THESIS

APPLICATION OF THE NEURAL DATA TRANSFORMER TO NON-AUTONOMOUS DYNAMICAL SYSTEMS

Submitted by

Domenick M. Mifsud

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer 2023

Master's Committee:

Advisor: Francisco R. Ortega
Co-Advisor: Charles Anderson

Michael Thomas
Armando Barreto

ABSTRACT

APPLICATION OF THE NEURAL DATA TRANSFORMER TO NON-AUTONOMOUS DYNAMICAL SYSTEMS

The Neural Data Transformer (NDT) is a novel non-recurrent neural network designed to model neural population activity, offering faster inference times and the potential to advance real-time applications in neuroscience. In this study, we expand the applicability of the NDT to non-autonomous dynamical systems by investigating its performance on modeling data from the Chaotic Recurrent Neural Network (RNN) with delta pulse inputs. Through adjustments to the NDT architecture, we demonstrate its capability to accurately capture non-autonomous neural population dynamics, making it suitable for a broader range of Brain-Computer Interface (BCI) control applications. Additionally, we introduce a modification to the model that enables the extraction of interpretable inferred inputs, further enhancing the utility of the NDT as a powerful and versatile tool for real-time BCI applications.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Francisco R. Ortega, and my co-advisor Charles Anderson for their support, encouragement, and mentorship. I also would like to thank my M.S. thesis committee members, Michael Thomas and Armando Barreto, for offering their time and advice.

DEDICATION

I would like to dedicate this thesis to my family and my dog Buddy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

A primary objective in neuroscience is to comprehend how the brain's diverse sensory, motor, and cognitive abilities emerge from neural population activity or the combined electrical signals generated by groups of neurons working together. Neural populations are essential in processing sensory inputs, generating motor outputs, and performing cognitive tasks such as decision-making, memory, and attention [1–5]. It has been theorized that the activity of these neural populations evolve over time according to a set of underlying dynamical principles [2, 6–9]. Yet, these dynamical systems are not fully autonomous as we consistently receive inputs from our senses and constantly drive control through intention.

Many studies have found that Recurrent Neural Networks (RNNs) exhibit similar dynamics as those found in the motor regions of the brain [10–14]. Leveraging these synthetic systems provides an effective approach to investigating the brain's underlying dynamics. It is also ideal for modeling how well a Latent Variable Model (LVM), a statistical model that relates a set of observed variables to a set of latent (unobserved) variables, can predict the underlying dynamics of the brain for a multitude of reasons.

These synthetic systems allow for precise control of the dynamics exhibited, the direct manipulation of the systems (e.g., external perturbations), providing unlimited freedom in the amount (number of trials, conditions), and specifications (number of neurons, bins, and bin size) of the data created. In the context of spiking data, synthetic systems also give us the ground truth firing rates from the spiking activity, which was sampled from a Poisson Process [15].

As described in [7], modeling the latent factors behind the dynamics exhibited in the motor cortex is of high priority for improving the control of Brain-Computer Interfaces (BCIs) and neural prostheses, two goals of which are crucial to improve the quality of life for those with neurological issues of all types. LVMs have proven useful in this regard as they can reveal the underlying states of the dynamical systems of interest and relate activity to behavior [16–19].

One such LVM, the Neural Data Transformer (NDT) [20], offers a promising solution to the application of nonlinear LVMs for real-time applications due to its non-recurrent architecture.

1.1 Neural Data Transformer

The NDT is a Transformer neural network based on the BERT [21] language model. Unlike the original Transformer model [22], NDT uses a series of stacked encoders with no decoders. It takes in neural spiking activity in the form of binned spikes and infers the underlying firing rates that produced the observed spiking activity. The firing rates are a smoother, more continuous estimate of a neuron's activity. The model is trained using masked language modeling (with 0's instead of mask tokens) and computes the Poisson negative log-likelihood between the inferred rates and the observed spikes. The architecture, as seen in Figure 1.1, is very simple as the transformer layers (light grey box) are only composed of multi-head attention, layer normalization, and a multi-layer perceptron (Feed-forward). In RNN LVMs such as Latent Factor Analysis via Dynamical Systems (LFADS) [23], each time step must be passed through the model independently, however in Transformer models, inference (in each layer) can be run for all time steps at once via the use of parallelization. The parallelized temporal routing of information is accomplished through the use of multi-head attention.

The authors of the NDT paper found that it could model autonomous systems such as the Lorenz system, spiking data from the monkey motor cortex, and an N-dimensional continuous time nonlinear "vanilla" RNN with no inputs. However, one of the biggest issues found with the model was that it could not model non-autonomous dynamical systems well. This was tested via the application of the NDT to the vanilla RNN with delta pulse inputs studied in [23]. That is,

$$\tau \dot{\mathbf{y}}(t) = -\mathbf{y}(t) + \gamma \mathbf{W}^{y} \tanh(\mathbf{y}(t)) + \mathbf{B}\mathbf{q}(t) \tag{1.1}$$

where the elements of the matrix $\mathbf{W}^{y}$ were drawn independently from a normal distribution with zero mean and a variance of 1/N. In the test performed by the authors, the NDT was applied to a dataset with γ = 2.5, N = 50, τ = 0.025 s and used Euler integration with Δt = 0.01 s.

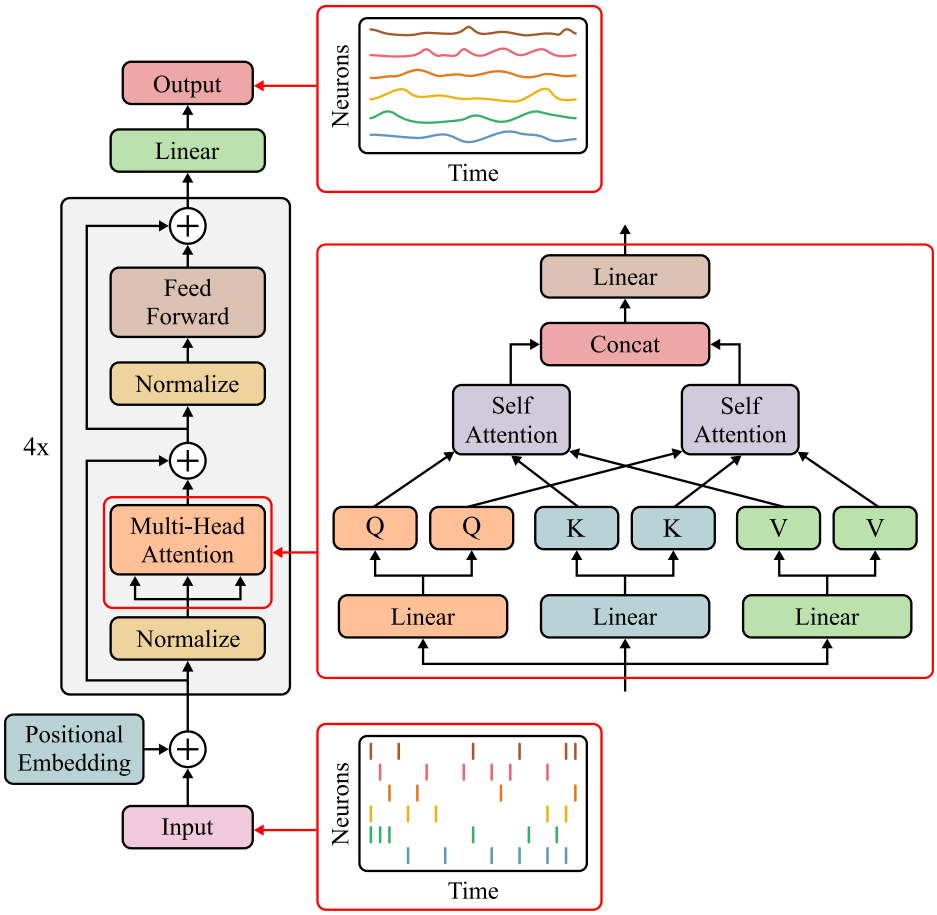

Figure 1.1: Architecture of the NDT

Spikes were generated by a Poisson process by first shifting and scaling $\tanh(\mathbf{y}(t))$ to give rates lying between 0 and 30 spikes/sec that were then used as firing rates to produce the spikes. To feed an input into the model, the components of $\mathbf{B}$ were independently sampled from a normal distribution with $\mu = 0$ and $\sigma = 1$. In each experiment, the network was perturbed by administering a delta pulse with a magnitude of 50 (given by $q(t) = 50\delta(t - t_{\text{pulse}})$ where δ is the Dirac delta function) at a random moment t_{pulse} between 0.25s and 0.75s (the total trial duration was 1s). This delta pulse influences the base rates generated by the data RNN, thereby altering the spike production mechanism and dynamical trajectory of the model. Fig. 1.2 shows underlying firing rates and spiking activity for a single trial. It can be seen that although the timing of the input (black triangle) is obvious in the rates, it is not evident whatsoever when just looking at the spikes (the input to NDT).

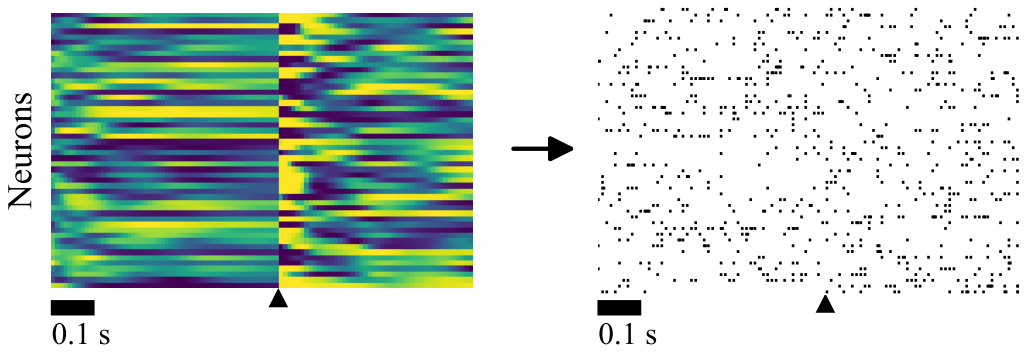

Figure 1.2: Conversion from firing rates to spikes

1.2 Neural Latents Benchmark

The Neural Latents Benchmark (NLB) [24] was developed as a machine learning benchmark to provide a standardized, quantitative evaluation of LVMs applied to neural data. The benchmark consists of four datasets of intracortical spiking data collected from non-human primates that span a variety of tasks and brain areas. The objective of the benchmark was to maximize the Poisson log-likelihood for an unseen or held-out set of neurons activity, given the activity of a held-in set of neurons. The Poisson log-likelihood was normalized and converted to bits per spike using the mean firing rates of each neuron as the baseline [25]. The normalized Poisson log-likelihood is referred to as bits per spike (bps), or

$$\text{bits/spike} = \frac{1}{n_{sp} \log 2} \sum_{n,t} \left(\mathscr{L}\left(\boldsymbol{\lambda}_{n,t}; \hat{\boldsymbol{y}}_{n,t}\right) - \mathscr{L}\left(\overline{\boldsymbol{\lambda}}_{n,:}; \hat{\boldsymbol{y}}_{n,t}\right) \right) \tag{1.2}$$

where $\hat{\boldsymbol{y}}$ is the activity for the held-out neuron n at time point t and $\boldsymbol{\lambda}$ is the predicted firing rates of that neuron. $\overline{\boldsymbol{\lambda}}_{n,:}$ is the mean firing rate for the neuron n, and n_{sp} is its total number of spikes. If bps is positive, then the model is inferring a neuron's time-varying activity better than a flat mean firing rate. The term "co-smoothing" in neuroscience refers to the prediction of the activity of held-out neurons based on the activity of held-in neurons on test data [26]. The main metric is referred to as co-smoothing bits per spikes (co-bps).

