Representation learning for neural population activity with Neural Data Transformers

Joel Ye[1], Chethan Pandarinath[1,2]
[1]Georgia Institute of Technology, [2]Emory University
joel.ye@gatech.edu, chethan@gatech.edu

Abstract

Neural population activity is theorized to reflect an underlying dynamical structure. This structure can be accurately captured using state space models with explicit dynamics, such as those based on recurrent neural networks (RNNs). However, using recurrence to explicitly model dynamics necessitates sequential processing of data, slowing real-time applications such as brain-computer interfaces. Here we introduce the Neural Data Transformer (NDT), a non-recurrent alternative. We test the NDT's ability to capture autonomous dynamical systems by applying it to synthetic datasets with known dynamics and data from monkey motor cortex during a reaching task well-modeled by RNNs. The NDT models these datasets as well as state-of-the-art recurrent models. Further, its non-recurrence enables 3.9ms inference, well within the loop time of real-time applications and more than 6 times faster than recurrent baselines on the monkey reaching dataset. These results suggest that an explicit dynamics model is not necessary to model autonomous neural population dynamics.

Code: github.com/snel-repo/neural-data-transformers.

1. Introduction

Neural populations are theorized to have an underlying dynamical structure which drives the evolution of population activity over time [30, 37, 44]. This structure can be explicitly modeled using linear [12, 19, 28] or switching linear dynamical systems [25, 34], or nonlinear dynamical systems such as recurrent neural networks (RNNs) [31, 33, 36]. In contrast to traditional analyses that average activity across repeated trials of the same behavior, these models have helped relate neural population activity to behavior in individual trials. In particular, an RNN-based method called latent factor analysis via dynamical systems (LFADS) has been shown to model single trial variability in neural spiking activity far better than traditional baselines like spike smoothing or GPFA [22, 31]. This precise modeling enables accurate prediction of subjects' behaviors on a moment-by-moment basis and millisecond timescale.

RNNs have also been used to model language, and have been analogously shown to capture linguistic structure in input sentences [42]. However, with the advent of massive language datasets and their costly training implications, the language modeling community has shifted away from recurrent networks and towards the Transformer architecture [43]. A Transformer receives a sequence of word tokens, or inputs, and processes each individual token in parallel. For example, a Transformer can classify the parts of speech of every word in a sentence simultaneously, whereas an RNN must process earlier words before later ones. A Transformer's parallelism enables it to be trained and operated on sequential data faster than an RNN. Though neuroscience datasets may not yet be large enough to realize much training benefit, reduced inference times could already benefit real-time applications where cycle times are critical, such as brain-computer interfaces or closed-loop neural stimulation.

Here we introduce the Neural Data Transformer (NDT), an architecture for modeling neural population spiking activity. The NDT is based on the BERT encoder [9] with modifications for application to neuroscientific datasets, specifically multi-electrode spiking activity. Modifications are needed as spiking activity has markedly different statistics than both language data and other time series [16, 47] previously modeled by Transformers. Further, neuroscientific datasets are generally much smaller than typical dataset sizes in other machine learning domains, necessitating careful training decisions [17][1].

We test the NDT on synthetic and real datasets to validate its performance. In our synthetic datasets, we generate firing rates using autonomous dynamical systems and sample spikes from the firing rates. We show the NDT can use the sampled spikes to recover the unobserved rates as well as LFADS. Further, when applied to activity recorded from

[1]Negative results such as "difficult training" are under-reported. Our regularization was inspired by discussion in this Twitter thread.

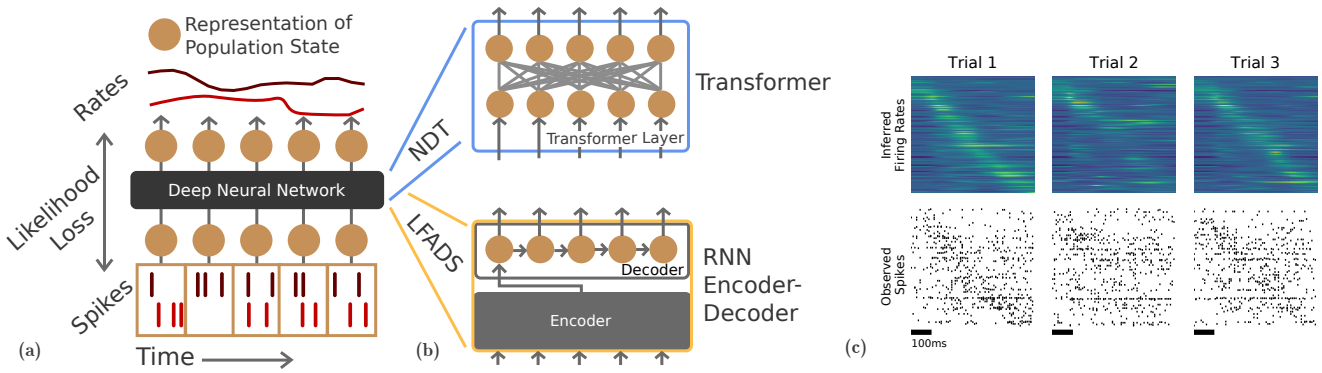

Figure 1. Sequential vs. parallel models. (a) Unsupervised models of sequential spiking activity take in binned spikes (with 2 channels in this schematic) and output inferred rates. A likelihood loss trains the network to output the most likely rates. (b) A Transformer architecture (top) performs parallel modeling, contrasting with RNNs (bottom) and methods like GPFA which use sequential processing. (c) Spike input and inferred rate examples for the NDT applied to the reaching dataset. Rows are sorted by the time of each channel's maximum rate in the first trial, so as to demonstrate correspondence between observed activity and rates.

monkey motor cortex, NDT-inferred firing rates enable prediction of simultaneously measured behavioral variables as well as rates from LFADS. We then demonstrate the NDT's inference efficiency, showing it performs inference in 3.9ms with minimal dependence on sequence length. On the monkey dataset, this enables inference 6.7× faster than LFADS. We also include an ablative study measuring the contributions of different design choices, and consider the tradeoffs of using an NDT with fewer layers.

Our results provide a proof-of-principle that recurrence is not necessary to accurately infer neural population firing rates on a single-trial basis, and unlocks for neuroscience an alternative modeling paradigm that has greatly advanced other fields using machine learning models.

2. The NDT Model

Both the NDT and LFADS transform sequences of binned spiking activity into inferred firing rates (Fig. 1a). In real-time applications, the sequence of spiking activity would come from a rolling window of recent activity that ends with the current timestep. Both models assume a Poisson emission model, meaning inferred rates are compared against the observed spiking activity to compute a Poisson likelihood-based training objective (negative log-likelihood, NLL).

LFADS is a sequential model (Fig. 1b, bottom). In the basic architecture optimized to model autonomous dynamics [31, 40], LFADS first encodes the full input sequence using a bidirectional RNN (not shown), and this encoding is used to set the decoder's initial state. The decoder then evolves its state across timesteps, modeling the dynamics of the neural population. At each timestep, the decoder's state is transformed directly into rate inferences. LFADS' sequential design, i.e., that the decoder must be initialized with an encoding of the full data window, prevents iterative state updates from a stream of individual timesteps in online inference. We discuss challenges to adapting LFADS into an efficient, iterative RNN further in Sec. 3.3, but also discuss benefits of the Transformer beyond speed in Sec. 4. Note also that the challenges in adapting LFADS for iterative inference are further compounded for the more powerful, non-autonomous (input-driven) dynamical architecture [22, 31].

