Neuronal Learning Analysis using Cycle-Consistent Adversarial Networks

Anonymous authors
Paper under double-blind review

Abstract

Recent advances in neural imaging technologies enable high-quality recordings from hundreds of neurons over multiple days, with the potential to uncover how activity in neural circuits reshapes over learning. However, the complexity and dimensionality of population responses pose significant challenges for analysis. To cope with this problem, existing methods for studying neuronal adaptation and learning often impose strong assumptions on the data or model that may result in biased descriptions of the activity changes. In this work, we avoid such biases by developing a data-driven analysis method for revealing activity changes due to task learning. We use a variant of cycle-consistent adversarial networks to learn the unknown mapping from pre- to post-learning neuronal responses. To do so, we develop an end-to-end pipeline to preprocess, train, validate and interpret the unsupervised learning framework with calcium imaging data. We validate our method on two synthetic datasets with known ground-truth transformation, as well as on V1 recordings obtained from behaving mice, where the mice transition from novice to expert-level performance in a visual-based behavioral experiment. We show that our models can identify neurons and spatiotemporal activity patterns relevant to learning the behavioral task, in terms of subpopulations maximizing behavioral decoding performance and task characteristics not explicitly used for training the models. Together, our results demonstrate that analyzing neuronal learning processes with data-driven deep unsupervised methods can unravel activity changes in complex datasets.

1 Introduction

One of the objectives in computational neuroscience is to study the dynamics of neural processing and how neural activity reshapes when learning a task. A major hurdle in this endeavor was the difficulty in obtaining high-quality neural recordings of the same set of neurons across an extended period of learning a task. With the advent of modern neural imaging technologies, it is now possible to monitor a large population of neurons over days or even weeks (Williams et al., 2018a; Steinmetz et al., 2021), thus allowing experimentalists to obtain *in vivo* recordings from the same set of neurons across different learning stages.

Significant efforts have been put into extracting interpretable descriptions of how cortical responses change with experience. Proposed approaches to model changes in neuronal activity include linear latent variable models such as PCA, TCA, GPFA, GPFADS, and PSID (Cunningham & Byron, 2014; Williams et al., 2018b; Sani et al., 2021; Yu et al., 2009; Rutten et al., 2020), as well as methods employing deep learning models but with linear changes or mapping including LFADS and PfLDS (Pandarinath et al., 2018; Gao et al., 2016). While these methods enabled substantial progress in understanding the structure of neuronal activity, they impose strong assumptions inherent in the modeling technique or the analysis, such as the linearity assumption in linear latent variable models. Therefore, making sense of the unknown mapping between pre- and post-learning neural activity in an unbiased manner remains a significant challenge, and a data-driven method to interpret the circuit dynamics in learning is highly desirable.

In recent years, deep neural networks (DNNs) have seen tremendous success in many biomedical applications (Cao et al., 2018; Zemouri et al., 2019; Piccialli et al., 2021) thanks to their ability to identify and learn features from complex data in a data-driven way. Specifically, deep generative networks have shown

promising results in analyzing and synthesizing neuronal activities. For instance, Pandarinath et al. (2018) developed a variational autoencoder (VAE) to learn latent dynamics from single-trial spiking activities and Prince et al. (2021) extended the framework to work with calcium imaging data. Moreover, numerous works have demonstrated that generative adversarial networks (GAN) are capable of synthesizing neuronal activities that capture the low-level statistics of recordings obtained from behaving animals (Molano-Mazon et al., 2018; Ramesh et al., 2019; Li et al., 2020), suggesting that this model class is a good candidate for revealing changes in long-term recordings.

In this work, we explore the use of cycle-consistent adversarial networks (Zhu et al., 2017), or CycleGAN, to learn the mapping between pre- and post-learning neuronal activities in an unsupervised and data-driven manner. In other words, given the neural recordings of a novice animal, can we translate the responses that correspond to the animal with expert-level performance, and vice versa? The resulting transformation summarizes these changes in response characteristics in a compact form and is obtained in a fully data-driven way. Such a transformation can be useful in follow-up studies to 1) identify neurons that are particularly important for describing the changes in the overall response statistics, not limited to first or second-order statistics; 2) detect response patterns relevant for changes from pre- to post-learning; 3) determine what experimental details are of particular interest for learning. To summarize, our work brings the following contributions:

- We derive an end-to-end procedure to train, validate and interpret the unsupervised learning framework on neuronal recordings.
- We empirically evaluate the CycleGAN performance with 4 different commonly used GAN objective formulations, including GAN (Goodfellow et al., 2014), LSGAN (Mao et al., 2017), WGANGP (Arjovsky et al., 2017) and DRAGAN (Kodali et al., 2017).
- We validate our method on 3 datasets: 1) a simulated dataset where we can compare the transformation result against the ground-truth neuron firing patterns; 2) an augmented dataset where a handcrafted transformation is applied to examine the reconstruction performance of the model from unpaired data; 3) a recorded dataset where V1 neurons from behaving mice were monitored across multiple days in a virtual environment, where the first and last day of the recording should reflect pre-learning and post-learning neuronal activities, respectively.
- We incorporate self-attention and feature-importance visualization techniques into the model architectures and framework, which enable us to visualize and identify neurons that the models deemed relevant in the transformation process.
- We perform a decoding analysis on two behavioral variables and show that using the top-30 neurons learned by the models, can achieve similar decoding performance as using all neurons.
- We propose a novel neuron ordering method that can improve the learning performance of convolutional-based neural networks by per-sorting the spatial order of the neurons as a preprocessing step.

Notations used in this manuscript are listed in Table A.1 for convenience.

2 Methods

2.1 CycleGAN

CycleGAN (Zhu et al., 2017) is a GAN-based unsupervised framework that learns the mapping between two unpaired distributions X and Y via adversarial training and cycle-consistency optimization. The framework has shown excellent results in a number of unsupervised translation tasks, including natural language translation (Gomez et al., 2018) and molecular optimization (Maziarka et al., 2020), to name a few. The CycleGAN framework is a core part of the unsupervised learning mechanism used in this work. Here, we provide a brief description of the method and a number of variations that we employed in this project.

Table 1: The optimization objectives for generator G and discriminator D_Y (symmetric to F and D_X) in GAN (Goodfellow et al., 2014), LSGAN (Mao et al., 2017), WGANGP (Arjovsky et al., 2017) and DRAGAN (Kodali et al., 2017) formulations. λ_{GP} denotes the gradient penalty coefficient in WGANGP and DRAGAN, ϵ is the $[0, 1]$ linear interpolation coefficient for WGANGP and c is the Gaussian standard deviation for DRAGAN.

Model	Loss functions of G and D_Y	
GAN	$\mathcal{L}^G$	$= - \mathop{\mathbb{E}}_{x \sim X} \Big[\log(D_Y(G(x)) \Big]$
	$\mathcal{L}^{D_Y}$	$= - \mathop{\mathbb{E}}_{y \sim Y} \Big[\log(D_Y(y)) \Big] - \mathop{\mathbb{E}}_{x \sim X} \Big[\log(1 - D_Y(G(x))) \Big]$
LSGAN	$\mathcal{L}^G$	$= - \mathop{\mathbb{E}}_{x \sim X} \Big[(D_Y(G(x) - 1)^2 \Big]$
	$\mathcal{L}^{D_Y}$	$= - \mathop{\mathbb{E}}_{y \sim Y} \Big[(D_Y(y) - 1)^2 \Big] + \mathop{\mathbb{E}}_{x \sim X} \Big[D_Y(G(x))^2 \Big]$
WGANGP	$\mathcal{L}^G$	$= - \mathop{\mathbb{E}}_{x \sim X} \Big[D_Y(G(x)) \Big]$
	$\mathcal{L}^{D_Y}$	$= \mathop{\mathbb{E}}_{x \sim X} \Big[D_Y(G(x)) \Big] - \mathop{\mathbb{E}}_{y \sim Y} \Big[D_Y(y) \Big]$
		$+ \lambda_{\text{GP}} \mathop{\mathbb{E}}_{x \sim X, y \sim Y} \Big[\big(\parallel \nabla D(\epsilon y + (1 - \epsilon) G(x)) \parallel_2 - 1 \big)^2 \Big]$
DRAGAN	$\mathcal{L}^G$	$= \mathop{\mathbb{E}}_{x \sim X} \Big[\log(1 - D_Y(G(x))) \Big]$
	$\mathcal{L}^{D_Y}$	$= - \mathop{\mathbb{E}}_{y \sim Y} \Big[\log(D_y(y)) \Big] - \mathop{\mathbb{E}}_{x \sim X} \Big[\log(1 - D_Y(G(x))) \Big]$
		$+ \lambda_{\text{GP}} \mathop{\mathbb{E}}_{y \sim Y, z \sim \mathcal{N}(0, c)} \Big[\big(\parallel \nabla D(y + z) \parallel_2 - 1 \big)^2 \Big]$

Let X and Y be two distributions with unpaired mappings. CycleGAN consists of four networks: generator $G : X \to Y$ that maps novice activities to expert activities and generator $F : Y \to X$ that maps expert activities to novice activities; discriminator $D_X : X \to [0, 1]$ and discriminator $D_Y : Y \to [0, 1]$ that learn to distinguish novice and expert neural activities, respectively. In a forward cycle step ($X \to Y \to X$, illustrated in Figure B.1), we first sample a novice recording x from distribution X and apply transformation G to obtain $\hat{y} = G(x)$. We expect $\hat{y}$ to resemble data from the expert distribution Y. Hence D_Y learns to minimize (1) $\mathcal{L}^{D_Y} = -\mathbb{E}_{y \sim Y}[(D_Y(y) - 1)^2] + \mathbb{E}_{x \sim X}[D_Y(G(x))^2]$. Similar to a typical GAN, generator G learns to deceive D_Y with the objective of (2) $\mathcal{L}^G = -\mathbb{E}_{x \sim X}[(D_Y(G(x)) - 1)^2]$. Note that these are the same objectives used in LSGAN (Mao et al., 2017). However, D_Y can only verify if $\hat{y} \in Y$, though cannot ensure that $\hat{y}$ is the corresponding expert activity of the novice recording x. Moreover, X and Y are not paired. Hence we cannot directly compare $\hat{y}$ with samples in Y. To tackle this issue, another transformation is applied to reconstruct the novice recording $\bar{x} = F(\hat{y})$, where the distance $\parallel x - \bar{x} \parallel$ or $\parallel x - F(G(x)) \parallel$ should be minimal. Therefore, the generators also optimize this cycle-consistent loss (3) $\mathcal{L}_{\text{cycle}} = \mathbb{E}_{x \sim X}[\parallel x - F(G(x)) \parallel] + \mathbb{E}_{y \sim Y}[\parallel y - G(F(y)) \parallel]$. Mean absolute error (`MAE`) was used as the distance function, though other distance functions can also be employed. In addition, we would expect $\hat{x} = F(x)$ and $\hat{y} = G(y)$ to be in distributions X and Y given that $F : Y \to X$ and $G : X \to Y$, hence the identity loss objective (4) $\mathcal{L}^G_{\text{identity}} = \mathbb{E}_{y \sim Y}[\parallel y - G(y) \parallel]$.

Taken all together, G optimizes the following objectives: (5) $\mathcal{L}^G_{\text{total}} = \mathcal{L}^G + \lambda_{\text{cycle}} \mathcal{L}_{\text{cycle}} + \lambda_{\text{identity}} \mathcal{L}^G_{\text{identity}}$, where $\lambda_{\text{identity}}$ and λ_{cycle} are hyper-parameters for identity and cycle loss coefficients. All four networks are trained jointly ($\mathcal{L}^F_{\text{total}}$ and $\mathcal{L}^{D_X}$ are the same as $\mathcal{L}^G_{\text{total}}$ and $\mathcal{L}^{D_Y}$ but in opposite directions). To the best of our knowledge, very few works empirically evaluate other GAN formulations in the CycleGAN framework. Here, we evaluate three common GAN objectives (see Table 1), including GAN (Goodfellow et al., 2014), WGANGP (Arjovsky et al., 2017) and DRAGAN (Kodali et al., 2017).

2.2 Model pipeline

We devised an end-to-end analysis framework[1], including data preprocessing and augmentation, model training, model interpretation, and evaluation of the translated calcium signals and their inferred spike trains. The entire framework pipeline is summarized in Figure 1. Here, we first describe the most basic form of our framework. Model architectures, synthetic and recorded datasets, as well as other improvements and modifications to the CycleGAN framework to make it compatible with neural data are discussed in later subsections.

Generally, we use X and Y to denote the two unpaired distributions of neuronal activity, where the two generators G and F should learn the mapping of X to Y and Y to X, respectively. The two distributions are represented in the form of calcium fluorescence signals of W neurons over N segments, each segment consist of H time-steps, therefore, we can represent the data as a matrix with shape (N, H, W). In order to take advantage of the spatiotemporal information in calcium responses using 2-dimensional convolutional neural networks (CNNs), we further convert the segment to shape $(N, H, W, 1)$, effectively treating each segment as a gray-scale image. We normalize each set to the range $[0, 1]$, and divide the datasets into train-validation-test sets with a ratio of 70%:15%:15%.

In addition to the cycle-consistency and identity loss comparison metrics to validate the transformation performance, we also compare the first and second order statistics of the translated responses $\hat{X} = F(Y)$ and $\hat{Y} = G(X)$ against X and Y. To that end, we first infer spike trains from the fluorescence signals using Cascade (Rupprecht et al., 2021), the state-of-the-art spike deconvolution algorithm. We then compute the following spike train similarities and statistics: (1) mean firing rate for evaluating single neuron statistics; (2) pairwise Pearson correlation for evaluating pairwise statistics; (3) pairwise van Rossum distance (Rossum, 2001) for evaluating general spike train similarity. These quantities are evaluated across the whole population for each neuron or neuron pair and we compare the resulting distributions over these quantities obtained from the original and translated data: (a) $X \mid \hat{X}$, (b) $X \mid \bar{X}$, (c) $Y \mid \hat{Y}$ and (d) $Y \mid \bar{Y}$. We, therefore, validate the whole spatiotemporal first and second-order statistics as well as general spike train similarities.

Furthermore, we incorporated a number of recently proposed model interpretation methods into our pipeline in order to improve the explainability of this work, detailed in Section 2.3 and Section 2.4. Briefly, we designed a self-attention generator architecture in which the model is encouraged to learn a set of attention masks that filters out irrelevant information in the input data. These masks can then be used to visualize regions or neurons in the calcium responses that the model self-identified to be important. The attention module, being part of the generator architecture, operates in a low-dimension latent space. Thus, it can be difficult to identify fine-grain information from neuronal activity. To address this shortcoming, we implemented an additional feature importance method that enables a more detailed visual explanation with respect to the virtual reality (VR) experiment (more in Section 2.6). GradCAM, introduced in Selvaraju et al. (2017), is a post hoc algorithm to identify regions in the input (e.g. natural image) that a CNN classifier attends to in its classification process. In this work, we adapted GradCAM to generate localization maps (or activation maps) for all four networks. Note that, unlike the self-attention module, GradCAM does not consist of any trainable parameters and is not part of the overall optimization objective. The two visualization techniques allowed us to confirm that the models are indeed learning meaningful features from the neuronal responses.

All models were trained with the Adam optimizer (Kingma & Ba, 2014) for 200 epochs where the models converged. A single NVIDIA A100 GPU was used to train the various networks which, on average, took 15 hours to complete. We selected the hyper-parameters by a random search (Bergstra & Bengio, 2012) on the simulated dataset, and the same settings were then used in the subsequent datasets. The final hyper-parameters are shown in Table B.1.

[1]The software codebase is attached as a supplementary file and it will be made publicly available on upon acceptance.

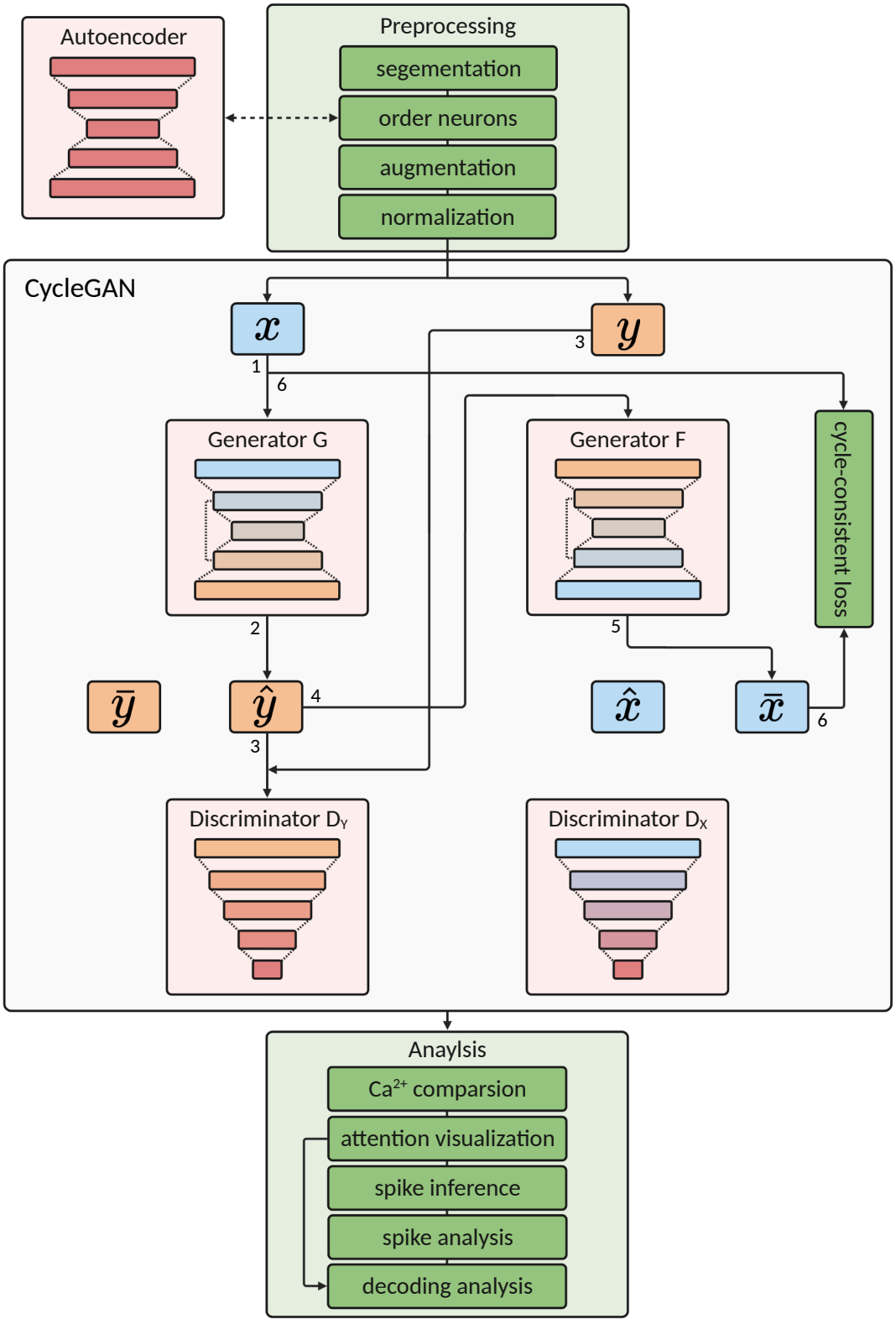


Figure 1: Illustration of the complete pipeline used in this work. Black directed lines represent the flow of data and the numbers indicate its order. Note that only the forward cycle step $X \rightarrow Y \rightarrow X$ is shown here for better readability.