On one of the datasets, the Area2 bump task, a monkey engaged in a visually guided reaching task, wherein each trial involved reaching to a visually presented target via a manipulandum.

Neural spiking activity was recorded from area 2 of the somatosensory cortex, an area that receives and processes proprioceptive information, or information about where the body is in space relative to one's self. To perturb the somatosensory area, in a random 50% of the trials, the monkey's arm was unexpectedly bumped in a random direction by the manipulandum before the reach cue, and it then had to perform a corrective response. This demonstrates that the model needed to be able to model non-autonomous systems (which receive input) to correctly describe the activity after the perturbation.

Table 1.1: Results from the Neural Latents Benchmark on the Area2 bump dataset

Model	co-bps	Velocity R^2	PSTH R^2	fp-bps
NDT	0.2623	0.8672	0.6619	0.1184
AutoLFADS	0.2569	0.8492	0.6318	0.1505

In Table 1.1, NDTs performance on the Area2 Bump dataset is compared with LFADS, one of the most popular RNN-based LVMs. The Velocity R^2 metric quantifies how well the true velocity of the monkey's arm can be linearly decoded, while the PSTH R^2 metric quantifies how well a PSTH (Peri-Stimulus Time Histogram) of the inferred rates matches one computed from smoothed spikes. The last metric, fp-bps (forward pass bits per spike), is similar to co-bps, except that it quantifies how well the models can predict future activity for all neurons. A surprising result is that the NDT outperformed LFADS on this dataset because the authors of the NDT found, on a preliminary analysis using the chaotic RNN with delta pulse inputs studied in [23], LFADS vastly outperformed the NDT. Interestingly, there were minimal changes between the NDT models trained for the benchmark and the models trained on the synthetic data in the NDT paper. The most significant difference is the use of four NDT layers for the NLB models and six layers for the synthetic data models. This led to the first aim of this thesis's research, to re-evaluate the claims that NDT cannot model non-autonomous dynamical systems using the same number of layers as the top Area2 Bump NDT model.

1.3 Inferring Inputs

One benefit that LFADS has over NDT is its ability to infer inputs when modeling non-autonomous dynamical systems. LFADS uses 3 RNNs to model the dynamics of a system. First, the bi-directional encoder extracts the inputs to the system as well as the initial condition for the dynamics. Then, the dynamics of the model are simulated using the controller and the generator in tandem. The outputs of the controller, which are fed into the generator, are referred to as the inferred inputs. In [23], it was found that if enough regularization was applied when training LFADS, then the inputs could be interpretable if the timescale of the dynamics was low ($\gamma = 1.5$). NDT, on the other hand, do not feature a separate model to feed in inputs. This means that even though the model was successfully applied to data that requires inferring inputs, there is no way of extracting the internal representation that the model uses to quantify the input at each time step. This issue leads to the second aim of this thesis's research, which is to force the model to produce interpretable representations of the inputs via modifications to the architecture. Inspired by the use of heavy regularization in LFADS to increase interpretability in LFADS, we also aimed to explore if certain hyperparameter sets could elicit more interpretable input representations with NDT.

While NDT lack a separate model to infer inputs, one unique feature of the model over LFADS is the use of multi-head attention (MHA). MHA is how NDT temporally route information throughout the input sequence in a non-autoregressive fashion. Each of the multiple attention heads apply scaled dot product attention, and because the information contained in each head is independent, this computation can be run in parallel. Scaled dot product attention is applied to example data, $\mathbf{X}$, by first transforming it into queries, keys, and values via:

$$\mathbf{Q}_i = \mathbf{X}\mathbf{W}_i^Q \quad \mathbf{K}_i = \mathbf{X}\mathbf{W}_i^K \quad \mathbf{V}_i = \mathbf{X}\mathbf{W}_i^V, \tag{1.3}$$

where $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, and $\mathbf{W}_i^V$ are the weight matrices of head i that linearly weight each the features (spikes from each neuron) for all time steps. Following that, scaled dot product attention is then

applied via:

$$\text{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) = \mathbf{A}\mathbf{V}_i \tag{1.4}$$

with

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}_i \mathbf{K}_i^\top}{\sqrt{d_k}}\right). \tag{1.5}$$

After this weighting, the queries ($\mathbf{Q}_i$) and the keys ($\mathbf{K}_i$) are multiplied together to form the attention Matrix ($\mathbf{A}$). The attention matrix is scaled by the dimensionality of the keys, d_k. Row-wise softmax is then applied to the attention matrix to get a probability distribution, that is, each row will sum to one. In our case, there is a mask applied to the attention that limits the number of time steps forward (in the future) and backward (in the past) that each time step can attend to. This is done by adding -∞ to out-of-context time steps before the softmax (essentially zeroing them out). The final attention matrix is then multiplied by the values matrix, $\mathbf{V}_i$. The output of the scaled dot product attention is a dynamically weighted combination (across time) of the values.

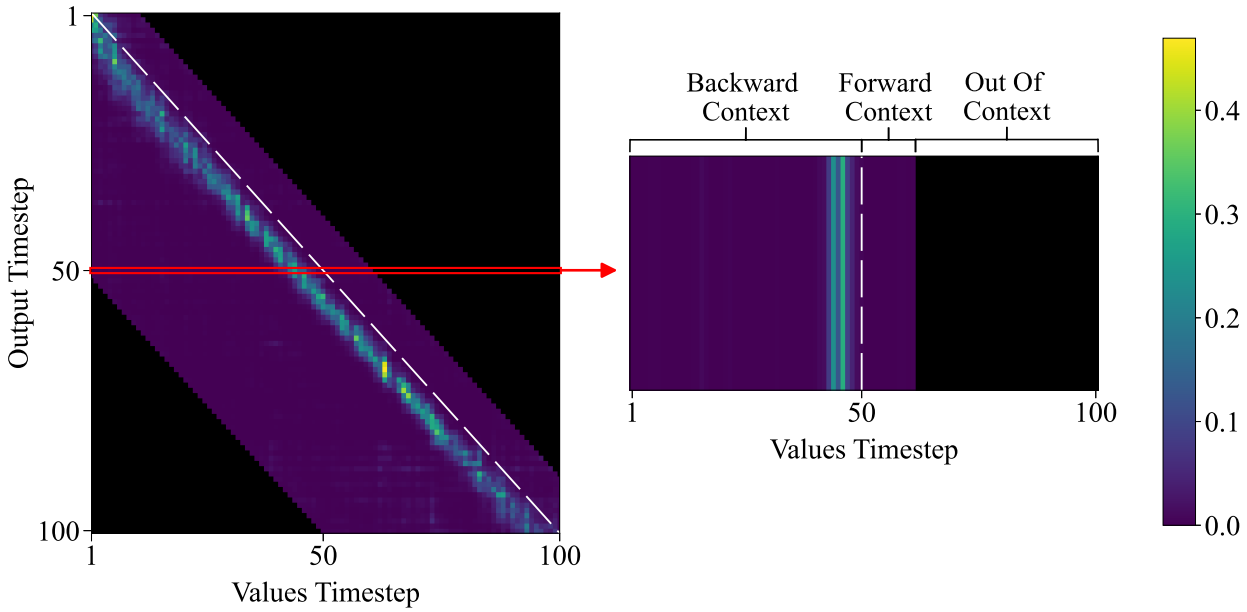

Figure 1.3: Interpreting the attention matrix

The attention matrix after softmax is illustrated in Figure 1.3, where we can see the post-softmax attention matrix on the left and the output time step 50 on the right. The areas in black represent an attention score of 0.0, while anything even minutely above that is dark purple. The dashed white line indicates where the time step is the same for both the outputs and the values. If we look at the right side of the figure, we can see how the value time steps are weighted to form the output at time step 50. The area to the left of the dashed white line represents the value time steps that are in the past (relative to output time step 50), and the values to the right are those in the future. In this example, the output for time step 50 seems to be mainly sourced from roughly five time steps in the past. This essentially pushes information further in time for the next layers of the model.

1.4 Input Extraction

In our case, we are interested in finding the model's representation for the input to the system. Because these delta pulse inputs heavily distort the dynamical trajectory, the time step in which these inputs occur should be relatively influential to all time steps that follow it. We use the attention matrix after softmax to estimate how "important" the value time step is by taking the mean across all output time steps, giving us a scalar value for each value time step. The formula used is,

$$\mathbf{M} = \text{Mean}(\mathbf{A}) = \frac{1}{t}\sum_{i=1}^{t}\mathbf{A}_{i,j} \tag{1.6}$$

where $\mathbf{M}$ is the inferred input magnitude or the average attention score across output time steps. This is illustrated in Figure 1.4, where we can see the attention matrix on the left, and the corresponding inferred input magnitude on the right. This example shows a strong inferred input pulse just before time step 50, which we can see on the left represents multiple output time steps in a row pulling all of their information from the same value time step. An inferred input value of 0.01 at each time step would mean that all output time steps are sourcing their information equally and that all time steps are of equal importance.

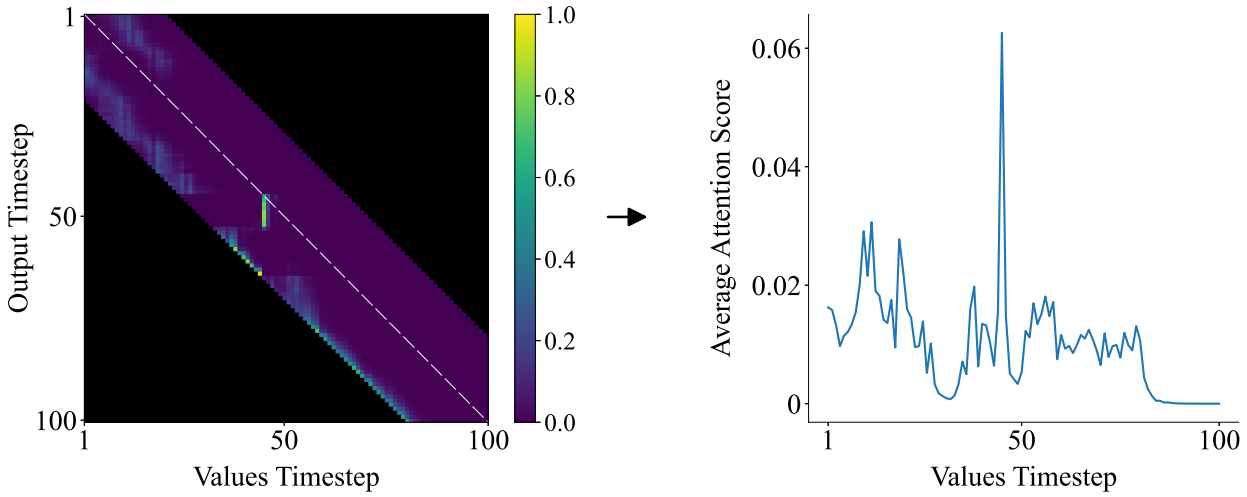

Figure 1.4: Extraction of inputs from the attention matrix

1.5 Normalized Inferred Input Magnitude

To quantify the signal-to-noise ratio of the inferred inputs, we created a metric that we will refer to as Normalized Inferred Input Magnitude (NIIM). The formula is,

$$\text{NIIM}(\mathbf{M}) = \frac{\mathbf{M}_k - \mu_{\mathbf{M}_t|t\neq k}}{\sigma_{\mathbf{M}_t|t\neq k}} \tag{1.7}$$

where $\mathbf{M}_k$ is the magnitude of the inferred input at the time k in which the delta pulse occurred. $\mu_{\mathbf{M}_t|t\neq k}$ is the mean of the inferred inputs, excluding the time in which the delta pulse occurred, and $\sigma_{\mathbf{M}_t|t\neq k}$ is the standard deviation of the inferred inputs, excluding the time in which the delta pulse occurred. Figure 1.5 visually describes the formulation of the metric. In Figure 1.6, three examples of the metric are presented to give some intuition behind the scalar values. In both figures, the black triangle represents the time step in which the input occurred.

