Neuronal Learning Analysis using Cycle-Consistent Adversarial Networks

Bryan M. Li

Master of Science by Research

Institute for Adaptive and Neural Computation
School of Informatics
University of Edinburgh

2021

Abstract

Understanding how cortical responses reshape over the course of learning has been the central theme of computational neuroscience. Thanks to the recent advances in neural imaging technologies, experimentalists are able to obtain high-quality recordings from hundreds of neurons over multiple days or even weeks. However, the complexity and dimensionality of population responses pose significant challenges for analysis. Existing methods of studying neuronal adaptation and learning often impose strong assumptions on the data or model, resulting in biased descriptions that do not generalize. In this work, we explore the use of a special type of deep generative model called – cycle-consistent adversarial networks (CycleGAN) to learn the unknown mapping between pre-learning and post-learning *in vivo* cellular activities. To do so, we develop a framework to preprocess, train and evaluate calcium signals. We first test our framework on a synthetic dataset with ground-truth transformation. Subsequently, we applied it to neuronal activity from rodent visual cortex across different days that mice transition from novice to expert-level performance on the experimental task. We compare our model performance in both generated calcium imaging signals and their inferred spike trains. To maximize the performance of our model, we derive a novel approach to pre-sort neurons such that convolutional-based deep neural networks can take advantage of the spatial information that exists in neuronal activities. In addition, we incorporate a number of model visualization methods to improve the explainability of our work and also gain insights into the learning process as manifested in the cellular activities. Together, our results demonstrate that analyzing neuronal learning processes with the data-driven deep unsupervised method holds tremendous potential.

Acknowledgements

My most sincere gratitude to my supervisors, Dr. Arno Onken and Dr. Nathalie Rochefort, for their tireless guidance throughout this research project and my postgraduate studies.

I thank Dr. Nina Kudryashova and Theoklitos Amvrosiadis for their assistance in understanding the calcium imaging data and also their meaningful insights from the neuroscience perspective.

I would like to thank Lazaros Mitskopoulos and Filippo Corponi for their ideas and suggestions in improving this work.

In addition, I would like to thank the UKRI Biomedical AI CDT administration team. For their effort in making sure this school year runs as smoothly as possible despite the looming pandemic.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Bryan M. Li*)

Table of Contents

Chapter 1

Introduction

One of the main objectives in computational neuroscience is to study the dynamics of neural processing and how neural activity reshape in the course of learning. One major hurdle was the difficulty in obtaining high-quality neural recordings of the same set of neurons across multiple experiments, though such limitation in recording techniques has seen tremendous improvements in recent years. Nevertheless, strong assumptions in the data or experiment are often required in order to analyze the high-dimensional neuronal activity recordings. Therefore, a data-driven method to interpret the circuit dynamics in learning is highly desirable. Moreover, understanding the learning process in the brain can also provide meaningful insights into other fields such as health science and machine learning. For instance, it would allow medical practitioners to monitor changes in cortical function for a given drug hence providing better treatment for pathologies that are associated with learning deficits [12, 106]. Moreover, despite the rapid development in deep learning in recent years, most artificial neural networks (ANNs) require a tremendous amount of data to learn from and often generalize poorly for unseen data [27]. Zero-shot and one-shot learning are areas that attempt to address the said issue, though, these methods are still quite far from human learning capabilities [6, 115, 117, 129].

With the advent of modern neural imaging technologies, it is now possible to monitor a large population of neurons over days or even weeks [104, 120], thus allowing practitioners to obtain *in vivo* recordings from the same set of neurons across different learning stages. Substantial effort has been put into extracting interpretable and unbiased descriptions of how cortical responses improve due to experience. Churchland et al.

[20] analyzed the per-trial firing rate from the motor cortex recordings to learn a latent representation of the neural dynamics over multiple trials. Driscoll et al. [28] used a generalized linear model to model the relationship between neuronal activity and the behaviour task. Williams et al. [121] applied tensor component analysis (TCA) in multi-timescale recordings to identify a set of low-dimensional factors that can describe within-trial dynamics as well as across-trial structure. Nevertheless, many of these methods have strong assumptions in the modelling technique or the data, such as the linearity assumption in TCA. Therefore, making sense of the unknown mapping between pre-learning and post-learning neural activity remains a significant challenge.

Thanks to their ability to self-identity and self-learn features from complex data, deep neural networks (DNNs) have seen tremendous success in many biomedical applications [11, 73, 88, 127]. Specifically, deep generative networks have shown promising results in analyzing and synthesizing neuronal activities in recent years. Pandarinath et al. [83] developed a variational autoencoder (VAE) to learn latent dynamics from single-trial spiking activities and Prince et al. [89] extended the framework work with calcium imaging data. Molano-Mazon et al. [75] demonstrated that generative adversarial networks (GANs) can synthesize binary spike trains from a small number of neurons in the salamander retina and match the low-level statistics of spiking activities observed in the recorded data. Li et al. [62] proposed a GAN framework that models the continuous calcium fluorescent signals instead of binary spike trains. They fitted the model with over a hundred neurons from the primary visual cortex and showed that the generate samples that resembled the underlying distributions of the recorded data.

Furthermore, when using these generative models in certain combinations, it is even possible to learn the transformation between two otherwise unpaired distributions [55, 125, 132]. Cycle-consistent adversarial networks (CycleGAN [132]), which utilize two GANs in conjunction to learn the mapping between two unaligned domains via cycle consistency, is one such prominent unsupervised method. The CycleGAN framework was originally designed for unsupervised image-to-image translation though it has been extended to other domains, such as natural language translation [34] and molecular optimization [71].

In this work, we explore the use of cycle-consistent adversarial networks to learn the unknown mapping between pre-learning and post-learning neuronal activities. In other words, given the neural recordings of a novice animal, can we translate the neuronal activities that correspond to the animal with expert-level performance and

vice versa? To this end, we derive a standardized procedure to train, evaluate and interpret the CycleGAN framework. To improve the explainability of our work, we employ the feature-importance visualization method – GradCAM [98] to interpret the learned features by the deep neural networks. In addition, we have introduced a ResNet-based [43] self-attention architecture – AGResNet. The self-attention mechanism allows the network to self-adjust its level of "attention" at different latent dimensions in the model. Such formulation has two key benefits: 1) improve model performance; 2) allow direct inspection of where and what the model deemed important w.r.t the input. To quantify the capability of the unsupervised learning method, we evaluate our method on two datasets: 1) an artificially constructed dataset with a known transformation between two distributions, and 2) recordings obtained from the primary visual cortex of a behaving animal across multiple days. We then compare several metrics and statistics between the generated and recorded data in calcium imaging traces and inferred spike trains.

1.1 Structure

Chapter 2 introduces the necessary background information for this work. Section 2.1 briefly discusses various neuronal activities techniques and then details the data and its collection process used in this work. Section 2.2 provides an overview of common techniques in modelling neuronal activities. Section 2.3 discusses generative adversarial networks (GANs) and its variations, and Section 2.4 details the cycle-consistent adversarial networks framework. Section 2.5 highlights some of the recent advancement in deep neural networks interpretation.

Next, we detail the methodology and evaluation framework of our work in Chapter 3. Section 3.1 outlines the entire pipeline of our framework, including data preprocessing, data augmentation, visualization and evaluation methods. Section 3.2 details the optimization objectives and DNNs architectures. Followed by the software implementation detail in Section 3.3.

In Chapter 4, we first present the results based on synthetic data with known ground-truth, in which we also provide a detailed analysis of the learned features from the model. We also compare different training objective formulations and model architectures. Then in Section 4.2, we demonstrate that our framework also works with recorded neuronal activities by evaluating the generated calcium fluorescent signals and their inferred

spike trains. Moreover, we employ a set of model visualization methods to interpret the learned features by the models.

Finally, Chapter 5 provides a summary of this work, including its contribution to the deep learning and computational neuroscience communities. We also discuss the limitations of our approach to the problem and propose various future research directions that consolidate cycle-consistent adversarial networks and neuronal learning analysis.

Chapter 2

Background

2.1 Neuronal activity recordings

Biological neurons propagate signals by means of electrochemical pulses (spikes) and are interconnected in large complex circuits that produce population responses with intricate structure [23, 46, 87]. Therefore, the ability to record these action potentials or spikes accurately is essential in understanding the neural information processing system. Electrophysiological recordings and calcium imaging are two popular recording techniques that enable enable recording neuronal activities from dozens to tens of thousands of neurons simultaneously [10, 51].

Large scale electrophysiology measures electrical current from the extracellular space, then spike trains from individual neurons can be extracted via spike sorting [13, 91]. This recording technique produces recording with high temporal resolution and is considered as the most accurate method to monitor spike activities [23, 91]. On the other hand, calcium imaging measures the calcium influx during the depolarization process as a proxy for an action potential activity [7, 95]. However, since calcium imaging is an indirect measurement of spiking activity, thus a transformation to convert calcium fluorescence signal to spiking activity is needed, and such deconvolution process remains a challenging task [119]. Nevertheless, this non-invasive recording technique enables practitioners to monitor a large population of neurons simultaneously and has seen rapid adoption in animal experiments in recent years [45, 81, 105, 119].

2.1.1 Animal experiment and data collection

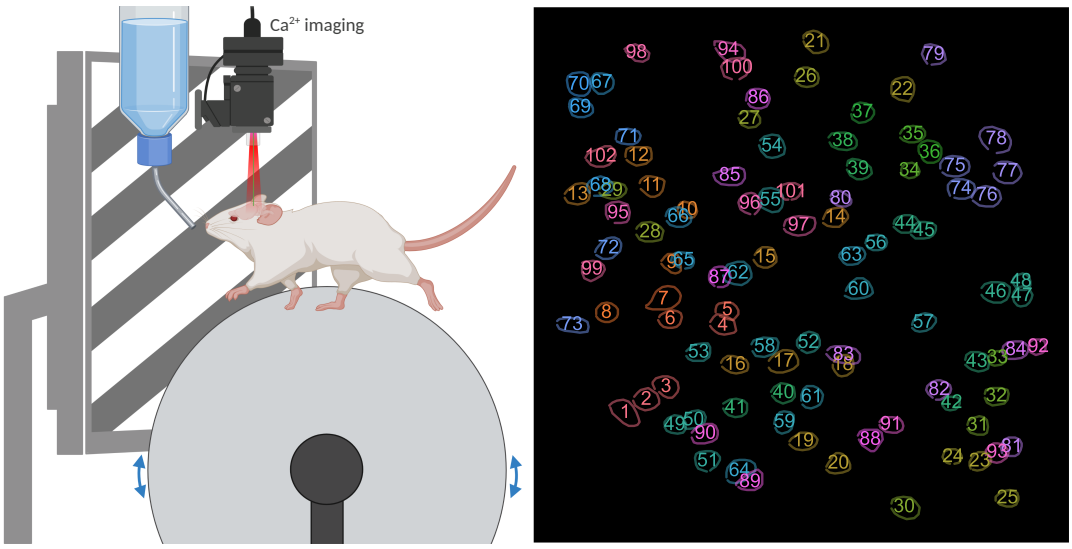
In this section, we briefly describe the virtual reality experiment setup employed to collect the calcium imaging data used in this work, which follows the same procedure as specified in Pakan et al. [81] and Henschke et al. [45].

Figure 2.1a illustrates the mouse virtual-corridor experiment setup. A head-fixed mouse was placed on a linear treadmill that allows the mouse to move forward and backward freely in virtual space while recording their virtual location and speed. A lick spout and two monitors were placed in front of the treadmill and a virtual corridor with defined grating pattern was shown to the mouse. A reward (water drop) would be made available if the mouse lick within the predefined reward location, in which the grating pattern disappears as a virtual clue. Hence the mouse should learn to utilize both visual information and self-motion feedback to maximize reward. Reward zones were set to be between 120cm to 140cm from the initial start point, and each trial would reset at 160cm regardless of the performance of the mouse.

The genetically encoded calcium indicator GCaMP6 was used to label the same set of 102 neurons in the primary visual cortex, the relative changes in fluorescence ($\Delta F/F_0$) over time were used as a proxy for an action potential. The neuron location and annotation order are shown in Figure 2.1b. Note that neurons were not annotated in a particular order, though the order displayed here indicates the row number of each neuron in the calcium signals data matrix (i.e. the first annotated neuron occupies row 0 in the data matrix, the second annotated neuron occupies row 1 in the data matrix and so on). Table 2.1 shows the basic statistics and information about the virtual-environment experiment. As the experiment progress, the number of licks by the mouse decreases while the total reward increases, suggesting that the mouse is learning the precise location to lick and maximizing its rewards. By the fourth day, the mouse was able to achieve success rate of $> 75\%$ and can be considered as "expert" at the task [81]. Hence, this dataset provides us with excellent insight into the change in neural population dynamics of the behaving mouse over different learning phrases.

2.2 Modelling neuronal activity

The ability to model realistic neuronal activity can provide us better insight in understanding the neural coding process. Numerous modelling techniques have been proposed throughout the years, ranging from maximum entropy models to deep generative models.



(a) Animal experiment illustration (b) X and Y original annotation order

Figure 2.1: (a) 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 120cm to 140cm from the initial start point, the grating pattern would disappear and replaced with blank screen. If the mouse lick within the virtual reward zone, then a droplet of water would be given to the mouse as a reward. Trials reset at 160cm. The figure is based on Figure 1 in Pakan et al. [81]. (b) 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. Neurons followed the same order across all experiments.

Here, we briefly highlight some popular neuronal modelling methods.

Spiking activities can be modelled as probability distributions with specific constraints under the Maximum Entropy principle (MaxEnt). Firing rate and pairwise correlations in neurons are two commonly used constraints to define the probability distributions [32, 97, 100]. The MaxEnt method has shown excellent results in modelling spiking activities of a small number of neurons. However, it does not scale well in high dimensional space as the number of states can grow exponentially w.r.t to the size of the population [107]. To address the computational limitation, Macke et al. [68] proposed the Dichotomized Gaussian (DG) model. The DG model simulates population activity by sampling from a multivariate normal distribution with specified mean and variance, then thresholded to generate binary spike trains.

DAY	NUM. TRIALS	AVG. DURATION	LICKS	REWARDS
1	129	6.94s	2813	140
2	177	5.08s	2364	182
3	192	4.67s	2217	198
4	203	4.43s	1671	213

Table 2.1: Basic information about the animal experiment across 4 days, including the number of trials, average duration of each trial, total number of licks and the total reward received by the mouse. The mouse achieved "expert" level by day 4 where it had a success rate of $> 75\%$ at the task. All data were recorded at a sampling rate of 24Hz. Note that the same mouse was used in the experiment.

Another popular approach is to model underlying population dynamics with latent variable models [31]. Common dimensionality reduction techniques such as factor analysis (FA) and principal component analysis (PCA) have shown success in modelling linear dependencies in latent states [20, 78]. Moreover, deep learning methods have proved to be a capable method to model non-linear dynamics [29, 83, 99]. Notably, Pandarinath et al. [83] introduced a variational autoencoder (VAE) framework that learns the non-linear latent dynamics in spiking activity with recurrent neural networks (RNNs).

Recently, the use of deep generative models to synthesize neuronal activities has gained significant interests thanks to their data-driven approach. Molano-Mazon et al. [75] demonstrated that generative adversarial networks (GANs, detailed in Section 2.3) can accurately generate spiking activity that matches the low-level statistics of a small number of neurons recorded from the salamander retina. In our previous work [62], we derived a GANs framework to synthesize fluorescent calcium signals from over a hundred neurons in the primary visual cortex and showed that our method was able to model the first and second-order statistics substantially better than the DG model.

2.3 Generative adversarial networks

Generative adversarial networks (GANs), first introduced in Goodfellow et al. [38], is an unsupervised method that learn to generate compelling samples to mimic a real data distribution via adversarial training. GANs has seen rapid adoption across

many fields since its introduction, including data synthesis [9, 26, 53], visual super-resolution [61, 131], and many more [86, 122, 128].

A typical GANs consists of a generator G and a discriminator D. The generator $G : Z \rightarrow X$ samples noise from probability distribution Z (usually Gaussian distribution), and learns to generate sample $\hat{x} = G(z)$ that resemble data from the real data distribution X. Whereas the discriminator attempts to distinguish generated samples $\hat{X}$ from real samples. The discriminator in the original proposed GANs framework [38] acts like a logistic classifier with the following loss function:

$$\mathcal{L}_D^{\text{GAN}} = -\mathop{\mathbb{E}}_{x\sim X}[\log D(x)] - \mathop{\mathbb{E}}_{z\sim Z}[\log(1 - D(G(z)))] \tag{2.1}$$

And the objective of the generator is to simply maximize the loss of the discriminator $\mathcal{L}_G^{\text{GAN}} = \mathop{\mathbb{E}}_{z\sim Z}[\log(1 - D(G(z))]$. However, the objective of minimizing the Jensen-Shannon distance is notoriously difficult to train. The minimax formulation is prone to model collapse, where the generator can simplify its task by simply generating a particular subset of data instead of learning the true data distribution [16]. It is also possible that the discriminator overpowers the generator in the early phase of training causes a vanishing gradient for the generator [36, 82]. Since the introduction of GANs, there has been a tremendous amount of work that aims to address the said issues, here we highlight some of the popular proposed solutions.

Mao et al. [70] proposed the use of least-square loss in the discriminator loss calculation instead, in which the authors showed that such formulation minimize the Pearson $\mathcal{X}^2$ divergence implicitly:

$$\mathcal{L}_D^{\text{LSGAN}} = -\mathop{\mathbb{E}}_{x\sim X}[(D(x) - 1)^2] + \mathop{\mathbb{E}}_{z\sim Z}[D(G(z))^2] \tag{2.2}$$

$$\mathcal{L}_G^{\text{LSGAN}} = -\mathop{\mathbb{E}}_{z\sim Z}[(D(G(z) - 1)^2] \tag{2.3}$$

Wasserstein GAN (WGAN) [5] proposed to minimize the continuous earth mover's distance (also known as the Wasserstein distance), in which the discriminator $D : X \rightarrow \mathbb{R}$ serves as a value function instead and that it has to be a 1-Lipschitz function. In order to enforce the Lipschitz condition on the discriminator, weights in the discriminator are restricted to be within $[-c, c]$ where c is a hyper-parameter. Gulrajani et al. [41] further improved the method with WGANGP, which enforce the Lipschitz condition via gradient penalty instead of weight-clipping. The authors take advantage of the fact that a differentiable function is 1-Lipschitz if the norm of the gradient is 1, hence they

introduced a regularization term that penalize the discriminator when its gradient norm is not 1:

$$\mathcal{L}_D^{\text{WGANGP}} = \mathop{\mathbb{E}}_{z\sim Z}[D(G(z))] - \mathop{\mathbb{E}}_{x\sim X}[D(x)] + \lambda_{\text{GP}} \mathop{\mathbb{E}}_{x\sim X, z\sim Z}[(\| \nabla D(\varepsilon x + (1-\varepsilon)G(z)) \|_2 - 1)^2] \tag{2.4}$$

$$\mathcal{L}_G^{\text{WGANGP}} = - \mathop{\mathbb{E}}_{z\sim Z}[D(G(z))] \tag{2.5}$$

where λ_{GP} is the gradient penalty coefficient hyper-parameter, and ε is a 0-to-1 linear interpolation coefficient sampled from a uniform distribution. Since the discriminator objective in WGANGP is more complicated, the authors proposed that the discriminator can be updated multiple steps for every generator step.

Kodali et al. [58] pointed out that the interpolation term $\varepsilon_{\text{GP}} x + (1-\varepsilon_{\text{GP}})G(z)$ is problematic due the fact that $G(z)$ could be very far away from the true distribution, especially in the early training phase. The authors suggested that noise can be added to the true sample, instead of sampling a random point $G(z)$, hence resulting in a point that is "close" to the true data distribution:

$$\mathcal{L}_D^{\text{DRAGAN}} = \mathcal{L}_D^{\text{GAN}} + \lambda_{\text{GP}} \mathop{\mathbb{E}}_{x\sim X, z\sim \mathcal{N}(0,c))}[(\| \nabla D(x+z) \|_2 - 1)^2] \tag{2.6}$$

$$\mathcal{L}_G^{\text{DRAGAN}} = \mathop{\mathbb{E}}_{z\sim Z}[\log(1 - D(G(z)))] \tag{2.7}$$

where c a hyper-parameter for the Gaussian distribution.

Nevertheless, the GANs objective function formulation is still an active area of research, with new frameworks being introduced frequently. Numerous empirical studies on GANs show that each method has its benefits and drawbacks, and many nuances remain in training GANs effectively [36, 59, 66, 85, 96].

2.4 Cycle-consistent adversarial networks

In the vanilla GANs framework, the generator learns to synthesize data that resemble a certain data distribution, though with no control over which domain within the distribution to generate. For instance, we cannot specify vanilla GANs, trained on images of dogs, to generate images of Bulldog specifically. To address this limitation, Mirza and Osindero [74] introduced Conditional GANs (cGANs) where the generator and discriminator also consider the class label, hence allowing the model to generate class-specific samples.

Suppose we want to perform a translation task, such as French-to-English translation. We can train a vanilla GANs where the generator inputs a French sentence and outputs a corresponding English sentence, and the discriminator learns to distinguish real English sentences from generated ones. Given an input French sentence "comment ça va", (one of) the correct translation "how are you" and incorrect translation "I go to school by bus" would receive similar scores from the discriminator, as both are valid English sentences. Hence, GANs or cGANs alone are inadequate in tackling this scenario. Similarly, if we attempt to train a GANs to learn the transformation of the neuronal activities between two different experiment sessions, we too cannot verify that the generated sample is a direct correspondence of the input. Moreover, unlike natural languages, it would be very challenging to obtain paired recording samples from behaving animal experiments, if not impossible. Interestingly, it is possible to learn the mapping between two unpaired distributions with multiple GANs working in conjunction. Here, we outline one such powerful unsupervised learning method.

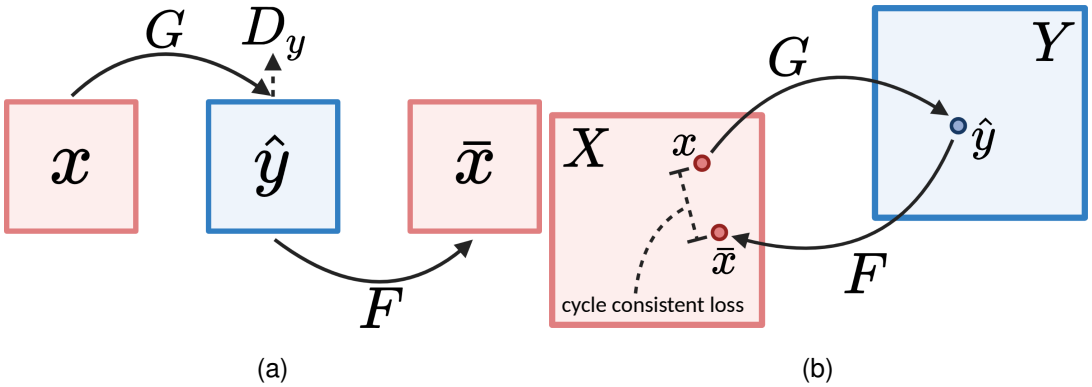


Figure 2.2: Illustration of (a) the data flow and (b) the cycle-consistent loss in a forward cycle $X \rightarrow Y \rightarrow X$. Illustration re-created from Figure 3 in Zhu et al. [132]. G and F are generators that learn the transformation of $X \rightarrow Y$ and $Y \rightarrow X$ respectively. 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 $Y \rightarrow X \rightarrow X$ is a mirrored but opposite operation that run concurrently with the forward cycle.