2.3 Networks architecture

The generator used in this work, shown in Figure 2, is based on the architecture in CycleGAN with two main modifications: (1) level-wise residual connection and (2) attention-gated residual connection. Generally, the model consists of 2 down-sampling blocks ($\texttt{DS}_1$ and $\texttt{DS}_2$), followed by 9 residual blocks ($\texttt{RB}_i$ for $1 \leq i \leq 9$), then 2 up-sampling blocks ($\texttt{US}_1$ and $\texttt{US}_2$). Each down-sampling block (see Blue box in Figure 2) uses a 2D strided convolution layer to reduce the spatiotemporal dimensions by a factor of 2, which is then followed by Instance Normalization (`IN`, Ulyanov et al. 2016), GELU (Hendrycks & Gimpel, 2016) activation and Spatial Dropout. Each up-sampling block has the same structure as the down-sampling blocks but with a transposed convolution layer instead. Each residual block consists of two convolution blocks with padding added to offset the dimensionality reduction and a skip connection that connects the input to the block with the output of the last convolution block via element-wise addition. A convolution layer with a filter size of 1 then compresses the channel of the output from $\texttt{US}_1$, and finally, a sigmoid activation is applied to scale the output to the range $[0, 1]$.

Residual connections are known to improve gradient flow in CNNs, thus mitigating the issue of vanishing gradients (He et al., 2016a;b; Huang et al., 2017). Therefore, shortcut connections are added between the down-sampling and up-sampling blocks of the same level. For instance, the output of down-sampling block $\texttt{DS}_2$ is concatenated with the output of residual block $\texttt{RB}_9$, then passes the resultant vector to the next up-sampling block $\texttt{US}_1$. Such level-wise residual connections were first introduced in Ronneberger et al. (2015). We denote the level-wise residual network as `ResNet`.

Furthermore, we adapted the additive attention gate (AG) module in Oktay et al. (2018) as a replacement for the concatenation operation in the residual connection described above. The yellow block in Figure 2 illustrates the AG structure. AG takes two inputs q and a, both with height H_{AG} and width W_{AG} but varying channels, where q is the output of the previous processing block and a is a shortcut connection from the down-sampling block of the same level. In $\texttt{AG}_1$ for instance, q and a are the output of $\texttt{RB}_9$ and $\texttt{DS}_2$ respectively. Both q and a are processed by two separate 1×1 convolution layers and `IN`. The two vectors are then summed element-wise such that overlapping regions from the two vectors would have higher intensity. We then apply ReLU activation to eliminate negative values, followed by a 1×1 convolution layer with 1 filter and `IN`, resulting in a vector with shape $(H_{\text{AG}}, W_{\text{AG}}, 1)$. Sigmoid activation is applied to obtain a $[0, 1]$ attention mask σ, where units closer to 1 indicate regions that are more relevant. We apply the sigmoid mask to a, and concatenate it with q. Here, q is a set of high-level features processed by the stack of residual blocks, whereas a is the low-dimensional representation of the original input. Therefore, the sigmoid attention mask should learn to eliminate information in the input that is less relevant to the output. Moreover, as the attention mask is of the same dimension as the input q, we can later superimpose the attention mask onto q to visualize the region of interest learned by the model. We denote the attention-gated `ResNet` as `AGResNet`.

As for the discriminator, we use a PatchGAN-based (Isola et al., 2017) architecture, as it provides more fine-grained discrimination information to the generators instead of the single value discrimination in the discriminator in vanilla GAN. D_X and D_Y contain 3 down-sampling blocks where each block reduces the spatiotemporal dimension by a factor of 2, like the down-sampling blocks in the generators. For an input sample with shape $(H = 2048, W = 102, C = 1)$, the discriminator outputs a sigmoid activated vector with shape $(256, 13, 1)$. Each element has range $[0, 1]$ where a value closer to 1 suggests that the corresponding patch is from a real sample. The discriminators are kept relatively simple so that the generators would not be over-powered, especially in the initial phase of training.

2.4 Decoding analysis

The aforementioned visual explanation methods enable us to identify regions or neurons in the responses to which the models are more attentive. For instance, we expect the generators to focus on the activities surrounding the reward zone in the virtual corridor as the grating patterns disappear on the monitors. We hypothesize that a subset of neurons is more informative in the neuronal learning transformation and that the models should learn to be more attentive toward this group of neurons. To that end, we trained a regression model to decode behavioral variables (position and velocity) when provided with calcium responses of (1) all neurons, (2) top-30 (out of 102) neurons according to the activation maps, (3) rest of the neurons and

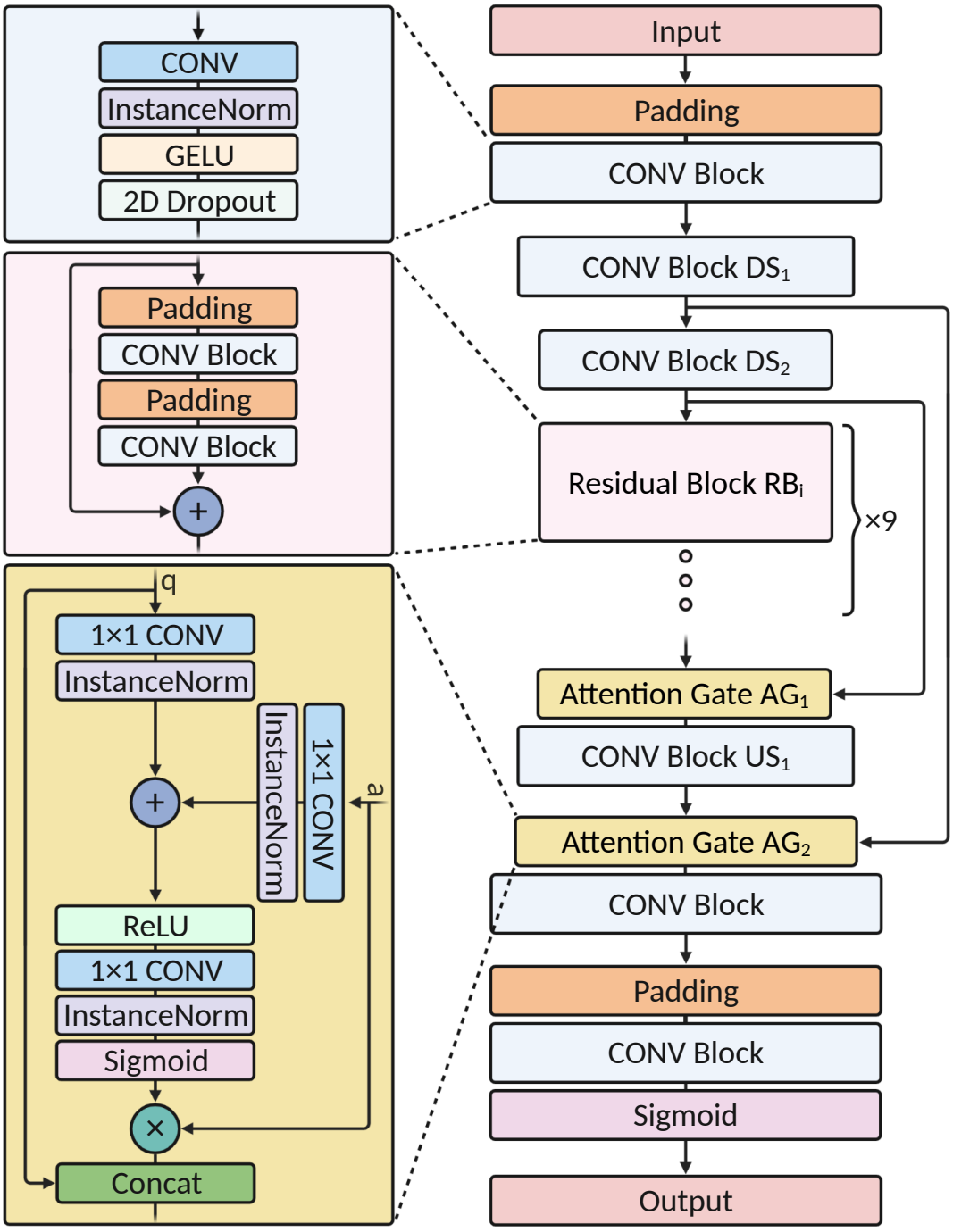


Figure 2: Architecture diagram of generator G and F. $+$ and $\times$ denote addition and element-wise multiplication respectively. Note that the additive attention gate (AG) block can be replaced by a concatenation operation between the output of the previous block and the output from the down-sampling block from the same level. For example, if AG is not used, then the input to US_1 is $\texttt{concat}(DS_2, RB_9)$.

(4) 30 randomly selected neurons. We kept the regression model fairly simple since we are only interested in the change in performance when training the model with different combinations of neurons. Here, we trained a recurrent neural network (RNN) decoder which consists of a GRU-layer (Cho et al., 2014) with 128 units followed by a fully-connected layer to output a scalar. The model was trained to optimize the mean-squared error (`MSE`) between its predictions and the behavioral variables using the Adam optimizer (Kingma & Ba, 2014). Finally, for each variable-neuron combination, we fit the decoder 20 times, each with a different random seed, and compare their performance in terms of R^2 on the test set. If the selected neurons are indeed more influential, then in contrast to using (1) all neurons, we expect a drop in performance when the model is trained with the (3) rest of the neurons. Moreover, the decoding accuracy when provided with (2) top-30 neurons should be significantly different from (4) selecting 30 neurons randomly.

2.5 Neuron ordering

As discussed in Section 2.3, we are using convolutional-based networks for the generators and discriminators. It has been shown that CNNs with a smaller kernel can often perform as well or even better than models with larger kernels while maintaining fewer trainable parameters thus easier to train (He et al., 2016a; Li et al., 2021). Nevertheless, a small kernel can potentially limit the receptive field of the model, or the region in the input that the model is exposed to in each convolution step (Araujo et al., 2019). In addition, the recordings obtained from the VR experiment were annotated based on how visible the neurons were in the calcium image, rather than ordered in a particular statistical manner (see Figure 3). This could potentially restrict CNNs with a small receptive field to learn meaningful spatial-temporal information from the population responses. To mitigate this issue, we propose a novel procedure to pre-sort X and Y, such that neurons that are highly correlated or relevant are nearby in their ordering. A naive approach is to sort the neurons by their firing rate or pairwise correlation, where the neuron with the highest firing rate or the neuron that, on average, is most correlated to other neurons is ranked first in the input array. However, it is possible that not all high-firing neurons or most correlated neurons are the most influential in the learning process. This calls for an automated and data-driven approach to rank neurons in a meaningful order. Deep autoencoders have shown excellent results in feature extraction and representation learning (Gondara, 2016; Wang et al., 2016; Tschannen et al., 2018), and we can take advantage of their unsupervised feature learning ability.

We employed a deep autoencoder `AE` which learns to reconstruct calcium signals in X and Y jointly. `AE` is a fairly typical convolution-based autoencoder with 3 (encoder) down-sampling and (decoder) up-sampling blocks, and a bottleneck layer in between. The down-sampling block consists of a 1D convolution layer followed by `IN`, GELU activation, and Spatial Dropout, whereas a 1D transpose convolution is used in the up-sampling block instead. We first optimize the mean-squared error (`MSE`) reconstruction loss on the training set of X and Y. Then we use the per-neuron reconstruction error on the validation set to sort neurons in ascending order: (6) neuron order $= \text{argsort}\Big(0.5 \times \big[\,\texttt{MSE}(X, \texttt{AE}(X)) + \texttt{MSE}(Y, \texttt{AE}(Y))\,\big]\Big)$. It is important to note that the proposed neuron ordering process is an optional data preprocessing step that allows 2D convolution-based models to take advantage of the spatial information presented in neuronal responses and is not mandatory for the unsupervised learning method to work. We also compare our method against neurons ordered by their original annotation, average firing rate and pairwise correlation. Furthermore, to demonstrate that 2D convolution can indeed better learn the spatial structure in neuronal responses, we added a 1D variant of `AGResNet` (denote as `1D-AGResNet`) as a baseline which disregards all spatial information in the data.

2.6 Recorded data

To record neuronal activities with pre- and post-learning responses, we conducted a virtual reality (VR) experiment that follows a similar procedure as in Pakan et al. (2018) and Henschke et al. (2020). Briefly, a head-fixed mouse was placed on a linear treadmill that allows it to move forward and backward. A lick spout and two monitors were placed in front of the treadmill and a virtual corridor with a defined grating pattern was shown to the mouse. A reward (water drop) was available if the mouse licked within the predefined reward location in the virtual corridor (at 120 to 140 cm), in which a black screen is shown as a visual clue, and the mouse learns to utilize both visual information and self-motion feedback to maximize reward. Figure 3

illustrates the experiment setup. The same set of neurons in the primary visual cortex was labeled with the GCaMP6 calcium indicator and monitored throughout 5 days of experiments. The fluorescence signals were then decontaminated and extracted from the calcium imaging data using FISSA (Keemink et al., 2018), and the relative changes in fluorescence ($\Delta F/F_0$) over time were used as a proxy for an action potential. Four mice were trained in the experiment and all mice achieved expert-level performance within 4 days of training. For instance, Mouse 1 took on average 36% less time to complete a trial with a 52% improvement in the received rewards from day 1 to day 4. Trial information of Mouse 1 is shown in Table 2 (see Appendix C for Mouse 2 - 4). This dataset provides excellent insights into how cortical responses reshape with experience, and therefore, we utilize the recordings obtained on the 1$^{\text{st}}$ (pre-learning) and 4$^{\text{th}}$ day[2] (post-learning) of the experiment as X_{rec} and Y_{rec}, respectively.

We want the models to identify patterns relevant to the animal experiment in a completely data-driven manner. To this end, the models get the fluorescence input signals with minimal data preprocessing. Briefly, we segment the two recording sessions X_{rec} and Y_{rec} with a sliding window of size $H = 2048$ along the temporal dimension (around 85s in wall-time), resulting in data with shape (N, H, W) where W is the number of neurons and N is the number of segments. We then follow the same pre-processing procedure as specified in Section 2.2.

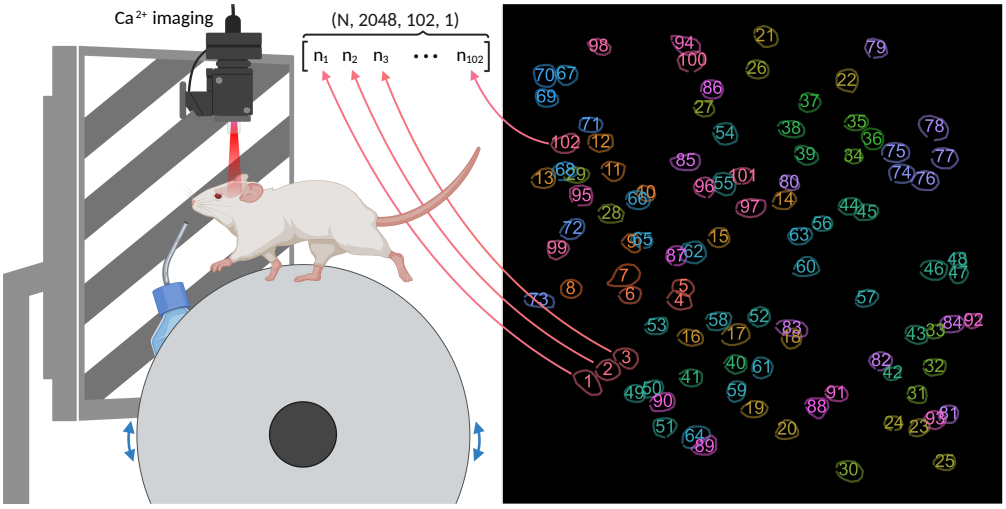


Figure 3: (Left) illustration of the mouse virtual-environment setup. A defined grating pattern is displayed on the monitors and the mouse can move forward and backward in the virtual corridor. When the mouse approaches the reward zone, which was set at 120 cm to 140 cm from the initial start point, the grating pattern would disappear and be replaced with a blank screen. If the mouse licked within the virtual reward zone, then a droplet of water was given to the mouse as a reward. Trials reset at 160 cm. The figure is based on Figure 1 in Pakan et al. (2018). (Right) original coordinates and annotation order of the 102 recorded neurons. i.e. neuron 1 here would be at index 0 in the data matrix, and neuron 65 would be at index 64.

Table 2: Mouse 1 trial information where 102 V1 neurons were monitored across 5 days of training, and the rodent achieved "expert" level by day 4 with a success rate of $> 75\%$ at the task.

Day	Duration	Num. trials	Avg. trial duration	Licks	Rewards
1	894.73s	129	6.94s	2813	140
2	898.68s	177	5.08s	2364	182
3	897.16s	192	4.67s	2217	198
4	898.45s	203	4.43s	1671	213
5	897.25s	264	3.40s	1298	327

[2]Mouse 2 and 3 performed worse on the 5$^{\text{th}}$ day, hence, here, we use the recordings obtained on the 4$^{\text{th}}$ day.