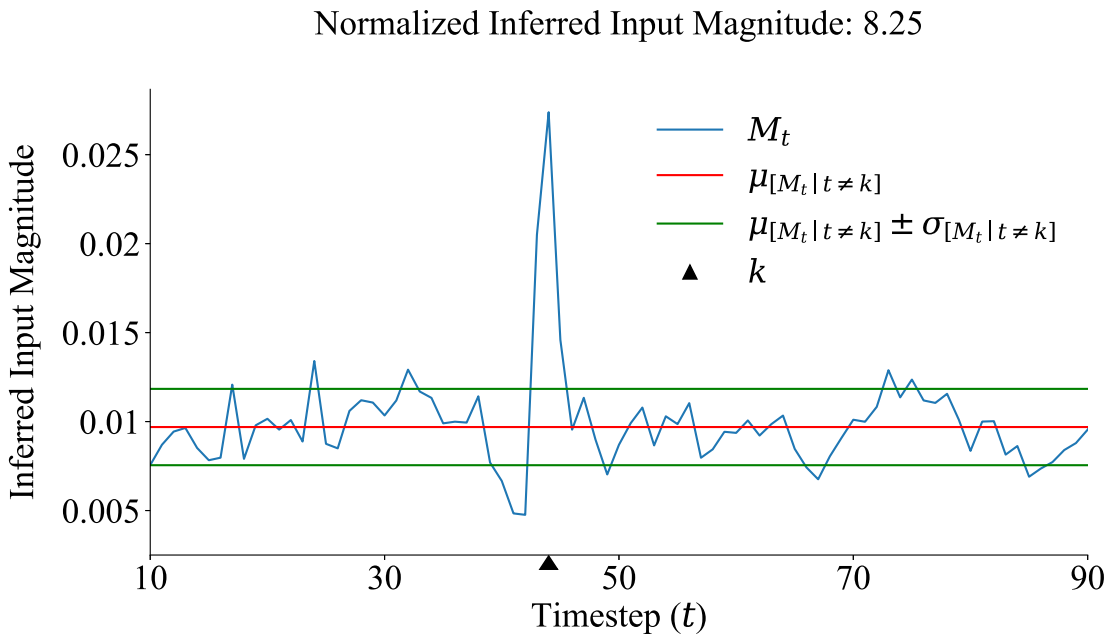

Figure 1.5: Quantification of the signal strength vs. noise of the inferred inputs

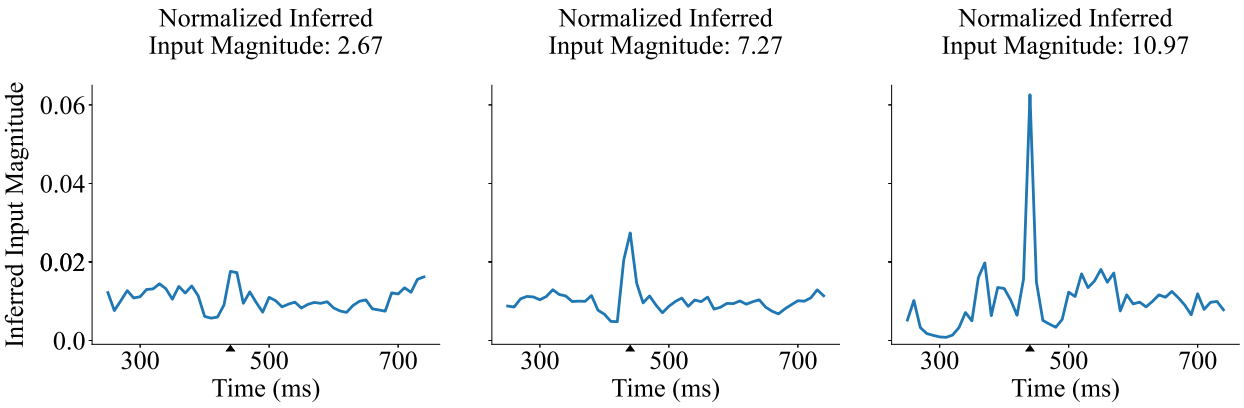

Figure 1.6: Normalized Inferred Input Magnitude examples

Chapter 2

Methods

To re-evaluate the claims that NDT cannot model non-autonomous dynamical systems, we applied the model to the same non-autonomous dynamical system used by the NDT authors. That is, the chaotic RNN with delta pulse inputs studied in [23]. While LFADS was only applied to chaotic RNNs with inputs at $\gamma = 1.5$ and $\gamma = 2.5$ in [23], in this analysis, a third timescale was included, $\gamma = 3.5$. The third gamma is meant to push the ability to infer the inputs to the limit, as the difficulty of separating the input from the dynamics increases with the timescale of the dynamics. That is to say, as the changes in the RNN become more rapid, it can be hard to distinguish the input from the underlying dynamics of the network when using binned spike counts.

To verify the results in [20], we also applied NDT to a chaotic RNN with no inputs. However, in this analysis, the same number of samples was used as in [23]. Lastly, to ensure that the ability to model non-autonomous dynamics was not limited to just one-dimensional inputs, NDT was applied to a chaotic RNN with two inputs, one of magnitude 5 and the other of magnitude 10. An example trial worth of Poisson rates produced by each of these chaotic RNNs is presented in Figure 2.1. As with [23], the Poisson rates were then sampled to get spiking activity, which was then fed into the NDT model.

In [23], the synthetic RNN data consisted of 400 conditions with 10 spiking trials sampled for each condition. To ensure the model was evaluated on unseen data, another 10 trials for each of the 400 conditions were sampled for use in the test set. This provided a total of 8000 trials, with 3200 used for training, 800 used for validation, and 4000 used for testing the final models. All results, tables, and figures throughout this work are from the test set (apart from Figure 3.1).

To successfully train a NDT model, a hyperparameter search must be performed. A random search (1000 runs) was performed on the simplest dataset, $\gamma = 1.5$, over the parameters found in Table 2.1. The uniform(x, y) found in the table represents values sampled from a uniform

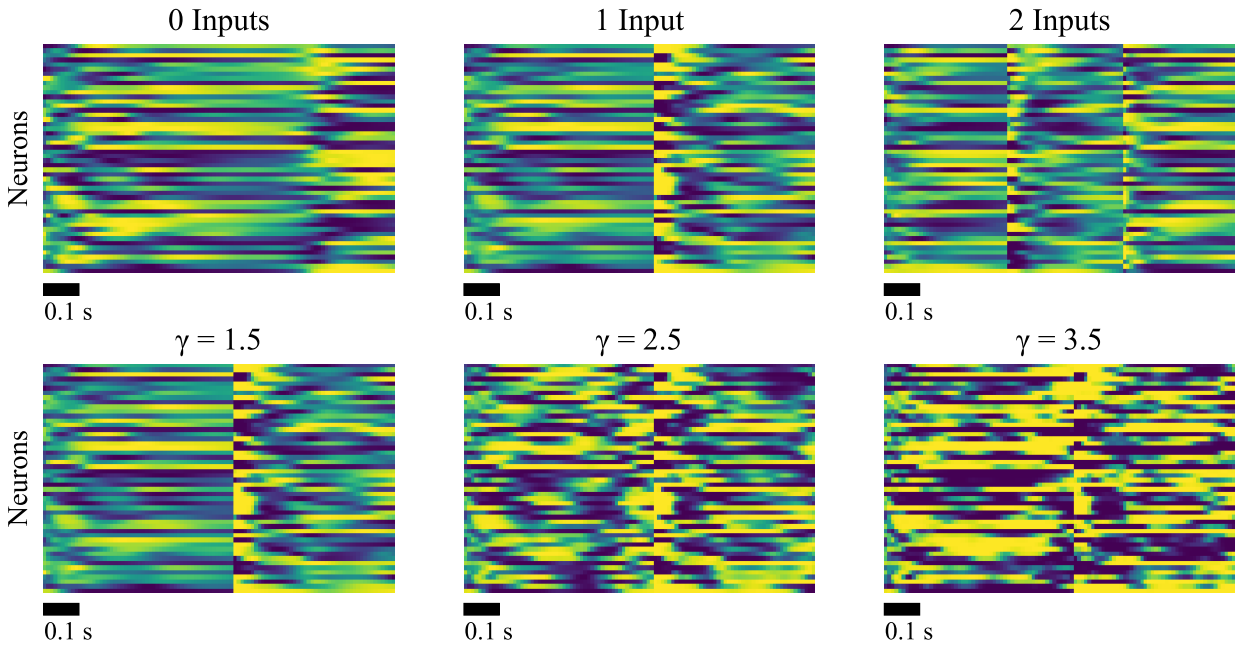

Figure 2.1: Overview of the datasets

distribution with minimum value x and maximum value y. To make sure that the architecture chosen was not a statistical fluke, three models were trained with different random seeds for each sampled hyperparameter set. Note that one run for a random search is equivalent to 3 models trained.

After the initial random search, two additional random searches were performed to find ways that NDT might be optimized to represent the inputs in a repeatable way. This was done by choosing the top model based on its average NIIM across three training seeds. The first random search (2000 runs) swept over the forward context, the backward context, and the head dimensionality. The exact parameters swept over can be found in Table 2.2. The context forwards was limited in the later layers to promote a more interpretable input. The intuition behind this being that we are interested in "important" time steps that have a large impact on the proceeding time steps, not the past. Lastly, a random search (1000 runs) over dropout was performed. The exact parameters swept over can be found in Table 2.3. The dropout ranges were altered to minimize the number of models that failed to converge.

Table 2.1: Initial random search hyperparameter search space

Hyperparameter	Search Space
Embedding Dimensions	128, 256
Hidden Size	128, 256, 512
Max Learning Rate	0.05, 0.01, 0.005
Context Forwards	5, 10, 20, 30, 50
Context Backwards	5, 10, 20, 30, 50
Weight Decay	5.0e-03, 5.0e-04, 5.0e-05
Dropout	Uniform(0.1, 0.5)
Dropout Rates	Uniform(0.1, 0.5)
Dropout Embedding	Uniform(0.1, 0.5)
Dropout Attention	Uniform(0.1, 0.7)

Table 2.2: Context and dimensionality random search hyperparameter search space

Hyperparameter	Search Space
Context Forwards	
Layer 1	0, 1, 50
Layers 2 - 4	0, 1
Context Backwards	1, 5, 10, 20, 50
Head Dimensionality	4, 8, 16, 32, 50, 64

To train the models, 5 NVIDIA A40 GPUs and 1 NVIDIA A100 GPU were used with 128 AMD EPYC 7452 32-Core CPUs. To run the random searches, Weights and Biases [27] was used. The total compute time (sum of training times for all models) for all random searches combined is 158 days, original NDT sweep: 49 days, modified context dimensionality sweep: 61 days, modified dropout sweep: 48 days.

