Mine Your Own vieW: Self-Supervised Learning Through Across-Sample Prediction

Mehdi Azabou [1] **Mohammad Gheshlaghi Azar** [2] **Ran Liu** [1] **Chi-Heng Lin** [1] **Erik C. Johnson** [3]
Kiran Bhaskaran-Nair [4] **Max Dabagia** [1] **Keith B. Hengen** [4] **William Gray-Roncal** [3] **Michal Valko** [5] **Eva Dyer** [1 6]

Abstract

State-of-the-art methods for self-supervised learning (SSL) build representations by maximizing the similarity between different augmented "views" of a sample. Because these approaches try to match views of the same sample, they can be too *myopic* and fail to produce meaningful results when augmentations are not sufficiently rich. This motivates the use of the dataset itself to find similar, yet distinct, samples to serve as views for one another. In this paper, we introduce Mine Your Own vieW (`MYOW`), a new approach for building across-sample prediction into SSL. The idea behind our approach is to actively *mine views*, finding samples that are close in the representation space of the network, and then predict, from one sample's latent representation, the representation of a nearby sample. In addition to showing the promise of `MYOW` on standard datasets used in computer vision, we highlight the power of this idea in a novel application in neuroscience where rich augmentations are not already established. When applied to neural datasets, `MYOW` outperforms other self-supervised approaches in all examples (in some cases by more than 10%), and surpasses the supervised baseline for most datasets. By learning to predict the latent representation of similar samples, we show that it is possible to learn good representations in new domains where augmentations are still limited.

1. Introduction

Self-supervised learning (SSL) methods have made impressive advances on a wide range of tasks in vision (Doersch et al., 2015; Perez & Wang, 2017; Oord et al., 2018; He et al., 2020; Chen et al., 2020; Grill et al., 2020; Cai et al., 2020; Song & Ermon, 2020), speech (Oord et al., 2018), network science (Veliveccković et al., 2018; Zhu et al., 2020), and reinforcement learning (RL) (Oord et al., 2018; Guo et al., 2020; Schwarzer et al., 2020). Most of these methods, at their core, aim to maximize the similarity between different augmented "views" of the same sample (positive examples) while avoiding collapse of the representation, typically by encouraging different samples (negative examples) to be dissimilar (Gutmann & Hyvärinen, 2010; Oord et al., 2018; He et al., 2020; Chen et al., 2020).

The choice of data augmentations is crucial to self-supervised learning; they control what the learned representations will be invariant to, effectively establishing strong implicit biases in the network (Chen et al., 2020; Tian et al., 2020). This makes it challenging to apply SSL, especially in new domains where effective augmentations are not already known (Purushwalkam & Gupta, 2020). We argue that when augmentations do not introduce sufficient variability into the generated views, the learned representations will be too *myopic* to model the relationships between different samples effectively (Grill et al., 2020; Wallace & Hariharan, 2020). To supplement augmentations, one may look within the dataset itself to find similar, yet distinct, views for self-supervised learning.

In this paper, we introduce **M**ine **Y**our **O**wn vie**W** (`MYOW`), a new approach for finding samples within the dataset that can serve as positive examples for one another. The idea behind our strategy is to actively *mine views*, finding samples that are close in the representation space of the network, and then predict, from one sample's latent representation, the representation of a nearby sample. Our approach builds on a recent predictive latent representation learning method called `BYOL` (Grill et al., 2020) that poses the SSL problem as prediction in the latent space across augmented views. To integrate across-sample prediction into this framework, we introduce a novel dual cascaded projector architecture that learns to predict across augmented views of the *same sample* in the first part of the network (like `BYOL`), and then to predict *across samples* through a separate network

[1]Georgia Tech [2]DeepMind, London, UK [3]Johns Hopkins University Applied Physics Laboratory [4]Washington University in St. Louis [5]DeepMind Paris [6]Emory University. Correspondence to: Mehdi Azabou <mazabou@gatech.edu>, Eva Dyer <evadyer@gatech.edu>.

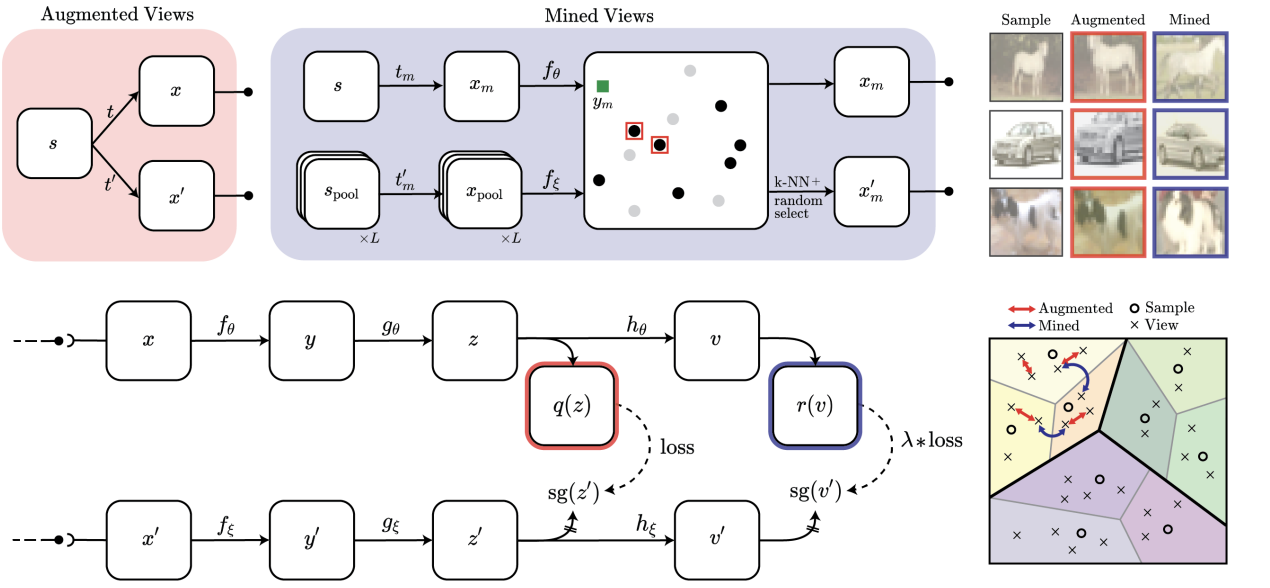


Figure 1. Overview of our approach for representation learning through across-sample prediction. The architecture of the system, shown in the bottom row, consists of two dual deep predictor networks, the online network (top) and the target network (below). There exists two sources of views, the augmented views block (top row, red) and the mined views block (top row, blue). Each type of view is handled by a dedicated predictor. During mining (top row, blue), we see the representation space with black dots representing the pool of candidates and gray dots representing unsampled candidates. Mined views are found by computing the k-nearest neighbors (red boxes) of the online representation y_m (green square) among the target representations of the pool of candidates (black dots). One of the nearest neighbors is randomly selected to be the mined view. On the bottom right, we illustrate the idea behind across-sample prediction and show how the two spaces emphasize different levels of similarity between data points.

that draws from the first projector's output (Figure 1). In empirical studies, we show that separating prediction across the two projectors allows the network to build different spaces where the data can be represented and the distinct prediction tasks can be solved effectively.

Overall, we make the following contributions:

- In Section 3, we introduce MYOW and in Section 4.1 apply it to standard benchmark image datasets where effective augmentations are well established. In these domains, we show that when provided with fewer classes of augmentations, MYOW is robust, even when other methods may be negatively impacted.