2.7 Synthetic data

Given that the transformation from pre-learning to post-learning neuronal activities is unknown, there is no simple method to assess the alignment performance in calcium imaging data. In addition, CycleGAN was first introduced for image-to-image translation. Albeit the two image distributions are not aligned and hence cannot be compared directly, one could still visually inspect whether or not $\hat{x} = F(y)$ and $\hat{y} = G(x)$ are reasonable transformations. However, evaluating a large number of calcium traces via visual inspection would be exceedingly difficult, if not impossible. Instead, we introduce two synthetic datasets with known ground truth: a simulated dataset (Section 2.7.1) and an augmented dataset (Section 2.7.2), which enable us to assess and visualize the translation performance when learning from unpaired calcium signals. For clarity purposes, we denote the simulated data as X_{sim} and Y_{sim}, the augmented data as X_{aug} and Y_{aug} and the original recorded data as X_{rec} and Y_{rec} for the rest of this paper.

2.7.1 Simulated data

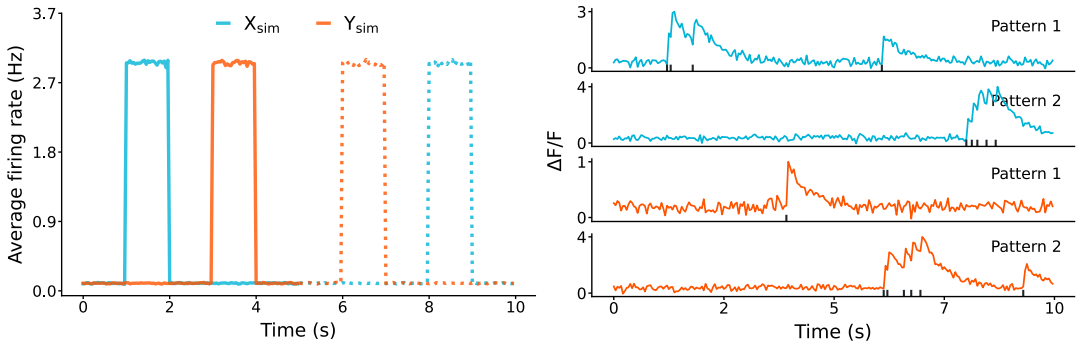


Figure 4: The (Left) average firing pattern of two simulated populations, (Blue) X_{sim} and (Orange) Y_{sim}. Each simulated population is divided in half, where each half exhibits different firing patterns sampled from a Poisson distribution. Neurons in X_{sim} are high firing (3Hz) from 17s to 34s (solid line) and from 59s to 73s; whereas Y_{sim} are active from 12s to 22s and from 46s to 60s. (Right) Two randomly selected spike trains and their convolved calcium-like traces from the two simulated populations, the top, and bottom panels show the first and second firing pattern respectively. Figure E.1 shows a sample population of x_{sim} and y_{sim}.

We first simulate the neuronal responses of two populations of size 128. Each trial is sampled from a Poisson distribution with a trial duration of 10s (i.e. 240 time-steps at 24Hz) for a total of 1400 trials (i.e. about the same as the Mouse 1 recorded data in Section 2.2). In addition to the background activity at ~ 0.1 Hz, each population consists of two distinct firing patterns. Half of the neurons in the first population are highly active (~ 3 Hz) from 17s to 34s, and the other half from 59s to 73s; whereas neurons in the second population are active from 12s to 22s and from 46s to 60s. We use an exponential onset and double decay function f_{Ca}, as described in Grewe et al. (2010), to obtain fluorescence-like traces from the spike times t:

$$f_{\text{Ca}}(t) = \begin{cases} 0 & \text{for } t \leq t_0 \\ \left[\, 1 - e^{-(t-t_0)/\tau_{\text{onset}}} \,\right] \left[\, A_1 e^{-(t-t_0)/\tau_1} + A_2 e^{-(t-t_0)/\tau_2} \,\right] & \text{otherwise} \end{cases} \tag{7}$$

where A_1, τ_1, A_2 and τ_2 are the amplitude and decay time parameters for the first and second exponential decay; τ_{onset} is the action potential onset time. We then add Gaussian noise (signal-to-noise ratio of 10) to the convolved traces to improve realism. We denote the simulated responses from the two populations as X_{sim} and Y_{sim}, their overall firing patterns and samples of calcium-like traces are shown in Figure 4. Notably, we shuffle the neuron index such that the traces appear less structured and thus increase the difficulty for the generators to learn the mapping between the two sets, an example of the shuffled populations is shown in Figure E.1. This simulated dataset allows us to directly compare the activity patterns of the transformed traces, $\hat{y}_{\text{sim}} = G(x_{\text{sim}})$ and $\hat{x}_{\text{sim}} = F(y_{\text{sim}})$, against the ground truth patterns.

2.7.2 Augmented data

We investigated whether or not the unsupervised learning framework can recover or translate neuron-specific responses from unpaired calcium traces; furthermore, to verify the aforementioned visual explanation methods can reveal meaningful patterns from the neuronal recordings, we introduced an additional synthetic dataset – the augmented dataset X_{aug} and Y_{aug}, where $Y_{\text{aug}} = \Phi(X_{\text{aug}})$ with a handcrafted spatiotemporal transformation: (8) $\Phi(x_{\text{aug}}) = m_{\text{diagonal}} x_{\text{aug}} + 0.5\eta$, where m_{diagonal} is a diagonal mask to zero-out the lower left corners of the signals and η is sampled from Gaussian noise $\mathcal{N}(\mu_x, \sigma_x^2)$ with the per-neuron mean and standard deviation from x_{aug}. Importantly, we shuffle the training and validation set after the augmentation procedure, so that X_{aug} and Y_{aug} are unpaired during model training, while maintaining the original neuron order in the test set, hence allowing us to compute $\| X_{\text{aug}} - F(Y_{\text{aug}}) \|$ and $\| Y_{\text{aug}} - G(X_{\text{aug}}) \|$ using common distance metrics. An example of x_{aug} and y_{aug} is available in Figure 5 (example traces are available in later Sections). Such spatiotemporal augmentation, though biologically unrealistic, allows easy visual verification of the transformation learned by the generators. In addition, one would expect the models to focus on regions surrounding the masked area in x_{aug} and y_{aug}, thus, allowing us to ensure that the attention gate modules and localization maps function as intended.

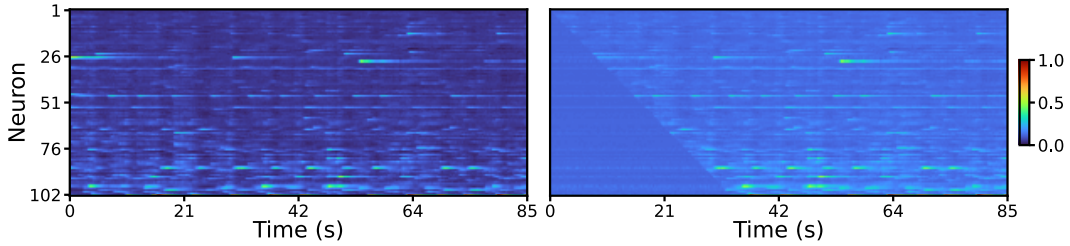


Figure 5: Example of a randomly selected segment from Mouse 1 recordings (Left) x_{aug} and its corresponding (Right) augmented $y_{\text{aug}} = \Phi(x_{\text{aug}})$. Note the bottom left corner in y_{aug} has been masked out with noise added to the segment. `TURBO` color-map is applied to improve readability.

3 Results

The main objective of the framework is to identify a meaningful transformation between two neuronal datasets in a data-driven manner. We first assessed the framework's ability to learn the mapping between two data distributions using two synthetic datasets with known ground truth pairing. We then applied the same method on recorded data obtained from the primary visual cortex of behaving mice in a VR experiment.

3.1 Simulated data

In order to ensure that the framework can learn to transform calcium signals from one firing pattern to another, we first fit our models on the simulated dataset with the `AGResNet` generator architecture and LSGAN objectives. In order to increase the difficulty of the mapping task, neurons in X_{sim} and Y_{sim} were originally shuffled jointly prior to model training (see the bottom row in Figure E.1), and we rearranged the neurons in the test set to their original order after inference. Figure 6 shows the average activity patterns (in $\Delta F/F$) of X_{sim} against $\hat{X}_{\text{sim}} = F(Y_{\text{sim}})$ and Y_{sim} against $\hat{Y}_{\text{sim}} = G(X_{\text{sim}})$ over the entire population. The transformations in both directions achieved excellent results with `NRMSE` $= 0.0494$ and $R^2 = 0.9680$ between X_{sim} and $\hat{X}_{\text{sim}}$, and conversely, `NRMSE` $= 0.0345$ and $R^2 = 0.9845$ between Y_{sim} and $\hat{Y}_{\text{sim}}$. In all cases, the translated responses can closely match the two activity patterns of their respective target distributions, and demonstrate the model's unsupervised learning capabilities when trained with calcium signals.

3.2 Augmented data

Next, we fit our models on the augmented dataset, with known handcrafted augmentation, to show that our method is capable of learning subtle differences and recovering responses from unpaired calcium traces.

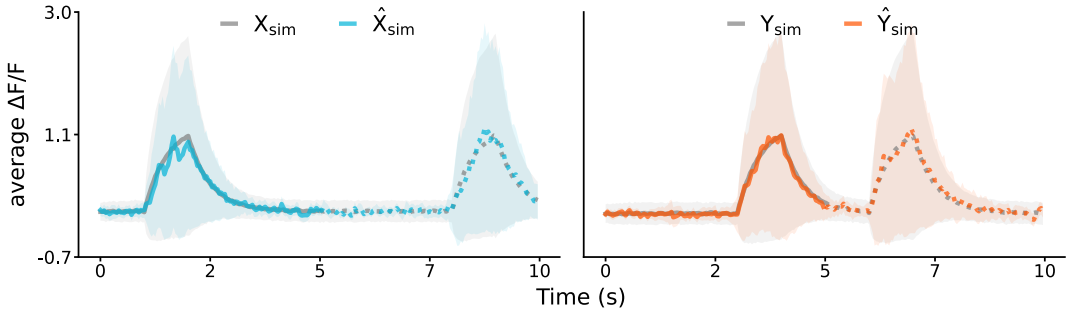


Figure 6: The average response patterns of (Left) $\hat{X}_{\text{sim}}$ against X_{sim} (`NRMSE` $= 0.0494$, $R^2 = 0.9680$), and (Right) $\hat{Y}_{\text{sim}}$ against Y_{sim} (`NRMSE` $= 0.0345$, $R^2 = 0.9845$). The solid and dotted lines indicate the two activity patterns in each population, the gray and colored lines correspond to the average simulated and translated responses, and the shaded areas show their variance. For reference, $\texttt{NRMSE}(X_{\text{sim}}, Y_{\text{sim}}) = 0.4577$ and $R^2(X_{\text{sim}}, Y_{\text{sim}}) = 0.1884$.

Figure 9 shows calcium signals of the forward and backward cycle transformation from 3 neurons in a population, each with a different level of masked activities. Without paired samples, G was able to learn the augmentation Φ and mask out the appropriate regions in x_{aug}, and conversely, F was able to recover responses from the masked regions in y_{aug}. For instance, F was able to recover the 4 spikes of fluorescence signals from the augmented input in Neuron 98 (bottom panel in Figure 9). Given that the difference between X_{aug} and Y_{aug} is the replacement of noise in the lower-ranked neurons (i.e. lower-left corner if we plot the population response as a 2D image), we expected D_Y to discriminate based on activities around the augmentation region. The activation map of $D_Y(y_{\text{aug}})$, shown in Figure 8, indeed demonstrates a high level of attention along the edge of the diagonal region. On the other hand, since no augmentation was done on the input to D_X, the localization map does not appear to have a particular structural area of focus. Interestingly, once we overlay the reward zones onto the input segment, we observed that the areas of focus learned by D_X are loosely aligned with the reward zones. Note that reward zones are external task-relevant regions that are expected to shape the neural activity in the primary visual cortex as the visual patterns change when the mouse enters the reward zone. These findings suggested that D_Y learned to distinguish an input by predominantly focusing on the edge of the masking area, whereas D_X learned distinctive patterns from highly ranked neurons around the reward zones. We then inspected what the generators have learned, with respect to their given inputs, via the attention gate modules. The attention masks AG_1 and AG_2 in G and F, displayed in Figure 7, suggested that both generators ignored responses in the augmentation region. This is likely caused by the fact that information in that area is not relevant to either transformation. G learns to replace the calcium traces from the to-be-masked region with noise, hence ignoring the information from that region; F learns to recover responses in the masked region, as the information from the masked region is noise and hence not relevant for the model to learn from.

X_{aug} and Y_{aug} are paired in the test set, which enables us to directly evaluate the transformation result with common distance metrics, making this a good testbed to compare different generator architectures, GAN objective formulations, and neuron ordering methods. Results on the test set are summarized in Table 3. We first compared generator architectures when trained with the LSGAN objective. The `AGResNet` achieved transformation errors of $\texttt{MAE}(X_{\text{aug}}, \hat{X}_{\text{aug}}) = 0.1508$ and $\texttt{MAE}(Y_{\text{aug}}, \hat{Y}_{\text{aug}}) = 0.2520$, outperforming the `ResNet` and baseline `identity` model. We then investigated how different commonly used GANs objectives perform in the CycleGAN settings. Interestingly, models trained with DRAGAN and WGANGP objectives obtained lower cycle-consistent errors than GAN and LSGAN yet performed poorly in the intermediate transformations $F(Y_{\text{aug}})$ and $G(X_{\text{aug}})$. We observed that $D_X(F(Y_{\text{aug}}))$ and $D_Y(G(X_{\text{aug}}))$ were neither informative nor impactful to the overall objective and that the generators focused on optimizing the cycle-consistent loss instead. It is likely that the gradient penalty term in DRAGAN and WGANGP further complicates the already perplexing overall optimization objective, hence hindering the discriminators from learning meaningful features and being overpowered by the generators. Finally, we compared if and how different neuron ordering methods affect the unsupervised mapping performance. `1D-AGResNet`, which disregards spatial information

in the data, performed significantly worse than `AGResNet` with 2D convolution, suggesting that the spatial structure in the neural activities is indeed essential and learnable by the models. Overall, models trained on sorted neurons achieved better results, and neurons ordered by the `AE` reconstruction loss is the most performant. The proposed neuron preprocessing step can thus be a simple to implement and effective method to enhance the learning performance of convolution-based models. In the remaining work, we use the LSGAN objective to train the generators with the `AGResNet` architecture along with neurons ordered based on `AE` reconstruction errors, as this combination achieved the best overall results on the augmented data.

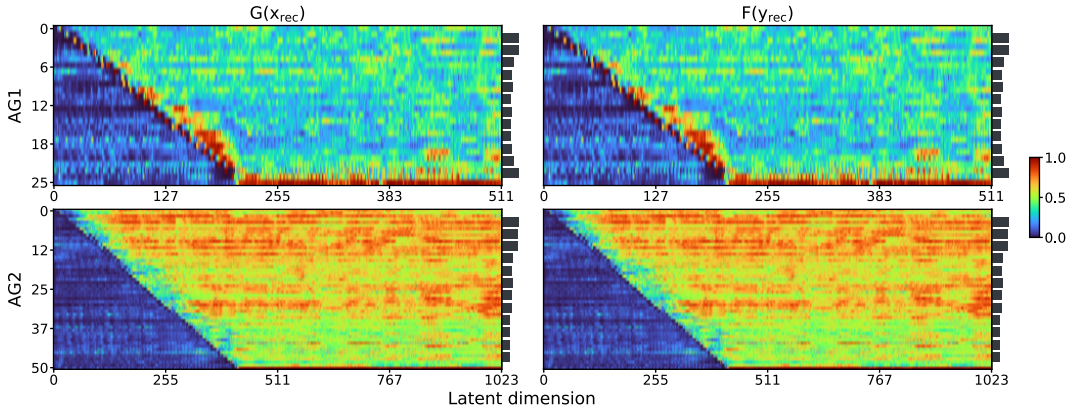


Figure 7: Learned attention masks AG_1 and AG_2 from `AGResNet` generators (Left) $G(x_{\text{aug}})$ and (Right) $F(y_{\text{aug}})$. The extracted sigmoid masks showed that the generators learned to ignore the augmentation region in the lower left corner of the latent dimension. `TURBO` color-map is used to improve visibility and the histogram on the right of each panel shows the neuron-wise attention distribution.

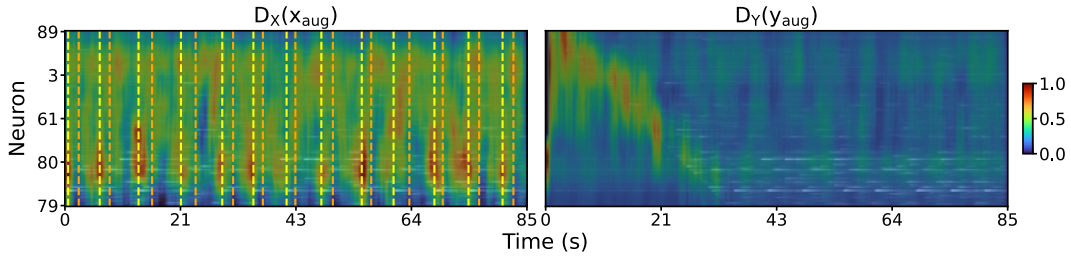


Figure 8: Activation maps of (Left) $D_X(x_{\text{aug}})$ and (Right) $D_Y(y_{\text{aug}})$ overlaid their respective input. As expected, D_Y were attentive toward activities along the diagonal masking area as the most prominent feature of y_{aug} is the augmentation region. Interestingly, D_X focused on neuronal activities surrounding the reward zones (indicated by the yellow and orange vertical dotted lines). Neurons were ordered based on `AE` reconstruction loss, the exact ordering is available in Table D.1.

