Synthesising Realistic Calcium Imaging Data of Neuronal Populations using Deep Generative Models

Bryan Man Yeung Li

Master of Science by Research
Institute for Adaptive and Neural Computation
School of Informatics
University of Edinburgh

2020

Abstract

Recordings of neuronal activities from behaving animals are essential for the study of information processing in the brain, and calcium imaging has become a powerful and popular technique to monitor the activity of large populations of neurons in *vivo*. However, for ethical considerations and despite recent technical developments, recordings are still constrained to a limited number of trials and animals. This limits the amount of data available from individual experiments and hinders the development of analysis techniques and models for a more realistic size of neuronal populations. The ability to artificially synthesize realistic neuronal calcium signals could greatly alleviate this problem by scaling up the number of trials. In this work, we (a) explore the use of Generative Adversarial Networks (GAN) to synthesize realistic fluorescent calcium indicator signals, and (b) develop a pipeline to pre-process, fit and evaluate both synthetic calcium signals and their inferred spike trains. We test the models on artificial data with known ground-truth, as well as real calcium signals recorded from the primary visual cortex of behaving mice. Together, our results demonstrate that the GAN framework is capable of synthesizing realistic fluorescent calcium indicator signals similar to those imaged in the somata of neuronal populations of behaving animals. Thereby providing the means to augment existing datasets of neuronal activity for enhanced data exploration and modelling.

Acknowledgements

My most sincere gratitude to my supervisor, Dr. Arno Onken, for his tireless guidance throughout my postgraduate studies. I would like to thank Dr. Nathalie Rochefort and Theoklitos Amvrosiadis for providing the calcium imaging data as well as providing meaningful insights from the neuroscience perspective. I would also like to thank Dr. Nina Kudryashova and Lazaros Mitskopoulos for their enlightenment and ideas.

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 Man Yeung Li*)

Table of Contents

Chapter 1

Introduction

Recordings of neuronal activities from behaving animals are essential for the study of information processing in the brain. With the advancement of neural recording techniques, such as electrophysiological recordings and calcium imaging, it has become increasingly easier to obtain high-quality neuronal activity data in *vivo*. However, due to ethical considerations, the acquired datasets are often limited by the number of trials or the duration of each trial on a live animal. This poses a problem for assessing analysis techniques that take higher-order correlations into account [6, 51, 53, 54]. Even for linear decoders, the number of trials can be more important for determining coding accuracy than the number of neurons [56].

Generative models of neuronal activity hold the promise of alleviating the above problem by enabling the synthesis of an unlimited number of realistic samples for assessing advanced analysis methods. Popular modelling approaches including the maximum entropy framework [17, 37, 44, 52] and the latent variable model [30, 31] have shown ample success in modelling spiking activities, though oftentimes these model require a strong assumption of the data and cannot generalize across different cortical areas. To this end, deep generative models could be a good candidate for modelling neuronal activities due to their ability to self-identify and self-learn features from the dataset [27].

Recently, the use of deep generative models, such as Variational Autoencoder (VAE) and Generative Adversarial Networks (GAN), on neuronal population spike train data have become increasingly popular. Latent Factor Analysis via Dynamical Systems (LFADS, [44]) uses the VAE framework to learn the population dynamics in latent

representation and extract "denoised" single-trial firing rates from neural spiking data. Spike-GAN [36] demonstrated that GAN can model binary spike trains of a small number of neurons and can reproduce the low-level statistics of neurons recorded from salamander retina. Ramesh et al. [47] extended the work by training a conditional GAN [35] to generate multivariate binary spike trains.

All of the aforementioned generative models operate on population spike trains. Spike trains are discrete in nature meaning that they cannot be subject to any continuous increment or decrement. Hence it remains a difficult task to optimize deep generative models for discrete data with back-propagation [7], which is key for training deep neural networks. Calcium imaging recordings, on the other hand, access changes in intracellular calcium concentration as a proxy for neuronal spiking activity, hence its data is continuous. The continuous nature of calcium fluorescence signals makes optimization via back-propagation a much more straightforward task as compared to spike-train data. Thus, calcium imaging datasets are more attractive candidates for training generative models.

In this work, we explore the feasibility of using the GAN framework to synthesize calcium imaging data. We detailed CalciumGAN, as a method to scale-up or augment the amount of neuronal population activity data. Work in this direction could open endless amount of possibility such as style transfer [26] and representation learning [9]. We validate the method on artificial data with known ground-truth and we synthesize data mimicking real two-photon calcium (Ca^{2+}) imaging data as recorded from the primary visual cortex of a behaving mouse [23, 42].

1.1 Structure

Chapter 2 introduces the necessary background information for this work. Section 2.1 provides an overview of various neuronal activities recording techniques. Section 2.2 discusses popular statistical models for synthesising spiking activities, including the maximum entropy framework, latent variable models and deep generative models. Section 2.3 briefly describe the various formulation of GAN, their common problems as well as possible solutions.

Next, the methods we proposed in this works are presented in Chapter 3. Section 3.1 describes in detail about the exact model architecture we used and Section 3.2 layout the complete pipeline of the model, from pre-processing to spike analysis. Section 3.2.1

and Section 3.2.2 discuss the possible variants of our CalciumGAN model that aim to address various aspect of the data.

In the result Chapter 4, we first specified the programming technical detail and model training parameters of CalciumGAN in Section 4.1. In order to produce an accurate measure of the statistics of the generated data, we first fitted the model with artificial data with known ground-truth statistics in Section 4.2. Then in Section 4.3, we trained CalciumGAN with real recorded data from a behaving mouse and compared their mean firing rate, pairwise correlation and van-Rossum distance. We also analysed the different variation of CalciumGAN in this section, including modeling the spatio-temporal dimensions of the data with 2-dimensional convolution kernels as well as training the model in frequency-domain.

Finally, Chapter 5 provides a summary of this work, including its contribution to the deep learning and computational neuroscience community. We also discuss the limitations of our approach to the problem and proposed various future research directions that consolidate GAN and neural coding.

Chapter 2

Background

2.1 Neuronal activities recordings

One of the main methods for neurons to propagate signals in the brain and nerve system is by generating electrochemical pulses, which are known as action potentials or spikes [10]. Information is represented and transmitted via sequences of spikes in a specific order and temporal patterns. In order to understand the neural information processing process, it is essential to be able to record spiking activities accurately, and in a large scale setting. Modern recording techniques enable us to simultaneously record neuronal activities from dozens of neurons to tens of thousands of neurons, each method having its own advantages and weaknesses. In this section, we briefly discuss the two popular in *vivo* recording techniques, electrophysiological recording and calcium imaging.

2.1.1 Electrophysiological recording

Electrophysiological recording, which measures the rate of change in voltage by micro-electrodes inserted in the cell membrane of a neuron, is considered the most accurate method to measure spike activities [10]. Though this method is not without flaws. For instance, a single microelectrode can only detect activity from a few neurons in close proximity, and extensive pre-processing is required to infer single-unit activity from a multi-unit signal. Disentangling circuit computations in neuronal populations of a large scale remains a difficult task, hence resulting in recordings with low spatial resolution but high temporal resolution [48].

2.1.2 Calcium imaging recording

Calcium imaging recording is another imaging technique that has become a powerful and popular method to monitor large neuron populations. When the membrane potential reaches a certain threshold voltage, numerous ions are released from the voltage-gated ion channels to depolarize the cell, including calcium (Ca^{2+}) ions, and causes an action potential [4, 50]. Calcium imaging takes advantage of calcium influx during the depolarization process, and monitor the green fluorescent protein indicator, as a proxy for neuronal spiking activity [55]. This recording method can monitor a large number of neurons simultaneously, and unlike electrophysiological recording, it can also track the activity of the same neuron populations over time [45]. However, since calcium imaging is an indirect read-out of the cellular activity, and the transformation from calcium fluorescence signal to spiking activity is still an active area of research, inferring spikes from calcium signals accurately remains a challenging task [60].

2.2 Synthesising neuronal activities

The ability to synthesize realistic spike activities can greatly improve our understanding of how neurons encode and decode information in our brain. Many studies have been done on mimicking the neuronal functional architecture using different statistical models, mostly synthesizing neuronal activities in the form of spike trains [17, 30, 31, 37, 44, 52]. The following sections outline some of the popular statistical methods for modelling neuronal activities, as well as the existing deep generative models that use the GAN framework to synthesis spike trains.

2.2.1 Maximum entropy method

The maximum entropy framework has seen considerable success in capturing the low-level statistics of spiking activities [17, 44, 52]. One key advantage of the maximum entropy method is that it allows us to progressively add layers of restriction to the otherwise unstructured model. Schneidman et al. [52] compares the Ising model to the spiking patterns of neural networks to capture the low-level statistics of spiking patterns. By computing the firing rate of N neurons and the correlation between each neuron pairs, the Ising model formulates the maximum entropy probability distribution P_{MaxEnt}

and energy function $\mathcal{H}$ as:

$$P_{MaxEnt}(s) = \frac{1}{Z(s)} \exp(-\mathcal{H}(s)) \tag{2.1}$$

$$\mathcal{H}(s) = \sum_{i=1}^{N} h_i s_i + \frac{1}{2} \sum_{i,j=1}^{N} J_{ij} s_i s_j \tag{2.2}$$

where $s_i = 1$ denotes a spike, $Z(s) = \sum_s \exp(-\mathcal{H}(s))$ is a normalization factor, and h_i and J_{ij} are the Lagrange multipliers, needed to be tuned to match the experimental data. This pairwise model has shown excellent result in capturing the low-order statistics of up to $N = 40$ retinal ganglion cells.

In order to reduce the computation cost of the maximum entropy method, Ganmor et al. [17] explored the possibility to reduce the number of connections and parameters in the network by model pruning. The authors observed that while the majority of the neuron pairs have weak correlation, some groups of neurons can be strongly correlated. They constructed a "motifs" network, consisting of a triplet of cells ("unfustrated" triangles in the Ising model) that are highly correlated or have adjacent receptive fields, and able to produce a very good approximation to the full pairwise model with a relatively small number of connections. While the pairwise models perform well with smaller networks ($N \leq 40$), it cannot model synchronous spikes in networks with larger populations. To address the said issue, Tkačik et al. [58] proposed the K-pairwise model to better formulate the spatial correlation in the data. They introduced an additional constraint into the energy function where the probability $P_N(K)$ of K out of N neurons spike at time-bin Δt is also taken into account:

$$\mathcal{H}(s) = \sum_{i=1}^{N} h_i s_i + \frac{1}{2} \sum_{i,j=1}^{N} J_{ij} s_i s_j + V \sum_{i=1}^{N} s_i, \tag{2.3}$$

where V is an effective potential, tuned to match the observed distribution $P_N(K)$. The additional K-spike constraint allows the Ising model to produce excellent results for larger populations of neurons ($N \approx 100$) while preserving the low-order statistics of the observed data.

2.2.2 Dichotomized Gaussian method

Despite the success of modelling the low-level statistics of binary neural population patterns, the Ising model cannot scale well in high dimensions as the number of possible states grows exponentially [57]. Macke et al. [31] proposed to model the binary spike

trains with the dichotomized Gaussian (DG) distribution instead, where the mean and covariance of the distribution are be specified. The model uses a multivariate normal distribution to generate latent continuous random variables which are then thresholded to generate binary variables representing spike trains.

2.2.3 Deep generative model method

The above-mentioned statistical models have shown great success in modelling correlated binary spike trains, though, many of these methods require optimizations or restrictions for specific cortical areas, and often cannot generalize. With the recent popularity of GAN being used across a vast variety of domains and data-types, including images [26], text [18], audio [24] and many more. The GAN framework could be an excellent alternative in modelling neuronal activities. (We discuss further about GAN and its formulation in Section 2.3)

Spike-GAN [36] demonstrated that GAN can model neural spikes that accurately match the statistics of real recorded spiking behaviour from a small number of neurons. Moreover, the discriminator in Spike-GAN is able to learn to detect which population activity pattern is the relevant feature, and this can provide insights into how a population of neurons encodes information. Ramesh et al. [47] trained a conditional GAN [35], conditioned on the stimulus, to generate multivariate binary spike trains. They fitted the generative model with recorded data in the V1 area of the macaque visual cortex, and GAN generated spike trains were able to capture the firing rate and pairwise correlation statistics better than the dichotomized Gaussian model and a deep supervised model.

Nevertheless, one major hurdle of applying deep learning methods on spike trains is that spike trains are discrete in nature, meaning that they cannot be subject to any continuous increment or decrement. For instance, Ramesh et al. [47] used REINFORCE gradient estimate [61] to train the generator in order to perform back-propagation on discrete data. Still, gradient estimation with the REINFORCE approach yields large variance, which is known to be challenging for optimization [32, 63]. It remains a difficult task to optimize deep generative models for discrete data with back-propagation, which is key for training deep neural networks. Hence, we are interested in the possibility of synthesizing the continuous fluorescent calcium signals with GAN, in addition to the advantages that calcium imaging brings.

2.3 Generative Adversarial Networks

A typical GAN model consists of a generator G, which attempts to generate convincing samples $\hat{x}$ from the latent space P_Z, whereas the discriminator D learns to distinguish generated sample distributions $P_{\hat{x}}$ from the real data distribution P_X. In the original GANs framework introduced by Goodfellow et al. [20], the discriminator functions like a logistic classifier, trained to identify real samples from generated samples. The discriminator has the loss function:

$$\mathcal{L}_D = \mathop{\mathbb{E}}_{x \sim P_X}[\log D(x)] - \mathop{\mathbb{E}}_{z \sim P_Z}[\log(1 - D(G(z)))] \tag{2.4}$$

Whereas the generator's objective is simply maximizing the loss of the discriminator, hence the two networks are trained jointly to perform this minimax game:

$$\arg\max_G \min_D \mathcal{L}_D \tag{2.5}$$

However, this zero-sum formulation and the objective of minimizing the Jensen-Shannon divergence between the original data distribution and generated data distribution made GAN models notoriously difficult to train. For instance, GAN is prone to mode collapse where the generator focuses on generating a small subset of the dataset, instead of learning the true data distribution [8]. Another major challenge of GAN is when the discriminator is doing too well such that the gradient signals for the generator vanish, an issue known as vanishing gradients [19, 43].

Wasserstein GAN (WGAN), introduced in Arjovsky et al. [2], propose instead to minimize the continuous and smoother Earth-Mover distance, also known as the 1st Wasserstein distance. The value function in WGAN uses the Kantorovich-Rubinstein duality formulation [59] to measure the Wasserstein distance between the real and generated data distribution:

$$W(P_X, P_{\hat{X}} = \mathop{\mathbb{E}}_{x \sim P_X}[F(x)] - \mathop{\mathbb{E}}_{\hat{x} \sim P_{\hat{X}}}[F(\hat{x})] \tag{2.6}$$

where F is a set of 1-Lipschitz functions. In WGAN, the discriminator is called the critic due to the fact that the discriminator is not behaving as a classifier, but a value function instead. In order to enforce the 1-Lipschitz condition on the critic, the weights of the critic are clipped within a predefined range $[-c, c]$ where c is a hyper-parameter. In the original GAN formulation, the loss function reflects how well the discriminator is performing, regardless of the quality of the generated samples. With WGAN, the value function appears to correlate with the generated sample quality. The weight constraint

method to enforce the Lipschitz condition is simple to implement though it can also be problematic: Gulrajani et al. [21] have shown that WGAN is extremely sensitive to the hyper-parameter c. Moreover, weight clipping also greatly limits the network capacity and its capability to model complex functions.

Gulrajani et al. [21] suggest an alternative method, WGAN with Gradient Penalty (WGAN-GP), to enforce the Lipschitz condition more effectively. WGAN-GP takes advantage of the fact that a differential function is 1-Lipschitz if and only if the norm of the gradient is 1, hence they propose an objective function that penalizes the critic if its gradient norm is not 1:

$$\mathcal{L}_D = \mathop{\mathbb{E}}_{z\sim P_Z}[D(G(z))] - \mathop{\mathbb{E}}_{x\sim P_X}[D(x)] + \lambda \mathop{\mathbb{E}}_{\tilde{x}\sim P_{\tilde{X}}}[(\| \nabla_{\tilde{x}} D(\tilde{x}) \|_2 - 1)^2] \tag{2.7}$$

where λ denotes the gradient penalty coefficient, $\tilde{x} = \varepsilon x + (1-\varepsilon)\hat{x}$ are samples taken between the real and generated data distribution. The gradient penalty formulation increases the computation complexity, though it significantly improves the network training stability and convergence. In this work, we attempt to model calcium imaging data using the WGAN-GP formulation (Equation 2.7) without the need of specific constraint or information of the neuronal activities into the model objective function.

Chapter 3

Methods

3.1 CalciumGAN architecture

A described in Section 2.3, a typical GAN model consists of two networks, a generator and a discriminator. A sample noise is drawn from a Gaussian distribution $z \sim \mathcal{N}$ with a given mean and standard deviation. The generator receives the input z and interpolates it, usually via layers of fully-connected or transposed convolution layers, and generate output $\hat{x}$ which has the same dimension as real data x. The discriminator then receives the generated output and real data and learns to distinguish between the two. One great advantage of the GAN framework is that the architectures of the generator and discriminator are modular while the training procedure remains the same, hence can be easily adapted to a different type of data. In this work, we adapted the WaveGAN architecture [11], which has shown promising results in audio signal generation. The following sections describe the detail of our GAN model, which we named CalciumGAN, Table 3.1 shows the exact architecture.

3.1.1 Generator

Similar to the generator of WaveGAN, we used 1-dimensional transposed convolution layers (sometimes referred to as deconvolution) to up-sample the input noise. Each transposed convolution layer (see Section 3.2.1 for additional detail) was followed by an activation layer. In addition, we added Layer Normalization [25] in between each convolution and activation, in order to stabilize training as well as to make the operation compatible with the WGAN-GP framework. To improve the model learning

performance and stability, the calcium signals were scaled to the range between 0 and 1 by normalizing with the maximum value of the calcium signal in the data. Correspondingly, we chose sigmoid activation in the output layer of the generator and then re-scaled the signals to their original range before inferring their spike trains.