- MYOW integrates augmented and mined views into a unified space through the use of a dual projector. In Section 4.1, we show how our cascaded architecture allows the network to multi-task, or effectively predict different types of relationships in data (augmented vs. mined views).

- In Section 4.2, we provide a novel application of SSL to recordings of hundreds of neurons in mammalian brains, where devising good augmentations is especially challenging. Along with new classes of augmentations for these data, we show that SSL methods can provide representations that are more robust than supervised methods trained on the same task. By linking "close" yet temporally separated brain states, MYOW yields significant improvement (in some cases, over 10%) in the decoding of neural states when compared to other self-supervised approaches.

- In Section 4.3, we study a setting where self-supervision without across-sample prediction fails and show how MYOW can rescue the network. This example shows the promise of MYOW in settings where data is limited and effective augmentations are unknown.

2. Background

In many state-of-the-art approaches for SSL, views are generated by applying different transformations to an example (e.g., blurring, cross-channel, Zhang et al., 2017, and multi-view prediction, Tian et al., 2019, masking, or applying other linear and non-linear transformations). The goal is to learn a representation that maximizes the similarity between two transformed versions of the same sample, while ensuring that the representation does not collapse.

Whereas contrastive learning approaches like SimCLR (Tian et al., 2019) need both positive and negative examples to learn, more recent methods like BYOL (Grill et al., 2020) and PBL (Guo et al., 2020) have shown that it is possible to learn a good representations without needing negative

examples (Richemond et al., 2020). As we will use BYOL as the backbone of our approach, we will provide further detail on this approach.

Bootstrap Your Own Latent (BYOL). BYOL is built on mirrored online and target networks that learn from one another, casting the problem of representation learning as a prediction task in the latent space. In particular BYOL learns by predicting the embeddings of different augmented views of the same sample $\mathbf{s}$. BYOL uses a set of transformations $\mathcal{T}$ to generate two random views of $\mathbf{s}$, $\mathbf{x} = t(\mathbf{s})$ and $\mathbf{x}' = t'(\mathbf{s})$, where $t, t' \sim \mathcal{T}$. The views are fed through the online and target networks' encoders. Let $\mathbf{y} = f_\theta(\mathbf{x})$ and $\mathbf{y}' = f_\xi(\mathbf{x}')$ denote the representations formed in the online and target encoders, which are parameterized by weights θ and ξ, respectively. These representations are then projected to smaller embeddings $\mathbf{z}$ and $\mathbf{z}'$ through the projection networks g_θ and g_ξ. The prediction network q_θ tries to predict the target $\mathbf{z}'$ from $\mathbf{z}$ by minimizing:

$$\mathcal{L}_{byol}(\theta) = d(q_\theta(\mathbf{z}), \mathbf{z}'), \tag{1}$$

where d is the normalized ℓ_2-distance $d(\mathbf{u}, \mathbf{v}) = \|\bar{\mathbf{u}} - \bar{\mathbf{v}}\|_2$, with $\bar{\mathbf{u}} = \mathbf{u}/\|\mathbf{u}\|_2$ for all $\mathbf{u}$ and $\mathbf{v}$. In our implementation, the loss is symmetrized: we separately feed $\mathbf{x}'$ through the online network and $\mathbf{x}$ through the target network and compute the prediction error. We let $\widetilde{\mathcal{L}}_{byol}$ denote the symmetrized version of the loss.

Theoretically the objective function of BYOL may collapse to a constant vector when it is optimized in terms of both θ and ξ. Instead we optimize $\mathcal{L}$ only in terms of θ and maintain ξ as the weighted average of past values of θ which avoids this collapse (Richemond et al., 2020). The training dynamics are given by:

$$\theta \leftarrow \text{optimize}(\theta, \nabla\widetilde{\mathcal{L}}_{byol}, \eta), \;\; \xi \leftarrow \tau\xi + (1-\tau)\theta, \tag{2}$$

where $\tau \in (0, 1)$ is a smoothing or momentum parameter, and η is the learning rate used to optimize the weights of online network through an appropriate optimizer.

3. Mine Your Own vieW (MYOW)

In this section, we introduce MYOW. See Algorithm 1 for the pseudocode.

3.1. Approach

Mining views. The aim of MYOW is to explicitly incorporate prediction *between samples* into SSL. We adaptively select (or "mine") samples that are neighbors in the representation space. Specifically, given an anchor sample $\mathbf{s}$, we generate a view $\mathbf{x}_m$ using $t_m \sim \mathcal{T}_m$ and map it to $\mathbf{y}_m = f_\theta(\mathbf{x}_m)$ through the online encoder. We then choose a batch of L samples at random from our dataset as prospective mined views and apply independent transforms sampled from $\mathcal{T}_m$ to each. We project all the candidate views in the target encoder's space to obtain $\mathcal{S} = \{f_\xi(\mathbf{x}_j)\}_L$ and then randomly select one of $\mathbf{y}_m$'s k-nearest neighbors in $\mathcal{S}$. The selected sample $\mathbf{x}'_m$ will be considered our mined view.

Cascaded projection network. In general, the task of predicting a mined view from the original sample could be more complex than predicting an augmentation. To address this challenge, we add a dedicated predictor that operates on mined views from the dataset. Specifically, mined views $(\mathbf{x}_m, \mathbf{x}'_m)$ are forwarded through the network to generate $\mathbf{v}_m = h_\theta(g_\theta(f_\theta(\mathbf{x}_m)))$ and $\mathbf{v}'_m = h_\xi(g_\xi(f_\xi(\mathbf{x}'_m)))$, where h_θ and h_ξ are the second online and target projectors. The network then predicts $\mathbf{v}'_m$ from $\mathbf{v}_m$ by using a separate predictor r_θ.

Loss function. To combine augmented and mined views, MYOW minimizes:

$$\mathcal{L}_{myow} = \widetilde{\mathcal{L}}_{byol} + \lambda\, d(r_\theta(\mathbf{v}_m), \mathbf{v}'_m), \tag{3}$$

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.

Initialize: $\xi \leftarrow \theta$

repeat
  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 for
  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 for
  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 $\mathcal{N}_k(\mathbf{y}_{m,i})$
    $\mathbf{v}_i = h_\theta\left(g_\theta\left(\mathbf{y}_{m,i}\right)\right); \mathbf{v}'_i = h_\xi\left(g_\xi\left(\mathbf{y}'_{m,i}\right)\right)$
  end for
  $\mathcal{L} = \sum_i d(q_\theta(\mathbf{z}_i), \mathbf{z}'_i) + d(q_\theta(\mathbf{u}'_i), \mathbf{u}_i) + \lambda d(r_\theta(\mathbf{v}_i), \mathbf{v}'_i)$
  $\theta \leftarrow \text{optimizer}(\theta, \mathcal{L}/B, \eta)$
  $\xi \leftarrow \tau\xi + (1-\tau)\theta$
until end of training

where $\widetilde{\mathcal{L}}_{byol}$ is the symmetric loss presented in Section 2 and λ is a dynamic weight that regulates the contribution of the mined views in the objective throughout learning.

3.2. Implementation details

Scheduling mined views and initial warmup period. To ensure that we find views that are good targets for self supervision, we warm up the network using only augmentation ($\lambda = 0$) and then gradually introduce mined views. More specifically, we linearly increase λ from 0 to $\lambda_{\max}$ for a number of epochs and then keep it constant for the rest of the training.

