Benchmarking Probabilistic Time Series Forecasting Models on Neural Activity

Ziyu Lu[1], **Anna J. Li**[2], **Alexander E. Ladd**[2], **Pascha Matveev**[2], **Aditya Deole**[2], **Eric Shea-Brown**[1,3,†], **J. Nathan Kutz**[1,4,†], **Nicholas A. Steinmetz**[2,†]
[1]Department of Applied Mathematics, University of Washington, Seattle, WA, USA.
[2]Department of Neurobiology and Biophysics, University of Washington, Seattle, WA, USA.
[3]Allen Institute for Brain Science, Seattle, WA, USA.
[4]Department of Electrical and Computer Engineering, University of Washington, Seattle, WA, USA.
[†] Authors jointly supervised.
Correspondence to: {luziyu, etsb, kutz, nsteinme} @ uw.edu

Abstract

Neural activity forecasting is central to understanding neural systems and enabling closed-loop control. While deep learning has recently advanced the state-of-the-art in the time series forecasting literature, its application to neural activity forecasting remains limited. To bridge this gap, we systematically evaluated eight probabilistic deep learning models, including two foundation models, that have demonstrated strong performance on general forecasting benchmarks. We compared them against four classical statistical models and two baseline methods on spontaneous neural activity recorded from mouse cortex via widefield imaging. Across prediction horizons, several deep learning models consistently outperformed classical approaches, with the best model producing informative forecasts up to 1.5 seconds into the future. Our findings point toward future control applications and open new avenues for probing the intrinsic temporal structure of neural activity.

1 Introduction

Predicting the future state of a system is central to understanding its underlying dynamics. Beyond its value from the basic science perspective, time series forecasting is also critical to a wide range of real-world applications – including weather prediction[1], renewable energy planning[2], and financial market analysis[3] – guiding decisions ranging from everyday choices to high-stakes operations. Among the most challenging and rewarding domains for forecasting is the brain – one of the most complex and sophisticated systems known to science. Accurate forecasting of brain activity can not only offer insights into the underlying neural mechanisms but also enable transformative applications, including early intervention for neurological disorders[4], the development of therapeutic brain stimulation paradigms[5], and the advancement of brain–machine interfaces[6].

The field of time series forecasting has evolved dramatically over the past decade, driven by the rise of machine learning and deep neural networks[7]. Several studies have assembled datasets across various domains including energy, sales, climate, and healthcare, and systematically assessed the performance of both modern deep learning-based models and classical statistical methods on them[8, 9, 10, 11, 12, 13, 14]. However, with the exception of one recent study[15] which focuses on brain activity of zebrafish, neural time series have been largely absent from established forecasting benchmarks, and their markedly different characteristics from the included series make it unclear whether prior conclusions would apply to them. For instance, neural recordings are often sampled or binned at seconds to milliseconds resolutions, whereas most benchmark datasets are sampled at much

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Data on the Brain & Mind.

coarser intervals (hourly, daily, or monthly). Furthermore, while oscillations are an important feature of brain activity[16], there is also a substantial amount of activity that does not exhibit persistent trends or periodicity/seasonality. In contrast, many real-world time series, such as climate records or energy consumption, are dominated by these patterns. These differences highlight the importance of systematically reassessing forecasting methods in the context of neural data.

Among neural recordings from different species, mouse data bring many important opportunities for time series forecasting. Foremost among the reasons is their relevance for developing and testing future neural control systems: recent advances in optogenetics[17, 18, 19] have enabled targeted manipulation of brain activity in mice, making accurate forecasts immediately applicable to the design of closed-loop control systems[20]. Moreover, modern recording technologies – such as Neuropixels probes[21, 22] for high-density, single-neuron resolution measurements and widefield calcium imaging[23] for simultaneously capturing activity across all dorsal cortical areas – allow large-scale monitoring at multiple spatial and temporal scales. Additional modalities, including behavior tracking[24, 25], cell-type labeling[26], and connectivity mapping[27], can further enhance forecasting performance and expand the range of scientific questions that forecasting can address. Finally, the abundance of publicly available mouse neural datasets[28, 29] provides a unique opportunity to develop large-scale foundation models[30] for neural activity forecasting in mice.

