Mine your own view: A self-supervised approach for learning representations of neural activity

Mehdi Azabou Georgia Tech
Mohammad Gheshlaghi Azar DeepMind
Ran Liu Georgia Tech
Chi-Heng Lin Georgia Tech
Erik C. Johnson JHU-APL
Kiran Bhaskaran-Nair WashU-St Louis
Max Dabagia Georgia Tech
Bernardo Avila-Pires DeepMind
Lindsey Kitchell JHU-APL
Keith B. Hengen WashU-St Louis
William Gray-Roncal JHU-APL
Michal Valko DeepMind
Eva L. Dyer Georgia Tech

Abstract

Traditionally, neural decoding has been performed through supervised approaches that aim to map specific behaviors or stimuli to specific neural activity patterns through labeled data. However, the representations learned through a supervised approach typically require simple trial structure and repetitive behaviors, and fail to generalize to new datasets. Here, we ask whether we can use self-supervised learning principles to learn more robust and generalizable representations of neural activity. Rather than using labels to guide learning, we essentially ask the network to build a representation that makes it easy to predict across nearby points in time, as well as across adaptively "mined" samples that are nonlocal but close in terms of their representations in the network. We show that by incorporating nonlocal mined views into the system, and predicting across distinct time points, the network can build representations that allow for more faithful decoding on downstream tasks.

1 Introduction

Traditionally, neural decoding has been performed through supervised approaches that aim to map neural activity patterns to specific behaviors or stimuli using labeled data [1]. However, in "real world" behavior, the concept of labels is often inappropriate, as motor patterns are variable, and higher order variables, such as attention, intention, and association, and engagement, are nearly impossible to assess. Thus, we need methods for learning representations of brain activity that do not rely on labels and can provide a robust signature of brain states that can be reliably decoded in diverse conditions.

The promise of unsupervised representation learning is that the meaningful "neural code" can be extracted without experimenter insight or bias. However, when the objective is efficient reconstruction, noise and other nuisance variables can easily overwhelm the learned representations, which only capture the latent factors when these factors also lead to high-fidelity reconstruction. As such, these approaches are incentivized to reconstruct noise, and often fail to generalize to unseen time points.

Here, we examine whether principles of "self-supervision" can be used to learn representations that reveal a more robust link between the brain and behavior. Rather than using labels to guide

Contact info: Mehdi Azabou - mazabou@gatech.edu, Eva L. Dyer - evadyer@gatech.edu. Code and demos: http://nerdslab.github.io/myow

Workshop on Self-Supervised Learning - Theory and Practice, 35th Conference on Neural Information Processing Systems

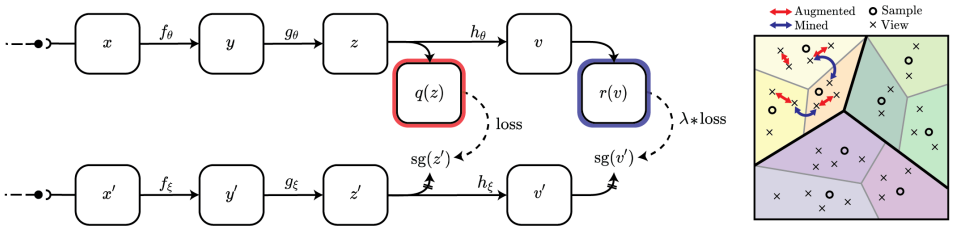


Figure 1: *Overview of our approach.* The architecture of the system, shown of the left, consists of two networks, the online network (top) and the target network (below). The augmented views are compared by the predictor highlighted in red, and the mined views by the predictor highlighted in blue. During mining, views are selected by computing the k-nearest neighbors of the anchor online representation among the target representations of the pool of candidates. One of the nearest neighbors is randomly selected to be the mined view.

learning, we essentially ask the network to build a representation that maps similar points in time to similar points in the latent space. Once a local predictive representation has been established, we then incorporate prediction over distant points in time as a separate learning task. We show that learning across nearby points in time as well as across "mined" samples that are close in terms of their representations in the network provides a combination of local and global losses that can more effectively organize and cluster related brain states.

We tested our approach on trial-based reaching datasets from non-human primates, and on free behavior in the rodent visual cortex and hippocampus. When tested on these diverse neural datasets, we found robust results using similar augmentations (i.e., temporal shifts, neuron dropout), suggesting that this idea can be used widely in different systems using the same basic principles. We show that by incorporating nonlocal but similar time points into the system, and predicting across these distinct time points, the network can build time-invariant representations that allow for more faithful decoding on downstream tasks.

2 Mine Your Own vieW (MYOW)

In this section, we introduce MYOW, a self-supervised approach for incorporating across-sample prediction as an auxiliary self-supervised task to complement prediction across augmented views (see Figure 1). A PyTorch implementation of MYOW is made available at: [hidden for review].

Approach. MYOW is built on top of a recent SSL approach called BYOL [2] which builds representations by maximizing the similarity between augmented views passed through dual networks. However, instead of only comparing positive views generated from the same sample, MYOW actively mines views by finding other samples that have similar representations using a simple nearest neighbors search.

More precisely, MYOW works by first taking a sample $\mathbf{s} \in \mathcal{D}$ from the dataset, and generating two augmented views $\mathbf{x}, \mathbf{x}'$ using transformations t, t' sampled from a set $\mathcal{T}$, where $\mathbf{x} = t(\mathbf{s})$ and $\mathbf{x}' = t'(\mathbf{s})$. The views are then fed through two networks called the *online* and *target* networks, parameterized by weights θ and ξ, respectively. The encoders produce representations $\mathbf{y} = f_\theta(\mathbf{x})$ and $\mathbf{y}' = f_\xi(\mathbf{x}')$, which are then passed through a projector (simple neural network) to obtain $\mathbf{z} = g_\theta(\mathbf{y})$ and $\mathbf{z}' = g_\xi(\mathbf{y}')$.