Ensuring diversity within mined views. It is important to encourage diversity in the samples being selected during mining. The set of candidate views that are sampled represent a small subset of the entire dataset. These views are sampled without replacement, and the sampler is reinitialized once it has cycled through the entire dataset. This ensures that data points are given different targets throughout learning. In our implementation, the same pool of candidates is used for all samples in a batch at a given training iteration. The choice of both L and k determines the scale at which the mined views are bootstrapped, as L is decreased or k increased, the mining process becomes more stochastic and the second predictor has a larger number of samples it needs to predict across.

Ensuring diversity between augmented views and mined views. One aspect of our method is the use of two different distributions for augmented and mined views, over which we hierarchically form representations. If the distributions are essentially the same, then the two projectors will see the same data and won't have any reason to build different representations. Thus, it is important to make sure that the predictions formed in both spaces are different. In natural image datasets like CIFAR, we get this diversity naturally, as no image can be obtained by applying an augmentation to another image in the dataset. However, this is not true when working with time-varying data where we can use nearby points in time as augmented views; in this case, we restrict the candidate views used for mining to be separated by a minimum distance in time. This ensures that the mining introduces views that are otherwise not linked through the first projector.

Memory management and distributed training. While there is additional overhead due to mining, our method does not require a large memory bank. The pool of candidates is resampled on-the-fly at each iteration step. 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 candidates of larger size. This means that our method comes with an extra computational overhead but minimal extra memory requirements. When mining, we forward the samples through the online and target networks; once the samples are selected, we resume from the encoder spaces and do not recompute the representation.

Table 1. Accuracy (in %) for classification on MNIST, CIFAR-10, and CIFAR-100.

Method	$\mathcal{T}$	MNIST	CIFAR-10	CIFAR-100
SimCLR	Crop	98.47	62.09	33.26
BYOL	Crop	97.27	78.90	43.63
MYOW	Crop	**99.20**	**80.87**	**43.92**
SimCLR	All	98.61	86.01	57.02
BYOL	All	98.03	83.87	57.15
MYOW	All	**99.33**	**86.10**	**57.73**

Code availability. The implementation of MYOW is made available at https://nerdslab.github.io/myow/.

4. Evaluations

4.1. Experiments on image datasets

Experimental setup. To train our model and other SSL approaches on natural images, we follow the procedures reported in previous work (Chen et al., 2020; Chen & He, 2020; Huang et al., 2021), including the choice of augmentations. For CIFAR (Krizhevsky, 2012), we use random cropping, color jittering, random horizontal flip, and random grayscale conversion. For our backbone, we use the CIFAR variant of ResNet-18 (He et al., 2015). When training MYOW, we warmup the network for 100 epochs before ramping up λ to 1.0, and when mining we use $L = 512$ and $k = 1$. To train the online network, we use an SGD optimizer with a learning rate of 2.0, for the target network, we use a momentum of 0.99. For all experiments, we train the networks with a batch size of 512, for 800 epochs using 2 GTX 2080Ti GPUs. To evaluate the quality of the representations after training the model, we freeze the weights of the encoder, train a supervised linear layer over it and report the accuracy on the validation set. For full details on training and evaluation procedures, including our setup for MNIST, see Appendix A.

Results on natural images. We tested MYOW, BYOL, and SimCLR on MNIST, CIFAR-10, and CIFAR-100 in two settings, one where only cropping is used and one where all available augmentations are used (Table 1). On both CIFAR-10 and CIFAR-100, when we consider the strong augmentation setting (*All*), we find that all three methods perform similarly, with MYOW outperforming BYOL even though they share the same random seed and the same hyperparameters. When considering only cropping (weaker augmentation), we find that both BYOL and MYOW improve over SimCLR by a significant margin (almost 10%) with MYOW achieving the

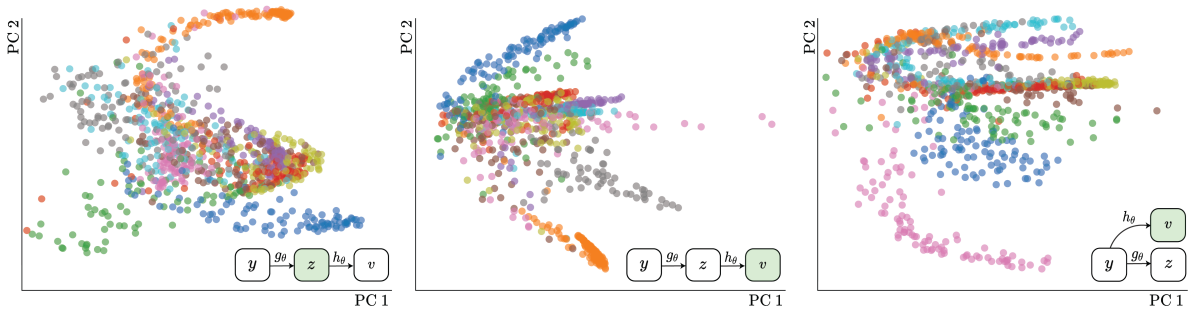


Figure 2. Visualization of the representations of MNIST learned by different architectures. Here, we project digits into the space of the first projector (left) and second projector (middle) for our cascaded architecture. On the right, we show embeddings of data in the second projector when using a parallel configuration. When cascaded, the second projector appears to build a more clustered space where digits are grouped.

Table 2. Comparing different projector architectures for incorporating mined views. MNIST classification accuracy (in %) with MYOW for different architectures.

Arch	Dimension	Acc
Cascaded	16	**99.20**
Cascaded	128	98.09
Parallel	16	96.33
Parallel	128	97.75
Single	16	97.13
Single	128	98.75

highest accuracies. On MNIST, MYOW performs very well in both settings. Overall, our results on natural image datasets demonstrate that MYOW is robust to the choice of augmentations and is competitive with state-of-the-art methods for self-supervised learning.

Architecture ablations. Next, we performed ablations to examine how the inclusion of a second projector as well as the dual projector layout impacts learning. In Table 2, we report results on the MNIST dataset in the weak augmentation setting (*Crop*), and refer the reader to Appendix C.1 for further experiments in the strong augmentation setting (*All*). While in some cases it can be possible to integrate augmented and mined views into a single projector, we generally find that the network is less stable, with the performance typically stabilizing to the baseline level that we also obtain with BYOL. If we use parallel projectors, we also observe similar performance. In terms of projector dimensions, we find that restricting the second projector to a lower dimension is also important.

To gain further insights, we visualize the latent spaces generated through these different architectures (Figure 2). In the space of the first projector, we observe that digits are organized by class but the class manifolds are still highly entangled when projected into lower dimensions. In the space of the second projector, we find that digits are well clustered along rays (due to the use of a normalized ℓ_2-distance) when projected into a lower dimension. We hypothesize that because the first projector is filtering out the variance due to augmentations (Xiao et al., 2020; Misra & Maaten, 2020), the second projector can focus on higher-level information and start grouping similar digits in this space.

4.2. Decoding brain states from neural recordings

Next, we wanted to see whether our approach could be applied to learn representations from spiking neural activity, a domain where rich augmentation classes have not yet been established.