The Transformer avoids sequential bottlenecks by using a stack of layers that process all inputs together (Fig. 1b, top, decomposes one such layer). A Transformer layer comprises several nonlinear sub-layers or blocks (Fig. 2b), in particular a self-attention block in which a new representation of each input is constructed by incorporating relevant information from every other input. We next detail this self-attention block.

The self-attention block is the only one that simultaneously transforms multiple inputs, $x_{[1:T]}$, into multiple outputs, $y_{[1:T]}$. It comprises three different learned weight matrices W^Q, W^K, W^V, termed the query, key, and value weights. All inputs are multiplied by these weight matrices to form three sets of intermediate representations, correspondingly termed the queries, keys, and values $q_{[1:T]}, k_{[1:T]}, v_{[1:T]}$, shown in Fig. 2c. For example, $q_i = W^Q x_i$. For simplicity and to enable stacking of Transformer layers, dimensionality is often fixed throughout the Transformer, *e.g.* given x_i of dimensionality d, W^Q, W^K, W^V are $d \times d$ matrices.

An output y_i is formed by taking a weighted sum of the values, with weights $w_i^{[1:T]}$. Each weight w_i^j represents the "attention" that step i pays to step j (Fig. 2d), and w_i^j is determined by calculating dot-product similarity between query i and key j, and then normalizing similarities over all

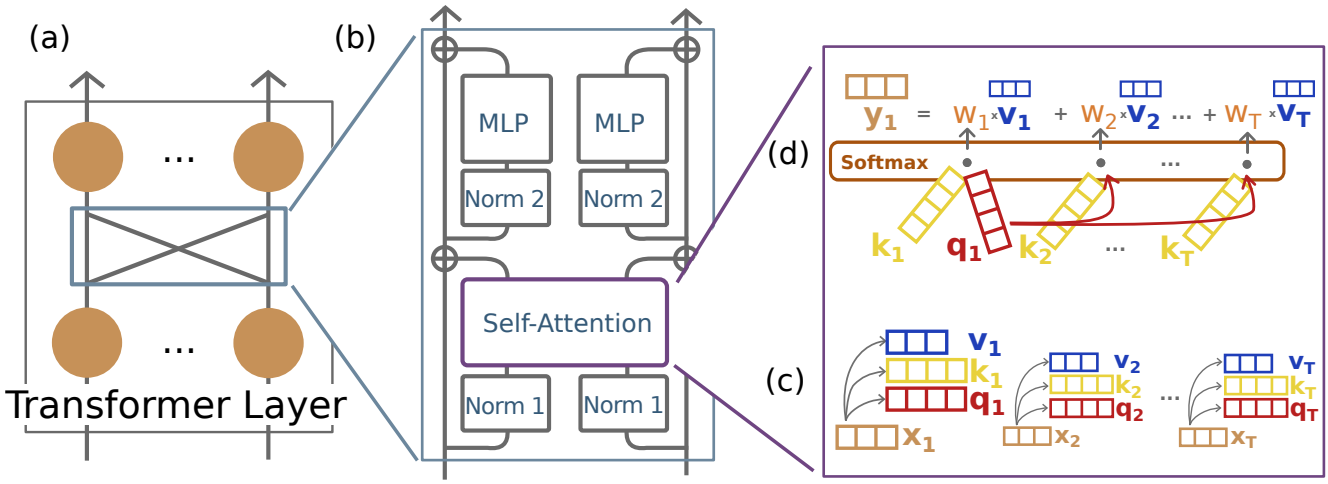

Figure 2. Transformer architecture. (a) A single Transformer layer, as in Fig. 1. The full encoder stacks several of these layers. (b) Inputs to Transformer layers are normalized ("Norm" blocks), enriched through contextual information ("Self-Attention" blocks), and passed through a feedforward module ("MLP" *i.e.* multi-layer perceptron blocks). Blocks with the same label share parameters. The circled plus symbols indicate addition. (c) Inputs at each time step are multiplied by three learned weight matrices (not shown) to create three sets of vectors: the queries, keys, and values. (d) Assembling a single timestep's output: Dot products are computed between the timestep's query and every key, yielding similarity scores. Scores are normalized to sum to 1 to form weights. With these weights, a weighted sum of value vectors are computed and returned as the timestep's output.

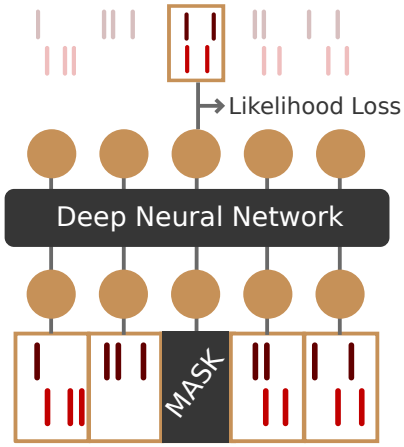

Figure 3. Transformer training. The model is trained with masked modeling [9, 21], that is, model outputs are optimized to maximize likelihood of the masked activity given the context provided by unmasked activity.

j with the softmax function. Formally:

$$y_i = \sum_{j=1}^{T} w_i^j v_j$$
$$p_i^j = q_i \cdot k^j$$
$$w_i^j = \frac{\exp(p_i^j)}{\sum_{l=1}^{T} \exp(p_i^l)}$$

In matrix notation, we can summarize the transformation, from inputs X to outputs Y as:

$$Q = W^Q X, K = W^K X, V = W^V X$$
$$Y = \text{softmax}(QK^T)V$$

It is possible that inputs may want to attend to each other in multiple complementary manners, which may be difficult to capture with a single set of query, key, and value matrices. Consequently, Transformers are often used with multi-headed attention, where each "head" comprises one query, key, and value matrix set. With h heads, we will get h sets of outputs. To preserve dimensionality, we simply concatenate these h outputs and linearly project them to the input shape. While most language Transformers benefit from $h \geq 8$, we do not find multi-headed attention useful in our experiments, so we keep to small values of $h = 1, 2$.

The body of the NDT architecture is a Transformer encoder with 6 layers in most of our experiments, as in Vaswani et al. [43]. We briefly discuss the option to use fewer layers in Sec. 3.3. Before entering the encoder, each channel of the observed activity s_i can be optionally projected to an n-dimensional embedding, *i.e.*, for activity with C channels, the dimension of each input representation is then Cn. To keep dimensionality small, we only consider $n \in \{1, 2\}$ in our experiments. We pass the Transformer encoder outputs through a linear layer and exponentiation (thus treating the linear layer outputs as log-firing rates) before calculating the NLL. Instead of the cross-entropy loss

used in language modeling, these log-firing rates are passed into a Poisson likelihood loss. Full notes on the Transformer encoder are provided in Sec. 7.2.

To train the model in an unsupervised manner, we adapt the masked modeling methodology used in BERT (Fig. 3). In masked modeling, the model is given an input sequence $s_1 \dots s_T$, with a random subset of the T input tokens masked. Subset size is typically a fixed ratio of the full sequence, *e.g.*, 20% of the inputs. The model is then asked to reproduce the original input for that masked subset. To do so, the model must learn how to leverage the context provided by the unmasked timesteps (*e.g.*, if firing rates in the dataset are temporally smooth, high spike counts in unmasked timesteps may imply high spike counts in masked timesteps). Readers familiar with LFADS might note that masked modeling resembles the coordinated dropout method developed to regularize LFADS models [21], only differing in that coordinated dropout masks individual dimensions (channels) of a given input timestep independently and is not constrained to mask entire input timesteps.