2 Related Work

Forecasting has long been an important approach in studying neural dynamics. Many earlier models, while not explicitly trained to optimize forecasting accuracy, are generative in nature and therefore can produce one-step-ahead or multi-step-ahead forecasts in an autoregressive fashion (e.g. [31, 32, 33]). As forecasting was only a secondary objective in these works, evaluations were generally restricted to a fixed prediction horizon, without systematic assessment across multiple horizons, and lack comparisons against forecasting baselines, such as the average of past activity or models capable of making direct-multi-step forecasts. Another major line of forecasting studies in neuroscience is motivated by brain-machine interface applications, where predicting neural activity is closely tied to predicting behavior. In this setting, models have been trained to solely predict behavior [34, 35, 36] or to jointly predict neural activity and behavior [37, 38]. However, since such experiments often involve highly structured tasks (e.g., a monkey reaching to target), the resulting neural dynamics tend to be stereotyped (e.g., with rotational structure[39]), making them easier to forecast compared to less structured scenarios such as spontaneous activity. Forecasting accuracy has also been combined with other performance measures for both training and evaluation, for example in [40, 41].

More recently, a growing number of models have begun explicitly adopting forecasting accuracy as their primary training objective. For example, [42] proposed a graph neural network based model for multi-channel neural activity forecasting; [43] used a low-rank linear autoregressive model to predict neural responses to holographic photostimulation; [44] developed a transformer-based model leveraging multi-modal inputs to autoregressively predict neural responses to visual stimuli; [45] introduced a diffusion-based model for joint forecasting of neural activity and behavior across sessions and subjects; [46] proposed a convolutional neural network based video model to directly forecast future frames of neural imaging videos; and [47] demonstrated forecasting of spontaneous neural activity across multiple sessions and subjects with a forecaster in the form of multilayer perceptron. While presented as neuroscience-specific applications, these models are fundamentally instances of general time series forecasting methods. Nevertheless, only a few ([46, 47]) have compared their performance against models from the broader time series forecasting literature, and even in those cases, the comparisons were limited. For example no recent foundation models (e.g., [48, 49]) were included. Moreover, all of these models produce only point forecasts. However, given the measurement and intrinsic noise in neural data [50], probabilistic forecasting [51] – which provides prediction intervals to quantify uncertainty – is particularly important, but remains unexplored in the context of neural time series forecasting.

To fill these gaps, we systematically benchmarked common baseline methods, classical statistical approaches, and state-of-the-art deep learning models from the probabilistic time series forecasting literature on spontaneous mice neural activity. By drawing on the broader forecasting literature, we aim to identify strong model backbones that future neuroscience-specific applications can build upon for more accurate and uncertainty-aware neural activity predictions.