Neural datasets. We considered two different neural decoding tasks. The first application of the method is to datasets acquired from the primary motor cortex (M1) of two non-human primates (NHPs) while they made reaching movements towards one of eight targets (Dyer et al., 2017). In these examples, the goal is to decode the intended reach direction and map it to one of the 8 possible targets. The activity of a population of roughly a hundred neurons was binned into 100ms intervals to generate around 1.3k data points per dataset. The second application is to 12-hour neural recordings acquired from the visual cortex of a freely behaving rat and the hippocampus of a freely behaving mouse. In these examples, the goal is to decode the arousal state of the rodent into one of three classes: rapid eye movement (REM) sleep, non-REM sleep, wake (Hengen et al., 2016; Ma et al., 2019). Here, neural activity was binned into 4s intervals to produce firing rates for 42 and 120 neurons, respectively. More details on the datasets, network architecture, and training procedures used for these experiments can be found in Appendix B.

Augmentations for spiking neural data. To apply our approach and other self-supervised baselines to neural data, we needed to first identify candidate augmentations of the data that could be used to seed learning. A first choice, given the temporal nature of the data, is to select samples

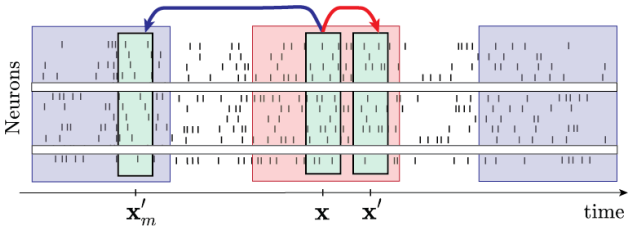


Figure 3. Visualization of augmentations used for neural activity. Within a small local window around each anchor sample, we consider the nearby samples (red) to be potential positive examples. Outside of a safe zone, we can label more distant samples (blue) as either negative examples (in contrastive 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.

Table 3. How augmentations impact our ability to decode brain states. Here, we compute the accuracy in our reach direction prediction task when we apply a given set of transformations.

	TS	RDrop	Noise	Pepper	Acc
BYOL	✓				41.75
		✓	✓	✓	55.70
	✓	✓			61.39
	✓	✓	✓	✓	63.80
MYOW	✓				46.61
		✓	✓	✓	53.15
	✓	✓			67.97
	✓	✓	✓	✓	70.41

that are close in time and use them as positive examples for one another (Temporal shift, *TS*) (Banville et al., 2020) (Figure 3). However, we found that temporal augmentations are, on their own, insufficient to drive learning and the representations collapse (Table 3, Figure 4). When we applied both temporal shift as well as randomized dropout (*RDrop*) that masks a random number of the neurons, we see a substantial increase in decoding accuracy, a trend observed throughout our tested datasets (see Appendix C.2).

In addition to these two main classes of transformations, we also tested (i) the sparse increase in activation of a random set of neurons (*Pepper*), and (ii) Gaussian Noise (*Noise*) added to the firing rates before dropout. As we include more augmentations (*Noise* + *Pepper*) the performance of both models increases further, but by smaller margins than the dropout augmentation. Again, we find that MYOW is more robust to the particular transformations used, and often outperforms BYOL by a significant margin (5-7%). This experiment speaks to the importance of finding good augmentations and shows how MYOW can harness across-sample prediction to learn, especially when augmented views alone may be insufficient to drive learning.

MYOW outperforms both supervised methods and other SSL methods on diverse neural datasets. After finding classes of augmentations to drive SSL, we next compared our approach with other self-supervised methods (SimCLR, BYOL) that are given access to the same positive examples to predict from. In addition, we trained an autoencoder (AE) and a Multi-layer Perceptron (MLP) classifier (Supervised) with weight regularization and dropout.

A general trend across these neural datasets is that the supervised methods readily overfit to the training data. When tested on validation data, MYOW beats supervised approaches by a significant margin, only lagging behind on one dataset. MYOW also outperforms the other self-supervised methods, especially on Chewie - Day 1 and Mihi - Day 1, where the margins are over 7%. We hypothesize that this is due to the domain shift that occurs across trials; by explicitly linking samples separated by time, the network is incentivized to build a representation that is invariant to the shifts that may occur in neural activity over time.

When we consider a slightly relaxed accuracy metric that we call the δ-Acc (akin to Top-k, see Appendix B for a formal definition), our method consistently scores above 80% δ-Acc on all datasets, outperforming the supervised baseline by over 10% on Mihi - Day 2. This result suggests that MYOW organizes representations in a way that is more reflective of the global task structure, placing similar reach directions close to one another. We thus show that by integrating diverse views (i.e., mined views as well as augmented views) into our prediction task, we can more accurately decode movement variables.

MYOW applied to the rodent brain during free behavior. In our final experiment on neural data, we applied MYOW to datasets from rodent cortex (V1) and hippocampus (CA1) (Table 4) during free behavior. These datasets provide us with a new setting under which we test our approach as they are from new brain regions and from a different species. Overall, the trends are similar to that of our earlier

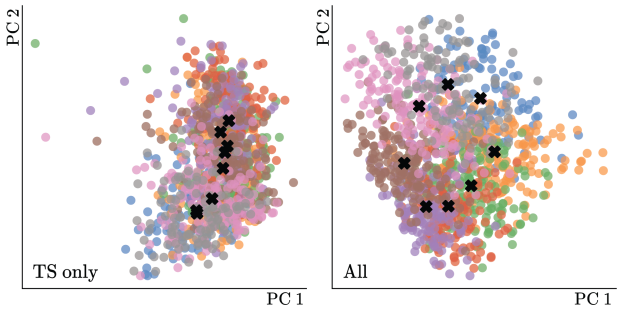


Figure 4. Representations obtained for neural data for (left) *TS* only and (right) *All* available transformations, with the centroids of all 8 classes highlighted. On the right, the reach directions are more separable and we find that the centroids are organized according to the circular structure of the movement task.

Table 4. Accuracy (in %) in the prediction of brain states from spiking neural activity. In the first four datasets labeled Reach, the task is to decode the reach direction from neurons in the primary cortex (M1) of two non-human primates (Chewie, Mihi) to predict which one of eight target positions the individual selects. In the next two datasets labeled Sleep, the task is to decode the arousal state (REM, nREM, Wake) from neurons in the primary visual cortex (Rat-V1) and from the hippocampus (Mouse-CA1).

	Reach								*Sleep*	
	Chewie-Day 1		Chewie-Day 2		Mihi-Day 1		Mihi-Day 2		Rat-V1	Mouse-CA1
	Acc	δ-Acc	Acc	δ-Acc	Acc	δ-Acc	Acc	δ-Acc	F1-score	F1-score
Supervised	63.29	77.22	**72.29**	**81.51**	63.64	79.02	61.49	68.44	86.34	93.01
Autoencoder	48.40	67.51	46.79	65.84	50.94	68.03	55.19	74.98	34.17	57.73
SimCLR	59.02	78.65	50.39	64.75	59.55	77.52	54.47	71.65	80.28	80.23
BYOL	63.80	81.90	57.17	77.36	59.50	79.78	60.82	78.30	85.42	93.24
MYOW	**70.41**	**86.24**	60.95	81.36	**70.48**	**83.24**	**64.35**	**80.58**	**88.01**	**93.70**

experiments, with MYOW providing robust performance, often exceeding that of the supervised baseline. We note that in these experiments, the AE does very poorly which we believe is due to the fact that the classes in the example are not well balanced, in contrast to our earlier experiments.