We adjust the training procedure as follows:

- In BERT, masked inputs are typically replaced with a special "[MASK]" token. Instead of using this special token, which introduces a large distribution shift between training and inference time [9], we use a "zero mask." That is, we simply zero out the spike inputs of a masked timestep, which was previously demonstrated to be an effective masking strategy for spiking data [21].

- We use intensive regularization to stabilize training, which we find especially important when dataset sizes are smaller. Specifically, in the dropout layers (see Sec. 7.2 for locations), dropout ratios are swept $\in [0.2, 0.6]$.

The importance of the design choices presented here are proven in an ablative study in Sec. 3.4.

3. Results

We compare the NDT with LFADS on both synthetic autonomous dynamics and M1 reaching activity, optimizing hyperparameters (ranges in Sec. 7.4) as follows:

- NDT is optimized using grid search. Using early stopping, we select the checkpoint with least validation NLL as measured without masking.

- LFADS is optimized using the AutoLFADS framework [22]. LFADS is known to benefit from Population-Based Training (PBT [18]) over simple grid search. (We find that NDT performs comparably between grid search and PBT.) AutoLFADS PBT is run with exponentially-smoothed validation NLL as the exploitation metric, and so we select the least smoothed validation NLL checkpoint [22].

Dataset	NDT ($R^2 \uparrow$)	LFADS ($R^2 \uparrow$)
Lorenz	0.934 ± 0.007	0.921 ± 0.009
Chaotic	0.846 ± 0.023	0.869 ± 0.002

Table 1. The NDT and AutoLFADS both infer firing rates that closely align with the generating firing rates on synthetic datasets.

Each search has 20 models. We run three searches for each experiment (a total of $3 * 20 = 60$ models are trained) and report the mean and 95% CI of the metrics achieved. We select our models according to likelihood, since likelihood does not require knowledge of the underlying system and is measurable in both synthetic and real-world settings. The NLL reported is averaged over all bin-channel observations. We apply AutoLFADS with fixed settings that were previously shown to work in a variety of applications [22]. Note that our goal is to use AutoLFADS to provide a baseline for comparison, and we do not exhaustively explore its design choices or alternate hyperparameter ranges to achieve a performance ceiling or minimize training/inference times.

3.1. The NDT achieves high-fidelity inference on synthetic autonomous dynamical systems

We first evaluate the NDT on two synthetic datasets where observed activity reflects autonomous dynamics: the Lorenz system and the chaotic RNN (details in Sec. 7.5). A trial in the Lorenz system [38, 49] is created by simulating a 3D state evolving according to the Lorenz equations, and projecting it to a specified higher dimensionality to form firing rates for a population of synthetic neurons. These rates are sampled according to a Poisson distribution to generate spikes. To create the dataset, we sample several initial states, generate firing rates for each initial state, and sample several trials of spiking activity for each set of firing rates. We denote trials with the same initial state as having the same condition. Similarly, the chaotic RNN dataset [40] is created by simulating dynamics using a vanilla RNN whose weights are initialized from the normal distribution. This system is motivated by the fact that many neural datasets are well modeled by RNNs (which are themselves nonlinear dynamical systems). The chaotic RNN is more complex than the Lorenz system - as measured by the number of principal components underlying the generating system - and is thus more challenging to model. As with the Lorenz dataset, conditions in the chaotic RNN dataset correspond to distinct initializations of the nonlinear dynamical system.

The synthetic setting allows us to evaluate inferred firing

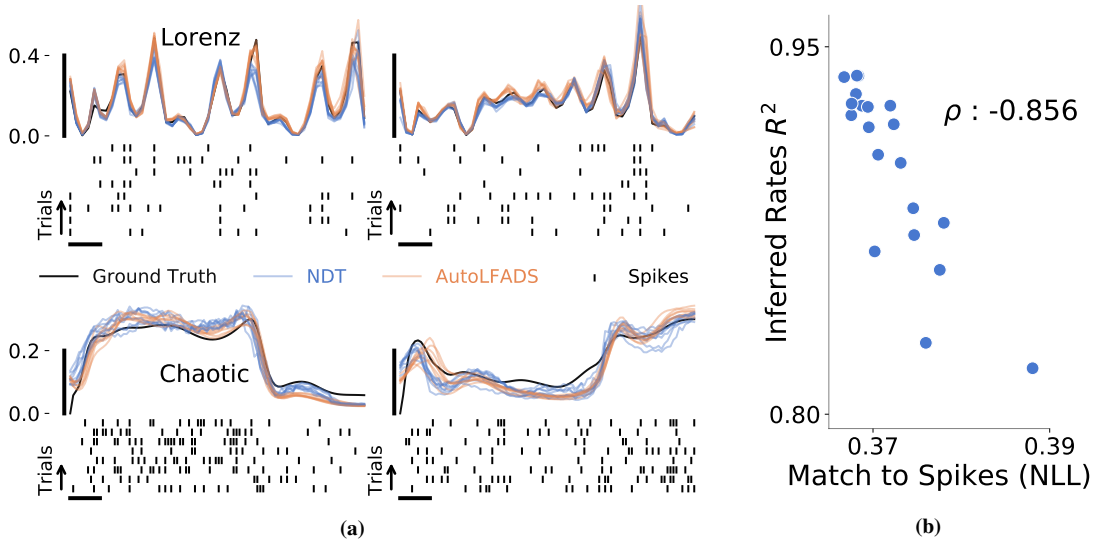

Figure 4. Modeling synthetic data. (a) In each quadrant, we plot the ground truth firing rate for a sampled neuron and information from 8 of its trials. We refer to these trials with the same initial conditions (and thus firing rates) as having the same condition and show 2 conditions (columns) for each of the synthetic datasets (rows). For these trials, we show the generated spikes (bottom) and inferred rates from AutoLFADS and NDT (top). Inferred firing rates closely match generating ground truth rates. Vertical bar denotes spikes per bin. Horizontal bar indicates 10% of the trial length, 5 bins for Lorenz and 10 bins for Chaotic RNN. (b) Across a hyperparameter sweep on the Lorenz dataset, models that achieve better likelihoods yield more accurate inference of the underlying rates. NLL is averaged across bins and channels.

rates by comparing against the ground truth rates that produced the synthetic spikes. In both datasets, NDT and AutoLFADS inferences closely match the ground truth, though NDT rates appear less smooth (Fig. 4a). We quantify model inference quality by measuring the correspondence between inferred and ground truth firing rates using the coefficient of determination (R^2 ; Tab. 1). In both datasets, the gap between the two models is small, indicating the NDT can accurately infer firing rates in autonomous dynamical systems. Importantly, we also find that within an HP search, NDT models with high data likelihoods (as computed on the observed spiking activity) tend to match the underlying systems well (as measured by correspondence with ground truth firing rates, Fig. 4b). This match between likelihoods and firing rate inference does not occur in LFADS models that lack coordinated dropout [21] and provides a key confirmation that the NDT's masking strategy works as desired. Verifying that likelihood correlates with recovery of underlying structure in synthetic data provides confidence that likelihood can be used to optimize and choose between NDT models in applications to real-world data.