Mined views are selected by finding two examples that are similar to each other in the latent space through a randomized nearest-neighbor algorithm. More precisely, the k-nearest neighbors of a point $\mathbf{y}_m = f_\theta(\mathbf{x}_m)$ are computed by projecting other samples through the online (or target) encoder and randomly selecting one of these neighbors as the mined view $\mathbf{y}'_m$. Mined views are passed into a second projector network to obtain $\mathbf{v}_m = h_\theta(g_\theta(\mathbf{y}_m))$ and $\mathbf{v}'_m = h_\xi(g_\xi(\mathbf{y}'_m))$. The projections in the target network are paired with their respective predictors: q_θ forms predictions across augmented views and r_θ forms predictions across mined views.

Finally, MYOW learns a representation by minimizing both augmented and mined prediction errors through the following loss:

$$\mathcal{L} = \underbrace{d(q_\theta(\mathbf{z}), \mathbf{z}')}_{\text{Augmentation Loss}} + \lambda \underbrace{d(r_\theta(\mathbf{v}_m), \mathbf{v}'_m)}_{\text{Mining Loss}}, \quad \text{with } d(\mathbf{u}, \boldsymbol{\nu}) = -\frac{\langle \mathbf{u}, \boldsymbol{\nu} \rangle}{\|\mathbf{u}\|_2 \|\boldsymbol{\nu}\|_2}, \tag{1}$$

Table 1: *Accuracy (in %) for classification on CIFAR-10, CIFAR-100 and Tiny Imagenet.* We report the linear evaluation accuracies for different architectures and datasets. For CIFAR-100, we report both accuracies under linear evaluation on CIFAR-100 and CIFAR-20. Results for SimCLR are reported from [3].

	ResNet-18				*ResNet-50*		
Method	CIFAR-10	CIFAR-100	CIFAR-20	Tiny ImageNet	CIFAR-10	CIFAR-100	CIFAR-20
SimCLR *	91.80	66.83	-	48.84	91.73	-	-
BYOL	91.71	66.70	76.90	51.56	92.12	67.87	77.38
MYOW	**92.10**	**67.91**	**78.10**	**52.58**	**93.18**	**68.69**	**78.87**

where λ is a weight that regulates the contribution of the mined views in the objective; in practice, λ has an initial linear warmup period of a few epochs. Just as in BYOL, we symmetrize the distance between augmented views by feeding $\mathbf{x}'$ and $\mathbf{x}$ to the online and target network, respectively.

The loss $\mathcal{L}$ is optimized only in terms of θ and ξ is updated according to a moving average of θ. In particular, we update the online and target networks according to the following:

$$\theta \leftarrow \text{optimize}(\theta, \nabla_\theta \mathcal{L}, \eta), \quad \xi \leftarrow \tau\xi + (1-\tau)\theta, \tag{2}$$

where $\tau \in [0, 1]$ is a momentum parameter, and η is the learning rate used to optimize the weights of the online network.

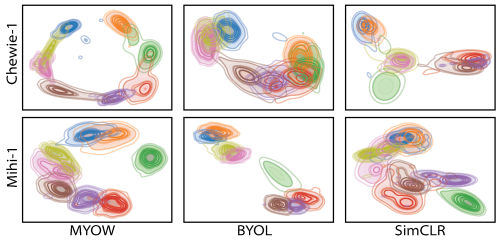
3 Evaluations

3.1 Validating our method on image datasets

The performance of self-supervised learning methods are typically quantified in the context of image recognition. To directly compare our approach with BYOL and other competitors, we applied MYOW to standard benchmarks of image recognition including, CIFAR-10, CIFAR-100 and Tiny ImageNet (Table 1). Consistently, MYOW yields competitive results with these state-of-the-art methods, and outperforms BYOL even when they share the same random seed and the same hyper-parameters. We provide ablation studies for our method in Appendices F-I. Overall, we find that MYOW provides competitive performance with state-of-the-art methods on the vision datasets we tested, providing evidence that our method works well in domains where SSL frameworks have been shown to yield stable and meaningful representations.

3.2 Neural datasets: Examining and comparing representation quality

Decoding movements from macaque motor cortex. In our experiments on neural data, we first considered decoding of reach direction from populations of neurons in the motor cortex. In these datasets, there is a direct connection between the neural state (brain activity across many neurons) and the underlying movements (behavior). Thus, we wanted to assess the quality of the representations learned from these datasets by asking how well we can predict the reach direction from neural activity. If we have a good representation, we should be able to better separate reach direction from the neural activities. To quantify this, we will use a linear readout to predict the reach direction, and report the classification accuracy. We describe our experimental setup in Appendix C, and in Appendix D, we establish the augmentations (Dropout, Temporal jitter, and more) used for spiking neural activity.



	Chewie-1	Chewie-2	Mihi-1	Mihi-2
Supervised	63.29	72.29	63.64	61.49
pi-VAE	65.63	60.60	62.44	63.26
AE	48.40	46.79	50.94	55.19
RP	59.21	50.69	57.78	53.76
TS	60.16	49.48	59.23	54.10
SimCLR	61.36	51.62	59.41	56.29
BYOL	66.65	64.56	72.64	67.44
MYOW	**70.54**	**72.33**	**73.40**	**71.80**

Figure 2: (Left) t-SNE projection of the learned representations obtained when different SSL methods are applied. This shows how MYOW reveals the underlying structure of the task, as clusters are organized in a circle. (Right) Accuracy (in %) in the prediction of reach direction from spiking neural activity.