4.3. Incorporating across-sample prediction with MYOW can provide a way to avoid collapse

Based upon our experiments on neural data, we conjectured that the diversity introduced by MYOW makes it possible to learn effectively, even when the augmentations provided to the network are insufficient to drive learning in BYOL. We thus designed an experiment using the dSprites dataset (Matthey et al., 2017), as it allows control over the generation of data over multiple latent positions. For both the scale and orientation parameters, we remove half of the possible latent positions in the training dataset (keeping 3/6 possible scales and 20/40 possible orientations), and then subsample the remaining latent variables further, keeping either 30%, 15%, or 7.5% of the remaining samples to train on. We also restrict the class of transformations to a single weak augmentation (random crop of $80-100\%$). More details on the experimental setup can be found in Appendix A.

When we train BYOL and MYOW on a sufficiently dense sampling of the latent positions (30%), we observe that both models can classify on unseen latent positions with nearly 100% accuracy (Table 5, Figure 5). However, when we consider the undersampled condition (7.5%), BYOL fails to generalize to the unseen positions, resulting in a low accuracy of around 60%. In contrast, MYOW maintains a high accuracy of 94% despite the limited training data. These findings suggest that in settings where the data manifold is sparsely sampled, MYOW provides a way to build predictions across different but similar data samples and this can help to rescue the method.

We wanted to examine this further and created a visualization of the links formed across augmentations and mined

Table 5. Accuracy (in %) on dSprites when latent positions in the training dataset are sparse. The shape classification accuracy obtained on the dSprites validation set when the model is only trained on a fraction of possible latent positions.

Method	Train split (%)	Acc
BYOL	30	99.83
	15	94.56
	7.5	63.12
MYOW	30	99.76
	15	99.42
	7.5	94.39

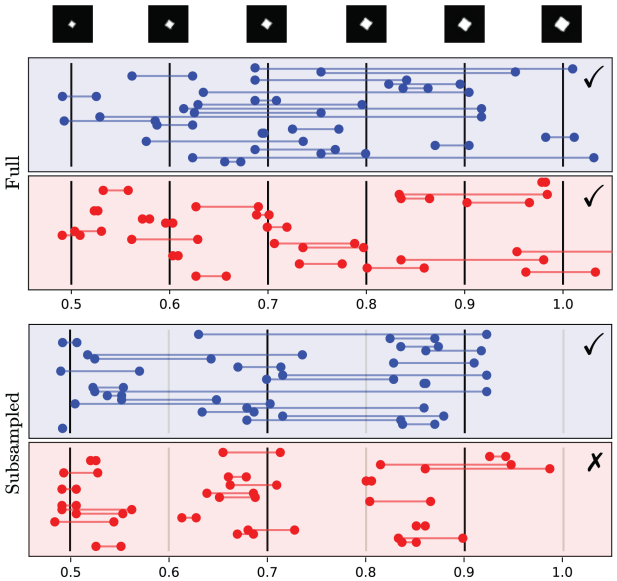


Figure 5. Understanding predictive learning when augmentations aren't sufficiently rich. Each segment represents a pair of views (red for augmented, blue for mined) of the corresponding latent scale (x-axis). The vertical lines represent the original scales of samples pre-augmentation. We examine the case where we have access to the full dataset (top) and when we have only half of the latent positions (3/6) and 7.5% of the remaining samples (bottom).

views (Figure 5). In this visualization, we show how gaps in the manifold of latent positions (Subsampled) cannot be filled through augmentations alone. Below, we show how MYOW builds predictions across different samples (blue lines) to fill in the gaps and thus enable learning in this highly undersampled regime.

5. Related Work

Augmentations and their impact on self-supervised learning. The choice of transformations dictates what invariances will be built into the representation (Misra & Maaten, 2020), and thus introduces inductive bias in the model. Recent triumphs on object-centric datasets like Imagenet have relied on "aggressive" augmentation strategies to achieve success (Purushwalkam & Gupta, 2020; Xiao et al., 2020). Performance is closely tied to the selection of transformations and the specific parameter setting (Dosovitskiy et al., 2015). As a result, when working in other image domains, such as satellite or biological imagery, transformations can hinder performance (Zhao et al., 2019; Wallace & Hariharan, 2020). A number of methods have been developed to perform well on downstream tasks without handpicking augmentations, whether by *learning* which invariances improve performance (Tian et al., 2020), or by constructing separate subspaces for different types of augmentations (Xiao et al., 2020; Lee et al., 2020). This last approach bears some architectural similarities to ours as they use multiple projectors to handle different transformation classes.

Unlike these approaches, which often start from known classes of augmentations and tune their parameters (Roth et al., 2015) or evaluate different augmentations for specific domains (Dosovitskiy et al., 2015), our approach looks to the dataset itself to provide views. This is somewhat similar to column selection methods used in large-scale kernel approximation, which aim to select samples from the dataset to express other samples rather than learning a basis to factorize the dataset (Mahoney & Drineas, 2009; Elhamifar & Vidal, 2013). By using other samples directly, we hope to avoid distorting the representations through undesirable invariances, as well as minimizing the "feature engineering" involved in handpicking augmentations.

Label propagation and learning by association. Multiple approaches use the idea of a prototype label (or "pseudo-label") for data in a network's latent space. When provided with a partially-labelled dataset, label propagation can extend sparse existing labels to samples which have similar representations, effectively "learning by association" (Haeusser et al., 2017). More recently, the Meta Pseudo Labels method was introduced (Pham et al., 2020), which trains dual teacher and student networks simultaneously; the student's performance on a test set is used as feedback to adapt the teacher's labelling strategy. Deep clustering methods do away with labels entirely, instead assigning samples to clusters based on their representations, and training the network to predict cluster assignments (Caron et al., 2018). More recently, (Caron et al., 2020) combined clustering with contrastive learning, by ensuring that cluster assignments group together augmentations of each sample.

Like many of these methods, we select samples with similar embeddings which can be compared as positive examples. Thus, we can think of this approach as adaptively finding pseudo-labels for data points. However, we predict features of similar, automatically-selected examples rather than just a class assignment, and moreover use more complex predictors for mined examples, which is meant to encourage richer representations with both high and low level features.

Self-supervised learning from sequences. Previous work in contrastive learning for sequential data often leverages a slowness assumption to use nearby samples as positive examples and farther samples as negative examples (Oord et al., 2018; Sermanet et al., 2018; Dwibedi et al., 2019; Le-Khac et al., 2020; Banville et al., 2020). Contrastive predictive coding (CPC) (Oord et al., 2018) builds upon the idea of temporal contrastive learning by building an AR-model that predicts future points given previous observed timesteps. PBL (Guo et al., 2020) uses a similar approach, however, they show similarly to BYOL that negative examples are not needed to learn a good representation in multi-step prediction for RL.

Our approach shares similarity with many existing approaches for temporal contrastive learning in how we build temporal structure into our augmentations. However, in our formulation, we ask the network to create predictions across samples that are nonlocal through the use of mined views. Another novel contribution of our work is the introduction of dropout for generating augmentations in the spiking neural datasets which are considered here.

6. Conclusion

This paper introduces a new method for SSL that mines the dataset to find positive examples and uses them for across-sample prediction. We show that our approach can be used to learn meaningful representations for both image and brain activity datasets, and demonstrate the promise of this method in settings where augmentations alone may be insufficient to drive learning.

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 (Rao &

Ballard, 1999). 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.

