Leveraging unlabelled data for generalizable neural population decoding

Ximeng Mao[*,1,2,✉] **Nanda H Krishna**[*,1,2,✉]
Avery Hee-Woon Ryoo[1,2] **Matthew G Perich**[†,1,2] **Guillaume Lajoie**[†,1,2,3,✉]
[1]Mila – Quebec AI Institute [2]Université de Montréal [3]Canada CIFAR AI Chair
[*]Co-first authors [†]Co-senior authors
[✉]ximengmao@gmail.com,{nanda.harishankar-krishna,guillaume.lajoie}@mila.quebec
mojo-brain.github.io

Abstract

Robust and accurate neural decoders are integral to neurotechnologies such as brain-computer interfaces and closed-loop experiments. Recent work has shown that tokenizing neural data at the spike level facilitates multi-session pretraining and delivers state-of-the-art decoding performance. However, current spike-based models are restricted to supervised learning (SL), limiting training to datasets with paired behavioural labels. To address this limitation, we introduce MOJO (**M**asked aut**O**encoder-based **JO**int training), a training framework for spike-tokenizing models that combines self-supervised learning (SSL) via masked autoencoding with SL. We evaluate MOJO on three spiking datasets spanning monkey motor cortex during reaching tasks and multi-region mouse recordings during vision and decision-making tasks, where it consistently outperforms purely SL-trained models. This improvement is especially pronounced in few-shot finetuning, where only a handful of labelled trials from a new session are available. Incorporating SSL also yields unit embeddings that better capture each neuron's firing properties and predict its brain region, which is never seen during training. We further show that MOJO generalizes beyond spiking data to human electrocorticography during speech, where it again outperforms purely SL-trained models and performs on par with neuro-foundation models (NFMs) designed specifically for continuous signals. Overall, augmenting spike-tokenizing models with SSL enables the use of unlabelled data across diverse datasets, improves performance in label-impoverished settings, and extends to other neural modalities. These results suggest a path towards more flexible and scalable data usage when training NFMs.

1 Introduction

Brain-computer interfaces (BCIs) and other neurotechnologies are powered by neural decoders: models that map neural activity to some behavioural variable. Artificial neural networks (ANNs) are well established in the literature as an effective candidate to learn these mappings, with earlier work achieving reasonable success using multilayer perceptrons (MLPs) [1] and recurrent networks (RNNs) [2–4]. However, Transformers [5] and state-space models (SSMs) [6, 7] have recently emerged as particularly adept architectures, primarily due to their scalability as well as their strong decoding performance across a diverse set of modalities and tasks [8–10]. These architectures are now widely applied to varied neural data modalities [11–13], notably on invasive neural spiking data, where different ways to process and tokenize input signals are under rapid development.

Among these approaches, the POYO model family [13, 14] – characterized by a tokenization scheme that operates at single-spike resolution – is particularly notable. This contrasts with the commonly

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

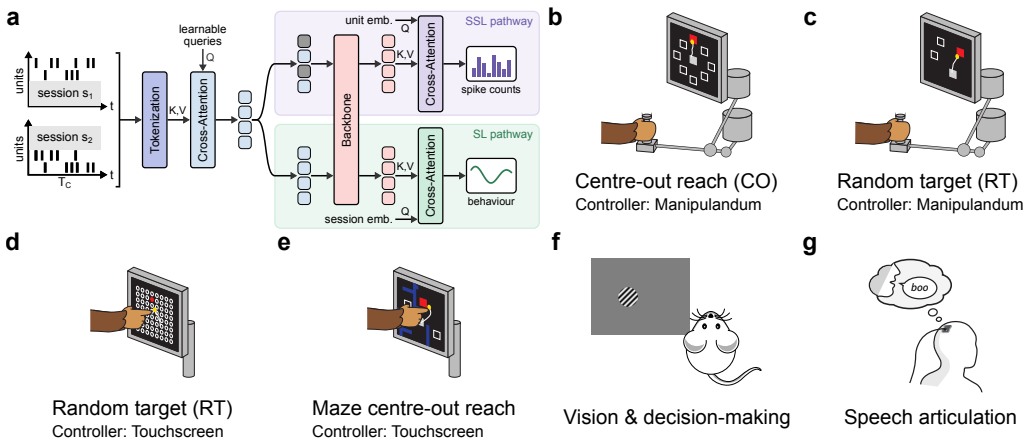


Figure 1: **Model and task schematics.** (a) Schematic showing a POYO-style model augmented with MOJO. Latent representations extracted from tokenized neural data are simultaneously used for supervised learning (SL) and self-supervised learning (SSL). The former is carried out by minimizing error in predicted behaviour while the latter is carried out by reconstructing spike counts from masked latents. (b–e) Schematics describing the monkey reaching tasks. (f) Schematic of the mouse vision and decision-making tasks. (g) Schematic of the human speech articulation task.

adopted approach that bins spike counts [15, 16], which imposes a rigid input formulation and requires additional stitching layers when training across different recording sessions. The flexibility of POYO-style spike tokens facilitates large-scale pretraining across multiple datasets, enabling efficient finetuning and high decoding accuracy across tasks.

Despite several landmark results on spike datasets, the potential of these POYO models is still limited by a major inflexibility: their exclusive reliance on supervised learning (SL) [17]. This limits their pretraining to neural datasets with paired behavioural labels (e.g., common applications include motor tasks in monkeys with neuronal spikes mapped to 2D arm velocities). This approach stands in stark contrast to the development of foundation models such as BERT [18] and GPT [19] in the language domain, and approaches such as SimCLR [20], MAE [21], and I-JEPA [22] in vision, which leverage vast amounts of unlabelled data to learn rich representations via self-supervised learning (SSL). Large amounts of diverse, unannotated neural data have been collected across neuroscience laboratories, highlighting a promising avenue for substantially expanding the size and diversity of pretraining data for models like POYO. SSL can also be applied to the neural part of labelled data, helping the model better learn the neural dynamics that underlie behaviour. Exploiting both requires a pretraining scheme that can (1) extract meaningful structure from neural data and (2) accommodate heterogeneous data formats within a unified pipeline. Although recent work has explored forecasting or input-masking SSL objectives for POYO-based models, these methods are designed for continuous calcium traces [23, 24], and it remains unclear how to design an effective scheme for sequences of spike tokens.

With these goals in mind, we introduce MOJO (**M**asked aut**O**encoder-based **JO**int training), a joint training framework in which SSL and SL objectives are optimized simultaneously. MOJO addresses the challenges of applying SSL on spike tokens by leveraging masked autoencoding [18, 21] on the encoded latents. Given the sparse and irregular nature of neural spikes, this avoids the need to distinguish between non-existent and masked spike tokens at the input level, and enables seamless integration with existing supervised POYO family backbones. A key feature of MOJO is that it optimizes the SSL and SL objectives jointly, so pretraining yields a trained behavioural readout as well as neural representations shaped by behaviour, which are therefore more task-relevant.

We evaluate MOJO with both Transformer and SSM backbones from the POYO family on three neural spiking datasets, spanning monkey reaching and mouse vision and decision-making tasks. To test its generalizability beyond spikes, we also apply MOJO to a human electrocorticography (ECoG) dataset on speech articulation. In all these settings, MOJO outperforms purely supervised methods on fully labelled neural-behaviour pairs, and gains further from additional unlabelled neural data. In particular,

joint SSL-SL training enables more efficient learning in label-impoverished regimes, such as few-shot finetuning. Incorporating SSL also yields unit embeddings that better capture neuron-level properties, and linear probing on these embeddings improves as more heterogeneous data is added, including neural data from other brain regions or even species. Our contributions are as follows:

- We propose MOJO, a joint SSL-SL framework that can leverage unlabelled data in spike-tokenizing models via masked autoencoding.
- We evaluate MOJO on neural datasets spanning multiple species, tasks, brain regions, and modalities, achieving efficient finetuning and improved decoding performance over purely SL-trained models, especially in label-impoverished settings.
- Through SSL, our learned unit embeddings better encode neuron-level properties such as firing statistics and brain region, the latter never being available during training.

2 Methods

As shown in Figure 1, MOJO maintains separate pathways for the SSL and SL objectives, which share most of their parameters. In this section, we describe these two pathways and the components essential to MOJO. We leave detailed descriptions of architectural components, model configurations, and training details to Appendices B and C.

2.1 Spike Tokenization and Input Cross-Attention

The spike tokenization scheme [13] represents each spike as a token pairing the neural unit that emitted it with the exact time it occurred. Each unit is identified only by an arbitrary integer ID, which is mapped to a learnable unit embedding. No other information about the unit, such as its brain region or electrode location, is explicitly encoded.

MOJO employs a POSSM-style encoder [14], which computes cross-attention separately within each contiguous *time chunk* of the data sample. For time chunk t, let $\mathbf{X}_t$ be the sequence of all spike tokens in that chunk and $\mathbf{c}_i^{(t)}$ the i-th learnable latent query. The encoder output is calculated using scaled dot-product attention [5] of $\mathbf{c}_i^{(t)}$ over $\mathbf{X}_t$, producing the latent token $\mathbf{z}_i^{(t)}$.

Unlike POYO, which performs input cross-attention over the entire sequence ($\mathbf{X}$ rather than $\mathbf{X}_t$), the POSSM-style encoder processes each time chunk $\mathbf{X}_t$ independently. In principle, MOJO could maintain both encoders, one per pathway, but this duplicates the input cross-attention computation, so by default we share a single encoder between the two pathways. Appendix D.3.1 compares the two formulations.

2.2 Backbones

MOJO requires a sequence model as its backbone. We consider the two main architectures used in existing spike-tokenizing models: attention-based [13] and recurrence-based [14].

Attention-based. The attention backbone is composed of stacked self-attention blocks operating over the entire latent sequence. Omitting residual connections and feed-forward layers, the hidden state from latent i in time chunk t at layer l is given by:

$$\mathbf{h}_{l,i}^{(t)} = \mathrm{softmax}(\mathbf{q}_{l,i}^{(t)}\mathbf{K}_l^\top/\sqrt{D})\mathbf{V}_l,$$

where $\mathbf{q}_{l,i}^{(t)}$ is projected from $\mathbf{h}_{l-1,i}^{(t)}$, $\mathbf{K}_l$ and $\mathbf{V}_l$ are projected from the layer-$(l-1)$ hidden states of all latents across the sequence, and $\mathbf{h}_{0,i}^{(t)} = \mathbf{z}_i^{(t)}$.

Recurrence-based. The recurrent backbone is a stack of SSM blocks whose hidden state follows the transition function $\mathbf{h}_l^{(t)} = f_{\text{SSM}}(\mathbf{h}_{l-1}^{(t)}, \mathbf{h}_l^{(t-1)})$, with $\mathbf{h}_0^{(t)} = [\mathbf{z}_1^{(t)}, \ldots, \mathbf{z}_{N_c}^{(t)}]$, the concatenation of N_c latent tokens in time chunk t.

2.3 Masked Autoencoder

In the SSL pathway, temporal masking is applied to the latent outputs of the input cross-attention, inspired by the masking strategy of Baevski et al. [25]. For each time chunk t, a mask indicator $m^{(t)} \sim \text{Bernoulli}(0.5)$ is drawn independently, and all latents in that chunk are replaced by:

$$\tilde{\mathbf{z}}_i^{(t)} = (1 - m^{(t)})\, \mathbf{z}_i^{(t)} + m^{(t)}\, \mathbf{e}_{\text{mask}}, \quad \forall i \in \{1, \ldots, N_c\},$$

where $\mathbf{e}_{\text{mask}}$ is a learnable mask token. As the input cross-attention is local to each time chunk, temporal masking removes all spiking information from the masked chunks in the SSL pathway.

2.4 Output Cross-Attention

For the SL pathway, we adopt the multi-task setting [26], where output queries for different tasks cross-attend to the same latent sequence. The outputs are then mapped to predictions using separate linear readouts for each task. The SSL pathway predicts spike rates using a unit query $\mathbf{u}$ that, like an input spike token, combines the unit embedding with a timestamp, here that of the predicted time bin. This design is conceptually similar to that of Duan et al. [23], adapted to spikes. The unit query is decoded with the same cross-attention operations as the SL pathway, except that $\mathbf{u}$ can only attend to latent tokens within its own time chunk.

2.5 Training Objectives

The `MOJO` training objective, $L_{\text{MOJO}} = \alpha_{\text{SSL}} L_{\text{SSL}} + \alpha_{\text{SL}} L_{\text{SL}}$, is a weighted sum of the losses from the two pathways. L_{SSL} is the Poisson negative log-likelihood of the observed spike counts in the masked time chunks under the predicted rates. L_{SL} is a weighted sum of supervised task losses in the multi-task setting (mean squared error for regression and cross-entropy for classification), computed only on samples with labels. Empirically, setting both coefficients to 1 yields strong performance across experiments, even when the labelled and unlabelled training sets differ greatly in size. Overall, the SSL pathway encourages the model to capture the underlying neural population dynamics, while the SL pathway pretrains task readouts and keeps the learned representations task-relevant.

2.6 Pathway Integration

Maintaining completely separate sets of parameters for the SSL and SL pathways would be costly, and also limit how much SSL can shape the representations used for decoding. Therefore, we integrate the two pathways by (1) sharing the output of the input cross-attention and (2) sharing the backbone parameters. The SSL pathway then adds only one output cross-attention module, which accounts for approximately 10% and 2% of the total parameters with the `POSSM` and `POYO` backbones, respectively. In each training step, the input cross-attention is computed once, the backbone is run twice (on masked latents for SSL and unmasked latents for SL), and each pathway then applies its own output cross-attention. This leads to only a modest increase in training time for the same backbone.

3 Experiments

3.1 Monkey Reaching Tasks

The first set of experiments focuses on reaching tasks in non-human primates, using five public monkey datasets from various labs that encompass centre-out (CO), random target (RT), and maze navigation tasks. We decode two-dimensional hand or cursor velocities from neural spiking activity recorded by electrode arrays implanted in primary motor, dorsal premotor, and primary somatosensory cortices, using the same pipeline and dataset splits as Ryoo et al. [14]. Apart from samples flagged as behavioural outliers, all pretraining data in these experiments is labelled, so SSL is applied mainly to the neural part of labelled data.

We evaluate `MOJO` with three backbones: `POGRU` (`POSSM` with GRU [2]), `POMAMBA` (`POSSM` with Mamba [6]), and `POYO`. Both pathways use 20 ms time bins, and each training sample is 1 s long. We follow the causal evaluation protocol of Ryoo et al. [14], which reflects real-time decoding. When adapting the pretrained model to unseen sessions, we use the two finetuning strategies of Azabou et al. [13]: unit identification (UI), which updates only the unit and session embeddings, and full finetuning (FT) (Appendix B.6).

Table 1: **Behavioural decoding results on monkey reaching tasks.** Values are mean $R^2 \pm$ SD over sessions. Best models are in boldface (1st) and underlined (2nd). *: $p < 0.05$ on a paired t-test over sessions vs. the best non-MOJO baseline. †Results reproduced from Ryoo et al. [14].