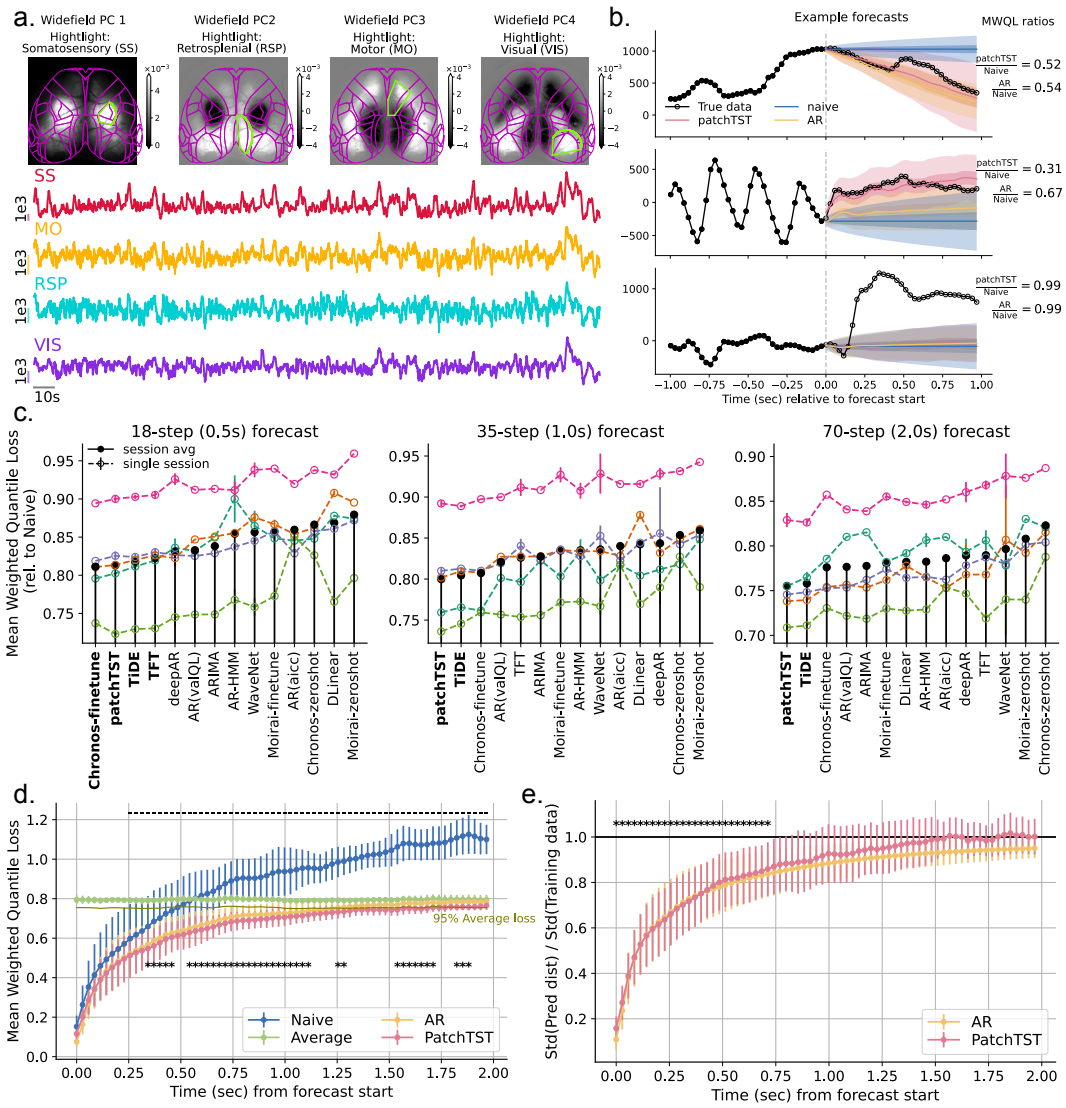

Figure 1: **a.** Widefield imaging data aligned to Allen CCF, with example activity traces from one session. **b.** True and predicted activity in three test samples; dashed lines mark forecast start. Shaded regions: prediction intervals (P.I.): dark = 20%, light = 60%. Top: PatchTST and AR perform similarly, both outperforming Naive. Middle: PatchTST outperforms both AR and Naive. Bottom: PatchTST and AR perform comparably to Naive. Quantitative performance in each sample is listed on the right. **c.** Model performance at three forecast horizons. Models sorted by mean performance across sessions. The y-axis indicates relative performance, computed as the Mean Weighted Quantile Loss (MWQL, see AppendixA.2 for definition) across all predicted steps of each model divided by that of the Naive baseline (i.e., y=1 corresponds to Naive performance). Colors indicate different sessions. Errorbar: mean $\pm$ std across 5 random seeds; fine-tuning Chronos yielded identical results across random seeds, likely due to an issue in the AutoGluon implementation. Performances of Theta and Average are omitted, as they are close to Naive and worse than all other models shown; for 70-step forecast, Average outperforms Naive but still not the others. AR(aicc) vs. AR(valQL): order chosen by AICC vs. validation MWQL. Models in bold significantly outperform AR(valQL) (one-sided paired t-test, $p < 0.05$). **d.** MWQL by forecast step. Errorbar: mean $\pm$ std across sessions. Bars and stars indicate steps where PatchTST significantly outperforms Naive and AR, respectively (one-sided paired t-test, $p < 0.05$). PatchTST and AR losses exceed 95% of the Average model loss (solid line) after 1.80s and 1.28s, respectively. **e.** Ratio of predicted distribution standard deviation (std) to training data std across forecast steps. Errorbar: mean $\pm$ std across sessions. Stars indicate steps where the PatchTST ratio is significantly less than 1 (one-sided t-test, $p < 0.05$). In **b**, **d**, **e**, AR(valQL) is used as AR. For PatchTST, in each session the model achieving median MWQL across 5 random seeds is used.