We compare our approach with several self-supervised methods, including state-of-the-art methods BYOL and SimCLR, as well as two widely used self-supervised tasks recently applied to EEG data

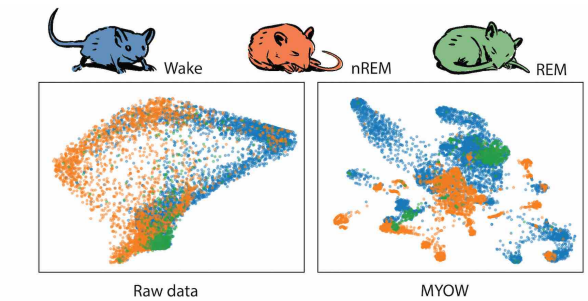
called Relative Positioning (RP) and Temporal Shuffling (TS) [4]. RP trains the network by classifying whether two samples are temporally close, while TS takes in three samples and learns whether they are in the right order or if they were shuffled. In addition to these self-supervised methods, we also train a Multi-layer Perceptron (MLP) classifier (Supervised) using weight regularization and dropout (in nodes in intermediate layers in the network), an autoencoder (AE), and a state-of-the-art supervised approach for generative modeling of neural activity (pi-VAE) that leverages behavioral labels to condition and decompose the latent space [5].

We find that MYOW consistently outperforms other approaches and that contrastive methods that rely on negative examples (SimCLR, RP and TS) fall behind both MYOW and BYOL. We also find that MYOW generalizes to unseen data more readily than others; in some cases, beating supervised approaches by a significant margin, with over 10% improvement on two datasets. These results are even more impressive considering that we only tune augmentations and hyperparameters on Chewie-1 and find that MYOW consistently generalizes across time and individuals. We thus show that by integrating diverse views (across trials) through mining into our prediction task, we can more accurately decode movement variables than supervised decoders.

When we visualize the learned representation in Figure 2, we notice that MYOW organizes representations in a way that is more reflective of the global task structure, placing reach directions in their correct circular order. In contrast, we find that in both individuals, other methods tend to distort the underlying latent structure of the behavior when visualized in low-dimensions (Appendix E). We conjecture that across-sample predictions (including those across different reach directions), may be responsible for capturing this kind of higher-level structure in the data.

Decoding arousal states from the rodent brain during free behavior. Next, we applied MYOW to datasets of single unit spiking from the rodent cortex and hippocampus, where we test our ability to decode arousal states (REM, nREM, Wake) from the learned representations in single unit spiking. Brain state is traditionally evaluated at the level of correlated changes in low-resolution physiological changes, such as EEG, EMG, and behavior. While brain states influence the summary statistics of population spiking activity [6], such changes are highly variable and are insufficient to predict brain state [7]. Here, we provide the first demonstration that individual epochs of wake, NREM, and REM are robustly encoded in the spiking of hippocampal and cortical neurons. This encoding takes place not in primitive statistics, but in a latent space that is readily learned by MYOW. Furthermore, despite the strong class imbalance, the trends are similar to that of our earlier experiments, with MYOW providing robust performance, exceeding that of the supervised baseline, and outperforming other self-supervised methods.

In these datasets, the animal is "untethered" and can roam around in its cage without any task or explicit instructions. Unsurprisingly, in these free-behaving conditions, we find a great deal of variability in the latent state beyond the coarse labels that we have access to. When we visualize the representation learned by MYOW in Figure 3, we find that the network separates different parts of the behavior space, revealing subspaces of neural states that are otherwise unobservable when examining the raw data.



	Rat-V1	Mouse-CA1
Supervised	86.34	93.01
pi-VAE	73.10	82.48
AE	34.17	57.73
RP	82.93	82.12
TS	82.45	81.93
SimCLR	81.03	81.94
BYOL	85.42	93.24
MYOW	**88.01**	**93.70**

Figure 3: (Left) Visualizations of raw and latent spaces (using t-SNE) of 12 hour recordings from mouse (CA1) during free behavior, including sleep and wake. One variable of interest is the arousal state (REM, nREM, Wake). (Right) F1-score (in %) in the prediction of arousal state from spiking neural activity.

4 Conclusion

This paper provides an important step towards applying self-supervised methods to learn representations of neural activity. Our results in this domain are compelling: we typically obtain better generalization than supervised methods trained with dropout and weight decay. Through the inclusion of temporal structure into our framework and architecture, we may be able to improve this approach even further and capture dynamics over longer timescales.

In our application to spiking neural data, we demonstrate that both dropout and temporal augmentations are necessary for building meaningful representations of different brain states. Similarly in neural circuits, neurons are unable to send direct signals to every other neuron in a downstream population; thus, target areas receiving signals may need to predict future brain states from partial information [8]. Our results suggest that it may be fruitful to try to understand how brains may leverage dropout to build predictive representations, and that a theoretical understanding of SSL might yield insight into these processes.

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*, vol. 7, no. 4, 2020.

[2] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, *et al.*, "Bootstrap your own latent: A new approach to self-supervised learning," *arXiv preprint arXiv:2006.07733*, 2020.

[3] A. Ermolov, A. Siarohin, E. Sangineto, and N. Sebe, "Whitening for self-supervised representation learning," *arXiv preprint arXiv:2007.06346*, 2020.

[4] H. Banville, I. Albuquerque, A. Hyvärinen, G. Moffat, D.-A. Engemann, and A. Gramfort, "Self-supervised representation learning from electroencephalography signals," in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, 2019.

[5] D. Zhou and X.-X. Wei, "Learning identifiable and interpretable latent models of high-dimensional neural activity using pi-VAE," *arXiv preprint arXiv:2011.04798*, 2020.

[6] B. O. Watson, D. Levenstein, J. P. Greene, J. N. Gelinas, and G. Buzsáki, "Network homeostasis and state dynamics of neocortical sleep," *Neuron*, vol. 90, no. 4, pp. 839–852, 2016.

[7] K. B. Hengen, A. T. Pacheco, J. N. McGregor, S. D. Van Hooser, and G. G. Turrigiano, "Neuronal firing rate homeostasis is inhibited by sleep and promoted by wake," *Cell*, vol. 165, no. 1, pp. 180–191, 2016.

[8] R. P. Rao and D. H. Ballard, "Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects," *Nature Neuroscience*, vol. 2, no. 1, pp. 79–87, 1999.