7. Acknowledgements

ELD, MA, CHL, KBH, and KBN were supported by NIH-1R01EB029852. KBH and KBN were supported by NIH-1R01NS118442. This work was also supported by an award from the McKnight Foundation and Sloan Foundation. We would like to thank Bilal Piot for helpful suggestions on the manuscript.

References

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

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

Buccino, A. P., Hurwitz, C. L., Magland, J., Garcia, S., Siegle, J. H., Hurwitz, R., and Hennig, M. H. Spikeinterface, a unified framework for spike sorting. *bioRxiv*, 2019. doi: 10.1101/796599. URL https://www.biorxiv.org/content/early/2019/10/07/796599.

Cai, Q., Wang, Y., Pan, Y., Yao, T., and Mei, T. Joint contrastive learning with infinite possibilities. *arXiv preprint arXiv:2009.14776*, 2020.

Caron, M., Bojanowski, P., Joulin, A., and Douze, M. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 132–149, 2018.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

Chen, X. and He, K. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020.

Chung, J. E., Magland, J. F., Barnett, A. H., Tolosa, V. M., Tooker, A. C., Lee, K. Y., Shah, K. G., Felix, S. H., Frank, L. M., and Greengard, L. F. A fully automated approach to spike sorting. *Neuron*, 95(6):1381 – 1394.e6, 2017. ISSN 0896-6273. doi: https://doi.org/10.1016/j.neuron.2017.08.030. URL http://www.sciencedirect.com/science/article/pii/S0896627317307456.

Doersch, C., Gupta, A., and Efros, A. A. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1422–1430, 2015.

Dosovitskiy, A., Fischer, P., Springenberg, J. T., Riedmiller, M., and Brox, T. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 38(9):1734–1747, 2015.

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

Dyer, E. L., Azar, M. G., Perich, M. G., Fernandes, H. L., Naufel, S., Miller, L. E., and Körding, K. P. A cryptography-based approach for movement decoding. *Nature Biomedical Engineering*, 1(12):967–976, 2017.

Elhamifar, E. and Vidal, R. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11): 2765–2781, 2013.

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

Guo, D., Pires, B. A., Piot, B., Grill, J.-b., Altché, F., Munos, R., and Azar, M. G. Bootstrap latent-predictive representations for multitask reinforcement learning. *arXiv preprint arXiv:2004.14646*, 2020.

Gutmann, M. and Hyvärinen, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 297–304. JMLR Workshop and Conference Proceedings, 2010.

Haeusser, P., Mordvintsev, A., and Cremers, D. Learning by association–a versatile semi-supervised training method for neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 89–98, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation

learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Hengen, K., Torrado Pacheco, A., McGregor, J., Van Hooser, S., and Turrigiano, G. Neuronal firing rate homeostasis is inhibited by sleep and promoted by wake. *Cell*, 165(1):180 – 191, 2016. ISSN 0092-8674. doi: https://doi.org/10.1016/j.cell.2016.01.046. URL http://www.sciencedirect.com/science/article/pii/S0092867416300605.

Huang, L., Zhang, C., and Zhang, H. Self-adaptive training: Bridging the supervised and self-supervised learning. *arXiv preprint arXiv:2101.08732*, 2021.

Krizhevsky, A. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

Le-Khac, P. H., Healy, G., and Smeaton, A. F. Contrastive representation learning: A framework and review. *IEEE Access*, 8:193907–193934, 2020. ISSN 2169-3536. doi: 10.1109/access.2020.3031549. URL http://dx.doi.org/10.1109/ACCESS.2020.3031549.

Lee, H., Hwang, S. J., and Shin, J. Self-supervised label augmentation via input transformations. In *International Conference on Machine Learning*, pp. 5714–5724. PMLR, 2020.

Ma, Z., Turrigiano, G. G., Wessel, R., and Hengen, K. B. Cortical circuit dynamics are homeostatically tuned to criticality in vivo. *Neuron*, 2019. ISSN 0896-6273. doi: https://doi.org/10.1016/j.neuron.2019.08.031. URL http://www.sciencedirect.com/science/article/pii/S0896627319307378.

Mahoney, M. W. and Drineas, P. Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, 2009.

Mathis, A., Mamidanna, P., Cury, K. M., Abe, T., Murthy, V. N., Mathis, M. W., and Bethge, M. Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*, 21:1281–1289, 2018. URL https://www.nature.com/articles/s41593-018-0209-y.

Matthey, L., Higgins, I., Hassabis, D., and Lerchner, A. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

Misra, I. and Maaten, L. v. d. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

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

Perez, L. and Wang, J. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

Pham, H., Xie, Q., Dai, Z., and Le, Q. V. Meta pseudo labels. *arXiv preprint arXiv:2003.10580*, 2020.

Purushwalkam, S. and Gupta, A. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases, 2020.

Rao, R. P. and Ballard, D. H. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1):79–87, 1999.

Richemond, P. H., Grill, J.-B., Altché, F., Tallec, C., Strub, F., Brock, A., Smith, S., De, S., Pascanu, R., Piot, B., et al. Byol works even without batch statistics. *arXiv preprint arXiv:2010.10241*, 2020.

Roth, H. R., Lee, C. T., Shin, H.-C., Seff, A., Kim, L., Yao, J., Lu, L., and Summers, R. M. Anatomy-specific classification of medical images using deep convolutional nets. In *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*, pp. 101–104. IEEE, 2015.

Schwarzer, M., Anand, A., Goel, R., Hjelm, R. D., Courville, A., and Bachman, P. Data-efficient reinforcement learning with momentum predictive representations. *arXiv preprint arXiv:2007.05929*, 2020.

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

Sheridan, R. P. Time-split cross-validation as a method for estimating the goodness of prospective prediction. *Journal of Chemical Information and Modeling*, 53(4):783–790, 2013.

Song, J. and Ermon, S. Multi-label contrastive predictive coding. *arXiv preprint arXiv:2007.09852*, 2020.

Tian, Y., Krishnan, D., and Isola, P. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. What makes for good views for contrastive learning. *arXiv preprint arXiv:2005.10243*, 2020.

Veliveccković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

Wallace, B. and Hariharan, B. Extending and analyzing self-supervised learning across domains. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M. (eds.), *Computer Vision – ECCV 2020*, pp. 717–734, Cham, 2020. Springer International Publishing.

Xiao, T., Wang, X., Efros, A. A., and Darrell, T. What should not be contrastive in contrastive learning. *arXiv preprint arXiv:2008.05659*, 2020.

Zhang, R., Isola, P., and Efros, A. A. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1058–1067, 2017.

Zhao, A., Balakrishnan, G., Durand, F., Guttag, J. V., and Dalca, A. V. Data augmentation using learned transformations for one-shot medical image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8543–8553, 2019.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020.

Appendix

A. Experimental details: Image datasets

A.1. CIFAR-10 & CIFAR-100

For both datasets, we use the same architecture and hyperparameters, unless mentioned otherwise.

Architecture: We use the CIFAR variant of ResNet-18 as our backbone: Similar to (**?**), we change the parameters of the first convolutional layer to have a kernel size of 3x3 and a stride of 1, we also remove the first max pooling layer. The representation $\mathbf{y}$ corresponds to the output of the final average pool layer, which has a feature dimension of 512. Like in (Grill et al., 2020), we use multi-layer perceptrons (MLPs) of depth 2 for projectors and predictors. Let MLP(i, h, o) consist of a linear layer with input size i and output size h, followed by batch normalization, rectified linear units (ReLU) and a linear layer of output size o. We use MLP$(512, 4096, 256)$ for the first projector g_θ (the only one in the case of `BYOL`) and MLP$(256, 4096, 256)$ for its corresponding predictor q_θ. And use MLP$(256, 1024, 64)$ for the second projector h_θ and MLP$(64, 1024, 64)$ for its corresponding predictor r_θ. When training on CIFAR-100, the dimension of the hidden layers for the first and second projectors (and predictors) are respectively 1024 and 128.