3 Results

We systematically evaluated 12 univariate probabilistic forecasting models and 2 baseline methods on spontaneous neural activity in mouse cortex recorded using widefield calcium imaging[23] from five experimental sessions (details in AppendixA.1). Such data can be combined with cortex-wide optogenetic perturbations[18] to test forecast-driven closed-loop control. Widefield imaging was performed at 35 Hz, yielding recordings of spontaneous activity lasting 24.5 ± 3.7 minutes (corresponding to $51{,}495 \pm 7{,}784$ timesteps, mean $\pm$ std across five sessions). Data were registered to the Allen Mouse Brain Common Coordinate Framework (CCFv3)[52], and we extracted average activity traces from four major brain regions: somatosensory (SS), motor (MO), visual (VIS), and retrosplenial (RSP) (Figure 1a).

For baselines, we consider the Naive method which repeats the last observed activity, and the Average method which takes the mean of past observed activity as prediction. Among the 12 models, 4 are classical statistical methods: the autoregressive (AR) model, autoregressive integrated moving average (ARIMA)[53], autoregressive hidden Markov model (AR-HMM)[54], and the Theta model[55]; 6 are deep learning based models: DeepAR[56], DLinear[57], Temporal Fusion Transformer (TFT)[58], PatchTST[59], Time-series Dense Encoder (TiDE)[60], and WaveNet[61]; and 2 are time series foundation models[30]: Chronos[48] and Moirai[49], for which we evaluated both zero-shot and fine-tuned performance. More details on models and training are provided in AppendixA.2, A.4.

The forecasting task was defined as predicting activity in the interval $[i, i+L)$ from the preceding observations in $[i-H, i)$, where L is the forecast horizon and H is the history length. Following previous forecasting literature[56, 58, 62], we partitioned all time series chronologically, using the earliest 60% of timesteps for training ($[0, T_{\text{train}})$), the next 20% for validation ($[T_{\text{train}}, T_{\text{val}})$), and the final 20% for testing ($[T_{\text{val}}, T_{\text{test}})$). Validation and test samples were generated using sliding windows with non-overlapping targets. For instance, the first test sample forecasts $[T_{\text{test}}, T_{\text{test}}+L)$ from $[T_{\text{test}}-H, T_{\text{test}})$, the second forecasts $[T_{\text{test}}+L, T_{\text{test}}+2L)$ from $[T_{\text{test}}+L-H, T_{\text{test}}+L)$, and so on. In this way the forecast targets for evaluation remain fixed while varying the history length. We experimented with $L = 18, 35,$ and 70, which correspond to about $0.5, 1,$ and 2 seconds. H is treated as a hyperparameter and is tuned for each L and each model individually.