[9] R. P. Sheridan, "Time-split cross-validation as a method for estimating the goodness of prospective prediction," *Journal of Chemical Information and Modeling*, vol. 53, no. 4, pp. 783–790, 2013.

[10] J. E. Chung, J. F. Magland, A. H. Barnett, V. M. Tolosa, A. C. Tooker, K. Y. Lee, K. G. Shah, S. H. Felix, L. M. Frank, and L. F. Greengard, "A fully automated approach to spike sorting," *Neuron*, vol. 95, no. 6, pp. 1381 – 1394.e6, 2017.

[11] A. P. Buccino, C. L. Hurwitz, J. Magland, S. Garcia, J. H. Siegle, R. Hurwitz, and M. H. Hennig, "Spikeinterface, a unified framework for spike sorting," *bioRxiv*, 2019.

[12] A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge, "Deeplabcut: markerless pose estimation of user-defined body parts with deep learning," *Nature Neuroscience*, vol. 21, pp. 1281–1289, 2018.

[13] J. Y. Cheng, H. Goh, K. Dogrusoz, O. Tuzel, and E. Azemi, "Subject-aware contrastive learning for biosignals," *arXiv preprint arXiv:2007.04871*, 2020.

[14] P. Sarkar and A. Etemad, "Self-supervised learning for ecg-based emotion recognition," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3217–3221, 2020.

[15] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[16] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, and S. Levine, "Time-contrastive networks: Self-supervised learning from video," *arXiv preprint arXiv:1704.06888*, 2018.

[17] D. Dwibedi, J. Tompson, C. Lynch, and P. Sermanet, "Learning actionable representations from visual observations," *arXiv preprint arXiv:1808.00928*, 2019.

[18] P. H. Le-Khac, G. Healy, and A. F. Smeaton, "Contrastive representation learning: A framework and review," *IEEE Access*, vol. 8, p. 193907–193934, 2020.

[19] H. Banville, O. Chehab, A. Hyvarinen, D. Engemann, and A. Gramfort, "Uncovering the structure of clinical EEG signals with self-supervised learning," *Journal of Neural Engineering*, 2020.

[20] X. Bouthillier, K. Konda, P. Vincent, and R. Memisevic, "Dropout as data augmentation," *arXiv preprint arXiv:1506.08700*, 2016.

Appendix

A Algorithm

Algorithm 1: Mine Your Own vieW - MYOW

input : Dataset $\mathcal{D}$; online network $f_\theta, g_\theta, h_\theta$; target network f_ξ, g_ξ, h_ξ; dual predictors q_θ, r_θ; learning rate η; momentum τ; mining weight λ; batch size B; pool batch size L.

init $\xi \leftarrow \theta$

while not converging **do**

// Augment views

Fetch a mini-batch $\{\mathbf{s}_i\}_B$ from $\mathcal{D}$

for $i \in \{1...B\}$ (in parallel) **do**

Draw functions: $t \sim \mathcal{T}, t' \sim \mathcal{T}$

$\mathbf{x}_i = t(\mathbf{s}_i),\ \mathbf{x}'_i = t'(\mathbf{s}_i)$

$\mathbf{z}_i = g_\theta(f_\theta(\mathbf{x}_i));\ \mathbf{z}'_i = g_\xi(f_\xi(\mathbf{x}'_i))$

$\mathbf{u}_i = g_\xi(f_\xi(\mathbf{x}_i));\ \mathbf{u}'_i = g_\theta(f_\theta(\mathbf{x}'_i))$

end

// Mine views

Fetch a mini-batch $\{\mathbf{c}_j\}_L$ from $\mathcal{D}$

for $j \in \{1...L\}$ (in parallel) **do**

Draw function: $t \sim \mathcal{T}_m$

$\mathbf{x}_{c,j} = t(\mathbf{c}_j);\ \mathbf{y}'_{c,j} = f_\xi(\mathbf{x}_{c,j})$

end

Let $\mathcal{S} = \{\mathbf{y}'_{c,j}\}_{j=1}^{L}$

for $i \in \{1...B\}$ (in parallel) **do**

Draw function: $t \sim \mathcal{T}_m$

$\mathbf{x}_{m,i} = t(\mathbf{s}_i);\ \mathbf{y}_{m,i} = f_\theta(\mathbf{x}_{m,i})$

Find $\mathcal{N}_k(\mathbf{y}_{m,i})$, the k-NNs of $\mathbf{y}_{m,i}$ in $\mathcal{S}$

Randomly select $\mathbf{y}'_{m,i}$ from $\mathcal{N}_k(\mathbf{y}_{m,i})$

$\mathbf{v}_i = h_\theta\left(g_\theta\left(\mathbf{y}_{m,i}\right)\right); \mathbf{v}'_i = h_\xi(g_\xi(\mathbf{y}'_{m,i}))$

end

// Update parameters

$\mathcal{L} = \sum_i d(q_\theta(\mathbf{z}_i), \mathbf{z}'_i) + d(q_\theta(\mathbf{u}'_i), \mathbf{u}_i) + \lambda d(r_\theta(\mathbf{v}_i), \mathbf{v}'_i)$

$\theta \leftarrow$ **Optimizer**$(\theta, \mathcal{L}/B, \eta)$

$\xi \leftarrow \tau\xi + (1-\tau)\theta$

end

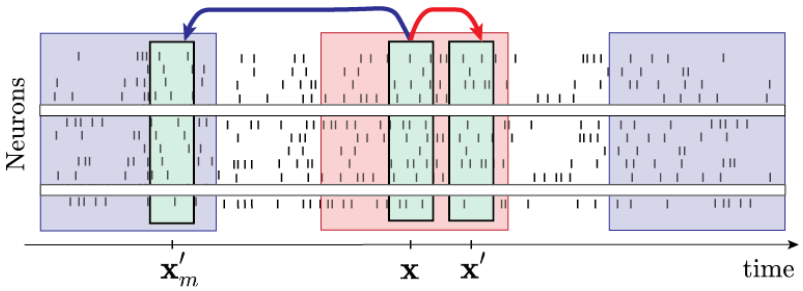


Figure S1: *Visualization of augmentations used for neural activity.* Within a small local window around each anchor sample, we consider the nearby samples (red) to be potential positive examples. Outside of a safe zone, we can label more distant samples (blue) as either negative examples (in contrastive learning) or we can also use these points as candidate views to mine from (in MYOW). Randomized dropout is illustrated via white bars corresponding to the dropping of the same neurons in all three views.