CycleGAN [132] learns the mapping between unpaired distributions X and Y with two generators: $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and two discriminators $D_X : X \rightarrow [0,1]$ and

$D_Y : Y \to [0,1]$. We sample x from distribution X and apply transformation G to obtain $\hat{y} = G(x)$. Then to verify $\hat{y}$ resemble distribution Y, we can use $D_Y(\hat{y})$ to discriminate it. The procedure thus far is very similar to a typical GANs, with the exception that we sample X instead of noise Z. However, as discussed previously, there is no direct way to ensure $\hat{y}$ corresponds to x. Ingeniously, the authors suggested that we could apply transformation F to $\hat{y}$ to obtain $\bar{x} = F(\hat{y})$. If both G and F transformations are sufficiently good, then $\| x - \bar{x} \|$ should be minimal. The authors called this different in cycle reconstruction the cycle-consistent loss. Notice that it is possible for G and F to obtain a good cycle-consistent loss while having a poor intermediate step representation $\hat{y} = G(x)$. For instance, if G and F are identity functions, then the cycle-consistent loss would always be zero. To mitigate this issue, the authors proposed that we should train both forward cycle $X \to Y \to X$ and backward cycle $Y \to X \to Y$ concurrently. The discriminators D_X and D_Y should discourage the possibility of $\hat{y} = G(x)$ and $\hat{x} = F(y)$ being identical to x and y by having a low discriminator score. Moreover, the cycle adversarial training procedure should minimize the likelihood of $\hat{y}$ and $\hat{x}$ being completely unrelated to x and y, nevertheless, it is still possible. Figure 2.2 illustrates the forward cycle step $X \to Y \to X$.

Here, we formalize the objective functions of the 4 networks in CycleGAN, which are based on the LSGAN formulation. The discriminators follows the same formulation as to Equation 2.2:

$$\begin{aligned} \mathcal{L}_{D_X}^{\text{CycleGAN}} &= - \mathop{\mathbb{E}}_{x \sim X}[(D(x)-1)^2] + \mathop{\mathbb{E}}_{y \sim Y}[D_X(F(y))^2] \\ \mathcal{L}_{D_Y}^{\text{CycleGAN}} &= - \mathop{\mathbb{E}}_{y \sim Y}[(D(y)-1)^2] + \mathop{\mathbb{E}}_{x \sim X}[D_Y(G(x))^2] \end{aligned} \tag{2.8}$$

The objective functions for the generators are similar to Equation 2.3:

$$\begin{aligned} \mathcal{L}_{G}^{\text{CycleGAN}} &= - \mathop{\mathbb{E}}_{x \sim X}[(D_Y(G(x)-1)^2] \\ \mathcal{L}_{F}^{\text{CycleGAN}} &= - \mathop{\mathbb{E}}_{y \sim Y}[(D_X(F(y)-1)^2] \end{aligned}$$

In addition, the forward and background cycle-consistent loss:

$$\mathcal{L}_{\text{cycle}}^{\text{CycleGAN}} = \mathop{\mathbb{E}}_{x \sim X}[\| x - F(G(x)) \|] + \mathop{\mathbb{E}}_{y \sim Y}[\| y - G(F(y)) \|]$$

In addition, when $x \sim X$ is provided to $F : Y \to X$, then the generated sample $\hat{x} = F(x) = x$ should still be of distribution X, thus the authors formulate the identity loss

$\mathcal{L}_{\text{identity}}$:

$$\mathcal{L}^{\text{CycleGAN}}_{\text{G identity}} = \mathop{\mathbb{E}}_{y\sim Y}[\| y - G(y) \|]$$
$$\mathcal{L}^{\text{CycleGAN}}_{\text{F identity}} = \mathop{\mathbb{E}}_{x\sim X}[\| x - F(x) \|]$$

The mean absolute error was used as the distance function for both $\mathcal{L}_{\text{cycle}}$ and $\mathcal{L}_{\text{identity}}$, though other common distance functions can also be used, such as mean-squared error or Huber loss. Taken together, the final objective functions for the generators are as follows:

$$\begin{aligned}\mathcal{L}^{\text{CycleGAN}}_{\text{total G}} &= \mathcal{L}^{\text{CycleGAN}}_{G} + \lambda_{\text{cycle}}\mathcal{L}^{\text{CycleGAN}}_{\text{cycle}} + \lambda_{\text{identity}}\mathcal{L}^{\text{CycleGAN}}_{\text{G identity}} \\ \mathcal{L}^{\text{CycleGAN}}_{\text{total F}} &= \mathcal{L}^{\text{CycleGAN}}_{F} + \lambda_{\text{cycle}}\mathcal{L}^{\text{CycleGAN}}_{\text{cycle}} + \lambda_{\text{identity}}\mathcal{L}^{\text{CycleGAN}}_{\text{F identity}}\end{aligned} \tag{2.9}$$

where λ_{cycle} and $\lambda_{\text{identity}}$ are the cycle-consistency loss and identity loss coefficient hyper-parameters respectively. Note that some popular CycleGAN implementations only include half the cycle-consistent loss in total generator loss [77], i.e. $\mathcal{L}^{\text{CycleGAN}}_{\text{total G}} = \mathcal{L}^{\text{CycleGAN}}_{G} + \lambda_{\text{cycle}} \mathop{\mathbb{E}}_{y\sim Y}[\| y - G(F(y)) \|] + \lambda_{\text{identity}}\mathcal{L}^{\text{CycleGAN}}_{\text{G identity}}$, though the original formulation performed better in our experiment on the horse-to-zebra dataset and we are not using this alteration in this work.

In essence, CycleGAN relies on the cycle-consistency of $F(G(x)) = \bar{x} \approx x$ and $G(F(y)) = \bar{y} \approx y$ to learn the mapping between the two otherwise unaligned distributions. Relating back to the French-to-English translation example, a sentence that translates from French-to-English and back from English-to-French, albeit not guaranteed, should be identical to the original phrase when trained with the cycle-consistent objective. Since the introduction of CycleGAN, this powerful unsupervised learning framework has shown success in other fields and disciplines, such as CT-to-MR scan translation [21, 123], cryptanalysis [34] and many more [18, 71, 109].

In this work, we adapt the CycleGAN framework, in combination with some of the GANs objective formulations mentioned in Section 2.3, to learn the unknown mapping between pre-learning and post-learning neuronal activities recorded from the animal experiments detailed in Section 2.1.1.

2.5 Explainable deep neural networks

Deep neural networks (DNNs) have outperformed many classical machine learning algorithms in a wide variety of tasks in no small part due to their ability to self-identify

implicit patterns in data [37, 60, 64]. Nevertheless, such a powerful self-learning mechanism could be a double-edged sword. For instance, Makino et al. [69] showed that DNNs would learn different features relative to those identified by radiologists, where some of these self-identified features could be spurious (to our knowledge), while others present new potential insights to the task. The uninterpretable nature of DNNs is problematic and risky in some fields, such as healthcare and biomedical science, where it is crucial that medical practitioners can understand and verify the learned features by the algorithm before making any life-dependent decisions. Therefore, the machine learning community has put significant efforts into improving the explainability of DNNs in recent years [30].

With respect to this work, we would like to interpret the learned features by the generators and discriminators. First, we would like to ensure the 4 networks are learning reasonable features from the neuroscience perspective, instead of something trivial. Second, the self-identified patterns by the networks could provide us better insight into the neuronal learning process. For instance, perhaps there exists a particular set of neurons or periods in the trial that is highly influential in the learning process, thus allow experimentalists to pay more attention to those areas. Here, we highlight some popular methods that can improve the interpretability in convolutional neural networks (CNNs).

2.5.1 Features visualization

As the vast majority of DNNs are trained with gradient descent, one popular approach to visualize what is learned by the deep learning model is the gradient-based approach. In a CNN image classifier, we can measure the gradient of each class w.r.t individual pixels in the input image as an indication of how each pixel influences the prediction. Simonyan et al. [101] proposed to visualize pixels of interest in the form of saliency maps. For an input image I, the saliency map at image location I_0 can be computed as $M_{\text{saliency}} = \frac{\partial S_c}{\partial I}\Big|_{I_0}$ where S_c is the class score function (which compute the logits prior to the softmax output) for each class c. Smilkov et al. [103] further improved the method by averaging the gradients of multiple images from the same distribution, as well as adding Gaussian noise to the images, to generate a smoother saliency map. However, saliency map can be affected by common data pre-possessing operations such as normalization and standardization [33, 56], hence a more robust solution is needed.

Class Activation Map (CAM) [130] has become a very popular alternative to visualize

CNN activation. Instead of the typical approach of using fully connected layers to down-sample the features maps from the last convolutional layers to the output softmax layer, the authors proposed to use the Global Average Pooling (GAP) layer followed by a linear layer instead. More formally, let $f_k(i,j)$ be the feature maps of the last convolutional layer at location (i,j) with depth k, then the class score function can be defined as $S_c = \sum_k [w_k^c \sum_{i,j} f_k(i,j)]$ where the inner summation is the GAP layer. With this architecture, the learned w_k^c can be interpreted as the relevancy of each feature map toward a specific class c, hence can be projected back to the input to highlight the important regions. The CAM of each class is therefore the weighted sum of the corresponding feature maps $M_{\text{CAM}}^c(i,j) = \sum_k w_k^c f_k(i,j)$. This approach does not require computing the gradient w.r.t the input hence is more robust and efficient, though CAM is not very versatile as it requires architecture change. Selvaraju et al. [98] suggested that the importance weight w_k^c could be estimated instead, hence allowing their method – GradCAM to work with any CNN architecture. The importance score α_k^c can be computed as the GAP of the gradient of logits y^c w.r.t feature maps $f_k(i,j)$:

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j -\frac{\partial y^c}{\partial f_k(i,j)} \tag{2.10}$$

Then the CAM can be computed as

$$M_{\text{GradCAM}}^c(i,j) = \text{ReLU}(\sum_k \alpha_k^c * f_k(i,j)) \tag{2.11}$$

`ReLU` non-linearity was added to remove features that influence the class of interest negatively. Thanks to its simplicity and generalizability, GradCAM has become a popular method to visualize and understand the area of interested learned by any CNN models.

2.5.2 Self-attention mechanism

Feature-importance methods such as saliency map and GradCAM allow practitioners to interpret what the algorithm has learned yet they do not improve the model performance. With the introduction of Transformer [114], which relies on attention modules instead of the transitional long-short term memory (LSTM) layer, attention-based models have dominated many natural language processing tasks, ranging from language translation to text generation [22, 25, 90, 124]. The Multi-Head Attention module Transformer uses a dictionary-like layer that store key-value pairs of the input sequence, then the model attempts learn the relationship between each word in a sentence. Such formulation works

very well with data that consists of temporal dependencies, such as natural language. However, the attention mechanism in language models does not work well with data that has a high dimensional spatial dependency, such as images. Nevertheless, one can see the attraction of the self-attention mechanism. On one hand, it can improve the model performance while decreasing the computation cost (in the case of language modelling). On the other hand, the learned attention units can improve the interpretability of the model. A great number of self-attention mechanisms have been introduced to work with CNNs in recent years, here we briefly discuss the method we are using.

Trainable attention in CNNs can be separated into two main categories [15]. Hard attention – a stochastic approach where patches of the image are exposed to the network which allows the model to focus better, yet, there is a chance that the selected patches are not of importance. Soft attention – a deterministic approach in which an attention mask is learned for the entire input image, though the attention learned is often more coarse-grained. Jetley et al. [52] introduced a soft-attention module into the VGG classifier [102], where the module learns an $[0, 1]$ attention mask from local and global features, then apply the mask over the output from the previous layer, and it saw significant improvement in various image classification tasks. These soft-attention modules enable easy integration into existing CNNs architectures while improving the explainability and the performance of the learning algorithm.

Chapter 3

Methods

3.1 Model pipeline

We are interested in whether or not the CycleGAN unsupervised learning framework can learn the mapping between pre-learning and post-learning neuronal activities. If so, can we interpret what the generators and discriminators deem relevant in the cellular activities during their transformation operations? Therefore, we devise a consistent analysis framework, including data preprocessing and augmentation, model tuning and interpretation, and evaluation of the generated calcium fluorescent signals and their inferred spike trains. Figure 3.1 illustrates the complete pipeline of our work[1].

3.1.1 Data preprocessing

As discussed in Section 2.1.1, the mouse achieved expert-level performance in the virtual-environment task by the fourth day of the experiment, a sharp contrast to the inexperienced performance exhibited on the first day when the mouse was new to the experiment. Hence, we denote the pre-learning and post-learning activities to be X and Y, which are random variables that represent recordings obtained from the first and the fourth day. As shown in Table 2.1, $W = 120$ neurons from the primary visual cortex were monitored at a sampling rate of 24Hz, as well as trial-per-trial information including the virtual distance, licks and rewards. In total, 21471 and 21556 calcium imaging samples were recorded on day 1 and 4 respectively (i.e. data with shape

[1]The software codebase of this work is attached in the dissertation submission and will be made publicly available upon publication.

$(21471, 102)$ and $(21556, 102)$). As we are interested in whether or not the generators and discriminators can identify patterns relevant to the animal experiment purely based on the cellular activities thus we do not want to incorporate any trial information into the training data. We first segment the two datasets with a sliding window of size $H = 2048$ along the temporal dimension, resulting in data with shape (N, H, W) for X and Y where N is the total number of segments. Note that a segment length of $H = 2048$ time-steps is ~ 85 seconds of wall time. We then normalize each set to range $[0, 1]$, for instance, we normalize X by $X = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$ where $X_{\min}$ and $X_{\max}$ are the minimize and maximum value in X. In order to take advantage of the spatial-temporal information in the neuronal activities with 2-dimensional convolutional layers, we further convert the two sets to have shape (N, H, W, C) where $C = 1$. We then divide the two datasets into train, validation and test set with 3038, 200 and 200 samples respectively.

3.1.2 Neuron spatial ordering

CNNs with a smaller kernel can often perform as well or even better than models with a larger kernel while having less trainable parameters [43, 63]. Nevertheless, kernel size is often proportional to the network's receptive field, the region in the input that the convolutional layer can learn from [3]. Such property is negligible with most image-based tasks, as images usually have localized spatial relations where a pixel is likely to be high-correlated with its surrounding pixels though not pixels far away [42, 65]. However, as mentioned in Section 2.1.1, the neuron annotations in our datasets are not ordered in a particular manner, rather than grouping neurons that are high-correlated, thus the limited receptive field could restrict the networks from learning meaningful spatial-temporal information.

To mitigate the said issue, we derive a procedure to pre-sort both X and Y, such that neurons that are highly correlated or relevant are closer in space. A naive approach is to sort the neurons by their firing rate, where the neuron with the highest firing rate is ranked first in the data matrix. However, it is possible that not all high-firing neurons are the most relevant in the population, therefore, we also explore a data-driven approach. Deep autoencoders have shown excellent results in feature extraction and representation learning [35, 111, 116], and we can take advantage of its unsupervised feature learning ability.

We develop two deep autoencoder models $AE_X(x)$ and $AE_Y(y)$ where each model learns to reconstruct X and Y respectively. Each model has 3 convolution down-sampling

blocks, follows by a bottleneck layer, then 3 transposed-convolution up-sampling blocks such that the reconstruction has the same dimension as the inputs. The down-sampling block consists of a convolution layer followed by Instance Normalization [112], Leaky ReLU (LReLU) activation [67] and Spatial Dropout [110], whereas a deconvolution layer is used in the up-sampling block instead. We first train the two models using the mean-squared error as the reconstruction objective, then we use the reconstruction error on the test sets to sort the neurons (in ascending order) in X and Y. Once we have the neuron orders for X and Y, we sort the input data in the data preprocessing step as shown in Figure 3.1.

3.1.3 Synthetic data

CycleGAN [132] was first introduced for image-to-image translation (see Section 2.4), albeit the two distributions are not aligned hence we cannot use common distance metrics to compare $\| x - F(y) \|$ and $\| y - G(x) \|$, one could still visually inspect whether or not $\hat{x} = F(y)$ and $\hat{y} = G(x)$ are reasonable transformations. However, it would be difficult to visually inspect the same transformations $F(y)$ and $G(x)$ on neuronal activities, and one could only rely on the reconstruction loss $\| x - F(G(x)) \|$ and $\| y - G(F(y)) \|$.

To improve this situation, we introduce an additional dataset $Y = \Phi(X)$ with a known transformation Φ, such that $G : X \to Y = \Phi(X)$ and $F : Y = \Phi(X) \to X$, then we could verify $G(x) = \hat{y} = \Phi(x)$ and $F(y) = \hat{x} = \Phi'(y)$, where Φ' is the inverse transformation. To this end, we defined the transformation Φ as:

$$\Phi(x) = m_{\text{diagonal}} \times x + 0.5\eta \qquad \eta \sim \mathcal{N}(\mu_x, \sigma_x^2)$$

where μ_x and σ_x are the neuron-wise mean and standard deviation of X, whereas m_{diagonal} is a diagonal mask to zero-out the lower left corners of the signals. This spatiotemporal transformation is chosen because it allows us to interpret and visualize the generated samples easily. Figure 3.2 shows an example of before and after augmenting X. Importantly, we shuffle the train set after the augmentation procedure so that X and Y appears to be unpaired to the model. However, we do not shuffle the test set, so that we can directly compare $\| X - F(Y) \|$ and $\| Y - G(X) \|$.

Keep in mind that in the scenario where we wish to take advantage of neuron spatial ordering discussed above, we would first sort the neurons in X as described in Section 3.1.2 then apply the augmentation to the sorted X.

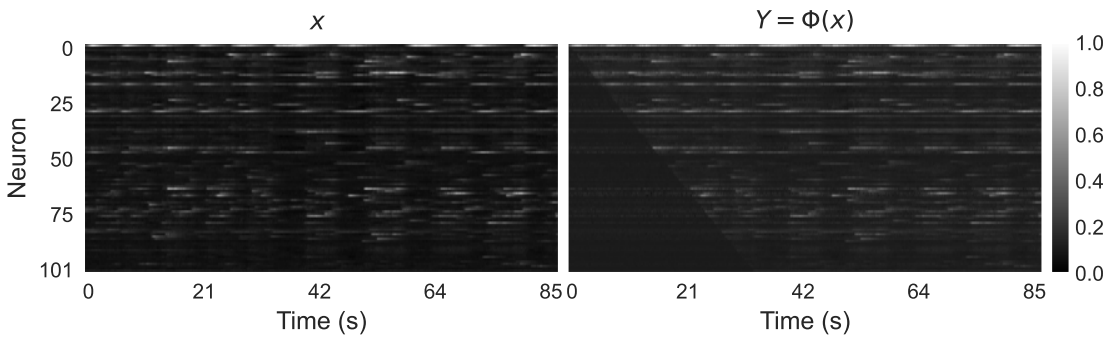


Figure 3.2: (left) original x and (right) augmented $\Phi(x)$ calcium traces. Note that the calcium traces here were normalized to $[0, 1]$ as part of the preprocessing.

3.1.4 Evaluation and metrics

To evaluate the generated results of G and F, we first directly evaluate the cycle-consistent loss of the $\| x - F(G(x)) \|$ and $\| y - G(F(y)) \|$. We also compare $\| x - F(x) \|$ and $\| y - G(y) \|$ as we expect that the generators should not modify the input if it is already in its target distribution. We then evaluate our model performance in terms of spike activities of the of following distribution combinations: $X \mid F(G(X))$, $Y \mid G(F(Y))$, $X \mid F(X)$ and $Y \mid G(Y)$. In this work, we use the deep learning based deconvolution algorithm – Cascade [94] to infer spike trains from the recorded and generated calcium signals. Since we inferred the spike trains from both generated and recorded data with the same algorithm, the bias in the deconvolution method should apply to all data. We measure spike train similarities and statistics with the following commonly used metrics: (a) mean firing rate for evaluating single neuron statistics; (b) pairwise Pearson correlation for evaluating pairwise statistics; (c) pairwise van Rossum distance [93] for evaluating general spike train similarity. We evaluate these quantities across the whole population for each neuron or neuron pair and each short time interval (100 ms) and compare the resulting distributions over these quantities obtained from training data as well as generated data. We, therefore, validate the whole spatial-temporal first and second-order statistics as well as general spike train similarities. We use the Electrophysiology Analysis Toolkit [24] to implement most of the spike analysis methods.

As discussed in Section 2.5, there has been a significant advancement in interpretable artificial intelligence, and we employ some of the mentioned methods in this work. We incorporate a self-attention module in the generator which would allow us to visualize the learned attention mask at different latent dimensions, we further detail the

generator architecture in Section 3.2.1. In addition, we generate localization maps via GradCAM [98] to visualize regions or neurons that each model concentrates on. On top of allowing machine learning practitioners to visualize and verify the networks are learning meaningful features, both feature-importance and self-attention methods can also provide experimentalists insights into relevant neurons and patterns in the animal's learning process.

3.2 Networks objective and architecture

Following the CycleGAN [132] framework (see Section 2.4), we employ two generators G and F, and two discriminators D_X and D_Y to learn mapping between X day 1 and Y day 4 recorded data (or $Y = X_{\text{augmented}}$ when trained with the synthetic dataset). As discussed in Section 2.3, there are numerous objective formulation in GANs, each with their own advantage and disadvantage. Hence, in addition to the LSGAN [70] objective which was also used in CycleGAN, specified in Equation 2.8 and 2.9, we also adapted WGANGP [5] and DRAGAN [58] into the loss calculation. To train CycleGAN with the WGANGP objective, we can define the forward cycle loss function for discriminator D_Y and generator G as:

$$\begin{aligned}
\mathcal{L}_{D_Y}^{\text{CycleWGANGP}} &= \mathop{\mathbb{E}}_{x\sim X}[D_Y(G(x))] - \mathop{\mathbb{E}}_{y\sim Y}[D_Y(y)] \\
&\quad + \lambda_{\text{GP}} \mathop{\mathbb{E}}_{x\sim X, y\sim Y}[(\parallel \nabla D(\varepsilon y + (1-\varepsilon)G(x)) \parallel_2 - 1)^2] \\
\mathcal{L}_{G}^{\text{CycleWGANGP}} &= - \mathop{\mathbb{E}}_{x\sim X}[D_Y(G(x))]
\end{aligned}$$

We can modify the CycleGAN framework with DRAGAN objective as follows:

$$\begin{aligned}
\mathcal{L}_{D_Y}^{\text{CycleDRAGAN}} &= \mathcal{L}_{D_Y}^{\text{GAN}} + \lambda_{\text{GP}} \mathop{\mathbb{E}}_{y\sim Y, z\sim\mathcal{N}(0,c)}[(\parallel \nabla D(y+z) \parallel_2 - 1)^2] \\
\mathcal{L}_{G}^{\text{CycleDRAGAN}} &= \mathop{\mathbb{E}}_{x\sim X}[\log(1 - D_Y(G(x)))]
\end{aligned}$$

Note that we still add the cycle-consistent loss $\mathcal{L}_{\text{cycle}}^{\text{CycleGAN}}$ and identity loss $\mathcal{L}_{\text{identity}}^{\text{CycleGAN}}$ to compute the total generator loss.