Table 2.3: Dropout hyperparameter search space

Hyperparameter	Search Space
Dropout	Uniform(0.1, 0.5)
Dropout Rates	Uniform(0.1, 0.3)
Dropout Embedding	Uniform(0.1, 0.4)
Dropout Attention	Uniform(0.1, 0.8)

Chapter 3

Results

3.1 Original NDT Random Search

As a verification of the results found in [20], we looked at the validation set inferred rates R^2 vs. bits per spike to validate that the primary metric used to select models (bits per spike) was truly finding models that infer the underlying rates well. Bits per spike was defined in equation (1.2) and inferred rates R^2 is defined as:

$$\text{Inferred Rates } R^2 = \frac{1}{N}\sum_{n=1}^{N}\left(1 - \frac{\sum_{t=1}^{T}(y_{nt} - \hat{y}_{nt})^2}{\sum_{t=1}^{T}(y_{nt} - \overline{y_n})^2}\right), \tag{3.1}$$

where N is the number of neurons, T is the number of time steps, y_{nt} is the true firing rate of neuron n at time step t, $\hat{y}_{nt}$ is the inferred firing rate of neuron n at time step t, and $\overline{y_n}$ is the mean firing rate of neuron n over all time steps. As shown in Figure 3.1, there is a strong relationship between bits per spike and inferred rates R^2 for all 3000 models from the random search. Following the same procedure as in [20], we compared NDT's performance to AutoLFADS [28]. AutoLFADS improves upon the training procedure in LFADS by using population-based training instead of a random search to select hyperparameters. Another improvement is the use of coordinated dropout, which essentially restricts the model to only back-propagate the loss for channels that were dropped out (zeroed out). This procedure is used to avoid the problem of identity overfitting (estimating firing rates that are identical to the spiking activity) by forcing the model to rely on the population activity as a whole to predict the activity of individual channels.

The top model was chosen for its average performance on the validation set across the three models trained for each HP set. The chosen parameters for the top model can be seen in Table 3.1 below.

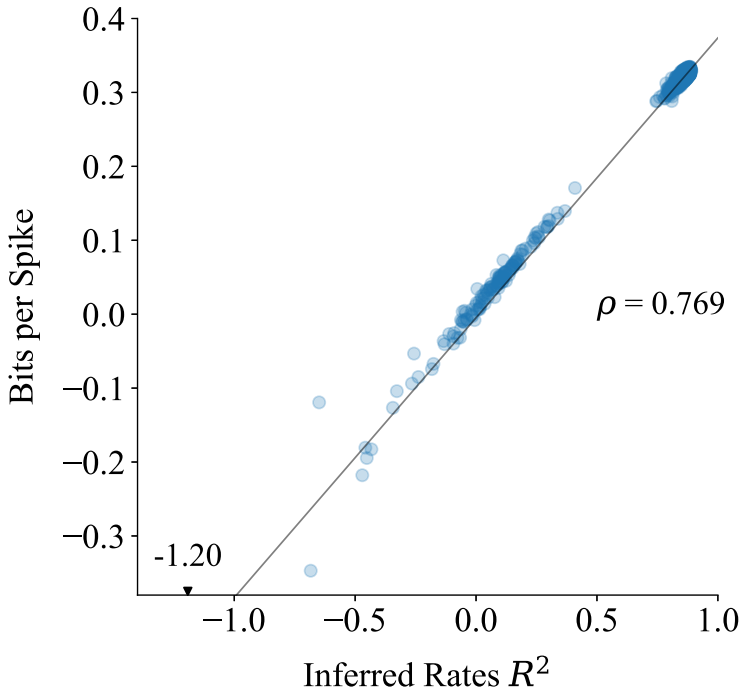

Figure 3.1: Validation set bits per spike vs inferred rates R^2

It can be seen visually in Figure 3.2 that NDT and LFADS perform similarly across the three levels of inputs (No inputs, one input, two inputs). This difference is quantified across the three models trained for both NDT and LFADS in Table 3.2. The NDT models were found to have remarkably similar performance across seeds (apart from the two inputs model, which had some low performing models).

Again in Figure 3.3, NDT and LFADS perform similarly across the three levels of gamma (1.5, 2.5, 3.5). NDT even performed slightly better in the higher gamma cases, where LFADS would often have spurious outliers after larger changes in the rates. This difference is quantified across the three models trained for both NDT and LFADS in Table 3.3. NDT does slightly better than LFADS across all timescales tested.

3.2 Modified NDT

In Figure 3.4, we can see that the model is predicting inputs in layer four because, at that time step, it begins to pull information from the future to the time step where the input oc-

Table 3.1: Unmodified NDT best architecture from random search

Hyperparameter	Value
Context Backwards	50
Context Forwards	10
Head Dimensions	50
Dropout	0.3774
Dropout Attention	0.2182
Dropout Embedding	0.1583
Dropout Rates	0.1428

Table 3.2: Unmodified NDT rate reconstruction across inputs

# of Inputs	NDT ($R^2\uparrow$)	AutoLFADS ($R^2\uparrow$)
0	$0.873_{\pm 0.000}$	$0.854_{\pm 0.009}$
1	$0.886_{\pm 0.001}$	$0.878_{\pm 0.002}$
2	$0.841_{\pm 0.017}$	$0.854_{\pm 0.002}$

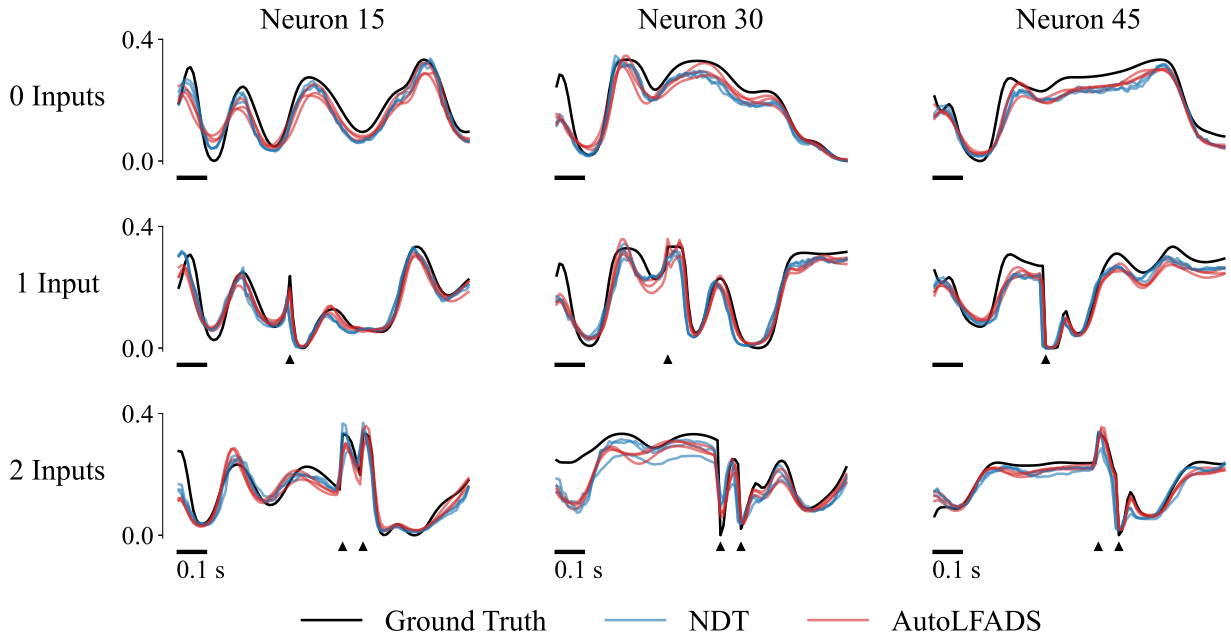

Figure 3.2: Best unmodified NDT model from random search vs. LFADS across three levels of Inputs for three neurons. The y-axis is the neurons firing rate and the x-axis is time

Table 3.3: Best unmodified NDT model from random search vs. LFADS across three levels of gamma

γ	NDT ($R^2\uparrow$)	AutoLFADS ($R^2\uparrow$)
1.5	$0.886_{\pm 0.001}$	$0.878_{\pm 0.002}$
2.5	$0.897_{\pm 0.000}$	$0.884_{\pm 0.004}$
3.5	$0.876_{\pm 0.000}$	$0.857_{\pm 0.002}$

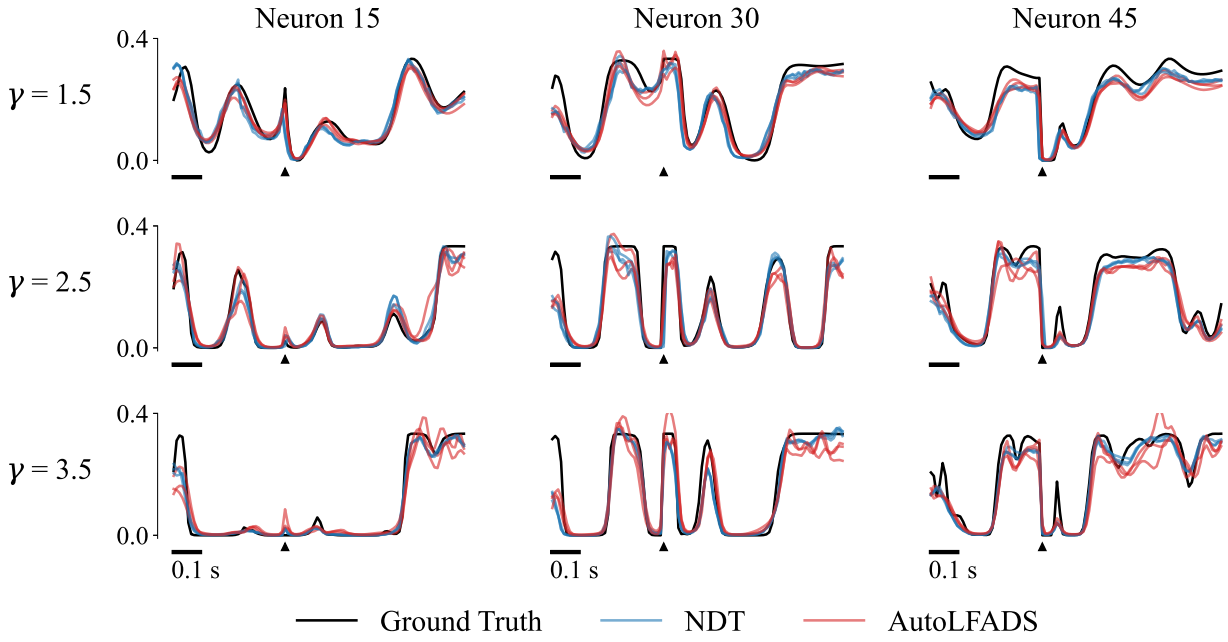

Figure 3.3: Best unmodified NDT model from random search vs. LFADS across three levels of gamma for 3 neurons. The y-axis is the neurons firing rate and the x-axis is time

curred. However, our method of extracting inputs looks at which time steps are pulling information from the input, so this would not get picked up as successfully inferring the input. When looking at Figure 3.5, we can see that the inputs were most likely to occur in the last layer, but in that layer, its location was split between the two heads. This results in training with different seeds leading to input localization in different heads. For reference, the NIIM for the best (validation bps) model from the initial unmodified NDT random search was μ=0.0614, σ=0.4926.

To address the issue of the input representation relocating heads across random initializations, the second head in the last layer was moved to the first layer. The location (layer one) of the last layer's second head was based on preliminary testing with the other layers as the

destination. After the two random searches to improve the interpretability of the inputs, the final model architecture can be found in Table 3.4. The changes are extremely evident when comparing the attention matrices between the modified model, found in Figure 3.6, and the unmodifed model from Figure 3.4. When looking at the attention matrix in layer 4 of the modified model, the timing of the input is visually evident even without performing any processing (such as averaging).

Table 3.4: Best modified model vs. LFADS across three levels of gamma

Hyperparameter	Value(s)
Context Backwards	50, 5, 5, 14
Context Forwards	50, 0, 0, 0
Dropout	0.3652
Dropout Attention	0.1420
Dropout Embedding	0.3828
Dropout Rates	0.2692

As verification that the changes made to elicit interpretability did not harm the performance of the model, we compared the results to the original model and found minimal impacts. The differences across the three levels of input can be found in Table 3.5. We can see that the modifications actually help the model when it comes to 2 inputs. The differences across the three different levels of gamma can be found in Table 3.6. It is evident that the modifications do appear to hurt NDT performance the most on the fastest timescale dynamics; However, the modified NDT is still outperforming LFADS.

