay Reid, Forrest Collman, Nuno Maçarico da Costa, Katrin Franke, Alexander S. Ecker, Jacob Reimer, Xaq Pitkow, Fabian H. Sinz, and Andreas S. Tolias. Foundation model of neural activity predicts response to new stimulus types. *Nature*, 640(8058):470–477, April 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-08829-y. URL <http://dx.doi.org/10.1038/s41586-025-08829-y>.
- [29] Konrad Kording. Neural decoding for bci: the objectives, principles and the future. In *2024 12th International Winter Conference on Brain-Computer Interface (BCI)*, page 1–2. IEEE,

- February 2024. doi: 10.1109/bci60775.2024.10480472. URL <http://dx.doi.org/10.1109/BCI60775.2024.10480472>.
- [30] Mijail D. Serruya, Nicholas G. Hatsopoulos, Liam Paninski, Matthew R. Fellows, and John P. Donoghue. Instant neural control of a movement signal. *Nature*, 416(6877):141–142, March 2002. ISSN 1476-4687. doi: 10.1038/416141a. URL <http://dx.doi.org/10.1038/416141a>.
 - [31] C. Ethier, E. R. Oby, M. J. Bauman, and L. E. Miller. Restoration of grasp following paralysis through brain-controlled stimulation of muscles. *Nature*, 485(7398):368–371, April 2012. ISSN 1476-4687. doi: 10.1038/nature10987. URL <http://dx.doi.org/10.1038/nature10987>.
 - [32] Joseph E. O’Doherty, Mikhail A. Lebedev, Peter J. Ifft, Katie Z. Zhuang, Solaiman Shokur, Hannes Bleuler, and Miguel A. L. Nicolelis. Active tactile exploration using a brain–machine–brain interface. *Nature*, 479(7372):228–231, October 2011. ISSN 1476-4687. doi: 10.1038/nature10489. URL <http://dx.doi.org/10.1038/nature10489>.
 - [33] Beata Jarosiewicz, Anish A. Sarma, Daniel Bacher, Nicolas Y. Masse, John D. Simeral, Brittany Sorice, Erin M. Oakley, Christine Blabe, Chethan Pandarinath, Vikash Gilja, Sydney S. Cash, Emad N. Eskandar, Gerhard Friehs, Jaimie M. Henderson, Krishna V. Shenoy, John P. Donoghue, and Leigh R. Hochberg. Virtual typing by people with tetraplegia using a self-calibrating intracortical brain-computer interface. *Science Translational Medicine*, 7(313), November 2015. ISSN 1946-6242. doi: 10.1126/scitranslmed.aac7328. URL <http://dx.doi.org/10.1126/scitranslmed.aac7328>.
 - [34] Francis R. Willett, Daniel R. Young, Brian A. Murphy, William D. Memberg, Christine H. Blabe, Chethan Pandarinath, Sergey D. Stavisky, Paymon Rezaei, Jad Saab, Benjamin L. Walter, Jennifer A. Sweet, Jonathan P. Miller, Jaimie M. Henderson, Krishna V. Shenoy, John D. Simeral, Beata Jarosiewicz, Leigh R. Hochberg, Robert F. Kirsch, and A. Bolu Ajiboye. Principled bci decoder design and parameter selection using a feedback control model. *Scientific Reports*, 9(1), June 2019. ISSN 2045-2322. doi: 10.1038/s41598-019-44166-7. URL <http://dx.doi.org/10.1038/s41598-019-44166-7>.
 - [35] David Sussillo, Rafal Jozefowicz, L. F. Abbott, and Chethan Pandarinath. Lfads - latent factor analysis via dynamical systems, 2016. URL <https://arxiv.org/abs/1608.06315>.
 - [36] Matthew G. Perich, Juan A. Gallego, and Lee E. Miller. A neural population mechanism for rapid learning. *Neuron*, 100(4):964–976.e7, November 2018. ISSN 0896-6273. doi: 10.1016/j.neuron.2018.09.030. URL <http://dx.doi.org/10.1016/j.neuron.2018.09.030>.
 - [37] Felix Pei, Joel Ye, David Zoltowski, Anqi Wu, Rameed H. Chowdhury, Hansem Sohn, Joseph E. O’Doherty, Krishna V. Shenoy, Matthew T. Kaufman, Mark Churchland, Mehrdad Jazayeri, Lee E. Miller, Jonathan Pillow, Il Memming Park, Eva L. Dyer, and Chethan Pandarinath. Neural latents benchmark ’21: Evaluating latent variable models of neural population activity, 2021. URL <https://arxiv.org/abs/2109.04463>.
 - [38] Joshua I. Glaser, Ari S. Benjamin, Rameed H. Chowdhury, Matthew G. Perich, Lee E. Miller, and Konrad P. Kording. Machine learning for neural decoding. *eneuro*, 7(4):ENEURO.0506–19.2020, July 2020. ISSN 2373-2822. doi: 10.1523/eneuro.0506-19.2020. URL <http://dx.doi.org/10.1523/ENEURO.0506-19.2020>.
 - [39] Ran Liu, Mehdi Azabou, Max Dabagia, Chi-Heng Lin, Mohammad Gheshlaghi Azar, Keith Hengen, Michal Valko, and Eva Dyer. Drop, swap, and generate: A self-supervised approach for generating neural activity. *Advances in neural information processing systems*, 34:10587–10599, 2021.
 - [40] Steven M Peterson, Rajesh PN Rao, and Bingni W Brunton. Learning neural decoders without labels using multiple data streams. *Journal of Neural Engineering*, 19(4):046032, 2022.
 - [41] Mohammad Reza Keshtkaran, Andrew R Sedler, Rameed H Chowdhury, Raghav Tandon, Diya Basrai, Sarah L Nguyen, Hansem Sohn, Mehrdad Jazayeri, Lee E Miller, and Chethan Pandarinath. A large-scale neural network training framework for generalized estimation of single-trial population dynamics. *Nature Methods*, pages 1–6, 2022.

- [42] Kai Malcolm and Josue Casco-Rodriguez. A comprehensive review of spiking neural networks: Interpretation, optimization, efficiency, and best practices, 2023. URL <https://arxiv.org/abs/2303.10780>.
- [43] Garrick Orchard, E. Paxon Frady, Daniel Ben Dayan Rubin, Sophia Sanborn, Sumit Bam Shrestha, Friedrich T. Sommer, and Mike Davies. Efficient neuromorphic signal processing with loihi 2. In *2021 IEEE Workshop on Signal Processing Systems (SiPS)*, pages 254–259, 2021. doi: 10.1109/SiPS52927.2021.00053.
- [44] Peter U. Diehl, Guido Zarrella, Andrew Cassidy, Bruno U. Pedroni, and Emre Neftci. Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware. In *2016 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8, 2016. doi: 10.1109/ICRC.2016.7738691.
- [45] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham China, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, January 2018. ISSN 1937-4143. doi: 10.1109/mm.2018.112130359. URL <http://dx.doi.org/10.1109/MM.2018.112130359>.
- [46] Chankyu Lee, Adarsh Kosta, Alex Zihao Zhu, Kenneth Chaney, Kostas Daniilidis, and Kaushik Roy. Spike-flownet: Event-based optical flow estimation with energy-efficient hybrid neural networks. *ArXiv*, abs/2003.06696, 2020. URL <https://api.semanticscholar.org/CorpusID:212725241>.
- [47] Federico Paredes-Vallés, Kirk Y. W. Scheper, and Guido C. H. E. de Croon. Unsupervised learning of a hierarchical spiking neural network for optical flow estimation: From events to global motion perception. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(8):2051–2064, 2020. doi: 10.1109/TPAMI.2019.2903179.
- [48] Yingfu Xu, Guangzhi Tang, Amirreza Yousefzadeh, Guido C.H.E. de Croon, and Manolis Sifalakis. Event-based optical flow on neuromorphic processor: Ann vs. snn comparison based on activation sparsification. *Neural Networks*, 188:107447, 2025. ISSN 0893-6080. doi: <https://doi.org/10.1016/j.neunet.2025.107447>. URL <https://www.sciencedirect.com/science/article/pii/S0893608025003260>.
- [49] Rui-Jie Zhu, Ziqing Wang, Leilani H. Gilpin, and Jason Eshraghian. Autonomous driving with spiking neural networks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=95VyH4VxN9>.
- [50] Julie Dethier, Paul Nuyujukian, Chris Eliasmith, Terry Stewart, Shauki A Ellassaad, Krishna V Shenoy, and Kwabena Boahen. A brain-machine interface operating with a real-time spiking neural network control algorithm. *Adv. Neural Inf. Process. Syst.*, 2011:2213–2221, 2011.
- [51] Julie Dethier, Paul Nuyujukian, Stephen I Ryu, Krishna V Shenoy, and Kwabena Boahen. Design and validation of a real-time spiking-neural-network decoder for brain-machine interfaces. *Journal of Neural Engineering*, 10(3):036008, April 2013. ISSN 1741-2552. doi: 10.1088/1741-2560/10/3/036008. URL <http://dx.doi.org/10.1088/1741-2560/10/3/036008>.
- [52] Jiawei Liao, Lars Widmer, Xiaying Wang, Alfio Di Mauro, Samuel R. Nason-Tomaszewski, Cynthia A. Chestek, Luca Benini, and Taekwang Jang. An energy-efficient spiking neural network for finger velocity decoding for implantable brain-machine interface. In *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, page 134–137. IEEE, June 2022. doi: 10.1109/aicas54282.2022.9869846. URL <http://dx.doi.org/10.1109/AICAS54282.2022.9869846>.
- [53] Gianluca Leone, Luigi Raffo, and Paolo Meloni. On-fpga spiking neural networks for end-to-end neural decoding. *IEEE Access*, 11:41387–41399, 2023. ISSN 2169-3536. doi: 10.1109/access.2023.3269598. URL <http://dx.doi.org/10.1109/ACCESS.2023.3269598>.