3.2.1 Generators

The generator architecture used in this work, shown in Figure 3.3, is based on the ResNet-like [43] generator in CycleGAN [132] with a number of improvements detailed below. 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 $\texttt{US}_2$). The down-sampling block uses a 2D strided convolution layer to reduce the spatiotemporal dimensions by factor of 2, which is then follows by Instance Normalization [112], LReLU [67] activation and Spatial Dropout [110]. The up-sampling block has the same structure as the down-sampling block but with a transposed convolution layer instead. The residual block consists of two convolution blocks with padding added to offset the dimensionality reduction and a skip connection that connect 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 $\texttt{US}_1$, followed by sigmoid activation to scale the final output to have range $[0,1]$.

Residual connections have been shown to improve gradient flow in DNNs thus mitigate the issue of vanishing gradient and allow deeper networks to be trained effectively [43, 44, 48]. 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 connection was introduced in the popular image segmentation model UNet [92]. We denote the level-wise residual connected network used in this work as `ResNet`.

Furthermore, we introduce an Attention Gate (AG) module, based on Oktay et al. [80], as a replacement for the concatenation operation in the residual connection described above. The yellow block in Figure 3.3 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 a (separate) 1×1 convolution layer followed by Instance Normalization. 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 Instance Normalization, resulting in a vector with shape $(H_{\text{AG}}, W_{\text{AG}}, 1)$. Sigmoid activation is applied to obtain a $[0,1]$ attention mask, where units closer to 1 indicate regions that are more relevant. Finally, we return the element-wise multiplication between the attention mask with q. 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. In other words, AG should learn what information from the input is important in the transformation. Moreover, as the attention mask is of the same dimension of the input q, we can later superimpose the attention mask on q to visualize the region of interest learned by the model. We denote the attention-gated network used in this work as `AGResNet`.

3.2.2 Discriminators

The discriminator in GANs, which behaves like a binary classifier, is usually made up of dimension reduction layers and outputs a single value to indicate whether or a given sample resembles data from a certain distribution. Such coarse-grained formulation, albeit simple, could be problematic in cases where part of the generated sample is compelling while the rest is not. For example, suppose we train a GANs to generate images of horses. The generated image may contain a detailed horse but with an uncompelling background, then the discriminator has to penalize the entire image as it only outputs a single score. Therefore, Isola et al. [49] proposed that the discriminator should output a 2D vector (for a 2D input) where each element in the output vector corresponds to different regions w.r.t the original image. In other words, the discriminator outputs a score for each sub-area of the original image, thus enabling the generator to receive fine-grained feedback. The discriminators D_X are D_Y used in this work also output a vector of shape $(H_{\text{out}}, W_{\text{out}}, 1)$ (instead of just $(1,)$) where each element focus on a particular region in the input calcium signal.

The discriminators contain 3 down-sampling blocks where each block reduces the spatiotemporal dimension by a factor of 2, as as the down-sampling blocks in Section 3.2.1. For an input sample with shape $(H = 2048, W = 102, C = 1)$, the discriminator output a sigmoid activated vector with shape $(H_{\text{out}} = 256, W_{\text{out}} = 13, 1)$. Each element has range $[0, 1]$ such that a value closer to 1 suggests that the patch is from a real sample.

3.3 Implementation detail

The vast majority of the software codebase was written in Python, where the DNNs were implemented in TensorFlow [1] thanks to its hardware optimization and vast community support. As training 4 large DNNs requires a significant amount of computing resources, especially in GPU memory, hence we have added multi-GPU training support in our codebase. In addition, to take advantage of half-precision computations that are

implemented in many recent accelerators [79, 39], our codebase also supports Mixed Precision training [72]. Hence, the majority of computations can be performed in `float16` representation, with the exception of gradient calculations. To avoid numeric underflow and overflow, gradient computations were done in single-precision `float32`. Effectively, allowing us to fit twice as much data to the accelerator. As a result, we fit all 4 networks with a batch size of $N_{\text{batch}} = 4$ on a single NVIDIA V100 GPU and it can scale up almost linearly with the number of GPUs.

GANs themselves are notoriously hard to train, with problems such as mode collapse, diminishing gradients and non-convergence to name a few [4, 8]. Training 2 GANs in conjunction further magnify the said challenges. Therefore, we have implemented and adapted several techniques to improve some of the model instability issues. Typically, the discriminator receives 0-1 class labels, where a label value of 0 indicates the given sample is generated whereas a label value of 1 indicates a real sample [5, 41, 70]. However, such formulation could cause the discriminator to be overconfident in its predictions, especially early on in the training process when the generated samples are not at all compelling [96]. To address the said issue, we applied label smoothing to both discriminators, a technique that is commonly used in classifiers [76], where values between $[0.9, 1.0]$ represent real samples and $[0.0, 0.1]$ for generated samples. In addition, we have adopted the two-time scale update rules, where generators and discriminators have different learning rates, thus allowing the generators to make small but steady improvements to fool the discriminators [47]. We set a learning rate of $\alpha_G = 0.0001$ and $\alpha_D = 0.0004$ for the generators and discriminators respectively and trained all 4 networks with Adam optimizer [57]. Table A.1 shows the hyper-parameters used to train our model with different GANs objective functions; note that AGResNet does not introduce any additional hyper-parameters.

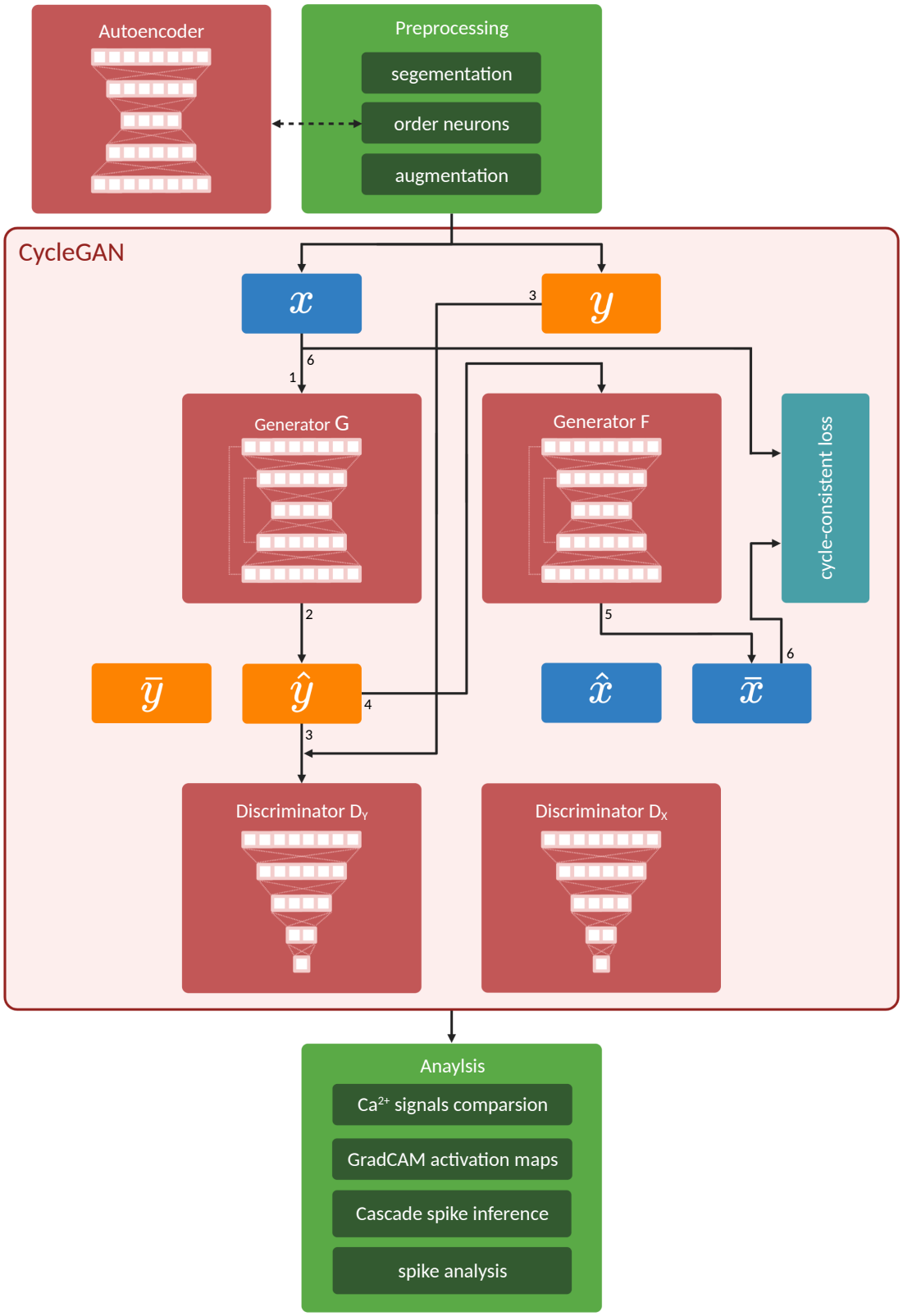


Figure 3.1: Illustration of the complete pipeline used in this work. Black directed lines represent the flow of data and the numbers indicate its order. Note that only the forward cycle step $X \rightarrow Y \rightarrow X$ is shown here for better readability.

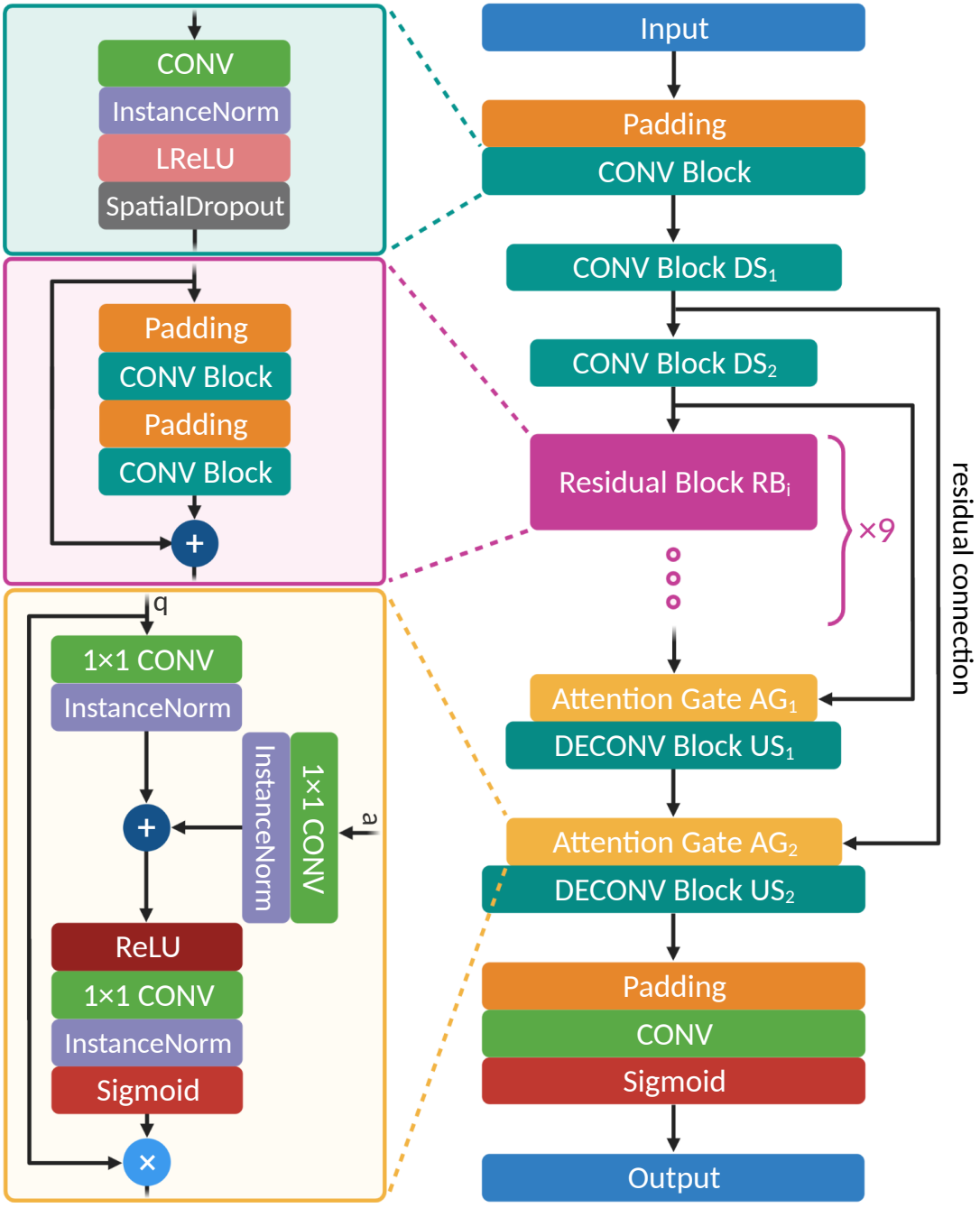


Figure 3.3: Architecture diagram of generator G and F. $+$ and $\times$ denotes addition and element-wise multiplication respectively. Note that the Attention Gate (AG) block can be replaced by a concatenation operation between the output of the previous block and the output from the down-sampling block from the same-level. e.g. if AG is not used, then the input to US_1 is $\texttt{concat}(DS_2, RB_9)$.

Chapter 4

Results

In this chapter, we present the results that demonstrate the CycleGAN framework can learn meaningful transformations between two distributions of neuronal activities with unknown mapping. In other words, given the calcium imaging signals recorded from a mouse on the first day of the behaviour experiment, can we translate the input to signals that correspond to expert-level activities and vice versa. To show that our method is capable of learning subtle differences in calcium traces, we first fit our model on the synthetic dataset, which has known ground truth thus allows direct comparison. In addition, we also investigate the GANs objective formulations and generator architectures described in Section 3.2. Next, we evaluate our method with data recorded from the mouse virtual-corridor experiment. We analyze both generated calcium signals and their inferred spike trains statistics. Moreover, we explore different neuron spatial ordering strategies and their effects on the model performance. We employ feature-importance and self-attention methods that enable us to interpret what is being learned by the models. All models presenting below were trained for 200 epochs where most models converge by ~ 180 epochs.

4.1 Synthetic data

In order to ensure that the CycleGAN framework can indeed learn transformation between two unpaired distributions of calcium fluorescent signals, hence we train the model on a synthetic dataset X and $Y = \Phi(X)$, where Φ is a known spatiotemporal transformation (see Section 3.1.3). Moreover, we randomize the index of the training set, so that X and $Y = \Phi(X)$ appears to be unaligned to the two generators, while we

can still directly compare the generated samples $|Y - G(X)|$ and $|X - F(Y)|$ on the aligned test set. In addition, as we can directly evaluate the performance of the model, we also use the synthetic dataset to compare different generator architectures and GANs objective formulations detailed in Section 3.2.

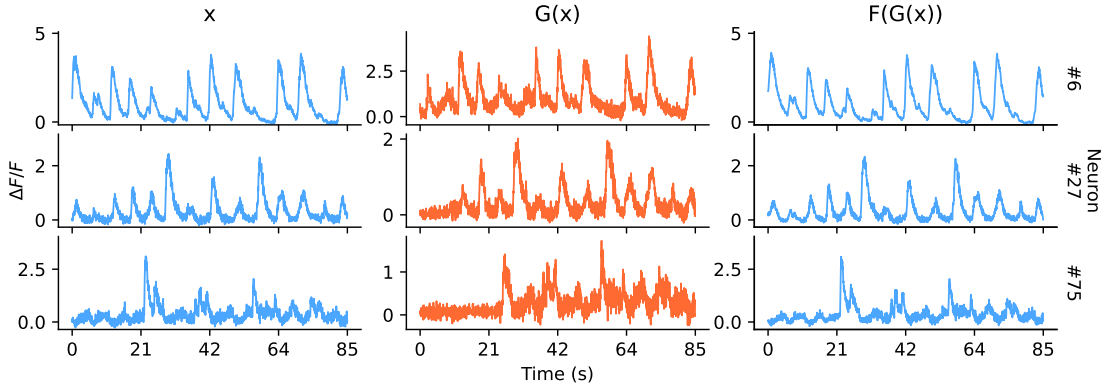
We first fit our model with different generator architectures with the LSGAN objective. In addition to ResNet and AGResNet, we have also included an identity model as a baseline. Figure 4.1 shows calcium signals of a forward and backward cycle of 3 neurons from AGResNet. Notice that as the neuron index increase, the region replaced with noise also lengthen. For instance, the first ~ 22 seconds of the signals in neuron #75 were noise in comparison to ~ 9 seconds for neuron #27 in Figure 4.1b. We can see that AGResNet can learn the transformation Φ from unpaired X and $Y = \Phi(X)$ both in terms of the masked area and noise level. Figure A.1 shows the forward and backward cycle step of all 102 neurons from a randomly selected sample, the diagonal augmentation can be clearly seen in the generated output $\hat{y} = G(x)$. Table 4.1a listed the mean absolute errors of the direct and cycle-consistent comparison of 200 test samples. Note that the identity model should have perfect cycle-consistent loss as $F(G(x))$ and $G(F(y))$ do not operate on the data. Nevertheless, both ResNet and AGResNet achieved better results than the identity model in the $|X - F(Y)|$ and $|Y - G(X)|$, despite the fact that Φ is a relatively simple augmentation and one would expect the difference between X and $\Phi(X)$ to be small. This suggests that the CycleGAN framework, along with our proposed generator architectures, can indeed learn subtle spatiotemporal transformations in calcium fluorescent signals.

Since $\Phi(X) = Y$ performed a systematic spatiotemporal transformation to X, one would expect the networks to learn features that focus on the augmented region of the data. We, therefore, use GradCAM [98] to visualize regions of interest learned by the discriminators. The localization map of discriminator $D_Y(Y)$ when given an augmented real sample $y = \Phi(x)$, shown in Figure 4.2b, demonstrates a high level of attention along the diagonal masked region on the input. This indicates that, intuitively, D_Y learned to distinguish whether or not a given sample is from distribution $Y = \Phi(X)$ by predominantly focusing on the edge of the masking area. On the other hand, no augmentation was done on the input to discriminator D_X, hence it should not be able to learn the distinct features as D_Y did. Figure 4.2a shows the GradCAM localization map of $D_X(X)$, which does not appear to have a particular structural area of focus at first. However, when we overlay the reward zones on the input (indicated by yellow-orange

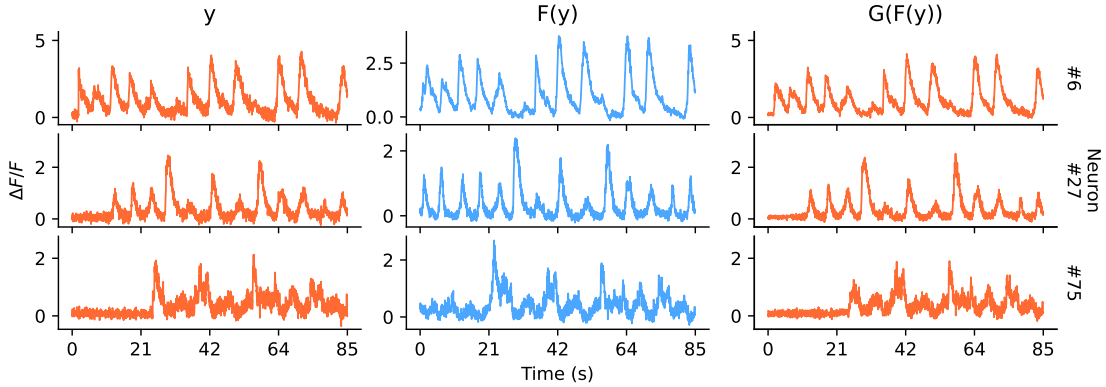
dotted lines), which were not provided in the training data or modelled in the training objectives, we can see that the focused regions are loosely aligned with the reward zones. Note that reward zones are external task-relevant regions that are expected to shape the neural activity in the primary visual cortex as the visual patterns change when the mouse enters the reward zone. Hence, suggesting that D_X could have learned distinctive patterns from neurons $\sim$ #5 $-$ #15 around the reward zones. We can also extract the learned sigmoid attention masks in AGResNet to visualize where in the input w.r.t the output that G and F were learning from. Figure 4.3 shows the sigmoid mask in $\texttt{AG}_1$ and $\texttt{AG}_2$ from G and F when given real input x and $y = \Phi(x)$. AG_1 and AG_2 in G showed that they ignored the augmentation region (bottom left corner), whereas F paid slightly higher attention to the masked area w.r.t the to the rest of the area. These are reasonable patterns as the augmented region in the generated output $\hat{y} = G(x)$ should be replaced by noise and not dependent on x, hence can be ignored. Conversely, F would pay more attention to the masked region in $y = \Phi(x)$ so that it could extract more relevant information to replace those area with signals.

Next, we evaluate how well different common GANs objective functions can cope with the CycleGAN framework. To mitigate the issues of vanishing gradient and mode collapse, WGANGP [5] and DRAGAN [58] both utilize gradient penalty regularization to enforce the 1-Lipschitz condition in the discriminator, and such method has been the go-to approach in unconditional GANs [50, 118, 82]. Nevertheless, in addition to the increase of computation cost, the gradient penalty term could further complicate the already perplexing CycleGAN objectives. To this end, we compare results of CycleGAN with AGResNet generators trained using the GAN [38], LSGAN [70], WGANGP [5] and DRAGAN [58] objectives. Based on our experiments with the synthetic data, LSGAN achieved slightly better results than GAN while both performed better than identity (see Table 4.1b). Interestingly, the two gradient penalty methods had lower cycle-consistent errors than GAN and LSGAN yet performed significantly worse in the $|X - F(Y)|$ and $|Y - G(X)|$ comparisons. This suggests that the discriminators could be overpowered by the generators when trained with WGANGP and DRAGAN, in which $D_X(F(Y))$ and $D_Y(G(X))$ are neither informative nor influential to the overall objective thus the intermediate outputs $\hat{Y} = G(X)$ and $\hat{X} = F(Y)$ are not as important. This allows the generators to focus on optimizing the cycle-consistent loss. Further investigation in the relationship between sensory data and different objective formulations in CycleGAN is needed. Nevertheless, thanks to its lower computational cost and better performance,

we use the LSGAN objective formulation, along with AGResNet as the generator architecture, for the remainder of this work unless otherwise stated.