3.1.2 Discriminator

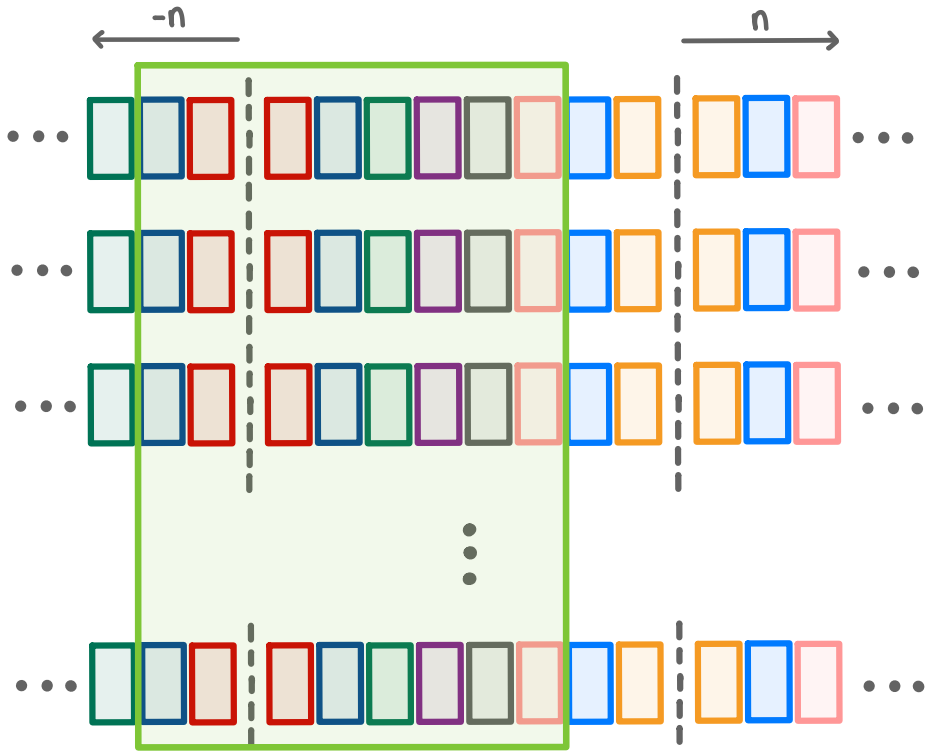


Figure 3.1: Illustration of the 1-dimensional Phase Shuffle mechanism. Each box denotes an activated unit after a convolution layer, and the units are mirrored along the grey dash lines. The large box in light green represents the output of the Phase Shuffle layer with $n = -2$.

The architecture of the discriminator in our model is largely a mirror of the generator, with the exception of the removal of Layer Normalization and instead of up-sampling the input with transposed convolution, we used a simple convolution layer instead. Samples generated using transposed convolution often exhibit the "checkerboard" artifacts described by Odena et al. [39], where the output exhibits repeated patterns (usually very subtle to the eye) due to a filter being applied unevenly to the receptive field. In the context of signal generation, the discrimination could exploit the periodic artifacts pattern and learn a naive policy to reject generated samples. Donahue et al. [11] proposed the

Phase Shuffle mechanism in the discriminator to address the aforementioned issue. The Phase Shuffle layer randomly shifts the activated units after each convolution layer by $[-n, n]$, in order to distort the periodic pattern. Hence, the resulting samples constitute a more challenging task for the discriminator. Figure 3.1 shows a simple illustration of the Phase Shuffle operation. In our network, we incorporated the Phase Shuffle operation, as well as using a kernel size that is divisible by the stride size, as suggested in Odena et al. [39]. This led to a noticeable improvement in the generated samples. We apply the Phase Shuffle operation after each convolution layer. We use the WGAN-GP formulation of the loss function without the need of incorporating any information of the neural activities into the objective function as specified in Equation 2.7.

3.2 CalciumGAN pipeline

We devised a consistent model analysis pipeline to evaluate the quality of samples generated by the model, as well as its ability to generalize, in the context of neuronal population spiking activities. The complete model analysis pipeline is shown in Figure 3.2.

In order to train and evaluate our GAN model, we have to first preprocess the calcium signals so that they have a standardized format. For calcium imaging data of N neurons with a recorded length of L, we would receive a raw data shape of (L, N). We then used a slicing window of size T to segment the data along the time dimension into M segments, resulting in a matrix with shape (M, T, N). To improve the network training performance, we scale the raw calcium signals x to the range $[0, 1]$ before we train our generative model:

$$x_{\min}, x_{\max} = min(x), max(x) \tag{3.1}$$

$$x_{[0,1]} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{3.2}$$

We use $a_{[0,1]}$ to denote data a that has a range of $[0, 1]$.

Since we evaluate our model performance in terms of spike activities, we needed a deconvolution algorithm to infer the spike trains from calcium signals. In this work, we used the Online Active Set method to Infer Spikes (OASIS) deconvolution algorithm [15] for its fast online deconvolution performance. Prior to inferring the spiking activities from the generated signals $\hat{x}_{[0,1]}$, we first have to scale the signal back to the

same range as the raw calcium signals:

$$\hat{x} = \hat{x}_{[0,1]}(x_{\max} - x_{\min}) + x_{\max} \tag{3.3}$$

We inferred the spike trains from the generated signals as well as the real recorded data with OASIS in order to ensure the possible biases of the deconvolution algorithm are the same for both data. Now that we have obtained spiking activities from both the real recorded data and generated samples, we then measure spike train similarities and statistics using the Electrophysiology Analysis Toolkit (Elephant) [38]. We evaluate the performance of our model with the following spike train statistics and similarities: (a) mean firing rate for evaluating single neuron statistics; (b) pairwise Pearson correlation coefficient for evaluating pairwise statistics; (c) pairwise van-Rossum [49] distance for evaluating general spike train similarity. Importantly, 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.

3.2.1 Spatial-temporal convolution

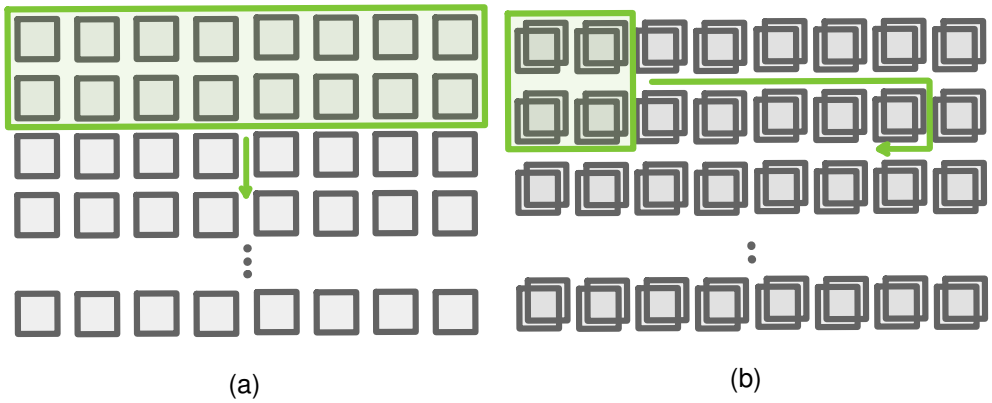


Figure 3.3: Illustrations of 1-dimensional and 2-dimensional kernels. (a) illustrates 1-dimension convolution with kernel size of 2; (b) illustrates 2-dimensional convolution with a 2×2 kernel, similar to the one in CalciumGAN-2D. In (b) each datum has 2 channels, as of the case for CalciumGAN-2D FFT.

1-dimensional transposed convolution layers were used in the generator of WaveGAN, where the filter interpolates on the temporal dimension. As mentioned in Section 2.2,

many works have shown that modelling both the spatial and temporal dimensions of spiking activities see significant improvement in the quality of the generated data [46, 57, 58]. Hence, in addition to the 1-dimensional transposed convolution along the temporal domain (and 1-dimensional convolution in the discriminator), we also experiment with 2-dimensional transposed convolution where the filters would apply to both spatial and temporal dimensions (denoted as CalciumGAN-2D). Figure 3.3 shows a simple illustration of 1-dimensional and 2-dimensional kernel. We simply add an additional dimension to our data, from shape (M,T,N) to $(M,T,N,1)$, similar to an image with a single channel. We also adapted the Phase Shuffle layer to work with 3-dimensional input, by randomly shifting the activated units by $[-m,m]$ and $[-n,n]$ in the spatial and temporal domain respectively. The architecture of CalciumGAN-2D is shown in Table A.2.

3.2.2 Frequency-domain representation

There is an increasing number of works done in audio synthesis where the deep neural network operates on raw waveform signal directly [29, 41]. Nevertheless, the use of spectrogram representation is still a common practice in many works [12, 13, 22]. We are interested in whether the frequency representation of the calcium signals, instead of in the time-domain, would allow the network to learn the additional structure from the data. To do so, we apply Fast Fourier Transform (FFT) to convert the segmented calcium signals in waveform into their frequency-domain representation. Since FFT provides a complex number, we separate the real and imaginary parts into two channels, yielding samples with shape $(M,T,N,2)$, hence we also use the CalciumGAN-2D variant here. After converting to frequency-domain, we scale the signals range to 0 to 1 and follow the same procedure as stated above. In the post-processing step, after rescaling the generated data to its original range, we convert the sample $\hat{x}$ to complex number by merging the real and imaginary part $\hat{x} = \hat{x}[\ldots,0] + \hat{x}[\ldots,1] * 1j$, then apply inverse Fast Fourier Transform (iFFT) to convert the sample to its time-domain representation and perform spike analysis. The orange boxes in Figure 3.2 show the additional steps we took in the CalciumGAN-2D FFT variant.

Layer	Output shape
Input	(bs, 32)
Dense	(bs, 2048)
LeakyRelu	(bs, 2048)
Reshape	(bs, 64, 32)
Conv1DTransposed	(bs, 128, 320)
LayerNorm	(bs, 128, 320)
LeakyRelu	(bs, 128, 320)
Conv1DTransposed	(bs, 256, 256)
LayerNorm	(bs, 256, 256)
LeakyRelu	(bs, 256, 256)
Conv1DTransposed	(bs, 512, 192)
LayerNorm	(bs, 512, 192)
LeakyRelu	(bs, 512, 192)
Conv1DTransposed	(bs, 1024, 128)
LayerNorm	(bs, 1024, 128)
LeakyRelu	(bs, 1024, 128)
Conv1DTransposed	(bs, 2048, 102)
LayerNorm	(bs, 2048, 102)
LeakyRelu	(bs, 2048, 102)
Dense	(bs, 2048, 102)
Sigmoid	(bs, 2048, 102)

(a) Generator architecture

Layer	Output shape
Input	(bs, 2048, 102)
Conv1D	(bs, 1024, 64)
LeakyRelu	(bs, 1024, 64)
PhaseShuffle	(bs, 1024, 64)
Conv1D	(bs, 512, 128)
LeakyRelu	(bs, 512, 128)
PhaseShuffle	(bs, 512, 128)
Conv1D	(bs, 256, 192)
LeakyRelu	(bs, 256, 192)
PhaseShuffle	(bs, 256, 192)
Conv1D	(bs, 128, 256)
LeakyRelu	(bs, 128, 256)
PhaseShuffle	(bs, 128, 256)
Conv1D	(bs, 64, 320)
LeakyRelu	(bs, 64, 320)
Flatten	(bs, 20480)
Dense	(bs, 1)

(b) Discriminator architecture

Table 3.1: The generator (a) and discriminator (b) architecture of CalciumGAN. The generator consists of 4,375,740 parameters, and the discriminator consists of 4,110,273 parameters. Note bs denotes batch size.

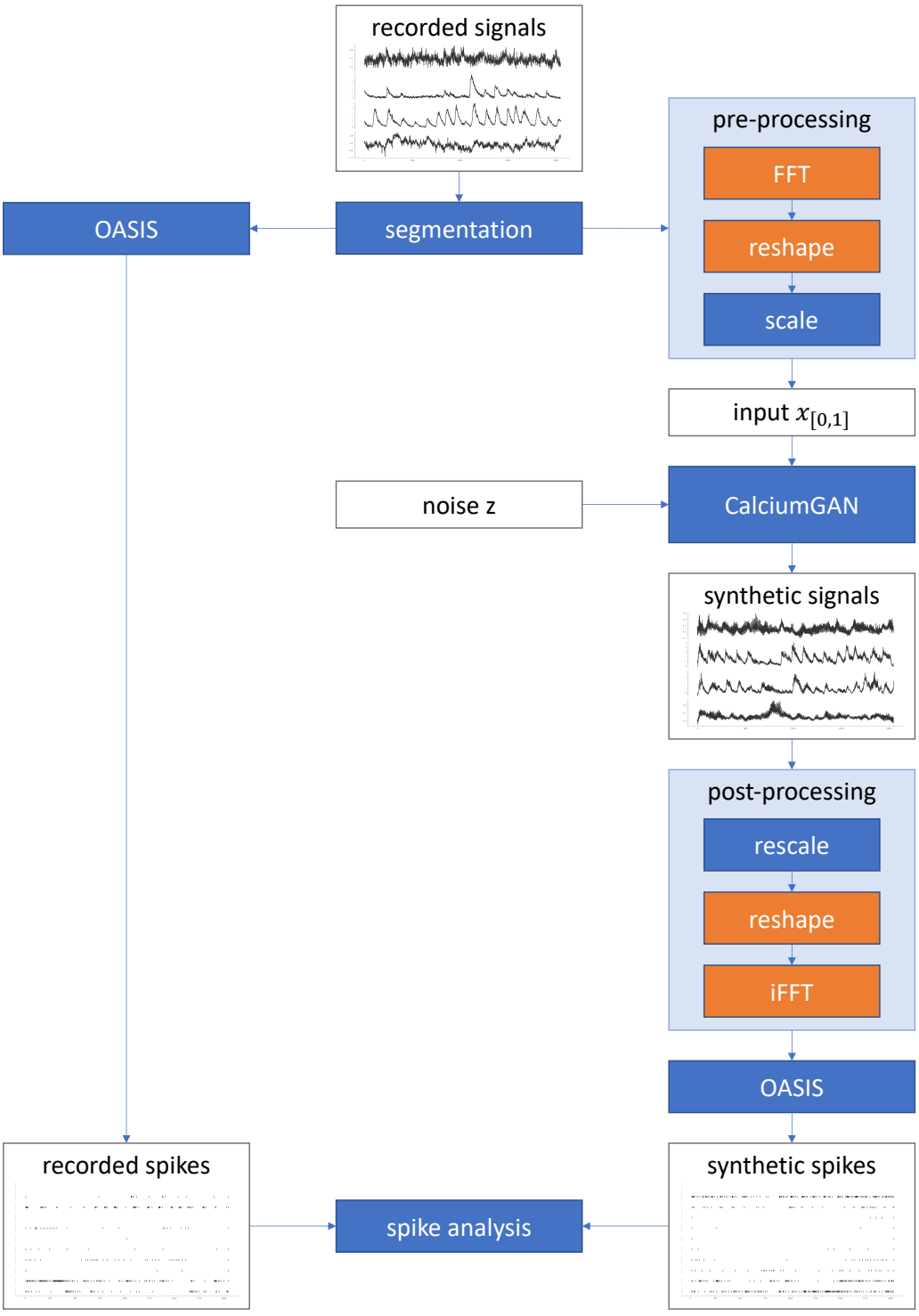


Figure 3.2: Pipeline diagram of a CalciumGAN analysis. White boxes illustrate data in different processing stages. Blue boxes illustrate operations and analysis steps. Orange boxes represent optional operations, see Section 3.2.1 and Section 3.2.2 for the CalciumGAN-2D specification and frequency-domain representation.

Chapter 4

Results

In this chapter, we present the training procedure and evaluate the performance of our model. We first fit CalciumGAN with synthetic data with known ground-truth statistics and show that the generated data by our model can closely resemble the underlying data distribution. We then train CalciumGAN and its variants with calcium imaging data recorded from the primary visual cortex of behaving mice and compare the statistics of the recorded and generated data. The software codebase for this project is available at `github.com/bryanlimy/calciumgan`.

4.1 CalciumGAN model training

We trained both the generator and discriminator with the WGAN-GP framework, with 5 discriminator update steps for each generator update step. We then used Adam optimizer [28] to optimize both networks. The vast majority of the code was written in Python, where the deep neural networks were implemented in TensorFlow [1] because of its vast community support and hardware optimization. In order to find the optimal hyper-parameters, we performed a Random Search [3] using the Keras Tuner API [40]. The exact hyper-parameters being used in this work can be found in Table 4.1. To speed up the training process, we incorporated Mixed Precision training [34] in our codebase. The majority of the computations was performed in `float16` representation and we only kept the loss calculation in `float32` to avoid numeric underflow or overflow, essentially allowing us to fit twice as many data to the computing hardware such as GPUs and TPUs. As a result, we are able to train our model with a batch size of $bs = 128$ on a single NVIDIA RTX 2080 TI GPU.

Hyper-parameters	Value
Filters	64
Kernel size	24
Stride	2
Noise dimension	32
Critic updates	5
Gradient penalty (λ)	10
Batch size (bs)	128
Epochs	400
Learning rate	0.0001
Phase shuffle (m)	10

Table 4.1: Hyperparamters of CalciumGAN after Random Search.

4.2 Dichotomized Gaussian data

In order to verify that CalciumGAN is able to learn the underlying distribution and statistics of the training data, we generated our own ground-truth dataset with pre-defined mean and covariance using the DG model discussed in Section 2.2.2. To generate data from the DG model, we used the sample means and sample covariances obtained from real recorded data (details of the recorded data are described in Section 4.3). In alignment with the recorded data, we generated correlated spike trains for $N = 102$ neurons with a duration of $L = 899$ seconds and at 24 Hz, hence a matrix with shape $(21576, 102)$. In order to obtain calcium-like signals c from spike trains s with length T, we convolved the generated spike trains with a calcium response kernel and added noise, as described in Friedrich et al. [15]:

$$s_t = g s_{t-1} + s_t \qquad 1 \leq t \leq T \tag{4.1}$$

$$c = b + s + \sigma u \qquad u \sim \mathcal{N}(0,1) \tag{4.2}$$

where g denotes a finite impulse response filter, b is the baseline value of the signal and σ is the noise standard deviation. In our work, we set $g = 0.95$, $\sigma = 0.3$ and $b = 0$. We scale the signals range from 0 to 1. The data is then segmented using a sliding window along the time dimension with a stride of 2 and a window size of $T = 2048$ (around 85 seconds in experiment time). We apply the segmentation procedure to both the signal and spike data, hence resulting in two matrices with shape $(9754, 2048, 102)$. Examples

of signals and spikes generated from the DG model can be found in Figure 4.2a.