3.3 Recorded data

The results from the two synthetic datasets demonstrated the framework's capability in learning the unknown mappings in calcium traces. Moreover, the visual explanation methods indicated that the networks are indeed learning meaningful features. We then applied our framework onto (X_{rec}) pre-learning and (Y_{rec}) post-learning V1 activities obtained from behaving rodents. Figure 10 shows the cycle transformation of 3 randomly selected neurons. Visually, G and F were able to reconstruct $\bar{x}_{\text{rec}} = F(G(x_{\text{rec}}))$ and $\bar{y}_{\text{rec}} = G(F(y_{\text{rec}}))$, and that the two generators were not simply passing through x_{rec} and y_{rec} in an intermediate step to translate $\hat{y}_{\text{rec}} = G(x_{\text{rec}})$ and $\hat{x}_{\text{rec}} = F(y_{\text{rec}})$. To better analyze the transformation performance, we first compared the translated fluorescence signals against the recorded data in the test set. The models achieved cycle-consistent losses of $\texttt{MAE}(X_{\text{rec}}, F(G(X_{\text{rec}}))) = 0.0733$ and $\texttt{MAE}(Y_{\text{rec}}, G(F(Y_{\text{rec}}))) = 0.0737$, as well as identity losses for $\texttt{MAE}(Y_{\text{rec}}, G(Y_{\text{rec}})) = 0.0101$ and $\texttt{MAE}(Y_{\text{rec}}, G(Y_{\text{rec}})) = 0.0069$, which are significantly better

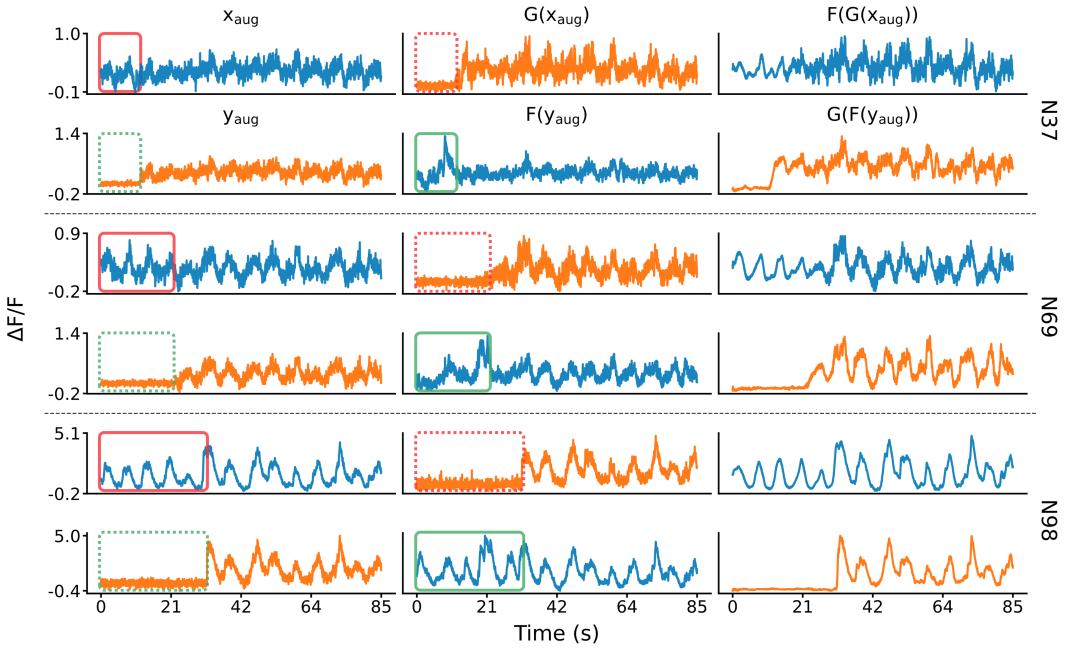


Figure 9: Forward and backward cycle steps of neurons 37, 69, and 98 (top, middle, and bottom of the population) from a randomly selected test segment. G learns to perturb the initial part of the traces (i.e. from red solid box to red dotted box) and F learns to reconstruct masked out regions in the traces (i.e. from green dotted box to green solid box). We expect the traces in the green solid box to resemble signals in the yellow solid box and the yellow dotted box in the green dotted box. Example transformation on the entire population is available in Figure F.1.

than the baseline $\texttt{MAE}(X_{\text{rec}}, Y_{\text{rec}}) = 0.3674$, indicating that G and F were not simply passing through the data. In addition, the low identity loss suggests that the generators can correctly identify whether or not the given input is already in their respective target distributions, hence performing no operation. Table 4 reports the cycle-consistent and identity loss with different neuron ordering methods. Again, as demonstrated in Section 3.2, `1D-AGResNet` performed significantly worse than 2D convolutional-based models where it cannot take advantage of the spatial information from the calcium responses. In all cases, the model trained with neurons ordered by `AE` reconstruction error is the most performant, and ordering neurons in any meaningful way brings measurable improvements.

Since we lack paired data in the *in vivo* recordings, we cannot directly compare X_{rec} with $\hat{X}_{\text{rec}} = F(Y_{\text{rec}})$ nor Y_{rec} with $\hat{Y}_{\text{rec}} = G(X_{\text{rec}})$, in contrast to the augmented dataset in Section 3.2. In order to verify the two intermediate transformations $\hat{Y}_{\text{rec}}$ and $\hat{X}_{\text{rec}}$ are indeed reasonable, we computed and compared a set of spike train statistics: (a) pairwise correlation, (b) firing rate and (c) pairwise van Rossum distance. We expected that the distribution of the translated data resembles the recorded data. We can therefore measure the KL divergence for each neuron to quantify the transformation performance. Table 5 summarizes the average KL divergence of the 3 spike statistics in different distribution combinations, the distribution comparison for each statistic are available in Section H. The firing rate distributions of $\hat{X}_{\text{rec}}$ and $\hat{Y}_{\text{rec}}$ closely matched the distributions of X_{rec} and Y_{rec}, with average KL divergence of 1.1648 and 1.0697, respectively. Similarly, we can compute the pairwise correlation of each neuron with respect to the population and compare the distribution between translated and recorded data. $X_{\text{rec}} \,|\, \hat{X}_{\text{rec}}$ and $Y \,|\, \hat{Y}_{\text{rec}}$ achieved an average KL divergence value of 0.0479 and 0.0493 in the pairwise correlation comparison. Note that both were significantly better than the baseline identity model.

We expected that each neuronal activity in $\hat{X}_{\text{rec}}$ should resemble a corresponding response in X. We, therefore, computed the van Rossum distance between X and $\hat{X}_{\text{rec}}$ for each neuron across 200 test samples, which can be represented in the form of a heatmap. We observed a clear diagonal line of low-intensity values

Table 3: Test performance (in MAE) in the augmented dataset: (a) identity, `ResNet` and `AGResNet` generators trained with LSGAN objective; (b) `AGResNet` trained with different objectives; and (c) neurons ordered by original annotation, firing rate, pairwise correlation and autoencoder reconstruction loss. `1D-AGResNet`, which disregards the neuron spatial structure, is used as a baseline. Best (lowest) values are marked in bold.

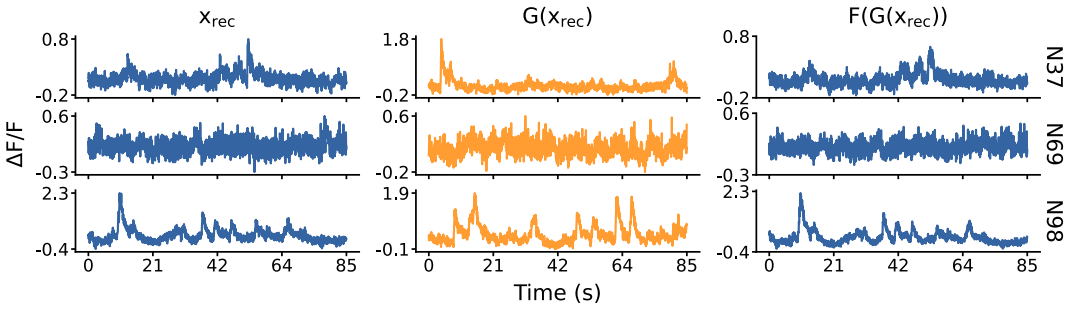
	$\lvert X_{\text{aug}} - F(Y_{\text{aug}}) \rvert$	$\lvert X_{\text{aug}} - F(G(X_{\text{aug}})) \rvert$	$\lvert Y_{\text{aug}} - G(X_{\text{aug}}) \rvert$	$\lvert Y_{\text{aug}} - G(F(Y_{\text{aug}})) \rvert$
(a) LSGAN objective against different model architectures				
identity	0.4234 ± 0.0172	**0**	0.4234 ± 0.0172	**0**
`ResNet`	0.1617 ± 0.0071	0.1173 ± 0.0043	0.3743 ± 0.0391	0.1247 ± 0.0067
`AGResNet`	$\mathbf{0.1508 \pm 0.0089}$	0.1107 ± 0.0051	$\mathbf{0.2520 \pm 0.0262}$	0.1467 ± 0.0084
(b) `AGResNet` architecture against different objectives				
GAN	0.1611 ± 0.0063	0.0948 ± 0.0069	0.2513 ± 0.0350	0.1491 ± 0.0050
LSGAN	$\mathbf{0.1508 \pm 0.0089}$	0.1107 ± 0.0051	$\mathbf{0.2520 \pm 0.0262}$	0.1467 ± 0.0084
WGANGP	0.2381 ± 0.0123	0.1600 ± 0.0098	0.3186 ± 0.0096	0.1960 ± 0.0093
DRAGAN	0.3832 ± 0.0115	$\mathbf{0.0434 \pm 0.0021}$	0.4012 ± 0.0207	$\mathbf{0.0568 \pm 0.0027}$
(c) `AGResNet` and LSGAN objective against different neuron ordering				
`1D-AGResNet`	0.2724 ± 0.0101	0.1878 ± 0.0115	0.3151 ± 0.0445	0.1655 ± 0.0115
original	0.1508 ± 0.0089	0.1107 ± 0.0051	0.2520 ± 0.0262	0.1467 ± 0.0084
firing rate	0.1578 ± 0.0079	0.0722 ± 0.0044	0.1304 ± 0.0306	0.0842 ± 0.0036
correlation	0.1556 ± 0.0044	0.0852 ± 0.0042	0.1369 ± 0.0209	0.0930 ± 0.0034
autoencoder	$\mathbf{0.1433 \pm 0.0083}$	$\mathbf{0.0639 \pm 0.0032}$	$\mathbf{0.1227 \pm 0.0135}$	$\mathbf{0.0671 \pm 0.0030}$

Table 4: Cycle-consistent and identity loss in the test set, where neurons were ordered by 1) original annotation, 2) firing rate 3) pairwise correlation, 4) `AE` reconstruction loss and 5) 1D variant of `AGResNet` (`1D-AGResNet`) where all spatial information of the neurons is disregarded. The `AGResNet` architecture was used for G and F and was optimized with LSGAN objectives. The lowest loss in each category is marked in bold. For reference, $|X - Y| = 0.3674 \pm 0.0236$ in the test set.

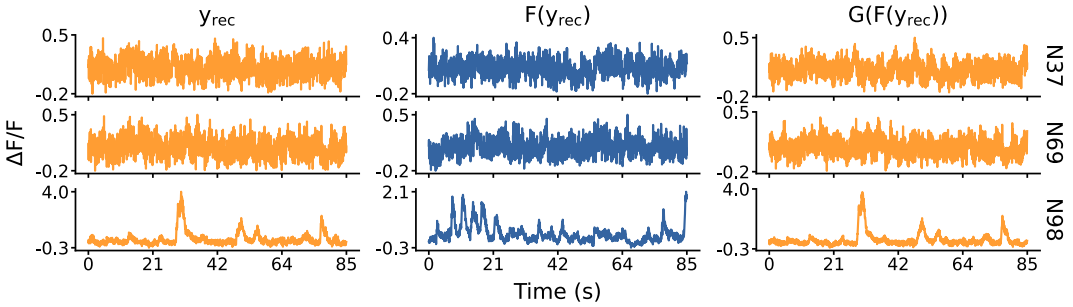
Order	$\lvert X_{\text{rec}} - F(G(X_{\text{rec}})) \rvert$	$\lvert X_{\text{rec}} - F(X_{\text{rec}}) \rvert$	$\lvert Y_{\text{rec}} - G(F(Y_{\text{rec}})) \rvert$	$\lvert Y_{\text{rec}} - G(Y_{\text{rec}}) \rvert$
`1D-AGResNet`	0.1806 ± 0.0077	0.1502 ± 0.0064	0.1811 ± 0.0163	0.1463 ± 0.0149
original	0.0874 ± 0.0037	0.0123 ± 0.0015	0.0766 ± 0.0025	0.0101 ± 0.0010
firing rate	0.0760 ± 0.0030	0.0108 ± 0.0013	0.0752 ± 0.0028	0.0070 ± 0.0005
correlation	0.0778 ± 0.0028	0.0111 ± 0.0012	0.0757 ± 0.0024	0.0089 ± 0.0022
autoencoder	$\mathbf{0.0733 \pm 0.0025}$	$\mathbf{0.0101 \pm 0.0012}$	$\mathbf{0.0737 \pm 0.0027}$	$\mathbf{0.0069 \pm 0.0007}$

in the heatmaps for most neurons (e.g. Figure H.3 and H.4 for G and F). To summarize the spike similarity results, we also compared KL divergence of the pairwise van Rossum distance distributions, which yielded divergence values of 0.2387 and 0.3031 for $X_{\text{rec}} \mid \hat{X}_{\text{rec}}$ and $Y \mid \hat{Y}_{\text{rec}}$. The spike statistics indicated that the generators can indeed learn the transformation from pre-learning and post-learning activities, and vice-versa, closely matching the first and second-order statistics of the recorded data. We additionally fitted the models on recordings from the other mice and obtained similar results, which are shown in Section I, J and K.

In the previous section, we were able to identify and interpret the learned features in a relatively straightforward manner due to the systematic augmentation we introduced into the data. However, visualizing and interpreting the attention maps on pre- and post-learning data is more challenging as there are no obvious patterns in the inputs to anticipate. Nevertheless, we expected a higher level of activities in the V1 neurons when the mouse was about to enter or was inside of the reward zone, where the grating pattern on the virtual walls turned blank. Subsequently, the models should learn meaningful features from responses surrounding the reward zones. We first visualized the sigmoid masks in `AGResNet`. Figure 11 shows the learned attention masks of G and F superimposed on the latent inputs. When the neurons were ordered, either by firing rate or autoencoder, we observed that the generators allocate more attention toward neurons that rank higher.



(a) Forward cycle: $X_{\text{rec}} \to Y \to X_{\text{rec}}$



(b) Backward cycle: $Y_{\text{rec}} \to X_{\text{rec}} \to Y_{\text{rec}}$

Figure 10: (a) forward and (b) backward cycle of neuron 37, 69 and 98 from a randomly selected test sample. Note that, unlike the synthetic dataset, the traces presented here are not unpaired. Hence, we cannot directly compare x_{rec} with $F(y_{\text{rec}})$ nor y with $G(x)$. The transformation of the entire population is available in Figure G.1.

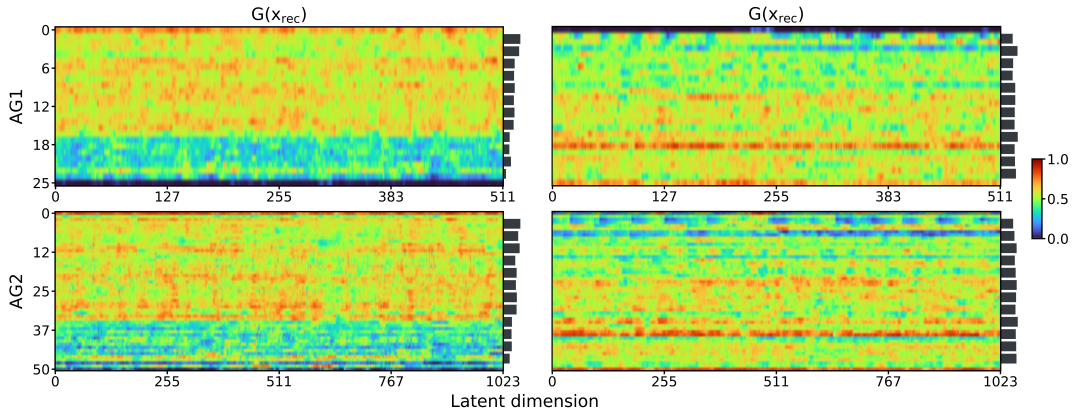


Figure 11: Attention masks AG_1 and AG_2 from generator G with (Left column) neurons sorted by `AE` reconstruction error and (Right column) no neuron ordering. The histogram to the right of each panel indicates the spatial attention intensity learned by the attention module. When the responses were ordered, a higher level of attention toward the neurons at the top (i.e. those with lower reconstruction loss) is very prominent across all attention modules, in contrast to attention distribution was uniform across neurons when neurons were not ordered. The same behavior is observed in generator F, shown in Figure G.2.

This suggests that by grouping neurons in a meaningful manner, the convolutional layers in the generators can extract relevant features more effectively as compared to when neurons were randomly ordered. The spike analysis showed that ordering neurons in a structured manner does indeed yield better results across the board. In most cases, ordering the neurons according to the reconstruction error achieved the best results.

Table 5: The average KL divergence between recorded and translated distributions in (a) pairwise correlation, (b) firing rate, and (c) pairwise van Rossum distance. We repeated the same experiments with different neuron ordering methods, in addition, we included an identity model as a baseline. Entries with the lowest value are marked in bold.