(a) Forward cycle: $X \to Y \to X$

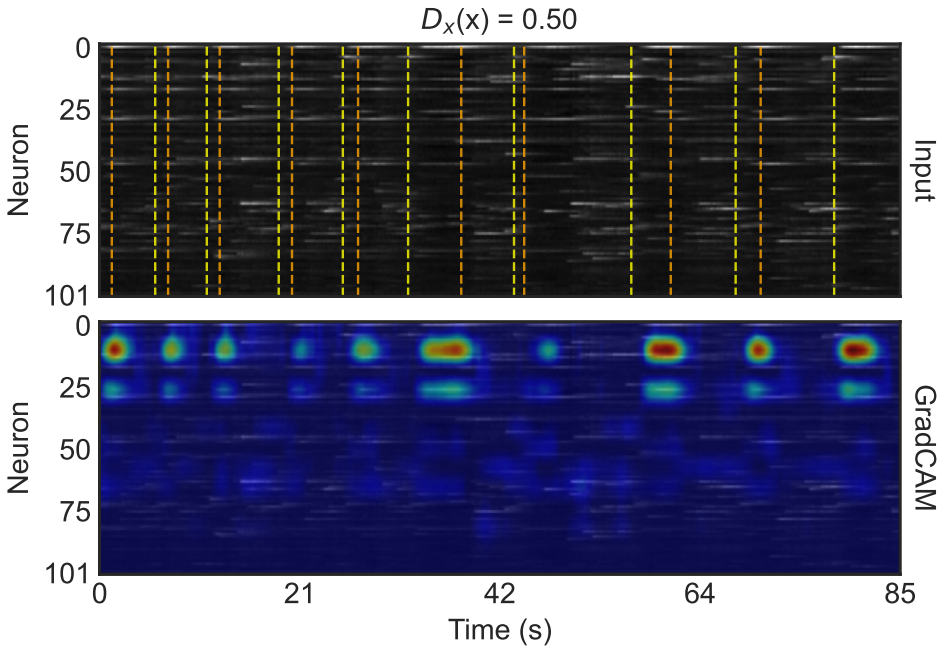


(b) Backward cycle: $Y \to X \to Y$

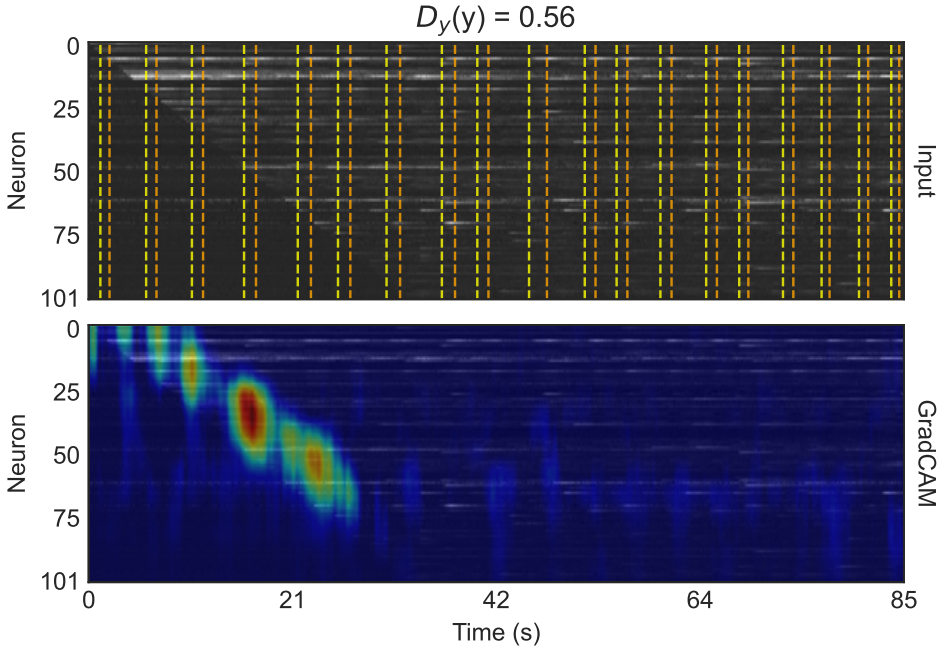
Figure 4.1: (a) forward and (b) backward cycle of neuron 6, 27 and 75 from AGResNet trained with LSGAN. Since X and Y in the test set are paired, we would expect $x \approx F(y)$ and $y \approx G(x)$ here (see Section 3.1.3). Notice that neurons with a higher index (e.g. neuron 75 in y) would have more units being masked out and replaced by noise. We can see that generator F was able to reconstruct the masked out regions (e.g. neuron 27 and 75 in $F(y)$) that resemble the traces in X.

	$\lvert X - F(Y)\rvert$	$\lvert X - F(G(X))\rvert$	$\lvert Y - G(X)\rvert$	$\lvert Y - G(F(Y))\rvert$
(A) DIFFERENT MODELS WITH LSGAN				
IDENTITY	0.247 ± 0.023	0	0.247 ± 0.023	0
RESNET	0.139 ± 0.012	0.107 ± 0.008	0.159 ± 0.022	0.088 ± 0.004
AGRESNET	$\mathbf{0.102 \pm 0.023}$	$\mathbf{0.056 \pm 0.003}$	$\mathbf{0.129 \pm 0.027}$	$\mathbf{0.068 \pm 0.008}$
(B) DIFFERENT OBJECTIVES WITH AGRESNET				
GAN	0.138 ± 0.014	0.073 ± 0.004	0.156 ± 0.022	0.118 ± 0.007
LSGAN	$\mathbf{0.102 \pm 0.023}$	0.056 ± 0.003	$\mathbf{0.129 \pm 0.027}$	0.068 ± 0.008
WGANGP	0.398 ± 0.053	$\mathbf{0.026 \pm 0.016}$	0.327 ± 0.013	0.052 ± 0.017
DRAGAN	0.377 ± 0.029	0.027 ± 0.003	0.258 ± 0.012	$\mathbf{0.048 \pm 0.003}$

Table 4.1: Mean absolute error similarity comparison of synthetic and generated calcium signals. (a) results from identity, ResNet and AGResNet trained with LSGAN objective and (b) results from AGResNet trained with GAN, LSGAN, WGANGP, DRAGAN objectives. Note that we can directly compare $X - F(X)$ and $Y - G(X)$ as the test set of X and $Y = \Phi(X)$ are paired. Lowest non-zero value in each category marked in bold.

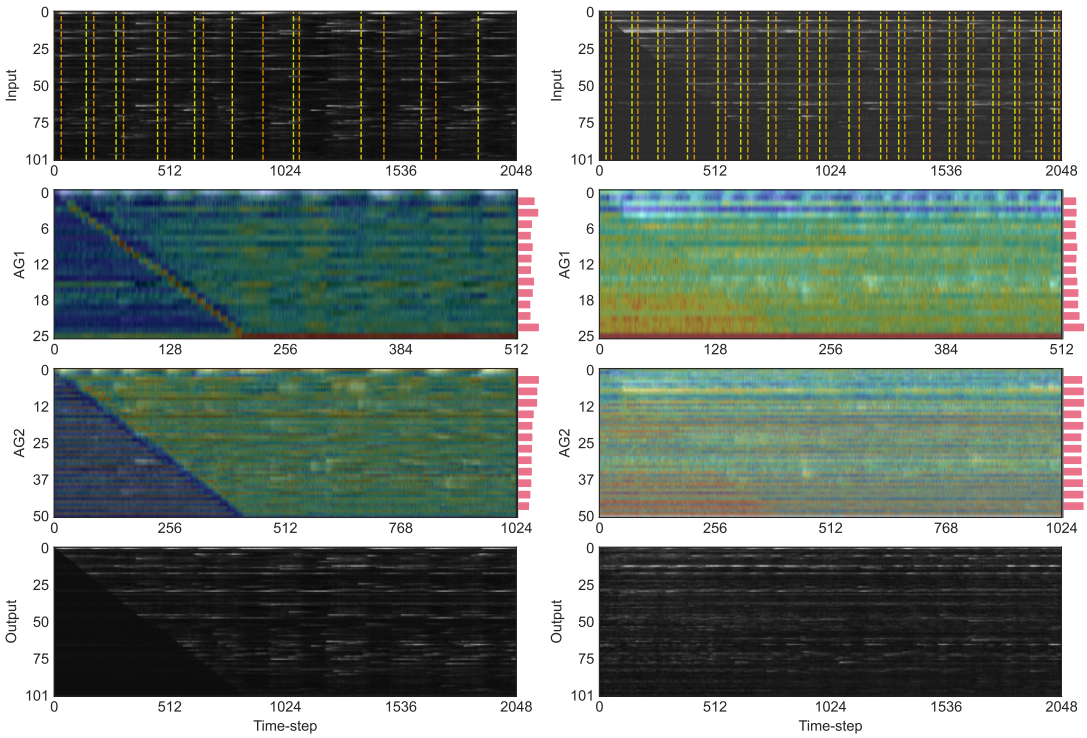


(a) GradCAM localization map of $D_X(x)$



(b) GradCAM localization map of $D_Y(y)$

Figure 4.2: GradCAM localization maps of (a) $D_X(x)$ and (b) $D_Y(y)$ given a randomly select $x \sim X$ and $y \sim \Phi(X)$. The top panel shows the input to the discriminator, where yellow and orange dotted lines mark the start and end of each reward zones. The second panel shows the GradCAM localization map superimposed on the input. Both discriminators were not certain with their predictions, with scores $D_X(x) = 0.5$ and $D_Y(y) = 0.56$. This suggests that the generators are generating compelling samples that the discriminators are not able to distinguish generated samples from real samples. Note that trial information such as reward zone locations were not provided to the networks, the pattern observed here was learned by the models themselves.



(a) AG sigmoid mask from $G(x)$ (b) AG sigmoid mask from $F(y)$

Figure 4.3: Attention masks in AGResNet from (a) $G(x)$ and (b) $F(y)$ with $x \sim X$ and $y \sim Y = \Phi(X)$ on the synthetic dataset (see Section 3.1.3). We can see that $G : X \rightarrow Y = \Phi(X)$ ignored the augmented region (bottom left triangle) in both latent dimensions, as it was not informative to the translation process. Whereas $F : Y = \Phi(X) \rightarrow X$ paid slightly more attention (w.r.t to the rest of the area) in the augmented region, which it had to fill with signals. Each column from top to bottom shows: the input, the sigmoid mask in AG_1 and AG_2 superimposed with the input to the respective AG, and the model output. The histogram on the right side of each AG panel shows spatial attention. Notice that dimensions in AG_1 and AG_2 are different from the original input. Dotted yellow and orange lines on the input panel mark the beginning and end of the reward zone in each trial.

4.2 Recorded data

As our proposed method has shown great results in learning the unpaired transformations in the synthetic dataset, we now move on to the data recorded from the mouse virtual-corridor experiment (see in Section 2.1.1) where we attempt to learn the unknown mapping between pre-learning and post-learning neuronal activity. Since the LSGAN objective formulation and AGResNet generator architecture achieved the best results in Section 4.1, we continue to use this combination on the recorded data.

Figure A.2 shows the forward and backward cycle traces of 3 neurons from a randomly selected sample, whereas Figure A.3 plots the entire population as a gray-scale image. Visually, G and F seems to be able to reconstruct $\bar{x} = F(G(x))$ and $\bar{y} = G(F(y))$. Based on the intermediate step $\hat{y} = G(x)$ and $\hat{x} = F(y)$, the two generators were not simply passing through x and y that optimize for the cycle-consistent loss. Nevertheless, unlike images where one could visually inspect the generated output, further statistical comparisons are needed for neuronal activity. Here, we first compare the generated calcium florescent signals with the recorded test set data. The cycle-consistent loss on the test set achieved a value of $|X - F(G(X))| = 0.092$ in the forward cycle step and $|Y - G(F(Y))| = 0.101$ in the backward cycle step. The identity loss, which measures the changes to the input when the opposite transformation is applied, is also almost perfect. For reference, the mean absolute differences between X and Y is $|X - Y| = 0.370$, and that both cycle-consistent and identity loss achieved significantly better error than that. This suggest G and F are not simply passing through the data without any processing. In addition, the low identity loss indicates that the generators can correctly identify whether or not the given input is already part of its target distribution. Table 4.2 reports the mean and standard deviation of the calcium signals comparisons.

To better analysis the two intermediate transformations $\hat{y} = G(x)$ and $\hat{x} = F(y)$, and show that G and F properly converted x and y into their respective distributions $\hat{y} \sim Y$ and $\hat{x} \sim X$, we compute and compare a set of spike trains statistics. Firing rate, pairwise correlation and pairwise van Rossum distance are spike statistics are commonly used in neuroscience to characterize a spike train. As such, the distributions of these statistics from the generated data should resemble the recorded data. We used the deconvolution toolbox Cascade [94] to infer spike trains from the calcium fluorescent signals for all 6 data distributions: X, Y, $G(X)$, $F(Y)$, $F(G(X))$ and $G(F(Y))$, we

can then the distributions in the following combinations: $X \mid F(Y)$, $X \mid F(G(X))$, $Y \mid G(X)$ and $Y \mid G(F(Y))$. The results of the mentioned spike trains statistics are shown in Figures A.4, A.5, A.6 and A.7. The firing rates and pairwise correlation distributions from $F(Y)$, $F(G(X))$, $G(X)$ and $G(F(Y))$ can closely match the recorded X and Y distributions. In addition, for certain neurons in the van Rossum distance heatmaps for $X \mid F(Y)$ and $Y \mid G(X)$ (e.g. neuron #6 and #27 in $X \mid F(Y)$), we can observe a diagonal line of low-intensity values. Hence, there exists a spike train in X and Y that corresponds to a generated spike train in $F(Y)$ and $G(X)$ respectively. Though this pattern did not exhibit in all neurons, this could be due to the fact that we shuffle both X and Y independently, where the corresponding activities for certain neurons were not in the test set, hence cannot be shown in the heatmap comparison. Nonetheless, to quantify the generation performance, Table 4.3 shows the average KL divergence of the spike statistics comparison. Overall, all 3 metrics used have achieved a low average KL divergence value, with the highest value being the firing rate of $KL(X, F(G(X))) = 1.396 \pm 1.265$. This indicates that the generators can indeed learn the distribution transformation from pre-learning to post-learning neuronal activities, and vice-versa.

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-learning and post-learning data could be more challenging as there would not be obvious patterns in the inputs to anticipate. Nevertheless, we would expect a higher level of activities in the V1 neurons when the rodent is about to enter or inside the reward zone, where the grating pattern on the virtual wall turns dark. Therefore, we suspect that the generators and discriminators would pay more attention to those areas with distinct neuronal activities. We first visualize the sigmoid attention masks in AGResNet. Note that the two AG modules operated at latent representations with dimensions that were 4 times and 2 times lower than the original input (i.e. the attention masks of $\texttt{AG}_1$ and $\texttt{AG}_2$ have shapes $(512, 26, 1)$ and $(1024, 512, 1)$ when the input shape is $(2048, 102, 1)$), hence we could not simply align the reward zones with the learned masks. Figure 4.4 shows the learned attention masks of $G(x)$ and $F(y)$ superimposed with the latent inputs of the corresponding AG. Both $\texttt{AG}_1$ masks in G and F appear to focus on neurons with higher index, whereas the spatial attention distributions were even in $\texttt{AG}_2$. Nevertheless, we were unable to identify clear patterns in the attention masks as we did with the synthetic

dataset.

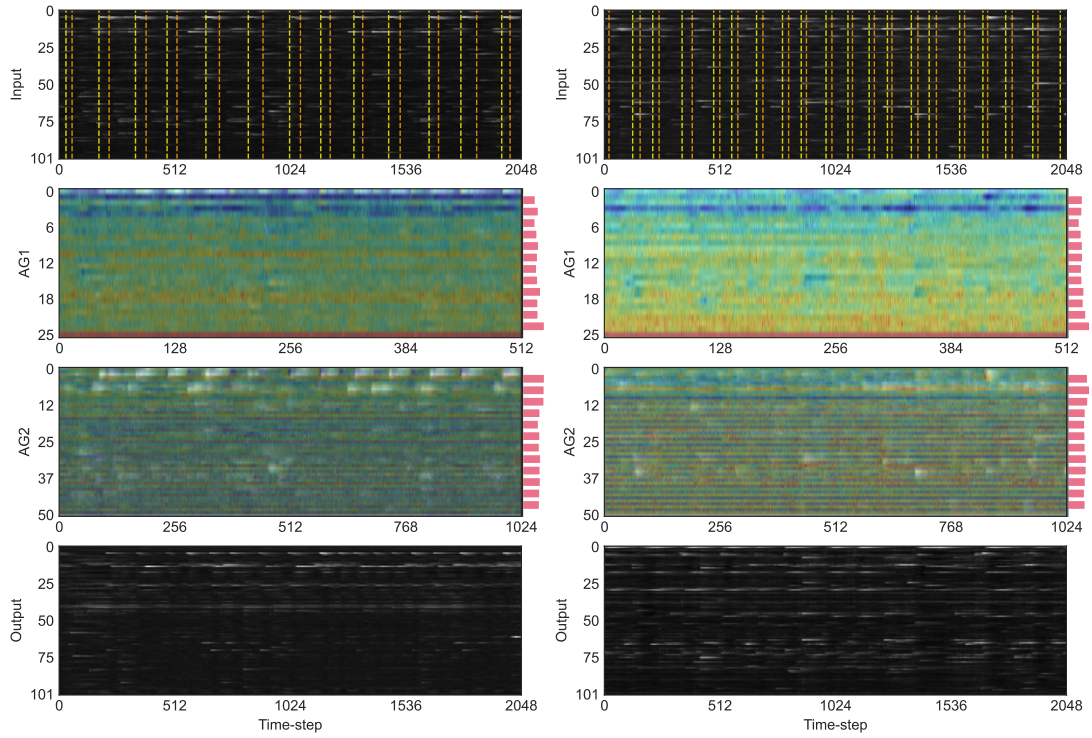
Since GradCAM projects activation values w.r.t to the input, instead of the latent representations, we could gain more insights about the learned features from the localization maps. Figure 4.5 shows the GradCAM localization maps for $D_X(x)$ and $D_Y(y)$ with a randomly selected $x \sim X$ and $y \sim Y$. We observed some regions of high attention around the reward zones in $D_X(x)$, albeit not prominent and even less so with $D_Y(y)$. To better visualize the relationship between area of focus learned by the model and the virtual-environment experiment, we aggregated the attention values in GradCAM w.r.t the virtual location of the rodent over 200 test samples (which contain about 30 trials). The results of such positional attention maps are shown in Figure 4.6. Both $D_X(X)$ and $D_Y(Y)$ had higher level of attention towards to first ~ 40 neurons. In which neurons $\sim \#0 - \#30$ in X received strong attention within the reward zone in D_X. Whereas the attention from $D_Y(Y)$ were more sparse, with above-average attention around 40 - 80cm then at the end of the reward zone. Keep in mind that the only objective the discriminators had was to distinguish if a given sample is from a particular distribution. Thereby it is possible that the discriminators were learning trivial features, instead of to the virtual-environment task. Since the positional attention maps were able to uncover these interesting patterns, we then extract the maps from $G(X)$ and $F(Y)$ following the same procedure. Figure 4.7 shows the positional attention maps for the two generators. Interestingly, generator G also paid high level of attention around the reward zone and in neurons located at the top part of the input, specifically neuron $\sim \#8$, similar to the attention map obtained from D_X. On the other hand, the attention map from $F(Y)$ indicates that the activities from neurons $\sim \#30 - \#80$ at the beginning of the reward zone were highly influential in its transformation process. Note that no trial information was incorporated into the training data nor was it formulated in the objective function. Hence, these interesting patterns we observe here were learned entirely by the networks themselves via the adversarial process.

MODEL	$\lvert X - F(G(X))\rvert$	$\lvert X - F(X)\rvert$	$\lvert Y - G(F(Y))\rvert$	$\lvert Y - G(Y)\rvert$
AGRESNET	0.092 ± 0.051	0.024 ± 0.020	0.101 ± 0.081	0.025 ± 0.011

Table 4.2: Cycle-consistent and identity loss of `AGResNet` on test set. For reference, $|X - Y| = 0.370 \pm 0.027$.

MODEL	$\mathrm{KL}(X, F(Y))$	$\mathrm{KL}(X, F(G(X)))$	$\mathrm{KL}(Y, G(X))$	$\mathrm{KL}(Y, G(F(Y)))$
		(A) **FIRING RATE**		
IDENTITY	8.131 ± 6.600	**0**	7.533 ± 6.754	**0**
AGRESNET	$\mathbf{1.191 \pm 1.077}$	1.396 ± 1.265	$\mathbf{1.336 \pm 0.979}$	1.320 ± 1.359
		(B) **PAIRWISE CORRELATION**		
IDENTITY	0.880 ± 0.531	**0**	0.824 ± 0.472	**0**
AGRESNET	$\mathbf{0.149 \pm 0.074}$	0.034 ± 0.023	$\mathbf{0.055 \pm 0.045}$	0.024 ± 0.018
		(C) **PAIRWISE VAN ROSSUM DISTANCE**		
IDENTITY	0.544 ± 0.297	**0**	$\mathbf{0.308 \pm 0.114}$	**0**
AGRESNET	$\mathbf{0.253 \pm 0.173}$	0.108 ± 0.037	0.333 ± 0.168	0.152 ± 0.063

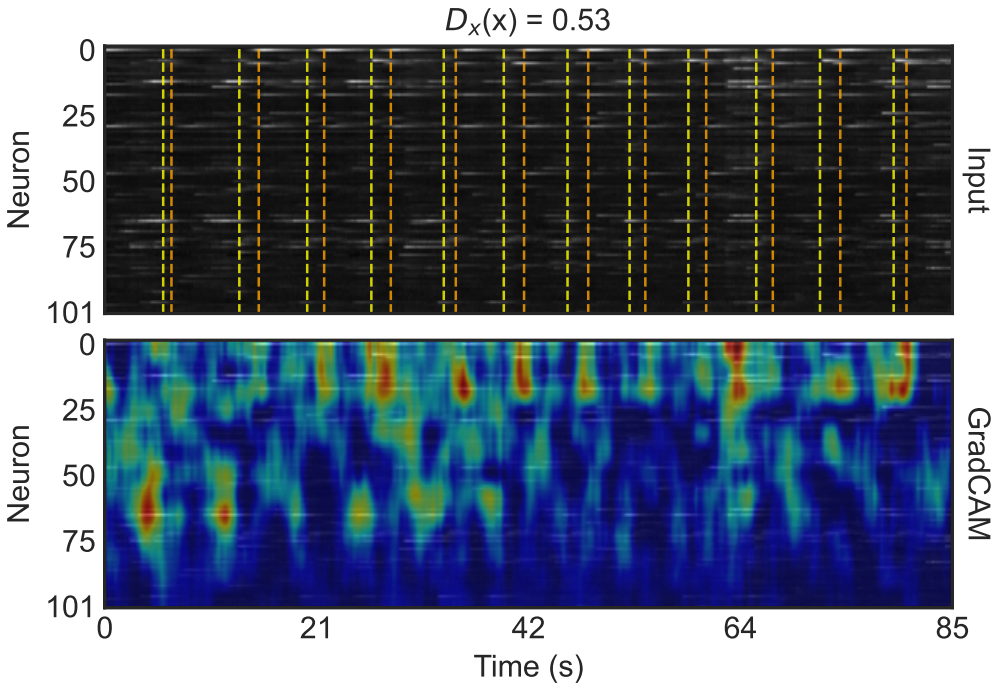
Table 4.3: Average KL divergence of (a) firing rate, (b) population pairwise correlation and (c) population pairwise van Rossum distance from inferred spike trains. Note that the identity model is added here as a baseline comparison and should obtain perfect cycle reconstruction. The per-neuron and per-trial distribution comparisons are shown in Appendix A.2.1. Entries with the lowest value are marked in bold.