4.2.1 Synthetic data mimicking dichotomized Gaussian data

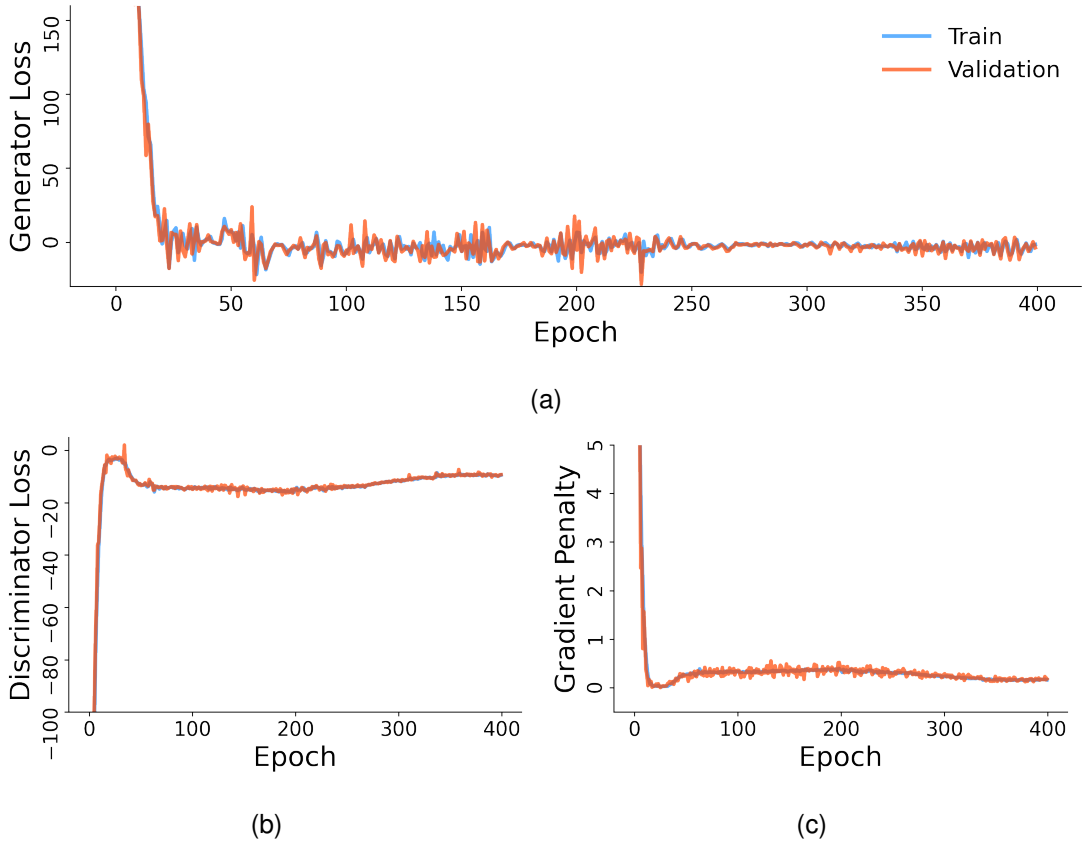
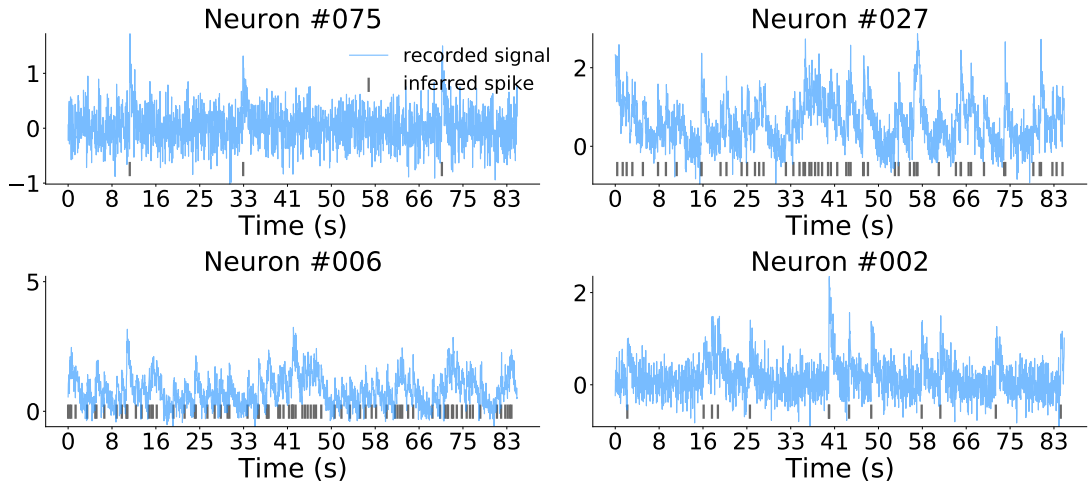


Figure 4.1: The (a) generator loss and the (b) discriminator loss gradient penalty of CalciumGAN trained on the DG data. The blue line and orange line indicate the training and validation performance respectively.

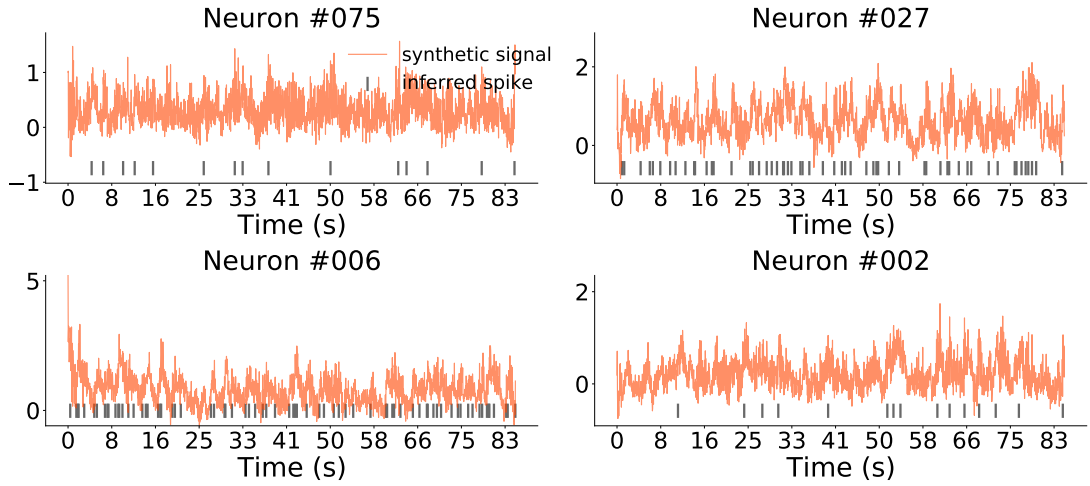
We first fit CalciumGAN to the artificial dataset sampled from the DG distribution until the model converges. We trained the model for 400 epochs with 8,754 samples and held out 1,000 samples for evaluation. Figure 4.1 shows the loss of the generator and discriminator as well as the gradient penalty of the GAN model, the training stabilized within 200 epochs in most cases. Since we defined the model from which we generated the training dataset, we can validate the statistics of the dataset generated by CalciumGAN on the known ground-truth directly. Examples of generated signals and spikes can be found in Figure 4.2b.

We estimated the mean firing rates and the covariances of data generated by CalciumGAN and compared it to the DG ones (Figure 4.3). We plotted the values of 5 samples

for each neuron and neuron-pair and sorted them by their mean in ascending order. Our model is able to reliably capture the firing rate very well, with a root mean square error of 0.0997 Hz. The variation of the firing rate across samples matched with those of the ground-truth data. The majority of the neuron pairs have low correlation which was also found in the generated data. The neuron pairs that have highly positive and highly negative covariance also have a greater variation across samples.

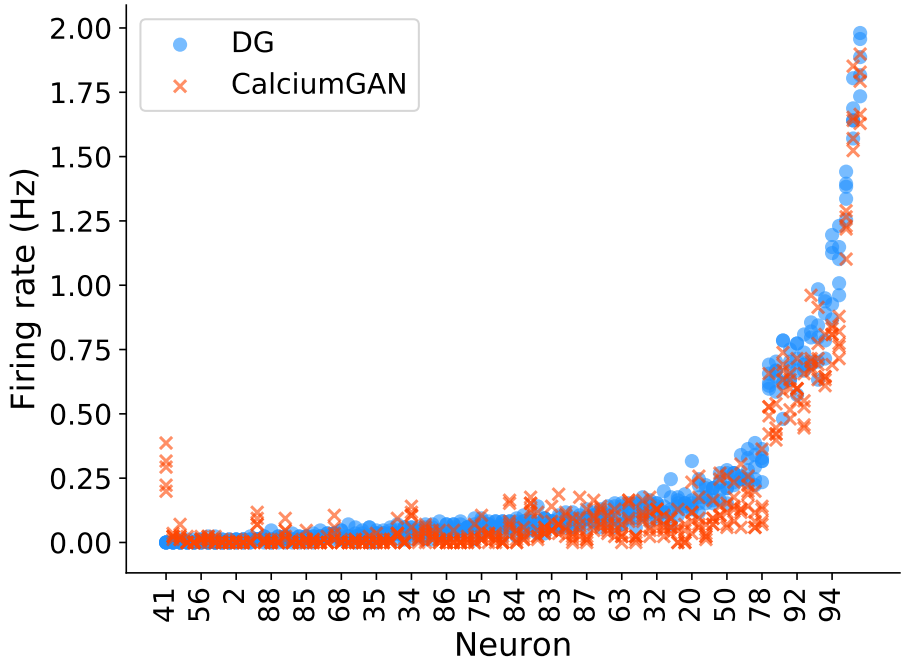


(a)

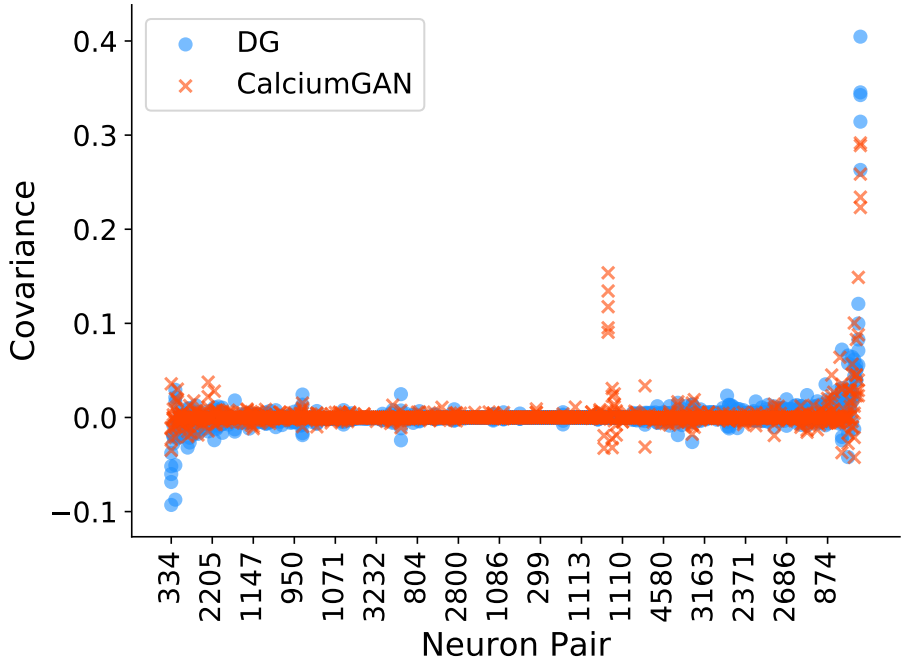


(b)

Figure 4.2: Calcium signals and inferred spike trains (in gray) of randomly selected neurons. (a) shows the DG data (in blue) and (b) shows synthetic data (in orange) generated by CalciumGAN trained on the DG data. Notice that the artificial signal data transformed from DG spike data do not have the peak and decay characteristics of typical calcium imaging data.



(a)



(b)

Figure 4.3: CalciumGAN trained on the DG dataset with known ground-truth. (a) Mean firing rate of each neuron. (b) Neuron pairwise covariance. Blue dots represent DG data and orange crosses present generated data. 5 randomly selected samples for each neuron and neuron-pair were displayed in both graphs, where the order on the x-axis was sorted by the mean of the firing rate and covariance respectively. In (b), only every 10^{th} pair is displayed, for clarity. Here, we compare both the trend and variation of the generated data statistics with the DG data.

4.3 Two-photon calcium imaging recorded data

Date	Sampling rate	Duration	Num. trials	Avg. duration	Avg. firing rate
Day 1	24	894.73s	129	6.94s	58.07Hz
Day 4	24	898.45s	203	4.43s	35.83Hz

Table 4.2: Information about the neuron population of $N = 102$ recorded from the primary visual cortex of a behaving mouse on the 1st day and 4th day of the experiment.

After validating our model on data with known ground-truth, we applied CalciumGAN on real two-photon calcium imaging data recorded in the primary visual cortex of mice performing a virtual reality task. The data were collected with the same setup as specified in Pakan et al. [42] and Henschke et al. [23]. Head-fixed mice were placed on a cylindrical treadmill and navigated a virtual corridor rendered on two monitors that covered the majority of their visual field. A lick spout was placed in front of the mice, where a water drop would be made available to the mice as a reward if it licked at the correct location within the virtual environment. Hence, the mice would learn to utilize both the visual information and self-motion feedback in order to maximize the rewards. Neuronal activity was monitored from the same primary visual cortex populations over multiple consecutive behavioural sessions. Table 4.2 shows the basic information of the data collected from the mice experiments. In this work, we are using neuron population data recorded on the 4th day of the experiment, where the mice were quite familiar with the virtual environment and the given task. In this particular recording, neurons were labelled with GCamP6f, and $N = 102$ neurons were recorded at a sampling rate of 24 Hz, the mouse performed 204 trials in $L = 898.2$ seconds (raw data shape $(21556, 102)$). Due to the fact that GAN models require a significant amount of training data, information about the trial and position of the mice in the virtual environment were not used in this work.

We applied the OASIS AR1 deconvolution algorithm to infer the spike activities from the recorded calcium signals and performed the same normalization and segmentation steps as mentioned in Section 4.2. Both calcium signals and inferred spike trains have shape $(9754, 2048, 102)$. Figure 4.4a shows examples of the recorded calcium signals and inferred spike trains. There are multiple challenges for both the generator and discriminator to learn from the calcium imaging signals. Since data were segmented with a sliding window and the information of the trial was not used, some samples

might consist of abnormal signal activity, such as a peak being cropped off. Generated signals could have the same number of peaks or ranges, though might not preserve the peak and decay characteristics of calcium imaging data. Real and synthetic activity from less active neurons might be more difficult for the discriminator to distinguish due to the absence of prominent spiking characteristics.

4.3.1 Synthetic data mimicking recorded data

We tested CalciumGAN with the data recorded from the primary visual cortex of a trained mouse. Similar to the DG analysis, we trained the model for 400 epochs, with 8,754 training samples, and 1,000 samples were held out for evaluation. Note that since we are not taking the trial and position of the mice in the virtual environment into consideration when training the model, the generated data and the evaluation data do not have a one-to-one mapping.

We first inspect the generated data and the deconvolved spike trains visually. The calcium signals and inferred spike trains of 6 randomly selected neurons from a randomly selected sample are shown in Figure 4.4. Both the synthetic raw traces as well as the inferred spikes visually match the characteristics of the recorded ones.

We then compared the spiking characteristics across the whole population. Figure 4.5 shows the inferred spike trains of the complete 102 neurons population from a randomly selected sample of the real and the synthetic data, with the distribution histogram plotted on the x and y axis. The synthetic data mimics the firing patterns across neurons and across time remarkably well with occasional small deviations in the rates at particular temporal intervals. Notably, the samples are clearly not identical meaning that the network did not just replicate the training set data.

In order to examine if CalciumGAN is able to capture the first and second-order statistics of the recorded data, we measured the mean firing rate, pairwise correlation, and van-Rossum distance (see Figure 4.6). The randomly selected neurons shown in Figure 4.6a all have very distinct firing rate distributions, and CalciumGAN is able to model all of them relatively well, with KL divergence of 0.42, 0.11, 0.09, 0.66, 0.25 and 0.40 with respect to the recorded firing rate over 1000 samples. We show the pairwise van-Rossum distance of the same neuron between recorded and generated data across 50 samples in Figure 4.6c as sorted heatmaps. Less active neurons, such as neuron 75, have a low distance value across samples, mainly due to the scarcity of

firing events. Conversely, a high-frequency neuron, such as neuron 6, exhibits a trend of lower distance values in the diagonal of the heatmap, implying the existence of a pair of a recorded and generated sample that is similar. In order to ensure that the data generated by our model capture the underlying distribution of the training data, we also compute the KL divergence between the distributions of the above-mentioned metrics (see Figure 4.7). CalciumGAN was able to model all 3 of the statistics of the recorded data, with most samples having KL divergence values of less than 1.5. Note that we measure the pairwise distance of the same neuron across 50 samples in Figure 4.6c, whereas, in Figure 4.7c, we measure the pairwise van-Rossum distance of each neuron with respect to other neurons within the same sample.

We also fit the DG model with the recorded data and measured the above-mentioned statistics of DG generated spike trains as a baseline. Table 4.3 showed the mean KL divergence of the DG generated data against the recorded data, as well as the statistics of CalciumGAN and its variations.

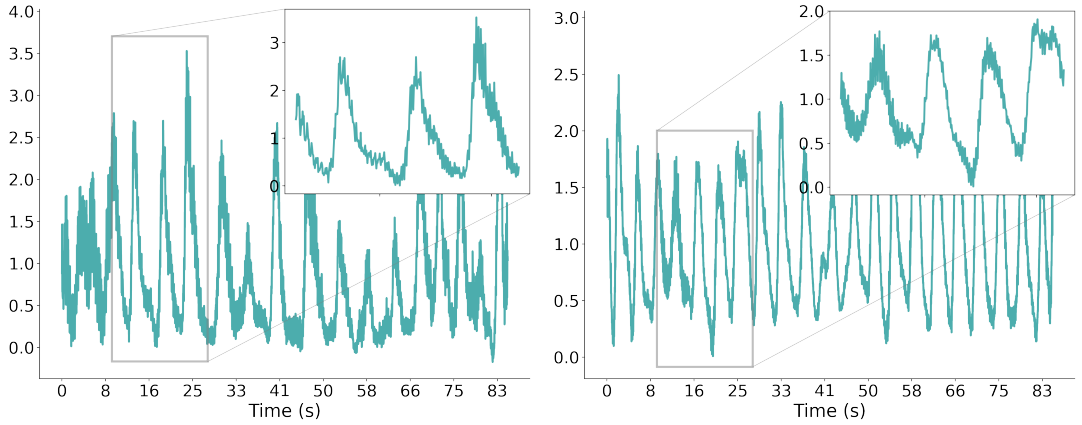
Model	mean firing rate	pairwise correlation	van-Rossum distance
CalciumGAN	**0.4536**	**0.0824**	0.5839
– without Phase Shuffle	1.0478	0.1148	0.7273
– 2D	0.5351	0.1195	0.5897
– 2D FFT	0.4992	0.1054	**0.5797**
DG	1.0592	0.3379	1.0287

Table 4.3: The mean KL divergence value in each metrics of different models. The result with the lowest value is highlighted.