In aggregated performance over test samples, all methods outperformed the Naive and Average controls (Figure 1c, also see AppendixA.5, Figure 2 for evaluation on additional metrics). Classical autoregressive models provided competitive baselines, but several deep learning models – PatchTST, TiDE, and fine-tuned Chronos – consistently achieved higher accuracy across different prediction horizons. The strong performance of PatchTST aligns with findings from a recent benchmarking study on time series from various domains, such as economics, energy, and retail[14]. In contrast, foundation models (Chronos and Moirai) pretrained on these domains transferred poorly to neural activity in a zero-shot setting, indicating that neural time series exhibit distinct characteristics from the data used for pretraining. Nevertheless, Chronos becomes competitive after fine-tuning on neural activity, suggesting that its architecture is sufficiently flexible to capture neural dynamics. At the level of individual test samples, we found that while some features of activity were captured by the models, there also exist large fluctuations that were not predicted by any model (Figure 1b).

The improved performance of models over controls extended across a range of forecast steps, out to more than 1 second. Note that in Figure 1c, model performance is aggregated across all predicted steps. Therefore, it is actually unclear whether models still outperform baselines specifically at step 70, as the advantage may arise from better performance at earlier steps. When ploting error as a function of prediction step (Figure 1d), we observed that performance of Naive, AR, and PatchTST all deteriorates with increasing horizon. While the Naive model continues to worsen, AR and PatchTST converge toward the Average model's score. This behavior is expected for AR models: it can be verified that for stationary time series, h-step-ahead AR forecast will converge to the mean as $h \to \infty$. A similar pattern can be found in other metrics (AppendixA.5, Figure 3). As PatchTST predicts all future steps simultaneously rather than autoregressively, separate models are optimized for each forecast horizon (18, 35, and 70 steps). We verified that the step-wise error curves are consistent across these models: for example, the first 18 steps of the 35- and 70-step models closely overlap with the 18-step model (AppendixA.5, Figure 4).

In addition, we quantified the uncertainty of probabilistic forecasts across steps, computed as the standard deviation of the predicted distribution over that of the training data (Figure 1e). For both

AR and PatchTST, this ratio of standard deviations increases with forecast horizon and approaches one (which is again expected for the AR model). Importantly, while wider prediction intervals (as suggested by larger ratio here) may be less informative for encompassing too many possibilities, intervals that are narrower are not necessarily better, as they may miss sources of uncertainty[63]. Figure 1d, e suggest that forecasts are most reliable within 35 steps (1 sec), and informative predictions may extend to about 50 steps (1.5 sec); beyond this, even the best of the current models (PatchTST) performs no better than simply predicting the mean and standard deviation of training data.

4 Discussion

Several recent studies have proposed models tailored to forecasting neural time series. Nevertheless, models developed in the broader forecasting literature have not been systematically evaluated on neural activity, and the important aspect of uncertainty quantification has been largely overlooked. Here we bridged this gap by benchmarking models ranging from classical autoregressive methods to modern deep learning and foundation models on widefield imaging data. Several deep learning models consistently outperformed the strong baseline set by classical statistical models, yet both accuracy and uncertainty estimates indicate that current forecasts may only be informative up to a certain horizon. Whether this limit comes from model design constraints of existing methods or instead reflects intrinsic sources of variability and time scales in neural activity remains an open question. To probe the former, it may be of interest to develop neural time series forecasting foundation models by combining strong backbones identified here with neuroscience-specific innovations, such as cross-subject training[34, 36, 47, 45]. Meanwhile, closed-loop control experiments informed by forecasting performance may help reveal the temporal structure inherent to neural system.

Acknowledgments

We gratefully acknowledge support from the National Science Foundation (NSF) (Research Grant 2024364 to ESB and NAS; CAREER award 2142911 to NAS). The work of ZL and JNK was supported in part by the NSF AI Institute for Dynamical Systems (dynamicsai.org), grant 2112085. Additional support was provided by the Pew Biomedical Scholars Program (to NAS), the Klingenstein-Simons Fellowship in Neuroscience (to NAS), the NIH Ruth Kirchenstein award (T32EY007031 and F31EY035880 to AJL), and the Simons Foundation Shenoy Undergraduate Research Fellowship (to PM). We thank Kimberly Miller for help with mouse breeding and husbandry.

References

[1] Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, et al. Probabilistic weather forecasting with machine learning. *Nature*, 637(8044):84–90, 2025.