B Mining: Implementation details

At a given training iteration, for every sample in the batch, we mine for views in the same pool of candidates of size L. Depending on the type of data, the mining for a given sample can be restricted to a subset of that pool of candidates.

When training MYOW on neural datasets, or temporal datasets in general, we restrict mining for a given sample to candidates that are temporally farther in time, as illustrated in Figure S1. Implementation-wise, we use a global pool of candidates of size L for simplicity, then when computing the distance matrix used to determine the k-nearest neighbors, we mask out the undesired correspondences in the matrix.

C Experimental details: Neural data

C.1 Application 1: Decoding movements from motor cortex

Details on neural and behavioral datasets in movement decoding task. Neural and behavioral data were collected from two rhesus macaque monkeys (Chewie, Mihi). Both individuals performed a standard delayed center-out movement paradigm (reaching experiment). The subjects were seated in a primate chair and grasped a handle of a custom 2-D planar manipulandum that controlled a computer cursor on a screen. In the first dataset from Chewie, the individual began each trial by moving to a 2 x 2 x 2 cm target in the center of the workspace, and was instructed to hold for 500-1500 ms before another 2 cm target was randomly displayed in one of eight outer positions regularly spaced at a radial distance of 8 cm. For Mihi, this is followed by another variable delay period of 500 to 1500 ms to plan the movement before an auditory 'Go' cue. The sessions with Chewie omitted this instructed delay period and the 'Go' cue was provided when the outer target appeared. Both individuals were required to reach to the target within 1000-1300 ms and hold within it for 500 ms to receive an auditory success tone and a liquid reward.

Both individuals were surgically implanted a 100- electrode array (Blackrock Microsystems, Salt Lake City) in their primary motor cortex (M1). To record the spiking activity of single neural units, threshold crossings of six times the root-mean square (RMS) noise on each of the 96 recording channels are initially recorded. After each session, the neural waveform data was sorted using Offline Sorter (Plexon, Inc, Dallas, TX) to identify single neurons and discarded all waveforms believed to be multi-unit activity.

Data is only recorded when the primate is performing the reaching task, we note such instance a "trial". We split the trials time-wise, using a $70/10/20$ ratio, to obtain our training, validation and test sets. The temporal splits gives us a better estimate of the prospective prediction compared to a random split [9]. The activity of individual neurons was binned (100 ms intervals) to produce firing rates for roughly 150 neurons across two days.

Class of transformations. During training, we generate augmented views and mined views using the following transformations ($\mathcal{T} = \mathcal{T}'$):

- Temporal Jitter: a sample within 200ms is used as a positive example.
- Dropout: mask out neurons with a probability uniformly sampled between $0.$ and 0.2.
- Noise: add gaussian noise with standard deviation of 1.5, with a probability of 0.5.
- Pepper or Sparse additive noise: increase the firing rate of a neuron by a 1.5 constant with a probability of 0.3. This augmentation is applied on the sample with a probability of 0.5.

Because these datasets correspond to a collection of trials, we restrict mining to candidates that are in different trials from the anchor sample.

Network Architecture. For the encoder, we use an MLP which is 4 blocks deep. Each block consists of a linear layer with output size 64 followed by batch normalization (BN) and rectified linear units (ReLU). The final layer has an output size of 32 and no BN or activation. We don't use projectors, predictor q_θ used for augmented views is MLP$(32, 128, 32)$, and predictor r_θ used for mined views is MLP$(32, 128, 32)$.

Training. We use the AdamW optimizer with a learning rate of 0.02 and weight decay of $2 * 10^{-5}$. After a linear warmup period of 100 epochs, the learning rate is decayed following a cosine decay scheduler. The exponential moving average parameter τ is also decayed from 0.98 to $1.$ following a cosine decay scheduler. We train `MYOW` for 1000 epochs and use a batch size of $B = 512$, as well as a pool batch size of $L = 1024$, and $k = 5$. We use a mining weight of $\lambda = 1.$ linearly ramped-up for 10 epochs. `BYOL` is trained using the same relevant hyperparameters.

Reach direction prediction task. The downstream task we use to evaluate the learned representation, is the prediction of the reach direction during movement. There are 8 possible reach direction in total. Unlike most classification tasks, there is an inherent cyclic ordering between the different classes. Thus, we estimate the angles corresponding to each reach direction, and evaluate their cosine and sine. The linear layer outputs a 2d vector $[x, y]$ that predicts $[\cos\theta_r, \sin\theta_r]$. We train the network using a mean-squared error loss. Once the network is trained, to readout out the predicted reach direction label, we use the following formula:

$$l_{\text{predicted}} = \lfloor \frac{4}{\pi}(\text{atan2}(y, x) \bmod 2\pi) \rceil \tag{3}$$

Evaluation Procedure. We train a linear classifier on top of the frozen representation of the encoder network and report the accuracy on the test sets. The linear layer is trained for 100 epochs using the AdamW optimizer with a learning rate of 0.01. We sweep over 20 values of the weight decay $\{2^{10}, 2^8, 2^6, \ldots, 2^6, 2^8, 2^{10}\}$ on the valudation set, and report the accuracies of the best validation hyperparameter on the test set.