	Same animal, other days	*New animal*	
Method	C – CO 2010 (5)	T – CO (6)	T – RT (6)
MLP†	0.5842 ± 0.2052	0.7940 ± 0.0341	0.6082 ± 0.3014
GRU†	0.7742 ± 0.0964	0.8389 ± 0.0248	0.7414 ± 0.0426
POYO (UI)	0.7759 ± 0.1003	0.8123 ± 0.0419	0.7011 ± 0.0975
POMAMBA (UI)	0.7283 ± 0.1138	0.8574 ± 0.0225	0.7283 ± 0.0846
POGRU (UI)	0.7632 ± 0.1013	0.8587 ± 0.0216	0.7331 ± 0.0775
MOJO-POYO (UI)	0.7846 ± 0.0897	0.8387 ± 0.0251	0.7470 ± 0.0617
MOJO-POMAMBA (UI)	<u>0.7937</u> ± 0.0735*	<u>0.8753</u> ± 0.0160*	**0.7591** ± 0.0588*
MOJO-POGRU (UI)	**0.7949** ± 0.0892*	**0.8772** ± 0.0189*	<u>0.7570</u> ± 0.0693*
NDT-2† (FT)	0.7846 ± 0.1167	0.7173 ± 0.0443	0.6323 ± 0.1339
NDT-3 (FT)	0.7524 ± 0.1322	0.8576 ± 0.0313	0.7066 ± 0.0980
NEDS (FT)	0.5968 ± 0.0760	0.7635 ± 0.0758	0.6121 ± 0.0918
POYO (FT)	<u>0.8244</u> ± 0.0753	0.8817 ± 0.0352	0.7624 ± 0.0815
POMAMBA (FT)	0.8142 ± 0.0763	0.8949 ± 0.0152	0.7580 ± 0.0745
POGRU (FT)	0.8126 ± 0.0892	0.8936 ± 0.0212	0.7575 ± 0.0875
MOJO-POYO (FT)	**0.8438** ± 0.0888*	**0.9131** ± 0.0177*	**0.7964** ± 0.0663*
MOJO-POMAMBA (FT)	0.8153 ± 0.0925	0.9043 ± 0.0182*	0.7675 ± 0.0806
MOJO-POGRU (FT)	0.8222 ± 0.0950	<u>0.9103</u> ± 0.0168*	<u>0.7776</u> ± 0.0710

Finetuning on Labelled Data. We first evaluate MOJO in the standard setting, where held-out sessions are fully labelled for finetuning. Table 1 shows decoding performance when pretrained models are transferred to new sessions of a seen animal and to a previously unseen animal. We compare against purely supervised spike-tokenizing models (POGRU, POMAMBA, and POYO) and SSL-based decoders that bin spikes (NDT-2 [27], NDT-3 [28], and NEDS [16]). MOJO achieves the best performance on all three evaluation sets, and every MOJO model outperforms its purely supervised counterpart under the same finetuning strategy (FT or UI) on every evaluation set. On the most challenging sessions (unseen monkey T, RT task), UI alone with MOJO-POSSM (<18K trainable parameters) matches fully finetuned supervised POSSM (>7.6M trainable parameters). Since UI updates only the unit and session embeddings, this suggests that much of MOJO's advantage is carried by its embeddings. Complete results, including other POSSM backbones and single-session models, are in Table 12.

Few-shot Finetuning. MOJO also enables efficient few-shot adaptation. Figure 2a compares MOJO-POGRU with POGRU when finetuned on only a few labelled trials from new sessions of an unseen animal (see Tables 20–22 for complete results). MOJO consistently outperforms the supervised baseline in the low-data regime, with the largest gains when labels are scarcest. MOJO can also use the remaining, unlabelled trials of the new session during finetuning. These trials let the SSL objective shape the new session's unit embeddings, which are learned from scratch, without requiring labels. Averaged over the six RT sessions of monkey T, MOJO with unlabelled data reaches 68.6%, 79.4%, 86.6%, and 93.9% of the 32-shot performance using 2, 4, 8, and 16 labelled trials, and needs around 4, 11, and 18 labelled trials to reach 80%, 90%, and 95%, compared with around 8, 15, and 26 for POGRU.

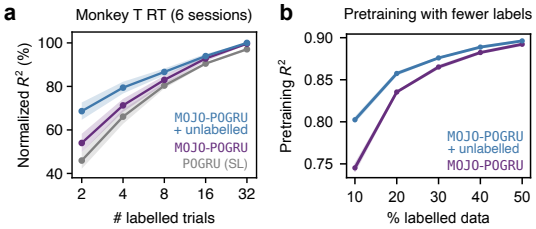


Figure 2: **Leveraging unlabelled data for finetuning and pretraining.** (a) With k labelled trials, MOJO improves few-shot finetuning over SL, and improves further by using the remaining $32 - k$ trials without labels. R^2 normalized to per-session 32-shot MOJO + unlabelled; mean ± SEM over 6 sessions. (b) When only a fraction of the pretraining data is labelled, MOJO improves decoding by also exploiting the remaining unlabelled data, especially when labels are scarce.

Finetuning with Random Behaviours. We further evaluate MOJO in a more challenging finetuning setting: all spiking data are retained, but behavioural labels are restricted to a sparse sub-phase (RANDOM) outside the cued reaching periods of each CO session, where any movements are self-initiated and likely akin to fidgeting. Evaluated on the reaching periods, MOJO achieves an average R^2 of 0.52 on new sessions of a seen animal, and $R^2 \geq 0.15$ in 3 of the 6 new sessions of an unseen animal. These results suggest that a pretrained MOJO can decode behaviour in structured tasks by extrapolating from the same behavioural variables observed in other periods. Appendix D.6 describes the phases of the CO sessions and gives further results on transfer across phases.

Unlabelled Data During Pretraining. MOJO can also leverage unlabelled data during pretraining. On the Perich et al. [29] dataset, we removed the behavioural labels from part of the pretraining data, keeping 10% to 50% labelled, and compared MOJO pretrained on all of the data with MOJO pretrained on the labelled part only (see Figure 2b). When labels are limited, adding the unlabelled data improves decoding of held-out data from the pretraining sessions, and performance increases steadily with the amount of unlabelled data (see Table 26). Brain region classification from the unit embeddings also improves. These gains do not carry over to finetuning on new sessions when no unlabelled data from those sessions is used, consistent with the gains residing largely in the session-specific unit embeddings (see Appendix D.7). Furthermore, even without any labels, pretraining remains useful: an SSL-only model finetunes better than single-session training (R^2 of 0.829 vs. 0.808), approaching SL-only pretraining (0.832), while joint pretraining performs best (0.845; see Table 16).

Unit Embedding Analyses. We probed the unit embeddings learned during pretraining for their associated metadata (brain region and subject ID) using logistic regression. Probing was done within each dataset (units from one dataset) and across datasets (units pooled from all datasets). Linear probes recover both brain region and subject far more accurately from MOJO embeddings than from those of purely supervised models (e.g., 95–99% vs. 72–75% for M1 vs. PMd on Churchland et al. [30]), although the model has no access to electrode, region, or subject metadata during pretraining (see Table 13).

3.2 Mouse Vision Tasks

In the second set of experiments, we evaluate MOJO on decoding visual stimuli from multi-region mouse recordings in the Allen Visual Coding Neuropixels dataset [31, 32], where Neuropixels probes recorded spiking activity across many brain regions, including multiple areas of the visual cortex, thalamus, and hippocampus. The 58 sessions span four genotypes (wild-type and three transgenic lines) and two stimulus sets. We hold out eight sessions for evaluation, one per genotype and stimulus set, and use the remaining 50 for pretraining. We consider three visual decoding tasks spanning natural and artificial stimuli: classification of natural scenes (NS), drifting-grating orientation (DG), and drifting-grating temporal frequency (TF). NS stimuli appear in only one stimulus set, and hence in four of the eight held-out sessions.

Table 2: **Natural scenes classification results on mouse vision tasks.** Values are mean accuracy $\pm$ SD over sessions.

Method	NS (4) Acc. (%)
MLP	83.17 ± 4.92
NEDS (FT)	93.96 ± 2.38
POGRU (FT)	91.09 ± 5.64
POYO (FT)	91.03 ± 4.07
MOJO-POGRU (FT)	94.47 ± 2.83
MOJO-POYO (FT)	94.12 ± 2.79
MOJO-POYO (A) (FT)	94.05 ± 2.51
MOJO-POYO (J) (FT)	94.62 ± 2.74
MOJO-POYO-L(J) (FT)	95.48 ± 2.05

In this experiment, we evaluate MOJO-POYO and MOJO-POGRU. All models see the entire trial before making a prediction (i.e., for POSSM, we place the behaviour timestamp at the end of each trial). As in the monkey experiments, we use 1 s sequences and 20 ms time chunks. We also pretrain MOJO with data beyond the three decoding tasks. Variants marked (A) use all data from the 50 pretraining sessions, including unlabelled periods with other stimuli or no stimulus, and variants marked (J) additionally include the monkey datasets of Section 3.1. For (J), we use a curriculum in which MOJO is first pretrained on the monkey datasets and then jointly on both, while -L denotes a larger model. Complete results are in Table 28. Furthermore, in a few-shot experiment, we find that unlabelled spontaneous activity helps as much as unlabelled trials of the downstream task (Appendix E.4).

Unit Embedding Analyses. We linearly probed the trained unit embeddings from all 50 pretraining sessions for their associated brain regions. A single classifier is trained across all animals, so it must capture region structure shared across subjects. We included 7 regions that are well represented

among the spike-sorted units and together cover the main areas in the dataset: cornu ammonis 1 (CA1), dentate gyrus (DG), lateral posterior nucleus (LP), dorsal lateral geniculate nucleus (LGd), primary visual cortex (VISp), anterolateral visual area (VISal), and anteromedial visual area (VISam). Figure 4a–b shows the confusion matrices of linear classifiers trained on MOJO-POYO-L(J) embeddings, for 3 broader areas (visual cortex, hippocampus, and thalamus) and for all 7 regions. Despite inter-subject variability, most of the probability mass lies on the diagonal, and most errors occur between anatomically or functionally related regions.

Beyond categorical region identity, we asked whether the unit embeddings capture single-neuron firing properties (see Figures 3 and 7, Appendix E.3). On the same 50 pretraining sessions, we linearly regressed 18 spike-statistic features per neuron [33] from the unit embeddings: moments of the inter-spike interval (ISI) distribution, firing rate, variability indices (CV, LV), gamma-distribution fits, and band-limited spike-train PSDs. Each probe is fitted on the neurons of 49 sessions and evaluated on all neurons of the held-out session. This tests a property the SSL objective is designed to provide: because the SSL pathway predicts each unit's spike counts through a query built from its unit embedding, it encourages the embeddings to capture each unit's firing patterns, which supervised training alone does not require. Consistent with this, MOJO embeddings predict every target better than those of purely supervised POYO, with the largest gains on rate-related features (e.g., for MOJO-POYO-L(J), $R^2 = 0.88$ vs. 0.53 for log mean firing rate and 0.60 vs. 0.33 for log median ISI) and smaller gains on rate-normalized measures (e.g., 0.19 vs. -0.02 for $\log(1 + \mathrm{CV})$).

Together, these analyses show that SSL makes unit embeddings capture each neuron's firing properties more faithfully, and its brain region beyond what such properties alone provide: 7-region classification from MOJO embeddings exceeds a baseline built from the same 18 hand-crafted features (see Table 29). This richer neuron-level structure may also help explain MOJO's strong UI performance, as UI adapts pretrained models to a new session by learning only unit and session embeddings. Note that unlike previous work [26], which classified session-averaged latent outputs of the input cross-attention, we probe the unit embeddings directly. Classification accuracy also improves as more unlabelled data is added when training MOJO-POYO: adding unlabelled data from the same visual coding dataset improved 7-region classification by 5.5 percentage points, and adding monkey data improved it by a further 2.5 points without sacrificing decoding performance (see Table 2). A larger joint model adds another 3.6 points, along with a small gain in NS accuracy on the held-out sessions. See Appendix E.2 and Table 29 for further probing results, and Table 28 for complete decoding results on the mouse vision tasks, including single-session models.

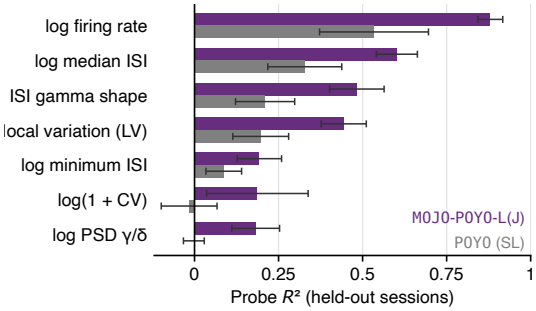


Figure 3: **Probing single-neuron firing statistics from unit embeddings.** Leave-one-session-out linear probes of seven key spike-statistic features from the unit embeddings of MOJO-POYO-L(J) and purely supervised POYO. We plot mean $R^2 \pm$ SD over 50 held-out sessions.

Joint Monkey-Mouse Pretraining. Joint monkey-mouse pretraining showed positive transfer to the mouse data, in the form of faster learning: NS validation accuracy after 10, 50, 100, and 200 epochs was 33%, 70%, 75%, and 78% with joint pretraining, compared with 2%, 35%, 63%, and 74% for mouse-only pretraining. Meanwhile, performance on the monkey reaching tasks was maintained. Joint pretraining also improved brain region classification from the unit embeddings, as shown above. Given the limited overlap between monkey motor and mouse visual data, we view these as gains in convergence speed and representation quality rather than a higher performance ceiling. Further details are in Appendix E.5.

3.3 Mouse Decision-Making Tasks

Our third set of experiments uses the IBL Reproducible Electrophysiology dataset [34], in which multiple labs recorded multi-region Neuropixels activity with probes targeting the same brain locations (the IBL "repeated site") during a visually driven decision-making task. Following Zhang et al. [16], we decode four behaviours (choice, block prior, wheel speed, and whisker motion energy), pretrain

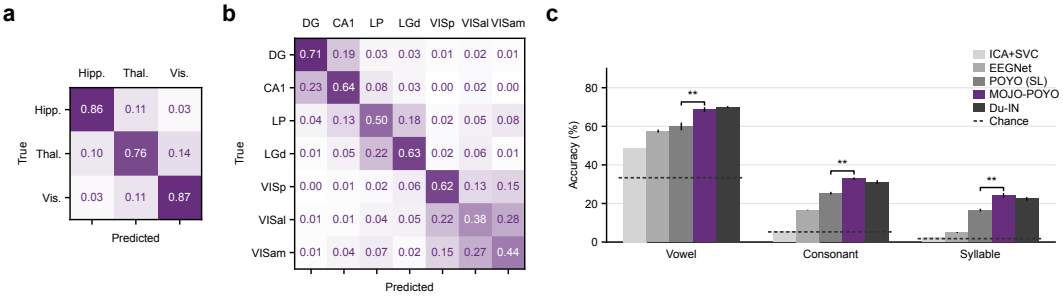


Figure 4: **Mouse brain region classification and human speech decoding results.** (a–b) Confusion matrices for (a) 3-class and (b) 7-class brain region classification of neurons in mice from the Allen Visual Coding Neuropixels dataset, using MOJO-POYO-L(J) unit embeddings, shown as row-normalized proportions averaged over 5 seeds. (c) Classification accuracy of vowels, consonants, and syllables for speech decoding from human electrocorticography, averaged over participants and 3 seeds; ICA+SVC and EEGNet are classical baselines. **: $p < 0.01$ for a paired t-test of MOJO-POYO vs. POYO across all participants and seeds; the difference between MOJO-POYO and Du-IN is not significant.

on 74 of the 84 sessions, and finetune on 10 pre-determined held-out sessions. As this dataset was previously evaluated non-causally, we use the POYO backbone for MOJO. Unlike Zhang et al. [16], for both MOJO and POYO we do not enforce trial alignment or exclude neurons by firing rate during training. However, behavioural labels still come only from the pre-determined intervals. Each interval is 2 s long, and we train with 1.6 s sequences and 20 ms time chunks.