- [54] Elijah A Taeckens and Sahil Shah. A spiking neural network with continuous local learning for robust online brain machine interface. *Journal of Neural Engineering*, 20(6):066042, December 2023. ISSN 1741-2552. doi: 10.1088/1741-2552/ad1787. URL <http://dx.doi.org/10.1088/1741-2552/ad1787>.
- [55] Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng YAN, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=frE4fUwz_h.
- [56] Malyaban Bal and Abhronil Sengupta. Spikingbert: Distilling bert to train spiking language models using implicit differentiation. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i10.28975. URL <https://doi.org/10.1609/aaai.v38i10.28975>.
- [57] Marie Engelen J. Obien, Kosmas Deligkaris, Torsten Bullmann, Douglas J. Bakkum, and Urs Frey. Revealing neuronal function through microelectrode array recordings. *Frontiers in Neuroscience*, 8, January 2015. ISSN 1662-453X. doi: 10.3389/fnins.2014.00423. URL <http://dx.doi.org/10.3389/fnins.2014.00423>.
- [58] Nicolette Driscoll, Brian Erickson, Brendan B. Murphy, Andrew G. Richardson, Gregory Robbins, Nicholas V. Apollo, Georgios Mentzelopoulos, Tyler Mathis, Kanit Hantanasirisakul, Puneet Bagga, Sarah E. Gullbrand, Matthew Sergison, Ravinder Reddy, John A. Wolf, H. Isaac Chen, Timothy H. Lucas, Timothy R. Dillingham, Kathryn A. Davis, Yury Gogotsi, John D. Medaglia, and Flavia Vitale. Mxene-infused bioelectronic interfaces for multiscale electrophysiology and stimulation. *Science Translational Medicine*, 13(612):eabf8629, 2021. doi: 10.1126/scitranslmed.abf8629. URL <https://www.science.org/doi/abs/10.1126/scitranslmed.abf8629>.
- [59] Frank Hofmann and Hilmar Bading. Long term recordings with microelectrode arrays: Studies of transcription-dependent neuronal plasticity and axonal regeneration. *Journal of Physiology-Paris*, 99(2–3):125–132, March 2006. ISSN 0928-4257. doi: 10.1016/j.jphysparis.2005.12.005. URL <http://dx.doi.org/10.1016/j.jphysparis.2005.12.005>.
- [60] Max Dabagia, Konrad P Kording, and Eva L Dyer. Aligning latent representations of neural activity. *Nature Biomedical Engineering*, 7(4):337–343, 2023.
- [61] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General perception with iterative attention, 2021. URL <https://arxiv.org/abs/2103.03206>.
- [62] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver io: A general architecture for structured inputs & outputs, 2021. URL <https://arxiv.org/abs/2107.14795>.
- [63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL <https://arxiv.org/abs/1706.03762>.
- [64] Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, December 1997. ISSN 0893-6080. doi: 10.1016/S0893-6080(97)00011-7. URL [http://dx.doi.org/10.1016/S0893-6080\(97\)00011-7](http://dx.doi.org/10.1016/S0893-6080(97)00011-7).
- [65] Kwabena Boahen. A neuromorph’s prospectus. *Computing in Science & Engineering*, 19(2): 14–28, 2017. doi: 10.1109/MCSE.2017.33.
- [66] John D Murray, Alberto Bernacchia, David J Freedman, Ranulfo Romo, Jonathan D Wallis, Xinying Cai, Camillo Padoa-Schioppa, Tatiana Pasternak, Hyojung Seo, Daeyeol Lee, and Xiao-Jing Wang. A hierarchy of intrinsic timescales across primate cortex. *Nature Neuroscience*,

- 17(12):1661–1663, November 2014. ISSN 1546-1726. doi: 10.1038/nn.3862. URL <http://dx.doi.org/10.1038/nn.3862>.
- [67] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. URL <https://arxiv.org/abs/1502.03167>.
 - [68] Shoeb Shaikh, Rosa So, Tafadzwa Sibindi, Camilo Libedinsky, and Arindam Basu. Real-time closed loop neural decoding on a neuromorphic chip. In *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*, page 670–673. IEEE, March 2019. doi: 10.1109/ner.2019.8717122. URL <http://dx.doi.org/10.1109/NER.2019.8717122>.
 - [69] Jiawei Liao, Oscar Toomey, Xiaying Wang, Lars Widmer, Cynthia A. Chestek, Luca Benini, and Taekwang Jang. A Spiking Neural Network Decoder for Implantable Brain Machine Interfaces and its Sparsity-aware Deployment on RISC-V Microcontrollers, 2024. URL <https://arxiv.org/abs/2405.02146>.
 - [70] Matthew G Perich and Lee E Miller. Altered tuning in primary motor cortex does not account for behavioral adaptation during force field learning. *Experimental brain research*, 235(9): 2689–2704, 2017.
 - [71] Joshua I Glaser, Matthew G Perich, Pavan Ramkumar, Lee E Miller, and Konrad P Kording. Population coding of conditional probability distributions in dorsal premotor cortex. *Nature communications*, 9(1):1788, 2018.
 - [72] Juan A Gallego, Matthew G Perich, Raaed H Chowdhury, Sara A Solla, and Lee E Miller. Long-term stability of cortical population dynamics underlying consistent behavior. *Nature neuroscience*, 23(2):260–270, 2020.
 - [73] Zhaokun Zhou, Kaiwei Che, Wei Fang, Keyu Tian, Yuesheng Zhu, Shuicheng Yan, Yonghong Tian, and Li Yuan. Spikformer v2: Join the high accuracy club on imagenet with an snn ticket, 2024. URL <https://arxiv.org/abs/2401.02020>.
 - [74] Yudong Li, Yunlin Lei, and Xu Yang. Spikeformer: Training high-performance spiking neural network with transformer. *Neurocomputing*, 574:127279, March 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2024.127279. URL <http://dx.doi.org/10.1016/j.neucom.2024.127279>.
 - [75] De Ma, Xiaofei Jin, Shichun Sun, Yitao Li, Xundong Wu, Youneng Hu, Fangchao Yang, Huajin Tang, Xiaolei Zhu, Peng Lin, and Gang Pan. Darwin3: A large-scale neuromorphic chip with a novel isa and on-chip learning, 2023. URL <https://arxiv.org/abs/2312.17582>.
 - [76] Nalinda Kulathunga, Nishath Rajiv Ranasinghe, Daniel Vrinceanu, Zackary Kinsman, Lei Huang, and Yunjiao Wang. Effects of the nonlinearity in activation functions on the performance of deep learning models, 2020. URL <https://arxiv.org/abs/2010.07359>.
 - [77] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its applications to conduction and excitation in nerve. *Journal of Physiology*, page 500–544, 1952. doi: 10.1113/jphysiol.1952.sp004764. URL <https://doi.org/10.1113/jphysiol.1952.sp004764>.
 - [78] Michele Migliore, Dax A. Hoffman, Jeffrey C. Magee, and Daniel Johnston. Role of an a-type k⁺ conductance in the back-propagation of action potentials in the dendrites of hippocampal pyramidal neurons. *Journal of Computational Neuroscience*, 7:5–15, 1999. URL <https://api.semanticscholar.org/CorpusID:15804693>.
 - [79] Beatriz Herrera, Amirsaman Sajad, Geoffrey F. Woodman, Jeffrey D. Schall, and Jorge J. Riera. A minimal biophysical model of neocortical pyramidal cells: Implications for frontal cortex microcircuitry and field potential generation. *Journal of Neuroscience*, 40(44):8513–8529, 2020. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.0221-20.2020. URL <https://www.jneurosci.org/content/40/44/8513>.

- [80] CM Sherff and B Mulloney. Tests of the motor neuron model of the local pattern-generating circuits in the swimmeret system. *Journal of Neuroscience*, 16(8):2839–2859, 1996. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.16-08-02839.1996. URL <https://www.jneurosci.org/content/16/8/2839>.
- [81] Wei Fang, Zhaofei Yu, Yanqing Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2641–2651, 2020. URL <https://api.semanticscholar.org/CorpusID:226976144>.
- [82] Xingrun Xing, Boyan Gao, Zheng Liu, David A. Clifton, Shitao Xiao, Wanpeng Zhang, Li Du, Zheng Zhang, Guoqi Li, and Jiajun Zhang. SpikeLLM: Scaling up spiking neural network to large language models via saliency-based spiking. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=Zadn10HsHv>.
- [83] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://papers.nips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.
- [84] Wei Fang, Yanqi Chen, Jianhao Ding, Zhaofei Yu, Timothée Masquelier, Ding Chen, Liwei Huang, Huihui Zhou, Guoqi Li, and Yonghong Tian. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40), October 2023. ISSN 2375-2548. doi: 10.1126/sciadv.adi1480. URL <http://dx.doi.org/10.1126/sciadv.adi1480>.
- [85] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes, 2019. URL <https://arxiv.org/abs/1904.00962>.
- [86] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [87] Chethan Pandarinath, K Cora Ames, Abigail A Russo, Ali Farshchian, Lee E Miller, Eva L Dyer, and Jonathan C Kao. Latent factors and dynamics in motor cortex and their application to brain–machine interfaces. *Journal of Neuroscience*, 38(44):9390–9401, 2018.
- [88] Jason K. Eshraghian, Max Ward, Emre O. Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 111(9):1016–1054, 2023. doi: 10.1109/JPROC.2023.3308088.
- [89] Mark Horowitz. 1.1 computing’s energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14, 2014. doi: 10.1109/ISSCC.2014.6757323.
- [90] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014. URL <https://arxiv.org/abs/1412.3555>.
- [91] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition, 2014. URL <https://arxiv.org/abs/1402.1128>.
- [92] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, February 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL <http://dx.doi.org/10.1016/j.neucom.2023.127063>.