[2] Huaizhi Wang, Zhenxing Lei, Xian Zhang, Bin Zhou, and Jianchun Peng. A review of deep learning for renewable energy forecasting. *Energy Conversion and Management*, 198:111799, 2019.

[3] Mahinda Mailagaha Kumbure, Christoph Lohrmann, Pasi Luukka, and Jari Porras. Machine learning techniques and data for stock market forecasting: A literature review. *Expert Systems with Applications*, 197:116659, 2022.

[4] Florian Mormann, Ralph G Andrzejak, Christian E Elger, and Klaus Lehnertz. Seizure prediction: the long and winding road. *Brain*, 130(2):314–333, 2007.

[5] Matteo De Matola and Carlo Miniussi. Brain state forecasting for precise brain stimulation: Current approaches and future perspectives. *NeuroImage*, page 121050, 2025.

[6] Mikhail A Lebedev and Miguel AL Nicolelis. Brain–machine interfaces: past, present and future. *TRENDS in Neurosciences*, 29(9):536–546, 2006.

[7] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

[8] Spyros Makridakis, Evangelos Spiliotis, Vassilios Assimakopoulos, Artemios-Anargyros Semenoglou, Gary Mulder, and Konstantinos Nikolopoulos. Statistical, machine learning and deep learning forecasting methods: Comparisons and ways forward. *Journal of the Operational Research Society*, 74(3):840–859, 2023.

[9] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020.

[10] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. M5 accuracy competition: Results, findings, and conclusions. *International journal of forecasting*, 38(4):1346–1364, 2022.

[11] Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I Webb, Rob J Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. *arXiv preprint arXiv:2105.06643*, 2021.

[12] Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S. Jensen, Zhenli Sheng, and Bin Yang. Tfb: Towards comprehensive and fair benchmarking of time series forecasting methods. *Proc. VLDB Endow.*, 17(9):2363–2377, 2024.

[13] Zezhi Shao, Fei Wang, Yongjun Xu, Wei Wei, Chengqing Yu, Zhao Zhang, Di Yao, Tao Sun, Guangyin Jin, Xin Cao, et al. Exploring progress in multivariate time series forecasting: Comprehensive benchmarking and heterogeneity analysis. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[14] Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Gift-eval: A benchmark for general time series forecasting model evaluation. *arXiv preprint arXiv:2410.10393*, 2024.

[15] Jan-Matthis Lueckmann, Alexander Immer, Alex Bo-Yuan Chen, Peter H Li, Mariela D Petkova, Nirmala A Iyer, Luuk Willem Hesselink, Aparna Dev, Gudrun Ihrke, Woohyun Park, et al. Zapbench: A benchmark for whole-brain activity prediction in zebrafish. *arXiv preprint arXiv:2503.02618*, 2025.

[16] Gregor Thut, Carlo Miniussi, and Joachim Gross. The functional importance of rhythmic activity in the brain. *Current Biology*, 22(16):R658–R663, 2012.

[17] Peter Zatka-Haas, Nicholas A Steinmetz, Matteo Carandini, and Kenneth D Harris. Sensory coding and the causal impact of mouse cortex in a visual decision. *Elife*, 10:e63163, 2021.

[18] Pascha Matveev, Anna J Li, Zhiwen Ye, Anna J Bowen, Ximena Opitz-Araya, Jonathan T Ting, and Nicholas A Steinmetz. Simultaneous mesoscopic measurement and manipulation of mouse cortical activity. *bioRxiv*, 2024.

[19] Anna Lakunina, Karolina Z Socha, Alexander Ladd, Anna J Bowen, Susu Chen, Jennifer Colonell, Anjal Doshi, Bill Karsh, Michael Krumin, Pavel Kulik, et al. Neuropixels opto: Combining high-resolution electrophysiology and optogenetics. *bioRxiv*, 2025.

[20] Logan Grosenick, James H Marshel, and Karl Deisseroth. Closed-loop and activity-guided optogenetic control. *Neuron*, 86(1):106–139, 2015.