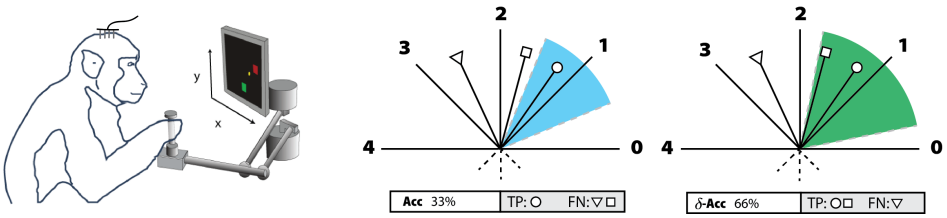
More specifically, we report two different metrics that are computed over the validation set. The Accuracy is the conventional classification accuracy that is obtained when assigning the predicted reach angle to the closest corresponding reach direction. The second metric, δ-Acc, is obtained when considering that a prediction is a true positive if it is within a slightly larger window around the true reach direction (an analogy to top-k metrics). (Fig S2-b).

$$\text{TP}_{\text{Acc}} = |\frac{4}{\pi}(\text{atan2}(y, x) \bmod 2\pi) - l| < 1 \qquad \text{TP}_{\delta-\text{Acc}} = |\frac{4}{\pi}(\text{atan2}(y, x) \bmod 2\pi) - l| < 1.5$$

C.2 Application 2: Decoding sleep states from rodent cortex

Details on neural and behavioral datasets in arousal state decoding. Extracellular single unit spiking was collected from chronically implanted, freely behaving animals. Tetrode arrays were implanted without drives into mouse CA1 (C57BL/6) and rat V1 (Long Evans). Following recovery, neural data were recorded at 25 kHz continuously during free behavior. Raw data were processed and clustered using standard pipelines. Data was bandpassed (500-10,000 Hz) and clustered using MountainSort [10, 11]. Single units were identified in the clustering output via XGBoost.

Arousal state was scored using standard polysomnographic methods. Local field potentials (LFP) from 8/64 channels were averaged together, lowpassed (250 Hz), and downsampled. Video (15 fps) was processed using a CNN [12] to track animal position and movement. Trained human scorers evaluated the LFP power spectral density and integral of animal movement to evaluate waking, NREM and REM sleep.



(a) (b)

Figure S2: *Reach direction prediction task.* (a) Sketch of primate performing reaching task. (b) Illustration depicting how the accuracy and δ-accuracy are computed. The three points have reach direction 1 as their ground truth. TP is true positive and FN is false negative. The highlighted areas correspond to the area a point should fall in to be considered a true positive and be counted towards the corresponding accuracy.

We split the 12 hour block of data temporally using an $70/10/20$ ratio, to obtain our training, validation and test sets. The activity of individual neurons was binned (4s intervals) to produce firing rates for roughly 40 and 120 neurons from CA1 and V1, respectively.

Training. We use the same hyperparameters as for the monkey datasets, except that the representation size is larger (64), and the temporal augmentations are different. With temporal jitter, we consider any two samples that are at most 12s apart to be positive examples and when mining we restrict the candidates to be at least 30min before or after the anchor sample.

Arousal state prediction task. We train a linear classifier on top of the frozen representation of the encoder network to predict the arousal state.

D Augmentations for spiking neural data

While self-supervised approaches have not been applied to the multi-neuron recordings that we consider, we take cues from other domains (video, graphs), as well as previous work on electroencephalogram (EEG) data [13, 14], to define simple classes of augmentations for our datasets. Specifically, we consider four different types of augmentations:

Temporal jitter. As in previous work in temporal contrastive learning [15, 16, 17, 18, 19], we can use nearby samples as positive examples for one another.

Randomized dropout. When working with neural data, we consider randomized dropout [20] as an augmentation. The dropout rate is uniformly sampled between p_{min} and p_{max}.

Gaussian noise. Random Gaussian noise with mean 0 and standard deviation 1.5 is applied before normalizing the firing rates.

Random pepper. In contrast to dropout, applying random pepper consists of randomly activating neurons. Similar to the dropout probability, a pepper probability is used to specify the probability of activating a neuron. The activation consists in adding a constant to the firing rate.

In Table S1, we show how different augmentations impact neural datasets. We test the inclusion and combination of these different augmentations, first on our `BYOL` backbone which uses augmented views only. While we find that temporal jitter alone is insufficient to drive learning, when we combine both jitter and dropout, we see a substantial increase in decoding accuracy and qualitative improvements in the resulting representations. In this case, our baseline SSL method, `BYOL`, quickly starts to create meaningful predictive relationships between data, as evidenced by our decoding results and qualitative evaluations of the representations. As we include additional augmentations (*Noise + Pepper*), the performance increases further, but by smaller margins than before. In general, we see these same trends observed throughout our remaining primate datasets and in our experiments on rodent, suggesting that these classes of transformations are good candidates for building SSL frameworks for neural activity.

After establishing a good set of simple augmentations, we then integrate *mined views* with `MYOW`. In this case, we can interpret mined views as *nonlocal brain states* that are not temporally close but can be semantically similar. For instance, in our reaching datasets, `MYOW` will mine outside of the current

Table S2: *How augmentations impact our ability to decode sleep and wake states accurately.* To understand how different augmentations impact the representations obtained with BYOL and MYOW for the two datasets labeled Sleep, we computed the F1-score for different classes of augmentations in two brain areas.

	TS	RDrop	F1-score	
			Rat-V1	Mouse-CA1
BYOL	✓		68.66	87.73
		✓	79.31	88.84
	✓	✓	85.42	93.24
MYOW	✓		72.13	90.01
		✓	85.60	83.33
	✓	✓	88.01	93.70

reach and look for other samples that it can use to build a more unified picture of the brain states as they evolve. Through combining simple augmentations with nonlocal samples with MYOW, we provide an impressive boost in performance over BYOL on this application.

Table S1: *How augmentations impact our ability to decode movements accurately.* To understand how different augmentations impact the representations obtained with BYOL and MYOW for all four datasets, we computed the Accuracy in our reach direction prediction task when we apply a given set of transformations.