- [93] Qi Xu, Yaxin Li, Jiangrong Shen, Jian K. Liu, Huajin Tang, and Gang Pan. Constructing deep spiking neural networks from artificial neural networks with knowledge distillation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7886–7895, 2023. doi: 10.1109/CVPR52729.2023.00762.
- [94] Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Di He, and Zhouchen Lin. On-line training through time for spiking neural networks. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 20717–20730. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/82846e19e6d42ebfd4ace4361def29ae-Paper-Conference.pdf.
- [95] Yuhang Li, Tamar Geller, Youngeun Kim, and Priyadarshini Panda. SEENN: Towards temporal spiking early exit neural networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=mbaNOYQTw>.
- [96] J. von Neumann. First draft of a report on the edvac. *IEEE Annals of the History of Computing*, 15(4):27–75, 1993. doi: 10.1109/85.238389.

Appendix

A Concept Primers

A.1 Spiking neuron models

Overview of spiking model history. The simplest neuron model available for machine learning is the standard ANN neuron, which statelessly aggregates incoming signals with addition and then passes the result through a predetermined non-linearity [76]. Bio-inspired models retain more of the complexity of biological neurons at the cost of being more technically involved. Amongst the first to quantify such models, Hodgkin and Huxley [77] developed a multi-parameter model to describe squid axon neurons. Other works have provided detailed models to approximate different types of biological neurons. For instance, Migliore et al. [78] and Herrera et al. [79] modeled various aspects of pyramidal neurons, whereas Sherff and Mulloney [80] modeled motor neurons in the swimmeret system of crayfish.*

The Leaky Integrate-and-Fire neuron. Between the simplicity of ANN neurons and fully biological neurons lie several models that trade off between the two, retaining the statefulness of biological systems while greatly simplifying the computational specifics to a much more tractable form. These models, generally termed *spiking neurons*, include state variables that typically correspond to potentials of various ion channels in biological systems. They also contain logic that dictates the evolution of these variables over time, typically expressed as a differential equation, as well as the necessary conditions for firing output spikes. The most ubiquitous such model is the Leaky Integrate-and-Fire (LIF) model, which approximates a neuron with a spiking RC circuit. In this model, the tracked state is a real, scalar number corresponding to the membrane voltage of the neuron. The state $v(t)$ obeys its own recurrent dynamics, modeled as an exponential decay, to which inputs are added through integration. The differential equation for the potential with time-varying input current $i(t)$,

$$\tau \cdot \dot{v}(t) = -v(t) + R \cdot i(t), \quad (7)$$

can be solved analytically,

$$v(t) = e^{-t/\tau} \cdot \left(v_0 + \int_{t_0}^t e^{s/\tau} \cdot \frac{R \cdot i(s)}{\tau} ds \right), \quad (8)$$

where the parameter τ controls the decay rate and R corresponds to an input resistance. A LIF neuron's firing behavior is then a simple matter of thresholding,

$$s(t) = \mathbb{1}[v(t) > V_{th}], \quad v(t + \delta t) = \begin{cases} v(t) + \dot{v}(t) \cdot \delta t & , \text{ if } s(t) = 0 \\ V_{rs} & , \text{ if } s(t) = 1 \end{cases}, \quad (9)$$

where $s(t)$ is the spike output and $\mathbb{1}$ is the indicator function. If the membrane voltage is above the threshold V_{th} , there will be a binary spike in the output, and the state will be reset according to the reset logic; one way to implement this would be to set the membrane back to a resting potential value V_{rs} , typically zero. Recently, it has been shown that these parameters can be *learned* [81]. We utilize this property in the present work to specialize different network components for separate (implicit) portions of the input signal.

Other spiking neurons. Other neuron models have found application in SNNs. In Orchard et al. [43], Resonate-and-Fire (RF) spiking neurons were used. These neurons are very similar to LIF, except that their state is *complex*, meaning that excitation induces an oscillatory behavior. Spiking only occurs when the imaginary part of the membrane state is zero, and the output is not binary, but instead the instantaneous value of the real part of the state. It is eventually shown that the resulting behavior is mathematically similar to the Short-Time Fourier Transform (STFT). Similarly, Xing et al. [82] introduced a generalized version of the LIF neuron. With integral outputs, the Generalized Integrate-and-Fire (GIF) neuron supported the authors' saliency-based spiking large language models. These works showcase the different directions in which spiking neurons can develop.

*A collection of computational neuroscience models can be retrieved at: <https://modeldb.science/>.

A.2 Spiking vs vanilla self attention

In this work, we use spiking self-attention (SSA) introduced by Zhou et al. [55] (an extension of the vanilla self-attention (VSA) introduced by Vaswani et al. [63]) which was specifically designed for SNNs. Here, we provide a brief overview of the differences between the two mechanisms.

Vanilla self attention. Let an input sequence $X \in \mathbb{R}^{T \times N \times D}$, where T , N , D refer to the number of time steps, the number of tokens in the sequence, and the dimensionality of each token, respectively. VSA relies on three components,

$$\text{Queries: } \mathbf{Q} = \mathbf{W}_Q \mathbf{X}, \quad \text{Keys: } \mathbf{K} = \mathbf{W}_K \mathbf{X}, \quad \text{and Values: } \mathbf{V} = \mathbf{W}_V \mathbf{X}, \quad (10)$$

and is computed as,

$$\text{VSA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax} \left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}} \right) \mathbf{V}, \quad (11)$$

where d_k refers to the feature dimensionality of $\mathbf{K}$.

VSA can be directly used on spiking sequences. However, it is unsuitable because: (1) the float-point matrix multiplication of $\mathbf{Q}$, $\mathbf{K}$ and softmax function (which contains exponent calculation and division operation) do not comply with the calculation rules of SNNs, and (2) computational and space requirements for VSA scale quadratically with the length of the input sequence $\mathbf{X}$ which does not meet the efficient computational requirements of SNNs [55].

Spiking self attention. Zhou et al. [55] proposed SSA, an alternative to VSA which is more suitable for SNNs. In their formulation, the sequence $\mathbf{X} \in \mathbb{R}^{T \times N \times D}$ is projected into equally shaped Queries ($\mathbf{Q}$), Keys ($\mathbf{K}$), and Values ($\mathbf{V}$),

$$\mathbf{Q} = \mathcal{SN}_Q(\text{BN}(\mathbf{W}_Q \mathbf{X})), \quad \mathbf{K} = \mathcal{SN}_K(\text{BN}(\mathbf{W}_K \mathbf{X})), \quad \mathbf{V} = \mathcal{SN}_V(\text{BN}(\mathbf{W}_V \mathbf{X})), \quad (12)$$

and compute the spiking self-attention matrix as,

$$\text{SSA}' = \mathcal{SN}(\mathbf{Q}\mathbf{K}^\top \mathbf{V} \cdot s), \quad (13)$$

where s is a scaling factor. SSA is then computed via the following spiking layers,

$$\text{SSA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathcal{SN}(\text{BN}(\mathbf{W}_{\text{ssa}} \text{SSA}')). \quad (14)$$

This formulation is better suited for spiking sequences than traditional VSA for several reasons. First, SSA operates independently at each time step, aligning naturally with the temporal dynamics of SNNs. Unlike VSA, SSA removes the use of the softmax function to normalize the attention matrix. Instead, it directly computes the attention by multiplying the query, key, and value sequences without additional normalization. This simplification is more efficient as softmax introduces unnecessary computational complexity. Since the spiking neuron layers $\mathcal{SN}_Q$ and $\mathcal{SN}_K$ output binary spike trains, the resulting attention maps are inherently non-negative. This removes the need for softmax to enforce non-negativity, as the attention mechanism already emphasizes relevant features and suppresses irrelevant ones through sparse, event-driven computation.

B Datasets

To train and evaluate our framework, we leveraged a diverse collection of publicly available datasets spanning multiple NHPs and various behavioral paradigms.

B.1 Datasets used for training and validation

To develop our model, we leveraged a rich and diverse dataset comprising 99 unique recording sessions from three NHPs (monkeys C, J, and M) engaged in two distinct behavioral tasks. These recordings were collected across four foundational studies [36, 70–72], and were later curated into a unified dataset by Azabou et al. [26]. This comprehensive resource is publicly available through [Dandi](#) (used in this work) and via the [brainsets](#) platform.

Across all sessions in the dataset, each NHP was seated in a primate chair, and a custom two-dimensional planar manipulandum was used to control a cursor displayed on a computer screen. During each recording session, the NHP performed one of two structured motor tasks:

- *Center-out Task (CO)*: In this classic reaching task, the monkey initiated movement from a central target toward one of eight peripheral targets arranged uniformly around a circle with an 8 cm radius. After holding at the center target for a variable duration, an auditory “go” cue signaled the monkey to move to a designated outer target. This task is widely used to investigate neural processes underlying movement planning, preparation, and execution.
- *Random Target Task (RT)*: Similar in design to the CO task, the RT task involved targets that were randomly distributed across the workspace rather than arranged in a circular pattern. The monkey was instructed, via an auditory cue, to move sequentially between four randomly positioned targets, introducing greater variability in movement direction and distance.