Results. As shown in Table 3, MOJO-POYO outperforms purely supervised POYO on all four tasks, significantly so for choice, wheel speed, whisker motion energy, and their average. The session splits and spike sorting of this dataset have changed since Zhang et al. [16], so we reran NEDS with its publicly released code on the updated dataset to ensure a fair comparison. We note that we were unable to reproduce the performance reported in their work.

Table 3: **Mouse decision-making task results.** Values for choice and block are mean balanced accuracy and for wheel and whisker R^2, all with $\pm$ SD over sessions. *: $p < 0.05$ on a paired t-test over sessions vs. POYO.

Task	NEDS	POYO	MOJO-POYO
Choice	0.820 ± 0.12	0.882 ± 0.09	0.913 ± 0.07*
Block	0.812 ± 0.09	0.848 ± 0.05	0.864 ± 0.04
Wheel	0.563 ± 0.08	0.630 ± 0.10	0.665 ± 0.09*
Whisker	0.507 ± 0.10	0.550 ± 0.11	0.581 ± 0.09*
Avg.	0.647 ± 0.09	0.727 ± 0.07	0.756 ± 0.05*

3.4 Human Speech Tasks

Finally, we test whether MOJO extends beyond spiking data using a human ECoG speech articulation dataset [35, 36], recorded from four participants (30 sessions in total) speaking consonant–vowel syllables. ECoG signals were recorded with a high-density 256-channel array implanted in ventral sensorimotor cortex during epilepsy treatment, while participants read commonly used syllables in American English. Following the original study, we extract the high-gamma band and downsample it to 200 Hz. Decoders are trained jointly to predict the vowel, the consonant, and the full syllable of each trial. For each trial, we use a 1.2 s window centred on the consonant–vowel transition time provided in the dataset (from 550 ms before to 650 ms after). Each training sample is a 1 s interval drawn from one trial and divided into 20 ms patches (i.e., patching the 200 Hz signal with a patch size of 4). To handle continuous signals, we use the POYO backbone with value embeddings [26].

Results. We train MOJO on all sessions from every participant and plot test accuracies in Figure 4c. Compared with purely SL-trained POYO, MOJO improves performance on all three tasks, most of all on the hardest one (a 47% relative improvement for syllable classification). MOJO also performs on par with Du-IN [37], a recent intracranial EEG foundation model, despite being designed for spike data. Overall, these results suggest that MOJO extends to neural modalities beyond spikes, highlighting its potential for future multi-modal training.

4 Related Work

Traditional Neural Decoders. Historically, neural decoding for continuous tasks such as cursor velocity decoding in BCIs was accomplished using statistical methods such as the Kalman filter [38–41]. Subsequent work also used simple artificial neural networks such as MLPs [1] and RNNs [42] with great success in single-session settings. With the advent of more sophisticated deep learning methods such as variational RNNs and sequential VAEs [43], LFADS [44] was proposed to extract temporally varying latent variables from high-dimensional neural spiking data. Such models have also been used to perform neural decoding tasks in real time [3]. More recently, NoMAD [45] adapted LFADS to also decode behavioural variables, using unsupervised alignment of latent dynamics to stabilize decoding across days.

Pretraining for Neural Decoding. With advances in large-scale deep learning models for sequential data [5, 7, 46] and the availability of large public datasets of neural recordings, many recent works have proposed neural decoding models that combine scalable architectures such as Transformers or SSMs with large-scale pretraining. These models achieve state-of-the-art decoding performance and generalize to new sessions, tasks, and subjects. Pretrained spike decoders fall broadly into two categories, depending on how they ingest neural data. The first category, NDT-style models such as NDT-2 [27], NDT-MtM [47], NDT-3 [28], and NEDS [16], processes binned or patched spike counts to decode behaviour. A key feature of these models is the incorporation of SSL: NDT-2 and NDT-MtM carry out SSL pretraining followed by downstream finetuning, NEDS is jointly trained on neural encoding and decoding with several masking schemes, and NDT-3 autoregressively predicts neural data and behaviour in a flat sequence.

The second category, POYO-style models, processes each neural spike as a separate token, allowing these models to exploit fine-grained temporal information and facilitating large-scale pretraining and efficient finetuning [13, 14]. These models have achieved state-of-the-art performance on several neural decoding tasks and demonstrated transfer of decoders to new sessions, tasks, subjects, and even species [14]. Beyond spikes, POYO-style architectures have been applied to calcium traces for decoding [26] and forecasting [23], treating fine-grained temporal patches as tokens. Most closely related to our work, Willeke et al. [24] proposed OmniMouse, a POYO-style model for calcium imaging data from mouse visual cortex that combines neural, behavioural, and video inputs. Their model is trained with many masking configurations, in the spirit of NDT-MtM and NEDS, to jointly support forecasting, encoding, and behavioural decoding, with a focus on multi-modal modelling and scaling. For spikes, however, POYO-style models have so far been trained only with supervised objectives and cannot leverage unlabelled data, unlike NDT-style models and masked autoencoders for other neural signals such as EEG [48, 49], stereo-EEG [11, 12, 37], and ECoG [50]. MOJO aims to fill this gap for spike-tokenizing models and study the value of learning from unlabelled data in this setting.

5 Discussion

Our results demonstrate that MOJO, our joint SSL-SL training framework, improves decoding performance, enables more label-efficient finetuning, and yields more informative unit embeddings across diverse datasets compared to purely supervised POYO-style models. These gains come with minimal parameter overhead (e.g., only 231K of the 9.89M parameters of MOJO-POYO), since we share most parameters between the SSL and SL objectives. This places MOJO in a strong position to harness large amounts of unlabelled spiking data across sessions, tasks, and species, while retaining the scalability and flexibility of POYO-style tokenization. Finally, we demonstrate that MOJO is applicable beyond spiking data, paving the way for unified and scalable neuro-foundation models.

Like other masked modelling approaches [24, 51], MOJO is data-hungry: it needs sufficient data to learn the underlying neural dynamics, and when trained on a single session, it can underperform purely supervised training (see Table 28). A poorly learned SSL objective yields a poor model of the neural dynamics, which can in turn hurt decoding performance. While our latent masking scheme integrates easily with the SL pathway, it is limited to temporal masking; an objective requiring spatial inference (e.g., predicting unseen neurons or brain regions) could further strengthen SSL. Moreover, much of the benefit of SSL comes through better unit embeddings, yet these are relearned from scratch for each new session, as spike-sorted units cannot be matched across sessions. This limitation, inherited from the POYO family, is especially wasteful with SSL, and may explain why additional

unlabelled pretraining data alone yields mixed downstream gains (none on new monkey reaching sessions and small gains with UI on the mouse vision tasks), whereas unlabelled data from the new session itself consistently improves few-shot finetuning. Lastly, applying MOJO to ECoG requires an additional linear layer for value embeddings, and this modality-specific design limits which datasets can be pooled during pretraining.

To address these limitations, future work will explore joint multi-modal training [16, 24]. Datasets from different modalities capture diverse aspects of neural function, so combining them would enrich pretraining, and prediction across modalities could encourage a more complete understanding of the underlying neural dynamics [52, 53]. To avoid relearning unit embeddings for each new session, amortized methods [54] and discrete codes [9] are promising directions for reusing the learned model when embedding new units. Overall, MOJO shows that spike-tokenizing decoders can benefit from unlabelled neural data, marking a step towards neuro-foundation models that make full use of the large amounts of unlabelled recordings already available.

Broader Impact

The method proposed in this paper could contribute to the advancement of brain-computer interfaces that restore motor and speech function. The datasets used in this work are publicly available and originate from animal and human experiments involving surgical implantation, which raise ethical concerns common to many medical technologies, including privacy protection, harm reduction, and animal welfare. Any new data collection for such systems must comply with the relevant institutional review and animal care protocols. If such systems are deployed in real-world settings, decoding errors must be minimized to avoid harming users, and privacy concerns must be carefully addressed. Therefore, rigorous testing and comprehensive evaluation are required before any practical deployment.

Software and Data

Our code is publicly available through our project page: mojo-brain.github.io. All datasets used in this work are already publicly available.

Acknowledgments and Disclosure of Funding

NHK acknowledges financial support from the Fonds de recherche du Québec (Secteur NT, grant 2009130). MGP acknowledges support of a J1 Chercheurs-boursiers en intelligence artificielle and a J2 Chercheurs-boursiers from the Fonds de recherche du Québec – Santé. GL acknowledges support from NSERC Discovery Grant RGPIN-2018-04821, the Canada Research Chair in Neural Computations and Interfacing, a Canada CIFAR AI Chair, IVADO, and the Canada First Research Excellence Fund. The authors also acknowledge the support of computational resources provided by Mila, the Digital Research Alliance of Canada, and NVIDIA that enabled this research.

References

[1] J. I. Glaser, A. S. Benjamin, R. H. Chowdhury, M. G. Perich, L. E. Miller, and K. P. Kording. "Machine Learning for Neural Decoding". *eNeuro* 7.4 (2020), ENEURO.0506–19.2020.

[2] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1724–1734.

[3] Y. H. Ali, K. Bodkin, M. Rigotti-Thompson, K. Patel, N. S. Card, B. Bhaduri, S. R. Nason-Tomaszewski, D. M. Mifsud, X. Hou, C. Nicolas, et al. "BRAND: a platform for closed-loop experiments with deep network models". *Journal of Neural Engineering* 21.2 (2024), p. 026046.

[4] D. Sussillo, S. D. Stavisky, J. C. Kao, S. I. Ryu, and K. V. Shenoy. "Making brain–machine interfaces robust to future neural variability". *Nature Communications* 7.1 (2016), p. 13749.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is All you Need". *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[6] A. Gu and T. Dao. "Mamba: Linear-Time Sequence Modeling with Selective State Spaces". *First Conference on Language Modeling*. 2024.

[7] T. Dao and A. Gu. "Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality". *Proceedings of the 41st International Conference on Machine Learning*. Vol. 235. Proceedings of Machine Learning Research. 2024, pp. 10041–10071.

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". *International Conference on Learning Representations*. 2021.

[9] W. Jiang, L. Zhao, and B.-l. Lu. "Large Brain Model for Learning Generic Representations with Tremendous EEG Data in BCI". *The Twelfth International Conference on Learning Representations*. 2024.

[10] B. Lenz, O. Lieber, A. Arazi, A. Bergman, A. Manevich, B. Peleg, B. Aviram, C. Almagor, C. Fridman, D. Padnos, et al. "Jamba: Hybrid Transformer-Mamba Language Models". *The Thirteenth International Conference on Learning Representations*. 2025.

[11] D. Zhang, Z. Yuan, Y. Yang, J. Chen, J. Wang, and Y. Li. "Brant: Foundation Model for Intracranial Neural Signal". *Advances in Neural Information Processing Systems*. Vol. 36. 2023, pp. 26304–26321.

[12] C. Wang, V. Subramaniam, A. U. Yaari, G. Kreiman, B. Katz, I. Cases, and A. Barbu. "BrainBERT: Self-supervised representation learning for intracranial recordings". *The Eleventh International Conference on Learning Representations*. 2023.

[13] M. Azabou, V. Arora, V. Ganesh, X. Mao, S. Nachimuthu, M. Mendelson, B. Richards, M. Perich, G. Lajoie, and E. Dyer. "A Unified, Scalable Framework for Neural Population Decoding". *Advances in Neural Information Processing Systems*. Vol. 36. 2023, pp. 44937–44956.

[14] A. H.-W. Ryoo, N. H. Krishna, X. Mao, M. Azabou, E. L. Dyer, M. G. Perich, and G. Lajoie. "Generalizable, real-time neural decoding with hybrid state-space models". *Advances in Neural Information Processing Systems*. Vol. 38. 2025, pp. 51764–51791.

[15] J. Ye and C. Pandarinath. "Representation learning for neural population activity with Neural Data Transformers". *Neurons, Behavior, Data analysis, and Theory* 5.3 (2021), pp. 1–18.

[16] Y. Zhang, Y. Wang, M. Azabou, A. Andre, Z. Wang, H. Lyu, International Brain Laboratory, E. L. Dyer, L. Paninski, and C. L. Hurwitz. "Neural Encoding and Decoding at Scale". *Proceedings of the 42nd International Conference on Machine Learning*. Vol. 267. Proceedings of Machine Learning Research. 2025, pp. 76175–76192.

[17] I. Goodfellow, Y. Bengio, and A. Courville. "Deep Learning". 2016.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 4171–4186.

[19] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. "Language Models are Few-Shot Learners". *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 1877–1901.

[20] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. "A Simple Framework for Contrastive Learning of Visual Representations". *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research. 2020, pp. 1597–1607.

[21] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. "Masked Autoencoders Are Scalable Vision Learners". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 16000–16009.

[22] M. Assran, Q. Duval, I. Misra, P. Bojanowski, P. Vincent, M. Rabbat, Y. LeCun, and N. Ballas. "Self-Supervised Learning From Images With a Joint-Embedding Predictive Architecture". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 15619–15629.

[23] Y. Duan, H. T. Chaudhry, M. B. Ahrens, C. Harvey, M. G. Perich, K. Deisseroth, and K. Rajan. "POCO: Scalable Neural Forecasting through Population Conditioning". *Advances in Neural Information Processing Systems*. Vol. 38. 2025, pp. 23631–23669.

[24] K. F. Willeke, P. Turishcheva, A. Gilbert, G. Chakrabarty, H. A. Bedel, P. G. Fahey, Y. Qiu, M. A. Weis, M. Vystrčilová, T. Muhammad, et al. "OmniMouse: Scaling properties of multi-modal, multi-task Brain Models on 150B Neural Tokens". *The Fourteenth International Conference on Learning Representations*. 2026.

[25] A. Baevski, H. Zhou, A. Mohamed, and M. Auli. "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations". *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 12449–12460.

[26] M. Azabou, K. X. Pan, V. Arora, I. J. Knight, E. L. Dyer, and B. A. Richards. "Multi-session, multi-task neural decoding from distinct cell-types and brain regions". *The Thirteenth International Conference on Learning Representations*. 2025.

[27] J. Ye, J. Collinger, L. Wehbe, and R. Gaunt. "Neural Data Transformer 2: Multi-context Pretraining for Neural Spiking Activity". *Advances in Neural Information Processing Systems*. Vol. 36. 2023, pp. 80352–80374.

[28] J. Ye, F. Rizzoglio, X. Ma, A. Smoulder, H. Mao, G. Blumenthal, W. Hockeimer, N. Kunigk, D. Moore, P. Marino, et al. "A Generalist Intracortical Motor Decoder". *Advances in Neural Information Processing Systems*. Vol. 38. 2025, pp. 110547–110587.