	$\mathrm{KL}(X_{\mathrm{rec}}, F(Y_{\mathrm{rec}}))$	$\mathrm{KL}(X_{\mathrm{rec}}, F(G(X_{\mathrm{rec}})))$	$\mathrm{KL}(Y_{\mathrm{rec}}, G(X_{\mathrm{rec}}))$	$\mathrm{KL}(Y_{\mathrm{rec}}, G(F(Y_{\mathrm{rec}})))$
		(a) pairwise correlation		
identity	0.0875 ± 0.0549	**0**	0.0821 ± 0.0471	**0**
`1D-AGResNet`	0.2027 ± 0.1040	0.4715 ± 0.2051	0.1901 ± 0.1003	0.4149 ± 0.2194
original	0.0552 ± 0.0419	0.0754 ± 0.0353	0.0583 ± 0.0553	0.0174 ± 0.0110
firing rate	0.0507 ± 0.0358	0.0266 ± 0.0146	0.0504 ± 0.0438	0.0267 ± 0.0176
correlation	0.0539 ± 0.0329	0.0339 ± 0.0176	0.0534 ± 0.0474	0.0205 ± 0.0133
autoencoder	$\mathbf{0.0479 \pm 0.0372}$	0.0329 ± 0.0163	$\mathbf{0.0493 \pm 0.0448}$	0.0283 ± 0.0206
		(b) firing rate		
identity	8.0705 ± 6.5500	**0**	7.7781 ± 6.7338	**0**
`1D-AGResNet`	3.5688 ± 3.8895	7.9101 ± 5.3517	3.0572 ± 3.1114	8.3185 ± 5.5950
original	1.5401 ± 1.2491	2.0442 ± 2.0936	1.8527 ± 1.3563	1.4697 ± 1.1412
firing rate	1.3402 ± 1.0450	1.2658 ± 1.0784	1.6994 ± 1.4170	1.4152 ± 1.2221
correlation	1.4006 ± 1.1079	1.5450 ± 1.0786	1.4088 ± 1.0828	1.4674 ± 1.3505
autoencoder	$\mathbf{1.1648 \pm 0.7934}$	1.4022 ± 1.2734	$\mathbf{1.0697 \pm 0.7689}$	1.2705 ± 1.1148
		(c) pairwise van Rossum distance		
identity	0.5510 ± 0.2960	**0**	0.3053 ± 0.1211	**0**
`1D-AGResNet`	0.3613 ± 0.1597	0.8045 ± 0.1846	0.3764 ± 0.1565	1.3897 ± 0.8256
original	0.2790 ± 0.2186	0.1878 ± 0.0477	0.3216 ± 0.1352	0.1581 ± 0.0664
firing rate	0.2539 ± 0.1708	0.1003 ± 0.0514	0.3080 ± 0.1173	0.1536 ± 0.0663
correlation	0.2629 ± 0.1877	0.1905 ± 0.0485	$\mathbf{0.2953 \pm 0.1230}$	0.1797 ± 0.0696
autoencoder	$\mathbf{0.2387 \pm 0.1488}$	0.1041 ± 0.0376	0.3031 ± 0.1138	0.1328 ± 0.0592

We then inspected the activation maps of the discriminators, shown in Figure 12. Similar to D_X in the synthetic dataset, we observed regions of high attention surrounding the reward zones in both D_X and D_Y. To better visualize the relationship between the area of focus learned by the model and the virtual corridor, we accumulated the per-neuron activation values over the virtual position (160cm) and result in the normalized positional activation maps shown in Figure 13. Effectively, these maps should represent the average attention learned by the models w.r.t. the visual location of the animal. The only objective the discriminators had was to distinguish if a given sample was from a particular distribution, and thus the discriminators could have learned trivial features. Instead, D_X focused on a specific group of neurons at 120-150cm, which coincides with the end of the reward zone. Moreover, D_Y learned to focus on a broader group of neurons with activation patterns that were also in alignment with the reward zone. Likewise, we could extract these positional attention maps for G and F following the same procedure, where we monitored the change in gradient in the last residual block $\mathtt{RB}_9$. Similar to the discriminators, both generators focused on responses towards the middle and end of the reward zone, with G concentrated on a very small subset of neurons. This suggests that to learn the transformation from post- to pre-learning responses, the activities the mouse exhibit as it approaches the reward zone were deemed more important by the networks.

3.3.1 Decoding performance

The positional attention maps highlighted very localized neurons in the responses with respect to the virtual position. To investigate our hypothesis that these neurons are more influential in the visual experiment and, subsequently, important for learning, we evaluated the decoding performance (i.e. position and velocity) with different subsets of neurons: (1) all neurons, (2) top-30 neurons, (3) rest of the neurons, and (4) 30 randomly selected neurons (see Section 2.4). Since G and F input X_{rec} and Y_{rec} respectively, the extracted

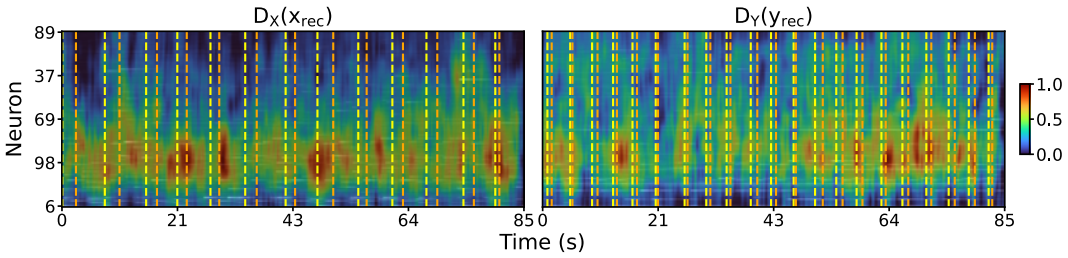


Figure 12: Activation maps of (Left) $D_X(x_{\text{rec}})$ and (Right) $D_Y(y_{\text{rec}})$ overlaid on their respective inputs. Without providing any trial-relevant information to the models, the discriminators were able to pick up information related to the reward zones (yellow and orange vertical dotted lines). Neurons in `AE` order.

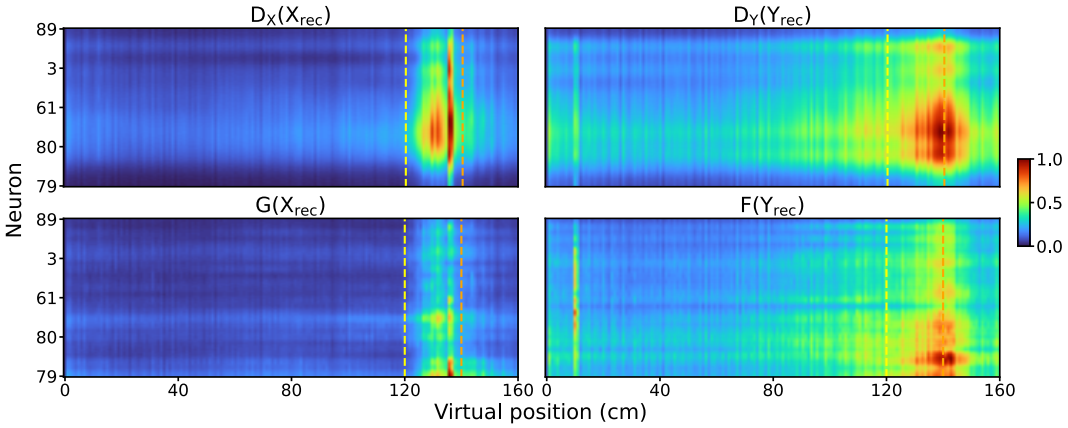


Figure 13: Positional activation maps of the discriminators and generators with respect to the virtual position. Overall, the models were focusing on a subset responses surrounding the reward zone (yellow and orange dotted lines) in their discrimination and transformation processes. Neurons in `AE` error.

positional activation maps represent different recording sessions. We therefore separately computed the top-30 neurons to decode behavior variables from Day 1 and Day 4 of the recording. Note that we are interested in the relative change in performance when providing different combinations of neurons, rather than the overall decoding accuracy. Figure 14 shows the decoding results on the two behavioral variables from Day 1 and Day 4. Overall, we observed a substantial drop in performance when the top-30 neurons were removed. In addition, the models trained with the top-30 neurons outperformed the models with 30 randomly selected neurons, except when decoding velocity in Day 1 recordings. However, for Day 1 velocity decoding, all decoders performed poorly. This shows that our framework identified neurons that are relevant for the task.

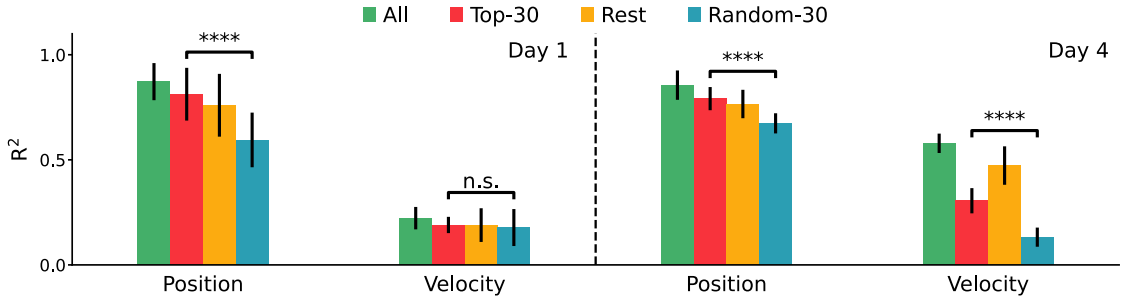


Figure 14: The decoding performance (R^2) of (a) virtual position and (b) velocity on Mouse 1 recordings when provided with: (1) all neurons, (2) top-30 neurons, (3) rest of the neurons and (4) 30 random neurons. The decoding accuracy values are listed in Table L.1.

4 Discussion

We demonstrated that the CycleGAN (Zhu et al., 2017) framework is a capable data-driven method to model the unknown transformation between pre- and post-learning neuronal responses. We evaluated our methods using two synthetic datasets (Section 3.1 and Section 3.2), with known ground-truth statistics and transformation, then on V1 recordings obtained from behaving animals (Section 3.3). With self-attention and feature-importance visualization methods, we were able to identify characteristics that the networks deemed important in their translation and discrimination process, and hence improved the interpretability of the framework. Finally, we introduced a novel and simple to implement neuron ordering method enabling more effective learning in convolutional-based networks.

Intriguingly, without providing trial information in the training process, the networks self-identified activities surrounding the reward zone in the VR experiment to be highly influential. This result aligns with our understanding from previous studies where responses in the visual cortex are shaped by the change in visual cues Pakan et al. (2018); Henschke et al. (2020). Moreover, the behavior decoding result showed that a subset of neurons, self-identified by the generators, achieved performance on par with when all neurons were provided, and performed significantly better than randomly selected neurons (see Section 3.3.1). This demonstrates the effectiveness of our method to self-identify neurons that are relevant to the task in contrast to previous works in identifying neuron contribution in decoding accuracy which requires manual iteration over neurons (Montijn et al., 2014). Interestingly, when analyzing the firing rate and pairwise correlation of this subset of neurons, we did not notice any significant distinction in their statistics as compared to the rest of the population, suggesting that the models have learned features that cannot be easily captured or identified.

Previous work in understanding or capturing the neuronal learning mechanism largely relied on linear models (Cunningham & Byron, 2014; Williams et al., 2018b; Sani et al., 2021) and non-linear latent-variable models (Pandarinath et al., 2018; Gao et al., 2016). To the best of our knowledge, this is the first work that applies an unsupervised learning method to learn the transformation in pre- to post-learning responses directly. In contrast to the aforementioned methods, our method is fully data-driven and, thus, does not require strong assumptions about the data or the task. Animal experiments are becoming increasingly large-scale and complex, in no small part due to the advancement of `in vivo` imaging technologies (e.g. Neuropixels (Steinmetz et al., 2021)). The data-driven nature of our CycleGAN framework makes it highly suitable and complementary for analyzing these latest datasets.

4.1 Limitations

The fully data-driven property of the proposed framework also comes with a number of limitations. First and foremost, as with most DNNs, this framework requires a significant amount of data, across the number of trials, the duration of each trial, and the number of neurons. For instance, we experimented with using fewer neurons, such as $W = 4$ or $W = 8$, and there was a significant decrease in performance.

Another notable constraint in our method is the fundamental one-to-one mapping limitation in the CycleGAN framework. The generators learn a deterministic mapping between the two domains and only associate each input with a single output. However, most cross-domain relationships consist of one-to-many or many-to-many mappings. More recently proposed methods, such as Augmented CycleGAN (Almahairi et al., 2018), aim to address such fundamental limitations by introducing auxiliary noise to the two distributions, and are thus able to generate outputs with variations. Nevertheless, these methods are most effective when trained in a semi-supervised manner which is not possible with our unpaired neural activity.

Lastly, a significant portion of the neuronal activity validation in Section 3.3 was performed in spike trains inferred from the recorded and generated calcium responses using Cascade (Rupprecht et al., 2021), which is a recently introduced method that has outperformed the existing model-based algorithms. However, reliable spike inference from calcium signals remains an active area of research (Theis et al., 2016). For instance, Vanwalleghem et al. (2020) demonstrated that spiking activities could be missed due to the implicit non-negativity assumption in calcium imaging data which exists in many deconvolution algorithms, including Cascade. Nonetheless, Cascade was used to deconvolve calcium signals for all datasets and thus all inferred spike trains were subject to the same bias.

4.2 Conclusion

As deep unsupervised methods have become more expressive and explainable, and neuronal activities in different learning phases from behaving animals have become more readily available, there is potential for novel insights into fundamental learning mechanisms. Future directions include sorting neurons in 2D space, as they were recorded, such that the model can take advantage of vertical and horizontal spatial information.

References

Amjad Almahairi, Sai Rajeshwar, Alessandro Sordoni, Philip Bachman, and Aaron Courville. Augmented cyclegan: Learning many-to-many mappings from unpaired data. In *International Conference on Machine Learning*, pp. 195–204. PMLR, 2018.

André Araujo, Wade Norris, and Jack Sim. Computing receptive fields of convolutional neural networks. *Distill*, 2019. doi: 10.23915/distill.00021. https://distill.pub/2019/computing-receptive-fields.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

Chensi Cao, Feng Liu, Hai Tan, Deshou Song, Wenjie Shu, Weizhong Li, Yiming Zhou, Xiaochen Bo, and Zhi Xie. Deep learning and its applications in biomedicine. *Genomics, proteomics & bioinformatics*, 16 (1):17–32, 2018.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

John P Cunningham and M Yu Byron. Dimensionality reduction for large-scale neural recordings. *Nature neuroscience*, 17(11):1500–1509, 2014.

Yuanjun Gao, Evan W Archer, Liam Paninski, and John P Cunningham. Linear dynamical neural population models through nonlinear embeddings. *Advances in neural information processing systems*, 29:163–171, 2016.

Aidan N Gomez, Sicong Huang, Ivan Zhang, Bryan M Li, Muhammad Osama, and Lukasz Kaiser. Unsupervised cipher cracking using discrete gans. *arXiv preprint arXiv:1801.04883*, 2018.

Lovedeep Gondara. Medical image denoising using convolutional denoising autoencoders. In *2016 IEEE 16th international conference on data mining workshops (ICDMW)*, pp. 241–246. IEEE, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Benjamin F Grewe, Dominik Langer, Hansjörg Kasper, Björn M Kampa, and Fritjof Helmchen. High-speed in vivo calcium imaging reveals neuronal network activity with near-millisecond precision. *Nature methods*, 7(5):399–405, 2010.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016b.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Julia U Henschke, Evelyn Dylda, Danai Katsanevaki, Nathalie Dupuy, Stephen P Currie, Theoklitos Amvrosiadis, Janelle MP Pakan, and Nathalie L Rochefort. Reward association enhances stimulus-specific representations in primary visual cortex. *Current Biology*, 2020.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

Sander W Keemink, Scott C Lowe, Janelle MP Pakan, Evelyn Dylda, Mark CW Van Rossum, and Nathalie L Rochefort. Fissa: A neuropil decontamination toolbox for calcium imaging signals. *Scientific reports*, 8 (1):1–12, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. On convergence and stability of gans. *arXiv preprint arXiv:1705.07215*, 2017.

Bryan M Li, Theoklitos Amvrosiadis, Nathalie Rochefort, and Arno Onken. Calciumgan: A generative adversarial network model for synthesising realistic calcium imaging data of neuronal populations. *arXiv preprint arXiv:2009.02707*, 2020.

Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2794–2802, 2017.

Łukasz Maziarka, Agnieszka Pocha, Jan Kaczmarczyk, Krzysztof Rataj, Tomasz Danel, and Michał Warchoł. Mol-cyclegan: a generative model for molecular optimization. *Journal of Cheminformatics*, 12(1):1–18, 2020.

Manuel Molano-Mazon, Arno Onken, Eugenio Piasini*, and Stefano Panzeri*. Synthesizing realistic neural population activity patterns using generative adversarial networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=r1VVsebAZ`.

Jorrit S Montijn, Martin Vinck, and Cyriel MA Pennartz. Population coding in mouse visual cortex: response reliability and dissociability of stimulus tuning and noise correlation. *Frontiers in computational neuroscience*, 8:58, 2014.

Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018.

Janelle MP Pakan, Stephen P Currie, Lukas Fischer, and Nathalie L Rochefort. The impact of visual cues, reward, and motor feedback on the representation of behaviorally relevant spatial locations in primary visual cortex. *Cell reports*, 24(10):2521–2528, 2018.

Chethan Pandarinath, Daniel J O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D Stavisky, Jonathan C Kao, Eric M Trautmann, Matthew T Kaufman, Stephen I Ryu, Leigh R Hochberg, et al. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature methods*, pp. 1, 2018.

Francesco Piccialli, Vittorio Di Somma, Fabio Giampaolo, Salvatore Cuomo, and Giancarlo Fortino. A survey on deep learning in medicine: Why, how and when? *Information Fusion*, 66:111–137, 2021.

Luke Yuri Prince, Shahab Bakhtiari, Colleen J Gillon, and Blake Aaron Richards. Ca{lfads}: latent factor analysis of dynamical systems in calcium imaging data, 2021. URL https://openreview.net/forum?id=J5LS3YJH7Zi.

Poornima Ramesh, Mohamad Atayi, and Jakob H. Macke. Adversarial training of neural encoding models on population spike trains. *2019 Conference on Cognitive Computational Neuroscience*, 2019.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

MCW van Rossum. A novel spike distance. *Neural computation*, 13(4):751–763, 2001.

Peter Rupprecht, Stefano Carta, Adrian Hoffmann, Mayumi Echizen, Antonin Blot, Alex C Kwan, Yang Dan, Sonja B Hofer, Kazuo Kitamura, Fritjof Helmchen, et al. A database and deep learning toolbox for noise-optimized, generalized spike inference from calcium imaging. *Nature Neuroscience*, pp. 1–14, 2021.

Virginia Rutten, Alberto Bernacchia, Maneesh Sahani, and Guillaume Hennequin. Non-reversible gaussian processes for identifying latent dynamical structure in neural data. *Advances in Neural Information Processing Systems*, 2020.

Omid G Sani, Hamidreza Abbaspourazad, Yan T Wong, Bijan Pesaran, and Maryam M Shanechi. Modeling behaviorally relevant neural dynamics enabled by preferential subspace identification. *Nature Neuroscience*, 24(1):140–149, 2021.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

Nicholas A Steinmetz, Cagatay Aydin, Anna Lebedeva, Michael Okun, Marius Pachitariu, Marius Bauza, Maxime Beau, Jai Bhagat, Claudia Böhm, Martijn Broux, et al. Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. *Science*, 372(6539), 2021.

Lucas Theis, Philipp Berens, Emmanouil Froudarakis, Jacob Reimer, Miroslav Román Rosón, Tom Baden, Thomas Euler, Andreas S Tolias, and Matthias Bethge. Benchmarking spike rate inference in population calcium imaging. *Neuron*, 90(3):471–482, 2016.

Michael Tschannen, Olivier Bachem, and Mario Lucic. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*, 2018.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Gilles Vanwalleghem, Lena Constantin, and Ethan K Scott. Calcium imaging and the curse of negativity. *Frontiers in neural circuits*, 14, 2020.