4.3.2 Phase Shuffle

In order to reduce the effect of the "checkerboard" artifact, we adapted the Phase Shuffle mechanism (see 3.1.2) in the discriminator. In this section, we examine the effectiveness of Phase Shuffle in terms of the visual quality of the generated traces as well as the effect it had on the inferred spike trains. A common characteristic of the calcium indicators when an action potential occurs is a sharp onset followed by a slow decay in the signal [16]. In Figure 4.8, we can see that such characteristics in the calcium traces were more prominent when Phase Shuffle was enabled. We believe that such differences in the generation quality exist mainly because of the repetitive patterns in the transposed

convolution layer [39], since the discriminator can simply distinguish generated samples from real samples by learning if such patterns exist. As the Phase Shuffle mechanism shifts the temporal dimension (by 10 units in our experiment) randomly, it forces the discriminator to learn from other features in the data instead of the "shortcut" provided by the (undesired) nature of transposed convolution.



(a) (b)

Figure 4.8: Generated traces of Neuron 6 from a randomly selected sample with (a) PhaseShuffle $= 10$ and (b) PhaseShuffle $= 0$. The sharp rise to peak followed by a tail of decaying signal is less observable in when Phase Shuffle is disabled.

Moreover, not only did Phase Shuffle affect the visual quality of the generated samples, but it also impacted the spike train statistics. The traces generated without Phase Shuffle lack the spiking characteristics, which made it more difficult for the deconvolution algorithm to register a spike in the data, thus increasing the inaccuracy of the inferred spike trains. When comparing the KL divergence of the spike train statistics, the samples generated without Phase Shuffle suffer worse results across the 3 statistics (see Table 4.3), especially with mean firing rate. The mean firing rate, pairwise correlation and van-Rossum distance of the randomly selected samples and neurons generated by CalciumGAN without Phase Shuffle are shown in Figure A.4, and the KL divergence between the generated samples and recorded data are shown in Figure A.3.

4.3.3 Spatio-temporal convolution

The generative network we have experimented with thus far uses a 1-dimensional convolution layer, where only the temporal dimension of the data is being modelled.

Here we expanded the network to use a 2-dimensional kernel which is supposed to take both spatial and temporal information into consideration. We modified the network to use a 8×8 kernel with a filter size of 20 in both the generator and discriminator so that the model has a similar number of trainable parameters as CalciumGAN (see Table A.2 for the exact model architecture). Figure A.6 shows a raster plot of all 102 neurons from a randomly selected trial, and Figure A.8 shows the statistics of the inferred spike trains measured against the recorded data. We expanded the Phase Shuffle layer to support 3-dimensional data, where the random shift operation was applied to both spatial and temporal dimensions. In our experiment, the effectiveness of the Phase Shuffle operation preserves the calcium signal characteristic in the generated data (see Figure A.5 for examples of generated traces). Overall, we observed slightly worse though similar results between the use of 1-dimensional convolution versus 2-dimensional convolution (see Table 4.3).

4.3.4 Frequency-domain data

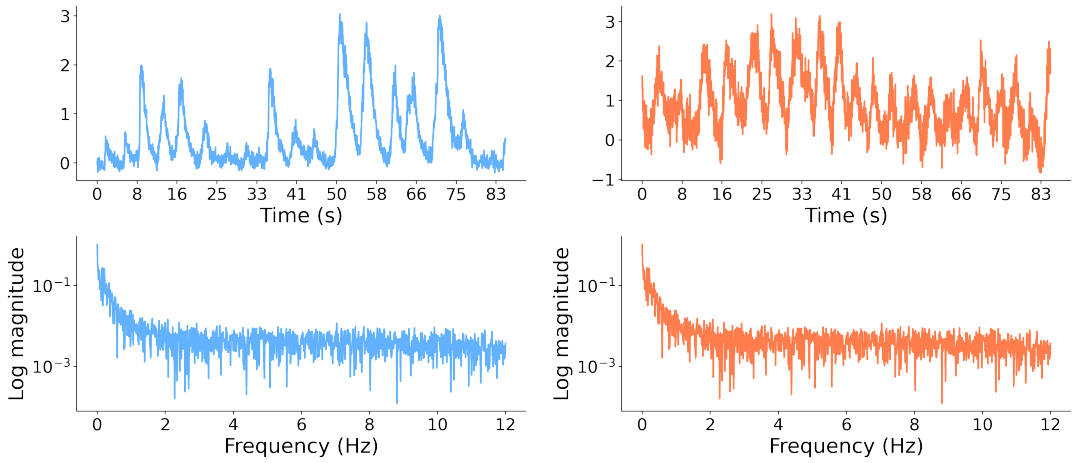


Figure 4.9: (a) recorded data and (b) generated sample of Neuron 27 in (top) time-domain and (bottom) frequency-domain.

Next, we examined whether the GAN model can better capture the statistics of the recorded data when representing them in their frequency-domain. We used CalciumGAN-2D as the sample model with the exception of the data having 2 channels instead of 1 channel because of the complex number representation. Samples of recorded and generated data in the frequency-domain are presented in Figure 4.9; the calcium traces and

spike trains statistics are shown in Section A.3. With the frequency-domain representation, we obtained marginally better results compared to CalciumGAN-2D, and the mean KL divergence of the van-Rossum distance is slightly better than for CalciumGAN (see Table 4.3). Moreover, with the additional computational cost for both 2-dimensional convolution as well as the Fourier transform and inverse Fourier transform operations, we use the raw calcium traces in the remainder of this work.

4.3.5 Untrained mice data

The fluorescent calcium signals we have been experimented with thus far were data recorded on the 4^{th} day of the animal experiment, where the mice were already familiar with the specific task. We also wanted to verify if CalciumGAN is able to learn from neural activities that are more stochastic and potentially less correlated. To this end, we trained the model on the neuronal population data recorded on the 1^{st} day of the mice experiment (average firing rate of 58.07 Hz on day 1 versus 35.83 Hz on day 4). As shown in the raster plots on the 1^{st} day (Figure 4.10) and 4^{th} day (Figure 4.5) of the mice experiment, most neurons recorded from the untrained mice were significantly more active. Nonetheless, the generated data were able to reflect the low-level statistics of the recorded data, with mean KL divergence of 0.38, 0.06 and 0.60 when comparing with the mean firing rate, pairwise correlation and van-Rossum distance, respectively (see 4.11). Figure A.13 shows first and second-order statistics of the generated samples. Overall, CalciumGAN was able to capture the statistics and underlying distribution of the real calcium imaging data acquired in the primary visual cortex of awake, behaving mice.

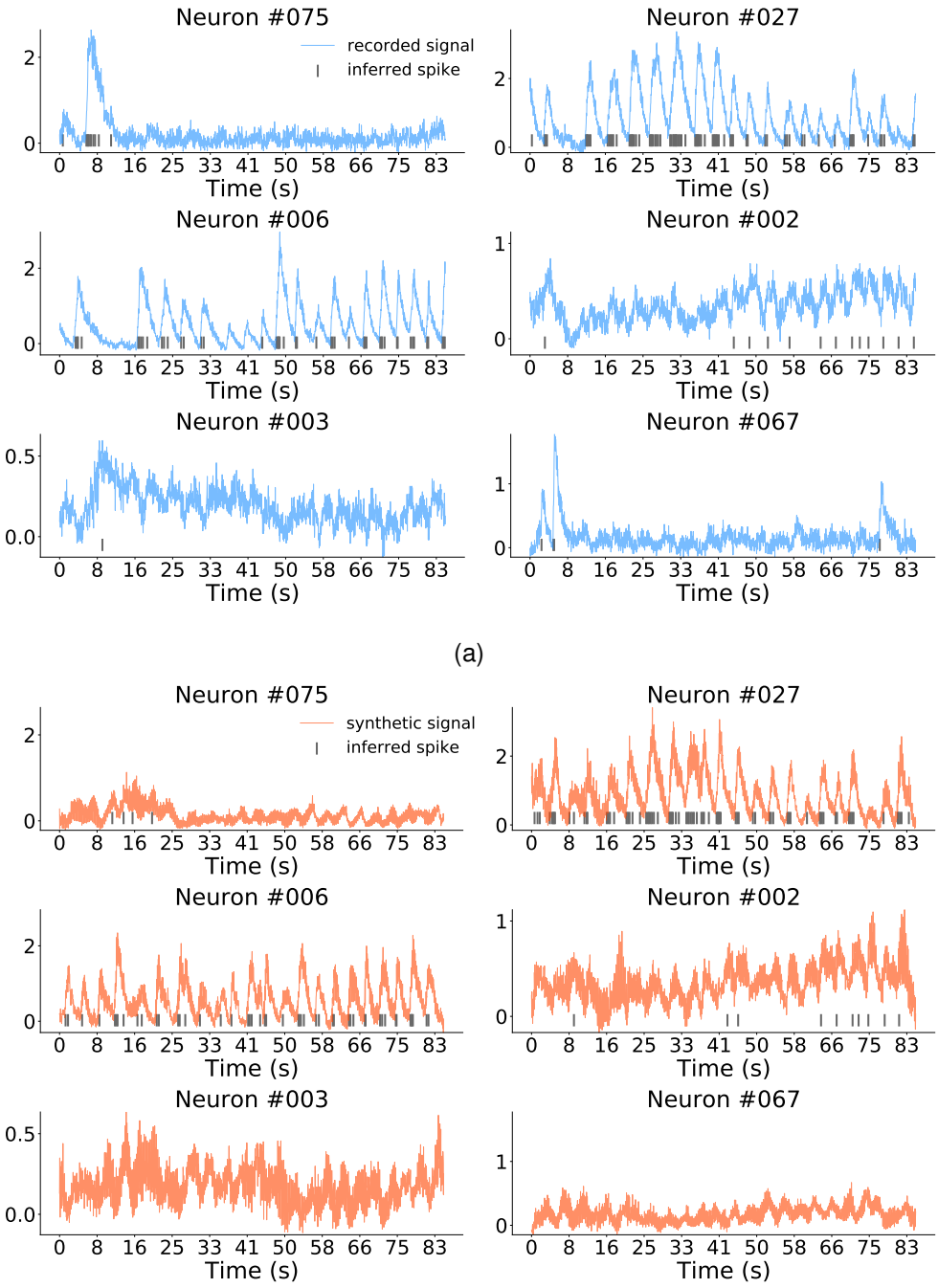


Figure 4.4: Calcium signals and inferred spike trains (in gray) of randomly selected neurons. (a) shows the recorded data (in blue) and (b) shows synthetic data (in orange) generated by CalciumGAN trained on recorded data. Note that the generated data should not be identical with the recorded data, because CalciumGAN should not replicate the signals and it could generate a sample corresponding to a different trial.

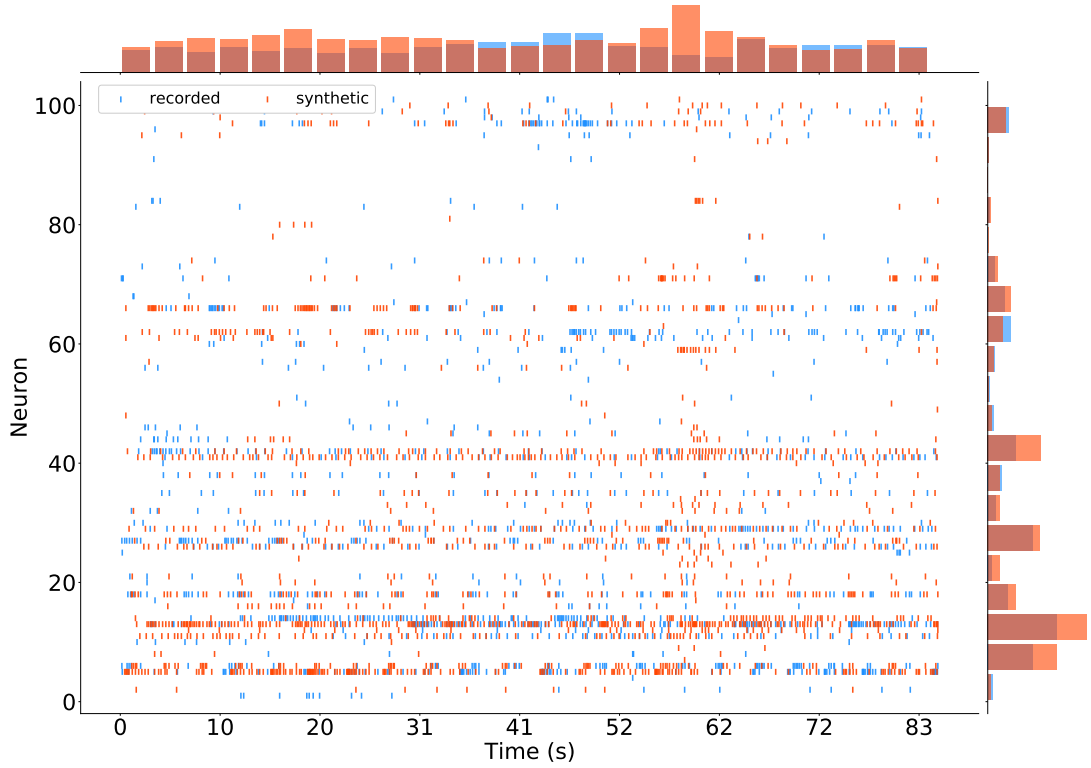
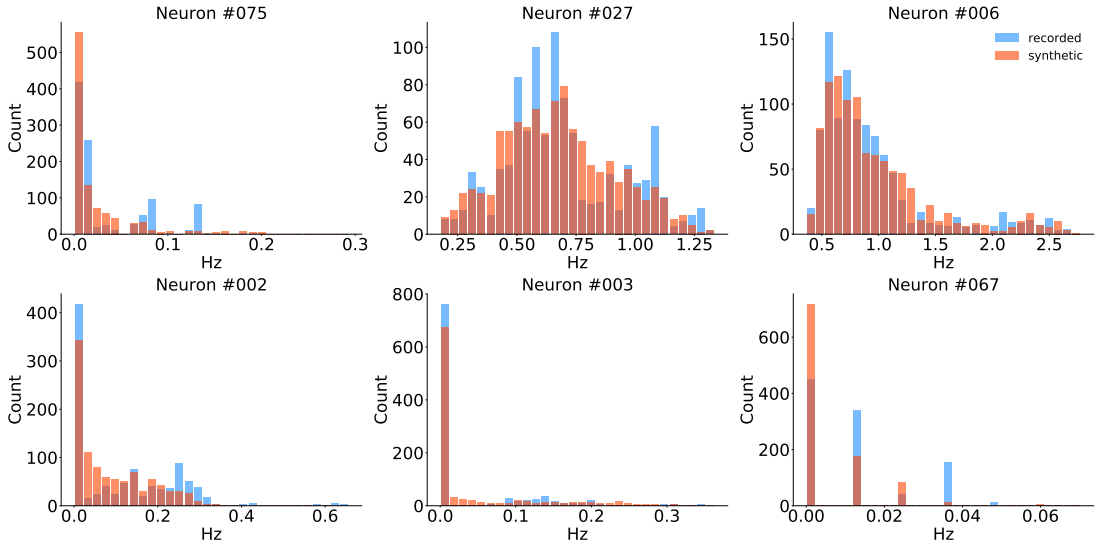
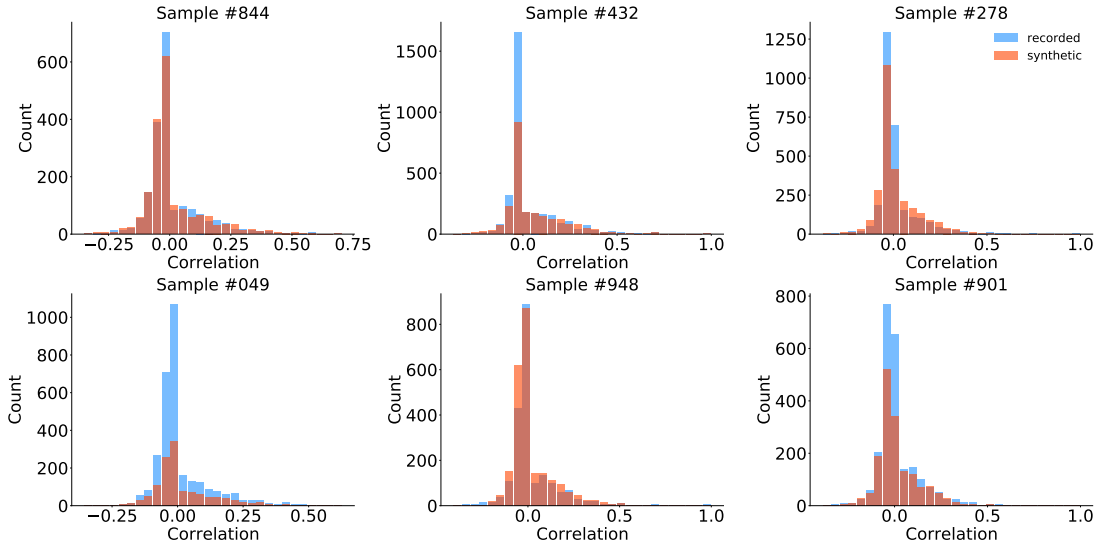


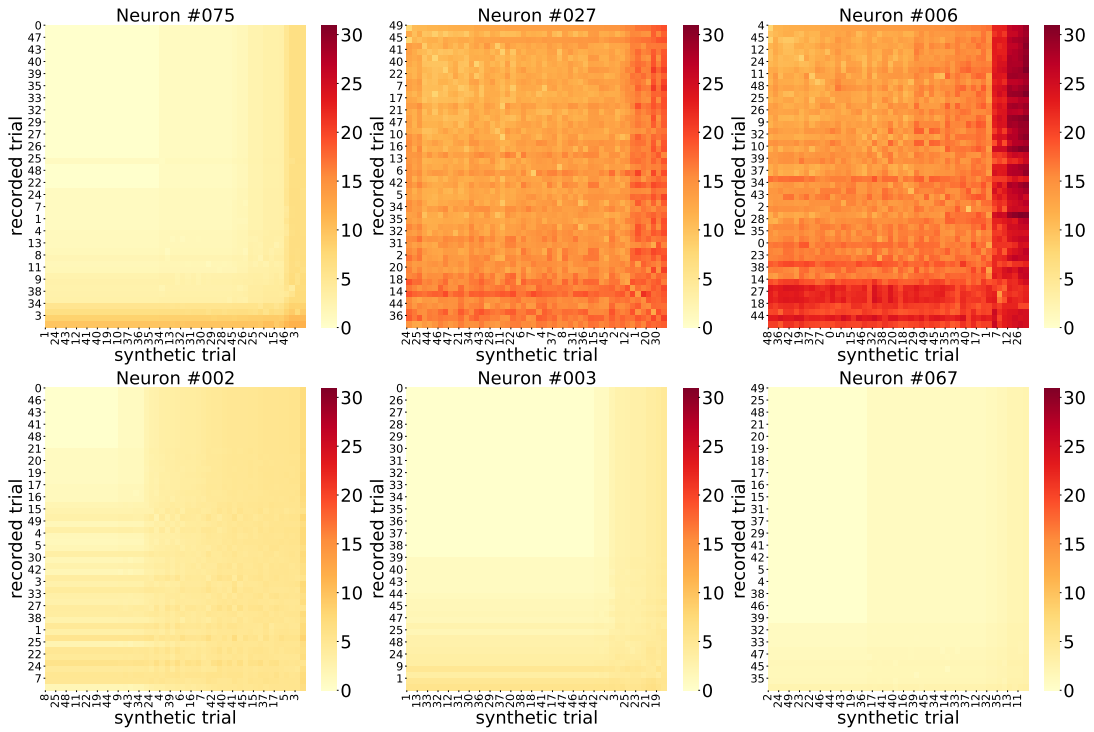
Figure 4.5: Raster plot of inferred real and synthetic spike trains of a randomly selected sample generated by CalciumGAN trained on recorded data. Blue markers indicate recorded data and orange markers indicate generated data. The histograms on the x and y axis indicate the number of spikes over the temporal dimension and neuron population respectively.