[29] M. G. Perich, L. E. Miller, M. Azabou, and E. L. Dyer. "Long-term recordings of motor and premotor cortical spiking activity during reaching in monkeys". DANDI Archive. Version 0.250122.1735. 2025.

[30] M. M. Churchland, J. P. Cunningham, M. T. Kaufman, J. D. Foster, P. Nuyujukian, S. I. Ryu, and K. V. Shenoy. "Neural population dynamics during reaching". *Nature* 487.7405 (2012), pp. 51–56.

[31] S. E. de Vries, J. H. Siegle, and C. Koch. "Sharing neurophysiology data from the Allen Brain Observatory". *eLife* 12 (2023), e85550.

[32] J. H. Siegle, X. Jia, S. Durand, S. Gale, C. Bennett, N. Graddis, G. Heller, T. K. Ramirez, H. Choi, J. A. Luviano, et al. "Survey of spiking in the mouse visual system reveals functional hierarchy". *Nature* 592.7852 (2021), pp. 86–92.

[33] A. Schneider, M. Azabou, L. McDougall-Vigier, D. F. Parks, S. Ensley, K. Bhaskaran-Nair, T. Nowakowski, E. L. Dyer, and K. B. Hengen. "Transcriptomic cell type structures in vivo neuronal activity across multiple timescales". *Cell Reports* 42.4 (2023), p. 112318.

[34] International Brain Laboratory, K. Banga, J. Benson, J. Bhagat, D. Biderman, D. Birman, N. Bonacchi, S. A. Bruijns, K. Buchanan, R. A. Campbell, et al. "Reproducibility of in vivo electrophysiological measurements in mice". *eLife* 13 (2025), RP100840.

[35] K. E. Bouchard, N. Mesgarani, K. Johnson, and E. F. Chang. "Functional organization of human sensorimotor cortex for speech articulation". *Nature* 495.7441 (2013), pp. 327–332.

[36] K. E. Bouchard and E. F. Chang. "Human ECoG speaking consonant-vowel syllables". figshare. 2019.

[37] H. Zheng, H.-T. Wang, W.-B. Jiang, Z.-T. Chen, L. He, P.-Y. Lin, P.-H. Wei, G.-G. Zhao, and Y.-Z. Liu. "Du-IN: Discrete units-guided mask modeling for decoding speech from Intracranial Neural signals". *Advances in Neural Information Processing Systems*. Vol. 37. 2024, pp. 79996–80033.

[38] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems". *Journal of Basic Engineering* 82.1 (1960), pp. 35–45.

[39] W. Wu, M. J. Black, Y. Gao, E. Bienenstock, M. Serruya, A. Shaikhouni, and J. P. Donoghue. "Neural Decoding of Cursor Motion Using a Kalman Filter". *Advances in Neural Information Processing Systems*. Vol. 15. 2002.

[40] S. Koyama, S. M. Chase, A. S. Whitford, M. Velliste, A. B. Schwartz, and R. E. Kass. "Comparison of brain–computer interface decoding algorithms in open-loop and closed-loop control". *Journal of Computational Neuroscience* 29.1-2 (2010), pp. 73–87.

[41] F. R. Willett, D. R. Young, B. A. Murphy, W. D. Memberg, C. H. Blabe, C. Pandarinath, S. D. Stavisky, P. Rezaii, J. Saab, B. L. Walter, et al. "Principled BCI Decoder Design and Parameter Selection Using a Feedback Control Model". *Scientific Reports* 9.1 (2019), p. 8881.

[42] D. Sussillo, P. Nuyujukian, J. M. Fan, J. C. Kao, S. D. Stavisky, S. Ryu, and K. Shenoy. "A recurrent neural network for closed-loop intracortical brain–machine interface decoders". *Journal of Neural Engineering* 9.2 (2012), p. 026027.

[43] J. Chung, K. Kastner, L. Dinh, K. Goel, A. Courville, and Y. Bengio. "A Recurrent Latent Variable Model for Sequential Data". *Advances in Neural Information Processing Systems*. Vol. 28. 2015.

[44] C. Pandarinath, D. J. O'Shea, J. Collins, R. Jozefowicz, S. D. Stavisky, J. C. Kao, E. M. Trautmann, M. T. Kaufman, S. I. Ryu, L. R. Hochberg, et al. "Inferring single-trial neural population dynamics using sequential auto-encoders". *Nature Methods* 15.10 (2018), pp. 805–815.

[45] B. M. Karpowicz, Y. H. Ali, L. N. Wimalasena, A. R. Sedler, M. R. Keshtkaran, K. Bodkin, X. Ma, D. B. Rubin, Z. M. Williams, S. S. Cash, et al. "Stabilizing brain-computer interfaces through alignment of latent dynamics". *Nature Communications* 16.1 (2025), p. 4662.

[46] A. Gu, K. Goel, and C. Re. "Efficiently Modeling Long Sequences with Structured State Spaces". *International Conference on Learning Representations*. 2022.

[47] Y. Zhang, Y. Wang, D. M. Jiménez-Benetó, Z. Wang, M. Azabou, B. Richards, R. Tung, O. Winter, The International Brain Laboratory, E. Dyer, et al. "Towards a "Universal Translator" for Neural Dynamics at Single-Cell, Single-Spike Resolution". *Advances in Neural Information Processing Systems*. Vol. 37. 2024, pp. 80495–80521.

[48] H.-Y. S. Chien, H. Goh, C. M. Sandino, and J. Y. Cheng. "MAEEG: Masked Auto-encoder for EEG Representation Learning". NeurIPS 2022 Workshop on Learning from Time Series for Health. 2022. arXiv: 2211.02625.

[49] D. Wu, S. Li, J. Yang, and M. Sawan. "Neuro-BERT: Rethinking Masked Autoencoding for Self-Supervised Neurological Pretraining". *IEEE Journal of Biomedical and Health Informatics* 30.2 (2026), pp. 970–980.

[50] A. R. Samar and J. G. Makin. "Pretraining with Masked Autoencoding Improves Speech Decoding from ECoG". *2025 12th International IEEE/EMBS Conference on Neural Engineering (NER)*. 2025, pp. 998–1003.

[51] Z. Xie, Z. Zhang, Y. Cao, Y. Lin, Y. Wei, Q. Dai, and H. Hu. "On Data Scaling in Masked Image Modeling". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 10365–10374.

[52] C. Fang, C. M. Sandino, B. Mahasseni, J. Minxha, H. Pouransari, E. Azemi, A. Moin, and E. L. Zippi. "Promoting cross-modal representations to improve multimodal foundation models for physiological signals". *Advancements In Medical Foundation Models: Explainability, Robustness, Security, and Beyond*. 2024.

[53] S. Gupta, S. Sundaram, C. Wang, S. Jegelka, and P. Isola. "Better Together: Leveraging Unpaired Multimodal Data for Stronger Unimodal Models". *The Fourteenth International Conference on Learning Representations*. 2026.

[54] V. Arora, D. Lachi, I. Knight, M. Azabou, B. Richards, C. Hurwitz, J. H. Siegle, and E. L. Dyer. "Know Thyself by Knowing Others: Learning Neuron Identity from Population Context". *Advances in Neural Information Processing Systems*. Vol. 38. 2025, pp. 99472–99501.

[55] R. D. Flint, E. W. Lindberg, L. R. Jordan, L. E. Miller, and M. W. Slutzky. "Accurate decoding of reaching movements from field potentials in the absence of spikes". *Journal of Neural Engineering* 9.4 (2012), p. 046006.

[56] J. E. O'Doherty, M. M. B. Cardoso, J. G. Makin, and P. N. Sabes. "Nonhuman Primate Reaching with Multichannel Sensorimotor Cortex Electrophysiology". Zenodo. 2020.

[57] F. Pei, J. Ye, D. Zoltowski, A. Wu, R. Chowdhury, H. Sohn, J. O'Doherty, K. V. Shenoy, M. Kaufman, M. Churchland, et al. "Neural Latents Benchmark '21: Evaluating latent variable models of neural population activity". *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*. Vol. 1. 2021.

[58] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. "RoFormer: Enhanced transformer with Rotary Position Embedding". *Neurocomputing* 568 (2024), p. 127063.

[59] A. Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer, et al. "Perceiver IO: A General Architecture for Structured Inputs & Outputs". *International Conference on Learning Representations*. 2022.

[60] L. N. Smith and N. Topin. "Super-convergence: very fast training of neural networks using large learning rates". *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*. Vol. 11006. Proc. SPIE. 2019, p. 1100612.

[61] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". *Advances in Neural Information Processing Systems*. Vol. 32. 2019.

[62] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. "Scikit-learn: Machine Learning in Python". *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[63] V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance. "EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces". *Journal of Neural Engineering* 15.5 (2018), p. 056013.

[64] Z. Zhang, S.-h. Zhong, and Y. Liu. "TorchEEGEMO: A deep learning toolbox towards EEG-based emotion recognition". *Expert Systems with Applications* 249 (2024), p. 123550.

[65] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C.-J. Hsieh. "Large Batch Optimization for Deep Learning: Training BERT in 76 minutes". *International Conference on Learning Representations*. 2020.

[66] I. Loshchilov and F. Hutter. "Decoupled Weight Decay Regularization". *International Conference on Learning Representations*. 2019.

[67] M. Beck, K. Pöppel, M. Spanring, A. Auer, O. Prudnikova, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter. "xLSTM: Extended Long Short-Term Memory". *Advances in Neural Information Processing Systems*. Vol. 37. 2024, pp. 107547–107603.

[68] A. van den Oord, Y. Li, and O. Vinyals. "Representation Learning with Contrastive Predictive Coding". 2018. arXiv: 1807.03748.

[69] F. R. Willett, D. T. Avansino, L. R. Hochberg, J. M. Henderson, and K. V. Shenoy. "High-performance brain-to-text communication via handwriting". *Nature* 593.7858 (2021), pp. 249–254.

Supplementary Material

A Additional Details on Datasets

A.1 Monkey Reaching Tasks

For experiments on the monkey reaching tasks, we adopted the same pretraining datasets and splits as Ryoo et al. [14]. For completeness, we list the details of these datasets in Table 4. We hold out 19 sessions from pretraining for evaluation: 2 sessions from monkey C (20161013 and 20161021) and all 12 sessions from monkey T in the Perich et al. [29] dataset, and all 5 sessions from Flint et al. [55]. These datasets consist primarily of neural recordings paired with cursor- or hand-velocity labels. For samples flagged as behavioural outliers, we discard the labels and use only the neural activity for self-supervised training.

Table 4: **Summary of the monkey reaching datasets.** Statistics reproduced from Ryoo et al. [14].

Dataset	Regions	Tasks	# Indiv.	# Sess.	# Units	# Spikes	# Bhvr.
Perich et al. [29]	M1, PMd	CO, RT	3	107	10 245	104.6M	14.0M
O'Doherty et al. [56]	M1, S1	RT	2	44	14 899	87.8M	10.4M
Churchland et al. [30]	M1, PMd	CO	2	10	1911	739M	85.0M
NLB Maze [57]	M1, PMd	Maze	1	1	182	3.64M	6.81M
Flint et al. [55]	M1	CO	1	5	957	7.88M	318K

A.2 Mouse Vision Tasks

The Allen Visual Coding Neuropixels dataset [31, 32] contains 58 sessions of mouse recordings during visual stimulation. Thirty sessions were recorded in wild-type mice and the remaining 28 in three transgenic lines: Pvalb (8), Sst (12), and Vip (8). In each session, the animal was shown one of two stimulus sets: "Brain Observatory 1.1" (set 1) or "Functional Connectivity" (set 2). Both sets include drifting gratings (2 s per trial), but only set 1 includes natural scenes (250 ms per image, with no grey period between images). In set 2, the drifting gratings have only 4 orientations and a single temporal frequency. We consider three classification tasks: drifting-grating orientation (8 classes), drifting-grating temporal frequency (5 classes), and natural scenes (119 classes). Besides the drifting-grating and natural-scene intervals, each session also contains periods with other visual stimuli and spontaneous blocks with no stimulus. We treat these periods as unlabelled neural data.

A.3 Mouse Decision-Making Tasks

The IBL Reproducible Electrophysiology dataset [34] contains 84 sessions of recordings from head-fixed mice performing a visual decision-making task, in which they report the location of a visual grating. The sessions come from 10 labs, with probes inserted to target the same brain locations. Following prior work on this dataset [16], we consider four behavioural tasks: two classification tasks, decoding choice (left or right) and block prior (the probability that the stimulus appears on the left: 0.2, 0.5, or 0.8); and two regression tasks, decoding wheel speed and whisker motion energy. The objective is to decode these variables within a fixed 2 s window aligned to stimulus onset (`stimOn_time`), from 500 ms before to 1500 ms after, with the continuous variables resampled to 50 Hz. We use the same data splits as Zhang et al. [16], following their publicly available code.

Our data preprocessing differs from Zhang et al. [16] in three ways. First, we do not enforce trial alignment during training: the training intervals are the union of the fixed windows and the full trial durations, extended by 200 ms on each side. The unlabelled data outside the fixed windows are used for self-supervised training. Second, we do not exclude neurons based on firing rate, and instead include spikes from all recorded neurons. Third, instead of min-max normalization of wheel speed and whisker motion energy, we leave the former unnormalized and scale the latter by a constant factor of 2.5, chosen so that the two variables have comparable magnitudes.

A.4 Human Speech Tasks

The human ECoG speech dataset [35, 36] contains 30 sessions of ECoG recordings from four participants reading aloud consonant–vowel (CV) syllables shown on a screen. Each trial pairs the ECoG recording with a syllable label, within a 1.2 s window centred on the consonant–vowel transition. Across participants, the dataset contains 57 syllable types, formed from 19 consonants and 3 vowels, and each with a similar number of repetitions. A small number of trials, mostly from participant EC9, carry labels outside these 57 types. During preprocessing, we assign every syllable type a unique index, including these outliers, but use only the 57 standard types as decoding targets. We split the data temporally, using the last 20% of trials in each session for testing. Due to some adjacent trial windows overlapping, we drop any trial that overlaps with a test trial to prevent leakage of neural activity into the test set. We split the remaining trials 9:1 into training and validation sets.

B Model Architectures

B.1 Spike Tokenization

The spike tokenization scheme [13] represents each spike as a token consisting of two components – the neural unit that emitted it and the time at which it occurred:

$$\mathbf{x} = (\mathrm{UnitEmb}(I_{\mathrm{spike}}),\, T_{\mathrm{spike}})\,,$$

where $\mathrm{UnitEmb}(\cdot)$ is a lookup table mapping each neural unit ID to a D-dimensional embedding, and T_{spike} is the spike timestamp, which is encoded with rotary positional embeddings (RoPE) [58]. Following prior work, we use "neural unit" as a general term whose meaning depends on the dataset: an individual neuron, multi-units from a single electrode, or a recording channel. For ECoG, the signal from each channel within each time patch is embedded by an additional linear layer and concatenated with the unit embedding [26]:

$$\mathbf{x} = \left(\left[\mathrm{UnitEmb}(I_{\mathrm{ch}}),\, \mathbf{W}_s\, \mathbf{s}_{\mathrm{ch,patch}}\right],\, T_{\mathrm{patch}}\right),$$

where $\mathbf{s}_{\mathrm{ch,patch}}$ is the signal from channel ch within the patch and T_{patch} is the patch timestamp. No metadata about a unit or channel (e.g., its brain region) is encoded beyond an arbitrary integer ID.