For Figure 3.7, the analyses performed in [23] were followed. We calculated the average inferred input strength around the delta pulses (averaged across all trials), as well as the average inferred input strength at time steps in which the delta pulse was not near it. That was compared against the inferred input strength for a model trained on data with no delta pulses applied. It was found that the NDT was much better at inferring inputs than LFADS, so we ex-

Table 3.5: Modified NDT vs. original NDT vs. LFADS across three levels of Inputs

# of Inputs	Original NDT ($R^2\uparrow$)	Modified NDT ($R^2\uparrow$)	AutoLFADS ($R^2\uparrow$)
0	$0.873_{\pm 0.000}$	$0.864_{\pm 0.000}$	$0.854_{\pm 0.009}$
1	$0.886_{\pm 0.001}$	$0.881_{\pm 0.000}$	$0.878_{\pm 0.002}$
2	$0.841_{\pm 0.017}$	$0.845_{\pm 0.000}$	$0.854_{\pm 0.002}$

Table 3.6: Modified NDT vs. original NDT vs. LFADS across three levels of gamma

γ	Original NDT ($R^2\uparrow$)	Modified NDT ($R^2\uparrow$)	AutoLFADS ($R^2\uparrow$)
1.5	$0.886_{\pm 0.001}$	$0.881_{\pm 0.000}$	$0.878_{\pm 0.002}$
2.5	$0.897_{\pm 0.000}$	$0.888_{\pm 0.000}$	$0.884_{\pm 0.004}$
3.5	$0.876_{\pm 0.000}$	$0.863_{\pm 0.000}$	$0.857_{\pm 0.002}$

tended the analysis performed to include γ=3.5. Similar to the results found in [23], it was clear that as the timescale of the dynamics increases, the inputs get less and less distinct.

As a follow-up analysis, the average response near the delta pulse was split into three time steps that comprise it, the time step before the delta pulse, the time step of the delta pulse, and the time step after the delta pulse. The results of this are pictured in Figure 3.8. That analysis was then expanded out to 20 time steps around the delta pulse (Figure 3.9), and it can be seen that the model is giving a strong signal at the input relative to the surrounding time steps, the difference decreases as the gamma increases. Single-trial responses are remarkably consistent (Figure 3.10); however, the time step before the delta pulse seems to often be mistaken as the true input time. This is confirmed by looking at the true delta pulse vs. predicted delta pulse time (Argmax of inferred inputs) in Figure 3.11.

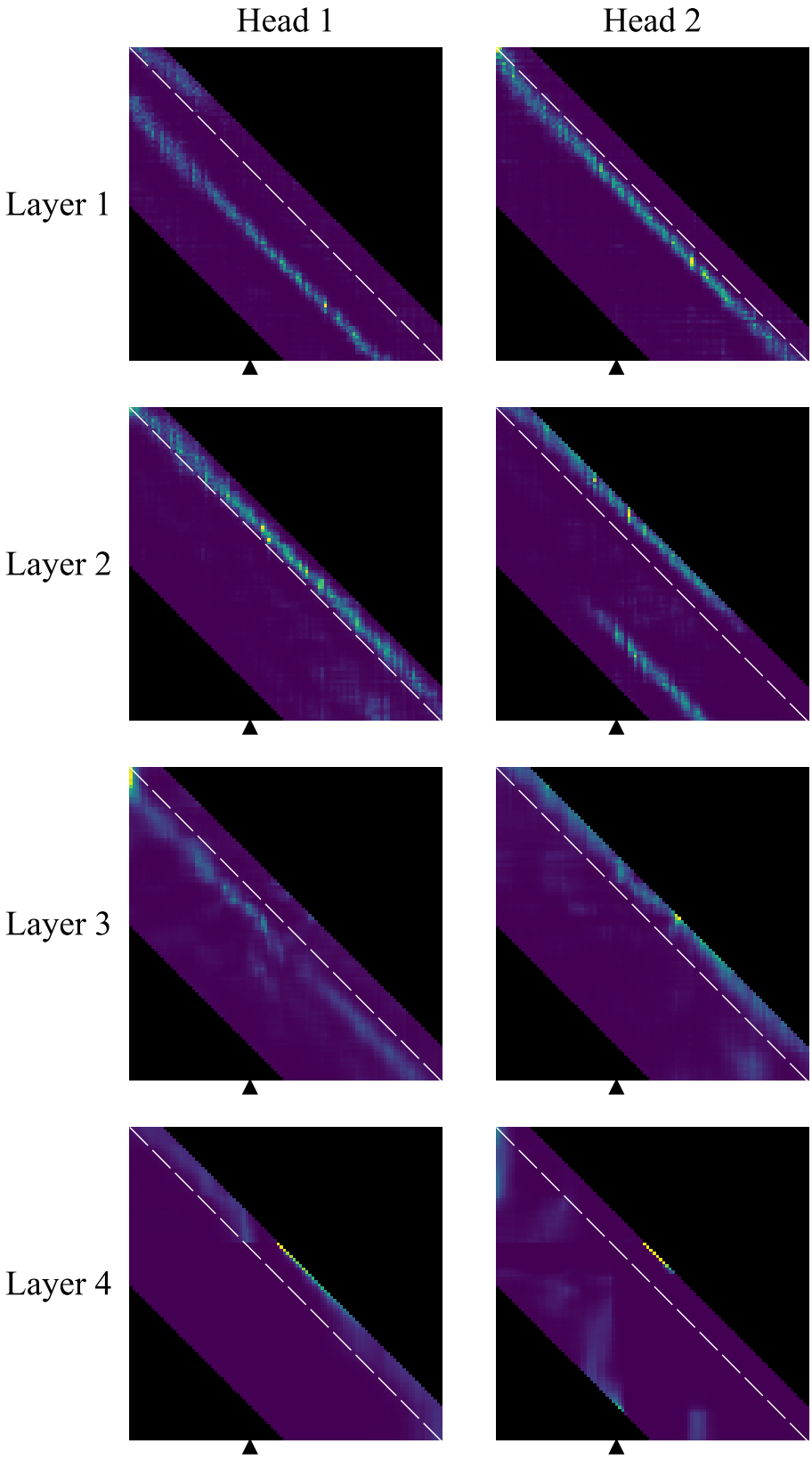

Figure 3.4: Attention matrices for all Layers and heads for one trial

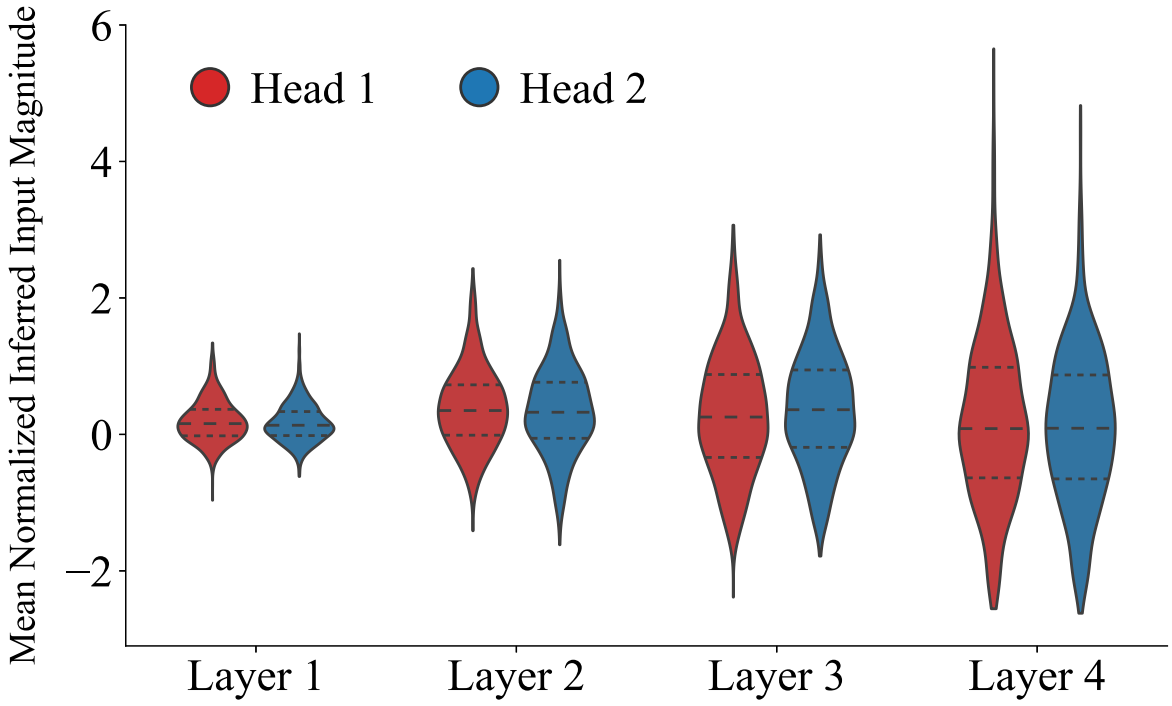

Figure 3.5: Histogram of input representation across heads and layers for all models in random search

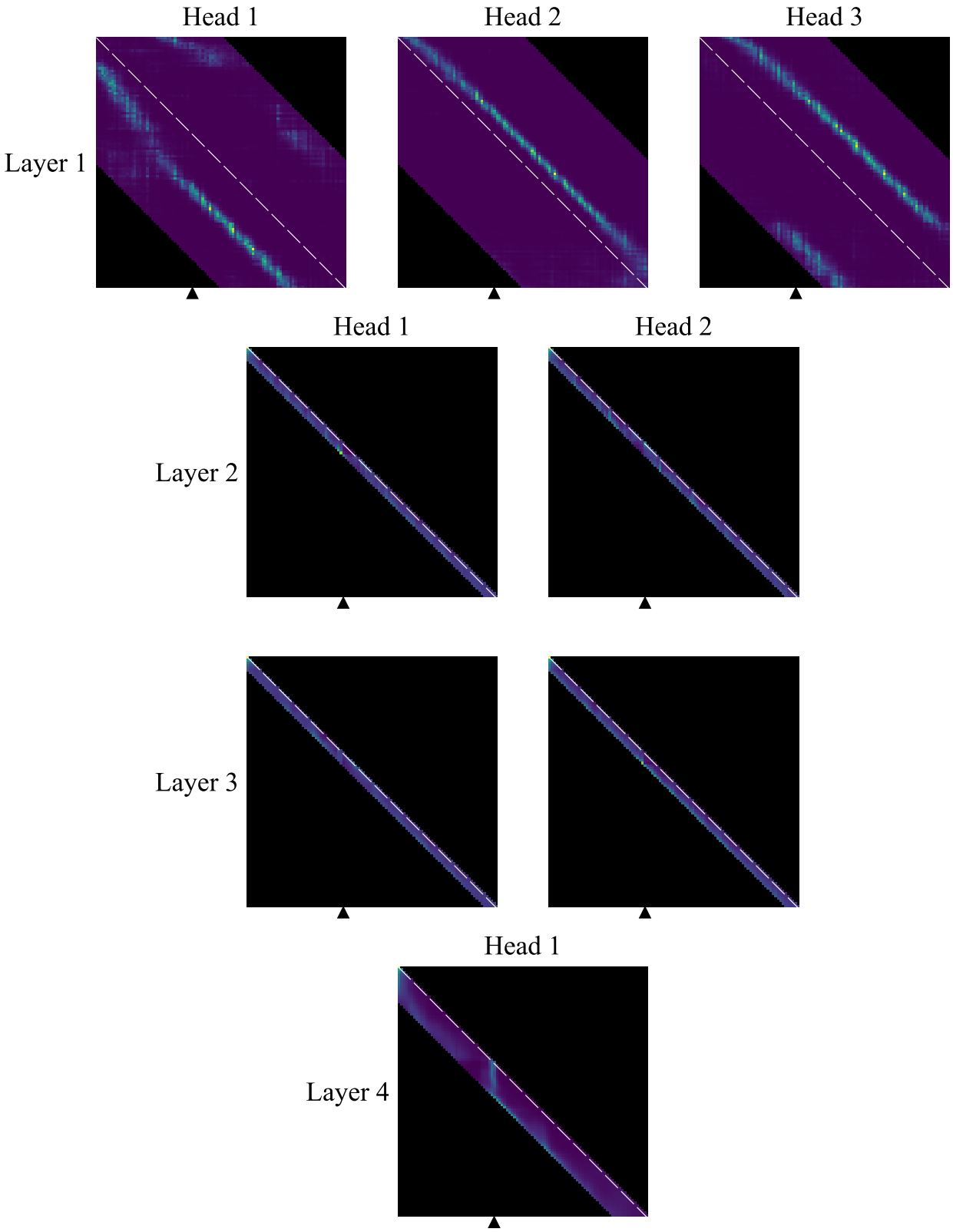

Figure 3.6: Attention matrices for modified model for all layers and heads for one trial