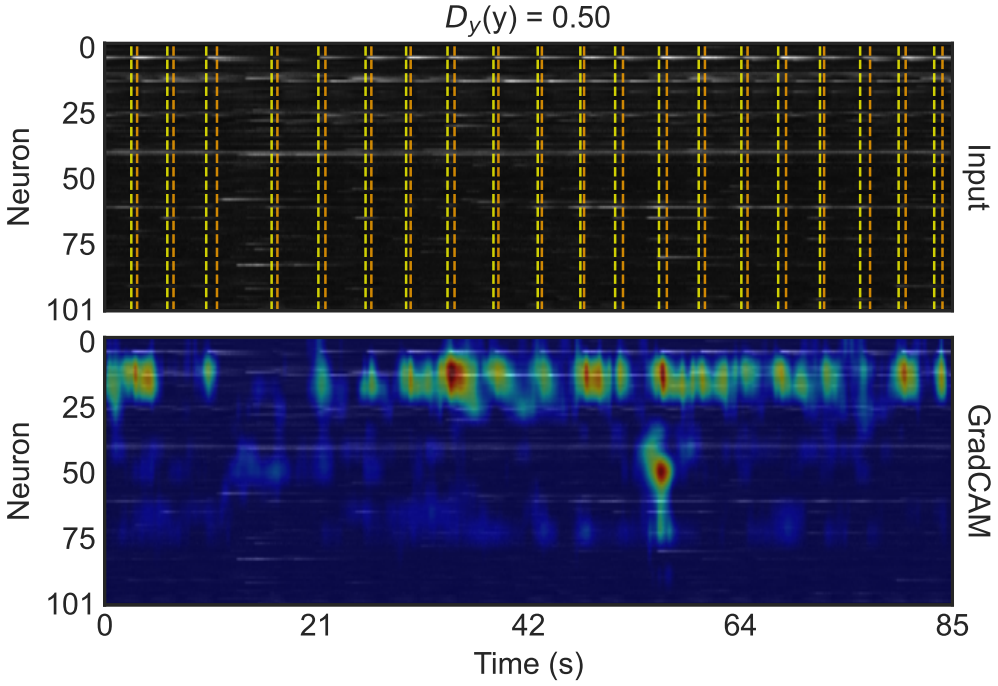
(a) AG sigmoid mask in $G(x)$

(b) AG sigmoid mask in $F(y)$

Figure 4.4: Attention masks in AGResNet from (a) $G(x)$ and (b) $F(y)$ with $x \sim X$ and $y \sim Y$ on the recorded dataset. No noticeable patterns were obtained from the learned attention masks in G and F. Each column from top to bottom shows: the input, the sigmoid mask in AG_1 and AG_2 superimposed with the input to the respective AG, and the model output. The histogram on the right side of each AG panel shows spatial attention. Notice that dimensions in AG_1 and AG_2 are different from the original input. Dotted yellow and orange lines on the input panel mark the beginning and end of the reward zone in each trial.

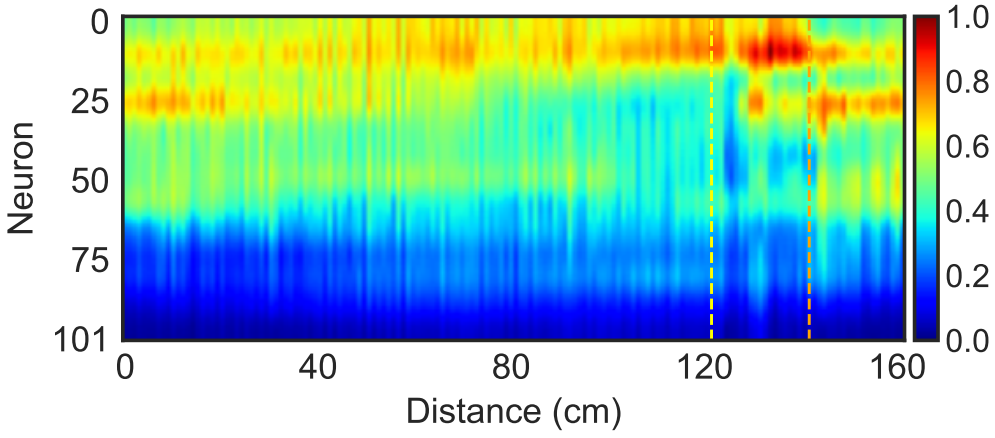


(a) GradCAM localization map of $D_X(x)$

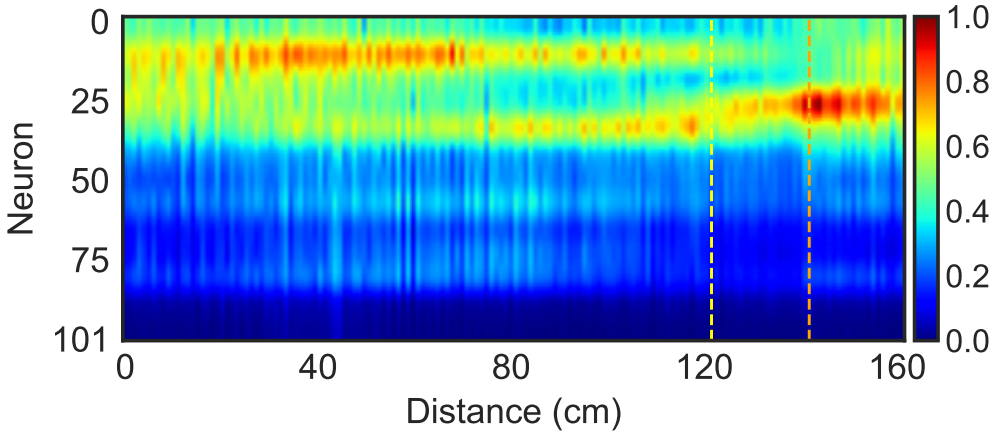


(b) GradCAM localization map of $D_Y(y)$

Figure 4.5: GradCAM localization maps of (a) $D_X(x)$ and (b) $D_Y(y)$ given a randomly select $x \sim X$ and $y \sim Y$. The top panel shows the input to the discriminator, where yellow and orange dotted lines mark the start and end of each reward zones. The second panel shows the GradCAM map superimposed with the input. The discriminators were not able to make a decisive prediction, with a score of $D_X(x) = 0.53$ and $D_Y(y) = 0.50$ (real samples should have a score of 1). This would suggest that the generators have been generating compelling samples such that the discriminators were not able distinguish them. However, this could also harm the generators as they no longer receive informative critics on their predictions.

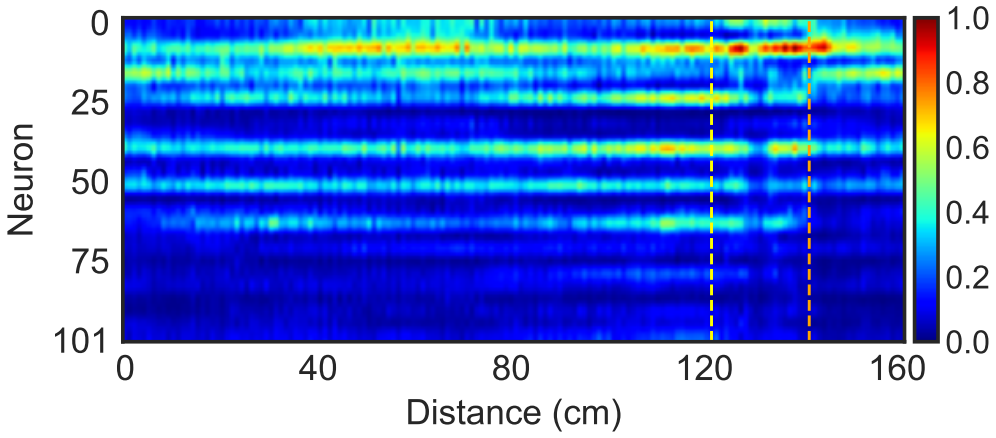


(a) $D_X(X)$ positional attention map

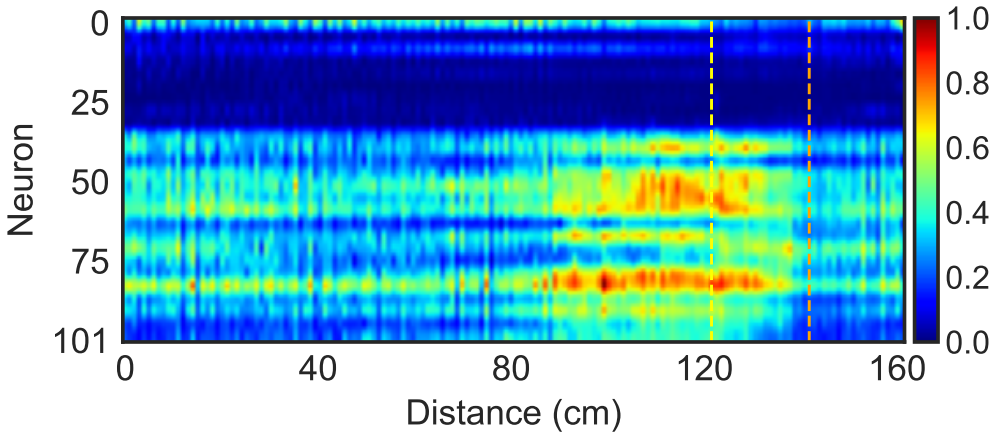


(b) $D_Y(Y)$ positional attention map

Figure 4.6: Positional attention maps of (a) $D_X(X)$ and (b) $D_Y(Y)$ w.r.t virtual distance in the animal experiment. Overall, the two discriminators were focusing on neurons with index $\sim \#0 - \#40$, with higher level of attention around the reward zone. These positional activation maps were generated by computing the GradCAM activation for all 200 test samples then average the activation value for each neuron and each position in the virtual environment (160cm in total). Yellow and orange dotted lines indicate the reward zone, which is located at 120cm to 140cm in each trial.



(a) $G(X)$ positional attention map



(b) $F(Y)$ positional attention map

Figure 4.7: Positional attention maps of (a) $G(X)$ and (b) $F(Y)$ w.r.t virtual distance in the animal experiment. Generator G had high attention specifically in neuron $\sim$ #8, within the reward zone in particular. Whereas F were focusing on neurons $\sim \#30 - \#80$ in the area before and during the reward zone. These positional activation maps were generated by computing the GradCAM activation for all 200 test samples then average the activation value for each neuron and each position in the virtual environment (160cm in total). Yellow and orange dotted lines indicate the reward zone, which is located at 120cm to 140cm in each trial.

4.2.1 Neuron spatial order

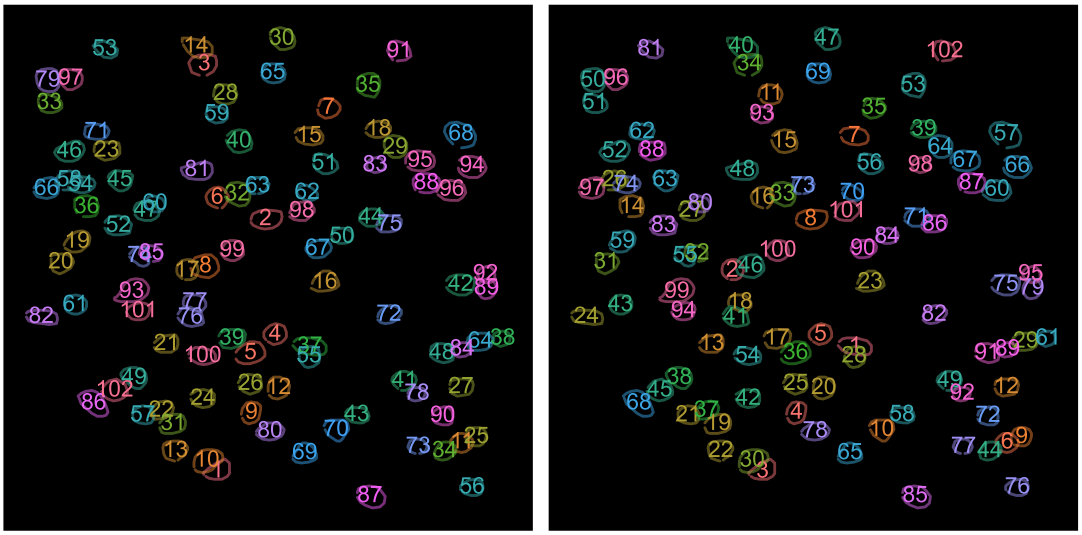
With the promising results shown in the synthetic and recorded datasets, we move on to investigate whether or not the neuron spatial orders can affect the performances of the networks. As discussed in Section 3.1.2, convolutional layers in DNNs have limited receptive field, that is, the region that the network can "see" in an instance is restricted by the kernel size. Hence, we propose that sorting the index of neurons in a systemic manner could improve the network's ability to learn spatiotemporal information from neuronal activities. To this end, we test two sorting approaches to order neurons in the data matrix: 1) firing rate in descending order, 2) deep autoencoder reconstruction loss in ascending order.

In the first approach, we simply sort the neurons by their firing rate in the train set where the neuron with the highest firing rate would be set to index 0 in the data matrix. With the second approach, we first trained two deep autoencoders $\texttt{AE}_X$ and $\texttt{AE}_Y$ which learn to reconstruct X and Y individually. We then use the reconstruction loss in the test set to rank the neurons where the neuron with the lowest reconstruction loss would be set to index 0. By doing so, we expect that the deep autoencoders would perform better on neurons with prominent activities (i.e. neurons that are more influential to the behaviour task) hence grouping relevant neurons together along the first axis in the data matrix. Figure 4.8 shows the neuron orders of X and Y based on the reconstruction loss from $\texttt{AE}_X$ and $\texttt{AE}_Y$. Notice that X and Y were sorted differently despite having the same original annotation order, though we did not observe any obvious spatial patterns in the ordering. Interestingly, we observed that the autoencoders performed better with low-firing neurons initially. And as the training progress, the models learned to reconstruct more active neurons better. Figure A.8 and Figure A.9 show the neuron spatial orders from $\texttt{AE}_X$ and $\texttt{AE}_Y$ at epoch 1, 50 and 10. In which the spiking distribution along the spatial dimension was quite even at epoch 1, then the general trend shifts upward as we train the model. Yet, the distribution histograms indicated that the neurons were not ordered by spiking activities entirely; for instance, some neurons in Y had higher firing rates though lower ranks in Figure A.9c. This suggests that the deep autoencoders were also learning other features from the neuronal activities rather than purely on spiking patterns.

Following the same training and validation procedure in Section 4.2, we have re-trained the AGResNet models with data sorted according to the firing rate, as well as autoencoder reconstruction loss. Note that to ensure a fair comparison, neurons were

re-arranged to their original order prior to any metrics calculation. Table 4.4 reports the cycle-consistent and identity calcium signals comparisons, and Table 4.5 reports the average KL divergence in the spike trains statistics. Across the board, models trained with ordered neurons achieved better results. Moreover, sorting neurons based on the reconstruction loss also outperformed the firing rate approach in most metrics, with the exception of the backward cycle-consistent loss $|Y - G(F(Y))|$ and pairwise van Rossum distance in $KL(X, F(G(X)))$, though the differences between the sorting approaches in those two statistics were less than 4%.

With neurons that are more influential being ranked higher in the spatial order, we would expect the attention of the generators also focus on the top region of the inputs, instead of the broad attention shown in Figure 4.4. Indeed, the learned attention masks shown in Figure 4.9 display an increase of focus onto the top area of their respectively latent inputs for both G and F. Hence, suggesting that ordering neurons that are correlated to be close in space can indeed improve the performance of convolutional-based DNNs. Moreover, the deep autoencoder approach could be an effective and data-driven method to learn such correlation.



(a) X annotation order (b) Y annotation order

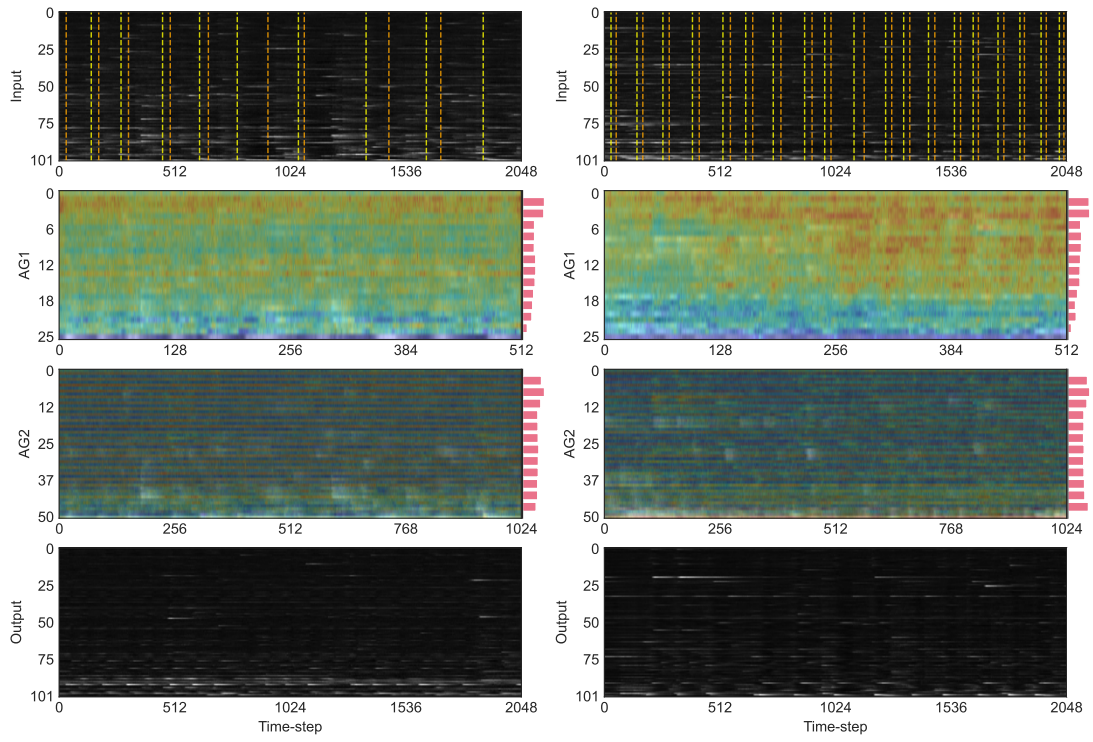
Figure 4.8: Annotation order of (a) X and (b) Y sorted by the reconstructions error from deep autoencoders $\texttt{AE}_X$ and $\texttt{AE}_Y$ (see Section 3.1.2). Figure 2.1b shows the original annotation order.

SORT BY	$\lvert X - F(G(X))\rvert$	$\lvert X - F(X)\rvert$	$\lvert Y - G(F(Y))\rvert$	$\lvert Y - G(Y)\rvert$
NONE	0.092 ± 0.051	0.024 ± 0.020	0.101 ± 0.081	0.025 ± 0.011
FIRING RATE	0.078 ± 0.027	0.011 ± 0.005	$\mathbf{0.062 \pm 0.015}$	0.017 ± 0.009
AUTOENCODER	$\mathbf{0.066 \pm 0.007}$	$\mathbf{0.010 \pm 0.001}$	0.064 ± 0.004	$\mathbf{0.009 \pm 0.001}$

Table 4.4: Cycle-consistent and identity loss of `AGResNet` with neurons ordered by 1) no order, 2) firing rate and 3) autoencoder reconstruction loss. The lowest loss in each category marked in bold.

SORT BY	$\mathrm{KL}(X, F(Y))$	$\mathrm{KL}(X, F(G(X)))$	$\mathrm{KL}(Y, G(X))$	$\mathrm{KL}(Y, G(F(Y)))$
(A) **FIRING RATE**				
NONE	1.191 ± 1.077	1.396 ± 1.265	1.336 ± 0.979	1.320 ± 1.359
FIRING RATE	1.159 ± 0.854	1.345 ± 1.254	1.218 ± 1.067	1.123 ± 0.890
AUTOENCODER	$\mathbf{1.156 \pm 0.934}$	$\mathbf{1.289 \pm 1.230}$	$\mathbf{1.148 \pm 0.772}$	$\mathbf{1.116 \pm 0.877}$
(B) **PAIRWISE CORRELATION**				
NONE	$0.149 \pm 0.0.074$	0.034 ± 0.023	0.055 ± 0.045	0.024 ± 0.018
FIRING RATE	0.047 ± 0.034	$\mathbf{0.030 \pm 0.019}$	0.051 ± 0.049	0.024 ± 0.019
AUTOENCODER	$\mathbf{0.042 \pm 0.025}$	$\mathbf{0.030 \pm 0.014}$	$\mathbf{0.047 \pm 0.043}$	$\mathbf{0.017 \pm 0.013}$
(C) **PAIRWISE VAN ROSSUM DISTANCE**				
NONE	0.253 ± 0.173	0.108 ± 0.037	0.333 ± 0.168	0.152 ± 0.063
FIRING RATE	0.252 ± 0.198	$\mathbf{0.097 \pm 0.032}$	0.310 ± 0.127	0.139 ± 0.052
AUTOENCODER	$\mathbf{0.237 \pm 0.162}$	0.100 ± 0.043	$\mathbf{0.308 \pm 0.122}$	$\mathbf{0.128 \pm 0.058}$

Table 4.5: KL-divergence comparison inferred spike trains from between recorded and generated data of neurons ordered by 1) no order, 2) firing rate and 3) autoencoder reconstruction loss. Entries with lowest average KL divergence are marked in bold.



(a) AG sigmoid mask in $G(x)$ (b) AG sigmoid mask in $F(g)$

Figure 4.9: Attention masks in AGResNet from (a) $G(x)$ and (b) $F(y)$ with $x \sim X$ and $y \sim Y$ where both X and Y were pre-sorted with autoencoder reconstruction loss (see Section 3.1.2). that was pre-sorted with autoencoder reconstruction loss. As compared to the model trained without a particular ordering (i.e. Figure 4.4), the spatial attention (represented by horizontal histograms) here showed that both G and F shifted their attention toward the top part of the inputs where relevant neurons were located. Along with the better results in calcium signals comparisons and spike statistics (see Table 4.4 and 4.5), the attention maps suggest that convolutional-based models can learn better features when relevant neurons are grouped together in space. Each column from top to bottom shows the input, the sigmoid mask in AG_1 and AG_2 superimposed with the input to the respective AG, and the model output. Notice that dimensions in AG_1 and AG_2 are different from the original input. Dotted yellow and orange lines on the input panel mark the start and end of the reward zone in each trial.

Chapter 5

Discussion

Understanding the learning process in the brain has been one of the main goals in the field of computational neuroscience, though extracting an unbiased and interpretable description of the high-dimensional neural responses in the course of learning is an arduous challenge. In this work, we demonstrated that the cycle-consistent adversarial networks [132] framework is a capable data-driven method to model the transformation between pre-learning and post-learning *in vivo* neuronal activities.

To verify that our method can indeed learn subtle changes in two unpaired distributions of calcium fluorescent signals, we first evaluate our framework with a synthetic dataset. Using the neuronal activities recorded in the primary visual cortex of a behaving mouse, we applied a spatiotemporal augmentation to the signals, hence resulting in two seemingly unaligned distributions where we could reverse the transformation and compare the generated samples directly. We showed that the generated samples from the CycleGAN framework could model both spatiotemporal transformation and added noise in the synthetic dataset with almost perfect reconstructions. Then, we fitted our model with data collected on the first and fourth day of the mouse virtual-environment experiment, which represents pre-learning and post-learning cellular activities. In addition to the calcium signals cycle-consistency comparison, we also compared 3 commonly used spike trains statistics, including firing rate, pairwise correlation and pairwise van Rossum distance. We demonstrated that the generated samples could closely match the spatiotemporal first and second-order statistics of the respective target distribution.