Following `POYO`, we also feed the encoder one special token per recorded unit so that the model receives information about units that did not fire. `POYO` uses two sets of these tokens as delimiters timestamped at the start and end of the sequence. Instead, we place one set in every time chunk and timestamp it at the midpoint of the chunk.

B.2 Input Cross-Attention

Like `POYO` and its variants, we use a cross-attention module to encode a variable-length input token sequence into a fixed-length sequence of N_z latent tokens. N_z is a hyperparameter, typically chosen to be much smaller than the number of input tokens, so the encoder compresses the input. The latent sequence can be viewed as T_c groups of N_c tokens (hence $N_z = N_c T_c$), where each group corresponds to one contiguous *time chunk* of the sample.

`MOJO` employs a `POSSM`-style encoder [14], in which cross-attention is computed separately within each time chunk. Given a sequence of spike tokens $\mathbf{X}_t$ in chunk t and the i-th learnable latent query $\mathbf{c}_i^{(t)}$, the encoder output is computed using scaled dot-product attention [5]:

$$\mathbf{z}_i^{(t)} = \mathrm{softmax}\left(\frac{\mathbf{q}_i^{(t)}\mathbf{K}_t^{\top}}{\sqrt{D}}\right)\mathbf{V}_t,$$

where $\mathbf{q}_i^{(t)} = \mathbf{c}_i^{(t)}\mathbf{W}_q$, $\mathbf{K}_t = \mathbf{X}_t\mathbf{W}_k$, and $\mathbf{V}_t = \mathbf{X}_t\mathbf{W}_v$ are the projected query, keys, and values, respectively. Repeating this for all T_c chunks gives the latent sequence $\mathbf{Z} = [\mathbf{z}_1^{(1)}, \ldots, \mathbf{z}_{N_c}^{(1)}, \ldots, \mathbf{z}_1^{(T_c)}, \ldots, \mathbf{z}_{N_c}^{(T_c)}]$.

B.3 Backbone Architecture

Attention-based. The attention-based backbone is a stack of self-attention blocks operating over the entire latent sequence [13]. We denote by $\mathbf{h}_{l,i}^{(t)}$ the hidden state for latent token i in chunk t at

layer l. Omitting residual connections, normalization, and feed-forward layers for brevity, we have:

$$\mathbf{h}_{l,i}^{(t)} = \mathrm{softmax}\left(\frac{\mathbf{q}_{l,i}^{(t)}\mathbf{K}_l^\top}{\sqrt{D}}\right)\mathbf{V}_l,$$

where $\mathbf{q}_{l,i}^{(t)} = \mathbf{h}_{l-1,i}^{(t)}\mathbf{W}_q$, $\mathbf{K}_l = \mathbf{H}_{l-1}\mathbf{W}_k$, and $\mathbf{V}_l = \mathbf{H}_{l-1}\mathbf{W}_v$, with $\mathbf{H}_{l-1}$ the full latent sequence at layer $l-1$ and $\mathbf{h}_{0,i}^{(t)} = \mathbf{z}_i^{(t)}$. The cost of this backbone is quadratic in N_z, which is independent of the number of spike tokens, so it remains tractable even for very long input sequences [59].

Recurrence-based. The recurrent backbone is a stack of SSM blocks [14] whose hidden state follows the transition function $\mathbf{h}_l^{(t)} = f_{\text{SSM}}\big(\mathbf{h}_{l-1}^{(t)}, \mathbf{h}_l^{(t-1)}\big)$, where $\mathbf{h}_0^{(t)} = g\big([\mathbf{z}_1^{(t)}, \ldots, \mathbf{z}_{N_c}^{(t)}]\big)$. Following Ryoo et al. [14], we set $N_c = 1$ and take $g(\cdot)$ to be the identity, so that $\mathbf{h}_0^{(t)} = \mathbf{z}_1^{(t)}$.

B.4 Masked Autoencoder

We apply temporal masking at the level of time chunks, with each chunk masked independently with probability 0.5. All latent tokens in a masked chunk are replaced with a learnable mask token. For the attention-based backbone, we then add a learnable position embedding to each latent token to encode which latent query it originated from. As shown in Section D.3.2, our method is robust to the choice of masking probability. At the output, for each masked chunk, we sample up to 10 neural units uniformly without replacement from the units recorded in the sample. The embedding of each sampled unit, timestamped at the chunk midpoint, serves as a query for predicting the unit's firing rate, and the Poisson NLL is computed against its spike count in that chunk.

B.5 Output Cross-Attention for the SL Pathway

In the SL pathway, each output query $\mathbf{q}_o$ attends to all latents with the `POYO` backbone, or to the $k = 3$ most recent latents with the `POSSM` backbone. The query is a learnable session embedding, and its prediction timestamp is encoded with RoPE. This design supports flexible output predictions, including unaligned, irregular, and even out-of-boundary queries [14].

Multi-Task Supervised Learning. In the multi-task setting, the SL pathway uses a separate linear readout for each task in the dataset. This setting can be used with both backbones, giving `POYO+` [26] and `POSSM+`. We use this setting by default and, for brevity, refer to these models as `POYO` and `POSSM`.

B.6 Finetuning Strategy

We adopt the two finetuning strategies from Azabou et al. [13] for adapting the pretrained model to previously unseen sessions: Unit Identification (UI) and Full Finetuning (FT).

Unit Identification. With UI, we update only the unit and session embeddings during finetuning and keep the rest of the model frozen. This transfers the pretrained model by mapping new units and sessions into the learned embedding space, allowing rapid adaptation to new sessions. When the new sessions involve new tasks, UI additionally unfreezes the task embeddings and linear readouts.

Full Finetuning. FT instead updates all model parameters, which more closely resembles single-session end-to-end training. FT is typically staged: UI is applied first for a number of epochs, after which the remaining parameters are unfrozen. Unless otherwise stated, we run UI for 100 epochs before unfreezing all parameters.

B.7 Model Configurations for the Monkey and Mouse Datasets

Tables 6, 7, 8, and 9 list the hyperparameters of `MOJO-POYO`, `MOJO-POYO-L`, `MOJO-POGRU` and `MOJO-POMAMBA`, and the `MOJO-POYO` model used for the mouse decision-making tasks, respectively. For `MOJO-POGRU` and `MOJO-POMAMBA`, we use the same architecture as Ryoo et al. [14], including the encoder, backbone, and decoder, with 20 ms chunks. The only addition is a decoder and readout for spike-count reconstruction, which modestly increases the parameter count relative to `POSSM`. For `MOJO-POYO`, we use a slightly smaller model than `POYO-1` [13] to more closely match the parameter counts

Table 5: **Comparison of model and training hyperparameters between MOJO and POSSM/POYO.**

Models	#p	opt	lr	lr(unit)	bs	wd	sch
POYO	11.76M	Lamb	0.002	0.002	256	1×10^{-4}	1Cycle
MOJO-POYO	9.89M	AdamW	2.5×10^{-4}	2.5×10^{-4}	64	1×10^{-4}	1Cycle
MOJO-POYO-L	12.09M	AdamW	2.5×10^{-4}	2.5×10^{-4}	64	1×10^{-4}	1Cycle
POGRU	7.65M	Lamb	0.002	0.002	256	1×10^{-4}	1Cycle
MOJO-POGRU	8.53M	Lamb	0.001	0.01	128	1×10^{-4}	1Cycle
POMAMBA	8.65M	Lamb	0.002	0.002	256	1×10^{-4}	1Cycle
MOJO-POMAMBA	9.53M	Lamb	0.001	0.01	128	1×10^{-4}	1Cycle

of the other MOJO backbones. We also use pathway integration to share the encoder computation between the SSL and SL pathways in MOJO-POYO, which results in 50 latent timesteps with 2 latents each, instead of 8 timesteps with 64 latents each in POYO-1. For the mouse decision-making tasks, we use a smaller MOJO-POYO model.

Table 5 reports the model and training hyperparameters for each main comparison: parameter count (#p), optimizer (opt), learning rates for the unit embedding layer and for the remaining parameters (lr), batch size (bs), weight decay (wd), and scheduler (sch). Parameter counts include the encoder, backbone, and decoder, and exclude all embedding layers. We use the OneCycleLR scheduler [60] as implemented in PyTorch [61], with `pct_start` $= 0.2$ and `div_factor` $= 1$. Full training details are given in Section C.

Table 6: **Hyperparameters for MOJO-POYO.** Parameter counts include the encoder, backbone, and decoder.

Hyperparameters	#
Input Dim	128
Backbone Layers	20
Latents per Chunk	2
Cross Heads	1
Self Heads	8
Params	9.89M

Table 7: **Hyperparameters for MOJO-POYO-L.** Parameter counts include the encoder, backbone, and decoder.

Hyperparameters	#
Input Dim	128
Backbone Layers	24
Latents per Chunk	4
Cross Heads	4
Self Heads	8
Params	12.09M

Table 8: **Hyperparameters for MOJO-POSSM (GRU or Mamba).** Parameter counts include the encoder, backbone, and decoder.

Hyperparameters	#
Input Dim	256
Backbone Layers	4
Latents per Chunk	1
Cross Heads	1
RNN Hidden Dim	512
Params	8.53M or 9.53M

Table 9: **Hyperparameters for MOJO-POYO on mouse decision-making tasks.** Parameter counts include the encoder, backbone, and decoder.

Hyperparameters	#
Input Dim	128
Backbone Layers	12
Latents per Chunk	4
Cross Heads	2
Self Heads	8
Params	6.29M

Table 10: **Hyperparameters for MOJO-POYO on the human ECoG dataset.** Parameter counts include the encoder, backbone, and decoder.

Hyperparameters	#
Input Dim	128
Backbone Layers	12
Encoder Self-Attention Layers	2
Latents per Chunk	8
Cross Heads	4
Self Heads	8
Patch Size	4
Params	7.46M
Optimizer	AdamW
Learning Rate	2.5×10^{-4}
Batch Size	128
Weight Decay	1×10^{-4}
Unit Dropout (Max/Min/Mode)	256/64/128
SSL Loss Coef	0.5

Table 11: **Hyperparameters for POYO on the human ECoG dataset.** Parameter counts include the encoder, backbone, and decoder.

Hyperparameters	#
Input Dim	128
Backbone Layers	6
Encoder Self-Attention Layers	2
Latents per Chunk	8
Cross Heads	4
Self Heads	8
Patch Size	4
Params	4.43M
Optimizer	AdamW
Learning Rate	5×10^{-4}
Batch Size	128
Weight Decay	1×10^{-4}
Unit Dropout (Max/Min/Mode)	128/32/64

B.8 Model Configurations for the Human ECoG Dataset

We tuned the hyperparameters of both MOJO-POYO and POYO on this dataset. Table 10 lists the hyperparameters of MOJO-POYO, which uses a larger model here, mainly to learn the SSL objective better. For POYO, larger models sometimes failed to train, so we use a smaller model (see Table 11). As a classical baseline, we apply independent component analysis (ICA) with 150 components to the neural data and fit a linear support vector classifier to the ICA-transformed data, using `FastICA` and `LinearSVC` from `scikit-learn` [62]. For EEGNet [63], we reimplement the training loop using the code and default hyperparameters of the TorchEEG package [64].

C Training Details

C.1 Monkey Reaching Tasks

In each session of the monkey reaching datasets, the decoder predicts two-dimensional hand or cursor velocity from spiking activity. For all experiments on this task, we used the mean squared error as the loss function. During training, behavioural targets were sampled at up to 100 Hz; for sessions or datasets with a higher behavioural sampling rate, we randomly subsampled the targets to enforce this limit.

We trained the POYO and POSSM baselines using the Lamb optimizer [65] for 500 epochs, with a learning rate of 0.004 and a batch size of 128 for single-session training, and a learning rate of 0.002 and a batch size of 256 for multi-session training. The single-session hyperparameters were also used for UI and FT.

For single-session MOJO, we used Lamb with a learning rate of 0.01 for the unit embedding layer and 0.001 for the rest of the model, and batch sizes of 128 for MOJO-POSSM and 64 for MOJO-POYO. For multi-session training, MOJO-POSSM used the same learning rates as the single-session models, while MOJO-POYO used AdamW [66] with a learning rate of 2.5×10^{-4} for all parameters; batch sizes were again 128 and 64, respectively. Empirically, training MOJO-POSSM with AdamW led to unstable optimization, and Lamb required a larger learning rate for the unit embeddings to learn the SSL objective well. For UI and FT, we used Lamb with learning rates of 0.01 and 0.001 for MOJO-POSSM, and AdamW with a learning rate of 5×10^{-4} for MOJO-POYO. All models were trained for 500 epochs. Multi-session training used four NVIDIA H100 GPUs, and single-session training and finetuning used a single NVIDIA L40S GPU.

The NDT-2 [27], NDT-3 [28], and NEDS [16] baselines were trained using the authors' released code, modified so that they were trained and evaluated on exactly the same data splits as MOJO. We followed the authors' recommended hyperparameter configurations and tuning strategies for both training and finetuning on our datasets.

C.2 Mouse Vision Tasks

Each session of the mouse vision tasks poses a multi-task problem with two or three multi-class classification tasks, depending on the stimulus set (natural scenes are shown only in stimulus set 1). We used the cross-entropy loss for all tasks.

We trained a separate MLP baseline for each task. For drifting-grating orientation and temporal frequency (2 s per trial), we used AdamW with a learning rate of 0.004, a batch size of 128, 20 ms chunks, and a sequence length of 0.5 s. For natural scenes (250 ms per trial), we reduced the learning rate to 0.0012 and used a sequence length of 250 ms; this lowered learning rate was necessary for stable training.

For multi-session training of POYO, we followed Azabou et al. [26] and used task-specific weights, assigning natural scenes a weight of 0.3 and all other tasks a weight of 1. POGRU struggled to learn with this weighting, so we instead weighted all tasks equally for it. MOJO-POYO and MOJO-POGRU follow the same choices as POYO and POGRU, respectively. All other hyperparameters were the same as for the monkey tasks.

For single-session training, UI, and FT, all tasks were weighted equally for all models. Hyperparameters were the same as for the monkey tasks for most models, with two exceptions: single-session MOJO used larger learning rates (0.02 for the unit embedding layer and 0.002 for the rest of the model), and FT of MOJO-POYO used a smaller learning rate (2.5×10^{-4}) to reduce overfitting.

The NEDS baseline was trained using the authors' publicly released code, with several modifications to support the Allen Visual Coding Neuropixels dataset and classification tasks that are not presented simultaneously, and to ensure consistent data splits. We trained NEDS for 500 epochs on the Allen data; all other training and finetuning hyperparameters followed the authors' recommendations.

C.3 Mouse Decision-Making Tasks

We used the cross-entropy loss for choice and block prior, and the mean squared error for wheel speed and whisker motion energy. We weighted the choice and block-prior losses by 0.2, since we found these tasks to be prone to overfitting. For MOJO, we used AdamW with a learning rate of 0.001 for pretraining and 1.25×10^{-4} for finetuning, with a batch size of 64 in both cases. Pretraining used two NVIDIA L40S GPUs, and finetuning used a single NVIDIA L40S GPU. We finetuned all parameters from the start, without a UI stage, as this was more stable across the four tasks. POYO was pretrained and finetuned using Lamb with a learning rate of 0.002 and a batch size of 64. NEDS was trained using the authors' publicly released code as-is for this dataset.