The behavioral sampling rate used across all sessions within this dataset was 100 Hz. We used 70% of the data from within each session for training and 10% for validation; the remaining 20% was held-out for testing.

B.2 Datasets held out for testing

We reserved 20% of the data from each session described above for testing. In addition, we fully held out all sessions from Monkey T, comprising 6 CO and 6 RT sessions, to assess the model’s ability to generalize across subjects.

To assess the model’s ability to generalize to novel tasks, we also held-out two standardized datasets from the Neural Latents Benchmark [37], MC-Maze and MC-RTT, which are described in detail below. In both datasets, the animals (different for MC-Maze and MC-RTT) performed the tasks by swiping on a touch screen instead of using a manipulandum.

- *MC Maze*: In this task, the monkey performed delayed reaches to visually presented targets while navigating around the boundaries of a virtual maze. The dataset includes a wide range of behavioral configurations, each defined by unique combinations of target positions, barrier counts, and barrier placements. This diversity results in both straight and curved reach trajectories. With thousands of trials, MC Maze provides a rich substrate for analyzing population-level neural dynamics. Moreover, its delayed reaching design facilitates a clear dissociation between neural activity related to movement preparation and execution.
- *Random Target Task (RTT)*: In this task, the monkey executes a variant of the RT task described previously. Instead of performing reaches separated via auditory cues, the animal completes continuous, point-to-point reaching movements between virtually presented targets. The movements begin and end at various locations, span variable distances, and include only a few repeated trajectories. This variability makes RTT particularly useful for evaluating a model’s ability to generalize across complex and less stereotyped behaviors.

The behavioral sampling rate for the RTT and Maze tasks was 1000 and 100 Hz, respectively. To standardize the sampling rate across all datasets used in this work, we downsampled behavioral samples for the RTT task to 100 Hz.

C Model Implementation Details

C.1 Spikachu implementation details

Spikachu is a general framework that supports various implementations. Here, we outline the one used in this work.

Learnable embeddings for units. We assign each unit a unique 32-dimensional learnable embedding.

Harmonizer. This is the ANN part of our architecture described in Sec. 3.1. The cross-attention block is implemented as described in Jaegle et al. [61] and is initialized with 128, 32-dimensional learnable latent queries. The linear layer following the cross-attention block projects the latents to a 128-dimensional vector.

Multi Scale SNN-I. The 128-dimensional latent vector is passed through 3 parallel spiking MLPs. Each MLP is composed of 4 layers and projects the 128-dimensional input vector to a 256-dimensional latent vector (all hidden layers and output layer have a dimensionality of 256). Each MLP has neurons whose membrane potential dynamics evolve at distinct temporal scales. We initialize the learnable decay constants of the neurons of the 3 parallel networks to values τ of 1.11, 1.46, and 434.79 and allow them to update freely during model training.

Spiking Self-Attention. The three 256-dimensional latents extracted by Multi Scale SNN-I are arranged in a sequence and processed using spiking self-attention as described by Zhou et al. [55]. The input is projected to equally shaped 512-dimensional keys, queries, and values prior to spiking self-attention which is carried out using 8 heads. The latent is then passed through a 2-layer spiking feed forward network with a hidden size of 256. Residual connections are employed after the attention calculation as well as after the feed forward network, as is employed in Spikformer [55].

Spiking MLP. The three 256-dimensional outputs of the attention operation are unrolled and are projected to a dimensionality of 384 via one linear layer followed by a spiking activation.

Multi Scale SNN-II. The 384-dimensional latent is passed through 2 parallel spiking MLPs composed of 4 layers each. The hidden size for each MLP is set to 384. The learnable decay constant for the neurons of each of the two parallel MLPs is set to an initial τ of 1.11, and 434.79. Following processing, the latents from the 2 parallel MLPs are concatenated, resulting in a latent with dimensionality 768.

Membrane potential observer layer. The 768-dimensional latent is then passed through a spiking activation. Instead of tracking the output of the neurons of this activation layer (they never fire), we keep track of their membrane potential instead. This results in a latent of the same dimensionality of 768.

Readout layer. The 768-dimensional latent is then projected to a 2-dimensional vector, which is our network's velocity prediction (v_x, v_y) , using a linear layer.

C.2 Baseline model implementation details

We introduced several baseline architectures for our experimental comparisons, which we describe in detail below.

MLP. Our MLP baseline comprises seven consecutive hidden dense layers of ANN neurons, with input shapes: [100, 256, 512, 1024, 1024, 512, 256]. We used the ReLU nonlinearity and dropout probability of 0.1. Since the model is not recurrent, we supplied a rolling window of 10 consecutive binned spike windows during inference.

GRU. Our GRU baseline comprises four sequential blocks, as implemented in PyTorch. Each block contains four cells, with a hidden dimension of 164. We used a dropout probability of 0.1.

LSTM. Our LSTM baseline comprises four sequential blocks, as implemented in PyTorch. Each block contains four cells, with a hidden dimension of 162. We used a dropout probability of 0.1.

POYO. Our POYO baseline was implemented using the model coded by Azabou et al. [26], made available at <https://github.com/neuro-galaxy/poyo>. We trained the model using $N=256$ latent tokens each with a dimensionality of 128.

Causal POYO. Our causal POYO baseline is implemented using the model coded by Azabou et al. [26]. On top of the provided model, we added causal masking in all the model's transformer blocks. Specifically, for each attention module described in Azabou et al. [26], let $\mathbf{Q}$ be the set of N latent query vectors with timestamps $t_i^Q, i = 1, \dots, N$, and $\mathbf{K}$ be the set of M key vectors with timestamps $t_i^K, i = 1, \dots, M$. We dynamically masked the attention multiplication $\mathbf{Q}\mathbf{K}^\top$ such that $t_i^Q \leq t_j^K$, for all pairs i, j for which the attention matrix is computed. This is a direct generalization of lower-triangular masking of the attention matrix commonly used in transformers to prevent tokens from accessing future tokens, effectively making the operation causal. The masks used for masking the (1) encoder cross-attention block, (2) each self-attention block, and (3) the decoder cross-attention block used in causal POYO are shown in Fig. 7.

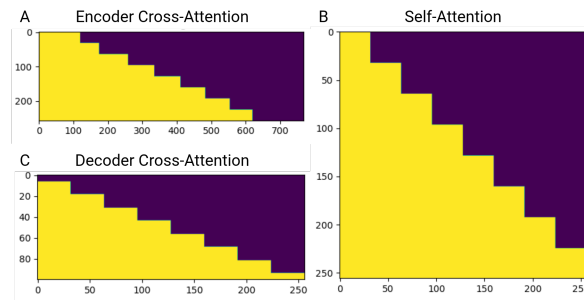


Figure 7: *Masking used to make POYO Causal.* Mask used in (A) encoder cross-attention, (B) self-attention, and (C) decoder cross-attention blocks.

D Model Training Details

All models were coded in PyTorch [83]. The spiking components of Spikachu were implemented using SpikingJelly [84].

D.1 Training hyperparameters

All models were trained using LAMB optimizer [85] with weight decay set to 10^{-4} . The learning rate was initialized at 2×10^{-3} and held constant for the first 75% of training epochs and was decayed to zero using a cosine schedule for the remaining 25%. Single-session models were trained for 1000 epochs with a batch size of 128, while multi-session models were trained for 400 epochs with a batch size of 512.

D.2 Compute

Model training was conducted on a multi-GPU cluster. Depending on resource availability, single-session models were trained using one of several NVIDIA GPUs, from the following: L40s, L40, A40, 3090, A6000, 2080 Ti, and A10. Training these models required under 11GB of GPU memory and typically less than an hour, with a few sessions taking up to 5 hours when trained on a 2080 Ti graphics card. Finetuning to individual sessions was performed on the same range of GPUs and took approximately the same amount of time as training from scratch.

Training of multi-session models was performed on an NVIDIA A40 GPU with 48GB of memory. All multi-session models, including Spikachu-mp, our most computationally intensive model, completed training in under 48 hours.

D.3 Data augmentation

For all model training (both single-session and multi-session) we employed unit dropout augmentation, introduced by Azabou et al. [26]. In this approach, a random subset of the recorded neural population is sampled within time windows and used for training, effectively simulating variability in unit availability. To prevent the augmentation from being overly destructive, we enforced a minimum population size of 30 units per window.

D.4 Training objective

All models were trained by minimizing the MSE loss between the true and predicted hand velocity sequences of the NHPs performing the various behavioral tasks. For the RT task, all behavioral timepoints were weighted equally during training. For the CO task, we increased the weight of the prediction during the reaching segments of the movements by a factor of 5 as has been done in previous works [26, 86].

D.5 Evaluation details

Although model training is performed on arbitrary segments of neural data, evaluation follows standardized protocols established in prior works [26, 37, 87]. For CO and MC-Maze, we report decoding performance only during the reaching phase of successfully completed trials. For RT, which includes a hold period followed by a sequence of 3 to 4 random reaches, we likewise evaluate performance only during the reaching movements. Since movements are continuous for MC-RTT, we report decoding accuracy across all available segments.

D.6 Surrogate gradients

Spiking neurons are inherently non-differentiable due to their discrete activation function (the Heaviside step function) which presents a fundamental challenge for gradient-based backpropagation. To overcome this limitation, we adopt a surrogate gradient approach, replacing the non-differentiable Heaviside function with the arctangent (atan) function when calculating the gradients used for backpropagation. This surrogate function has been demonstrated to support the stable training of deep SNNs in prior works [84, 88].