Yasi Wang, Hongxun Yao, and Sicheng Zhao. Auto-encoder based dimensionality reduction. *Neurocomputing*, 184:232–242, 2016.

Alex H. Williams, Tony Hyun Kim, Forea Wang, Saurabh Vyas, Stephen I. Ryu, Krishna V. Shenoy, Mark Schnitzer, Tamara G. Kolda, and Surya Ganguli. Unsupervised discovery of demixed, low-dimensional neural dynamics across multiple timescales through tensor component analysis. *Neuron*, 98(6):1099–1115.e8, 2018a. ISSN 0896-6273. doi: https://doi.org/10.1016/j.neuron.2018.05.015. URL https://www.sciencedirect.com/science/article/pii/S0896627318303878.

Alex H Williams, Tony Hyun Kim, Forea Wang, Saurabh Vyas, Stephen I Ryu, Krishna V Shenoy, Mark Schnitzer, Tamara G Kolda, and Surya Ganguli. Unsupervised discovery of demixed, low-dimensional neural dynamics across multiple timescales through tensor component analysis. *Neuron*, 98(6):1099–1115, 2018b.

Byron M Yu, John P Cunningham, Gopal Santhanam, Stephen I Ryu, Krishna V Shenoy, and Maneesh Sahani. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *Journal of neurophysiology*, 102(1):614–635, 2009.

Ryad Zemouri, Noureddine Zerhouni, and Daniel Racoceanu. Deep learning in the biomedical applications: Recent and future status. *Applied Sciences*, 9(8):1526, 2019.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

A Appendix

Table A.1: List of notations and their descriptions used in this manuscript.

Term	Color	Description
N		total number of segments.
H		the duration (in time-steps) of each segment.
W		the number of neurons.
X, Y		Two unpaired data distributions.
G, F		generators that learn the mapping of $X \rightarrow Y$ and $Y \rightarrow X$.
D_X, D_Y		Discriminators that learn to distinguish if samples are of distribution X and Y, respectively.
x, y		Sample from X and Y.
$\hat{x}, \hat{y}$		the intermediate transformation output $\hat{x} = G(x)$ and $\hat{y} = F(y)$.
$\bar{x}, \bar{y}$		the cycle transformation output $\bar{x} = F(\hat{y})$ and $\bar{y} = F(\hat{x})$.
X_{sim}		First group of simulated neurons.
Y_{sim}		Second group of simulated neurons.
Φ		The handcrafted augmentation function to mask out the lower left corner of the population.
X_{aug}		V1 neuron recordings obtained from the 1^{st} day of the VR experiment.
Y_{aug}		Augmented X_{aug} using the transformation function Φ.
X_{rec}		V1 neuron recordings obtained from the 1^{st} day (i.e. pre-learning) of the VR experiment.
Y_{rec}		V1 neuron recordings obtained from the 4^{th} day (i.e. post-learning) of the VR experiment.
		Start of reward zone.
		End of reward zone.
CNN		Convolution neural network.
RNN		Recurrent neural network.
`AE`		The autoencoder used to pre-sort neuron spatial order.
`ResNet`		The ResNet-like generator architecture with level-wise residual connection (Ronneberger et al., 2015).
$\texttt{US}_i, \texttt{DS}_i$		The i^{th} down-sampling and up-sampling block in `ResNet`.
$\texttt{RB}_i$		The i^{th} residual block in `ResNet`.
`AGResNet`		The proposed `ResNet` generator architecture with Additive Attention Gate module (Oktay et al., 2018).
$\texttt{AG}_i$		The i^{th} Additive Attention Gate module (layer) in `AGResNet`.
`1D-AGResNet`		`AGResNet` using 1D convolutional layers.
`MAE`		Mean absolute error.
`MSE`		Mean squared error.
`NRMSE`		Normalized root mean squared error: `NRMSE(a, b)` $= \frac{\sqrt{\texttt{MSE}(a.b)}}{\max(b)-\min(b)}$
S_C		Cosine similarity.
R^2		The coefficient of determination or R-squared.

B CycleGAN

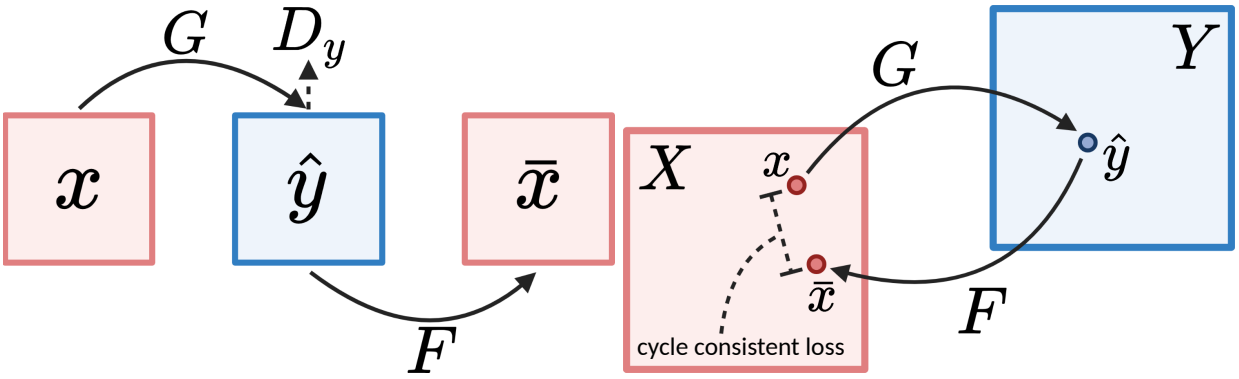


Figure B.1: Illustration of (a) the data flow and (b) the cycle-consistent loss in a forward cycle $X \to Y \to X$. G and F are generators that learn the transformation of $X \to Y$ and $Y \to X$ respectively. We first sample $x \sim X$, then apply transformation G to obtain $\hat{y} = G(x)$. To ensure $\hat{y}$ resemble distribution Y, we train discriminator D_Y to distinguish generated samples from real samples. However, even if $\hat{y}$ is of distribution Y, we cannot verify that $\hat{y}$ is the direct correspondent of x. Hence, we apply transformation F which convert $\bar{x} = G(\hat{y})$ back to domain X. If both F and G are reasonable transformations, then the cycle-consistency $|x - \bar{x}|$ should be minimal. The backward cycle $Y \to X \to X$ is a mirrored but opposite operation that run concurrently with the forward cycle. Illustration re-created from Figure 3 in Zhu et al. (2017).

Table B.1: The hyper-parameters used for each objective formulation. α_G and α_D denote the learning rates of the generators and discriminators. λ_{GP} is the gradient penalty coefficient for WGANGP and DRAGAN and c is the Gaussian variance hyper-parameter in DRAGAN. To make it compatible with the training framework in WGANGP, we added the option to train the discriminator n_D steps for every generator update.

Hyper-parameters	GAN	LSGAN	WGANGP	DRAGAN
Filters	32			
Kernel size	4			
Reduction factor	2			
Activation	LReLU			
Normalization	InstanceNorm			
Spatial Dropout	0.25			
Weight Initialization	random normal $\mathcal{N}(0, 0.02)$			
λ_{cycle}	10			
$\lambda_{\text{identity}}$	5			
λ_{GP}	N/A	N/A	10	10
c	N/A	N/A	N/A	10
n_D discriminators update	1	1	5	1
α_G	0.0001			
α_D	0.0004			
Distance Function	mean absolute error			

C Mouse information

The trial information of Mouse 2 to 4 in the VR experiment (see Section 2.6).

Table C.1: Mouse 2 performance in the experiment where 59 V1 neurons were monitored across 5 days of training.

Day	Duration	Num. trials	Avg. trial duration	Licks	Rewards
1	897.84s	61	14.72s	1038	75
2	892.29s	107	8.34s	1572	115
3	898.20s	196	4.58s	2330	204
4	897.07s	199	4.51s	1338	304
5	895.48s	122	7.34s	1069	157

Table C.2: Mouse 3 performance in the experiment where 21 V1 neurons were monitored across 5 days of training.

Day	Duration	Num. trials	Avg. trial duration	Licks	Rewards
1	895.19s	68	13.16s	919	98
2	897.11s	76	11.80s	1369	86
3	898.85s	131	6.86s	1146	173
4	895.78s	177	5.06s	1065	302
5	898.56s	190	4.73s	2334	196

Table C.3: Mouse 4 performance in the experiment where 32 V1 neurons were monitored across 5 days of training.

Day	Duration	Num. trials	Avg. trial duration	Licks	Rewards
1	895.06s	147	6.09s	1239	192
2	898.54s	300	3.00s	1024	487
3	895.91s	215	4.17s	1982	220
4	896.83s	227	3.95s	2493	230
5	897.88s	299	3.00s	1750	303

D Neuron ordering

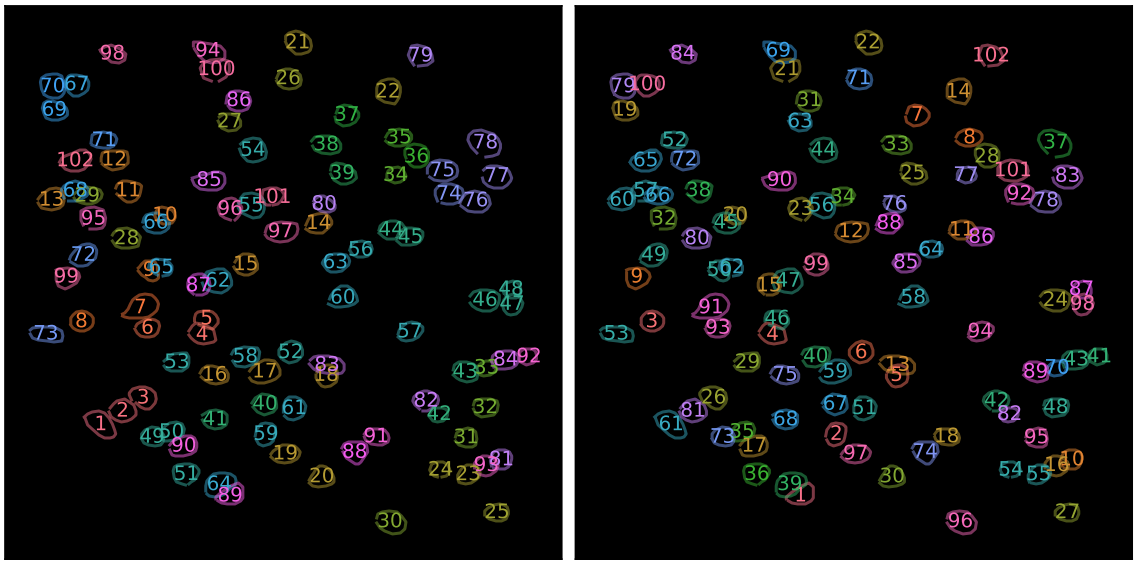


Figure D.1: Neuron ordering based on (Left) original annotation, (Right) autoencoder reconstruction loss. The original order was based on how visible the neuron was in the calcium image, hence not sorted in a particular manner. We proposed to train an autoencoder that learns to reconstruct X and Y jointly, and sort neurons based on the average reconstruction error on the validation set (see Section 2.5).

Table D.1: Mouse 1 neuron ordering based on (a) original annotation, (b) firing rate, (c) average pairwise correlation, and (d) autoencoder reconstruction loss.

Method	Order
(a) N/A	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102
(b) Firing rate	18, 14, 12, 30, 8, 15, 36, 4, 21, 19, 3, 7, 43, 33, 20, 42, 13, 6, 11, 39, 2, 22, 75, 28, 55, 100, 31, 62, 10, 67, 63, 54, 17, 40, 52, 46, 99, 88, 61, 77, 57, 34, 85, 41, 27, 98, 84, 47, 65, 73, 5, 1, 44, 101, 58, 80, 16, 29, 87, 9, 26, 83, 92, 74, 24, 45, 49, 23, 97, 48, 68, 60, 71, 76, 59, 53, 70, 89, 25, 93, 32, 56, 66, 81, 72, 94, 38, 64, 79, 82, 50, 51, 96, 90, 37, 86, 95, 91, 102, 69, 35, 78
(c) Correlation	36, 27, 46, 28, 39, 30, 42, 20, 92, 10, 18, 11, 67, 14, 4, 33, 19, 77, 75, 13, 24, 99, 8, 43, 65, 101, 63, 7, 25, 44, 12, 76, 80, 9, 47, 3, 34, 71, 87, 52, 22, 1, 85, 61, 84, 29, 45, 31, 93, 100, 5, 58, 57, 17, 74, 21, 96, 55, 82, 91, 2, 48, 6, 56, 83, 62, 49, 16, 26, 81, 97, 53, 73, 94, 89, 59, 40, 95, 23, 32, 54, 66, 98, 72, 35, 88, 15, 41, 50, 60, 90, 70, 78, 68, 69, 86, 38, 51, 64, 79, 37, 102
(d) Autoencoder	89, 59, 8, 4, 18, 52, 37, 35, 99, 81, 44, 97, 83, 22, 87, 93, 90, 91, 69, 10, 100, 21, 96, 46, 39, 3, 25, 36, 53, 20, 86, 95, 38, 101, 50, 51, 78, 11, 64, 58, 92, 82, 84, 54, 66, 5, 62, 32, 72, 9, 61, 71, 73, 24, 23, 55, 68, 60, 17, 13, 1, 65, 27, 56, 102, 29, 40, 41, 94, 33, 26, 12, 49, 88, 16, 80, 34, 76, 70, 28, 2, 42, 77, 98, 63, 45, 48, 14, 43, 85, 7, 74, 6, 57, 31, 30, 19, 47, 15, 67, 75, 79

E Simulation data

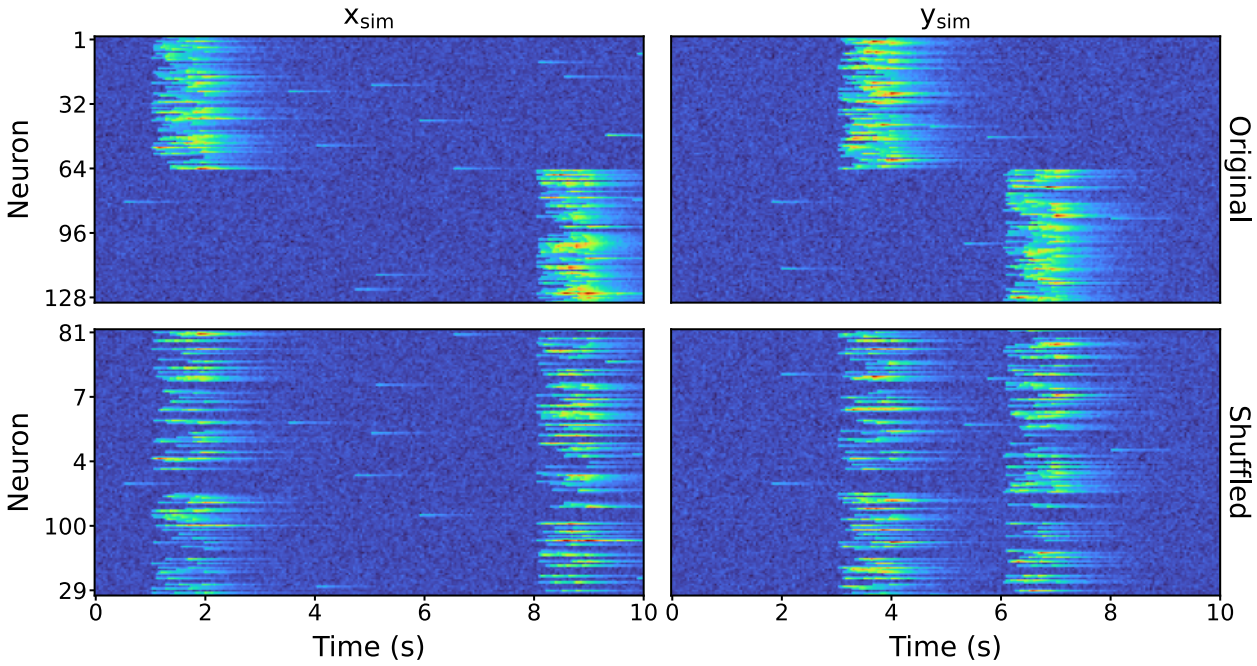


Figure E.1: The top panels show the calcium-like traces of all 128 neurons from $x_{\text{sim}} \sim X_{\text{sim}}$ and $y_{\text{sim}} \sim Y_{\text{sim}}$, and the bottom panels show the same population with neuron ordered shuffled, which are then feed into the unsupervised learning framework. The shuffling process is added to increase the difficulties for the generators to learn the transformation as the responses are less structured. `TURBO` color-map is used to improve visibility.

F Augmented data

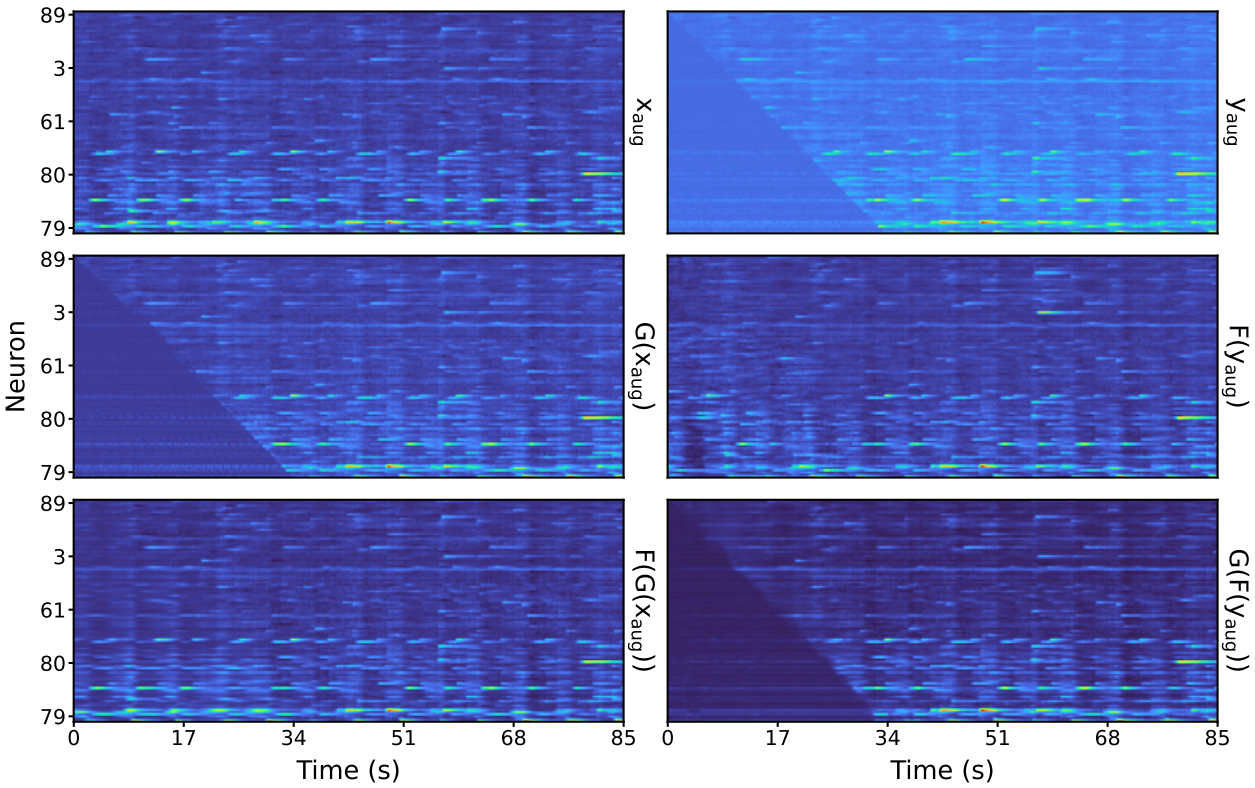


Figure F.1: The (Left column) forward and (Right column) backward cycle of the entire neuron population from a randomly selected trial, using the `AGResNet` architecture and trained with LSGAN objectives. G learns the augmentation function Φ (described in Section 2.7.2) which mask out the lower left corner of input x_{aug}, whereas F learns to recover the masked regions from unpaired data. `TURBO` color-map is used to improve visibility.