Class of transformations: During training, we generate augmented views using the following transformations ($\mathcal{T}$) (Chen & He, 2020; Huang et al., 2021):

- Random cropping: a random crop of an area uniformly sampled between 0.2 and 1.0 of the area of the original image, and a random aspect ratio logarithmically sampled between $3/4$ and $4/3$ of the original aspect ratio are made. This crop is then resized to a 32x32 image using bicubic interpolation.
- Random horizontal flip: the image is fliped left to right with a probability of 0.5.
- Color jittering: the brightness, contrast, saturation and hue of the image are randomly changed with strengths of $(0.4, 0.4, 0.4, 0.1)$. This augmentation has a probability of 0.8 of being applied.
- Color dropping: the image is converted to gray scale with a probability of 0.2.

When mining, we use a different class of transformation $\mathcal{T}'$ which includes:

- Random cropping: same as in $\mathcal{T}$, except the area is uniformly sampled between 0.8 and 1.0 of the area of the original image to make the change in scale less severe.
- Random horizontal flip: same as in $\mathcal{T}$.

After augmentations, the images are normalized according to the average value and the standard deviation of the original images in the training set.

Training `BYOL` and `MYOW`: We use the SGD optimizer with a learning rate of 0.16 and a momentum of 0.9. The learning rate is decayed using a cosine decay scheduler over 800 epochs, with a warm-up period of 30 epochs. We also use a weight decay parameter $5 * 10^{-5}$ on all parameters except the biases and batch normalization parameters (Grill et al., 2020). The exponential moving average parameter τ is also decayed from 0.99 to $1.$ over the 800 epochs using a cosine decay scheduler. With `MYOW`, we use a pool batch size of $L = 512$, and $k = 1$. We use a mining weight of $\lambda = 1.$ ramped-up starting at epoch 100 until epoch 110, after which it is kept constant.

Training `SimCLR`: We use the same architecture and the same set of augmentation ($\mathcal{T}$) to train `SimCLR`. Following (Chen & He, 2020), we use a projector with a hidden dimension of 256. The model is trained using the SGD optimizer with a learning rate of 0.16, a momentum of 0.9, and a weight decay of $5 * 10^{-4}$. A cosine decay scheduler is also used.

Evaluation Protocol: Following the evaluation procedures described in (**?**Richemond et al., 2020), we train a linear classifier on top of the frozen representation of the encoder network and report the accuracy on the test sets. No augmentations are used during training, the images are only normalized. We use the SGD optimizer with a learning rate of 0.04 and a momentum of 0.9. The linear layer is trained, using a batch size of 512, over 100 epochs (120 epochs for CIFAR-100). The learning rate is decayed at epochs 60 and 80 by a factor of 0.1. We use the public train/test split for both CIFAR datasets.

A.2. MNIST

Architecture: We use a convolutional residual network with 13 layers (Adapted from the ResNet18 network) as our

backbone. The representation $\mathbf{y}$ corresponds to the output of the final average pool layer, which has a feature dimension of 128. We use MLP$(128, 512, 64)$ for the first projector g_θ (the only one in the case of BYOL) and MLP$(64, 512, 64)$ for its corresponding predictor q_θ. And use MLP$(64, 512, 16)$ for the second projector h_θ and MLP$(16, 512, 16)$ for its corresponding predictor r_θ.

Class of transformations: During training, we generate augmented views using the following transformations ($\mathcal{T}$):

- Random cropping: a random crop of an area uniformly sampled between 0.4 and 1.0 of the area of the original image, and a random aspect ratio logarithmically sampled between $3/4$ and $4/3$ of the original aspect ratio are made. This crop is then resized to a 16x16 image using bicubic interpolation.
- Random rotation: the image is rotated using an angle uniformly sampled between -20 and 20 degrees. The area outside the transform in the output image is filled with 0.
- Gaussian blurring: a Gaussian kernel of size 3x3 is used, along a standard deviation of 1.5. This transformation is applied with a probability of 0.1.

When mining, we use the following transformations:

- Random cropping: same as in $\mathcal{T}$, except the area is uniformly sampled between 0.8 and 1.0 of the area of the original image.

Training BYOL and MYOW: We use the LARS optimizer with a learning rate of 0.4 and a momentum of 0.9. The learning rate is decayed using a cosine decay scheduler over 200 epochs, with a warm-up period of 1 epoch. We also use a weight decay parameter $5 * 10^{-4}$ on all parameters except the biases and batch normalization parameters (Grill et al., 2020). The exponential moving average parameter τ is also decayed from 0.996 to $1.$ over the 200 epochs using a cosine decay scheduler. With MYOW, we use a pool batch size of $L = 256$, and $k = 2$. We use a mining weight of $\lambda = 10.$ ramped-up starting at epoch 2 until epoch 10, after which it is kept constant.

Training SimCLR: We use the same architecture and the same set of augmentation ($\mathcal{T}$) to train SimCLR. We use a projector with a hidden dimension of 64. The model is trained using the Adam optimizer with a learning rate of $1 * 10^{-3}$ and a weight decay of $1 * 10^{-6}$.

Evaluation Protocol: For the linear evaluation, we used the Adam optimizer with learning rate 0.001 and weight decay $1 * 10^{-5}$ to train the linear classifier for 10 epochs. We use no augmentations. We report the accuracy on the validation set. We use the public train/val split for the MNIST dataset.

A.3. dSprites

In our final experiment on the The dSprites dataset (Matthey et al., 2017), we use the same architecture and training protocol from the MNIST experiment. When training the self-supervised methods, we only use a single transformation, random cropping. The image is padded (with a padding of 4 pixels) and then randomly cropped using an area uniformly sampled between 0.7 and 1.1 of the area of the original image. This same augmentation is used during view mining. During evaluation, we train the network to classify the shape in the image (square, ellipse or heart).

The dSprites dataset is comprised of a total of 737,280 images. Each image has an associated shape, orientation, scale and 2D position. Each one of these latent variables has a finite number of possible values because of the procedural nature of the dataset. To generate the downsampled training sets used in our experiment, we uniformly sample 50% of the orientation latent values as well as 50% of the scale latent values, and only consider the corresponding images, thus effectively creating holes in the latent manifold. The dataset is further downsampled at a given rate r to generate the train set, the remaining images form the test set. The size of the train set is effectively $0.25 * r$ that of the entire dataset. In our experiment, we generate training sets that are 30%, 15% and 7.5% the size of the dataset.

B. Experimental details: Neural data

B.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 an $80/20$ ratio, to obtain our training and validation sets. The temporal splits gives us a better estimate of the prospective prediction compared to a random split (Sheridan, 2013). The activity of individual neurons was binned (100 ms intervals) to produce firing rates for roughly 150 neurons across two days.