3.2. NDT infers motor cortical firing rates in autonomous settings with high fidelity

To test performance in real-world neural recordings, we apply NDT to the Monkey J Maze dataset [20]. These data were previously used to evaluate LFADS and AutoLFADS [21, 22, 31] and serve as a benchmark for models of autonomous dynamics. In this dataset, spiking activity from 202 neurons in the primary motor and dorsal premotor cortices was recorded as a monkey performed a delayed reaching task with a variety of straight and curved reaches. The reaching dataset consists of 2296 trials across 108 different reach conditions, where a given condition is specified by targets and obstacles present. Each trial has a random delay period that separates target presentation from a "Go" cue that prompts the monkey to begin its reach, which provides a time period for the monkey to plan before executing the reach. Previous analyses of this paradigm demonstrated that neural activity is well modeled as an autonomous dynamical system, where plan activity serves as an initial state that predicts the activity patterns observed during movement execution [7, 31, 37]. We train our models on activity during this autonomous period, spanning 250 ms before movement onset to 450 ms after. We perform most experiments by binning the spike sequences at 10ms ; we find similar results for bin sizes varying from 2ms to 20ms (results not shown).

We compute peri-stimulus time histograms (PSTHs) for the models by averaging inferred rates across repeated trials of the same reach condition (Fig. 5a). Both NDT and AutoLFADS exhibit low across-trial variance (as shown by the shaded errorbars), indicating that the models produce consistent inferred rates for different trials of the same condition. We also calculate a spike smoothing baseline by

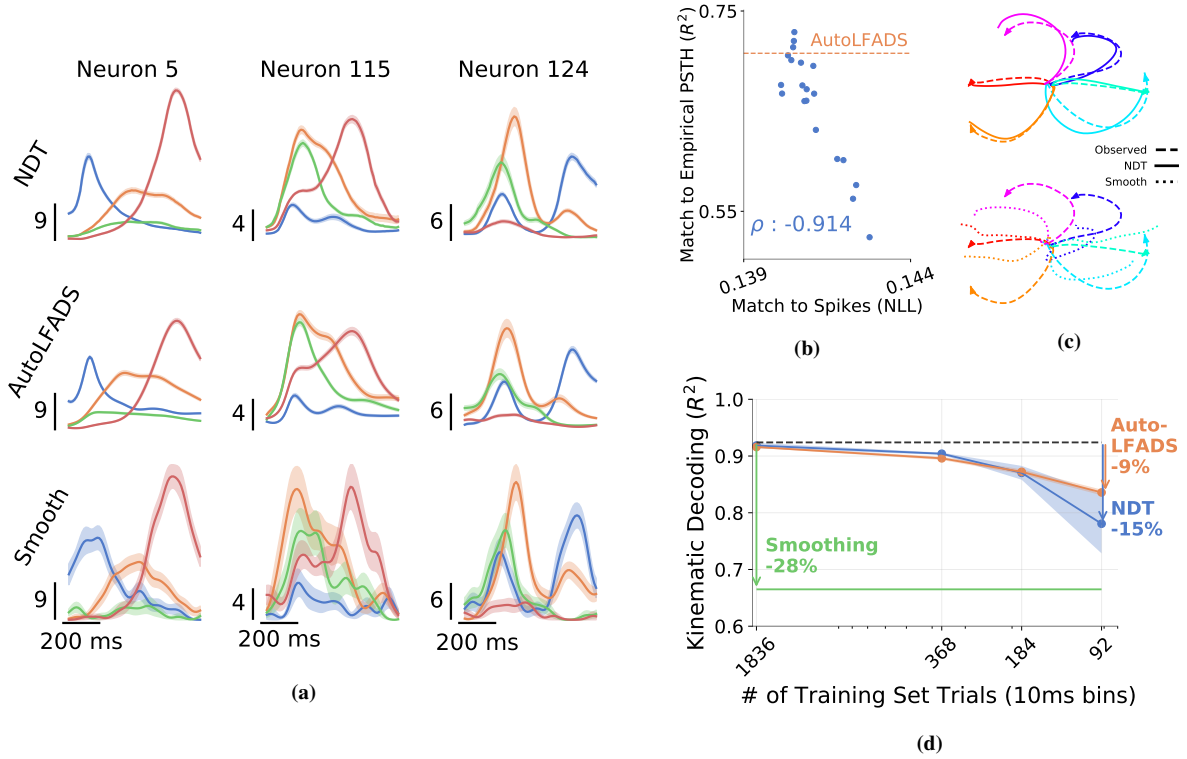

Figure 5. NDT inference on monkey reaching data. (a) Trial-averaged inferred rates and smoothed activity (4 of 108 conditions shown). Vertical bar denotes spikes/sec. Shading indicates SEM. (b) Across a hyperparameter sweep, models that converge to better likelihoods tend to achieve better match to empirical PSTHs. The NLL reported is averaged over the NLL of all bin-channel observations. (c) Example predicted reach trajectories based on NDT-inferred rates (dashed) align more closely with the true trajectories (solid) than predictions based on smoothed spikes (dotted). (d) Performance on kinematic decoding with smaller training sets still greatly improves on spike smoothing down to 92 training trials. Shading indicates 95% confidence interval across three sweeps.

first passing observed spiking activity through a Gaussian kernel with 30 ms standard deviation, and then averaging across trials to form empirical PSTHs (Fig. 5a, bottom). These exhibit larger across-trial variance than the model-inferred firing rates, as spike smoothing produces noisy estimates on single trials [31]. To quantify the quality of the models' inferred rates, we measure the correspondence between inferred PSTHs and empirical PSTHs. NDT models with greater likelihoods tend to have better R^2 (Fig. 5b). The highest-performing models perform on par with AutoLFADS.

For motor cortical datasets, another method to evaluate the quality of inferred firing rates is through behavioral decoding, *i.e.*, testing how well simultaneously-recorded behavioral variables can be decoded from the models' inferred rates. We use optimal linear estimation to map firing rates onto hand velocities (details in Sec. 7.6) and find that NDT enables accurate behavioral decoding that matches AutoLFADS (0.918 and 0.915 R^2, respectively). These velocity predictions can be integrated to produce predicted reaching trajectories (Fig. 5c). The large number of trials (2000) also allows us to evaluate each model's sensitivity to dataset size by subsampling from the full dataset. NDT comfortably outperforms the spike smoothing baseline, even when scaling to as few as 92 training trials (Fig. 5d). While a 6-layer NDT performs worse than AutoLFADS at 92 trials, we show that a 2-layer NDT closes the gap in Sec. 3.3.1.

	Train Time	Inference Time	# Parameters	Reaching R^2
AutoLFADS	45m	26ms	280K	0.915
NDT-6	9.4hr	3.9ms	1.36M	0.918
NDT-2	45m	2.2ms	480K	0.921
NDT-1	20m	0.98ms	270K	0.886

Table 2. Speed gains from reducing model size. On one sweep for each variant, we report the training time of the best model, inference time on 70 bins, number of trainable parameters, and kinematic decoding performance.