G Recorded data

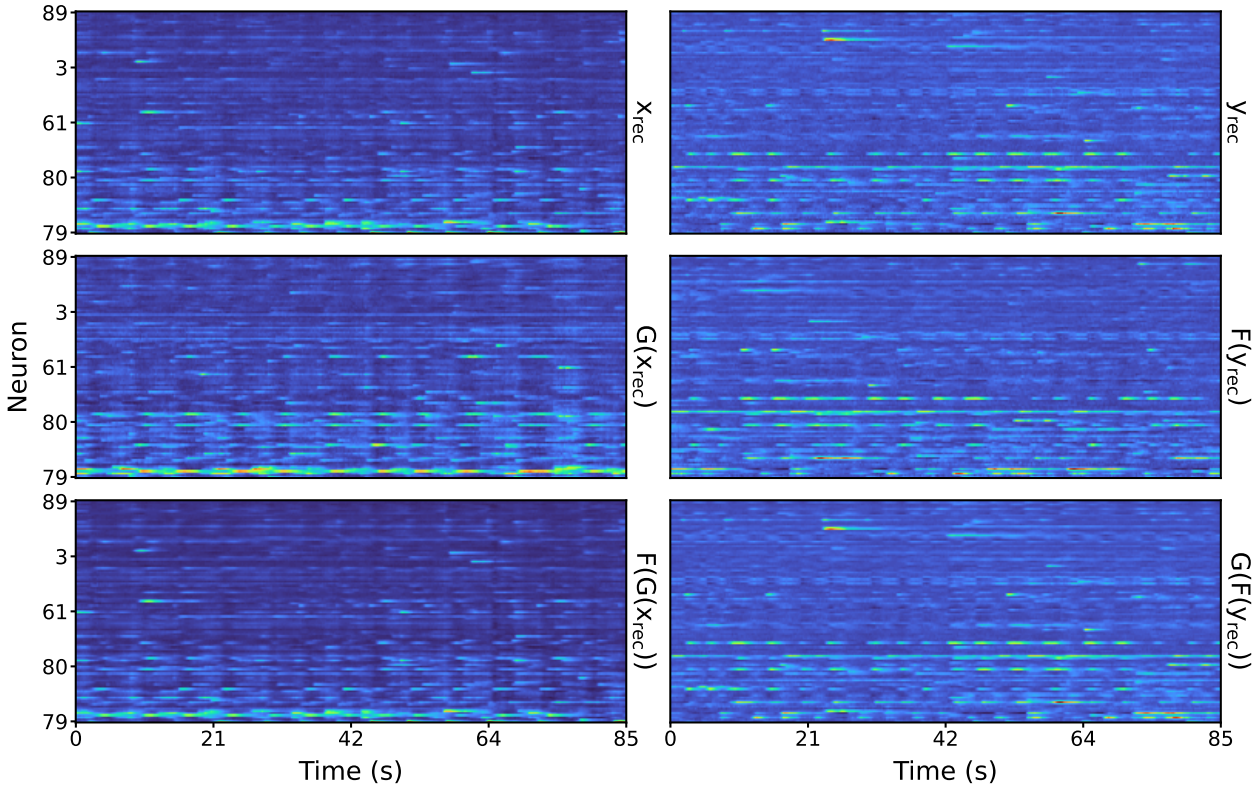


Figure G.1: The (Left column) forward and (Right column) backward cycle of the entire population from a randomly selected segment $x_{\text{rec}} \sim X_{\text{rec}}$ and $y_{\text{rec}} \sim Y_{\text{rec}}$. The model was trained with `AGResNet` generators using the LSGAN objective on the recorded dataset and neurons were in `AE` order. `TURBO` color-map is used to improve visibility.

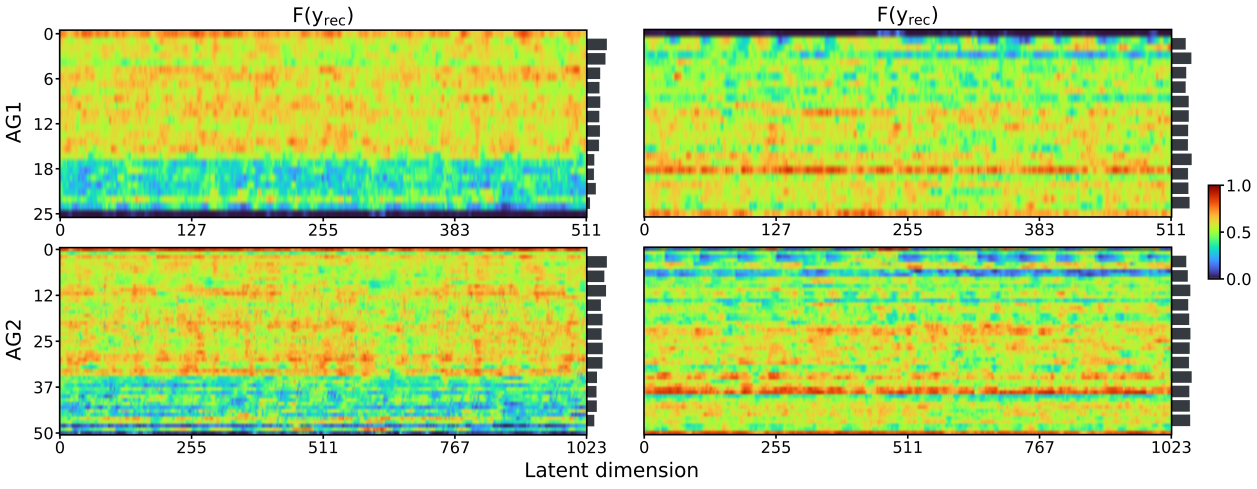


Figure G.2: Attention masks AG_1 and AG_2 from generator F with (Left column) neurons sorted by `AE` reconstruction error and (Right column) no neuron ordering. The histogram to the right of each panel indicates the spatial attention intensity learned by the attention module.

H Mouse 1 spike analysis

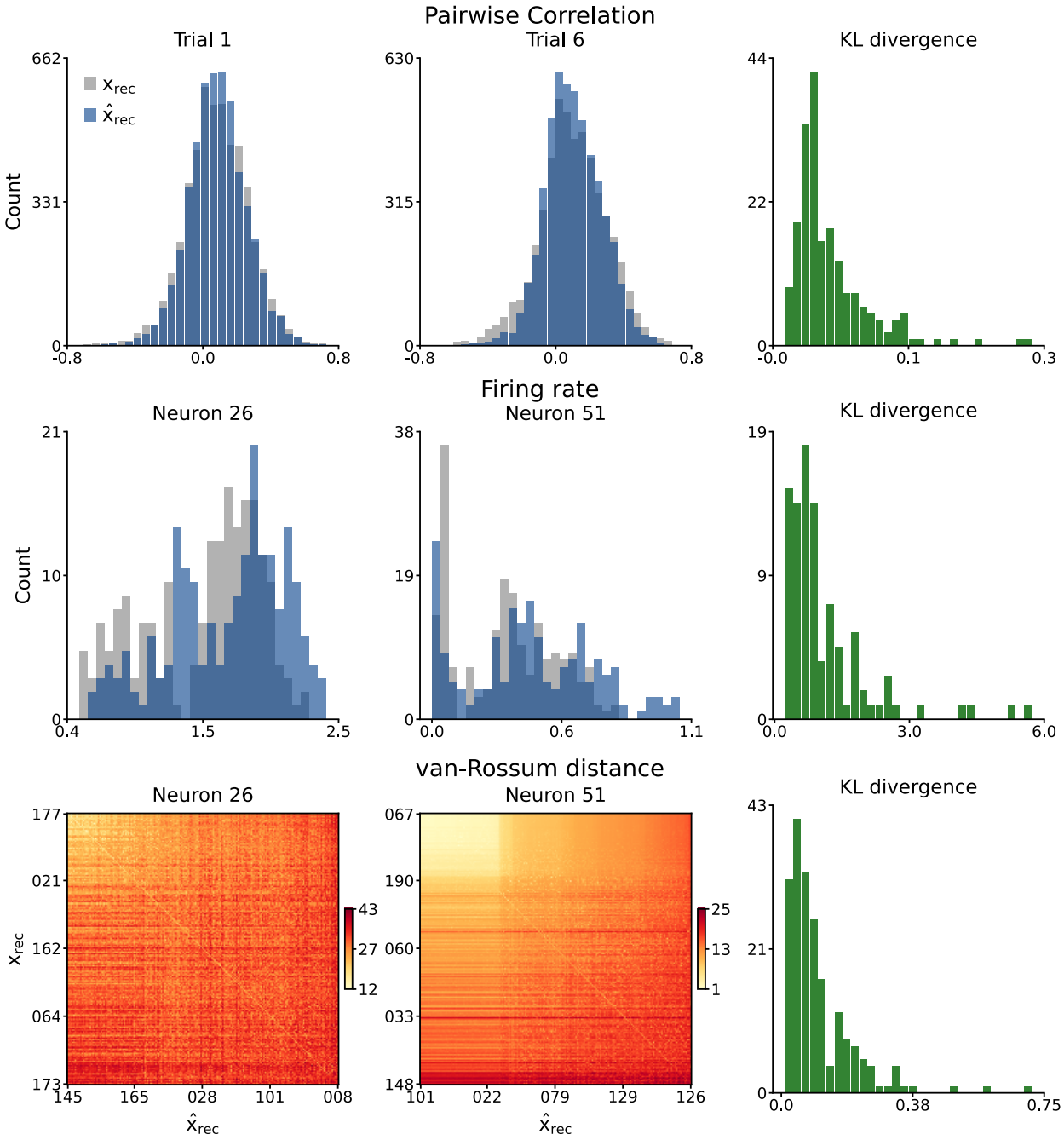


Figure H.1: Spike statistics comparison between X_{rec} and $\hat{X}_{\text{rec}} = F(Y_{\text{rec}})$ of (Top row) pairwise correlation from 2 randomly selected samples, (Middle row) firing rate of 2 randomly selected neurons and (Bottom row) van Rossum distance of 2 randomly selected segments. The 3$^{\text{rd}}$ column shows the KL divergence of each metric and Table 5 shows the mean and standard deviation of the KL divergence comparisons. Note, neurons were ordered by autoencoder reconstruction loss and the ground-truth X_{rec} is in gray color.

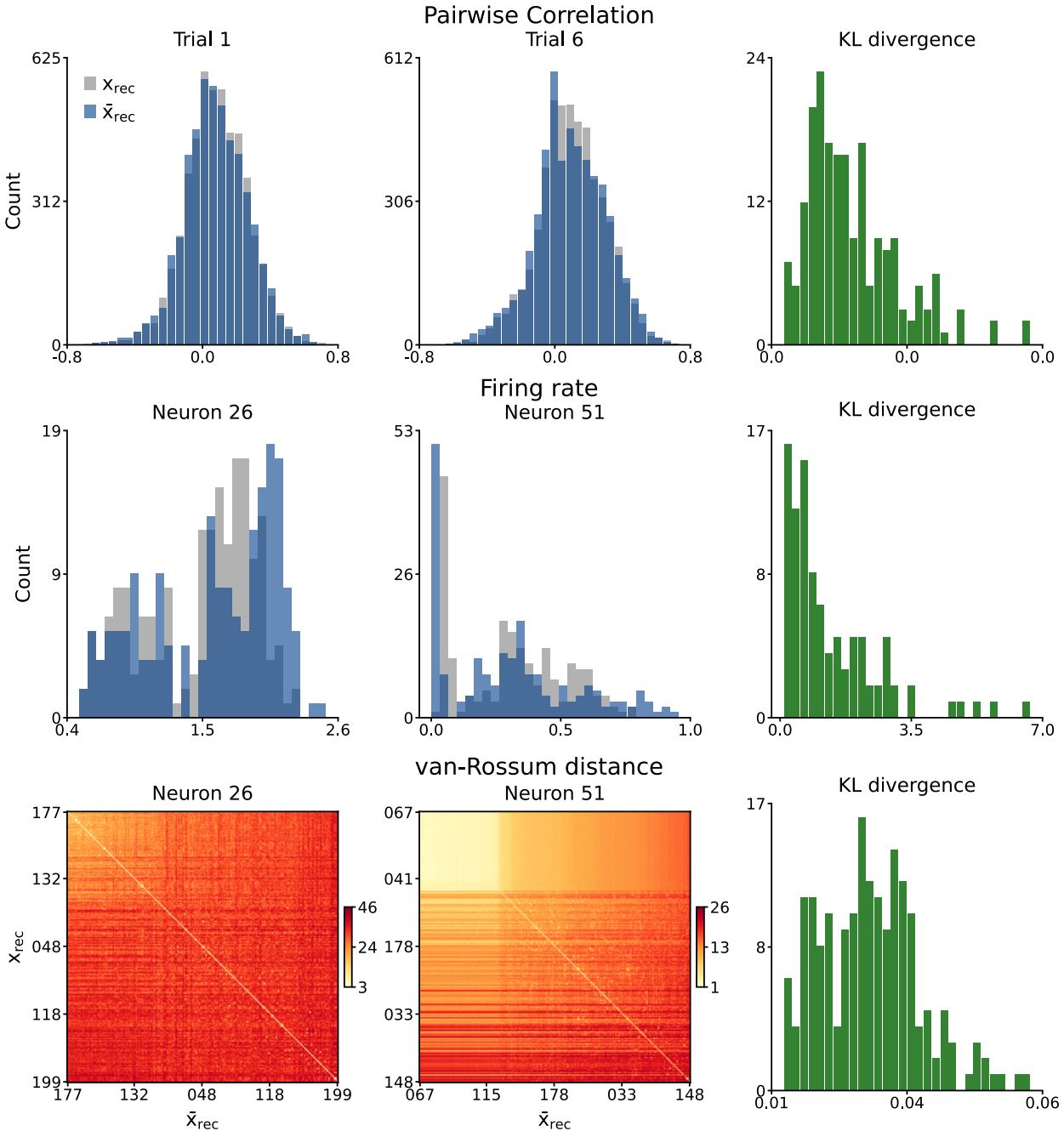


Figure H.2: Spike statistics comparison between X_{rec} and $\bar{X}_{\text{rec}} = F(G(X_{\text{rec}}))$ of (Top row) pairwise correlation from 2 randomly selected samples, (Middle row) firing rate of 2 randomly selected neurons and (Bottom row) van Rossum distance of 2 randomly selected segments. The 3$^{\text{rd}}$ column shows the KL divergence of each metric and Table 5 shows the mean and standard deviation of the KL divergence comparisons. Note, neurons were ordered by autoencoder reconstruction loss and the ground-truth X_{rec} is in gray color.