D.7 Smoothing

Prior to gradient computation, model predictions were smoothed along the temporal axis using a moving average with a window size of 20 samples. To ensure a fair comparison, identical smoothing was used across all models.

D.8 Finetuning/Transferring details

The training procedure for finetuning and transferring Spikachu-mp to individual sessions is identical to that of training single-session models from scratch, except: models trained from scratch were initialized randomly (using seed 42); finetuned and transferred models were initialized with the pretrained weights from the multi-session pretrained models. In both cases, all learnable parameters were updated throughout training.

E Energy Analysis

Estimating energy consumption for any machine learning model consists of approximating the total amount of floating-point operations (FLOPs) necessary to execute the forward pass of the model. These are split into two categories, multiply-accumulate (MAC) and accumulate (AC) operations. A MAC operation multiplies two operands and adds the result to an accumulator. Symbolically, if x, y are the operands and a is the accumulator, a MAC is defined as: $a \leftarrow a + x \cdot y$. An AC operation acts on a single operand by adding it to an accumulator. If x is the operand and a is the accumulator, an AC is defined as: $a \leftarrow a + x$.

Each instance of these operations consumes a certain amount of energy, and by summing, we can estimate the total energy consumption of a model's forward pass. As quantified by Horowitz [89], performing ACs is cheaper than MACs in hardware, specifically: $E_{\text{MAC}} \simeq 4.6pJ$, $E_{\text{AC}} \simeq 0.9pJ$. Therefore, a model's total energy becomes: $E_{\text{total}} = N_{\text{AC}} \cdot E_{\text{AC}} + N_{\text{MAC}} \cdot E_{\text{MAC}}$. The N variables represent the total count of the respective operations in the architecture.

For reference, a matrix multiplication of the form $A \cdot B$, $A \in \mathbb{R}^{n \times m}$, $B \in \mathbb{R}^{m \times k}$, requires $(n \cdot m \cdot k)$ MAC operations, and vector addition of dimension d requires d AC operations.

E.1 Baseline energy calculation

Here, we describe the methodology used to compute the energy required per inference for each baseline model in this work.

E.1.1 MLP

Each linear layer of an MLP is parameterized by a weight matrix $\mathbf{W} \in \mathbb{R}^{n_i \times n_o}$ and a bias vector $\mathbf{b} \in \mathbb{R}^{n_o}$. Given an input $\mathbf{x} \in \mathbb{R}^{n_i}$, the operation performed is: $\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$. The energy expenditure of an L -layer MLP is therefore,

$$E_{\text{MLP}} = \sum_{l=1}^L n_i^{(l)} n_o^{(l)} \cdot E_{\text{MAC}} + n_o^{(l)} \cdot E_{\text{AC}}, \quad (15)$$

where $l \in \{1, \dots, L\}$ is the layer index.

E.1.2 GRU

The equations for a GRU cell are described in detail by Chung et al. [90]. Briefly, each cell can be expressed as,

$$\begin{aligned} r_t &= \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr}), \\ z_t &= \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz}), \\ n_t &= \tanh(W_{in}x_t + b_{in} + r_t \odot (W_{hn}h_{(t-1)} + b_{hn})), \\ h_t &= (1 - z_t) \odot n_t + z_t \odot h_{(t-1)}, \end{aligned}$$

where x is the input, W -variables refer to weight matrices, b -variables to bias vectors, h is the hidden state, t the timestep index, i denotes variables acting directly on the input, and r, z, n are the reset, update, and new gates, respectively. We also denote the sigmoid function with σ and the Hadamard product with $\odot$. From these equations, we can calculate the total operations performed in each cell as a function of its input and hidden state sizes as,

$$N_{\text{MAC}}^{\text{GRU}} = 3 \cdot (n_h(n_i + n_h) + n_h) + 4n_h, \text{ and } N_{\text{AC}}^{\text{GRU}} = 6n_h, \quad (16)$$

where n_i is the input size and n_h is the hidden size. For the full module comprising multiple cells, we sum the counts of the above operations for each cell to obtain the total energy,

$$E_{\text{GRU}} = N_B \cdot \left(E_{\text{MAC}} \sum_{c=1}^C N_{\text{MAC}}^{\text{GRU}}(c) + E_{\text{AC}} \sum_{c=1}^C N_{\text{AC}}^{\text{GRU}}(c) \right) + E_{\text{MLP}}, \quad (17)$$

where c is an index over the C cells in each block, N_B is the total number of multi-cell blocks, and E_{MLP} represents the energy of an MLP used for readout, with energy given by Eq. 15.

E.1.3 LSTM

The exact implementations of the LSTM architecture can be found in Sak et al. [91]. Briefly, each cell can be described as,

$$\begin{aligned} i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}), \\ f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}), \\ g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg}), \\ o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}), \\ c_t &= f_t \odot c_{(t-1)} + i_t \odot g_t, \\ h_t &= o_t \odot \tanh(c_t), \end{aligned}$$

where x is the input, W -variables refer to weight matrices, b -variables to bias vectors, h is the hidden state, t the timestep index, i, f, g, o are the input, forget, cell, and output gates, respectively, and c is the cell state. From these equations, the total operations can be counted as,

$$N_{\text{MAC}}^{\text{LSTM}} = 4 \cdot (n_h(n_i + n_h) + n_h) + 3n_h, \text{ and } N_{\text{AC}}^{\text{LSTM}} = 9n_h, \quad (18)$$

where n_i is the input size and n_h is the hidden size. For the full module comprising multiple cells, we sum each cell's counts of the above operations to obtain the total energy,

$$E_{\text{LSTM}} = N_B \cdot \left(E_{\text{MAC}} \sum_{c=1}^C N_{\text{MAC}}^{\text{LSTM}}(c) + E_{\text{AC}} \sum_{c=1}^C N_{\text{AC}}^{\text{LSTM}}(c) \right) + E_{\text{MLP}}, \quad (19)$$

where c is an index over the C cells in each block, N_B is the total number of multi-cell blocks, and E_{MLP} represents the energy of a small MLP used for readout, with energy given by Eq. 15.

E.1.4 POYO

Attention. We will first describe how we calculate the energy for a single attention module, based on which POYO is built [26]. To compute the energy requirements of the attention operation (see Eq. 11), we cache the dimensions of the query, key, and value tensors at runtime. We can then precisely calculate the FLOPs necessary for the matrix multiplications, scaling, and/or softmax operations performed in attention. The following derivation explains the energy requirements for a single cross-attention block.[†]

Let L_1, L_2 be the input and output sequence lengths respectively, E the embedding dimension, $\mathbf{Q} \in \mathbb{R}^{L_1 \times E}$ the query tensor, $\mathbf{K} \in \mathbb{R}^{L_2 \times E}$ the key tensor, and $\mathbf{V} \in \mathbb{R}^{L_2 \times D}$ the value tensor. The computational requirements following Eq. 11, for a single attention head, are computed as follows,

$$\begin{aligned} N_{\text{MAC}}^{\mathbf{QK}^\top} &= EL_1L_2, \quad N_{\text{MAC}}^{\text{norm}} = 3L_1L_2, \quad N_{\text{MAC}}^{\mathbf{V}} = DL_2^2 \Rightarrow \\ N_{\text{MAC}}^{\text{CA}} &= L_2 \cdot ((E + 3)L_1 + DL_2). \end{aligned}$$

For *multi-head attention*, we need only multiply the above by the number of heads. Letting H be the number of heads, the total energy for the cross attention operation is,

$$E_{\text{CA}} = H \cdot E_{\text{MAC}} \cdot N_{\text{MAC}}^{\text{CA}}. \quad (20)$$

Note that the cost for projecting inputs to $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ and projecting outputs to the appropriate downstream shape is computed separately (linear layers, computation follows the logic of E.1.1).

POYO. By iteratively applying Eq. 20, computing the energy requirements per inference for POYO becomes straightforward. For each attention block, we need to account for the attention energy, and also the energy necessary for projections, given by Eq. 15. Specifically,

$$E_{\text{proj}} = E_{\text{proj}}^{\mathbf{Q}} + E_{\text{proj}}^{\mathbf{K}} + E_{\text{proj}}^{\mathbf{V}} + E_{\text{proj}}^{\mathbf{o}}, \quad (21)$$

where the summands correspond to the energy required for the query, key, value, and output projections respectively. The total energy requirement for POYO can now be expressed as,

$$E_{\text{POYO}} = (E_{\text{CA}}^{\text{enc}} + E_{\text{proj}}^{\text{enc}}) + N_{\text{SA}}(E_{\text{SA}} + E_{\text{proj}}^{\text{SA}}) + (E_{\text{CA}}^{\text{dec}} + E_{\text{proj}}^{\text{CA}}) + E_{\text{MLP}}, \quad (22)$$

[†]For self-attention, we need only set the sequence lengths to be equal; the derivation is otherwise identical.

where N^{SA} is the number of consecutive self-attention blocks used in the model. We note that one more subcomponent exists for this model, namely positional encoding using RoPE [92]. However, since the operations involved in computing the positional encoding are very limited, we did not take them into account when estimating POYO's energy requirements.

E.2 SNN energy

Computing the energy requirement for an SNN follows the same principles used for ANNs. However, SNNs achieve energy savings via two properties of their processing: (a) not all neurons spike at each point in time, and (b) in the case of binary spikes (unit magnitude), matrix multiplications simplify to indexing and accumulation (addition) of elements of the weight matrices. These savings become tangible during inference when deploying on specialized neuromorphic hardware [44, 45]. Following [49, 93–95], we calculate these savings while training and running our models on GPUs. To achieve this, a careful accounting is performed of the proportion of neurons that spiked at each spiking neuron layer for a given forward pass of a model. That rate subsequently reduces the next layer's FLOP count. For example, if the next layer is dense, the FLOP count for that layer is reduced proportionately to the spiking rate of the given layer. Furthermore, as was explained earlier, ACs in hardware are cheaper than MACs. This is another source of savings for SNNs, where each spike's forward computation is significantly cheaper than a whole floating-point MAC.