Ultimately, we are interested in interpreting what and how neuronal activities evolve

with experience, and the two visualization methods we employed have shown interesting patterns learned by the generators and discriminators. First, we incorporated GradCAM [98] into our framework, which generates localization maps based on the gradient of the model to highlight areas of interest w.r.t to the input. Second, based on the ResNet architecture [43] and Attention Gate module [80], we introduced a self-attention generator architecture – AGResNet. The self-attention mechanism enabled the generators to self-adjust their level of attention w.r.t the latent representation of the input. With the synthetic dataset, we observed a high level of attention around the augmented region in the artificial data, which is reasonable as the spatiotemporal transformation and the order of the data were the only two differences in the otherwise paired domains. Whereas in the recorded dataset, the extracted attention maps showed that the networks focused on areas surrounding the reward zones. This suggests the networks were able to self-identify activities around the reward zones that are most relevant in their respective translation process. These findings were closely aligned with our expectations where neural activities are shaped by the external task, in this case, in the reward zones where the visual patterns changed and the rodent would be rewarded given the right action. Importantly, we did not provide any trial information into the training data nor incorporated them into the training objectives. Despite that, the self-attention mechanism along with the CycleGAN framework was able to self-identify this information.

We believe this work is beneficial to both the computational neuroscience community and the machine learning community. We demonstrated that CycleGAN is an effective, and more importantly, data-driven method to model complex and high-dimensional neuronal activities. Visualization methods such as GradCAM and self-attention networks enable practitioners to understand what is being learned by a deep neural network, which is known as a black-box model as there is no direct to interpret its prediction [14, 40]. For instance, the positional attention maps, such as the ones presented in Figure 4.7, could guide experimentalists to conduct further analysis at specific neurons or locations with high attention values. Furthermore, we introduced a novel approach to pre-sort the spatial order of each neuron, such that neurons that are more relevant to each other are closer in space and enable more effective learning by CNNs. In our experiments, such preprocessing procedure saw measurable improvements in all metrics, and the self-attention masks showed that the generators were able to centralize their focus onto the top region of the input. Computational neuroscientists could improve the

performance of their convolutional-based deep learning models in neuronal activities related applications by applying this simple pre-sorting operation.

From the technical point of view, we have investigated various aspect of training very large deep generative models. We incorporated 5 popular GANs objective formulations into the CycleGAN framework, including GAN [38], LSGAN [70], WGANGP [5] and DRAGAN [58]. Our evaluation results have shown that the LSGAN formulation yield the generation performance, partly due to the fact that the inclusion of gradient penalty would further complicate the already complex discriminator objective formulation. In addition, we showed that techniques that are commonly used in vanilla GANs, such as label smoothing and two-time scale update rules, also work well in the CycleGAN framework.

5.1 Limitations

We would like to highlight a number of biases and limitations of our work. A significant portion of the neuronal activity analysis in Section 4.2 were performed in spike trains inferred from the recorded and generated calcium fluorescent signals. We used the deep learning ensemble deconvolution algorithm Cascade [94] to perform spike inference, which is a recently introduced method that has outperformed existing model-based algorithms. However, spike inference from fluorescent calcium indicators signals remains an active area of research [108]. For instance, Vanwalleghem et al. [113] demonstrated that spiking activities could be missed due to the implicit non-negative assumption in calcium imaging data which exists in many deconvolution algorithms, including Cascade. Nonetheless, we would like to emphasize that Cascade was used to deconvolve calcium signals for all distributions of data ergo all inferred spike trains experienced the same bias.

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 mapping. Relating back to the French-English translation example, the french sentence “comment ça va” has multiple correct translations, such as “how are you” but also “how is it going”. Hence, such diverse output cannot be modelled with the CycleGAN framework. Almahairi et al. [2] extended the framework to learn many-to-many mapping

by introducing auxiliary noise to the two distributions (i.e. $G : X \times Z_X \to Y$), thus able to generate outputs with variations. To make compatible with the cycle-consistency objective, the authors added two encoders to the framework, each learns to generate the auxiliary noise for their respective domains (i.e. $E_X : X \times Y \to Z_X$ for encoder E_X in X domain). Nevertheless, their method is most effective when trained in a semi-supervised manner which is not possible with neural activity as it is not possible to obtain paired data. Moreover, the two DNN-based encoders introduce additional trainable parameters, further increasing the already extensive computation cost.

5.2 Future Work

One promising research direction based on this work is to conduct further analysis on the learned features by the generators and discriminators. Notably, the aggregated positional attention plots (i.e. Figure 4.7 and 4.6) showed interesting attention structure around the reward zone, despite such information was not incorporated in the training data nor the objective function. Further analysis is needed to identify why the networks were focusing on those regions in particular, and more importantly, can the highlighted regions provide meaningful insights into the learning process in the brain.

Another interesting neuroscience question is to investigate the neural spatial ordering learned by the autoencoders described in Section 3.1.2. We obtained different neuron ordering in the pre-learning and post-learning data, suggesting that the level of importance in certain neurons changed in the course of learning. In addition, our current pre-sorting method produces a 1-dimensional spatial ordering, we can also consider 2-dimensional ordering and use 3-dimensional convolutional layers (instead of the current 2-dimensional) so that height and width spatial information can be considered.

From the machine learning perspective, one potential research direction is to explore contrastive learning in cellular activity. Loss functions and distances functions such as mean absolute error and cross-entropy loss are objectives commonly used in machine learning models, including our work. However, most of these pixel-wise (or element-wise) loss functions do not consider spatiotemporal information. If we shift a generated calcium signals $\bar{x} = F(G(y))$ forward or backward by 10 time-steps, then it is possible to obtain a better cycle-consistent loss (i.e. $\text{MAE}(x, \bar{x})$) with another sample that doesn't capture the latent dynamics. Chopra et al. [19] proposed contrastive loss to combat this limitation. Rather than a unit-wise comparsion between the output and target samples,

we use an encoder to learn a latent representation of the output and target and compare the two embeddings in latent space instead. Such formulation proved successful in many computer vision-related tasks [17, 54, 84, 126] and could allow us to formulate more expressive objectives in deep learning models for sensory data such as calcium imaging.

5.3 Conclusion

As a concluding remark, we have demonstrated the cycle-consistent adversarial networks are an effective and data-driven method to learn changes in neuronal activities over the course of learning. The proposed self-attention generator architecture and feature-importance visualization method provided intuitive approaches to interpret the learned features in deep neural networks. In addition, the novel neuron spatial ordering method enabled deep convolutional models to learn better representation from the neuronal activity. As deep unsupervised methods have become more expressive and explainable, and that neuronal activities in different learning phases from behaved animals have become more readily available, the intersection of the two fields has limitless potentials.

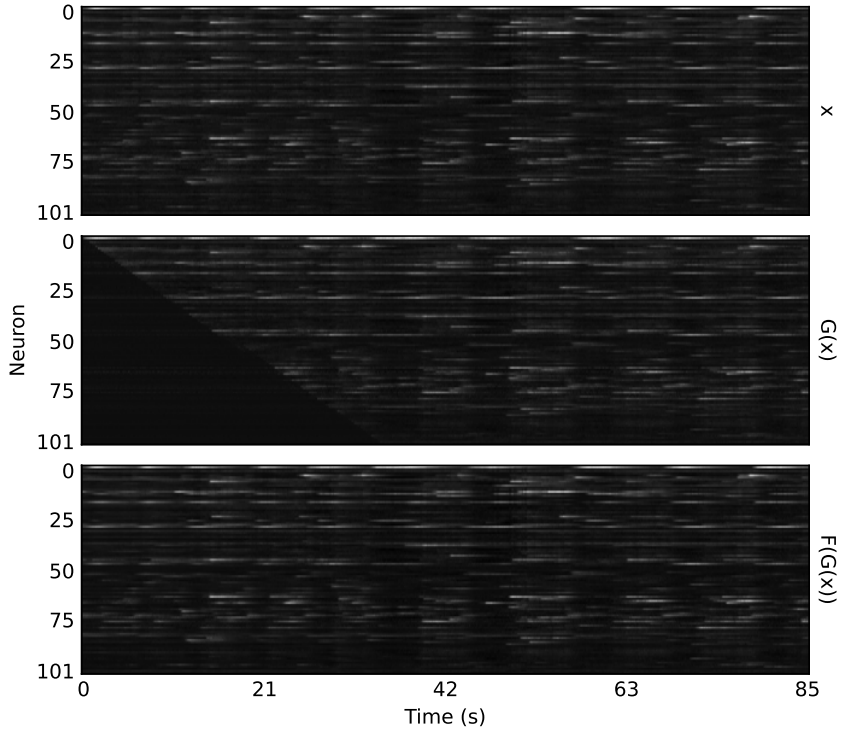
Appendix A

Appendix

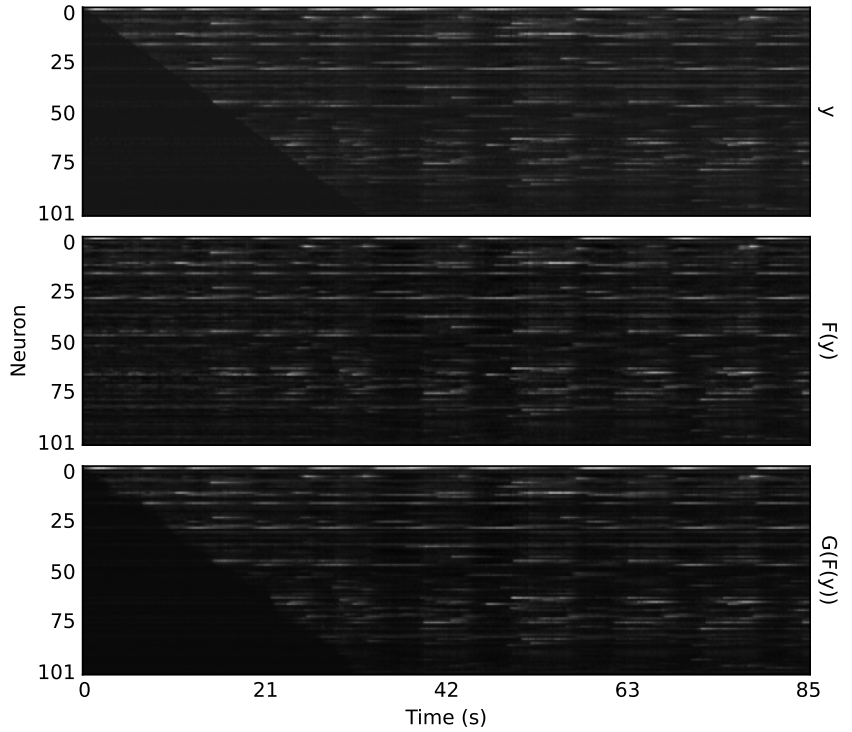
HYPER-PARAMETERS	GAN	LSGAN	WGANGP	DRAGAN
FILTERS	32			
KERNEL SIZE	4			
REDUCTION FACTOR	2			
ACTIVATION	LRELU			
NORMALIZATION	INSTANCENORM			
SPATIAL DROPOUT	0.25			
WEIGHT INITIALIZATION	RANDOM NORMAL $\mu = 0$ AND $\sigma = 0.02$			
λ_{CYCLE}	10			
$\lambda_{\text{IDENTITY}}$	5			
λ_{GP}	N/A	N/A	10	10
c	N/A	N/A	N/A	10
NUM. DIS UPDATE	1	1	5	1
α_G	0.0001			
α_D	0.0004			
DISTANCE FUNCTION	MEAN ABSOLUTE ERROR			

Table A.1: The hyper-parameters used for each objective formulation. NUM. DIS UPDATE is the number of discriminator updates for every generator update, such procedure was introduced in optimizing WGANGP [5]. α_G and α_D denotes 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.

A.1 Synthetic data experiment results



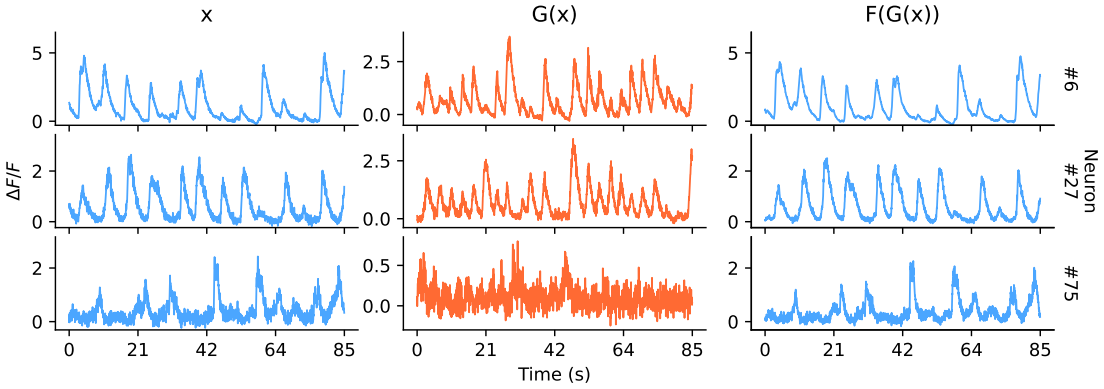
(a) Forward cycle: $X \to Y \to X$



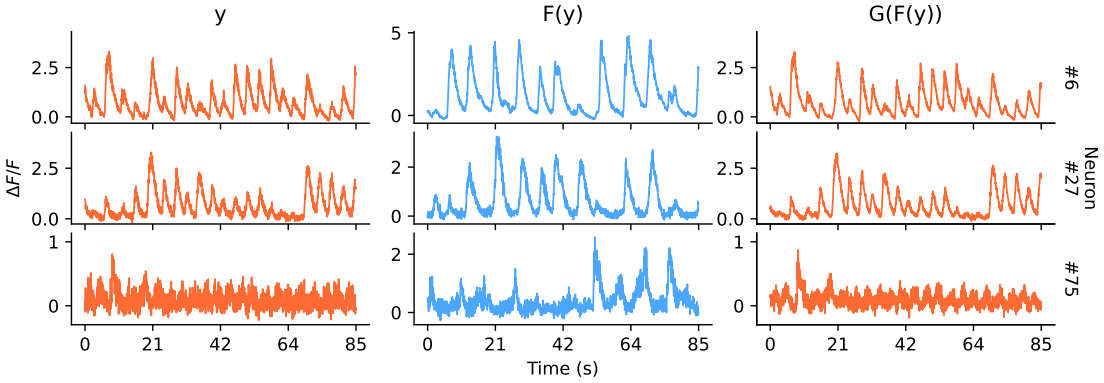
(b) Backward cycle: $Y \to X \to Y$

Figure A.1: (a) forward and (b) backward cycle of the entire 102 neurons from a randomly selected segment. Model was trained with AGResNet using the LSGAN objective on the synthetic dataset where $Y = \Phi(X)$, see Section 3.1.3 for detail.

A.2 Recorded data experiment results

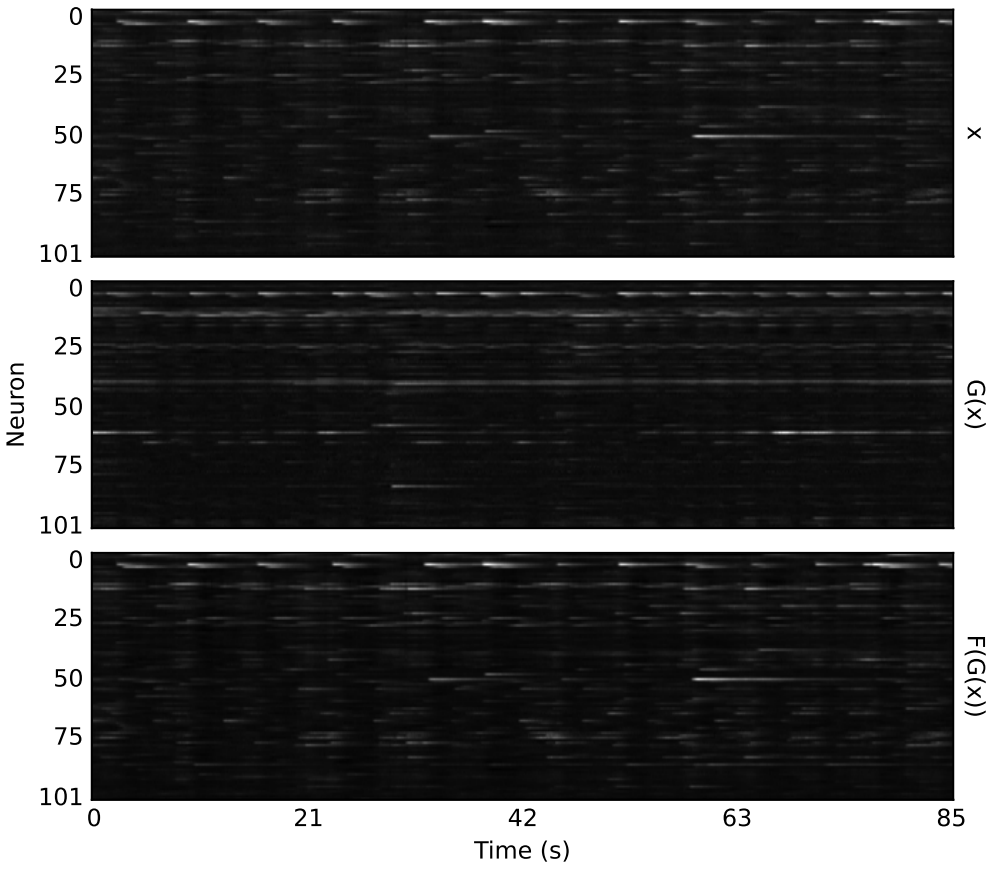


(a) Forward cycle: $X \to Y \to X$

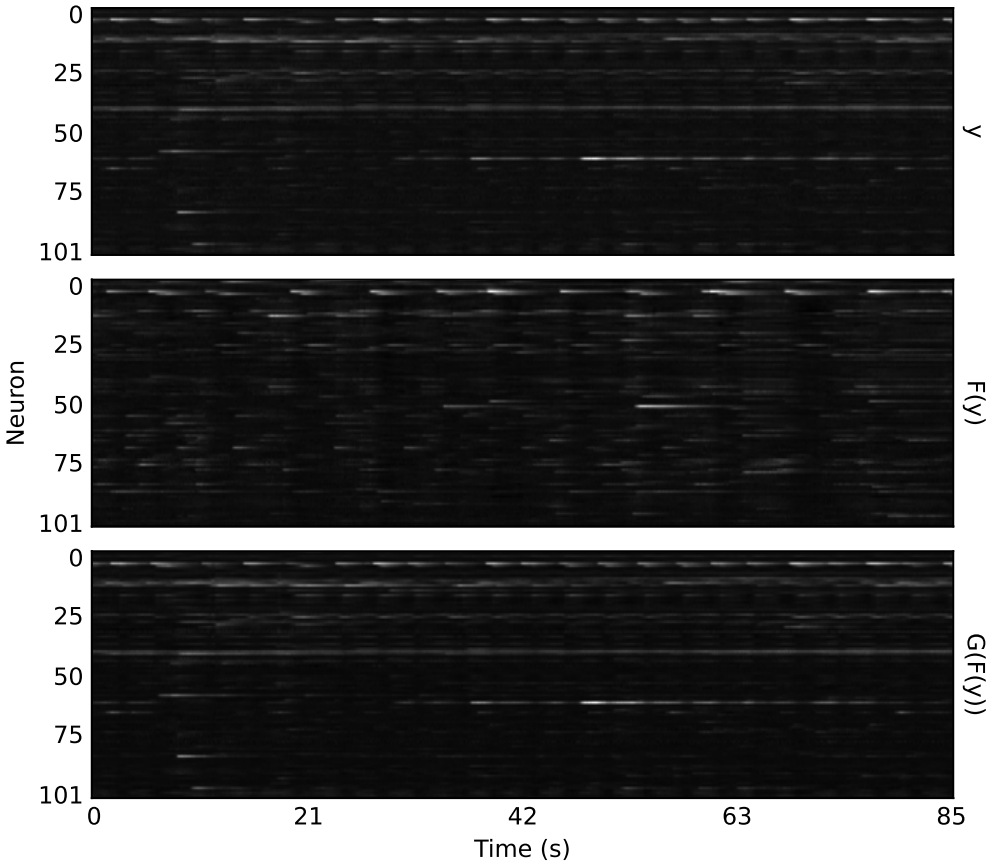


(b) Backward cycle: $Y \to X \to Y$

Figure A.2: (a) forward and (b) backward cycle of neuron 6, 27 and 75 from a randomly selected segment. Model was trained with AGResNet using the LSGAN objective on the recorded dataset. Note that, unlike the synthetic dataset, the traces presented here are not unpaired. Hence, we cannot directly compare x with $F(y)$ or y with $G(x)$.



(a) Forward cycle: $X \to Y \to X$



(b) Backward cycle: $Y \to X \to Y$

Figure A.3: (a) forward and (b) backward cycle of the entire 102 neurons from a randomly selected segment. Model was trained with AGResNet using the LSGAN objective on the recorded dataset.