3.3. Efficiency Gains from Parallelism

Inference Speeds. In neural prosthetic applications such as brain-machine interfaces (BMIs) or closed-loop neural

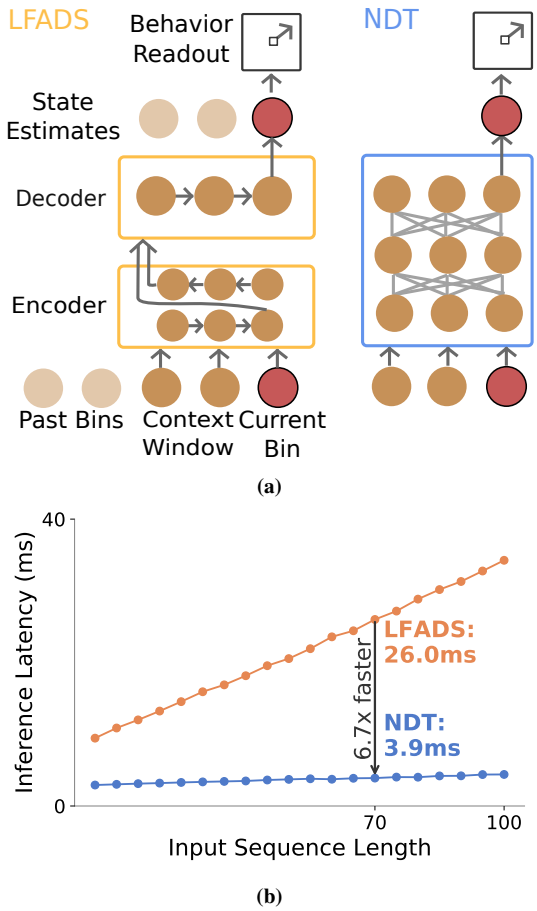

Figure 6. (a) Schematic of LFADS (simplified) and NDT performing online inference. (b) Sequential vs. parallel runtime: Recurrent architectures, such as LFADS, have inference times that grow with input sequence length. The NDT has near-constant runtime. In the reaching dataset with input sequences of length 70, the NDT infers 6.7× faster than LFADS.

stimulation, neural population states are used to guide actions, such as steering a computer cursor or robotic arm, or deciding whether or how to stimulate the brain. In these settings, where neural activity is read-in and acted upon in real-time, even delays on the scale of milliseconds can decrease the quality of closed-loop control [4, 5, 8, 13, 15]. Thus, inference latencies present a critical barrier to translating computationally-intensive methods for neural state estimation to real-time applications.

We reasoned that the NDT could enable substantial speedups in neural population state inference relative to LFADS, due to the latter architecture's reliance on sequential processing. LFADS and NDT both require a span of observed activity to infer the neural population state at the current time step. To minimize latency in real-time applications, this context would come from a rolling window of observed data (Fig. 6a). However, despite the overlap in consecutive windows, both models would need to process each window from scratch. LFADS requires a bidirectional reading of the entire input sequence to initialize its dynamics model, which is then run forward for the full input length to infer the state at the current time step. Note that a new initial condition and subsequent unrolling may not be necessary in autonomous settings (which we use in this work) as activity beyond a certain length into the future should not inform the initial state. However, in real-time, non-autonomous applications, LFADS requires the full context for its controller to make precise interventions in the unrolled dynamics [31, 40]. Similarly, for a multi-layer NDT, most computations would need to be performed anew to take into account the most recent data, and thus we would not expect a large benefit from maintaining state across windows. Consequently, we can compare inference speeds in the offline setting and expect trends to hold in online translation.

We measured the inference time for each model as a function of sequence length. Since the NDT models a given input sequence in parallel, we should expect a roughly constant inference speed with respect to input sequence length. The NDT's non-recurrence enables 3.9ms inference (Fig. 6b, with details in Sec. 7.7), comfortably within the loop time of many real-time applications. In practice, we find the NDT's inference times increase slightly with increased bin lengths; in contrast, LFADS inference times increase substantially. In the reaching dataset with sequence length 70, this amounts to a 6.7x speedup. Thus if a recording module reported activity every $x = 10$ms, an NDT-based BMI could reflect the latest activity in $10+4 = 14$ms, whereas an LFADS-based BMI would use $10+26 = 36$ms. Smaller x yields further benefits. For reference, prior work that achieved high-performing online decoding uses windows with 20 bins of 15ms [19]; even with this reduced bin count our method provides a 4x speedup.

For completeness, we note that it may be possible to speed up a recurrent architecture like LFADS if it could maintain state across windows, as is done in traditional iterative state space models such as Kalman filters. The changed model would then only need to integrate one new bin of information, which should reduce the required inference time. However, to our knowledge, such an approach has not yet been demonstrated, suggesting that training such an iterative model is non-trivial. For example, LFADS' bidirectional encoding, which precludes iterative updates, is a design choice empirically needed for good performance: in unpublished tests, we find that forward-only encoding diminishes performance.

3.3.1 Smaller NDTs Improve Training Speed and Data Efficiency

The fixed computational complexity of the NDT's parallel architecture should grant faster training in addition to inference [43]. The 6-layer NDT used in previous experiments, however, does train for significantly longer than our LFADS model (Tab. 2). We note that training times of different models across an HP search can vary widely, *i.e.*, we see NDT 6-layer times between 3 and 18 hours. However, training times can be reduced substantially by simply using a smaller NDT. We find a 1-layer NDT, with around the same number of parameters as our LFADS model, trains under 30 minutes (and infers in under 1ms). Remarkably, this 1-layer NDT achieves 0.89 R^2 on kinematics decoding in the Maze dataset (-0.02 R^2 against the 6-layer baseline), and a 2-layer NDT matches the 6-layer performance. Note that the shallower NDTs train faster than LFADS again due to parallelism, as parallelism avoids the costly backpropagation through time used to train recurrent networks. In our case, the 6-layer NDT was much larger than the AutoLFADS model; AutoLFADS training times are more appropriately compared with the 2 or 1-layer NDT.

Smaller models may also be more performant in limited data settings. The 2-layer model achieves 0.866 R^2 when training on just 92 trials (not shown), outperforming the AutoLFADS model. Regularization is still critical for the smaller 2-layer model: performance drops to 0.4 R^2 when the dropout range is confined to $[0.0, 0.3]$ instead of $[0.2, 0.6]$ (not shown). Though non-exhaustive, this result indicates that the gap between AutoLFADS and NDT when limited to 92 trials (Fig. 5d) may be due to 6-layer models being oversized. Extrapolating beyond this dataset, neural datasets, though smaller than in other domains, may be well-modeled by Transformers so long as the models are appropriately scaled.

3.4. Ablative Analysis

We empirically justify three key design choices of the 6-layer NDT by removing them and evaluating the degraded performance on the reaching dataset. Each is critical to achieving high performance (Tab. 3): without these subtle choices, performance is much worse and more variable. For example, models that infer rates instead of lograte train more slowly and fail to converge to a good solution over a wide set of hyperparameters (Fig. 7). Notably, inferring rates instead of lograte regresses performance in both 2-layer models and 6-layer models, *i.e.*, constraining models to output in log space is important even with increased capacity. This suggests the NDT cannot learn to exponentiate well, even under long training regimes; this contrasts with findings in [12] where feedforward networks improve on exponential nonlinearities for fitting linear dynamical systems on neural population activity. We also experimented with a few training details relevant in other Transformer works, such as variable length mask spans or adding embedding layers, but found their contributions on the reaching dataset to be marginal, on the order of 1-2% R^2 .

	R^2 (↑)
Baseline	0.918 ± 0.005
Output rates, not lograte	0.292 ± 0.203
Use "[MASK]" tokens, not zeroes	0.022 ± 0.024
Constrain dropout $\in [0.0, 0.2]$	0.549 ± 0.103

Table 3. The use of lograte, zero masks, and heavy regularization are all critical to the NDT's performance.