E.2.1 Spiking MLPs

Here we describe the energy required per inference for a spiking MLP. Given a spiking MLP of L layers, let T denote the number of simulation steps used to update the state of each neuron. To clarify, $T > 1$ values are commonly used in SNN works to signify that the same, time-independent input (e.g., a static image) is presented to the SNN T times in succession, allowing neuron dynamics to converge. The result of the network is then obtained by aggregation of the output layer's values over this artificially introduced time dimension. In our case, $T = 1$, since each sample of the input does not need to be repeated, due to our data's inherently temporal nature. Furthermore, let $\rho_{l-1} \in [0, 1]$ be the proportion of the previous layer's neurons that spiked, and E_l be the energy consumed by the l -th layer. Also let E_W denote the energy per algebraic operation for a dense layer with weight matrix $\mathbf{W} \in \mathbb{R}^{n_i \times n_o}$ and bias vector $\mathbf{b} \in \mathbb{R}^{n_o}$, with n_i, n_o being the input and output shapes respectively.

Then, for a batch size of 1, the equations for the energy consumption of a spiking MLP are as follows,

$$E_{\text{SMLP}} = \sum_{l=1}^L T \cdot \rho_{l-1} \cdot n_i^{(l)} n_o^{(l)} \cdot E_W + n_o^{(l)} \cdot E_{\text{AC}}, \quad (23)$$

where $l \in \{1, 2, \dots, N-1\}$ is the layer index. This is similar to Eq. 15, the only differences being that there are spiking rates to help reduce the computational load, and that input values are binary spikes, meaning that: $E_W = E_{\text{AC}} < E_{\text{MAC}}$. The latter holds because matrix-vector multiplication can be expressed purely in terms of indexing and accumulation when the vector is binary.

E.2.2 Spiking self-attention

We calculate spiking self-attention energy based on the formulation in App. A.2. Let L be the sequence length, E the embedding dimension, $\mathbf{Q} \in \mathbb{R}^{L \times E}$ the query tensor, $\mathbf{K} \in \mathbb{R}^{L \times E}$ the key tensor, and $\mathbf{V} \in \mathbb{R}^{L \times D}$ the value tensor. The computational requirements following Eq. 13, for a single attention head, are computed as follows,

$$N_{\text{AC}}^{\mathbf{QK}^\top} = L^2 E, \quad N_{\text{MAC}}^{\text{norm}} = L^2, \quad N_{\text{AC}}^{\mathbf{V}} = L^2 D \Rightarrow \\ N_{\text{AC}}^{\text{SSA}} = L^2 \cdot (E + D), \quad \text{and} \quad N_{\text{MAC}}^{\text{SSA}} = L^2.$$

For *multi-head attention*, we need only multiply the above by the number of heads. Letting H be the number of heads, the total energy for the cross attention operation is,

$$E_{\text{SSA}} = H \cdot (E_{\text{MAC}} \cdot N_{\text{MAC}}^{\text{SSA}} + E_{\text{AC}} \cdot N_{\text{AC}}^{\text{SSA}}). \quad (24)$$

Note that the cost for projecting inputs to $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ and projecting outputs to the appropriate downstream shape is computed separately (spiking linear layers, computation follows the logic of E.2.1).

E.2.3 Spikachu

To compute the total energy consumed by Spikachu during a forward pass, we compute the sum of the energy of its modules, described below.

Harmonizer. This is a CA block followed by a linear layer. Its energy is computed following Eq. 20, with the addition of the energy cost of the projection layers (denoted $E_h^Q, E_h^K, E_h^V, E_h^{\text{out}}$), and the extra linear layer (denoted E_h^{SMLP}),

$$E_h = H \cdot E_{\text{MAC}} \cdot N_{\text{MAC}}^{\text{CA}} + E_h^Q + E_h^K + E_h^V + E_h^{\text{out}} + E_h^{\text{SMLP}}, \quad (25)$$

where each of the summands outside the parentheses is a direct application of Eq. 15.

Multi Scale SNN-I. This is a combination of parallel spiking MLPs, based on Eq. 23,

$$E_{s,1} = \sum_{p=1}^P E_{\text{SMLP}}^p = E_{\text{AC}} \cdot P \cdot \sum_{l=1}^L T \cdot \rho_{l-1} \cdot n_i^{(l)} n_o^{(l)} + n_o^{(l)}, \quad (26)$$

where P is the number of parallel spiking MLPs and $l \in \{1, \dots, L\}$ is the layer index.

Spiking Self-Attention. SSA is implemented as described in Sec. A.2, with energy computed based on Eq. 24,

$$E_a = H \cdot (E_{\text{MAC}} \cdot N_{\text{MAC}}^{\text{SSA}} + E_{\text{AC}} \cdot N_{\text{AC}}^{\text{SSA}}) + E_{\text{SSA}}^Q + E_{\text{SSA}}^K + E_{\text{SSA}}^V + E_{\text{SSA}}^{\text{out}}, \quad (27)$$

where the added summands are spiking MLPs used for projections, their energy calculated using Eq. 23.

Spiking MLP. Mixing module, energy given by Eq. 23, denoted E_m .

Multi Scale SNN-II. Same as Multi Scale SNN—I, denoted $E_{s,2}$.

Membrane potential observer layer. Layer containing only leaky integrators, energy per time-step is 1 AC operation per neuron,

$$E_o = n_o \cdot E_{\text{AC}}, \quad (28)$$

where n_o is the number of observer neurons.

Readout layer. Single linear layer, equivalent to a single layer of Eq. 23, denoted by E_r .

Thus, the total energy per inference for Spikachu is given by,

$$E_{\text{Spikachu}} = E_h + E_{s,1} + E_a + E_m + E_{s,2} + E_o + E_r. \quad (29)$$

F Additional Results

F.1 Scaling laws of multi-session, multi-subject training: Continued

In Sec. 4.5, we demonstrated that pretraining Spikachu on multi-session, multi-subject datasets improves both decoding performance and energy efficiency. Here, we present additional evidence that complements those findings.

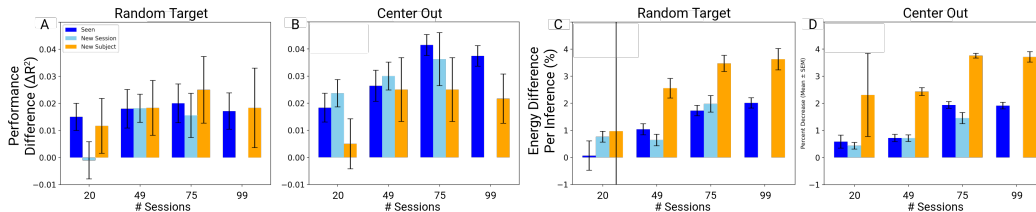


Figure 8: *Benefits of scaling up training dataset size.* (A, B) Performance difference between the single-subject models trained from scratch vs finetuned/transferred models from pretrained for the RT, and CO tasks. (C, D) Percent difference in energy consumption between the same groups for the same tasks.

Fig. 8A, B shows the improvement in per-session test set decoding performance achieved by the pretrained models after (1) finetuning on sessions seen during pretraining (monkeys C, J, and M; blue), (2) transferring to new sessions from animals seen during pretraining (monkeys C, J, and M; skyblue), and (3) transferring to new sessions from a new animal, unseen during pretraining (monkey T; orange). Across all conditions, we observe consistent performance gains that increase with the number of sessions used for pretraining, with gains possibly saturating beyond ~ 75 sessions.

In Fig. 8C, D, we report the relative reduction in energy consumption per inference,

$$\Delta E = -\frac{E_{\text{msf}} - E_{\text{ss}}}{E_{\text{ss}}},$$

where E_{msf} is the energy required per inference by the pretrained model after finetuning, and E_{ss} is the energy used by a single-session model trained from scratch. Across all finetuning conditions, energy savings scale positively with the size of the pretraining dataset. Given that typical BCI sessions require tens of thousands of inferences per session, these efficiency gains are highly practical in a real-world setting.

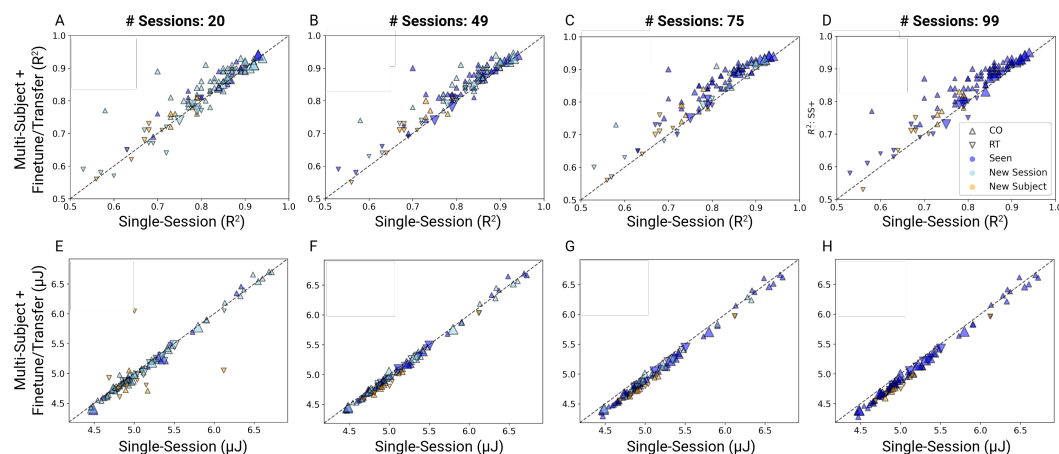


Figure 9: *Decoding performance and energy benefits of scaling up.* (A, B, C, D) Head-to-head decoding performance comparison between single-session models trained from scratch vs (1) the pretrained models + finetuning (seen; blue) and (2) pretrained models + transfer (new session; skyblue and new subject; orange). (E, F, G, H) Head-to-head comparison of the energy consumed per inference for the same groups.