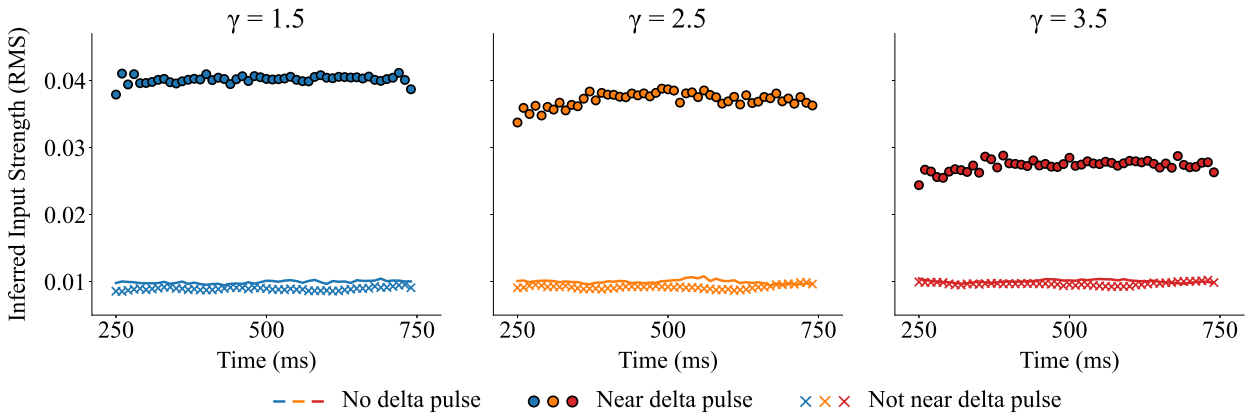

Figure 3.7: Inferred input magnitude strength for three levels of gamma

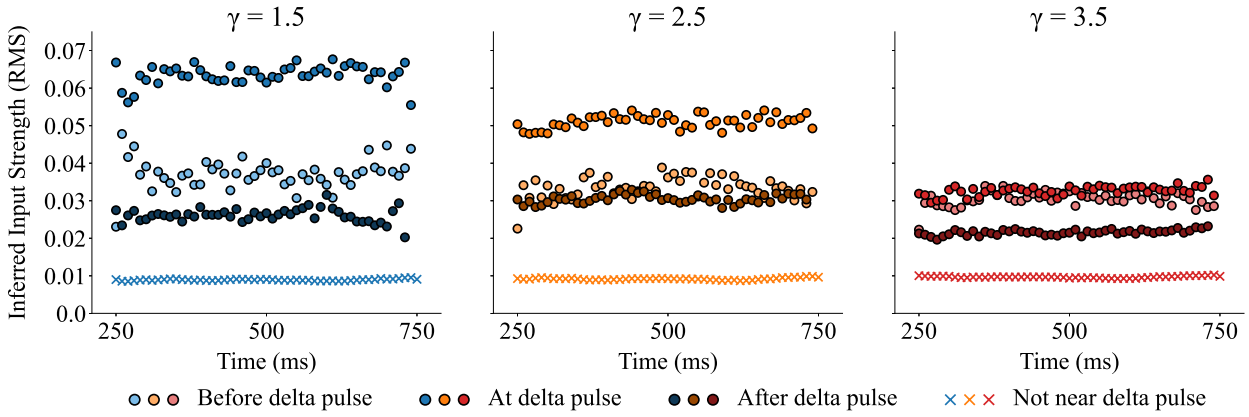

Figure 3.8: Inferred input magnitude strength before, at, and after the delta pulse for three levels of gamma

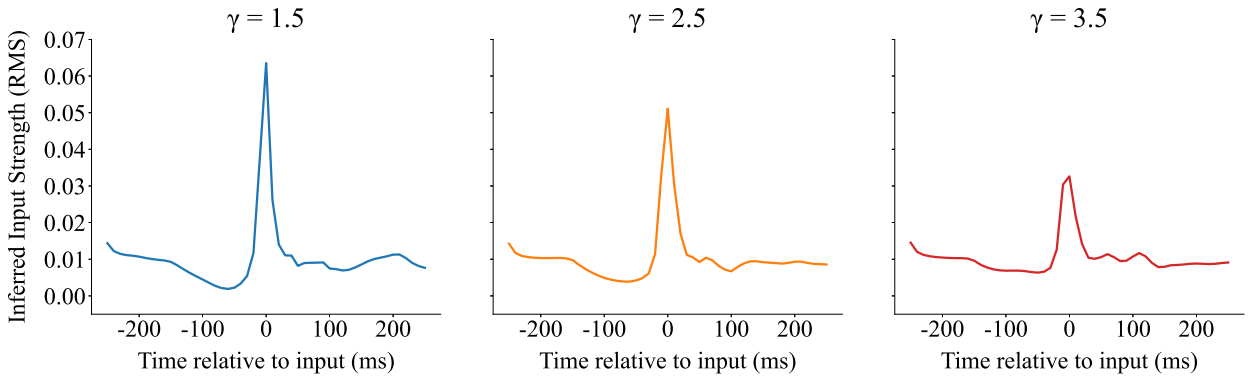

Figure 3.9: Average distribution of inferred inputs around the time of delta pulse

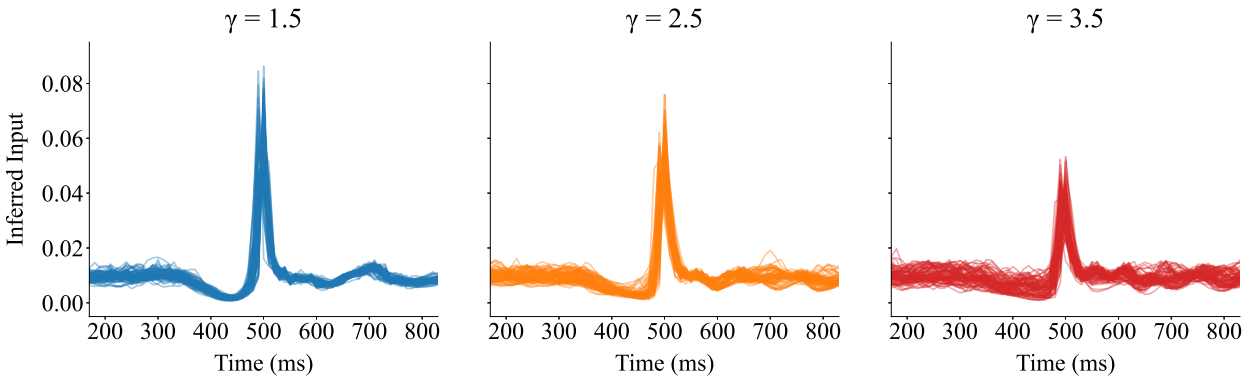

Figure 3.10: Inferred inputs for trials with a delta pulse at 500ms

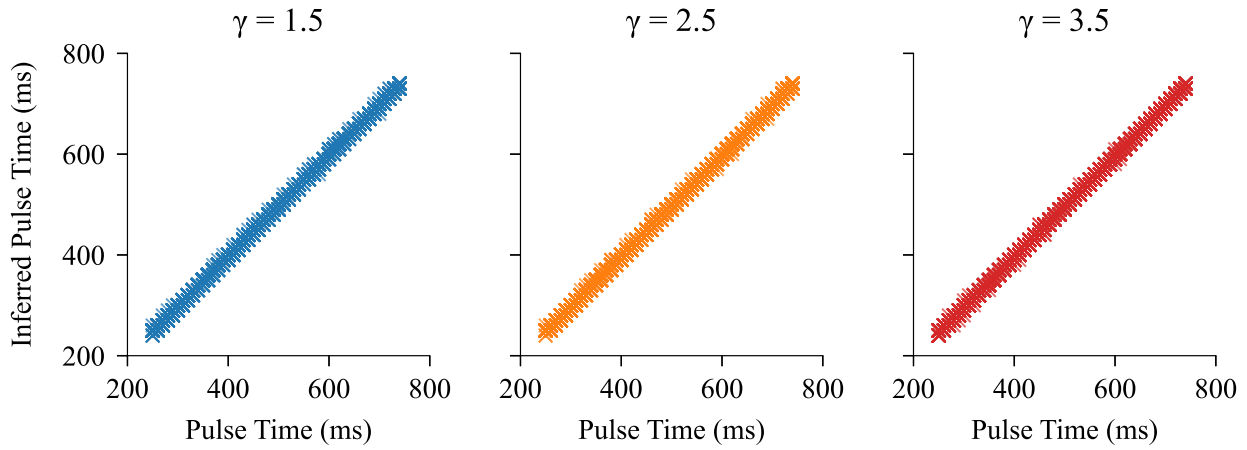

Figure 3.11: Predicted vs. true delta pulse times

3.3 Ablations

To test what specific change in the modified model resulted in the ability to infer inputs, we removed specific changes and compared them to the original model (Figure 3.12). Note that to quantify NIIM for NDT with the original number of heads (as well as the unmodified/original NDT), the head with the greatest NIIM was used (layer 4, head 1). The ability to infer inputs benefited the most from a combination of both the layer-specific context and the moving of the attention head from the last to the first layer. Dropout had a slight but insignificant effect on interpretability.

We looked at how the inferred rates R^2 were affected by the changes and found that the individual modifications had little impact on the R^2 score (Figure 3.13). To test how the modified architecture does across different dynamics, we applied it to a chaotic RNN with a different seed. We found that the model did well on this new dataset in terms of NIIM (Figure 3.7) and inferred rates R^2 (Figure 3.8). To test if the NDT was truly learning dynamics or if it was simply applying an advanced smoothing technique, we also applied the model to the RNN with different dynamics and found that it did worse than LFADS (also Figure 3.8). However, when testing the NIIM on this new seed, the NIIM was extremely low. To further investigate, we applied the same analyses performed on the seed 5 dataset and found that the model still could predict the timing of the inputs but strangely predicted it early very consistently (Figure 3.14, Figure 3.15, Figure 3.16, Figure 3.17, Figure 3.18). The NIIM is very sensitive to these early predictions, and they are heavily penalized, leading to the low score found above.