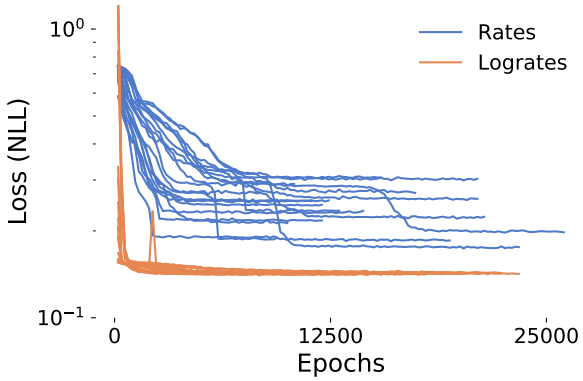

Figure 7. Lograte vs. rate inference. We plot loss trajectories of two hyperparameter sweeps of size 20. One sweep uses firing lograte (model output is exponentiated before calculating loss) and the other uses firing rates. When using rates, models train less stably and converge to poor solutions.

4. Discussion

We have introduced the NDT, a parallel neural network architecture for neural spiking activity, and shown it can be competitive with RNNs in autonomous dynamical settings while achieving substantially faster inference. Further, with careful architecture choices, the NDT could even match RNN performance on datasets with as few as 92 training trials (0.2 Mb). This indicates that Transformers are compatible with dataset sizes that are typically available in systems neuroscience.

The most critical limitation of the NDT, and thus an important avenue for future work, is its inability to model non-autonomous dynamics, *i.e.* systems with unpredictable external perturbations. This occurs when unmonitored brain areas send signals to the recorded area. For example, unpredictable experiment cues that are first processed in so-

matosensory or visual areas will propagate and perturb the dynamics of recorded motor areas. LFADS, which explicitly models inputs for such non-autonomous settings, outperforms NDT by over 20% R^2 in a preliminary experiment with a synthetic, non-autonomous dataset, the Chaotic RNN with Inputs studied in Sussillo et al. [40]. Enabling the NDT to learn sample-efficiently in such non-autonomous settings would likely require architectural modifications to explicitly model both dynamics and inputs.

Despite this limitation, we put forth the NDT as a forward-looking proposal. We believe the NDT and more generally the Transformer can benefit neuroscience due to the Transformer's rapid rise in the broader machine learning community. This broader community has advanced and will continue to advance Transformer tooling, analysis, and theory; we provide an overview of recent directions in Sec. 7.3. Many of these advances could translate to neuroscientific applications. We provide two such examples:

- Story generation requires modeling of both sensible short-term sentence structure and a coherent long-term storyline. While RNNs struggle to learn long-term dependencies, the Transformer's parallel design makes it less biased with respect to either short or long term dependencies. This enables the Transformer to produce long passages of coherent text [35]. Analogously, a single Transformer model may yield insights around both fast and slow features of neural activity, uncovering hierarchy within the activity that maps naturally to the multi-scale nature of animal behavior [2, 3].

- Transformers have been productively used to understand the interaction of data from multiple modalities. For example, vision-language transformers [27] produce language representations that are contextualized by accompanying images. Similar techniques could be applied to build models which incorporate recordings of multiple brain areas, different recording modalities, and behavioral measurements.

However, the major driver of the Transformer's popularity is its ability to scale to large amounts of training data better than RNNs (*i.e.* through faster training). As increasing training data generally improves machine learning models across domains, we anticipate that larger datasets from new recording technologies and dataset aggregation will further improve the NDT's performance and applicability, possibly past recurrent methods. Notably, these large datasets need not be excessively difficult to collect. For example, they could consist of neural activity that is continuously collected without constrained or even measured behavior. In other domains, the largest datasets tend to be similarly unstructured, naturally-occurring data, such as freeform text extracted from the internet. In a large-scale "pretraining" step, networks can learn deep representations of such data in a self-supervised manner, using methods such as those used to train LFADS and NDT. Pretrained representations make subsequent learning for downstream tasks much more data-efficient. The seeming universality of the representations learned in these tasks, for example, has prompted the GLUE language benchmark [45] to assess how well single models perform on 9 different language tasks. An analogous effort in neuroscience may help reveal all the different computational roles of a given neural population, much as prior work has sought to find preferential tuning properties for single neurons.

One promising avenue in the analysis of trained RNNs is the application of techniques from nonlinear dynamical systems theory to interrogate the RNNs' learned dynamical structure [14, 29, 38, 39]. The Transformer is currently disconnected from these dynamical techniques, as it lacks a recurrent structure to analyze. It would be useful, even beyond the computational neuroscience community, to try to bridge this gap and understand how the Transformer represents dynamical structure.

5. Acknowledgements

We thank Andrew Sedler, Yahia Ali, and Ruyi Marone for their insights and conversations. We also thank Krishna Shenoy, Mark Churchland, Matt Kaufman, and Stephen Ryu for sharing the Monkey J Maze dataset. This work was supported by the Emory Neuromodulation and Technology Innovation Center (ENTICe), NSF NCS 1835364, DARPA PA-18-02-04-INI-FP-021, NIH Eunice Kennedy Shriver NICHD K12HD073945, the Alfred P. Sloan Foundation, and the Simons Foundation as part of the Simons-Emory International Consortium on Motor Control (CP). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government, or any sponsor.

6. Author Contributions

JY and CP jointly contributed towards conceptualization, writing, and revision. JY was responsible for investigation and software, and CP was responsible for funding acquisition and resources.

7. Methods

7.1. Data Availability

The Lorenz dataset, along with generation scripts for the Chaotic RNN dataset, are available in the code repo. The Maze dataset will be released upon publication.

7.2. Architectural Details

We provide additional information about the Transformer encoder and self-attention mechanism [23, 43] for more details. Inputs to a Transformer layer pass through a self-attention block, a layer norm block [1], an MLP, and another layer norm.

Attention Scaling. In the original Transformer [43], the authors found scaling the dot product by a factor $\sqrt{d}$, where d is input dimensionality, improves learning. That is:

$$Y = \text{softmax}(QK^T)V \text{ becomes}$$
$$Y = \text{softmax}(QK^T/\sqrt{d})V$$

Per the implementation of attention available through Pytorch, we maintain this scaling.

Position Embeddings. Self-attention lets inputs query for relevant information from other inputs. However, if we directly feed population representations, inputs would be unable to query for information from a particular timestep, *i.e.* there is no intrinsic ordering of the inputs. This is inappropriate in most cases, including ours. To account for input order, we add a learned position embedding (*i.e.* a unique vector representing the identity of the input timestep) to each input before it is fed into the transformer layers.

The rest of the Transformer layer. Following self-attention, we have layer normalization and an MLP. Each layer normalization block receives a single population state vector as input and normalizes this input using the mean and variance of its elements. The MLP comprises 2 linear layers joined with a non-linear ReLU activation, and similarly transforms a single input to a single output. Dropout layers are added right before the inputs enter the transformer body (the consecutive transformer layers), right after they exit the body, and right after each linear layer in the MLP of each transformer layer.

7.3. Further Directions in Transformer Literature.

The Transformer's empirical success has prompted several avenues of further research. One direction, which this work falls under, is the application of the Transformer to new domains, such as for protein folding [10] or reinforcement learning [32]. Other work, often in natural language processing, focuses on improving Transformer performance along some axis. Task metrics can be improved with data engineering and modifying training curricula [26], while memory and compute efficiency can be improved with both architectural changes and post-training procedures like pruning or distillation [11, 41]. The Transformer's attention mechanism has also been applied generally in graph neural networks [48].