To further illustrate these benefits, Fig. 9A–D presents head-to-head comparisons of decoding performance between finetuned models and their scratch-trained counterparts for pretraining on 20, 49, 75, and 99 sessions. In each case, the finetuned models outperform their baseline equivalents, with the performance gap widening as pretraining data increases. Figs. 9E–H compare energy usage per inference under the same conditions. Here, we observe that finetuned models consistently consume less energy than their scratch-trained counterparts, emphasizing the energy efficiency achieved through pretraining. The energy savings also scale positively with the amount of pretraining data.

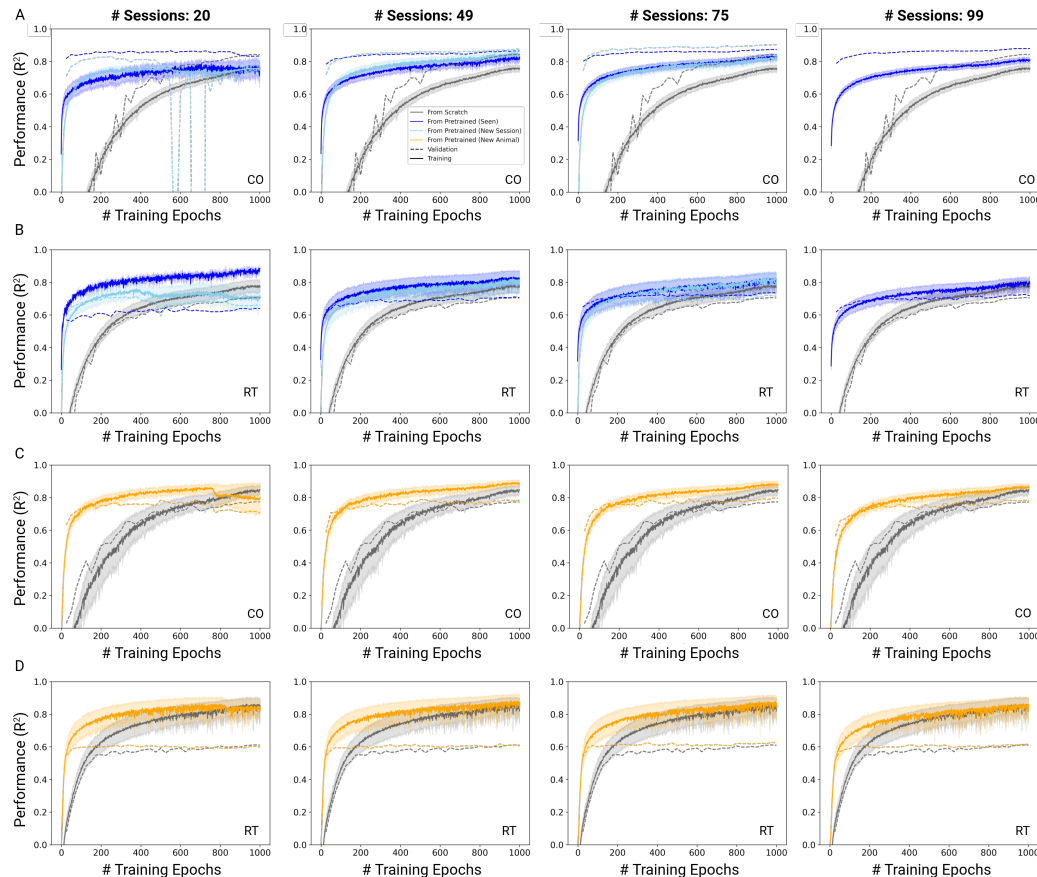


Figure 10: *Learning speedup achieved through pretraining.* (A) Learning Curves for the CO task for (1) single-session models trained from scratch (gray), (2) the pretrained models + finetuning (seen; blue), and (3) pretrained models + transfer (new session; skyblue). (B) Learning Curves for the RT task for (1) single-session models trained from scratch (gray), (2) the pretrained models + finetuning (seen; blue), and (3) pretrained models + transfer (new session; skyblue). (C) Learning Curves for the CO task for (1) single-session models trained from scratch (gray), and (2) pretrained models + transfer (new subject; orange). (D) Learning Curves for the RT task for (1) single-session models trained from scratch (gray), and (2) pretrained models + transfer (new subject; orange).

Finally, Figs. 10A–D show the mean learning curves under the same three finetuning conditions, alongside the baseline learning curves of scratch-trained single-session models (gray). Across all levels of pretraining, finetuning the pretrained models to single sessions converges much faster than training single-session models from scratch. This demonstrates that the pretrained models learn transferable, general-purpose neural representations shared across both sessions and subjects that can be transferred very efficiently.

F.2 Ablation study

F.2.1 Building blocks

To identify the contribution of each building block of Spikachu to the decoding performance of our models, we performed an ablation study using the 99 recording sessions from monkeys C, J, M drawn from Perich et al. [36]. Specifically, we trained variants of our proposed architecture, each time removing a building block and assessing the model’s decoding performance. To ensure a fair comparison, training hyperparameters were kept identical across all training runs.

The building blocks of our architecture are described in App. C.1 (see also Fig. 1). For convenience, we also list them here: Harmonizer (Harm.), Multi scale SNN-I (Multi-Scale I), Spiking Self Attention (SSA), Spiking MLP (sMLP), Multi scale SNN-II (Multi-Scale II), Membrane Potential Observer Layer, Readout Layer.

We note that by ablating the multi-scale SNNs we used one spiking MLP instead of multiple parallel spiking MLPs. We did not ablate the sMLP because there is no straightforward way to project the output of the SSA block to the input dimensionality of Multi Scale SNN-II without it. For the same reason, we did not ablate the “Readout Layer” (needed to project the output of “Membrane Potential Observer Layer” to the output dimensionality of the cursor velocity). We also did not ablate the “Membrane Potential Observer Layer” because there is no straightforward way to track continuous variables (as is the velocity tracked in this work) without this layer when using SNNs.

The results of the ablation study when training Spikachu and variants in single sessions are shown in Fig. 11A, B. We observed that the spiking components of the architecture (“Multi Scale SNN-I” and “Multi Scale SNN-II”) influence model performance most, indicating that our model did not rely on the ANN part (the harmonizer) to perform. We also observed that the performance difference when ablating the SSA block was negligible. To further investigate the utility of the SSA block, we trained variants of our architecture with the SSA block ablated on multiple sessions. We observed that the SSA block did affect the model’s performance when scaling up (see Fig. 11C, D for model performance on the validation set when training on 75 sessions).

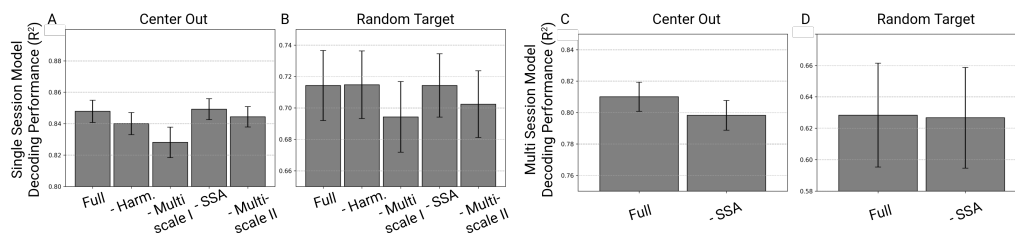


Figure 11: *Summary of ablation results.* (A, B) Single session training decoding performance (mean $\pm$ sem) in the validation set for the (A) CO and (B) RT tasks. (C, D) Multi-session training decoding performance (mean $\pm$ sem) in the validation set for the (A) CO and (B) RT tasks for models trained on 75 sessions from 3 animals.

F.2.2 Spiking mechanism of neurons

To determine whether our model’s performance was driven by its spiking mechanism or simply by network connectivity, we conducted an additional ablation study. In this experiment, we re-implemented Spikachu as an ANN, preserving the exact same connectivity but replacing stateful LIF neurons with stateless ReLU neurons, thereby removing any intrinsic temporal dynamics. We refer to this variant as Spikachu-ANN.

We evaluated Spikachu-ANN on the same 99 recording sessions from monkeys C, J, and M from Perich et al. [36] described in Section 4.2, considering two different training conditions: (1) using only the current timebin (identical to the setup for Spikachu; ANN in Tab. 3), and (2) providing

Table 3: *Model performance for Spikachu and ANN Variants.*

Model Variant	Decod. Perf. (R^2) $\uparrow$	
	CO	RT
ANN	0.5332	0.3642
ANN + context	0.5348	0.3643
SNN	0.8398	0.6761

additional temporal context by including the four preceding timebins as inputs (ANN + context in Tab. 3).

As summarized in Tab. 3, Spikachu (SNN in Tab. 3) consistently outperformed its ANN counterpart under both conditions. Importantly, even when explicitly supplied with temporal context, Spikachu-ANN underperformed the spiking model. These findings demonstrate that Spikachu’s performance arises from the LIF neuron’s spiking dynamics and is not due to network connectivity alone.