C.4 Human Speech Tasks

Classifying the vowel, consonant, and consonant–vowel syllable from ECoG signals are multi-class classification tasks with 3, 19, and 57 classes, respectively, and we used the cross-entropy loss for all three. Since ECoG is a new modality for POYO-style models [13, 14, 26], we performed hyperparameter tuning and found AdamW to work well empirically for both MOJO-POYO and POYO.

For POYO, the readout layer's output dimension was set to the total number of syllable types in the dataset for all three tasks. We tuned the learning rate over $\{2.5 \times 10^{-4}, 5 \times 10^{-4}\}$, the weight decay over $\{10^{-3}, 10^{-4}\}$, and the unit dropout (maximum/minimum/mode number of units) over {256/64/128, 128/32/64, 64/16/32, 32/8/16}. We also compared depths of 6 and 12, and a shared output dimension (the total number of syllable types) with task-specific output dimensions (the number of classes per task). The final POYO model was trained on a single NVIDIA L40S GPU using AdamW with a learning rate of 5×10^{-4}, a weight decay of 1×10^{-4}, a unit dropout of 128/32/64, a depth of 6, and a shared output dimension.

For MOJO-POYO, we adopted a curriculum where we first trained with the SSL objective only for 1000 epochs, then continued with joint SSL and SL training. Warm-starting from this SSL-only

checkpoint improved the subsequent joint training. We also tuned the SSL loss coefficient in the joint objective. The final MOJO-POYO model was trained on a single NVIDIA L40S GPU using AdamW with a learning rate of 2.5×10^{-4}, a weight decay of 1×10^{-4}, a unit dropout of 256/64/128, a depth of 12, and an SSL loss coefficient of 0.5. Section F reports the performance of MOJO-POYO at a model size matched to POYO.

The ICA and linear SVC baseline was fit separately for each session on CPU. We ran three random initializations and found the variance across them to be negligible. EEGNet [63] was trained separately for each session and task using AdamW with a learning rate of 0.001 and a batch size of 32. The Du-IN [37] baseline was trained using the authors' publicly released code, modified so that it was trained and evaluated on exactly the same data splits as MOJO-POYO.

D Additional Experiments on Monkey Reaching Datasets

D.1 Full Results on Reaching Tasks

Table 12 reports the complete results on the monkey reaching tasks, including single-session results and two additional POSSM backbones: Mamba2 [7] and xLSTM [67]. Two backbones – the Transformer (MOJO-POYO) and the GRU (MOJO-POGRU) – consistently decode better across evaluation sessions. In our experience, the Transformer backbone usually achieves better SSL performance during pretraining, likely because the self-attention is non-causal, and this advantage carries over to decoding even under our causal evaluation strategy [14]. The Transformer backbone also gives better FT and single-session performance on the C – CO 2010 sessions, possibly because those sessions contain more neural units. However, its inference speed is still slower than that of POSSM, consistent with Ryoo et al. [14], so we focus on POSSM backbones for real-time decoding. Among the recurrent backbones, the GRU performs best, especially for UI and single-session training; for example, single-session MOJO-POGRU performs on par with many pretrained models. The xLSTM is also worth noting, as it performs well across both pretraining and finetuning.

D.2 Additional Results on Brain Region Classification

Table 13 reports the full results of brain region classification from the unit embeddings of the pretrained monkey models. We used `LogisticRegressionCV` from `scikit-learn` [62] as the classifier, with the regularization strength selected by 5-fold cross-validation. Each classification task was repeated over 5 random data splits, and we report the mean and standard deviation across splits.

D.3 Ablation Studies

D.3.1 Pathway Integration

We compared the performance of MOJO-POYO with and without our pathway integration strategy. Without pathway integration, the SL pathway has its own input cross-attention computation, identical to POYO, with 8 sets of latents spaced 125 ms apart. In both cases, we used the same number of latents per time chunk ($N_c = 2$), so that all latent queries were shared between the SSL and SL pathways. As shown in Table 14, pathway integration improved FT performance but reduced UI performance slightly on two of the three evaluation sets. We attribute the FT gains to the shared input cross-attention computation. Even with separate computations, the encoder parameters are shared, so the encoder must handle latents at two time resolutions: 20 ms for the SSL pathway and 125 ms for the SL pathway. Pathway integration removes this mismatch. Conversely, separate computations may make the unit embeddings easier to learn, which UI relies on, since each pathway attends to them at its own time resolution. Without pathway integration, inference is also slightly slower, because the input cross-attention is computed twice.

Table 12: **Behavioural decoding R^2 on monkey reaching tasks with 20 ms time chunks.** Values are mean $R^2 \pm$ SD over sessions. Best performing models are in boldface (1st) and underlined (2nd). †Results reproduced from Ryoo et al. [14].

		Same Animal, Other days	*New animal*	
	Method	C – CO 2010 (5)	T – CO (6)	T – RT (6)
From scratch	MLP†	0.5842 ± 0.2052	0.7940 ± 0.0341	0.6082 ± 0.3014
	Mamba†	0.6840 ± 0.0936	0.7318 ± 0.0426	0.6653 ± 0.0978
	GRU†	0.7742 ± 0.0964	0.8389 ± 0.0248	0.7414 ± 0.0426
	POYO-SS	0.7465 ± 0.1119	0.8509 ± 0.0423	0.6929 ± 0.0978
	POMAMBA-SS†	0.7691 ± 0.0786	0.8613 ± 0.0121	0.7300 ± 0.0719
	POGRU-SS†	<u>0.7780</u> ± 0.0980	0.8724 ± 0.0190	0.7429 ± 0.0708
	MOJO-POYO-SS	**0.8135** ± 0.0922	<u>0.8803</u> ± 0.0191	<u>0.7443</u> ± 0.0789
	MOJO-POMAMBA-SS	0.7752 ± 0.0878	0.8753 ± 0.0171	0.7428 ± 0.0861
	MOJO-POGRU-SS	0.7768 ± 0.1283	**0.8834** ± 0.0167	**0.7711** ± 0.0596
Pretrained (UI)	POYO (UI)	0.7759 ± 0.1003	0.8123 ± 0.0419	0.7011 ± 0.0975
	POMAMBA (UI)	0.7283 ± 0.1138	0.8574 ± 0.0225	0.7283 ± 0.0846
	POGRU (UI)	0.7632 ± 0.1013	0.8587 ± 0.0216	0.7331 ± 0.0775
	MOJO-POYO (UI)	0.7846 ± 0.0897	0.8387 ± 0.0251	0.7470 ± 0.0617
	MOJO-POMAMBA (UI)	<u>0.7937</u> ± 0.0735	<u>0.8753</u> ± 0.0160	<u>0.7591</u> ± 0.0588
	MOJO-POGRU (UI)	**0.7949** ± 0.0892	**0.8772** ± 0.0189	0.7570 ± 0.0693
	MOJO-POMAMBA2 (UI)	0.7720 ± 0.0872	0.8637 ± 0.0158	0.7565 ± 0.0582
	MOJO-POXLSTM (UI)	0.7832 ± 0.1045	0.8671 ± 0.0255	**0.7615** ± 0.0539
Pretrained (FT)	NDT-2† (FT)	0.7846 ± 0.1167	0.7173 ± 0.0443	0.6323 ± 0.1339
	NDT-3 (FT)	0.7524 ± 0.1322	0.8576 ± 0.0313	0.7066 ± 0.0980
	NEDS (FT)	0.5968 ± 0.0760	0.7635 ± 0.0758	0.6121 ± 0.0918
	POYO (FT)	<u>0.8244</u> ± 0.0753	0.8817 ± 0.0352	0.7624 ± 0.0815
	POMAMBA (FT)	0.8142 ± 0.0763	0.8949 ± 0.0152	0.7580 ± 0.0745
	POGRU (FT)	0.8126 ± 0.0892	0.8936 ± 0.0212	0.7575 ± 0.0875
	MOJO-POYO (FT)	**0.8438** ± 0.0888	**0.9131** ± 0.0177	**0.7964** ± 0.0663
	MOJO-POMAMBA (FT)	0.8153 ± 0.0925	0.9043 ± 0.0182	0.7675 ± 0.0806
	MOJO-POGRU (FT)	0.8222 ± 0.0950	<u>0.9103</u> ± 0.0168	<u>0.7776</u> ± 0.0710
	MOJO-POMAMBA2 (FT)	0.8189 ± 0.0918	0.9045 ± 0.0162	0.7623 ± 0.0731
	MOJO-POXLSTM (FT)	0.8207 ± 0.0928	0.9082 ± 0.0185	0.7712 ± 0.0670

Table 14: **Ablation results on pathway integration for MOJO-POYO.**

Method	C – CO 2010 (5)	T – CO (6)	T – RT (6)
MOJO-POYO (Shared Enc) (UI)	0.7846 ± 0.0897	0.8387 ± 0.0251	0.7470 ± 0.0617
MOJO-POYO (Separate Enc) (UI)	0.7902 ± 0.0916	0.8523 ± 0.0200	0.7432 ± 0.0876
MOJO-POYO (Shared Enc) (FT)	0.8438 ± 0.0888	0.9131 ± 0.0177	0.7964 ± 0.0663
MOJO-POYO (Separate Enc) (FT)	0.8324 ± 0.1030	0.9030 ± 0.0164	0.7810 ± 0.0926

Table 13: **Brain region classification accuracies on pretrained monkey reaching datasets.** M1: primary motor cortex, PMd: dorsal premotor cortex, S1: primary somatosensory cortex. Values are mean accuracies ± SD over 5 seeds. Best performing models are in boldface (1st) and underlined (2nd).

	Perich et al. [29]	Churchland et al. [30]	O'Doherty et al. [56]	*Joint Monkey*		
Method	M1 vs. PMd	M1 vs. PMd	M1 vs. S1	Multi-region	Multi-subject	M1 in Multi-subject
Chance	50.00	50.00	50.00	33.33	16.67	16.67
POYO	64.23 ± 0.75	74.36 ± 1.81	81.17 ± 0.99	69.65 ± 0.45	49.67 ± 0.72	61.50 ± 0.51
POMAMBA	64.06 ± 0.64	74.78 ± 2.09	82.15 ± 0.83	69.54 ± 0.37	46.36 ± 0.24	60.74 ± 0.43
POGRU	65.77 ± 0.90	71.70 ± 1.35	81.64 ± 0.70	69.51 ± 0.47	46.57 ± 0.55	60.30 ± 0.92
MOJO-POYO	**81.08** ± 0.49	<u>98.17</u> ± 0.47	**89.34** ± 0.46	**78.24** ± 0.42	<u>72.69</u> ± 0.87	<u>84.51</u> ± 0.63
MOJO-POMAMBA	77.32 ± 0.51	95.30 ± 1.08	87.72 ± 0.60	73.91 ± 0.52	68.18 ± 0.55	78.91 ± 0.33
MOJO-POGRU	<u>80.43</u> ± 0.40	**98.54** ± 0.54	<u>88.40</u> ± 0.17	<u>76.88</u> ± 0.40	**75.74** ± 0.43	**85.28** ± 0.36

Table 15: **Ablation results on latent masking strategy and probability.** Results are shown for temporal and forward latent masking strategies: test R^2, SL test loss, and SSL test loss for pretraining, and test R^2 for finetuning.

	Masking Probability	Pretraining			Finetuning
		R^2	MSE	Poisson NLL	R^2
Temporal	0.02	0.9166	0.01333	0.3848	0.8453
	0.05	0.9167	0.01303	0.3894	0.8447
	0.1	0.9178	0.01307	0.3861	0.8449
	0.3	0.9181	0.01295	0.3868	0.8442
	0.5	0.9178	0.01299	0.3871	0.8449
	0.7	0.9173	0.01324	0.3894	0.8430
	0.9	0.9167	0.01325	0.3964	0.8420
Forward	0.3	0.9161	0.01326	0.3874	0.8423
	0.5	0.9172	0.01303	0.3874	0.8467
	0.7	0.9158	0.01327	0.3915	0.8449

D.3.2 Masking Strategy and Probability

We pretrained MOJO-POGRU on the Perich et al. [29] dataset, excluding the evaluation sessions, with two masking strategies and a range of masking parameters, and report finetuning performance on the held-out monkey T sessions in Table 15. For temporal masking, where each time chunk is masked independently at random, we varied the masking probability from 0.02 to 0.9. For forward masking, where the last chunks of each 1 s segment are masked, we varied the masked fraction from 0.3 to 0.7. Although the masking parameter affects the SSL objective, the downstream decoding performance remains robust. This is consistent with SSL benefiting decoding mainly through the unit embeddings (Appendix E.3): predicting each unit's spike counts teaches its embedding the unit's firing properties across a wide range of masking parameters.

D.3.3 Pretraining with SSL-Only Objective

To isolate the effect of the joint SSL-SL objective, we pretrained MOJO-POGRU with the SSL objective only on the Perich et al. [29] dataset for 500 epochs, and then finetuned it with SL on held-out sessions of the unseen monkey T. As shown in Table 16, the SSL-only model performs worse than the SL-only and joint SSL-SL models, but still better than single-session training, indicating that SSL-only pretraining remains useful when no labels are available. We then took the SSL-only pretrained model and continued pretraining with SL only for another 500 epochs. As shown in

Table 16: **Ablation results on pretraining strategy for finetuning.** Results shown are mean finetuning R^2 on sessions from an unseen animal.

	with Pretraining			Single-Session
	Joint SSL-SL	SL-Only	SSL-Only	
Finetuning R^2	0.8449	0.8319	0.8287	0.8077

Table 17: **Ablation results on pretraining strategy for SL pretraining.** Results shown are the mean supervised and self-supervised metrics over all pretraining sessions.

Strategy	test R^2	MSE	Poisson NLL (epoch 50)
Joint SSL-SL	0.9178	0.01299	0.390
SSL then SL	0.9122	0.01395	0.394
SL-Only	0.9107	0.01449	N/A

Table 17, SSL-only pretraining converges more slowly on the SSL objective than joint pretraining (higher Poisson NLL at epoch 50), and the subsequent SL stage reaches worse SL performance. Together, these results show that label-free pretraining already yields transferable representations, but using labels jointly with SSL remains advantageous, as it also pretrains the task readouts and keeps the representations task-relevant.

D.3.4 Alternative SSL Methods

Table 18 compares `MOJO` with alternative SSL methods, each pretrained on the Perich et al. [29] dataset and finetuned on held-out monkey T sessions. Whereas `MOJO` uses masked autoencoding with temporal masking of the latents, we consider two families of alternatives, each trained jointly with SL. The first masks spikes from the input token sequence along the spatial dimension, removing (1) randomly selected neurons in each time bin, (2) the same randomly selected neurons across all time bins, or (3) all neurons from a randomly selected brain region (M1 or PMd). The second uses contrastive predictive coding (CPC) [68], with negative samples drawn either from the same sequence or from other sessions. We did not finetune the region-masking model, as it underperformed substantially during pretraining, or CPC with negatives from other sessions, as single-session finetuning provides no other sessions to draw negatives from. `MOJO` outperforms all these alternative methods on this dataset. CPC with negatives from the same sequence also overfit severely on the contrastive loss, suggesting that the CPC and SL objectives were poorly integrated in this case.