A final direction of interest is Transformer interpretability. Transformer models can be analyzed in much the same, possibly contentious, ways as other deep models, through probes of model activations [42] or through gradient-baseds visualization [6]. Uniquely, the Transformer is often interpreted with respect to its attention weights. These weights are often intuitive, *e.g.* in language, when an adjective's attention to the preceding contextualizing adverb is high. However, to date there has been no widely accepted theoretical reasoning for how to interpret such attention, and so attention-based analysis is often debated [46].

7.4. Hyperparameters

NDT searches are swept over:

- Dropout ratio, as described in Sec. 3.
- Context span, the number of timesteps forward and backward each input aggregates information from. Span is swept between 4 and 32 steps in both directions for the synthetic datasets, and 10 and 50 in the reaching datasets.
- The ratio of masked tokens that are replaced with a random input instead of a zero mask, and the ratio that are not replaced at all (a methodology from BERT to reduce train-test distribution shift). Zero mask ratio is between 0.5 and 1.0 on synthetic datasets, and 0.6 and 1.0 on the reaching dataset. Of the remaining masked tokens, between 0.9 and 1.0 are replaced with random inputs on synthetic datasets, between 0.6 and 1.0 on the reaching dataset.
- Length of masked span [24] is set between 1 and 5 in synthetic datasets, and 1 and 7 in reaching dataset.

AutoLFADS PBT optimizes over:

- Dropout, from 0.0 to 0.6
- Coordinated Dropout [21] rate, from 0.01 to 0.7
- L^2 penalties for the generator from 1e-4 to 1.0
- KL penalties for the initial condition from 1e-5 to 1e-3

Both models optimize learning rate, from 1e-5 to 5e-3. The LFADS controller is kept off as we study autonomous settings. Note that although we find these AutoLFADS settings outperform the ranges reported in [22], we only claim they are sufficient and not necessary for achieving reported results. PBT settings such as early stopping metrics and epochs per generation are as in [22]. Other hyperparameters are available in the code.

7.5. Synthetic Dataset

The train-val split is 0.8 and 0.2 for each dataset. The Lorenz dataset has 1560 total trials, 50 timesteps, and 29 channels. These trials comprise 65 conditions (firing rate trajectories) with 24 trials sampled per condition. The chaotic RNN dataset is generated with $\gamma = 1.5$ and has 1300 total trials (with 100 conditions and 13 trials per condition), 100 timesteps, and 50 channels. R^2 is calculated by flattening timesteps and trials, and averaging across input channels, as done in [22].

7.6. Kinematic Decoding

We decode 2D hand velocity from inferred rates at single timesteps using ridge regression with $\alpha = 0.01$. As in [31], we find improved decoding performance by applying a 90 ms lag between neural activity and the corresponding kinematics, i.e., while rates are inferred between a (-250ms, 450ms) window around movement onset, kinematics are predicted only around (-160ms, 450ms).

7.7. Timing Tests

We report the time of a forward pass through the NDT and LFADS models, i.e. the time it takes to infer rates from spike inputs. 1 posterior sample is used for LFADS. Times are averaged over 1300 trials. Measurements were taken on a machine (on CPU) with 32GB RAM and a 4-core i7-4790K processor running at 4.2 GHz.

References

[1] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization, 2016.

[2] G. J. Berman. Measuring behavior across scales. *BMC Biology*, 16(1):23, Dec. 2018. ISSN 1741-7007. doi: 10.1186/s12915-018-0494-7. URL https://bmcbiol.biomedcentral.com/articles/10.1186/s12915-018-0494-7.

[3] G. J. Berman, W. Bialek, and J. W. Shaevitz. Predictability and hierarchy in drosophila behavior. *Proceedings of the National Academy of Sciences*, 113(42):11943–11948, 2016. ISSN 0027-8424. doi: 10.1073/pnas.1607601113. URL https://www.pnas.org/content/113/42/11943.

[4] M. F. Bolus, A. A. Willats, C. J. Whitmire, C. J. Rozell, and G. B. Stanley. Design strategies for dynamic closed-loop optogenetic neurocontrol *in vivo*. *Journal of Neural Engineering*, 15(2):026011, Apr. 2018. ISSN 1741-2560, 1741-2552. doi: 10.1088/1741-2552/aaa506. URL https://iopscience.iop.org/article/10.1088/1741-2552/aaa506.

[5] M. F. Bolus, A. A. Willats, C. J. Rozell, and G. B. Stanley. State-space optimal feedback control of optogenetically driven neural activity. *bioRxiv*, 2020. doi: 10.1101/2020.06.25.171785. URL https://www.biorxiv.org/content/early/2020/08/25/2020.06.25.171785.

[6] H. Chefer, S. Gur, and L. Wolf. Transformer interpretability beyond attention visualization. *arXiv preprint arXiv:2012.09838*, 2020.

[7] M. M. Churchland, J. P. Cunningham, M. T. Kaufman, J. D. Foster, P. Nuyujukian, S. I. Ryu, and K. V. Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405): 51–56, July 2012. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature11129. URL http://www.nature.com/articles/nature11129.

[8] J. P. Cunningham, P. Nuyujukian, V. Gilja, C. A. Chestek, S. I. Ryu, and K. V. Shenoy. A closed-loop human simulator for investigating the role of feedback control in brain-machine interfaces. *Journal of Neurophysiology*, 105(4): 1932–1949, Oct. 2010. ISSN 0022-3077. doi: 10.1152/jn.00503.2010. URL https://journals.physiology.org/doi/full/10.1152/jn.00503.2010.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. 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. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

[10] I. Drori, D. Thaker, A. Srivatsa, D. Jeong, Y. Wang, L. Nan, F. Wu, D. Leggas, J. Lei, W. Lu, W. Fu, Y. Gao, S. Karri, A. Kannan, A. Moretti, M. AlQuraishi, C. Keasar, and I. Pe'er. Accurate protein structure prediction by embeddings and deep learning representations, 2019.

[11] Q. Fournier, G. M. Caron, and D. Aloise. A practical survey on faster and lighter transformers, 2021.

[12] Y. Gao, E. W. Archer, L. Paninski, and J. P. Cunningham. Linear dynamical neural population models through nonlinear embeddings. *Advances in neural information processing systems*, 29:163–171, 2016.

[13] V. Gilja, C. Pandarinath, C. H. Blabe, P. Nuyujukian, J. D. Simeral, A. A. Sarma, B. L. Sorice, J. A. Perge, B. Jarosiewicz, L. R. Hochberg, K. V. Shenoy, and J. M. Henderson. Clinical translation of a high-performance neural prosthesis. *Nature Medicine*, 21(10):1142–1145, Oct. 2015. ISSN 1546-170X. doi: 10.1038/nm.3953. URL https://www.nature.com/articles/nm.3953.

[14] M. D. Golub and D. Sussillo. Fixedpointfinder: A tensorflow toolbox for identifying and characterizing fixed points in recurrent neural networks. *Journal of Open Source Software*, 3(31):1003, 2018. doi: 10.21105/joss.01003. URL https://doi.org/10.21105/joss.01003.

[15] L. Grosenick, J. H. Marshel, and K. Deisseroth. Closed-Loop and Activity-Guided Optogenetic Control. *Neuron*, 86 (1):106–139, Apr. 2015. ISSN 0896-6273. doi: 10.1016/j.neuron.2015.03.034. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4775736/.

[16] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck. Music transformer, 2018.