Neuron #075
Neuron #027
Neuron #006
recorded
synthetic
Neuron #002
Neuron #003
Neuron #067
Count
Hz

(a)

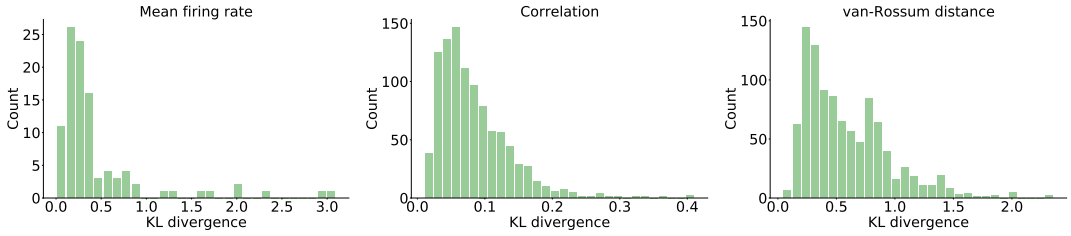

Sample #844
Sample #432
Sample #278
recorded
synthetic
Sample #049
Sample #948
Sample #901
Count
Correlation

(b)



(c)

Figure 4.6: First and second-order statistics of data generated from CalciumGAN trained on the recorded data. Shown neurons and samples were randomly selected. (a) Mean firing rate distribution over 1000 samples per neuron. (b) Pearson correlation coefficient distribution. (c) van-Rossum distance between recorded and generated spike trains over 50 samples. Heatmaps were sorted where the pair with the smallest distance value was placed at the top left corner, followed by the pair with the second smallest distance at the second-row second column, and so on.



(a) (b) (c)

Figure 4.7: KL divergence of recorded data and generated data distributions. (a) Mean firing rate of each neuron over 1000 trials. (b) Pairwise Pearson correlation coefficient over 1000 trials. (c) Pairwise spike train van-Rossum distance over 1000 trials. The mean KL divergence of each statistics are 0.45, 0.08 and 0.58 respectively.

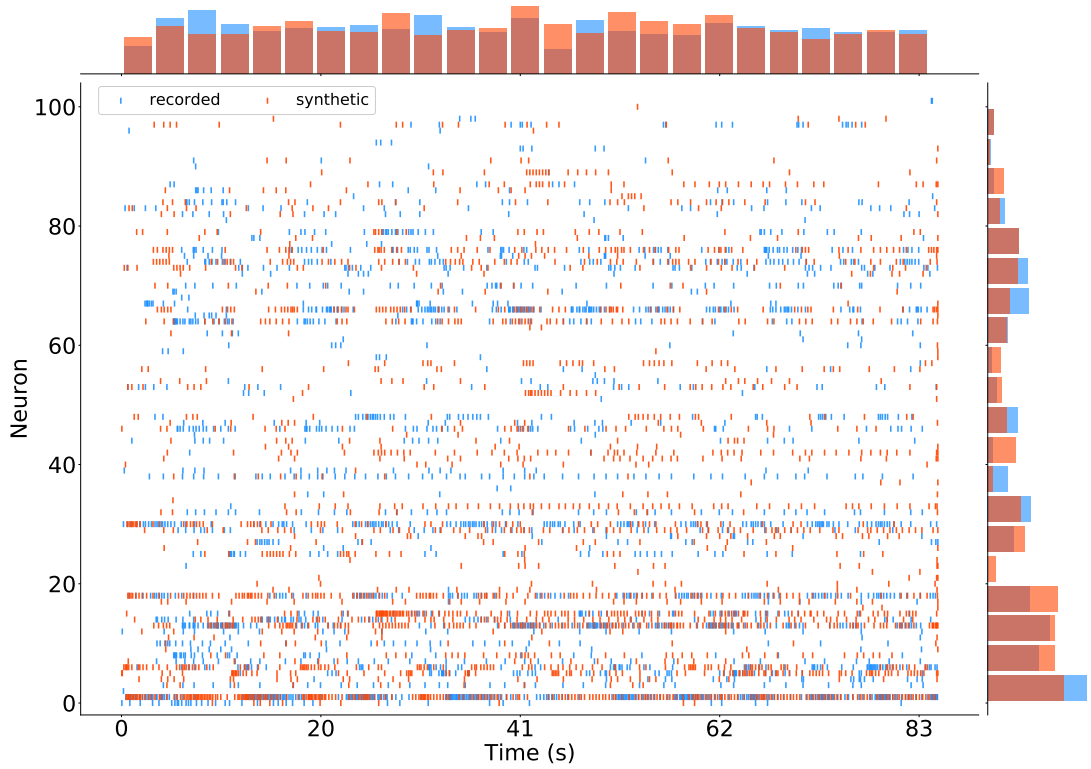
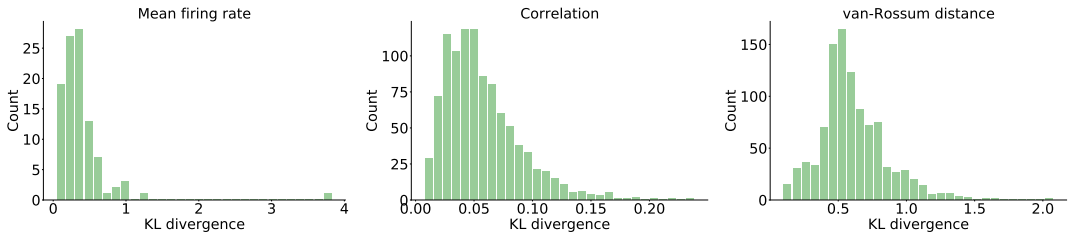


Figure 4.10: Raster plot of inferred real and synthetic spike trains of a randomly selected sample generated by CalciumGAN trained on calcium imaging data recorded on the 1st day of the animal experiment. Blue markers indicate recorded data and orange markers indicate generated data. The histograms on the x and y axis indicate the number of spikes over the temporal dimension and neuron population respectively. Compared to recordings acquired on the 4th day of the experiment, most neurons recorded in the untrained animal are more active.



(a) (b) (c)

Figure 4.11: KL divergence of calcium data recorded on day 1 of the animal experiment and generated data distributions. (a) Mean firing rate of each neuron over 1000 trials. (b) Pairwise Pearson correlation coefficient over 1000 trials. (c) Pairwise spike train van-Rossum distance over 1000 trials. The mean KL divergence of each statistics are 0.38, 0.06 and 0.60 respectively.

Chapter 5

Discussion

We demonstrated that the GAN framework is capable of synthesizing realistic fluorescent calcium indicator signals similar to those imaged in the somata of neuronal populations of behaving animals. To achieve this, we adapted the WaveGAN architecture with the Wasserstein distance training objective. We generated artificial neuronal activities using a dichotomized Gaussian model, showing that CalciumGAN is able to learn the underlying distribution of the data. We then fitted our model to imaging data from the primary visual cortex of behaving mice. Moreover, we showed that the Phase Shuffle mechanism in both 1-dimensional and 2-dimensional settings can improve the quality of the generated data, and subsequently the inferred spike train statistics. We also examined the use of 2-dimensional convolution to model the spatio-temporal information as well as representing the signals in their frequency-domain, though it did not yield significant improvement and greatly increased the computational cost of the model. Nevertheless, we demonstrated that the GAN framework can model calcium imaging data without the need of incorporating any information of the neuronal activities into the objective function, unlike many existing statistical generative models.

We believe that this work is beneficial to both the computational neuroscience community, as well as the machine learning community. Despite the recent advancement and popularity of calcium imaging of neuronal activity *in vivo*, the number of trials and the duration of imaging sessions in animal experiments is limited due to ethical and practical considerations. Since a GAN based model can generate unlimited amounts of data, one could potentially reduce the number of trials and their duration in animal experiments, in accordance with the 3Rs principles of ethical animal research.

From the technical aspect, we utilized many of the recent advancements in the field of deep learning and computing. For instance, we incorporated mixed-precision training which took advantage of the hardware acceleration for half-precision compute in NVIDIA Turing GPUs [33], which significantly improved our training speed with minimal loss in accuracy.

5.1 Limitations

We would like to highlight one potential bias in this work. To infer spike trains from the real and synthetic calcium traces, we used the OASIS deconvolution algorithm [15], a method which has great real-time deconvolution performance, as well as an existing Python implementation of the algorithm by the authors [14]. Speed was a crucial characteristic for evaluating a large number of trials. Nonetheless, we found that this advantage often came at the cost of performance in the form of clearly missed spikes (c.f. Figure 4.4). However, we stress that these shortcomings apply to both the real data and the synthetic data in exactly the same way. In the end, we use the inferred spikes as a way to validate the plausibility of the synthesized traces. The comparison is fair as long as real and synthetic deconvolutions are subject to the same biases.

In addition, we didn't observe significant improvement with the CalciumGAN-2D variant in our experiment on real recorded data. One possible issue with our approach was that, as the neurons in recorded data were not sorted based on their physical location *in vivo*, modelling the spatial information of unordered data would not be helpful to the network. Using a larger kernel in the spatial dimension to cover more neurons could be one potential solution, though that would increase the number of parameters and make the comparison unfair. Moreover, 2-dimensional convolution increased the computational cost significantly in our experiment.

5.2 Future Work

As the work in deep generative models continues to develop and expand, there is a limitless number of possibilities to explore at the intersection of the GAN framework and neural coding. One potential future direction for this work is to provide a meaningful interpretation for the latent generator representation z. In many image generation tasks with GAN [5, 26], it has been shown that the output image can be modified or targeted

by interpolating the latent variable that is fed to the generator. Similarly, one could potentially have final control of the generated calcium signals by exploring the synthetic calcium signals generated after interpolating samples in the latent space. Thereby, one could generate calcium imaging data that resemble the neuronal activities of an animal performing a particular novel task.

Another interesting research direction would be performing style transfer or unpaired translation on neuronal activities. GANs have shown compelling results in style transfer, where the generator would receive data input (instead of noise) and convert the input to a specific class or artistic style. For instance, Yi et al. [62] introduced a framework that uses two GANs to learn the transformation of images taken in bright daylight and dark at night, all without paired samples. Huang et al. [24] demonstrated that GANs can convert timbre sound samples from one instrument to another while preserving the musical content. Similarly, we can use a GAN (or multiple GANs) to learn the relationship between different neuron populations or to reveal changes in the activity of the same neuronal population in different training phases of an animal learning a behavioural task. This could be achieved by using, for instance, CycleGAN [64], an unsupervised learning model that can learn the mapping between two distributions without paired data, as a potential model architecture.

As a concluding remark, this work has demonstrated GAN's excellent capability in modelling calcium imaging data of neuronal activities. In addition, we provided an effective workflow and optimization on how to work with neuronal activity recordings in a deep learning setting. The ability to produce realistic artificial neuronal activity patterns and datasets will facilitate the study of the distinct dynamics of neuronal activity, which is essential to the understanding of the encoding strategies utilized by neurons. As deep generative models, especially GANs, are becoming more mature, and calcium imaging is widely used in the neuroscience community, the incorporation of the two technologies suggests a promising direction into the understanding of neural information processing.

Appendix A

Appendix

A.1 CalciumGAN without Phase Shuffle

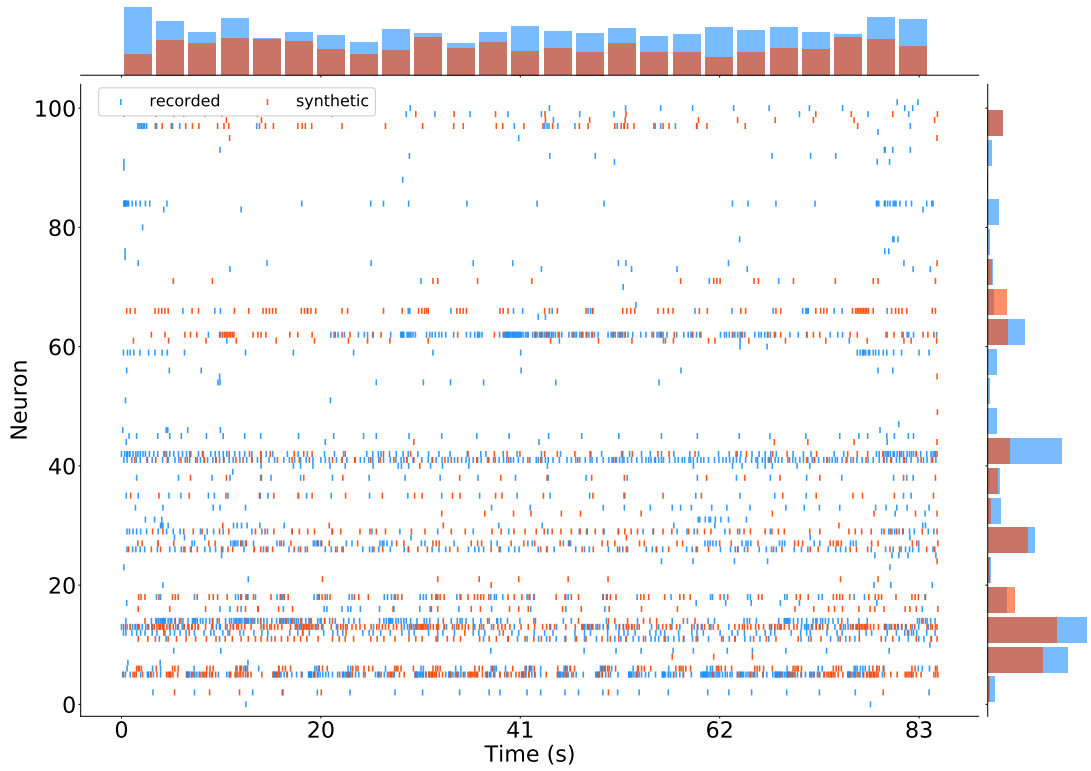


Figure A.1: Raster plot of inferred real and synthetic spike trains of a randomly selected sample generated by CalciumGAN without Phase Shuffle. Blue markers indicate recorded data and orange markers indicate generated data. The histograms on the x and y axis indicate number of spikes over the temporal dimension and neuron population respectively.