D.4 Transferring to Human Handwriting

We evaluated transfer from a pretrained monkey model to a human handwriting task. Ryoo et al. [14] showed that such transfer is viable, and we verified that it also holds for `MOJO`. The Willett et al. [69] dataset contains 9 sessions in which a human participant attempted handwriting movements for single characters. Neural activity was recorded with two 96-channel microelectrode arrays implanted in motor cortex and is provided as threshold-crossing counts binned at 10 ms. We compared pretrained and from-scratch `MOJO-POGRU` models on decoding the intended characters. As shown in Table 19, pretrained `MOJO` outperforms the same architecture trained from scratch, showing that `MOJO` transfers from monkey reaching to human handwriting.

Table 18: **Ablation results on alternative SSL methods.** Results are shown for three input-masking strategies and for CPC with two choices of negative samples: test R^2, SL test loss, and p-value vs. MOJO-POGRU for pretraining, and test R^2 for finetuning.

Model	Pretraining			Finetuning
	R^2	MSE	p vs. MOJO-POGRU	R^2
POGRU	0.9107	0.01449	<0.001	0.8319
MOJO-POGRU	0.9178	0.01299	N/A	0.8449
Input Random Mask	0.9080	0.01491	0.003	0.8318
Input Neuron Mask	0.9068	0.01502	<0.001	0.8322
Input Region Mask	0.3419	0.12182	<0.001	N/A
CPC Same Sequence	0.9114	0.01441	<0.001	0.8274
CPC Mixed Session	0.9060	0.01534	<0.001	N/A

Table 19: **Human handwriting classification accuracies for MOJO-POGRU.** Values are mean accuracy $\pm$ SD over 3 seeds.

Method	Acc. (%) $\uparrow$
From scratch	95.34 ± 0.36
Pretrained	97.73 ± 0.41

D.5 Additional Results on Few-shot Finetuning

Tables 20, 21, and 22 report additional few-shot finetuning results on three held-out sessions (T-20130819, T-20130820, and C-20161021) from the Perich et al. [29] dataset for three backbones (MOJO-POYO, MOJO-POMAMBA, and MOJO-POGRU). We also average over all six held-out RT sessions of the unseen monkey T (Figure 2a): within each session, we normalize the few-shot R^2 of every model by that of MOJO-POGRU finetuned on all 32 labelled trials plus any remaining unlabelled data from the session, and then average the normalized R^2 across sessions. MOJO-POGRU with unlabelled data reaches 68.6%, 79.4%, 86.6%, and 93.9% of this reference using 2, 4, 8, and 16 labelled trials (SEM of 4.0, 2.7, 1.6, and 1.0 percentage points across sessions). Interpolating between shot counts on a log scale, it needs approximately 4, 11, and 18 labelled trials to reach 80%, 90%, and 95% of the reference, compared with approximately 8, 15, and 26 for POGRU. The benefit of SSL and additional unlabelled data is therefore largest when labels are scarce.

D.6 Additional Results on Behavioural Phase Transfer During Finetuning

Following Azabou et al. [13], we divide each CO session into 5 behavioural phases. All 5 phases are included during training, with the loss weights given below, but only the REACH phase is used for evaluation (for CO sessions).

- HOLD (weight 0.1): in each valid trial, from `target_on_time` to `go_cue_time`.
- REACH (weight 5.0): in each valid trial, from `go_cue_time` to `stop_time`.
- RETURN (weight 1.0): in each valid trial, from `stop_time` to the end of the trial.
- INVALID (weight 0.1): the full duration of invalid trials.
- RANDOM (weight 1.0): all remaining periods, excluding behavioural outliers (weight 0).

To test whether SSL on unlabelled data helps decode behaviour whose labels are never seen, we conducted a series of phase-transfer experiments in which MOJO has access to the labels of only a

Table 20: **Few-shot finetuning for held-out session T-20130819.** Results are shown for three backbones with $k = 2$ to 32 labelled trials; +U additionally uses the remaining $32 - k$ trials without labels. R^2 values are averaged over 5 seeds.

Backbone	Strategy	2	4	8	16	32
POYO	SL	0.154	0.231	0.431	0.562	0.697
	MOJO	0.259	0.336	0.604	0.722	0.833
	MOJO +U	0.271	0.371	0.663	0.761	0.826
POMAMBA	SL	0.214	0.323	0.548	0.674	0.788
	MOJO	0.245	0.314	0.584	0.700	0.816
	MOJO +U	0.281	0.403	0.637	0.730	0.818
POGRU	SL	0.195	0.302	0.567	0.705	0.796
	MOJO	0.211	0.318	0.575	0.689	0.804
	MOJO +U	0.224	0.345	0.638	0.718	0.806

Table 21: **Few-shot finetuning for held-out session T-20130820.** Results are shown for three backbones with $k = 2$ to 32 labelled trials; +U additionally uses the remaining $32 - k$ trials without labels. R^2 values are averaged over 5 seeds.

Backbone	Strategy	2	4	8	16	32
POYO	SL	0.264	0.393	0.503	0.575	0.614
	MOJO	0.394	0.490	0.574	0.636	0.692
	MOJO +U	0.516	0.590	0.639	0.671	0.704
POMAMBA	SL	0.312	0.440	0.521	0.618	0.641
	MOJO	0.387	0.478	0.548	0.614	0.662
	MOJO +U	0.458	0.522	0.569	0.616	0.667
POGRU	SL	0.285	0.472	0.542	0.623	0.669
	MOJO	0.369	0.492	0.552	0.637	0.668
	MOJO +U	0.473	0.540	0.585	0.638	0.684

Table 22: **Few-shot finetuning for held-out session C-20161021.** Results are shown for three backbones with $k = 2$ to 32 labelled trials; +U additionally uses the remaining $32 - k$ trials without labels. R^2 values are averaged over 5 seeds.

Backbone	Strategy	2	4	8	16	32
POYO	SL	0.430	0.583	0.738	0.828	0.900
	MOJO	0.462	0.666	0.752	0.877	0.932
	MOJO +U	0.454	0.682	0.828	0.885	0.931
POMAMBA	SL	0.509	0.681	0.790	0.846	0.929
	MOJO	0.522	0.661	0.777	0.882	0.927
	MOJO +U	0.532	0.699	0.820	0.887	0.927
POGRU	SL	0.467	0.604	0.782	0.868	0.929
	MOJO	0.503	0.668	0.788	0.889	0.930
	MOJO +U	0.517	0.737	0.839	0.896	0.930

Table 23: **Finetuning to a new session from pretrained MOJO-POGRU, with only RANDOM phase labels.** MOJO had access to all unlabelled data and only RANDOM labels. Values are mean $R^2 \pm$ SD over 5 seeds.

C – CO		T – CO					
C-20161013	C-20161021	T-20130819	T-20130821	T-20130823	T-20130903	T-20130905	T-20130909
0.53 ± 0.06	0.52 ± 0.09	0.29 ± 0.12	-0.01 ± 0.03	-0.07 ± 0.22	0.09 ± 0.12	0.16 ± 0.13	0.17 ± 0.17

subset of phases when finetuning on a new session, and is evaluated on the REACH phase. In the first experiment, only the RANDOM phase is labelled, while all spikes from the session's training set are available for SSL (Table 23). Since RANDOM typically makes up less than 1% of the data, we did not run this experiment for purely supervised models, which would have too few labels to train on.

In the second experiment, labels from both the RANDOM and INVALID phases are available (Table 24). In the third, both the spikes and the labels come only from the INVALID phase (Table 25).

Table 24: **Comparison of phase transfer during finetuning with labels from two phases.** All spikes and only the RANDOM and INVALID phase labels are available to the models. Values are mean $R^2 \pm$ SD over sessions.

Method	C – CO (2)	T – CO (5)
POGRU	0.798 ± 0.03	0.653 ± 0.08
MOJO-POGRU	0.891 ± 0.01	0.811 ± 0.03

Table 25: **Comparison of phase transfer during finetuning with only data from INVALID phase.** Spikes and labels of only INVALID phase are available to the models. Values are mean $R^2 \pm$ SD over sessions.

Method	C – CO (2)	T – CO (6)
POGRU	0.752 ± 0.02	0.685 ± 0.07
MOJO-POGRU	0.752 ± 0.03	0.738 ± 0.06

The results show that MOJO can leverage additional unlabelled data to improve decoding of a phase whose labels are never seen during finetuning, including when transferring to a new animal.

Finally, the RANDOM phase labels, despite their small size, are essential: without them, MOJO fails to decode above chance even on new sessions of a seen animal. During finetuning, the model has no access to metadata such as animal identity, recording date, or lab. Having been pretrained on heterogeneous data, it has learned many possible neural-to-behaviour mappings, and the information that links neural activity to behaviour in a given session is largely stored in the session and unit embeddings (as in POYO). These embeddings are relearned for each new session, and while SSL can shape them without labels, a small amount of labelled data is needed to anchor them to the correct mapping.

D.7 Additional Results on Pretraining with Unlabelled Data

We also finetuned the models pretrained with different fractions of unlabelled data on new sessions (see Figure 5a). Additional unlabelled pretraining data does not improve full-data finetuning performance, and lowers UI performance in particular. Unlike in the few-shot experiments of Section 3.1, no additional unlabelled data from the new sessions is used during finetuning here, since all finetuning data are labelled. A plausible explanation is that the additional unlabelled data biases pretraining towards the SSL objective, yielding better unit embeddings for the pretraining sessions (see Figure 5b), but also making the SL pathway rely more on these session-specific embeddings, which must be relearned for each new session. This reliance hurts generalization, especially for UI, which updates only the embeddings. Consistent with this, UI performance increases with the fraction of labelled pretraining data, underscoring the importance of labels during pretraining.

Holding the labelled subset fixed at 10%, we also varied the fraction of the remaining data used without labels. As shown in Table 26, decoding performance on held-out data from the pretraining sessions increases with the amount of unlabelled data.

Table 26: **Effect of the amount of unlabelled data during pretraining.** With labelled data fixed at 10%, results are shown as mean pretraining test R^2 for varying amounts of unlabelled data.

% unlabelled data	0	30	50	70	100
test R^2	0.7452	0.7589	0.7671	0.7771	0.8024

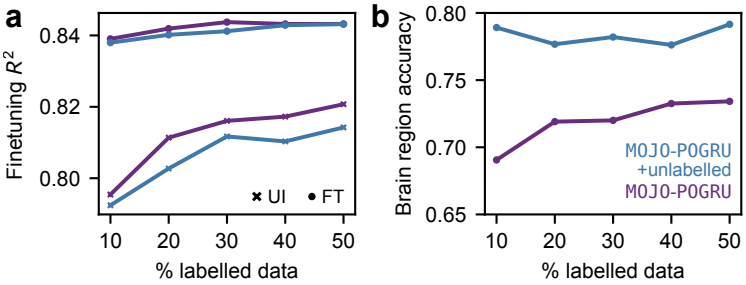


Figure 5: **Additional results on pretraining with unlabelled data.** (a) Finetuning performance with additional unlabelled data during pretraining. (b) Brain region classification performance with additional unlabelled data during pretraining.

D.8 Training Time Comparison

Table 27 compares training times for MOJO models, POGRU, and POYO on the monkey reaching datasets, measured as the time for one full training epoch on four NVIDIA H100 GPUs. With the POGRU backbone (first two rows), MOJO adds little compute overhead, owing to pathway integration (Section 2.6). The POYO-backbone numbers are not directly comparable, since MOJO-POYO uses a smaller backbone and both MOJO-POYO and MOJO-POYO-L use fewer latent tokens in total than POYO (100 and 200 vs. 512 per 1 s sequence); we include them for completeness.

Table 27: **Per-epoch training time comparison on monkey reaching datasets.**

Models	Epoch time (s)
POGRU	220
MOJO-POGRU	246
POYO	522
MOJO-POYO	320
MOJO-POYO-L	476

E Additional Experiments on Mouse Vision Datasets

E.1 Full Results on Mouse Vision Tasks

Table 28 reports the complete results on the mouse vision tasks, including single-session results and all tasks in both stimulus sets: natural scenes (NS, 119 classes), drifting-grating orientation (DG, 8 classes), and drifting-grating temporal frequency (TF, 5 classes). In stimulus set 2, the drifting gratings have only 4 orientations and a single temporal frequency, so we omit TF for set 2. Unlike on the monkey datasets, single-session MOJO models underperform the purely supervised models, except for MOJO-POYO on DG. We hypothesize that this difference is largely due to the number of units per session: evaluation sessions average 676 units in the mouse data, compared with 57 for monkey T and 191 for monkey C (CO). With many more units spanning more diverse brain regions, the SSL objective is harder to learn from a single session, as each unit embedding must be learned

from scratch. This is consistent with masked modelling being data-hungry. In contrast, pretrained MOJO performs best on every task, reaching nearly 100% accuracy on both DG and TF.

Table 28: **Visual stimuli classification accuracies on mouse vision tasks with 20 ms time chunks.** NS: Natural Scenes; DG: Drifting-Grating Orientation; TF: Drifting-Grating Temporal Frequency. (A): with additional unlabelled data from the Allen dataset. (J): with a joint monkey-mouse dataset. Values are mean accuracies ± SD over sessions. Best performing models are in boldface (1st) and underlined (2nd), except for DG from stimulus set 2.

		Stimulus Set 1			*Stimulus Set 2*
	Method	NS (4)	DG (4)	TF (4)	DG (4)
FROM SCRATCH	Chance	0.84	12.50	20.00	25.00
	MLP	83.17 ± 4.92	90.18 ± 2.03	87.11 ± 2.61	99.09 ± 0.63
	POYO-SS	**91.11** ± 3.36	91.72 ± 3.29	**97.60** ± 2.36	99.95 ± 0.10
	POGRU-SS	<u>90.04</u> ± 3.92	<u>98.33</u> ± 2.10	<u>97.29</u> ± 5.00	99.79 ± 0.24
	MOJO-POYO-SS	87.48 ± 5.26	**98.39** ± 1.99	96.51 ± 5.61	99.69 ± 0.27
	MOJO-POGRU-SS	89.08 ± 3.42	95.73 ± 3.40	94.27 ± 2.99	99.74 ± 0.10
PRETRAINED (UI)	POYO (UI)	88.17 ± 3.17	93.54 ± 2.06	98.70 ± 5.98	99.90 ± 0.12
	POGRU (UI)	88.87 ± 2.55	99.17 ± 0.76	98.54 ± 0.85	99.95 ± 0.10
	MOJO-POYO (UI)	91.53 ± 1.99	<u>99.74</u> ± 0.31	<u>99.69</u> ± 0.36	99.95 ± 0.10
	MOJO-POGRU (UI)	92.54 ± 3.41	99.53 ± 0.20	99.01 ± 0.55	99.95 ± 0.10
	MOJO-POYO (A) (UI)	92.06 ± 2.62	**99.84** ± 0.31	99.64 ± 0.36	99.84 ± 0.20
	MOJO-POYO (J) (UI)	<u>93.15</u> ± 2.39	99.43 ± 1.15	99.43 ± 1.01	99.95 ± 0.10
	MOJO-POYO-L(J) (UI)	**94.18** ± 2.00	**99.84** ± 0.20	**99.79** ± 0.29	99.95 ± 0.10
PRETRAINED (FT)	POYO (FT)	91.03 ± 4.07	93.39 ± 4.50	98.54 ± 0.66	99.69 ± 0.21
	POGRU (FT)	91.09 ± 5.64	99.48 ± 0.62	98.85 ± 1.23	99.95 ± 0.10
	MOJO-POYO (FT)	94.12 ± 2.79	**100.00** ± 0.00	<u>99.69</u> ± 0.40	99.95 ± 0.10
	MOJO-POGRU (FT)	94.47 ± 2.83	99.22 ± 0.91	99.17 ± 1.01	99.74 ± 0.40
	MOJO-POYO (A) (FT)	94.05 ± 2.51	**100.00** ± 0.00	99.27 ± 0.52	99.95 ± 0.10
	MOJO-POYO (J) (FT)	<u>94.62</u> ± 2.74	99.69 ± 0.64	99.38 ± 0.66	99.79 ± 0.29
	MOJO-POYO-L(J) (FT)	**95.48** ± 2.05	<u>99.74</u> ± 0.52	**99.74** ± 0.31	99.95 ± 0.10

E.2 Additional Results on Brain Region Classification

We used logistic regression as the classifier and evaluated it with 5-fold cross-validation over units, repeated across 5 seeds. Unit counts vary substantially across brain regions in this dataset, so to balance the classes, we subsampled each region to the same number of units before fitting each classifier.