	TJ	Drop	Noise	Pepper	Accuracy			
					Chewie-1	Chewie-2	Mihi-1	Mihi-2
BYOL	✓				41.75	40.83	43.98	44.10
		✓	✓	✓	55.70	49.37	47.61	43.12
	✓	✓			61.39	56.48	59.53	58.37
	✓	✓	✓	✓	63.80	57.17	59.50	60.82
MYOW	✓				46.61	42.91	42.08	44.13
		✓	✓	✓	53.15	46.17	51.44	48.72
	✓	✓			67.97	58.21	68.93	63.90
	✓	✓	✓	✓	70.41	60.95	70.48	64.35

In Table S2, we show the impact of both temporal shift and dropout on the performance on rodent datasets. Here, we also find that both components are important to achieving good performance.

E Visualization of the latent neural space

In Figure S3, we provide the visualizations of the latent spaces for all four monkey reach dataset and can identify a common pattern in the structure uncovered by the different methods.

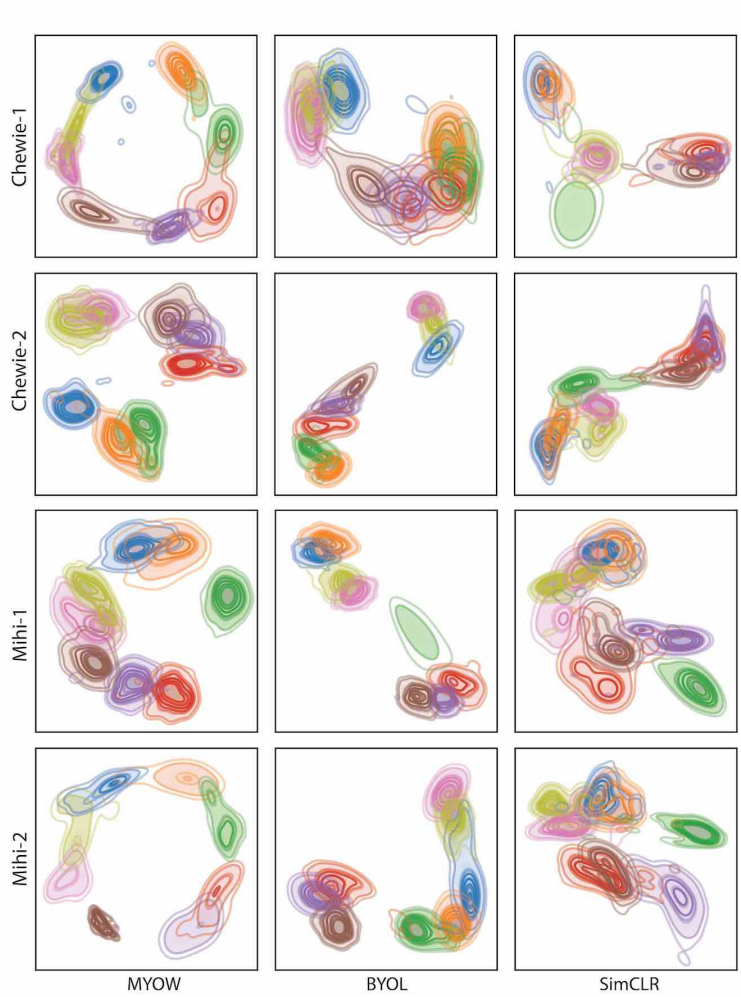


Figure S3: *Visualization of the learned representation*. Using t-SNE, we visualize the representation spaces when training `MYOW`, `BYOL` and `SimCLR` on all four monkey reach datasets.

F Is MYOW worth the extra computational load?

In our experiments, the pool of candidates is resampled on-the-fly at each iteration and thus MYOW does not require a memory bank. While there is an additional, but negligible (less than 2%), memory overhead due to the k-NN operation, the memory requirements for training MYOW are not different from BYOL 's when $L \leq B$. This is because augmented and mined views are forwarded sequentially through the network and gradients are accumulated before updating the weights. To reduce the extra computational overhead due to mining, we use the candidates' target representations instead of their online representations and avoid an extra forward pass. We empirically find that mining in either the online or target network leads to similar results (Appendix G) and thus use this strategy in practice. In this case, MYOW requires 1.5x computation time when compared to BYOL. When memory is not an issue, computation time can be reduced significantly by feeding in all views at the same time. When using a multi-GPU setup, we distribute the computation of the candidate's representations over all GPUs and then have them broadcast their local pools to each other, effectively building a pool of mining candidates of larger size.

In one iteration, MYOW receives 3 batches worth of views, compared to 2 for BYOL. Thus, there is a possibility that MYOW performs better than BYOL simply because of the higher effective batch size used during training. To rule this possibility out, we try both training BYOL for 50% more epochs and training BYOL using a 50% bigger batch size, and report the results in Table S3. We show that the improvements we find through with MYOW go beyond extra training time.

Table S3: *Training BYOL with adjusted batch size and number of epochs.* We report the linear evaluation accuracies on CIFAR-10 using ResNet-18.

	Batch size	Number of epochs	Accuracy
BYOL	512	800	91.71
BYOL	512	1200	91.75
BYOL	768	800	91.65
MYOW	512	800	92.10

When we examine the accuracy curves during training (Figure S4), we find that MYOW surpasses the final accuracy of BYOL after only 300 epochs of training. Thus, in the case of this dataset, we can justify the extra computational load that comes with using MYOW, as it yields better results early on in training.

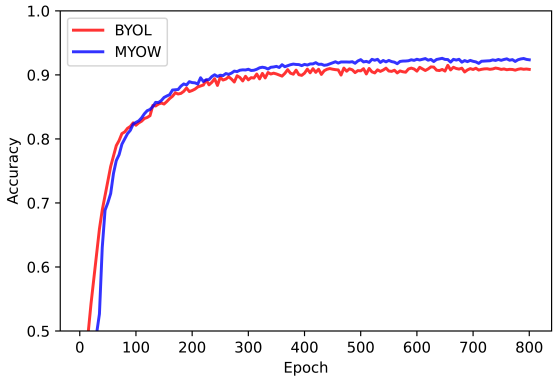


Figure S4: *Accuracy under linear evaluation, CIFAR10, ResNet18.* BYOL (bottom), MYOW (top).