A.2.1 Spike statistic analysis

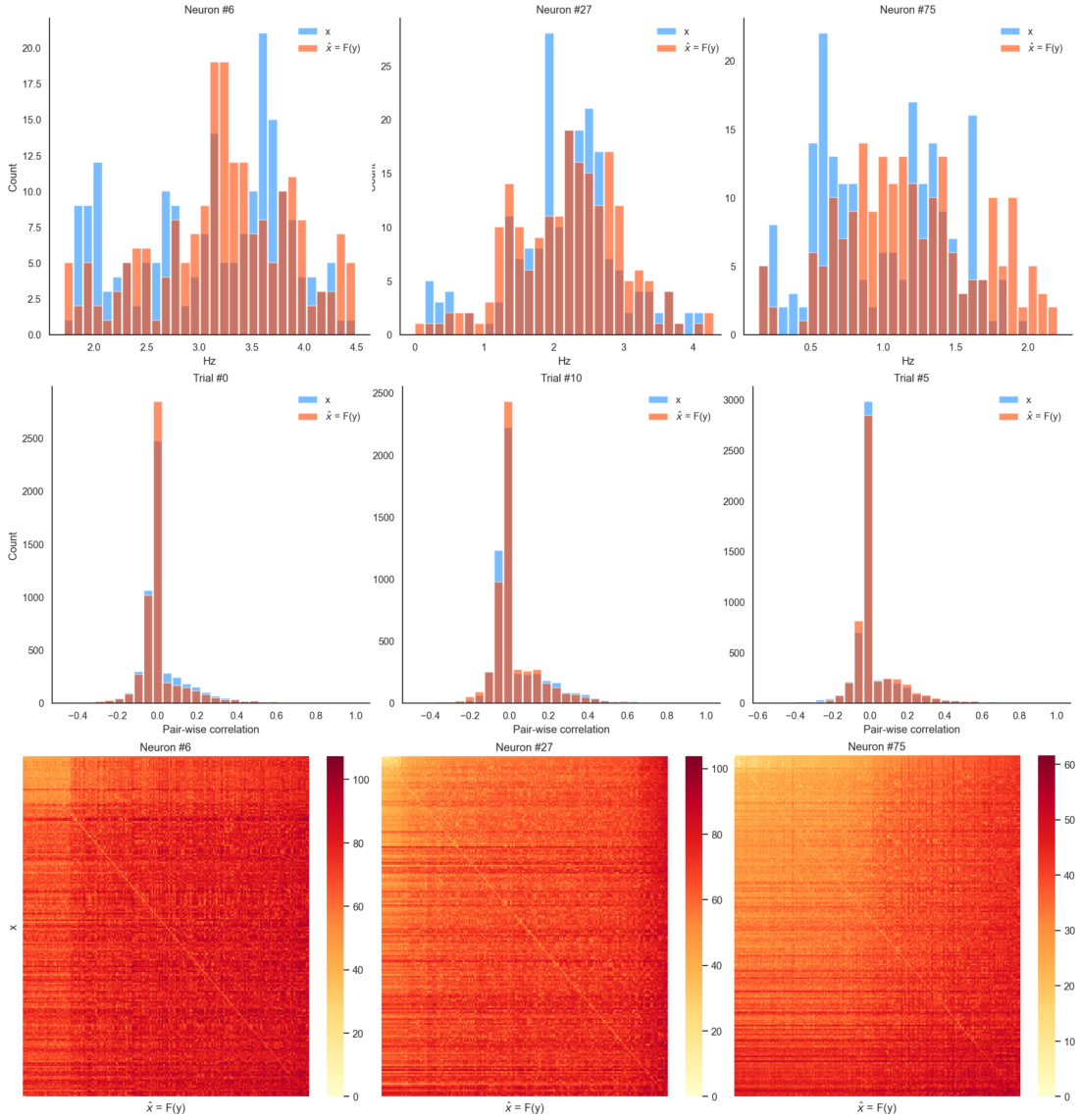


Figure A.4: (top-row) firing rate, (mid-row) pairwise correlation and (bottom-row) van Rossum distance statistic comparisons between X and $F(Y)$. We expect the distributions in firing rate and pairwise correlations in the generated data $\hat{X} = F(Y)$ resemble the the real distributions in X. Table 4.3 shows the mean KL divergence of each distributions. Whereas with van Rossum distance, we there would be at least one pair of spike trains in X and $\hat{X}$ to be similar, hence a diagonal line in the sorted heatmaps. However, due to the random shuffling done in the preprocessing, it is possible that the corresponding spike train $\hat{y} \sim F(Y)$ exists in the test set of X.

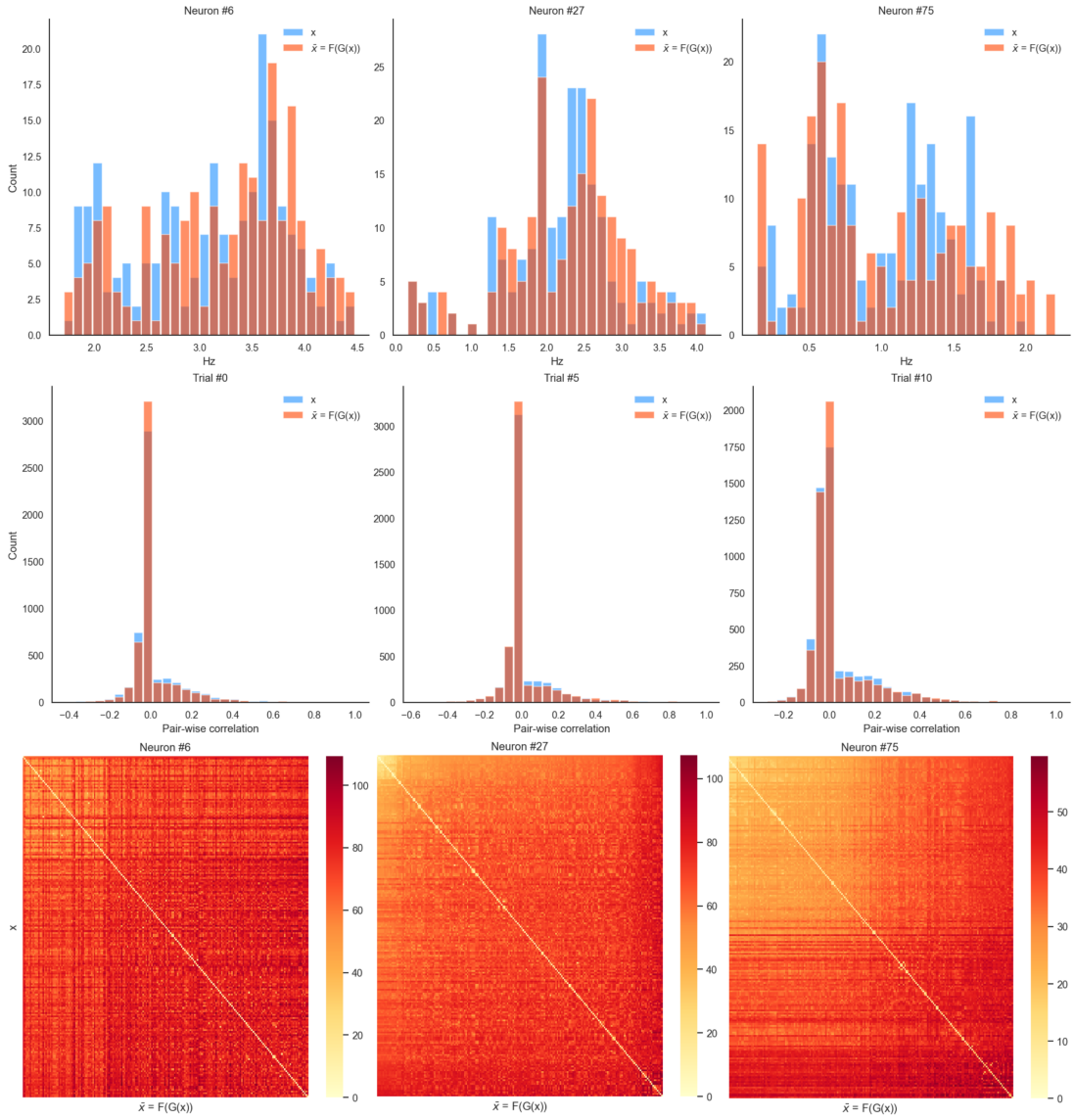


Figure A.5: (top-row) firing rate, (mid-row) pairwise correlation and (bottom-row) van Rossum distance statistic comparisons between X and $F(G(X))$. We expect the distributions in firing rate and pairwise correlations in the cycled data $\bar{x} = F(G(x))$ to be closely matched with the real distributions in X. Moreover, we would expect there exists a pair of spike trains in X and $\bar{X}$ to be very similar, resulting in a clear diagonal line in the sorted heatmaps. Table 4.3 shows the mean KL divergence of each distributions.

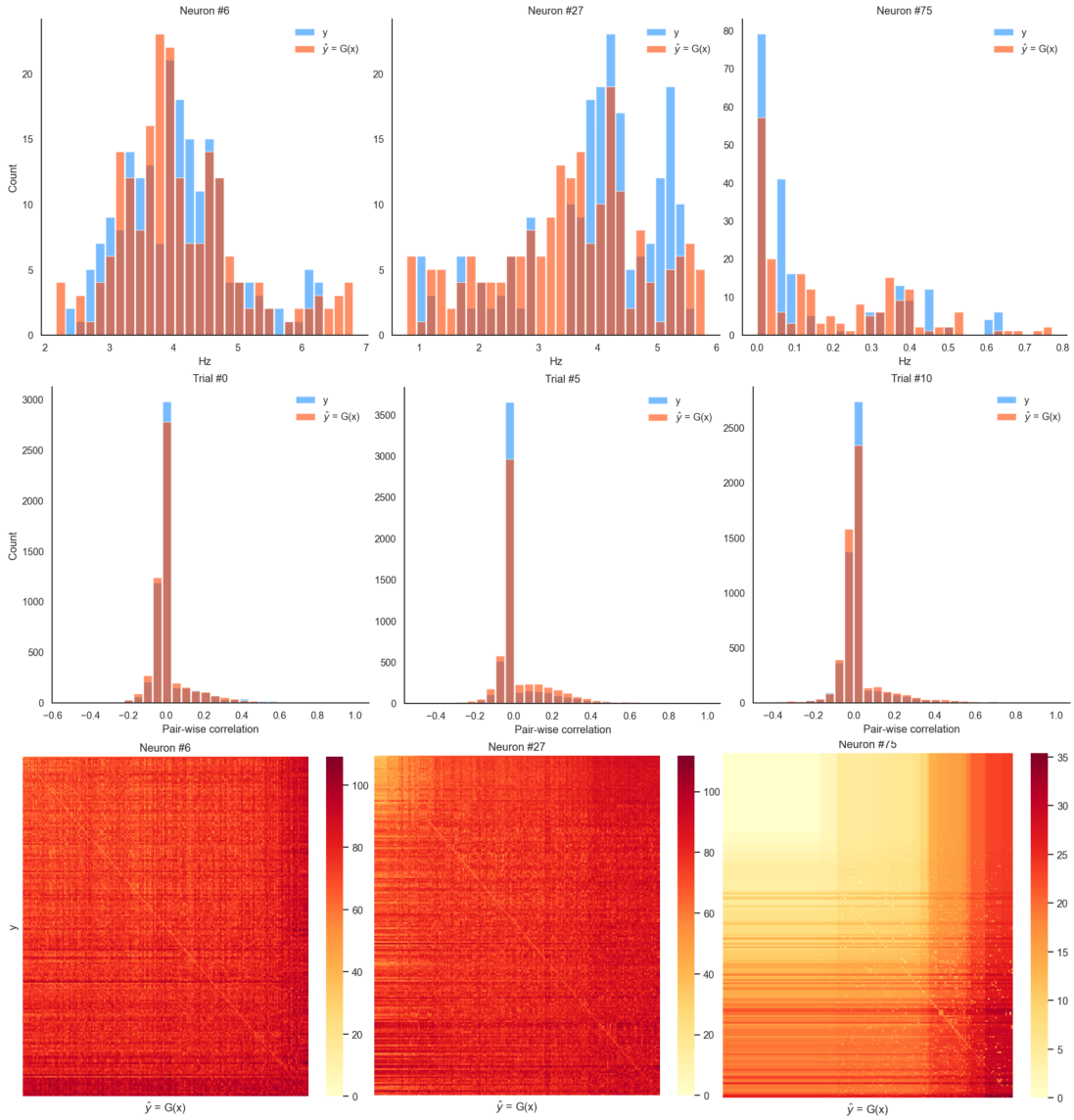


Figure A.6: (top-row) firing rate, (mid-row) pairwise correlation and (bottom-row) van Rossum distance statistic comparisons between Y and $G(X)$. We expect the distributions in firing rate and pairwise correlations in the generated data $\hat{Y} = G(X)$ resemble the the real distributions in Y. Table 4.3 shows the mean KL divergence of each distributions. Whereas with van Rossum distance, we there would be at least one pair of spike trains in Y and $\hat{Y}$ to be similar, hence a diagonal line in the sorted heatmaps. However, due to the random shuffling done in the preprocessing, it is possible that the corresponding spike train $\hat{y} \sim G(X)$ exists in the test set of Y.

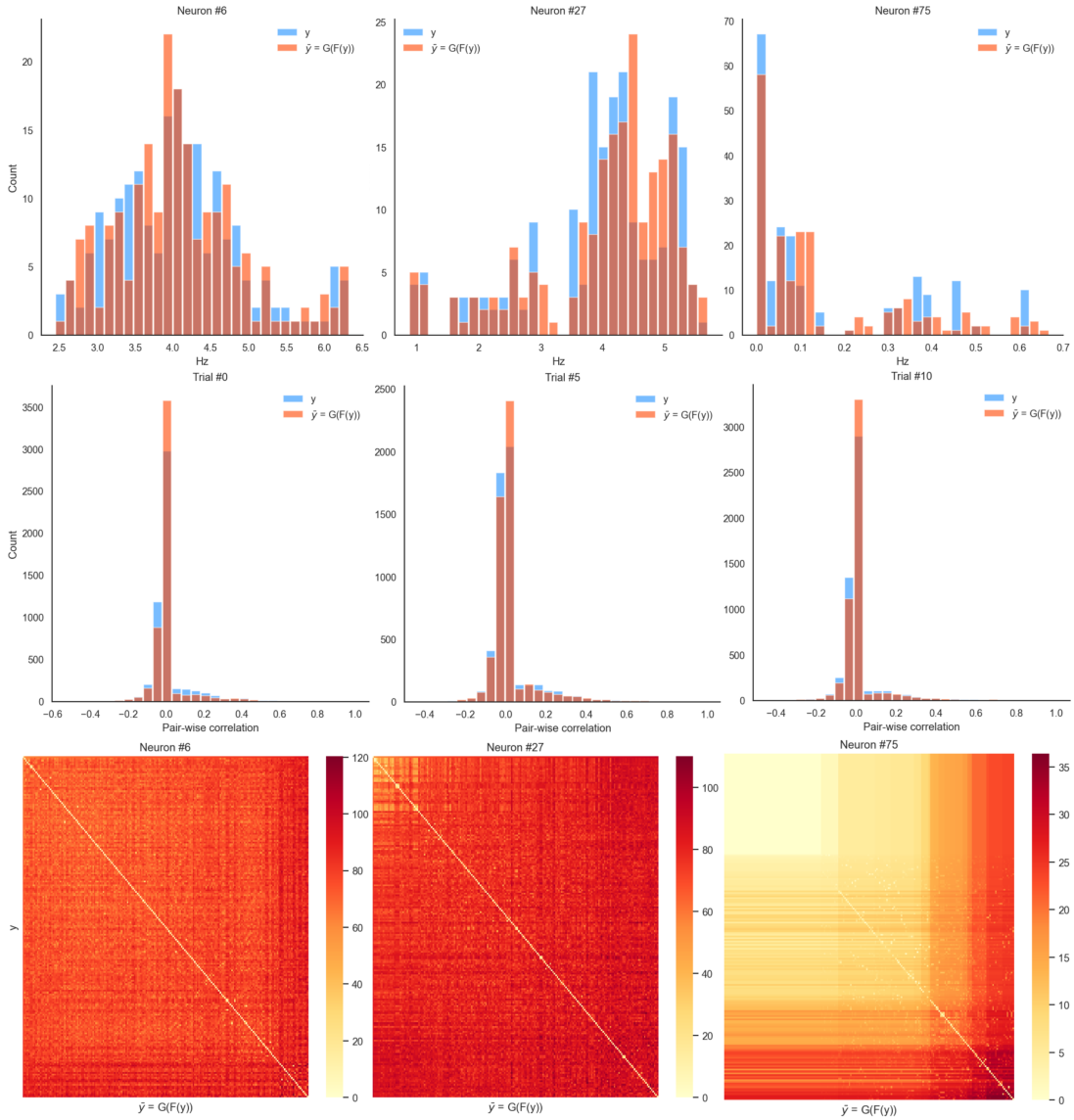
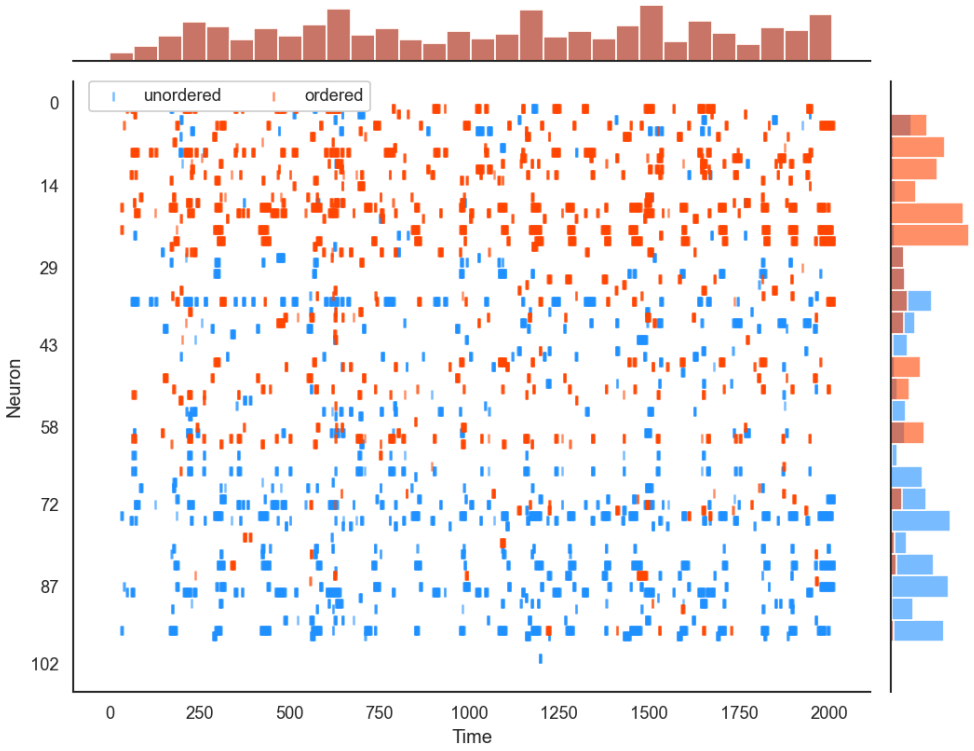


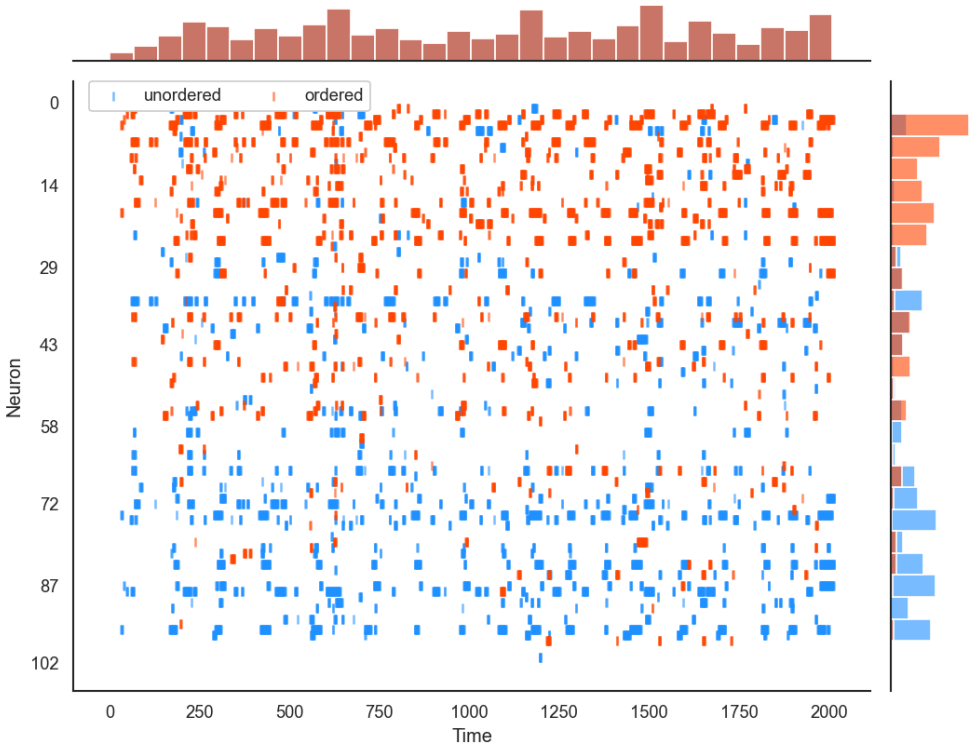
Figure A.7: (top-row) firing rate, (mid-row) pairwise correlation and (bottom-row) van Rossum distance statistic comparisons between Y and $G(F(Y))$. We expect the distributions in firing rate and pairwise correlations in the cycled data $\bar{y} = G(F(y))$ to be closely matched with the real distributions in X. Moreover, we would expect there exists a pair of spike trains in Y and $\bar{Y}$ to be very similar, resulting in a clear diagonal line in the sorted heatmaps. Table 4.3 shows the mean KL divergence of each distributions.