Figure 6 shows confusion matrices for 7-region classification from MOJO-POYO models pretrained on (a) labelled data only, (b) additional unlabelled data from the same dataset (MOJO-POYO (A)), and (c) additional unlabelled and monkey data (MOJO-POYO (J)). The diagonal becomes more pronounced as more data is added to pretraining, consistent with the increasing 7-region test accuracy in Table 29. We also compare with two non-MOJO baselines: purely supervised POYO, and a feature-based baseline [33] that computes 18 hand-crafted features per neuron (moments of the ISI distribution, firing rate, variability indices, gamma-distribution fits, and band-limited spike-train PSDs). These features are computed from the full recordings, including the held-out data not seen by MOJO during pretraining, which makes this comparison conservative. All MOJO-POYO variants outperform these baselines.

To test whether the probes exploit session-specific structure, we also evaluated session-based splits: we randomly divided the sessions into 5 folds and reported the average test accuracy over 5-fold cross-validation (Table 30). The train-to-test ratio remains 4:1. Probing accuracies with session-based splits are comparable to those in Table 29, and the ranking of methods is unchanged.

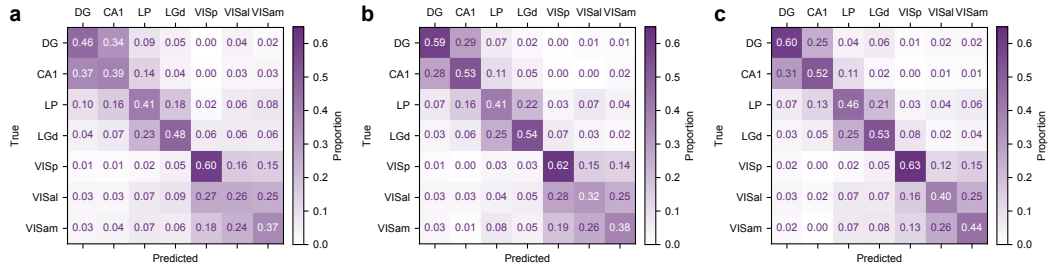


Figure 6: **Confusion matrices for mouse brain region classification from different models.** (a) MOJO-POYO trained on paired Allen data; (b) with additional unlabelled Allen data (MOJO-POYO (A)); (c) also with monkey data (MOJO-POYO (J)).

Table 29: **Brain region classification accuracies on pretrained mouse vision datasets.** Values are mean accuracies ± SD over 5 seeds. (A): with additional unlabelled data from Allen datasets. (J): with joint monkey-mouse dataset. Best performing models are in boldface (1st) and underlined (2nd).

Method	3 merged-regions	7 regions
Chance	33.33	14.29
ISI Baseline [33]	68.46	38.46
POGRU	56.66 ± 0.78	26.93 ± 0.85
POYO	58.45 ± 0.79	27.99 ± 0.39
MOJO-POGRU	70.09 ± 0.58	36.77 ± 0.86
MOJO-POYO	74.48 ± 0.64	43.17 ± 1.35
MOJO-POYO (A)	79.29 ± 0.65	48.67 ± 0.72
MOJO-POYO (J)	<u>80.41</u> ± 0.83	<u>51.16</u> ± 1.03
MOJO-POYO-L(J)	**82.72** ± 0.40	**54.78** ± 1.50

E.3 Additional Results on Spike-Statistic Probing

We linearly regressed each of the 18 spike-statistic features of Schneider et al. [33] from the unit embeddings (Figure 7). To prevent leakage between neurons recorded in the same session, we used leave-one-session-out evaluation: for each of the 50 sessions (34,606 neurons in total), we standardized the embeddings with statistics from the other 49 sessions, fitted a ridge-regression probe

Table 30: **Brain region classification accuracies on pretrained mouse vision datasets with session-based splits.** Values are mean accuracies over the folds of 5-fold cross-validation.

Method	3 merged-regions	7 regions
POYO	56.4	27.2
MOJO-POYO	73.7	41.8
MOJO-POYO (A)	78.7	47.0
MOJO-POYO-L(J)	81.7	52.5

on those sessions with its penalty selected from 13 log-spaced values between 10^{-3} and 10^{3}, and evaluated it on all neurons of the held-out session. We report the mean and SD of the per-session R^2 across the 50 held-out sessions. Since the coefficient of variation (CV) is extremely heavy-tailed, we regressed $\log(1 + \mathrm{CV})$ in its place. We find that MOJO embeddings outperform those of purely supervised POYO on every feature. Across MOJO variants, R^2 ranges from 0.57 to 0.61 for log median ISI, from 0.87 to 0.88 for log mean firing rate, and from 0.12 to 0.23 for $\log(1 + \mathrm{CV})$, compared with 0.33, 0.53, and -0.02 for POYO.

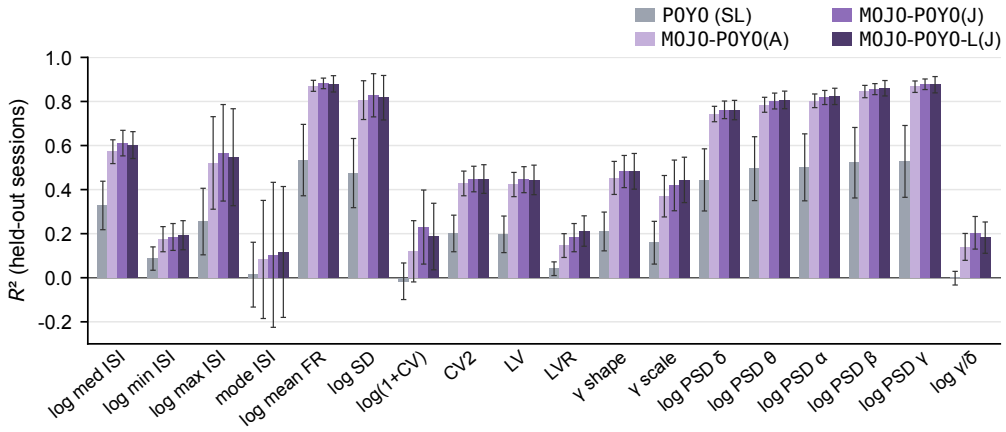


Figure 7: **Single-neuron firing statistics are encoded better by MOJO than by pure-SL POYO.** Leave-one-session-out linear probes of 18 single-neuron spike-statistic features (ISI moments, firing rate, variability indices, gamma-distribution fit parameters, and band-limited PSDs of the binned spike train) from each model's unit embeddings on the Allen Visual Coding Neuropixels dataset. Bars show mean R^2 over 50 held-out sessions and error bars indicate SD across sessions.

E.4 Additional Results on Few-shot Finetuning with Different Sources of Unlabelled Data

We conducted an additional few-shot experiment on one session from stimulus set 1 of the mouse vision dataset, finetuning MOJO-POYO on 5-shot, 8-way DG orientation classification with two sources of unlabelled data from the new session:

- **Task:** other DG trials with their orientation labels removed.
- **Spontaneous:** blocks of spontaneous activity with no visual stimulus.

To control for the amount of unlabelled data, we used two matched scales: a full setting with 190 DG trials (about 380 s of neural activity) or about 390 s of spontaneous activity, and a half setting with 95 trials (about 190 s) or about 195 s of spontaneous activity. For the spontaneous data, we evaluated two different selections of blocks (Spont. 1 and Spont. 2). As shown in Figure 8, both sources substantially improve accuracy over using no unlabelled data (from 64.7% to 77.6–82.2%), spontaneous activity helps as much as unlabelled task trials, and accuracy increases with the amount of unlabelled data. Through t-tests, we find that every unlabelled data source at the full amount improves over no unlabelled data ($p < 0.05$), and using spontaneous activity does not differ from using unlabelled task trials at either amount ($p > 0.3$). Since spontaneous activity carries no information about the task, this suggests that the gains come mainly from SSL shaping the new session's unit embeddings, which any neural activity from that session can provide.

E.5 Additional Results on Joint Monkey-Mouse Pretraining

As shown in Table 31, joint monkey-mouse training with MOJO speeds up learning of the mouse vision tasks, with higher validation performance than the mouse-only model in early epochs. The joint model also maintains its performance on the monkey data: it reaches a decoding R^2 of 0.8551 averaged over monkey T sessions and 77.51% multi-region classification accuracy from the unit embeddings of the pretraining monkey sessions, compared with 0.8548 and 78.24% for the monkey-only model. A larger jointly trained model improves monkey region classification further to 81.02% with similar decoding performance (0.8533).

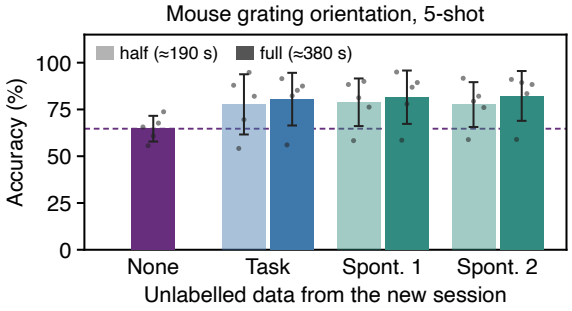


Figure 8: **Few-shot finetuning with different sources of unlabelled data.** 5-shot, 8-way DG orientation accuracy of MOJO-POYO on a held-out mouse session, finetuned with additional unlabelled data from that session: task trials with labels removed, or spontaneous activity with no stimulus (two selections of blocks). Light and dark bars use the half (about 190 s) and full (about 380 s) amounts of unlabelled activity respectively. Bars show mean ± SD over 5 seeds and dots show individual seeds.

Table 31: **Mouse validation decoding accuracy at epoch 200.** Best performing models are in boldface (1st) and underlined (2nd).

		Validation Acc. (%)			
	Method	NS	DG	TF	Avg.
MOUSE	POYO (Lamb)	71.33	94.55	98.36	88.08
	POYO (AdamW)	51.08	94.55	91.59	79.07
	MOJO-POYO	<u>73.01</u>	<u>98.02</u>	<u>99.23</u>	<u>90.09</u>
	MOJO-POYO (A)	**74.47**	**98.82**	**99.49**	**90.93**
JOINT	POYO (Lamb)(J)	65.72	95.29	97.01	86.00
	POYO (AdamW)(J)	77.04	96.25	98.52	90.60
	MOJO-POYO (J)	<u>78.06</u>	**98.20**	<u>99.00</u>	<u>91.76</u>
	MOJO-POYO-L(J)	**79.50**	<u>97.86</u>	**99.21**	**92.19**

We ran the same curriculum with POYO, and the outcome depended on the optimizer. With Lamb, we observed no clear transfer in the validation curves. On the other hand, with AdamW, the mouse-only model learned much more slowly, while the joint model learned the mouse tasks much faster. AdamW also changed POYO's unit embeddings more: their standard deviation rose from 0.053 at the start of joint training to 0.093 after 200 epochs, compared with 0.031 to 0.036 with Lamb, possibly because of Lamb's layer-wise update rescaling [65]. We hypothesize that small embedding updates suffice when training on homogeneous data, but that transferring to highly heterogeneous data, such as across species and tasks, benefits from larger updates to the unit embeddings; confirming this would require further experiments.

F Additional Experiments on Human Speech Datasets

Table 32 reports the performance on the human speech tasks of MOJO with a model size matched to POYO, denoted MOJO-POYO (M). Even at the same size, MOJO outperforms purely supervised POYO on all tasks.

Table 32: **Results on human speech tasks.** Accuracy (%), averaged over 4 participants and 3 seeds. (M): model size matched to POYO.

Models	Vowel (%)	Consonant (%)	Syllable (%)
POYO	60.0	25.3	16.5
MOJO-POYO (M)	67.1	31.5	22.9
MOJO-POYO	68.6	32.9	24.3

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: Our main claims and contributions are stated in the abstract and introduction, and each is supported by the experiments in Section 3 and the Appendices. Aspirational goals are stated as such.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: The limitations of our work are covered in the Discussion section, where we also propose directions for future work to address them.

   Guidelines:

   - The answer [N/A] 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: [N/A]

Justification: There are no theoretical results in this work.

Guidelines:

- The answer [N/A] 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: Datasets and preprocessing, model architectures and configurations, and training details are described in Appendices A, B, and C, respectively. Our code is publicly available through our project page: `mojo-brain.github.io`.

   Guidelines:

   - The answer [N/A] 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: All datasets used in this work are publicly available and cited. Our code, with instructions to reproduce the main experiments, is publicly available through our project page: mojo-brain.github.io.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.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://neurips.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) necessary to understand the results?

Answer: [Yes]

Justification: Datasets and preprocessing, model architectures and configurations, and training details are described in Appendices A, B, and C, respectively.

Guidelines:

- The answer [N/A] 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: Standard deviations and the results of statistical significance tests are reported in the tables and figures where relevant, and each caption states what they are computed over.

Guidelines:

- The answer [N/A] 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: GPU types and numbers of epochs are given in Appendix C. Multi-session pretraining of `MOJO-POYO` and `MOJO-POGRU` takes 44.5 and 34.2 hours, respectively, on four NVIDIA H100 GPUs. Each finetuning run typically takes under 25 minutes on a single NVIDIA L40S GPU.

   Guidelines:

   - The answer [N/A] 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: We have reviewed the NeurIPS Code of Ethics, and our research conforms to it in every respect.

   Guidelines:

   - The answer [N/A] 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: We discuss potential positive and negative impacts of our work in a dedicated Broader Impact section following the Discussion.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: Our work does not pose any such risks.

Guidelines:

- The answer [N/A] 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 datasets, models, and code packages used in this work have been cited.

Guidelines:

- The answer [N/A] 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, along with documentation, is publicly available through our project page: `mojo-brain.github.io`.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: Our work does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: Our work only uses publicly available data and does not conduct research with human subjects.

    Guidelines:

- The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not a component of the core methods introduced in our work.

    Guidelines:

    - The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.