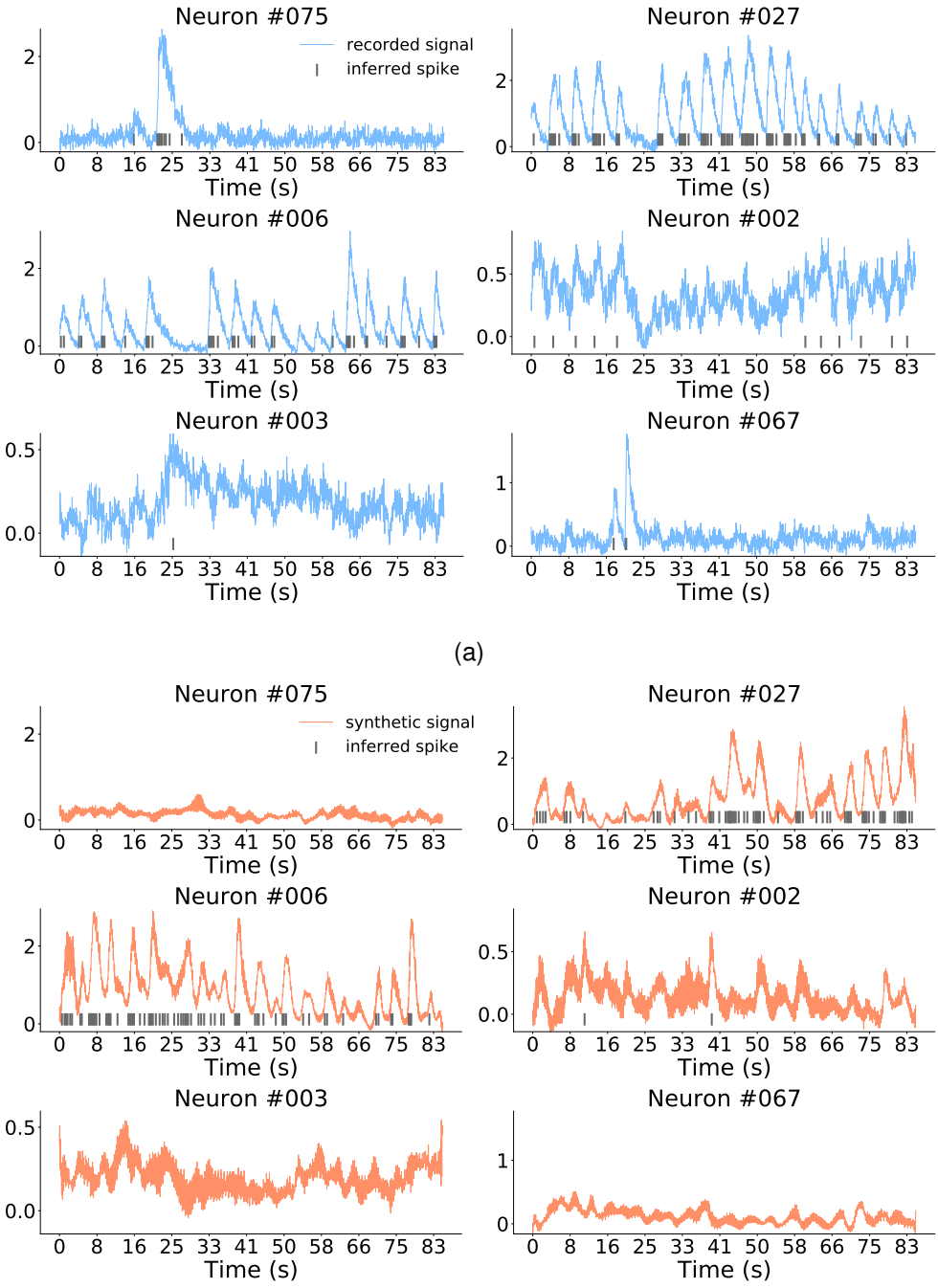
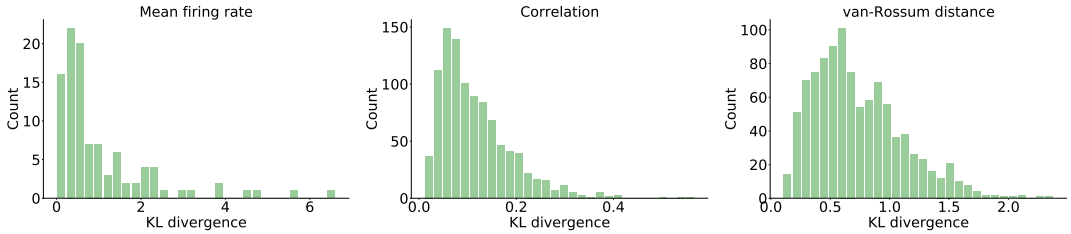
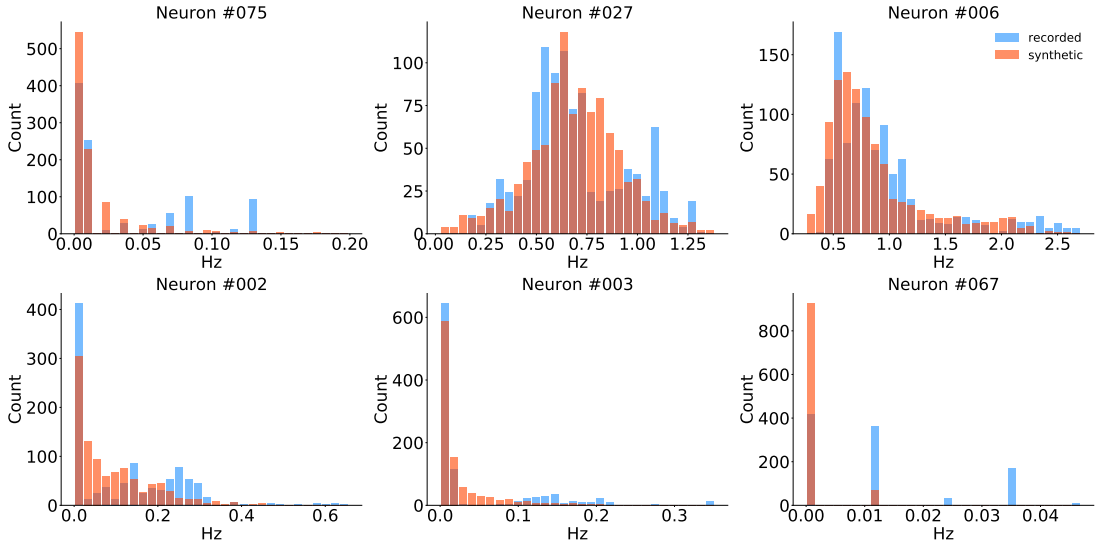


Figure A.2: Calcium signals and inferred spike trains (in gray) of randomly selected neurons generated by CalciumGAN without Phase Shuffle. (a) shows the recorded data (in blue) and (b) shows synthetic data (in orange). Notice the typical calcium signal characteristics of the sharp rise to peak followed by a tail of decaying signal are less observable in the generated data.

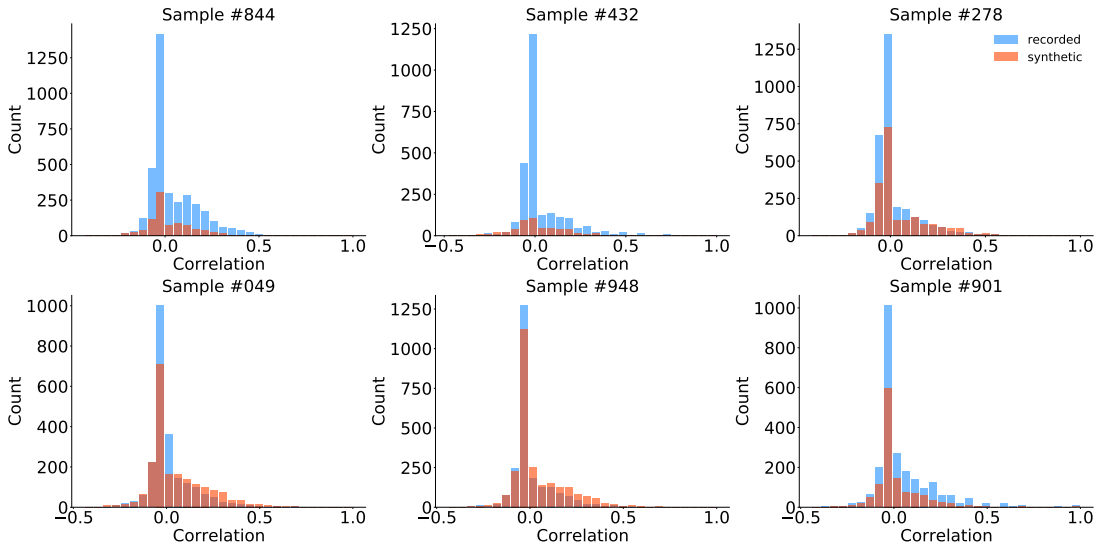


(a) (b) (c)

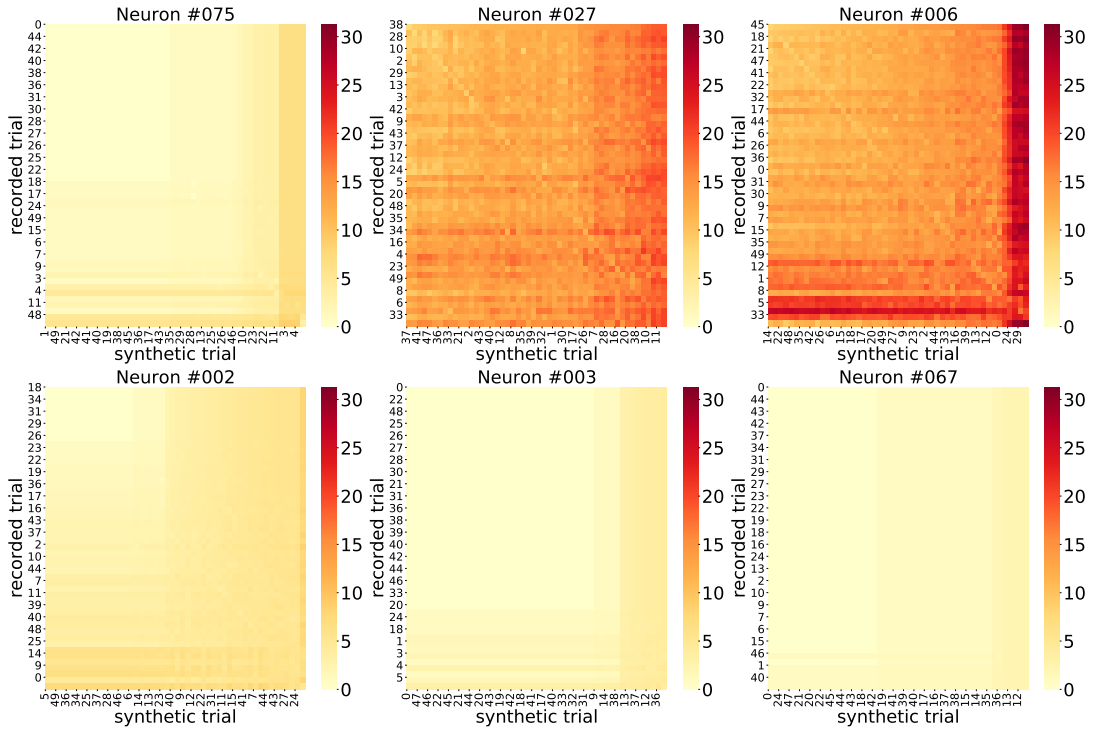
Figure A.3: KL divergence of recorded data and generated data distributions. The mean KL divergence of each statistics are 1.05, 0.11 and 0.73 respectively.



(a)



(b)



(c)

Figure A.4: First and second order statistics of data generated from CalciumGAN without Phase Shuffle. Shown neurons and samples were randomly selected. (a) Mean firing rate distribution over 1000 samples per neuron. (b) Pearson correlation coefficient distribution. (c) van-Rossum distance between recorded and generated spike trains over 50 samples.

A.2 CalciumGAN-2D

Hyper-parameters	Value
Filters	20
Kernel size	(8, 8)
Stride	(2, 1)
Noise dimension	32
Critic update	5
Gradient Penalty (λ)	10
Batch size (bs)	128
Epochs	400
Learning rate	0.0001
Phase shift (m)	10
Phase shift (n)	10

Table A.1: CalciumGAN-2D hyperparamters

Layer	Output shape
Input	(bs, 32)
Dense	(bs, 104448)
LeakyRelu	(bs, 104448)
Reshape	(bs, 64, 51, 32)
Conv2DTranspose	(bs, 128, 51, 100)
LayerNorm	(bs, 128, 51, 100)
LeakyRelu	(bs, 128, 51, 100)
Conv2DTranspose	(bs, 256, 51, 60)
LayerNorm	(bs, 256, 51, 60)
LeakyRelu	(bs, 256, 51, 60)
Conv2DTranspose	(bs, 512, 102, 40)
LayerNorm	(bs, 512, 102, 40)
LeakyRelu	(bs, 512, 102, 40)
Conv2DTranspose	(bs, 1024, 102, 20)
LayerNorm	(bs, 1024, 102, 20)
LeakyRelu	(bs, 1024, 102, 20)
Conv2DTranspose	(bs, 2048, 102, 1)
LayerNorm	(bs, 2048, 102, 1)
LeakyRelu	(bs, 2048, 102, 1)
Dense	(bs, 2048, 102, 1)
Sigmoid	(bs, 2048, 102, 1)

(a) Generator architecture

Layer	Output shape
Input	(bs, 2048, 102, 1)
Conv2D	(bs, 512, 102, 20)
LeakyRelu	(bs, 512, 102, 20)
PhaseShuffle	(bs, 512, 102, 20)
Conv2D	(bs, 128, 102, 40)
LeakyRelu	(bs, 128, 102, 40)
PhaseShuffle	(bs, 128, 102, 40)
Conv2D	(bs, 32, 102, 60)
LeakyRelu	(bs, 32, 102, 60)
PhaseShuffle	(bs, 32, 102, 60)
Conv2D	(bs, 8, 102, 80)
LeakyRelu	(bs, 8, 102, 80)
PhaseShuffle	(bs, 8, 102, 80)
Conv2D	(bs, 2, 102, 100)
LeakyRelu	(bs, 2, 102, 100)
Flatten	(bs, 20400)
Dense	(bs, 1)

(b) Discriminator architecture

Table A.2: The generator (a) and discriminator (b) architecture of CalciumGAN-2D. The main differences between CalciumGAN and its 2D variant is the use of 2D transposed convolution layer in the generator, and 2D convolution and Phase Shift layer in the discriminator. The generator consists of 4,242,329 parameters, and the discriminator consists of 4,121,821 parameters. Note bs denotes batch size.

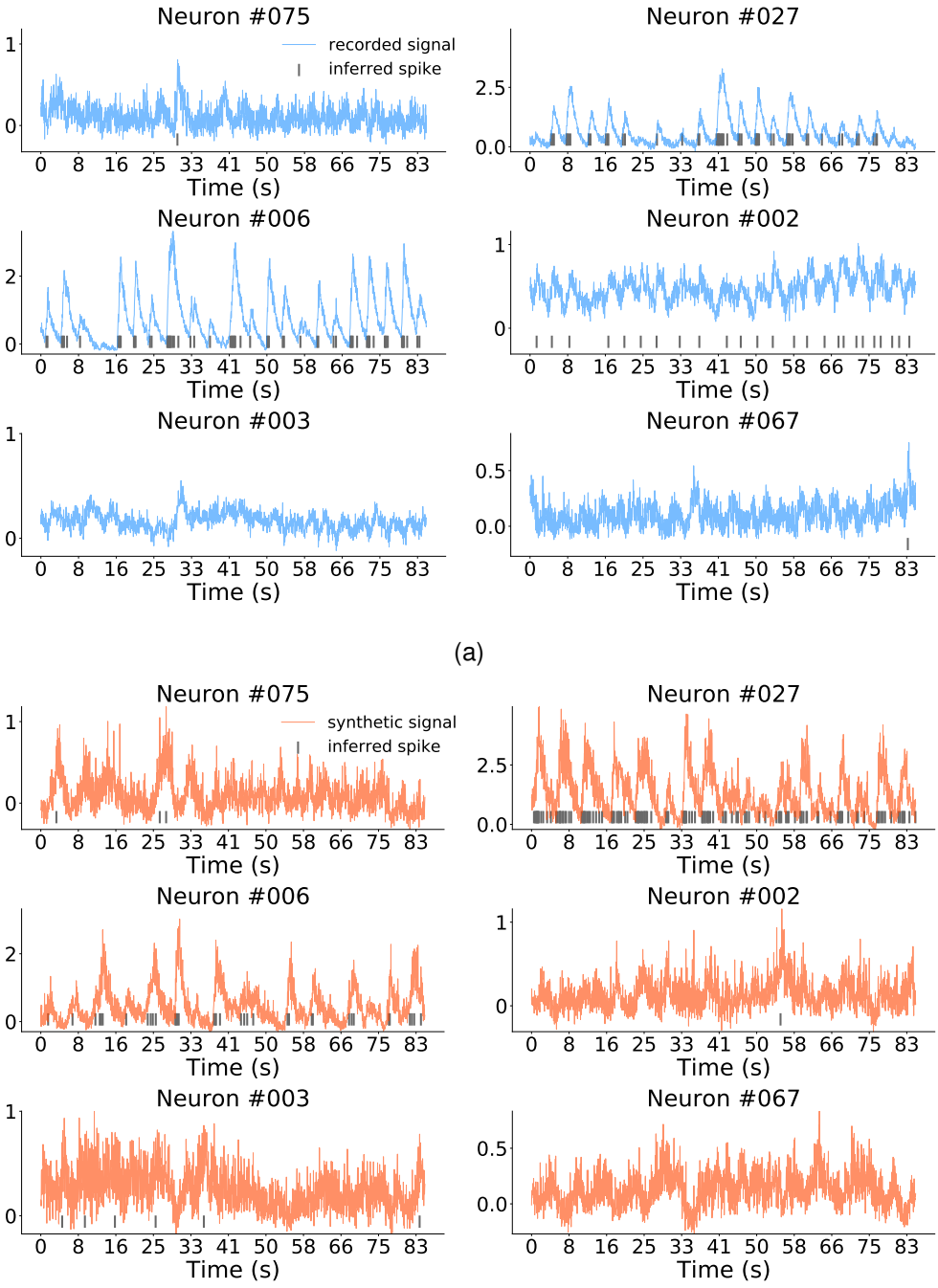


Figure A.5: Calcium signals and inferred spike trains (in gray) of randomly selected neurons. (a) shows the recorded data (in blue) and (b) shows synthetic data (in orange) generated by CalciumGAN-2D trained on recorded data.