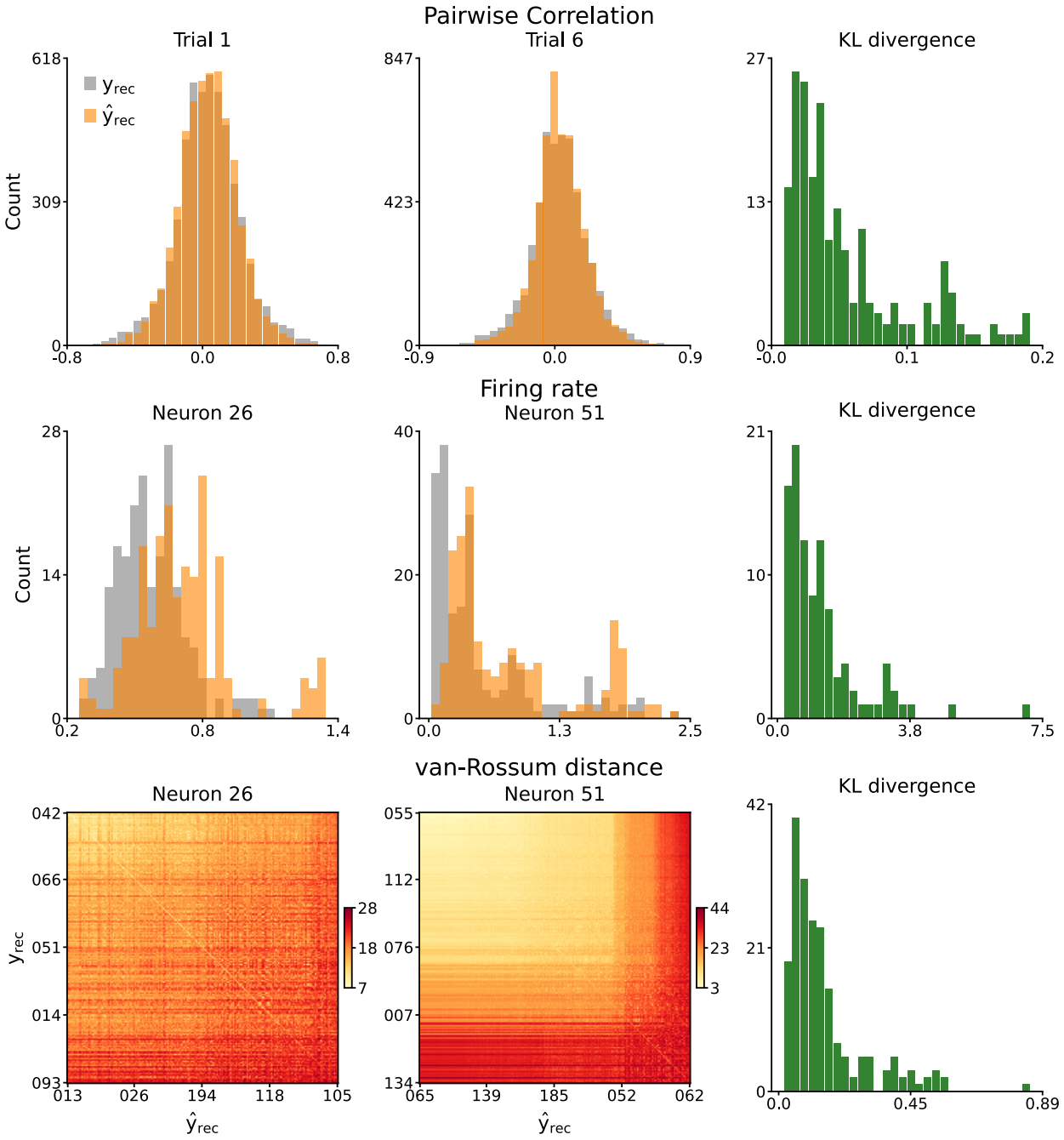


Figure H.3: Spike statistics comparison between Y_{rec} and $\hat{Y}_{\text{rec}} = G(X_{\text{rec}})$ of (Top row) pairwise correlation from 2 randomly selected samples, (Middle row) firing rate of 2 randomly selected neurons and (Bottom row) van Rossum distance of 2 randomly selected segments. The 3$^{\text{rd}}$ column shows the KL divergence of each metric and Table 5 shows the mean and standard deviation of the KL divergence comparisons. Note, neurons were ordered by autoencoder reconstruction loss and the ground-truth Y_{rec} is in gray color.

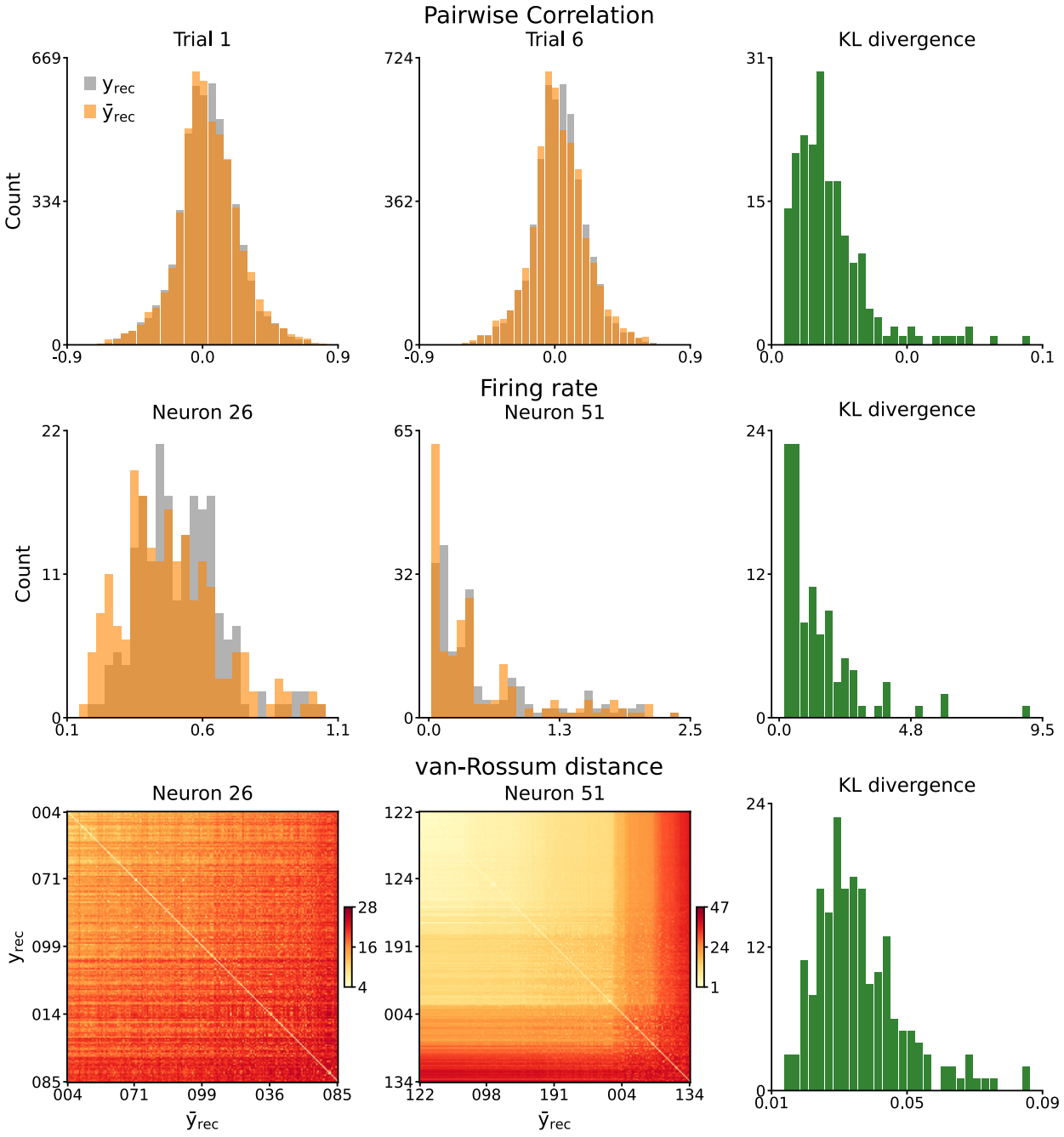


Figure H.4: Spike statistics comparison between Y_{rec} and $\bar{Y}_{\text{rec}} = F(G(Y_{\text{rec}}))$ of (Top row) pairwise correlation from 2 randomly selected samples, (Middle row) firing rate of 2 randomly selected neurons and (Bottom row) van Rossum distance of 2 randomly selected segments. The 3$^{\text{rd}}$ column shows the KL divergence of each metric and Table 5 shows the mean and standard deviation of the KL divergence comparisons. Note, neurons were ordered by autoencoder reconstruction loss and the ground-truth Y_{rec} is in gray color.

I Mouse 2 results

Table I.1: Cycle-consistent and identity loss of `AGResNet` on Mouse 2 recordings, where neurons were ordered by 1) original annotation, 2) firing rate and 3) autoencoder reconstruction loss. For reference, $|X - Y| = 0.6057 \pm 0.1146$ in the test set. The lowest loss in each category is marked in bold.

Order	$\lvert X - F(G(X))\rvert$	$\lvert X - F(X)\rvert$	$\lvert Y - G(F(Y))\rvert$	$\lvert Y - G(Y)\rvert$
none	0.5875 ± 0.1050	0.1292 ± 0.0168	0.4416 ± 0.0763	0.0923 ± 0.0064
firing rate	0.5794 ± 0.1055	0.1276 ± 0.0152	0.4396 ± 0.0793	0.0894 ± 0.0048
autoencoder	$\mathbf{0.5692 \pm 0.1008}$	$\mathbf{0.1030 \pm 0.0099}$	$\mathbf{0.4378 \pm 0.0769}$	$\mathbf{0.0101 \pm 0.0018}$

Table I.2: The average KL divergence between generated and recorded distributions of Mouse 2 in (a) pairwise correlation, (b) firing rate, and (c) population pairwise van Rossum distance. We compare `AGResNet` results with different neuron ordering including 1) original annotation, 2) firing rate and 3) autoencoder reconstruction loss. Note that we added the identity model (first row of each sub-table) as a baseline where we should obtain perfect cycle reconstruction. Entries with the lowest value are marked in bold.

	$\mathrm{KL}(X, F(Y))$	$\mathrm{KL}(X, F(G(X)))$	$\mathrm{KL}(Y, G(X))$	$\mathrm{KL}(Y, G(F(Y)))$
(a) pairwise correlation				
Identity	0.6528 ± 0.4980	**0**	0.4583 ± 0.4366	**0**
N/A	0.5523 ± 0.4251	0.1617 ± 0.0715	0.1212 ± 0.0833	0.0499 ± 0.0266
firing rate	0.5639 ± 0.4679	0.1951 ± 0.1031	$\mathbf{0.1126 \pm 0.0831}$	0.0399 ± 0.0231
autoencoder	$\mathbf{0.5209 \pm 0.5554}$	0.0582 ± 0.0361	0.1231 ± 0.0988	0.0352 ± 0.0228
(b) firing rate				
Identity	8.3096 ± 6.1580	**0**	5.5783 ± 5.8451	**0**
N/A	1.2881 ± 1.1147	2.5786 ± 2.7222	1.5782 ± 1.2217	1.6722 ± 1.3286
firing rate	1.2181 ± 0.9909	2.4912 ± 2.5037	1.3656 ± 1.1475	1.1767 ± 1.0625
autoencoder	$\mathbf{0.8087 \pm 0.5764}$	1.1326 ± 1.3149	$\mathbf{1.2521 \pm 0.9649}$	1.0592 ± 1.0722
(c) pairwise van Rossum distance				
Identity	1.3894 ± 2.0529	**0**	1.1240 ± 1.5159	**0**
N/A	1.3392 ± 1.6653	0.5782 ± 0.9743	0.6043 ± 0.5250	0.2497 ± 0.2443
firing rate	1.2464 ± 1.7505	0.5946 ± 0.9352	0.5638 ± 0.4181	0.1977 ± 0.1234
autoencoder	$\mathbf{0.6946 \pm 0.5687}$	0.1996 ± 0.3232	$\mathbf{0.5287 \pm 0.3897}$	0.1775 ± 0.0959

J Mouse 3 results

Table J.1: Cycle-consistent and identity loss of `AGResNet` on Mouse 3 recordings, where neurons were ordered by 1) original annotation, 2) firing rate and 3) autoencoder reconstruction loss. For reference, $|X - Y| = 0.4764 \pm 0.1520$ in the test set. The lowest loss in each category is marked in bold.

Order	$\lvert X - F(G(X))\rvert$	$\lvert X - F(X)\rvert$	$\lvert Y - G(F(Y))\rvert$	$\lvert Y - G(Y)\rvert$
none	0.2684 ± 0.0290	0.0656 ± 0.0037	0.3229 ± 0.0476	0.0796 ± 0.0047
firing rate	0.2679 ± 0.0309	0.0585 ± 0.0034	$\mathbf{0.3192 \pm 0.0477}$	0.0777 ± 0.0043
autoencoder	$\mathbf{0.2677 \pm 0.0282}$	$\mathbf{0.0554 \pm 0.0023}$	0.3199 ± 0.0487	$\mathbf{0.0672 \pm 0.0034}$

Table J.2: The average KL divergence between generated and recorded distributions of Mouse 3 in (a) pairwise correlation, (b) firing rate, and (c) population pairwise van Rossum distance. We compare `AGResNet` results with different neuron ordering including 1) original annotation, 2) firing rate and 3) autoencoder reconstruction loss. Note that we added the identity model (first row of each sub-table) as a baseline comparison and should obtain perfect cycle reconstruction. Entries with the lowest value are marked in bold.

	$\mathrm{KL}(X, F(Y))$	$\mathrm{KL}(X, F(G(X)))$	$\mathrm{KL}(Y, G(X))$	$\mathrm{KL}(Y, G(F(Y)))$
(a) pairwise correlation				
Identity	1.0188 ± 0.5731	**0**	0.7363 ± 0.3732	**0**
N/A	0.5361 ± 0.2817	0.5678 ± 0.3145	0.6975 ± 0.2202	0.7381 ± 0.2977
firing rate	$\mathbf{0.5021 \pm 0.2596}$	0.5184 ± 0.2536	0.6281 ± 0.2830	0.6616 ± 0.2850
autoencoder	0.5140 ± 0.2538	0.4751 ± 0.2421	$\mathbf{0.6137 \pm 0.2997}$	0.4625 ± 0.2443
(b) firing rate				
Identity	12.2077 ± 7.3556	**0**	12.4075 ± 7.3156	**0**
N/A	1.0164 ± 0.7129	1.8203 ± 1.9280	1.2904 ± 0.9448	1.4786 ± 1.4374
firing rate	0.9371 ± 0.6735	1.7893 ± 2.5419	$\mathbf{1.0712 \pm 0.7793}$	1.2805 ± 1.5136
autoencoder	$\mathbf{0.8936 \pm 0.5655}$	1.1152 ± 0.6797	1.2114 ± 0.7281	0.6928 ± 0.4643
(c) pairwise van Rossum distance				
Identity	4.2704 ± 2.0834	**0**	4.9623 ± 1.4393	**0**
N/A	3.0412 ± 1.8467	2.0246 ± 1.3422	4.6059 ± 2.0664	3.0293 ± 1.5854
firing rate	2.9009 ± 1.7587	1.6458 ± 1.2375	4.1910 ± 1.7950	2.8613 ± 1.7788
autoencoder	$\mathbf{2.8383 \pm 1.5942}$	1.4747 ± 1.1150	$\mathbf{3.9709 \pm 1.7732}$	1.4767 ± 1.0195

K Mouse 4 results

Table K.1: Cycle-consistent and identity loss of `AGResNet` on Mouse 4 recordings, where neurons were ordered by 1) original annotation, 2) firing rate, and 3) autoencoder reconstruction loss. For reference, $|X - Y| = 0.4383 \pm 0.2354$ in the test set. The lowest loss in each category is marked in bold.

Order	$\lvert X - F(G(X))\rvert$	$\lvert X - F(X)\rvert$	$\lvert Y - G(F(Y))\rvert$	$\lvert Y - G(Y)\rvert$
none	0.2538 ± 0.0399	0.0443 ± 0.0015	0.2403 ± 0.0395	0.0808 ± 0.0061
firing rate	0.2511 ± 0.0389	$\mathbf{0.0376 \pm 0.0015}$	0.2388 ± 0.0406	0.0764 ± 0.0067
autoencoder	$\mathbf{0.2489 \pm 0.0381}$	0.0382 ± 0.0012	$\mathbf{0.2367 \pm 0.0396}$	$\mathbf{0.0764 \pm 0.0053}$

Table K.2: The average KL divergence between generated and recorded distributions of Mouse 4 in (a) pairwise correlation, (b) firing rate and (c) population pairwise van Rossum distance. We compare `AGResNet` results with different neuron ordering including 1) original annotation, 2) firing rate and 3) autoencoder reconstruction loss. Note that we added the identity model (first row of each sub-table) as a baseline comparison and should obtain perfect cycle reconstruction. Entries with the lowest value are marked in bold.

	$\mathrm{KL}(X, F(Y))$	$\mathrm{KL}(X, F(G(X)))$	$\mathrm{KL}(Y, G(X))$	$\mathrm{KL}(Y, G(F(Y)))$
(a) pairwise correlation				
Identity	0.3724 ± 0.2169	**0**	0.5124 ± 0.3238	**0**
N/A	0.2849 ± 0.1552	0.1735 ± 0.0918	0.3536 ± 0.2541	0.5750 ± 0.2883
firing rate	0.2482 ± 0.1502	0.1478 ± 0.0848	0.3482 ± 0.2561	0.5471 ± 0.2577
autoencoder	$\mathbf{0.2096 \pm 0.1155}$	0.1587 ± 0.0867	$\mathbf{0.3460 \pm 0.2568}$	0.4795 ± 0.2457
(b) firing rate				
Identity	5.8031 ± 4.8030	**0**	5.1383 ± 5.4684	**0**
N/A	1.3062 ± 1.0097	0.6034 ± 0.6294	1.4253 ± 1.5599	2.9196 ± 3.1077
firing rate	1.0818 ± 0.9274	0.5480 ± 0.5043	1.2120 ± 1.2971	2.8206 ± 2.7266
autoencoder	$\mathbf{1.0564 \pm 1.1415}$	0.5474 ± 0.5223	$\mathbf{1.1570 \pm 1.0830}$	2.1015 ± 2.2399
(c) pairwise van Rossum distance				
Identity	2.2670 ± 1.2707	**0**	2.8134 ± 1.5536	**0**
N/A	1.8698 ± 1.1525	0.5625 ± 0.4399	2.4011 ± 1.4879	3.3849 ± 1.7608
firing rate	1.5416 ± 0.9327	0.3931 ± 0.2821	$\mathbf{2.1379 \pm 1.4338}$	3.3865 ± 1.9320
autoencoder	$\mathbf{1.3246 \pm 0.8537}$	0.4578 ± 0.3639	2.2134 ± 1.3838	2.6537 ± 1.6526

L Decoding performance

Table L.1: The decoding performances (R^2) in (a) virtual position and (b) velocity on Mouse 1 recordings from Day 1 and Day 4 of the VR experiment. We trained RNN regression models on the calcium responses of (1) all 102 neurons, (2) top-30 neurons, (3) the rest of the neurons, and (4) 30 randomly selected neurons. We fit the regression models 20 times using different random seeds and compute the p-value between (2) and (4). As the GradCAM activation map from G and F are extracted with respect to inputs X_{rec} and Y_{rec}, and thus we selected the (2) top-30 neurons for Day 1 and Day 4 separately, each according to the positional activation maps from the two generators (see Figure 13). The best result for each decoding task is shown in bold.

	(1) all	(2) top-30	(3) rest	(4) random-30	p-value
			(a) Virtual position		
Day 1	0.8721 ± 0.0883	0.8123 ± 0.1255	0.7597 ± 0.1494	0.5947 ± 0.1300	6.0858×10^{-6} (****)
Day 4	0.8555 ± 0.0698	0.7912 ± 0.0551	0.7657 ± 0.0676	0.6736 ± 0.0479	2.3413×10^{-8} (****)
			(b) Velocity		
Day 1	0.2225 ± 0.0533	0.1900 ± 0.0387	0.1894 ± 0.0802	0.1778 ± 0.0879	0.5827 (n.s.)
Day 4	0.5786 ± 0.0461	0.3053 ± 0.0600	0.4728 ± 0.0915	0.1320 ± 0.0455	3.1032×10^{-12} (****)