Table 3.7: Dataset seed and its impact on Normalized Inferred Input Magnitude for the modified NDT model

Dataset seed trained on	Dataset seed tested on	$NIIM\uparrow$
5	5	$10.329_{\pm 0.326}$
678	678	$3.283_{\pm 0.056}$
5	678	$0.028_{\pm 0.024}$

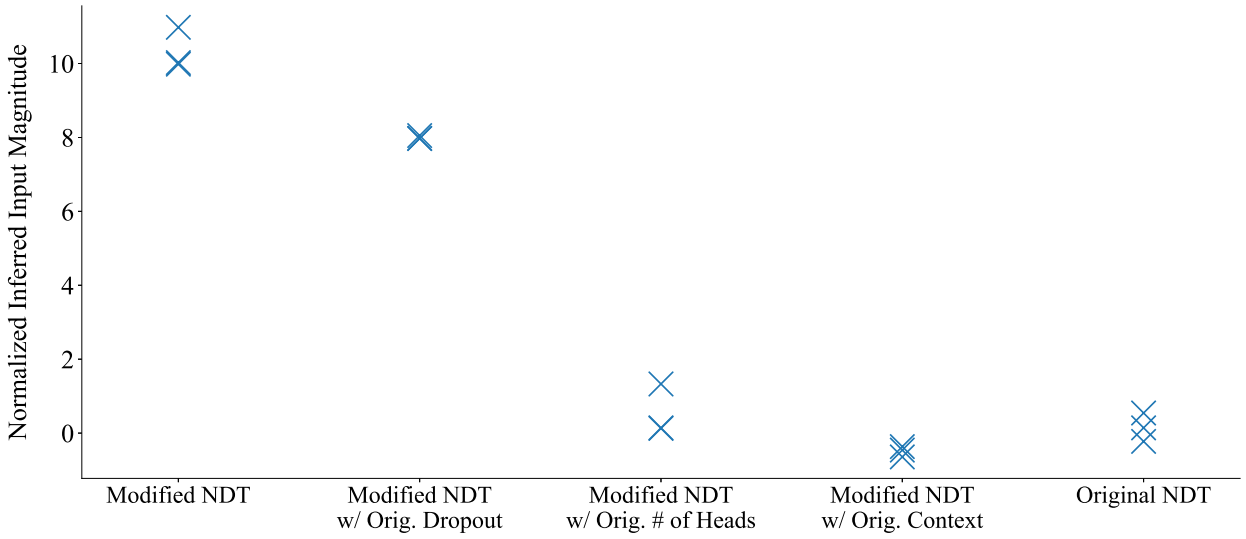

Figure 3.12: Ablations and their impact on Normalized Inferred Input Magnitude

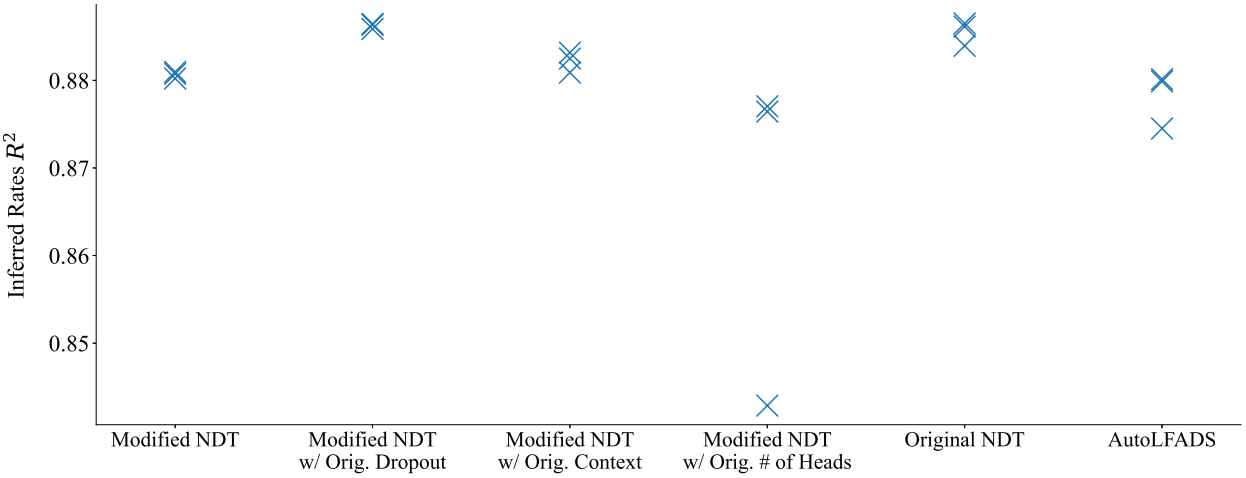

Figure 3.13: Ablations and their impact on inferred rates R^2

Table 3.8: Dataset seed and its impact on inferred rates R^2

Dataset seed trained on	Dataset seed tested on	Original NDT ($R^2\uparrow$)	Modified NDT ($R^2\uparrow$)	AutoLFADS ($R^2\uparrow$)
5	5	$0.886_{\pm 0.001}$	$0.881_{\pm 0.000}$	$0.878_{\pm 0.002}$
678	678	$0.885_{\pm 0.000}$	$0.875_{\pm 0.000}$	$0.877_{\pm 0.001}$
5	678	$-0.710_{\pm 0.006}$	$-0.691_{\pm 0.003}$	$-0.512_{\pm 0.030}$

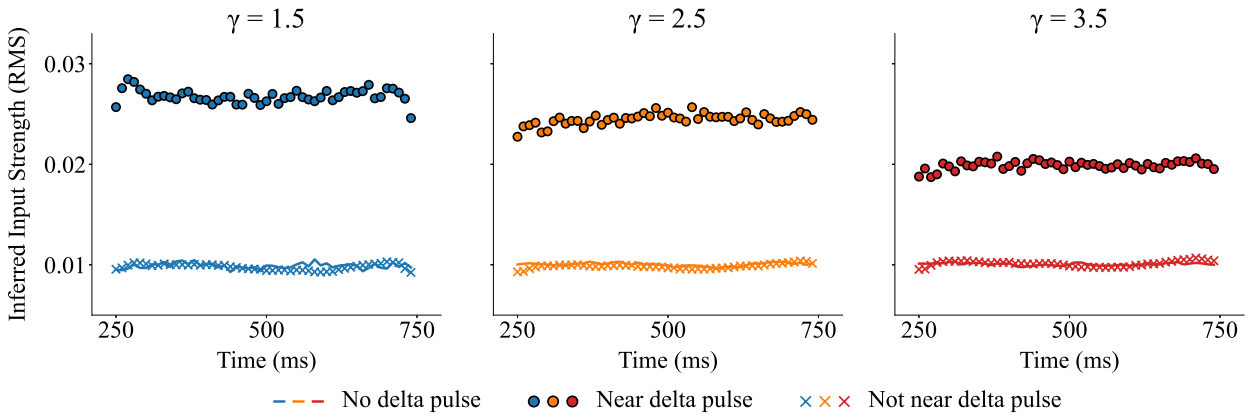

Figure 3.14: Inferred input magnitude strength for three levels of gamma for the dataset with a seed of 678

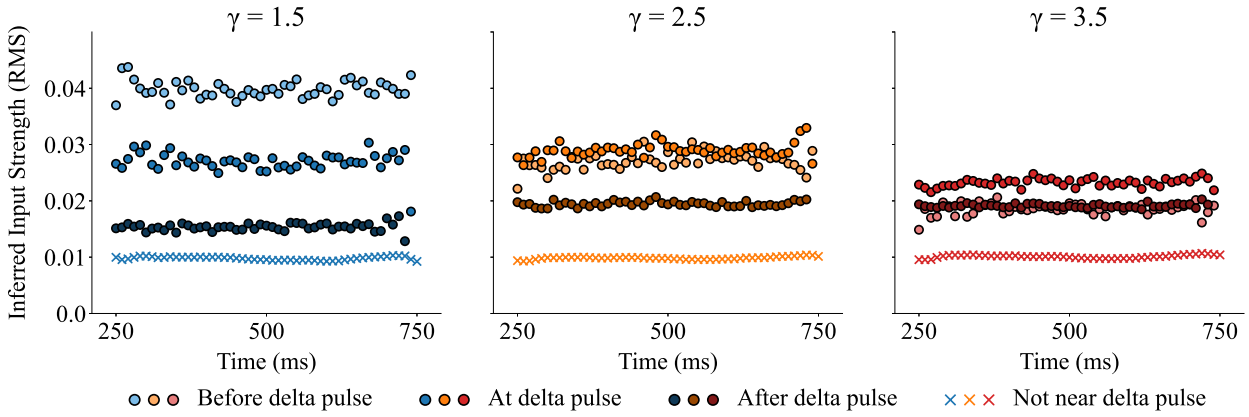

Figure 3.15: Inferred input magnitude strength before, at, and after the delta pulse for three levels of gamma for the dataset with a seed of 678

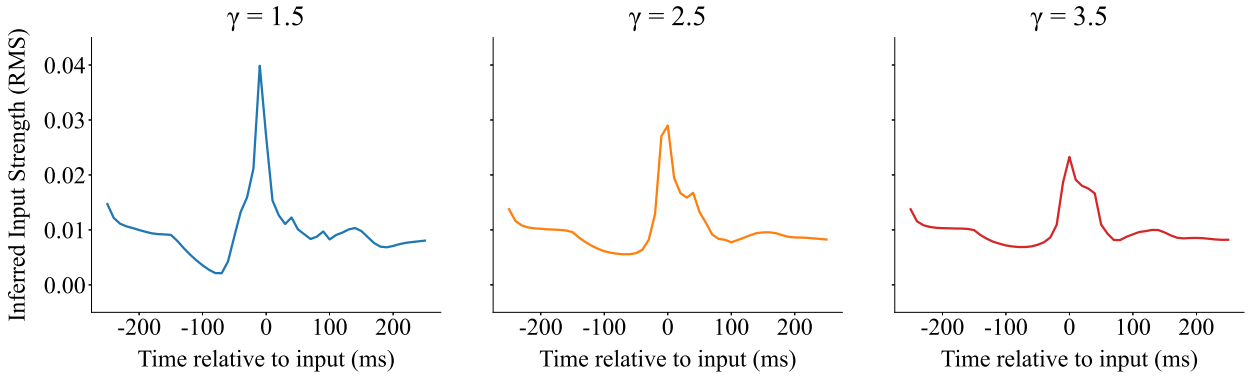

Figure 3.16: Average distribution of inferred inputs around the time of delta pulse for the dataset with a seed of 678

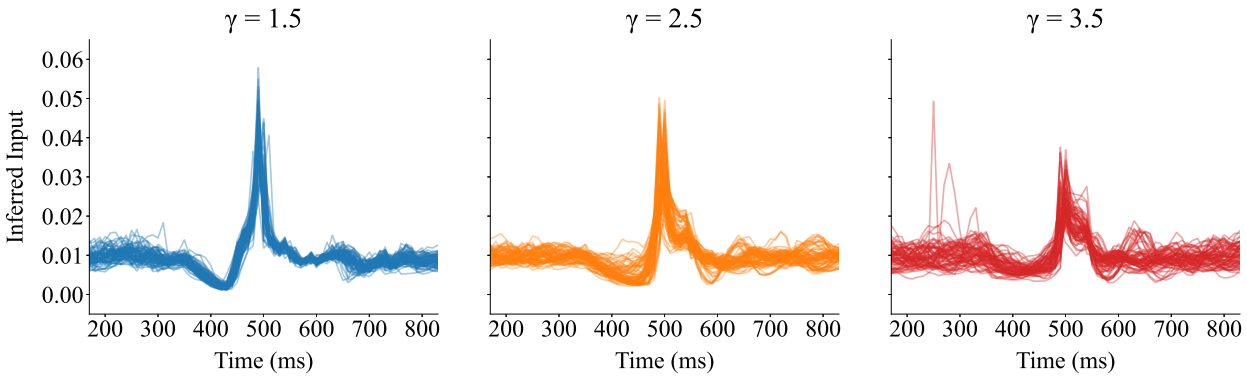

Figure 3.17: Inferred inputs for trials with a delta pulse at 500ms for dataset with seed of 678