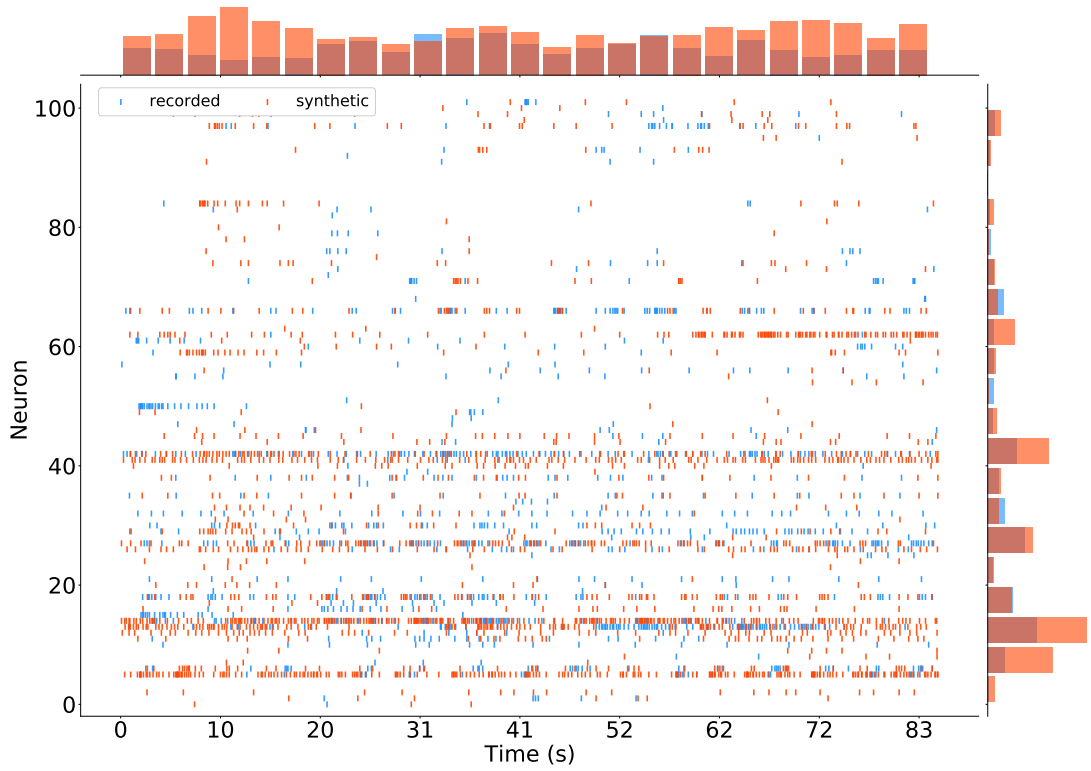
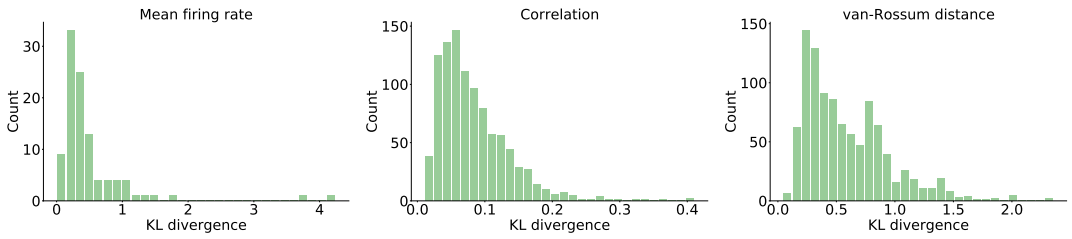
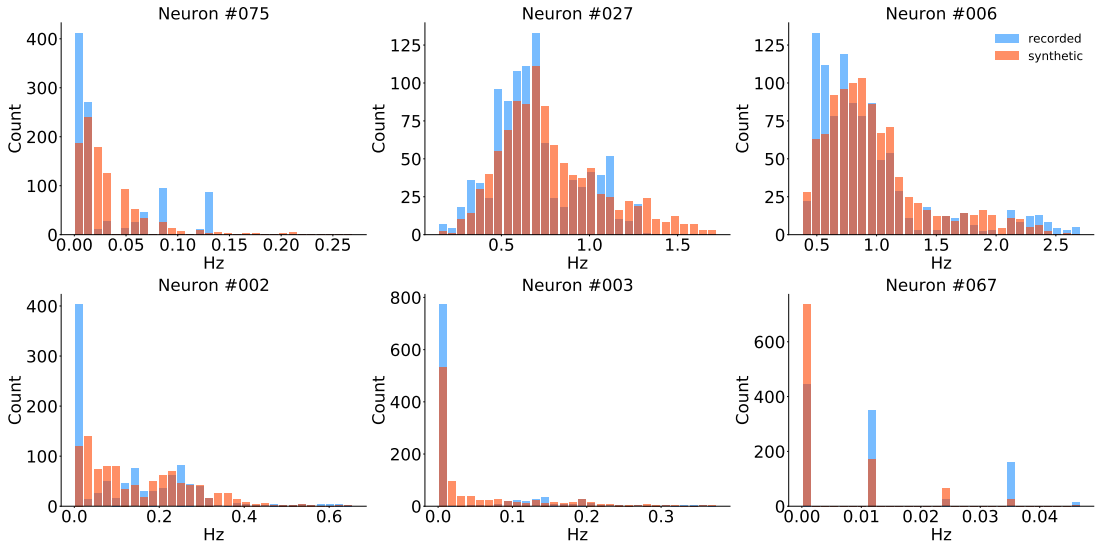


Figure A.6: Raster plot of inferred real and synthetic spike trains of a randomly selected sample generated by CalciumGAN-2D trained on recorded data. Blue markers indicate recorded data and orange markers indicate generated data. The histograms on the x and y axis indicate number of spikes over the temporal dimension and neuron population respectively.

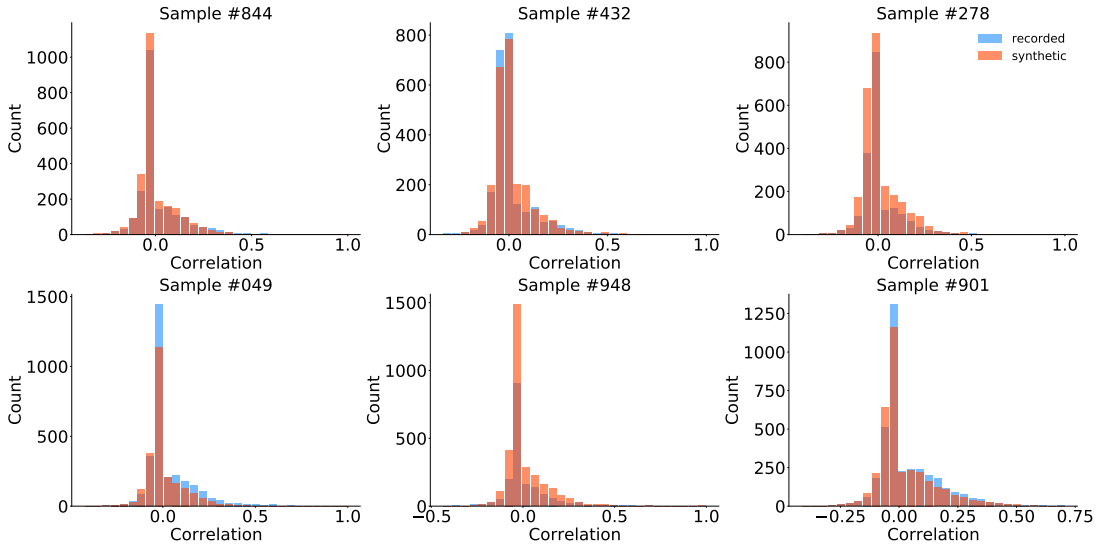


(a) (b) (c)

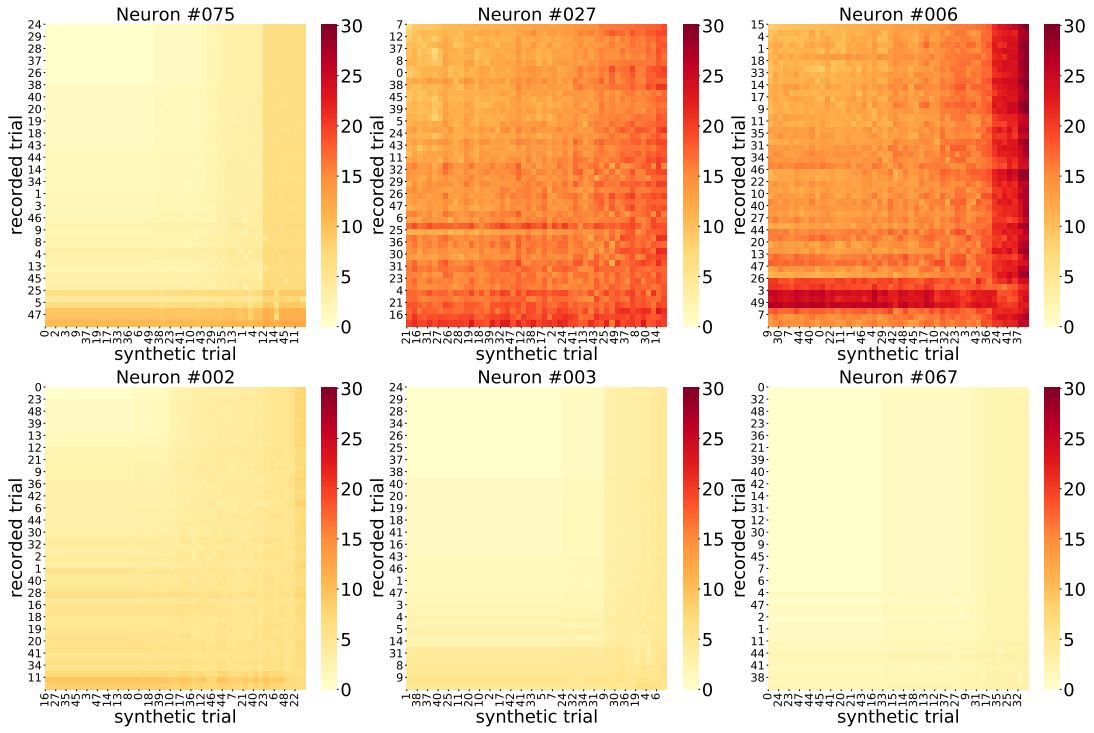
Figure A.7: KL divergence of recorded data and generated data distributions. The mean KL divergence of each statistics are 0.54, 0.12 and 0.59 respectively.


Neuron #075
Neuron #027
Neuron #006
recorded
synthetic
Neuron #002
Neuron #003
Neuron #067
Count
Hz

(a)


Sample #844
Sample #432
Sample #278
recorded
synthetic
Sample #049
Sample #948
Sample #901
Count
Correlation

(b)



(c)

Figure A.8: First and second order statistics of data generated from CalciumGAN trained on the recorded data. Shown neurons and samples were randomly selected. (a) Mean firing rate distribution over 1000 samples per neuron. (b) Pearson correlation coefficient distribution. (c) van-Rossum distance between recorded and generated spike trains over 50 samples.

A.3 CalciumGAN-2D FFT

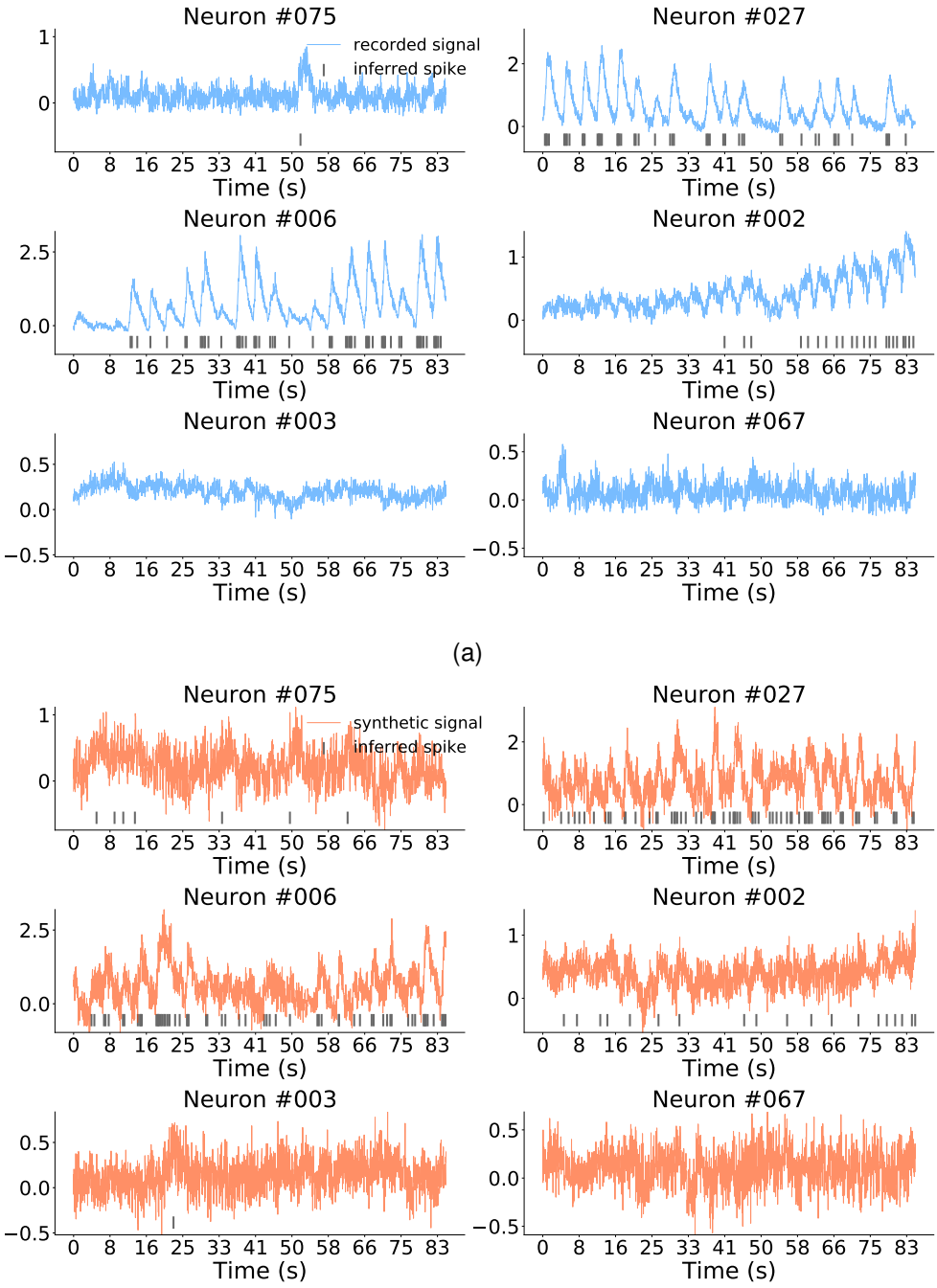


Figure A.9: Calcium signals and inferred spike trains (in gray) of randomly selected neurons generated by CalciumGAN-2D FFT. (a) shows the recorded data (in blue) and (b) shows synthetic data (in orange).

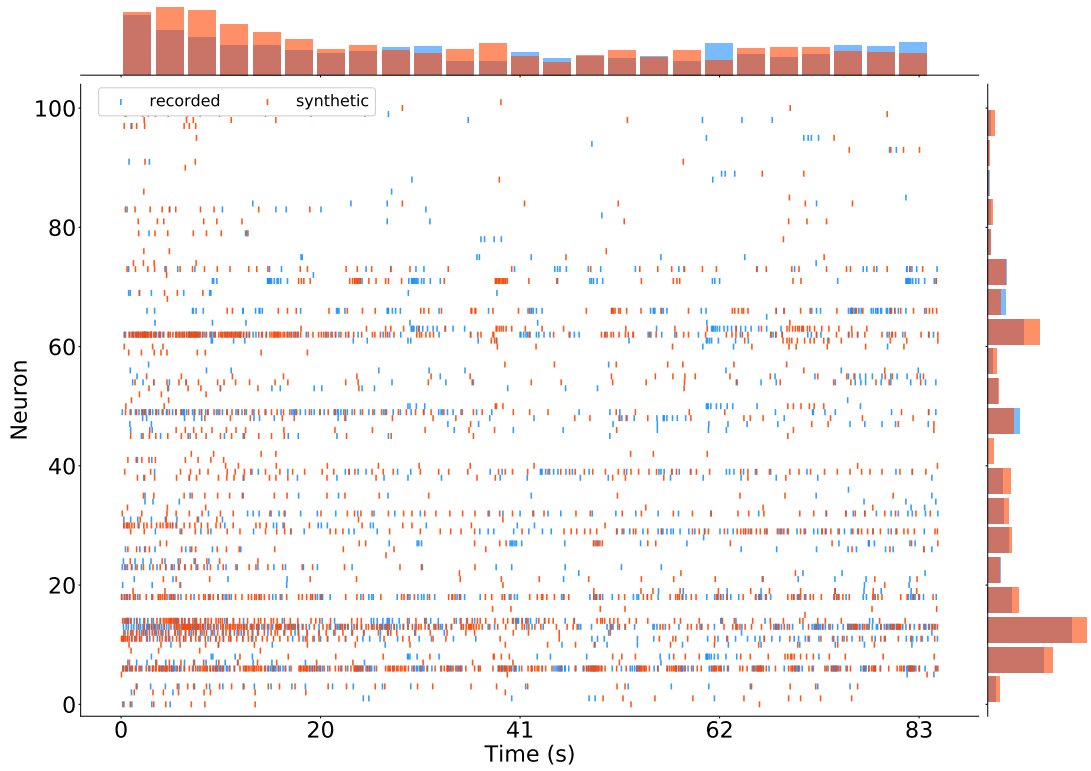


Figure A.10: Raster plot of inferred real and synthetic spike trains of a randomly selected sample generated by CalciumGAN-2D FFT. Blue markers indicate recorded data and orange markers indicate generated data. The histograms on the x and y axis indicate number of spikes over the temporal dimension and neuron population respectively.