[17] X. S. Huang, F. Perez, J. Ba, and M. Volkovs. Improving transformer optimization through better initialization. In *In-*

ternational Conference on Machine Learning, pages 4475–4483. PMLR, 2020.

[18] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, C. Fernando, and K. Kavukcuoglu. Population based training of neural networks, 2017.

[19] J. C. Kao, P. Nuyujukian, S. I. Ryu, M. M. Churchland, J. P. Cunningham, and K. V. Shenoy. Single-trial dynamics of motor cortex and their applications to brain-machine interfaces. *Nature Communications*, 6(1):7759, July 2015. ISSN 2041-1723. doi: 10.1038/ncomms8759. URL https://www.nature.com/articles/ncomms8759.

[20] M. T. Kaufman, J. S. Seely, D. Sussillo, S. I. Ryu, K. V. Shenoy, and M. M. Churchland. The Largest Response Component in the Motor Cortex Reflects Movement Timing but Not Movement Type. *eNeuro*, 3(4), Aug. 2016. ISSN 2373-2822. doi: 10.1523/ENEURO.0085-16.2016. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5069299/.

[21] M. R. Keshtkaran and C. Pandarinath. Enabling hyperparameter optimization in sequential autoencoders for spiking neural data. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 15937–15947. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/6948bd44c91acd2b54ecdd1b132f10fb-Paper.pdf.

[22] M. R. Keshtkaran, A. R. Sedler, R. H. Chowdhury, R. Tandon, D. Basrai, S. L. Nguyen, H. Sohn, M. Jazayeri, L. E. Miller, and C. Pandarinath. A large-scale neural network training framework for generalized estimation of single-trial population dynamics. *bioRxiv*, 2021. doi: 10.1101/2021.01.13.426570. URL https://www.biorxiv.org/content/early/2021/01/15/2021.01.13.426570.

[23] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017. doi: 10.18653/v1/P17-4012. URL https://doi.org/10.18653/v1/P17-4012.

[24] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://www.aclweb.org/anthology/2020.acl-main.703.

[25] S. Linderman, M. Johnson, A. Miller, R. Adams, D. Blei, and L. Paninski. Bayesian learning and inference in recurrent switching linear dynamical systems. In *Artificial Intelligence and Statistics*, pages 914–922, 2017.

[26] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[27] J. Lu, D. Batra, D. Parikh, and S. Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 13–23. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/c74d97b01eae257e44aa9d5bade97baf-Paper.pdf.

[28] J. H. Macke, L. Buesing, J. P. Cunningham, B. M. Yu, K. V. Shenoy, and M. 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, pages 1350–1358. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper/2011/file/7143d7fbadfa4693b9eec507d9d37443-Paper.pdf.

[29] N. Maheswaranathan, A. Williams, M. Golub, S. Ganguli, and D. Sussillo. Reverse engineering recurrent networks for sentiment classification reveals line attractor dynamics. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 15696–15705. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/d921c3c762b1522c475ac8fc0811bb0f-Paper.pdf.

[30] C. Pandarinath, K. C. Ames, A. A. Russo, A. Farshchian, L. E. Miller, E. L. Dyer, and J. C. Kao. Latent factors and dynamics in motor cortex and their application to brain–machine interfaces. *Journal of Neuroscience*, 38(44):9390–9401, 2018.

[31] C. Pandarinath, D. J. O'Shea, J. Collins, R. Jozefowicz, S. D. Stavisky, J. C. Kao, E. M. Trautmann, M. T. Kaufman, S. I. Ryu, L. R. Hochberg, J. M. Henderson, K. V. Shenoy, L. F. Abbott, and D. Sussillo. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature Methods*, 15(10):805–815, Oct. 2018. ISSN 1548-7091, 1548-7105. doi: 10.1038/s41592-018-0109-9. URL http://www.nature.com/articles/s41592-018-0109-9.

[32] E. Parisotto, H. F. Song, J. W. Rae, R. Pascanu, C. Gulcehre, S. M. Jayakumar, M. Jaderberg, R. L. Kaufman, A. Clark, S. Noury, M. M. Botvinick, N. Heess, and R. Hadsell. Stabilizing transformers for reinforcement learning, 2019.

[33] M. G. Perich, C. Arlt, S. Soares, M. E. Young, C. P. Mosher, J. Minxha, E. Carter, U. Rutishauser, P. H. Rudebeck, C. D. Harvey, et al. Inferring brain-wide interactions using data-constrained recurrent neural network models. *bioRxiv*, 2020.

[34] B. Petreska, M. Y. Byron, J. P. Cunningham, G. Santhanam, S. I. Ryu, K. V. Shenoy, and M. Sahani. Dynamical segmentation of single trials from population neural data. In *Advances in neural information processing systems*, pages 756–764, 2011.

[35] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

[36] Q. She and A. Wu. Neural dynamics discovery via gaussian process recurrent neural networks. In *Uncertainty in Artificial Intelligence*, pages 454–464. PMLR, 2020.

[37] K. V. Shenoy, M. Sahani, and M. M. Churchland. Cortical Control of Arm Movements: A Dynamical Systems Perspective. *Annual Review of Neuroscience*, 36(1):337–359, July 2013. ISSN 0147-006X, 1545-4126. doi: 10.1146/annurev-neuro-062111-150509. URL http://www.annualreviews.org/doi/10.1146/annurev-neuro-

062111-150509.

[38] D. Sussillo. Neural circuits as computational dynamical systems. *Current Opinion in Neurobiology*, 25:156–163, Apr. 2014. ISSN 09594388. doi: 10.1016/j.conb.2014.01.008. URL https://linkinghub.elsevier.com/retrieve/pii/S0959438814000166.

[39] D. Sussillo and O. Barak. Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks. *Neural Computation*, 25(3):626–649, Mar. 2013. ISSN 0899-7667, 1530-888X. doi: 10.1162/NECO_a_00409. URL https://www.mitpressjournals.org/doi/abs/10.1162/NECO_a_00409.

[40] D. Sussillo, R. Jozefowicz, L. F. Abbott, and C. Pandarinath. LFADS - Latent Factor Analysis via Dynamical Systems. *arXiv:1608.06315 [cs, q-bio, stat]*, Aug. 2016.

[41] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler. Efficient transformers: A survey, 2020.

[42] I. Tenney, D. Das, and E. Pavlick. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1452. URL https://www.aclweb.org/anthology/P19-1452.

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[44] S. Vyas, M. D. Golub, D. Sussillo, and K. V. Shenoy. Computation through neural population dynamics. *Annual Review of Neuroscience*, 43(1):249–275, 2020. doi: 10.1146/annurev-neuro-092619-094115. URL https://doi.org/10.1146/annurev-neuro-092619-094115. PMID: 32640928.

[45] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJ4km2R5t7.

[46] S. Wiegreffe and Y. Pinter. Attention is not not explanation, 2019.

[47] N. Wu, B. Green, X. Ben, and S. O'Banion. Deep transformer models for time series forecasting: The influenza prevalence case, 2020.

[48] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, Jan 2021. ISSN 2162-2388. doi: 10.1109/tnnls.2020.2978386. URL http://dx.doi.org/10.1109/TNNLS.2020.2978386.

[49] Y. Zhao and I. M. Park. Variational Latent Gaussian Process for Recovering Single-Trial Dynamics from Population Spike Trains. *Neural Computation*, 29(5):1293–1316, Mar. 2017. ISSN 0899-7667. doi: 10.1162/NECO_a_00953. URL https://doi.org/10.1162/NECO_a_00953.