A.2.2 Neuron spatial order

(a) Epoch 1

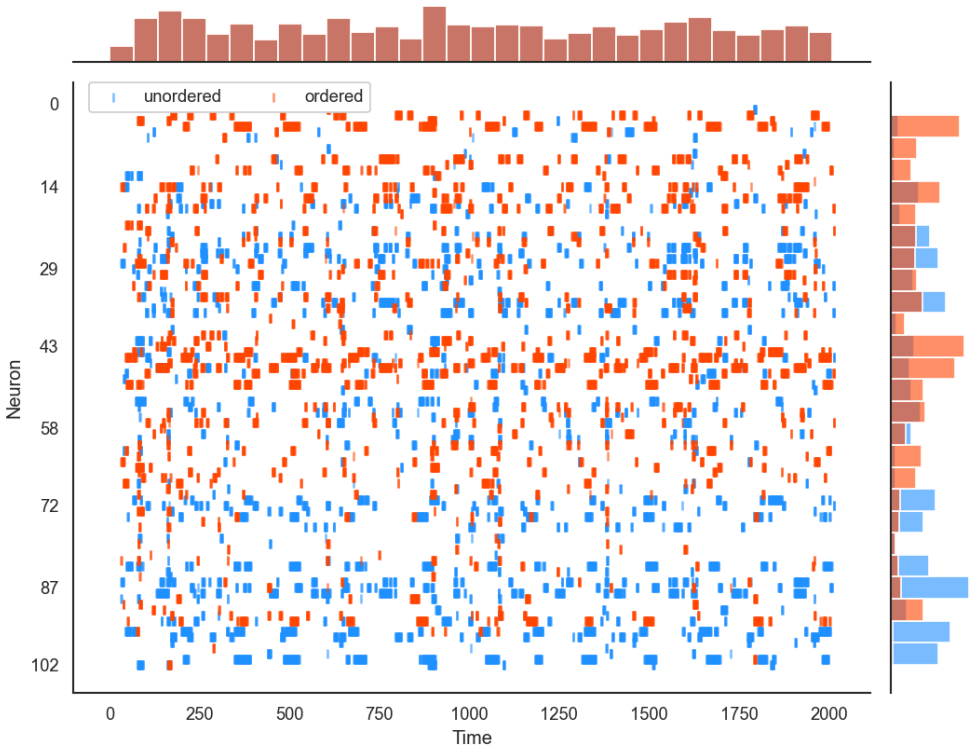


(b) Epoch 50

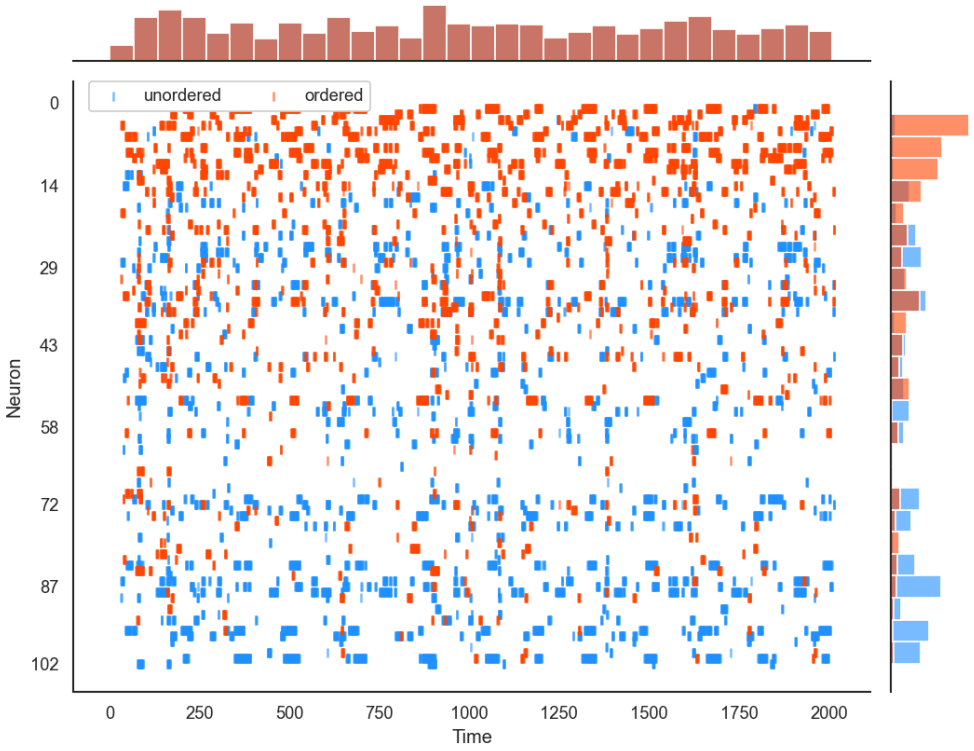


(c) Epoch 100

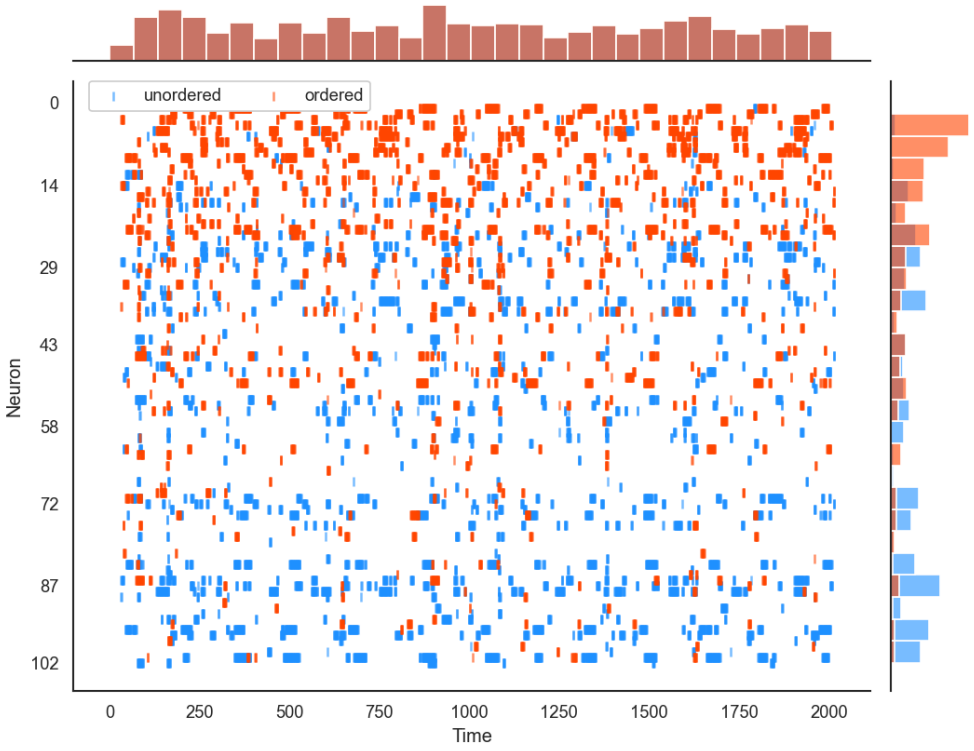
Figure A.8: Deep autoencoder $\mathtt{AE}_X$ neuron ordering progress on X at epoch (a) 1, (b) 50 and (c) 100. The inferred blue spike trains were following its original order whereas the orange spike trains were ordered by the reconstruction loss at the above-mentioned epochs. The histograms on the x and y axis show the firing distributions in the temporal and spatial dimension.



(a) Epoch 1



(b) Epoch 50



(c) Epoch 100

Figure A.9: Deep autoencoder $\mathtt{AE}_Y$ neuron ordering progress on Y at epoch (a) 1, (b) 50 and (c) 100. The inferred blue spike trains were following its original order whereas the orange spike trains were ordered by the reconstruction loss at the above-mentioned epochs. The histograms on the x and y axis show the firing distributions in the temporal and spatial dimension.

Bibliography

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[2] Almahairi, A., Rajeshwar, S., Sordoni, A., Bachman, P., and Courville, A. (2018). Augmented cyclegan: Learning many-to-many mappings from unpaired data. In *International Conference on Machine Learning*, pages 195–204. PMLR.

[3] Araujo, A., Norris, W., and Sim, J. (2019). Computing receptive fields of convolutional neural networks. *Distill*. https://distill.pub/2019/computing-receptive-fields.

[4] Arjovsky, M. and Bottou, L. (2017). Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*.

[5] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

[6] Barrett, D. G., Morcos, A. S., and Macke, J. H. (2019). Analyzing biological and artificial neural networks: challenges with opportunities for synergy? *Current opinion in neurobiology*, 55:55–64.

[7] Berridge, M. J., Lipp, P., and Bootman, M. D. (2000). The versatility and universality of calcium signalling. *Nature reviews Molecular cell biology*, 1(1):11–21.

[8] Berthelot, D., Schumm, T., and Metz, L. (2017). Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*.

[9] Brock, A., Donahue, J., and Simonyan, K. (2018). Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*.

[10] Buzsáki, G. (2004). Large-scale recording of neuronal ensembles. *Nature neuroscience*, 7(5):446–451.

[11] Cao, C., Liu, F., Tan, H., Song, D., Shu, W., Li, W., Zhou, Y., Bo, X., and Xie, Z. (2018). Deep learning and its applications in biomedicine. *Genomics, proteomics & bioinformatics*, 16(1):17–32.

[12] Caracciolo, L., Marosi, M., Mazzitelli, J., Latifi, S., Sano, Y., Galvan, L., Kawaguchi, R., Holley, S., Levine, M., Coppola, G., et al. (2018). Creb controls cortical circuit plasticity and functional recovery after stroke. *Nature communications*, 9(1):1–16.

[13] Carlson, D. and Carin, L. (2019). Continuing progress of spike sorting in the era of big data. *Current opinion in neurobiology*, 55:90–96.

[14] Casalicchio, G., Molnar, C., and Bischl, B. (2018). Visualizing the feature importance for black box models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 655–670. Springer.

[15] Chaudhari, S., Mithal, V., Polatkan, G., and Ramanath, R. (2019). An attentive survey of attention models. *arXiv preprint arXiv:1904.02874*.

[16] Che, T., Li, Y., Jacob, A. P., Bengio, Y., and Li, W. (2016). Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*.

[17] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

[18] Chen, X., Xu, C., Yang, X., and Tao, D. (2018). Attention-gan for object transfiguration in wild images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 164–180.

[19] Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE.

[20] Churchland, M. M., Cunningham, J. P., Kaufman, M. T., Foster, J. D., Nuyujukian,

P., Ryu, S. I., and Shenoy, K. V. (2012). Neural population dynamics during reaching. *Nature*, 487(7405):51–56.

[21] Cohen, J. P., Luck, M., and Honari, S. (2018). Distribution matching losses can hallucinate features in medical image translation. In *International conference on medical image computing and computer-assisted intervention*, pages 529–536. Springer.

[22] Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

[23] Dayan, P. and Abbott, L. F. (2001). Theoretical neuroscience: computational and mathematical modeling of neural systems.

[24] Denker, M., Yegenoglu, A., and Grün, S. (2018). Collaborative HPC-enabled workflows on the HBP Collaboratory using the Elephant framework. In *Neuroinformatics 2018*, page P19.

[25] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[26] Donahue, C., McAuley, J., and Puckette, M. (2018). Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208*.

[27] Dong, S., Wang, P., and Abbas, K. (2021). A survey on deep learning and its applications. *Computer Science Review*, 40:100379.

[28] Driscoll, L. N., Pettit, N. L., Minderer, M., Chettih, S. N., and Harvey, C. D. (2017). Dynamic reorganization of neuronal activity patterns in parietal cortex. *Cell*, 170(5):986–999.

[29] Durstewitz, D. (2017). A state space approach for piecewise-linear recurrent neural networks for identifying computational dynamics from neural measurements. *PLoS computational biology*, 13(6):e1005542.

[30] Fan, F.-L., Xiong, J., Li, M., and Wang, G. (2021). On interpretability of artificial neural networks: A survey. *IEEE Transactions on Radiation and Plasma Medical Sciences*.

[31] Gallego, J. A., Perich, M. G., Chowdhury, R. H., Solla, S. A., and Miller, L. E.

(2020). Long-term stability of cortical population dynamics underlying consistent behavior. *Nature neuroscience*, 23(2):260–270.

[32] Ganmor, E., Segev, R., and Schneidman, E. (2011). The architecture of functional interaction networks in the retina. *Journal of Neuroscience*, 31(8):3044–3054.

[33] Ghorbani, A., Abid, A., and Zou, J. (2019). Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3681–3688.

[34] Gomez, A. N., Huang, S., Zhang, I., Li, B. M., Osama, M., and Kaiser, L. (2018). Unsupervised cipher cracking using discrete gans. *arXiv preprint arXiv:1801.04883*.

[35] Gondara, L. (2016). Medical image denoising using convolutional denoising autoencoders. In *2016 IEEE 16th international conference on data mining workshops (ICDMW)*, pages 241–246. IEEE.

[36] Goodfellow, I. (2016). Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*.

[37] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[38] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

[39] Google (2021). Using bfloat16 with tensorflow models | cloud tpu | google cloud.

[40] Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., and Pedreschi, D. (2018). A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42.

[41] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777.

[42] Haldekar, M., Ganesan, A., and Oates, T. (2017). Identifying spatial relations in images using convolutional neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 3593–3600. IEEE.

[43] He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[44] He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer.

[45] Henschke, J. U., Dylda, E., Katsanevaki, D., Dupuy, N., Currie, S. P., Amvrosiadis, T., Pakan, J. M., and Rochefort, N. L. (2020). Reward association enhances stimulus-specific representations in primary visual cortex. *Current Biology*.

[46] Herculano-Houzel, S. (2012). The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost. *Proceedings of the National Academy of Sciences*, 109(Supplement 1):10661–10668.

[47] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

[48] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

[49] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

[50] Jabbar, A., Li, X., and Omar, B. (2020). A survey on generative adversarial networks: Variants, applications, and training. *arXiv preprint arXiv:2006.05132*.

[51] Jercog, P., Rogerson, T., and Schnitzer, M. J. (2016). Large-scale fluorescence calcium-imaging methods for studies of long-term memory in behaving mammals. *Cold Spring Harbor perspectives in biology*, 8(5):a021824.

[52] Jetley, S., Lord, N. A., Lee, N., and Torr, P. H. (2018). Learn to pay attention. *arXiv preprint arXiv:1804.02391*.

[53] Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*.

[54] Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A.,

Liu, C., and Krishnan, D. (2020). Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*.

[55] Kim, T., Cha, M., Kim, H., Lee, J. K., and Kim, J. (2017). Learning to discover cross-domain relations with generative adversarial networks. In *International Conference on Machine Learning*, pages 1857–1865. PMLR.

[56] Kindermans, P.-J., Hooker, S., Adebayo, J., Alber, M., Schütt, K. T., Dähne, S., Erhan, D., and Kim, B. (2019). The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer.

[57] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[58] Kodali, N., Abernethy, J., Hays, J., and Kira, Z. (2017). On convergence and stability of gans. *arXiv preprint arXiv:1705.07215*.

[59] Kurach, K., Lučić, M., Zhai, X., Michalski, M., and Gelly, S. (2019). A large-scale study on regularization and normalization in gans. In *International Conference on Machine Learning*, pages 3581–3590. PMLR.

[60] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

[61] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690.

[62] Li, B. M., Amvrosiadis, T., Rochefort, N., and Onken, A. (2020). Calciumgan: A generative adversarial network model for synthesising realistic calcium imaging data of neuronal populations. *arXiv preprint arXiv:2009.02707*.

[63] Li, Z., Liu, F., Yang, W., Peng, S., and Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*.

[64] Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., and Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26.

[65] Long, J. L., Zhang, N., and Darrell, T. (2014). Do convnets learn correspondence? *Advances in neural information processing systems*, 27:1601–1609.

[66] Lucic, M., Kurach, K., Michalski, M., Gelly, S., and Bousquet, O. (2017). Are gans created equal? a large-scale study. *arXiv preprint arXiv:1711.10337*.

[67] Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer.

[68] Macke, J. H., Berens, P., Ecker, A. S., Tolias, A. S., and Bethge, M. (2009). Generating spike trains with specified correlation coefficients. *Neural computation*, 21(2):397–423.

[69] Makino, T., Jastrzebski, S., Oleszkiewicz, W., Chacko, C., Ehrenpreis, R., Samreen, N., Chhor, C., Kim, E., Lee, J., Pysarenko, K., Reig, B., Toth, H., Awal, D., Du, L., Kim, A., Park, J., Sodickson, D. K., Heacock, L., Moy, L., Cho, K., and Geras, K. J. (2020). Differences between human and machine perception in medical diagnosis.

[70] Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Paul Smolley, S. (2017). Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802.

[71] Maziarka, Ł., Pocha, A., Kaczmarczyk, J., Rataj, K., Danel, T., and Warchoł, M. (2020). Mol-cyclegan: a generative model for molecular optimization. *Journal of Cheminformatics*, 12(1):1–18.

[72] Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., et al. (2017). Mixed precision training. *arXiv preprint arXiv:1710.03740*.

[73] Miotto, R., Wang, F., Wang, S., Jiang, X., and Dudley, J. T. (2018). Deep learning for healthcare: review, opportunities and challenges. *Briefings in bioinformatics*, 19(6):1236–1246.

[74] Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

[75] Molano-Mazon, M., Onken, A., Piasini*, E., and Panzeri*, S. (2018). Synthesizing realistic neural population activity patterns using generative adversarial networks. In *International Conference on Learning Representations*.

[76] Müller, R., Kornblith, S., and Hinton, G. (2019). When does label smoothing help? *arXiv preprint arXiv:1906.02629*.

[77] Nain, A. K. (2020). Keras documentation: Cyclegan.

[78] Nemati, S., Linderman, S. W., Chen, Z., et al. (2014). A probabilistic modeling approach for uncovering neural population rotational dynamics. *Cosyne*, (180106).

[79] Nvidia (2021). Training with mixed precision - nvidia deep learning performance documentation.

[80] Oktay, O., Schlemper, J., Folgoc, L. L., Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N. Y., Kainz, B., et al. (2018). Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*.

[81] Pakan, J. M., Currie, S. P., Fischer, L., and Rochefort, N. L. (2018). 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.

[82] Pan, Z., Yu, W., Yi, X., Khan, A., Yuan, F., and Zheng, Y. (2019). Recent progress on generative adversarial networks (gans): A survey. *IEEE Access*, 7:36322–36333.

[83] Pandarinath, C., O'Shea, D. J., Collins, J., Jozefowicz, R., Stavisky, S. D., Kao, J. C., Trautmann, E. M., Kaufman, M. T., Ryu, S. I., Hochberg, L. R., et al. (2018). Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature methods*, page 1.

[84] Park, T., Efros, A. A., Zhang, R., and Zhu, J.-Y. (2020). Contrastive learning for unpaired image-to-image translation. In *European Conference on Computer Vision*, pages 319–345. Springer.

[85] Pasini, M. (2019). 10 lessons i learned training gans for a year.

[86] Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. (2016). Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544.

[87] Pedreira, C., Martinez, J., Ison, M. J., and Quiroga, R. Q. (2012). How many neurons can we see with current spike sorting algorithms? *Journal of neuroscience methods*, 211(1):58–65.

[88] Piccialli, F., Di Somma, V., Giampaolo, F., Cuomo, S., and Fortino, G. (2021). A

survey on deep learning in medicine: Why, how and when? *Information Fusion*, 66:111–137.

[89] Prince, L. Y., Bakhtiari, S., Gillon, C. J., and Richards, B. A. (2020). Calfads: latent factor analysis of dynamical systems in calcium imaging data.

[90] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

[91] Rey, H. G., Pedreira, C., and Quiroga, R. Q. (2015). Past, present and future of spike sorting techniques. *Brain research bulletin*, 119:106–117.

[92] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

[93] Rossum, M. v. (2001). A novel spike distance. *Neural computation*, 13(4):751–763.

[94] Rupprecht, P., Carta, S., Hoffmann, A., Echizen, M., Blot, A., Kwan, A. C., Dan, Y., Hofer, S. B., Kitamura, K., Helmchen, F., et al. (2021). A database and deep learning toolbox for noise-optimized, generalized spike inference from calcium imaging. *Nature Neuroscience*, pages 1–14.

[95] Russell, J. T. (2011). Imaging calcium signals in vivo: a powerful tool in physiology and pharmacology. *British journal of pharmacology*, 163(8):1605–1625.

[96] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. *Advances in neural information processing systems*, 29:2234–2242.

[97] Schneidman, E., Berry, M. J., Segev, R., and Bialek, W. (2006). Weak pairwise correlations imply strongly correlated network states in a neural population. *Nature*, 440(7087):1007–1012.

[98] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626.

[99] She, Q. and Wu, A. (2020). Neural dynamics discovery via gaussian process

recurrent neural networks. In *Uncertainty in Artificial Intelligence*, pages 454–464. PMLR.

[100] Shlens, J., Field, G. D., Gauthier, J. L., Grivich, M. I., Petrusca, D., Sher, A., Litke, A. M., and Chichilnisky, E. (2006). The structure of multi-neuron firing patterns in primate retina. *Journal of Neuroscience*, 26(32):8254–8266.

[101] Simonyan, K., Vedaldi, A., and Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

[102] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[103] Smilkov, D., Thorat, N., Kim, B., Viégas, F., and Wattenberg, M. (2017). Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*.

[104] Steinmetz, N. A., Aydin, C., Lebedeva, A., Okun, M., Pachitariu, M., Bauza, M., Beau, M., Bhagat, J., Böhm, C., Broux, M., et al. (2021). Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. *Science*, 372(6539).

[105] Stosiek, C., Garaschuk, O., Holthoff, K., and Konnerth, A. (2003). In vivo two-photon calcium imaging of neuronal networks. *Proceedings of the National Academy of Sciences*, 100(12):7319–7324.

[106] Sweetnam, D., Holmes, A., Tennant, K. A., Zamani, A., Walle, M., Jones, P., Wong, C., and Brown, C. E. (2012). Diabetes impairs cortical plasticity and functional recovery following ischemic stroke. *Journal of Neuroscience*, 32(15):5132–5143.

[107] Tang, A., Jackson, D., Hobbs, J., Chen, W., Smith, J. L., Patel, H., Prieto, A., Petrusca, D., Grivich, M. I., Sher, A., et al. (2008). A maximum entropy model applied to spatial and temporal correlations from cortical networks in vitro. *Journal of Neuroscience*, 28(2):505–518.

[108] Theis, L., Berens, P., Froudarakis, E., Reimer, J., Rosón, M. R., Baden, T., Euler, T., Tolias, A. S., and Bethge, M. (2016). Benchmarking spike rate inference in population calcium imaging. *Neuron*, 90(3):471–482.

[109] Tmenova, O., Martin, R., and Duong, L. (2019). Cyclegan for style transfer in

x-ray angiography. *International journal of computer assisted radiology and surgery*, 14(10):1785–1794.

[110] Tompson, J., Goroshin, R., Jain, A., LeCun, Y., and Bregler, C. (2015). Efficient object localization using convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 648–656.

[111] Tschannen, M., Bachem, O., and Lucic, M. (2018). Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*.

[112] Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*.

[113] Vanwalleghem, G., Constantin, L., and Scott, E. K. (2020). Calcium imaging and the curse of negativity. *Frontiers in neural circuits*, 14.

[114] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

[115] Wang, W., Zheng, V. W., Yu, H., and Miao, C. (2019). A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–37.

[116] Wang, Y., Yao, H., and Zhao, S. (2016). Auto-encoder based dimensionality reduction. *Neurocomputing*, 184:232–242.

[117] Wang, Y., Yao, Q., Kwok, J. T., and Ni, L. M. (2020). Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3):1–34.

[118] Wang, Z., She, Q., and Ward, T. E. (2021). Generative adversarial networks in computer vision: A survey and taxonomy. *ACM Computing Surveys (CSUR)*, 54(2):1–38.

[119] Wei, Z., Lin, B.-J., Chen, T.-W., Daie, K., Svoboda, K., and Druckmann, S. (2020). A comparison of neuronal population dynamics measured with calcium imaging and electrophysiology. *PLoS computational biology*, 16(9):e1008198.

[120] Williams, A. H., Kim, T. H., Wang, F., Vyas, S., Ryu, S. I., Shenoy, K. V., Schnitzer, M., Kolda, T. G., and Ganguli, S. (2018a). Unsupervised discovery

of demixed, low-dimensional neural dynamics across multiple timescales through tensor component analysis. *Neuron*, 98(6):1099–1115.e8.

[121] Williams, A. H., Kim, T. H., Wang, F., Vyas, S., Ryu, S. I., Shenoy, K. V., Schnitzer, M., Kolda, T. G., and Ganguli, S. (2018b). Unsupervised discovery of demixed, low-dimensional neural dynamics across multiple timescales through tensor component analysis. *Neuron*, 98(6):1099–1115.

[122] Wu, J., Zhang, C., Xue, T., Freeman, W. T., and Tenenbaum, J. B. (2016). Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 82–90.

[123] Yang, H., Sun, J., Carass, A., Zhao, C., Lee, J., Xu, Z., and Prince, J. (2018). Unpaired brain mr-to-ct synthesis using a structure-constrained cyclegan. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 174–182. Springer.

[124] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

[125] Yi, Z., Zhang, H., Tan, P., and Gong, M. (2017). Dualgan: Unsupervised dual learning for image-to-image translation. In *Proceedings of the IEEE international conference on computer vision*, pages 2849–2857.

[126] Yu, N., Liu, G., Dundar, A., Tao, A., Catanzaro, B., Davis, L., and Fritz, M. (2021). Dual contrastive loss and attention for gans. *arXiv preprint arXiv:2103.16748*.

[127] Zemouri, R., Zerhouni, N., and Racoceanu, D. (2019). Deep learning in the biomedical applications: Recent and future status. *Applied Sciences*, 9(8):1526.

[128] Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., and Metaxas, D. N. (2017). Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915.

[129] Zhong, J., Haoran, W., Yunlong, Y., and Yanwei, P. (2019). A decadal survey of zero-shot image classification. *SCIENTIA SINICA Informationis*, 49(10):1299–1320.

[130] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929.

[131] Zhu, J., Yang, G., and Lio, P. (2019). How can we make gan perform better in single medical image super-resolution? a lesion focused multi-scale approach. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 1669–1673. IEEE.

[132] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232.