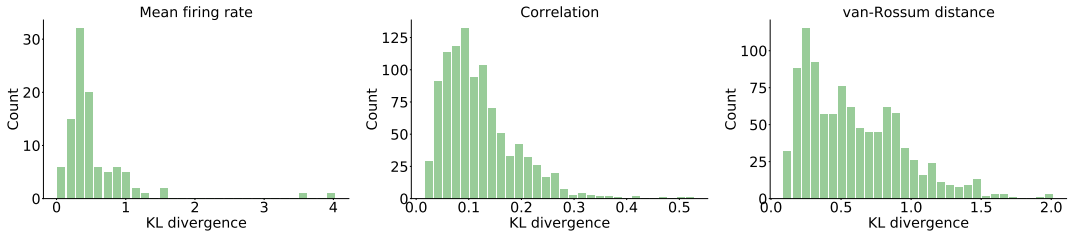
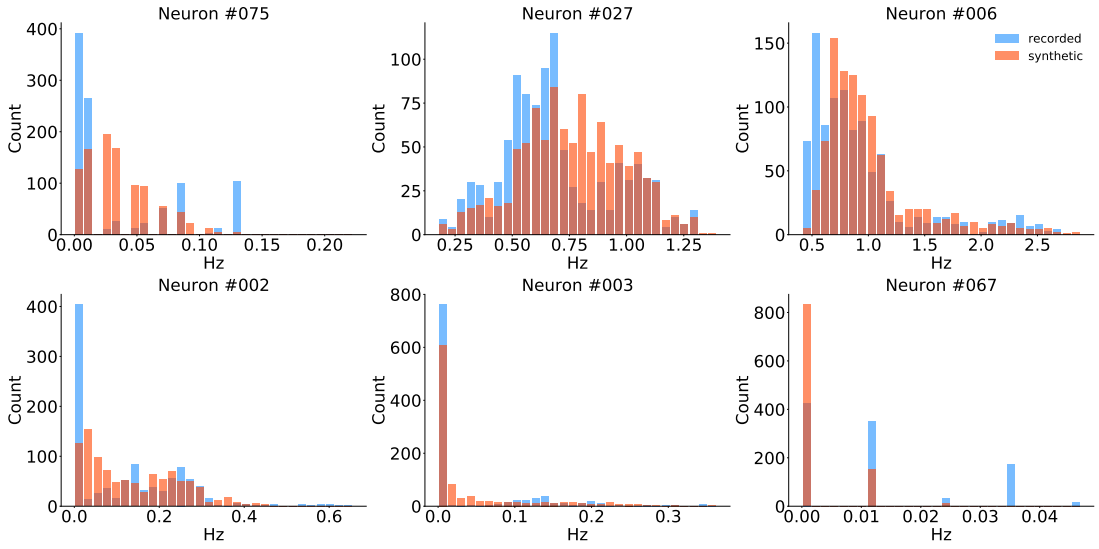
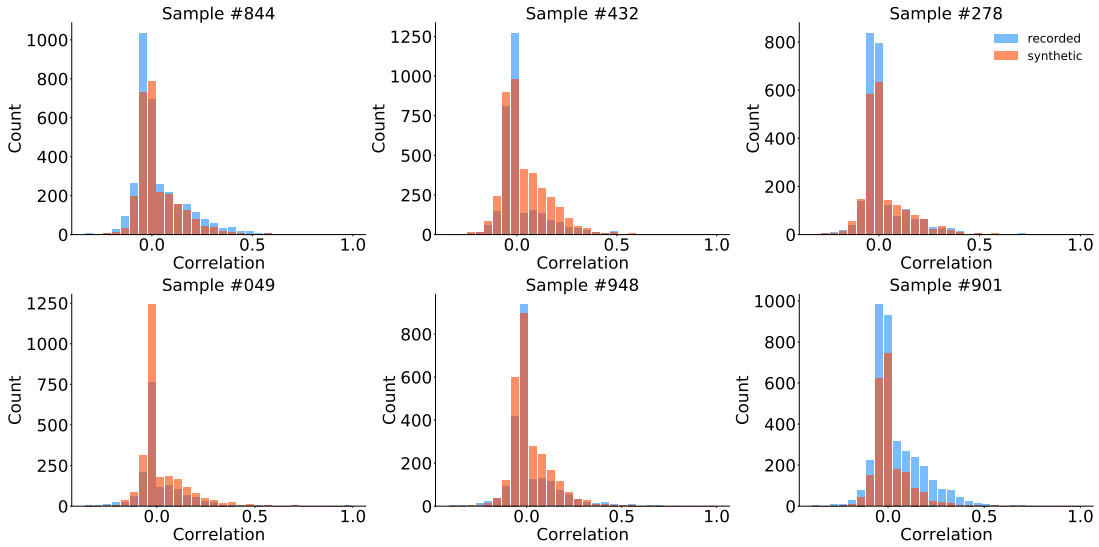


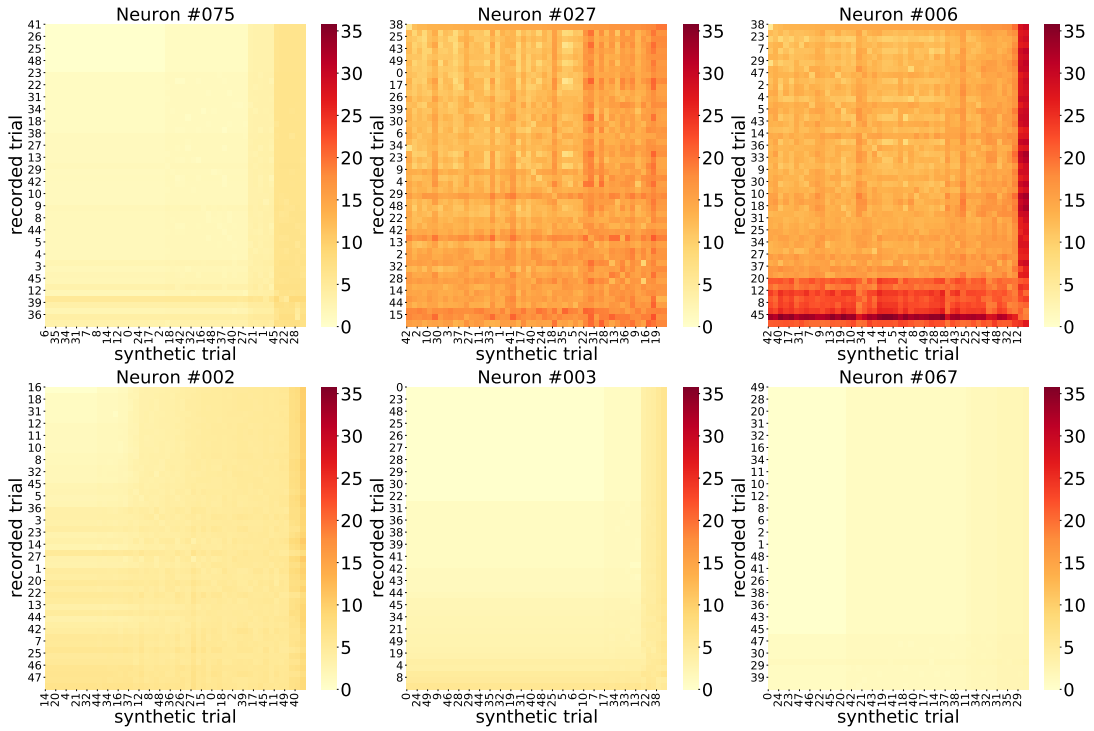
Figure A.11: KL divergence of recorded data and generated data distributions. The mean KL divergence of each statistics are 0.50, 0.11 and 0.58 respectively.


Neuron #075
Neuron #027
Neuron #006
recorded
synthetic
Neuron #002
Neuron #003
Neuron #067
Count
Hz

(a)


Sample #844
Sample #432
Sample #278
recorded
synthetic
Sample #049
Sample #948
Sample #901
Count
Correlation

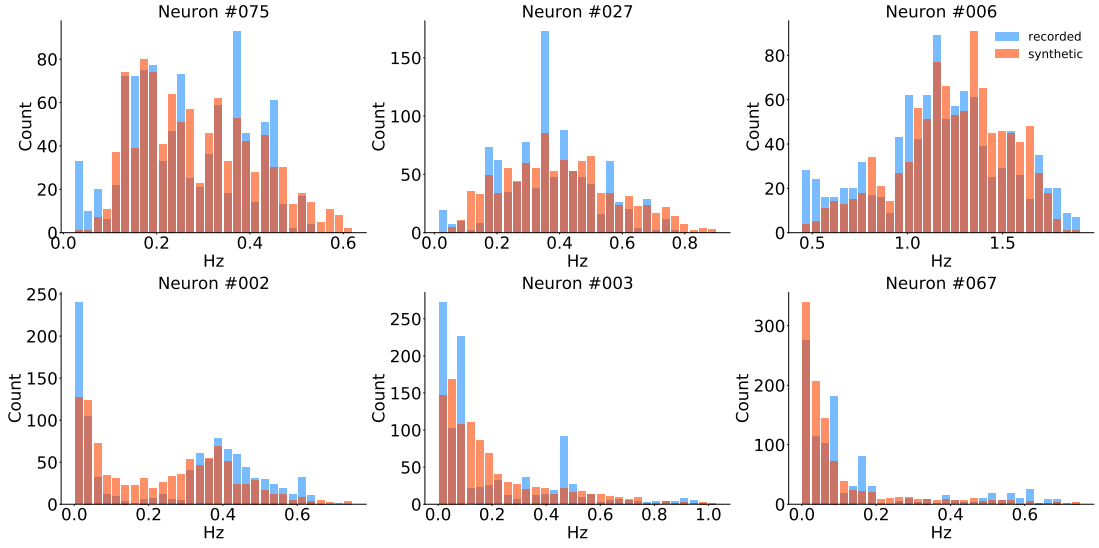
(b)



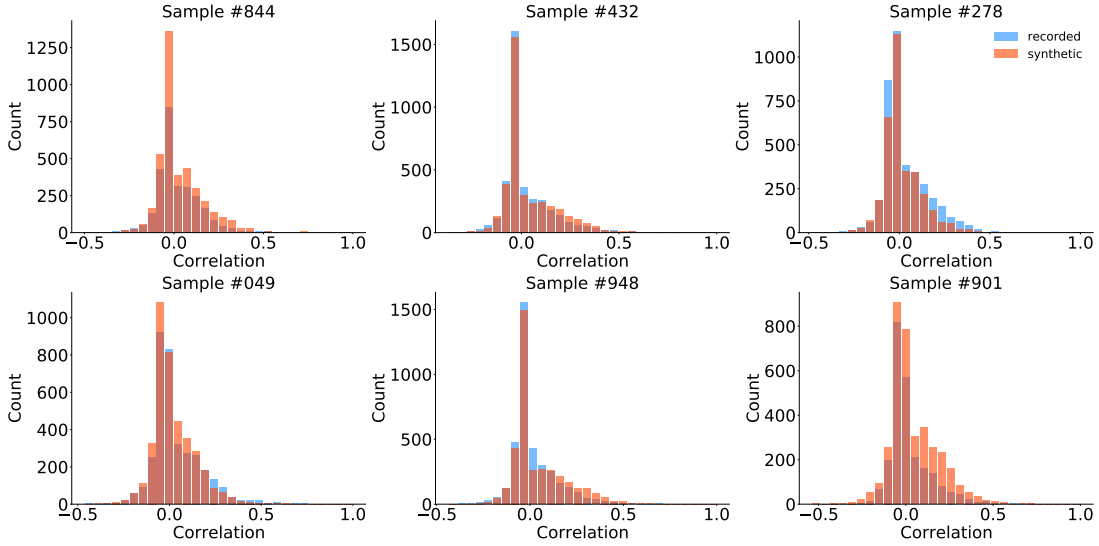
(c)

Figure A.12: First and second order statistics of data generated from CalciumGAN-2D trained on data in frequecy-domain. Shown neurons and samples were randomly selected. (a) Mean firing rate distribution over 1000 samples per neuron. (b) Pearson correlation coefficient distribution. (c) van-Rossum distance between recorded and generated spike trains over 50 samples.

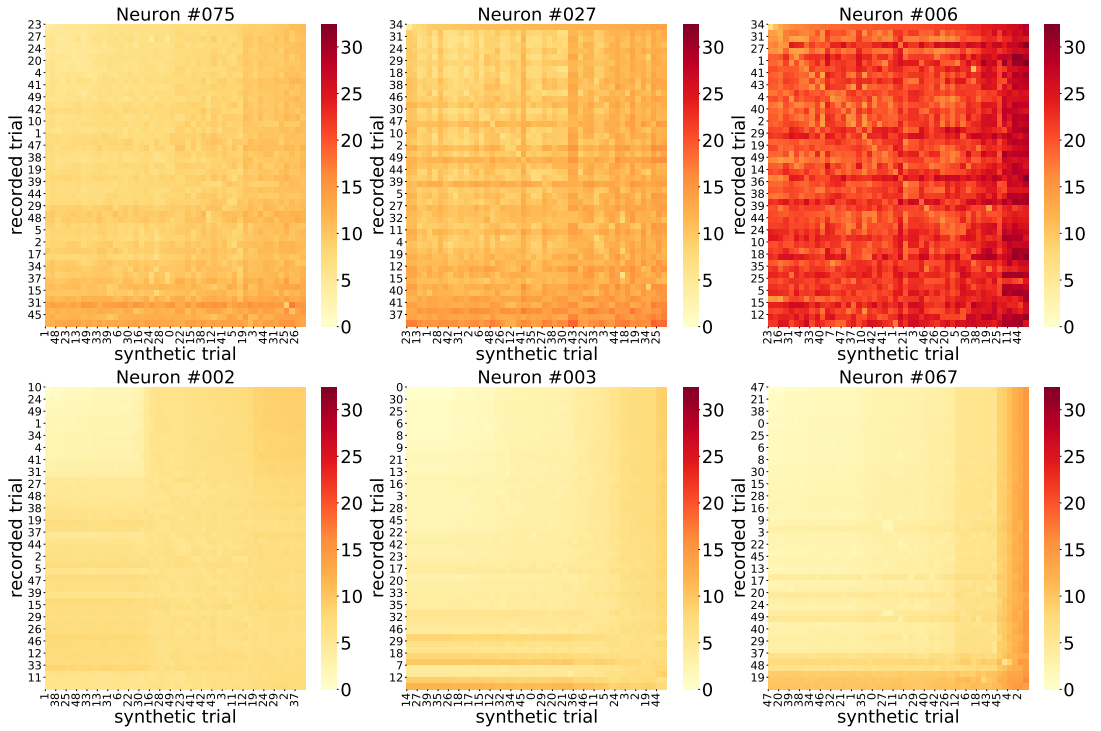
A.4 Untrained mice data



(a)



(b)



(c)

Figure A.13: First and second order statistics of data generated from CalciumGAN trained on calcium imaging data recorded on day one of the animal experiment. Shown neurons and samples were randomly selected. (a) Mean firing rate distribution over 1000 samples per neuron. (b) Pearson correlation coefficient distribution. (c) van-Rossum distance between recorded and generated spike trains over 50 samples. Heatmaps were sorted where the pair with the smallest distance value was placed at the top left corner, followed by the pair with the second smallest distance at the second row second column, and so on.

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

[3] Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305.

[4] 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.

[5] Bojanowski, P., Joulin, A., Lopez-Paz, D., and Szlam, A. (2017). Optimizing the latent space of generative networks. *arXiv preprint arXiv:1707.05776*.

[6] Brown, E. N., Kass, R. E., and Mitra, P. P. (2004). Multiple neural spike train data analysis: state-of-the-art and future challenges. *Nature neuroscience*, 7(5):456–461.

[7] Caccia, M., Caccia, L., Fedus, W., Larochelle, H., Pineau, J., and Charlin, L. (2018). Language gans falling short. *arXiv preprint arXiv:1811.02549*.

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

[9] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P.

(2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180.

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

[11] Donahue, C., McAuley, J., and Puckette, M. (2019). Adversarial audio synthesis. In *International Conference on Learning Representations*.

[12] Engel, J., Agrawal, K. K., Chen, S., Gulrajani, I., Donahue, C., and Roberts, A. (2019). Gansynth: Adversarial neural audio synthesis. *arXiv preprint arXiv:1902.08710*.

[13] Engel, J., Resnick, C., Roberts, A., Dieleman, S., Norouzi, M., Eck, D., and Simonyan, K. (2017). Neural audio synthesis of musical notes with wavenet autoencoders. In *International Conference on Machine Learning*, pages 1068–1077.

[14] Friedrich, J. (2017). Oasis. `https://github.com/j-friedrich/OASIS`.

[15] Friedrich, J., Zhou, P., and Paninski, L. (2017). Fast online deconvolution of calcium imaging data. *PLoS computational biology*, 13(3):e1005423.

[16] Fröhlich, F. (2016). Chapter 11 - optical measurements and perturbations. In Fröhlich, F., editor, *Network Neuroscience*, pages 145 – 159. Academic Press, San Diego.

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

[18] 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*.

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

[20] 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.

[21] 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.

[22] Hartmann, K. G., Schirrmeister, R. T., and Ball, T. (2018). Eeg-gan: Generative adversarial networks for electroencephalograhic (eeg) brain signals. *arXiv preprint arXiv:1806.01875*.

[23] 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*.

[24] Huang, S., Li, Q., Anil, C., Bao, X., Oore, S., and Grosse, R. B. (2018). Timbretron: A wavenet (cyclegan (cqt (audio))) pipeline for musical timbre transfer. *arXiv preprint arXiv:1811.09620*.

[25] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

[26] 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*.

[27] Kawaguchi, K., Kaelbling, L. P., and Bengio, Y. (2017). Generalization in deep learning. *arXiv preprint arXiv:1710.05468*.

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

[29] Lee, J., Park, J., Kim, K. L., and Nam, J. (2017). Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms. *arXiv preprint arXiv:1703.01789*.

[30] Lyamzin, D. R., Macke, J. H., and Lesica, N. A. (2010). Modeling population spike trains with specified time-varying spike rates, trial-to-trial variability, and pairwise signal and noise correlations. *Frontiers in computational neuroscience*, 4:144.

[31] 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.

[32] Maddison, C. J., Mnih, A., and Teh, Y. W. (2016). The concrete distribution: A continuous relaxation of discrete random variables.

[33] Micikevicius, P. (2017). Mixed-precision training of deep neural networks.

[34] 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*.

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

[36] 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*.

[37] Nasser, H., Marre, O., and Cessac, B. (2013). Spatio-temporal spike train analysis for large scale networks using the maximum entropy principle and monte carlo method. *Journal of Statistical Mechanics: Theory and Experiment*, 2013(03):P03006.

[38] NeuralEnsemble (2019). Elephant.

[39] Odena, A., Dumoulin, V., and Olah, C. (2016). Deconvolution and checkerboard artifacts. *Distill*.

[40] O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L., et al. (2019). Keras Tuner. `https://github.com/keras-team/keras-tuner`.

[41] Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

[42] 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.

[43] 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.

[44] 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.

[45] Peters, A. J., Chen, S. X., and Komiyama, T. (2014). Emergence of reproducible spatiotemporal activity during motor learning. *Nature*, 510(7504):263–267.

[46] Pillow, J. W., Shlens, J., Paninski, L., Sher, A., Litke, A. M., Chichilnisky, E., and Simoncelli, E. P. (2008). Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature*, 454(7207):995–999.

[47] Ramesh, P., Atayi, M., and Macke, J. H. (2019). Adversarial training of neural encoding models on population spike trains.

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

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

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

[51] Saxena, S. and Cunningham, J. P. (2019). Towards the neural population doctrine.

[52] 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.

[53] Staude, B., Rotter, S., and Grün, S. (2010). Cubic: cumulant based inference of higher-order correlations in massively parallel spike trains. *Journal of computational neuroscience*, 29(1-2):327–350.

[54] Stevenson, I. H. and Kording, K. P. (2011). How advances in neural recording affect data analysis. *Nature Neuroscience*, 14(2):139–142.

[55] 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.

[56] Stringer, C., Michaelos, M., and Pachitariu, M. (2019). High precision coding in visual cortex. *bioRxiv*.

[57] 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.

[58] Tkačik, G., Marre, O., Amodei, D., Schneidman, E., Bialek, W., and Berry II, M. J. (2014). Searching for collective behavior in a large network of sensory neurons. *PLoS Comput Biol*, 10(1):e1003408.

[59] Villani, C. (2008). *Optimal transport: old and new*, volume 338. Springer Science & Business Media.

[60] Wei, Z., Lin, B.-J., Chen, T.-W., Daie, K., Svoboda, K., and Druckmann, S. (2019). A comparison of neuronal population dynamics measured with calcium imaging and electrophysiology. *bioRxiv*, page 840686.

[61] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

[62] 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.

[63] Zhang, Y., Gan, Z., Fan, K., Chen, Z., Henao, R., Shen, D., and Carin, L. (2017). Adversarial feature matching for text generation. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 4006–4015. JMLR. org.

[64] 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.