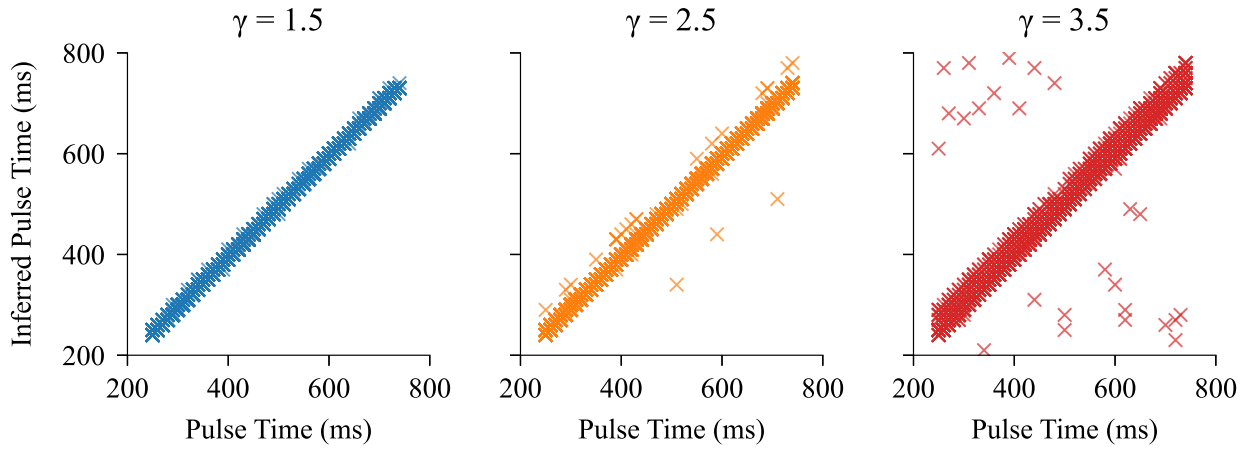

Figure 3.18: Predicted vs. true delta pulse times for the dataset with a seed of 678

Chapter 4

Conclusions

4.1 Summary

Contrary to [20], the results of this work suggest that the NDT can, given the proper hyperparameters, model non-autonomous dynamical systems. We found that the NDT can be successfully applied to chaotic RNNs with delta pulse inputs by using the changes made to the architecture for the NLB (four layers vs. six layers), along with an extensive hyperparameter search. This finding opens the gates to a vast number of BCI applications that were once thought to be impossible with this model. One of the reasons that LFADS was considered a more likely target for online use was the fact that it could model motor cortex dynamics in the context of perturbations. Now that it has been shown that the NDT can indeed model dynamics under this regime, the fast inference speeds, along with the performance on the NLB, makes the NDT a very likely contender for online BCI use.

We also found that specific arrangements of the attention heads, and their context windows, can evoke consistently interpretable representations of the inputs to the modeled system. As with LFADS, the magnitude of these inferred inputs shrink as the timescale of the dynamics increases (as bigger jumps are taken between steps). One difference though, is that under these fast timescale dynamics LFADS seems to completely fail to infer the inputs, while the NDT maintains a consistent inferred input magnitude across all time steps where the input was applied. We found that the changes made in the dropout (to improve interpretability) seemed to negatively impact the inferred rates R^2 the most and had the least impact on normalized inferred input magnitude, making it a likely candidate for removal in order to maximize the rate reconstruction performance. The ability of NDTs to infer inputs much better than LFADS is surprising, due to the fact that LFADS has an entire RNN model dedicated to just this task. Inferring inputs in an online setting is not something that has been explored yet, possibly due to the lim-

itations that LFADS faces when trying to infer fast timescale dynamics. The results found here offer an exciting possibility into the online estimation of error signals or perturbations, which may aid in the ability to correct for mistakes made during BCI control.

4.2 Limitations

One limitation is that the inferred inputs are from an attention matrix that can only see backward in time, so there will be a bias for the first and last 14 time steps (size of the backward context) due to the nature of Softmax. The first 14 do not have as big of a context, so the time steps that are seen will have an overinflated importance. The reverse is true for the last 14 time steps, they do not have as many time steps that can attend to them because there is only backward attention, so the importance of these time steps will be over deflated.

Another limitation of the method used for extracting inputs is its inability to extract time-varying inputs or multidimensional inputs. The inferred input extraction takes place via a simple column-wise averaging of the attention matrix. While this extraction procedure is simple, there is a trade-off between the simplicity of the method and the types of inputs can be inferred from the extracted "importance" signal. This inferred input signal may be useful to locate discrete events, such as an error signal, but without any information about the content of the event, it would be hard to interpret what the even was or why it was important.

To successfully apply this technique in vivo spiking data from either humans or animals, the amount of training data needed for this approach to successfully train must also be fully understood. In this study, a rather large (4000 trials) amount of training data was used to follow the procedure outlined in [23]. In real-world situations, you are often limited to a relatively small number of trials (sometimes less than 100), which would necessitate an entirely new approach to training the model as transformers often do not train well with limited data [29]. To understand how the modified NDT scales to different size datasets, one could simply run a similar analysis as performed in this study, but while also sweeping across a suite of training set sizes. The synthetic example used here also only used 50 neurons, while modern BCI implants may

soon have thousands of channels [30]. It is unknown how these techniques used will scale to larger channel counts, and methods such as dimensionality reduction may need to be applied.

4.3 Future Directions

A crucial direction for the future of this work would be the development of methods in which the inferred input could be extracted from the keys, or $\mathbf{K}$ from equation (1.3). These vectors represent the information contained in each time step in high dimensional space, so a successful extraction from them would elevate all limitations that stem from the simplicity of the extraction procedure, and may even give a cleaner signal. This, of course, relies on being able to fit in a supervised fashion, which would mean that known perturbations must be made to the dynamical system of interest. Another important future direction would be an investigation into the circumstances under which the model might fail to train. For example, this might involve training models across a wide range of dataset sizes or number of channels used. Lastly, a successful application of this work in an online setting would involve the use of a sliding window of spiking activity as the input. It is currently unknown what effect this may have on the inferred firing rates and inputs, making this a prime target for future works to explore.

Bibliography

[1] Michael Okun, Nicholas A. Steinmetz, Lee Cossell, M. Florencia Iacaruso, Ho Ko, Péter Barthó, Tirin Moore, Sonja B. Hofer, Thomas D. Mrsic-Flogel, Matteo Carandini, and Kenneth D. Harris. Diverse coupling of neurons to populations in sensory cortex. *Nature*, 521(7553):511–515, 2015.

[2] Krishna V Shenoy, Maneesh Sahani, and Mark M Churchland. Cortical control of arm movements: a dynamical systems perspective. *Annual review of neuroscience*, 36(1):337–359, 2013.

[3] Brett L. Foster, Mohammad Dastjerdi, and Josef Parvizi. Neural populations in human posteromedial cortex display opposing responses during memory and numerical processing. *Proceedings of the National Academy of Sciences*, 109(38):15514–15519, 2012.

[4] Stefano Fusi, Earl K Miller, and Mattia Rigotti. Why neurons mix: high dimensionality for higher cognition. *Current Opinion in Neurobiology*, 37:66–74, 2016. Neurobiology of cognitive behavior.

[5] Anqi Wu, Stan Pashkovski, Sandeep R Datta, and Jonathan W Pillow. Learning a latent manifold of odor representations from neural responses in piriform cortex. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[6] Saurabh Vyas, Matthew D Golub, David Sussillo, and Krishna V Shenoy. Computation through neural population dynamics. *Annual review of neuroscience*, 43:249–275, 2020.

[7] Chethan Pandarinath, K Cora Ames, Abigail A Russo, Ali Farshchian, Lee E Miller, Eva L Dyer, and Jonathan C Kao. Latent factors and dynamics in motor cortex and their application to brain-machine interfaces. *J Neurosci*, 38(44):9390–9401, Oct 2018.

[8] Peiran Gao, Eric Trautmann, Byron Yu, Gopal Santhanam, Stephen Ryu, Krishna Shenoy, and Surya Ganguli. A theory of multineuronal dimensionality, dynamics and measurement. *bioRxiv*, 2017.

[9] Mark M Churchland, John P Cunningham, Matthew T Kaufman, Stephen I Ryu, and Krishna V Shenoy. Cortical preparatory activity: representation of movement or first cog in a dynamical machine? *Neuron*, 68(3):387–400, Nov 2010. PMID: 21040842.

[10] Valerio Mante, David Sussillo, Krishna V. Shenoy, and William T. Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78–84, 2013.

[11] David Sussillo and Larry F Abbott. Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63(4):544–557, 2009.

[12] Kanaka Rajan, Christopher D Harvey, and David W Tank. Recurrent network models of sequence generation and memory. *Neuron*, 90(1):128–142, 2016.

[13] David Sussillo and Omri Barak. Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks. *Neural Computation*, 25(3):626–649, 03 2013.

[14] David Sussillo, Mark M Churchland, Matthew T Kaufman, and Krishna V Shenoy. A neural network that finds a naturalistic solution for the production of muscle activity. *Nature Neuroscience*, 18(7):1025–1033, 2015.

[15] Michael N. Shadlen and William T. Newsome. The variable discharge of cortical neurons: Implications for connectivity, computation, and information coding. *Journal of Neuroscience*, 18(10):3870–3896, 1998.

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

[17] Stephen L Keeley, David M Zoltowski, Mikio C Aoi, and Jonathan W Pillow. Modeling statistical dependencies in multi-region spike train data. *Current opinion in neurobiology*, 65:194–202, Dec 2020.

[18] Max Dabagia, Konrad P Kording, and Eva L Dyer. Comparing high-dimensional neural recordings by aligning their low-dimensional latent representations. *arXiv preprint arXiv:2205.08413*, 2022.

[19] Jakob Macke, Lars Buesing, John Cunningham, Byron Yu, Krishna Shenoy, and Maneesh Sahani. *Empirical models of spiking in neural populations*, volume 24, pages 1350–1358. 01 2011.

[20] Joel Ye and Chethan Pandarinath. Representation learning for neural population activity with neural data transformers. *Neurons, Behavior, Data analysis, and Theory*, 5(3), 2021.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[23] David Sussillo, Rafal Jozefowicz, LF Abbott, and Chethan Pandarinath. Lfads-latent factor analysis via dynamical systems. *arXiv preprint arXiv:1608.06315*, 2016.

[24] Felix Pei, Joel Ye, David Zoltowski, David Zoltowski, Anqi Wu, Raeed Chowdhury, Hansem Sohn, Joseph O' Doherty, Krishna V Shenoy, Matthew Kaufman, Mark Churchland, Mehrdad Jazayeri, Lee Miller, Jonathan Pillow, Il Memming Park, Eva Dyer, and Chethan

Pandarinath. Neural latents benchmark '21: Evaluating latent variable models of neural population activity. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. Curran, 2021.

[25] Joshua I. Glaser, Ari S. Benjamin, Raeed H. Chowdhury, Matthew G. Perich, Lee E. Miller, and Konrad P. Kording. Machine learning for neural decoding. *eNeuro*, 7(4), 2020.

[26] Jakob H Macke, Lars Buesing, John P Cunningham, Byron M Yu, Krishna V Shenoy, and Maneesh Sahani. Empirical models of spiking in neural populations. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[27] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[28] Mohammad Reza Keshtkaran, Andrew R. Sedler, Raeed H. Chowdhury, Raghav Tandon, Diya Basrai, Sarah L. Nguyen, Hansem Sohn, Mehrdad Jazayeri, Lee E. Miller, and Chethan Pandarinath. A large-scale neural network training framework for generalized estimation of single-trial population dynamics. *bioRxiv*, 2021.

[29] Guang Yang, Suhuai Luo, and Peter Greer. A novel vision transformer model for skin cancer classification. *Neural Processing Letters*, 2023.

[30] Yang Wang, Xinze Yang, Xiwen Zhang, Yijun Wang, and Weihua Pei. Implantable intracortical microelectrodes: reviewing the present with a focus on the future. *Microsystems & Nanoengineering*, 9(1):1–4, 2023.