G What makes for good mined views?

In Table S4, we compare the outcomes of using the online representations of the candidates compared to their target representations when looking for the k-nn of the online representation of the

anchor sample. We find that both strategies yield similar results while mining in the target is less computationally expensive.

Table S4: *Mining in online versus. target space.* We report the linear evaluation accuracies on CIFAR-10 using ResNet-18, as well as an approximation of the computational load factor with BYOL as the baseline.

	Mining in	Computational factor	Accuracy
BYOL	-	1.00	91.71
MYOW	online	1.75	92.13
MYOW	target	1.50	92.10

We analyse the views that are being mined when training MYOW on CIFAR-10. In Figure S5, we show a random collection of views paired during mining.

MYOW relies on mining views that are semantically similar, but it is not clear how robust MYOW is to "bad" mined views. While we are not able to give a definitive answer to this question, we find that even when certain mined views have a different class from the anchor samples, MYOW still yields competitive results. In Figure S6, we look at the mining class accuracy, defined as the percentage of mined views that share the same class as their anchor samples, and find that the accuracy steadily increases during training and that the relatively low accuracy at the beginning of training does not hinder the performance of MYOW. The mining class accuracy gives us a better understanding of the mining, but it is not a reflection of the goodness of the mining, as we do not know what makes for a good mined view and whether a inter-class mined views could be "good". We also visualize, in Figure S7, the mining class confusion matrices at epochs 100 and 700 of training.

H Ablation on the projector

In Table S5 and Table S6, we report the results of MYOW on the MNIST and CIFAR-10 datasets for different architectures used for incorporating mined views into our objective: cascaded projectors (used in MYOW), parallel projectors and single projector For MNIST, we show the results for two different settings, weak augmentation (Crop only) and strong augmentation (All). Overall, we find that separating the projection spaces for augmented and mined views is better, with the cascading yielding the best results.

I Ablation on the class of transformations

We study how the choice of the set of transformations used in the mining process, impacts the quality of the representation. In Table S7, we report the accuracies under linear evaluation when we use different classes of transformation $\mathcal{T}'$.

Table S7: *Class of transformation for mined views.* We report the accuracies under linear evaluation of MYOW trained on CIFAR-10 using ResNet-18, for different classes of transformation $\mathcal{T}'$

Crop	Flip	Color jitter	CIFAR-10
✓$(0.2-1.0)$	✓	✓	91.63
✓$(0.8-1.0)$			92.10
			92.08

Figure S5: *Examples of views mined by MYOW.* We visualize the views mined by MYOW during training on the CIFAR-10 dataset at epoch 400.

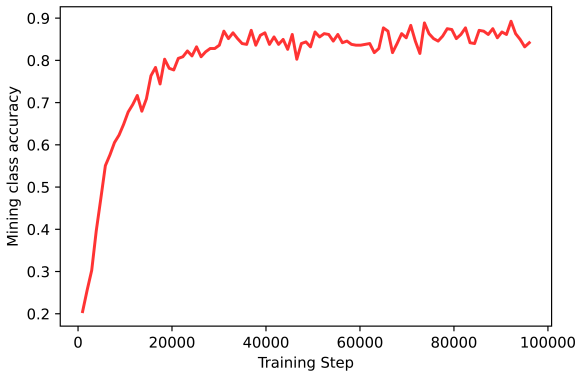


Figure S6: *Mining class accuracy during training.* This metric is reported on CIFAR-10 using ResNet-18.

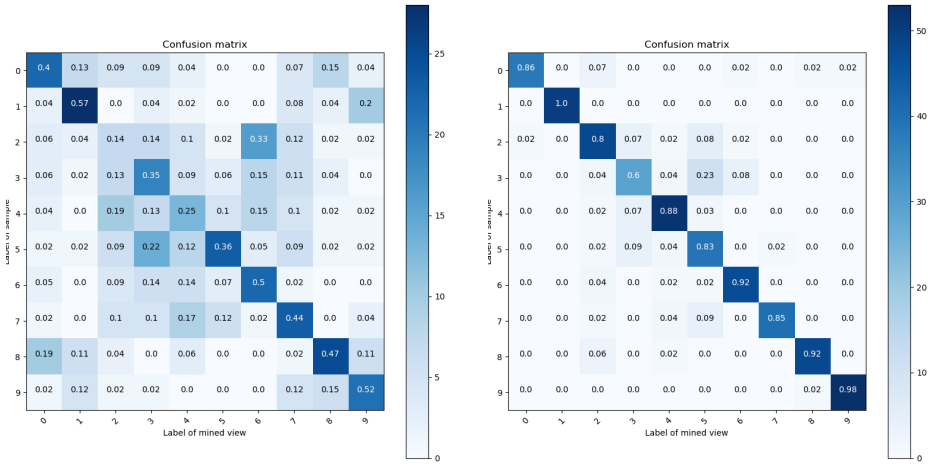


Figure S7: *Mining class confusion matrices at different stages of learning*. We compute the confusion matrix at epochs 100 (right) and (700) when training on CIFAR-10.

Table S5: *Comparing different projector architectures for incorporating mined views.* MNIST classification accuracy (in %) with `MYOW` for different architectures.

Arch	Dimension	MNIST	
		Crop only	All
Cascaded	16	**99.20**	**99.33**
Cascaded	128	98.09	98.80
Parallel	16	96.33	98.71
Parallel	128	97.75	98.12
Single	16	97.13	97.48
Single	128	98.75	98.31

Table S6: *Comparing different projector architectures for incorporating mined views.* CIFAR-10 classification accuracy (in %) with `MYOW` for different architectures.

Arch	CIFAR-10
Cascaded projectors	**92.10**
Parallel projectors	92.01
Single projector	91.84