F.3 Testing the utility of the neural “harmonizer” on baseline models

In this work, we introduce the neural harmonizer, a novel, causal method for aligning heterogeneous neural recordings across sessions and subjects, as detailed in Sec. 3.1. This approach addresses a key barrier to scaling neural decoding models to multi-session and multi-subject datasets: the reliance of traditional architectures, such as MLPs and GRUs, on homogeneous input structures and consistent neural correspondences. By projecting the disparate neural signals from different datasets into a unified representation, the harmonizer enables any standard model (see Glaser et al. [38] for an overview) to be trained effectively across sessions and subjects, which could enhance the model’s generalizability.

Although the homogenizer was developed as part of the Spikachu framework, we demonstrate its broader applicability by integrating it with baseline models (LSTM, MLP, and GRU) and training them across all 99 neural recording sessions from monkeys C, J, and M in the dataset from Perich et al. [36]. The corresponding test set results, reported separately for the CO and RT tasks, are shown in Tab. 4. The results indicate that the standard models perform well when paired with the neural harmonizer and even see performance gains in some cases. We note that we did not perform any hyperparameter tuning for the harmonizer or the baseline models when combining them into a single pipeline. Instead, we used the harmonizer configuration optimized for Spikachu and the baseline models as trained in the single-session setting described in Sec. 4.2. This highlights the potential of our approach to scale model training across multi-session, multi-subject neural datasets, offering a flexible and powerful foundation for generalized neural decoding.

Table 4: *Impact of harmonizer on model performance.*

Model	Center Out (R^2)		Random Target (R^2)	
	- Harm.	+ Harm.	- Harm.	+ Harm.
LSTM	0.4935	0.5804	0.4214	0.5919
MLP	0.7424	0.6415	0.5724	0.5229
GRU	0.8336	0.8187	0.6681	0.6110

F.4 Representation analysis of the unit embedding space of Spikachu-mp

In this section, we investigated whether any structure emerged in the latent space of Spikachu-mp (see Sec. 4.3), the model we trained on 99 recording sessions from monkeys C, J, M performing the CO and RT tasks from Perich et al. [36].

Since our model does not have any explicit session-specific embeddings, for this analysis, we used the unit embeddings for each electrode of each session. To investigate whether meaningful structure emerged in the latent space of our trained model, we aggregated the unit embeddings within each session into a 2D matrix and applied PCA, retaining the first five principal components as a summary representation for each session. We then used these session-level representations as features in two distinct Linear Discriminant Analyses (LDA): one to assess separability by subject (monkeys C, J, and M) and another by task (Center Out vs. Random Target). Visualizing the session embeddings projected into the LDA space revealed clear clustering both by subject (see Fig. 12A) and by task (see Fig. 12B).

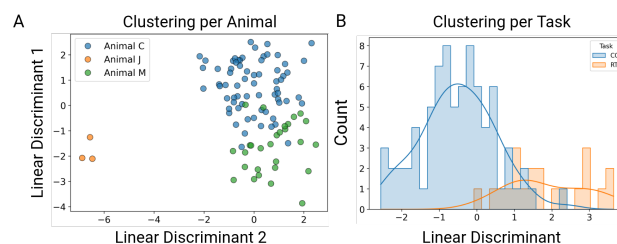


Figure 12: *Linear Discriminant Analysis of session embeddings.* Visualization of session-level embeddings in the latent space when maximizing separability for (A) Subject, (B) Task.

To quantify the separability, we trained an SVM classifier (using scikit-learn’s SVC with default parameters) using 5-fold cross-validation with features the summary representations of each session (the first five principal components of the aggregated unit embeddings; same as those used in LDA) to predict (1) the subject and (2) the task associated with each session. We achieved an average accuracy of 0.69 ± 0.02 for subject classification (chance = 0.33) and 0.79 ± 0.02 for task classification (chance = 0.50), suggesting that Spikachu-mp learned latent representations that reflect high-level structure in the data even though it is not explicitly trained to do so.

F.5 Profiling Spikachu in terms of memory access costs

In Sec. 4.2, we benchmarked Spikachu’s performance against various baseline models commonly used for neural decoding. In this section, we provide comparisons between Spikachu and baselines in terms of computational and memory-access costs per inference. We note that this comparison is important since the total energy expenditure per inference consists of the energy required for computation, as well as the energy required to load (LOAD Ops) and store (STORE Ops) data in memory.

For this analysis, we used the 99 single-session models trained on the recording sessions from monkeys C, J, and M performing the CO and RT tasks from Perich et al. [36] (models produced by the experiment in Sec. 4.2). For each model and session, we calculated the number of FLOPs (i.e. MACs, and ACs) required for inference (see Sec. E). Our findings for the number of necessary operations for each model are presented in Tab. 5. Notably, Spikachu has a higher proportion of AC operations compared to other models, due to the binary output of each spiking neuron layer, which simplifies downstream computation. *In conclusion, Spikachu requires the least amount of such operations, thanks to its SNN backbone.* ‡

Table 5: *FLOP counts for Spikachu and baselines.* Best performing model is in bold and second best model is underlined. M stands for millions.

Model	MAC (M) ↓	AC (M) ↓	Total FLOPs (M) ↓
LSTM	3.27	0.02	3.29
MLP	2.63	0.02	2.65
POYO-causal	466	1.0	467
GRU	<u>2.53</u>	0.01	<u>2.54</u>
POYO	466	1.0	467
Spikachu	0.97	<u>0.78</u>	1.75

Next, we use the aforementioned results to estimate memory access requirements by converting FLOPs to LOAD and STORE operations. As described in Liao et al. [52], we assume $N_{MAC}^{LOAD} = 3$ LOAD and $N_{MAC}^{STORE} = 1$ STORE operation per MAC and $N_{AC}^{LOAD} = 2$ LOAD and $N_{AC}^{STORE} = 1$ STORE operation per AC. Therefore, memory operations can be estimated as,

$$N^{LOAD} = N_{MAC}^{LOAD} \cdot N_{MAC} + N_{AC}^{LOAD} \cdot N_{AC} \quad (30)$$

$$N^{STORE} = N_{MAC}^{STORE} \cdot N_{MAC} + N_{AC}^{STORE} \cdot N_{AC} \quad (31)$$

where N_{AC} and N_{MAC} denote the total number of AC and MAC operations for a given architecture, respectively.

The results, averaged across all sessions and tasks for Spikachu and baselines, are summarized in Tab. 6. *Notably, Spikachu required the fewest memory access costs across the board, highlighting its energy efficiency not only from reduced computational demands but also from minimized memory costs.* This result further strengthens the promise of energy-efficient deployment of Spikachu for fully implantable BCIs.

Table 6: *Memory access counts for Spikachu and baselines.* Best performing model is in bold and second best model is underlined. M stands for millions.

Model	Load (M) ↓	Store (M) ↓	Total Ops (M) ↓
LSTM	9.85	3.29	13.14
MLP	7.93	2.65	10.58
POYO-causal	1400	467	1860
GRU	<u>7.61</u>	<u>2.54</u>	<u>10.15</u>
POYO	1400	467	1860
Spikachu	4.47	1.75	6.22

We also note that while this analysis is hardware-agnostic, the practical energy savings are likely substantially greater than what is suggested here. The present analysis is most directly applicable to von Neumann architectures (e.g., conventional CPUs and GPUs), where further memory overheads

‡In terms of energy expenditure, AC is significantly cheaper than MAC [89].

arise from maintaining DRAM blocks even when their contents are not accessed during a clock cycle [96]. In contrast, neuromorphic hardware (the intended deployment platform for Spikachu) typically relies on local SRAM memory, which incurs negligible keep-alive costs and only needs to supply data on a per-core basis. This architecture effectively eliminates the bulk of memory-related costs compared to von Neumann systems. As a result, energy savings are expected to be significantly more pronounced on neuromorphic hardware, with total memory costs likely falling below the estimates derived from our hardware-agnostic analysis presented in this section.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: To the best of our knowledge, all claims made in the abstract and introduction are supported by our experiments described in Sec. 4 and App. F.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: In Sec. 5, we discuss the limitations of this work.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The only theoretical results in this work describe our process for calculating energy requirements for various machine learning models in App. E. To the best of our knowledge, they are complete and correct.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Detailed information required to reproduce all experiments can be found in Sec. 4 and App. C, D, and E.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The dataset used in this work has been made publically available by Azabou et al. [26] here:

<https://github.com/neuro-galaxy/poyo>. We will make our code publically available upon acceptance of this manuscript.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: All details of our training can be found in App. D. We also share the data splits and random seed used to generate them in App. B.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Error bars report mean $\pm$ standard error of the mean for all plots, unless otherwise mentioned. No experiments to assess statistical significance were performed in this work.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: App. D.2 describes the compute resources used in this work.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: To the best of our knowledge, this work complies in every aspect with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Please see Sec. 1 and 5. We have not discussed negative societal impacts, since there are none to the best of our knowledge.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We have not discussed such safeguards, as there is no foreseeable potential for misuse of our work. We hope that the community understands the motivation behind our approach and applies it responsibly to improve the lives of the intended patients.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All assets used for our work are licensed for academic use. We also have an extensive list of citations, ensuring proper attribution of the works of others we used in this manuscript. Where applicable, we have acquired explicit permission to reproduce explanatory graphics in our figures.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Our code is well documented and will be made publically available upon acceptance of this manuscript. There are no other assets generated by this work that require documentation.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This work does not involve any research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: This work does not involve any research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: LLMs did not contribute to the research in this work.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.