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:

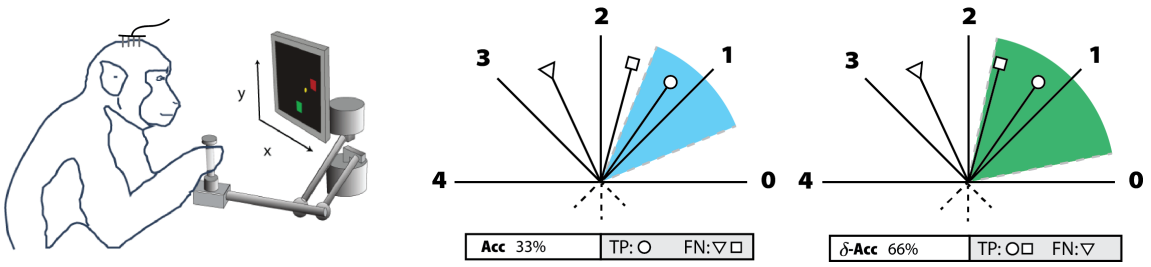
$$l_{\text{predicted}} = \lfloor \frac{4}{\pi}(\text{atan2}(y, x) \bmod 2\pi) \rceil \tag{4}$$

Network Architecture: For our encoder, we use a multi-layer perceptron (MLP) which is 5 blocks deep. Each block consists of a linear layer with output size 128 followed by batch normalization and rectified linear units (ReLU). We use MLP$(128, 256, 32)$ (See definition in Appendix A) for the first projector g_θ (the only one in the case of `BYOL`) and MLP$(32, 256, 32)$ for its corresponding predictor q_θ. And use MLP$(32, 64, 16)$ for the second projector h_θ and MLP$(16, 64, 16)$ for its corresponding predictor r_θ.

Training `BYOL` and `MYOW`: We use the LARS optimizer with a learning rate of 0.8 and a momentum of 0.9. The learning rate is decayed using a cosine decay scheduler over 2000 epochs, with a warm-up period of 10 epoch. We also use a weight decay parameter $1 * 10^{-6}$ on all parameters except the biases and batch normalization parameters. The exponential moving average parameter τ is also decayed from 0.9 to 1. over the 2000 epochs using a cosine decay scheduler. With `MYOW`, we use a pool batch size of $L = 512$, and $k = 3$. We use a mining weight of $\lambda = 0.1$ ramped-up starting at epoch 50 until epoch 60, after which it is kept constant.

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. When training the linear layer, we use the same augmentation used in self-supervised training. We use the Adam optimizer with a learning rate of 0.01 and a weight decay of $1 * 10^{-6}$. The linear layer is trained, using a batch size of 256, over 100 epochs.

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 S1-b).



(a) (b)

Figure S1. 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.

B.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 (Chung et al., 2017; Buccino et al., 2019). 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 (Mathis et al., 2018) 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.

We split the 12 hour block of data temporally using an $80/20$ ratio, to obtain our training 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.

Network Architecture: For our encoder, we use a multi-layer perceptron (MLP) which is 4 blocks deep. Each block consists of a linear layer followed by batch normalization and rectified linear units (ReLU). The output sizes of each block are respectively $32, 32, 16, 16$. We use MLP$(16, 48, 16)$ for the first projector g_θ (the only one in the case of `BYOL`) and MLP$(16, 48, 16)$ for its corresponding predictor q_θ. And use MLP$(16, 48, 16)$ for the second projector h_θ and MLP$(16, 48, 16)$ for its corresponding predictor r_θ.

Training `BYOL` and `MYOW`: We use the LARS optimizer with a learning rate of 1.0 and a momentum of 0.9. The learning rate is decayed using a cosine decay scheduler over 100 epochs, with a warm-up period of 3 epochs. We also use a weight decay parameter $1 * 10^{-6}$ on all parameters except the biases and batch normalization parameters. The exponential moving average parameter τ is also decayed from 0.98 to $1.$ over the 100 epochs using a cosine decay scheduler. With `MYOW`, we use a pool batch size of $L = 64$, and $k = 4$. We use a mining weight of $\lambda = 0.6$ ramped-up starting at epoch 5 until epoch 15, after which it is kept constant.

Evaluation Procedure: We freeze the weights of the online encoder and train a linear layer over the representation space. We use the Adam optimizer with a learning rate of 0.01 and weight decay of 10^-6. The linear layer is trained for 10 epochs over the training set and the same class of transformations used to train the self-supervised method are used as augmentations.

B.3. Augmentations for neural data

Temporal shift: As in previous work in temporal contrastive learning (Oord et al., 2018; Sermanet et al., 2018; Dwibedi et al., 2019; Le-Khac et al., 2020; Banville et al., 2020), we can use nearby samples as positive examples for one another.

We thus use temporal shift (TS) as a transformation: given sample $\mathbf{s}$, a firing rate vector recorded at time step t, $t(\mathbf{s})$ is the firing rate at time step $t + i * \delta$ where δ is the size of the window used when binning the spike data and i is sampled from $(-K/2, K/2)$. Augmented views $\mathbf{x}$ and $\mathbf{x}'$ are effectively found in a $(-K, K)$ window around sample $\mathbf{s}$. We use different parameters for each dataset, for the monkey and mouse datasets we use $K = 5$, and for the rat dataset we use $K = 1$.

Randomized dropout: When working with neural data, we consider randomized dropout (Bouthillier et al., 2016) as an augmentation. The dropout rate is uniformly sampled between p_{min} and p_{max}. In our monkey experiments, we find that $p_{\text{min}} = 0.0$ and $p_{\text{max}} = 0.8$ is a good range for the dropout rate. Note that for the Chewie-day 1 dataset, in a $100ms$ window 14.36% of the neurons on average are active, which would make the effective dropout rate actually lower. For the rodent dataset, which is binned using a $4s$ window, we use $p_{\text{min}} = 0.2$ and $p_{\text{max}} = 0.2$.

Gaussian noise: Random Gaussian noise with mean 0 and standard deviation 0.1 is applied after 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, representing 0.8σ where σ is the standard deviation of that neuron.

Note on structured transform: There are two ways of applying the proposed transformations. We can apply two different independant random transformations $t, t' \sim \mathcal{T}$ to generate augmented views. Or we can apply temporal shift to obtain $\mathbf{s}_1$ and $\mathbf{s}_2$ from $\mathbf{s}$ and then apply the same $t \sim \{\text{RDrop, Noise, Pepper}\}$ to both sample to get $\mathbf{x} = t(\mathbf{s}_1)$ and $\mathbf{x}' = t(\mathbf{s}_2)$. We call the latter "structured transform" and use it in our monkey experiments, where it can be seen as generating new trials as opposed to generating new individual firing rate vectors.

C. Additional Experiments

C.1. Experiments with projector layout

In Table S1, we report the results of MYOW on the MNIST dataset for different architectures used for incorporating mined views into our objective. We show the results for two different settings, weak augmentation (Crop only) and strong augmentation (All).

Table S1. 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

C.2. Impact of different augmentations on decoding accuracy

In Table S2, we show how different augmentations impact neural datasets not detailed in the main text. The findings are echoed through all monkey datasets.

In Table S3, 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.

Table S2. 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 labeled *Reach*, we computed the Accuracy in our reach direction prediction task when we apply a given set of transformations.

	TS	RDrop	Noise	Pepper	Accuracy			
					Chewie-Day 1	Chewie-Day 2	Mihi-Day 1	Mihi-Day 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

Table S3. 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

Figure S2. Examples of views mined by MYOW. We visualize the views mined by MYOW during training on the CIFAR-10 dataset.