[21] Nicholas A Steinmetz, Cagatay Aydin, Anna Lebedeva, Michael Okun, Marius Pachitariu, Marius Bauza, Maxime Beau, Jai Bhagat, Claudia Böhm, Martijn Broux, et al. Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. *Science*, 372(6539):eabf4588, 2021.

[22] Maxwell D Melin, Anup Khanal, Marvin Vasquez, Michael B Ryan, Anne K Churchland, and Joao Couto. Large scale, simultaneous, chronic neural recordings from multiple brain areas. *bioRxiv*, pages 2023–12, 2024.

[23] Chi Ren and Takaki Komiyama. Characterizing cortex-wide dynamics with wide-field calcium imaging. *Journal of Neuroscience*, 41(19):4160–4168, 2021.

[24] Alexander Mathis, Pranav Mamidanna, Kevin M Cury, Taiga Abe, Venkatesh N Murthy, Mackenzie Weygandt Mathis, and Matthias Bethge. Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature neuroscience*, 21(9):1281–1289, 2018.

[25] Atika Syeda, Lin Zhong, Renee Tung, Will Long, Marius Pachitariu, and Carsen Stringer. Facemap: a framework for modeling neural activity based on orofacial tracking. *Nature neuroscience*, 27(1):187–195, 2024.

[26] Stephane Bugeon, Joshua Duffield, Mario Dipoppa, Anne Ritoux, Isabelle Prankerd, Dimitris Nicoloutsopoulos, David Orme, Maxwell Shinn, Han Peng, Hamish Forrest, et al. A transcriptomic axis predicts state modulation of cortical interneurons. *Nature*, 607(7918):330–338, 2022.

[27] Functional connectomics spanning multiple areas of mouse visual cortex. *Nature*, 640(8058):435–447, 2025.

[28] Saskia EJ de Vries, Joshua H Siegle, and Christof Koch. Sharing neurophysiology data from the allen brain observatory. *Elife*, 12:e85550, 2023.

[29] Kenneth D Harris, Max Hunter, Cyrille Rossant, Maho Sasaki, Shan Shen, Nicholas A Steinmetz, Edgar Y Walker, Olivier Winter, and Miles Wells. Data architecture for a large-scale neuroscience collaboration. *bioRxiv*, 2019.

[30] Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. Foundation models for time series analysis: A tutorial and survey. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pages 6555–6565, 2024.

[31] Carina Curto, Shuzo Sakata, Stephan Marguet, Vladimir Itskov, and Kenneth D Harris. A simple model of cortical dynamics explains variability and state dependence of sensory responses in urethane-anesthetized auditory cortex. *Journal of neuroscience*, 29(34):10600–10612, 2009.

[32] Joshua Glaser, Matthew Whiteway, John P Cunningham, Liam Paninski, and Scott Linderman. Recurrent switching dynamical systems models for multiple interacting neural populations. *Advances in neural information processing systems*, 33:14867–14878, 2020.

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

[34] Mehdi Azabou, Vinam Arora, Venkataramana Ganesh, Ximeng Mao, Santosh Nachimuthu, Michael Mendelson, Blake Richards, Matthew Perich, Guillaume Lajoie, and Eva Dyer. A unified, scalable framework for neural population decoding. *Advances in Neural Information Processing Systems*, 36:44937–44956, 2023.

[35] Avery Hee-Woon Ryoo, Nanda H Krishna, Ximeng Mao, Mehdi Azabou, Eva L Dyer, Matthew G Perich, and Guillaume Lajoie. Generalizable, real-time neural decoding with hybrid state-space models. *arXiv preprint arXiv:2506.05320*, 2025.

[36] Mehdi Azabou, Krystal Xuejing Pan, Vinam Arora, Ian Jarratt Knight, Eva L Dyer, and Blake Aaron Richards. Multi-session, multi-task neural decoding from distinct cell-types and brain regions. In *The Thirteenth International Conference on Learning Representations*, 2025.

[37] Omid G Sani, Bijan Pesaran, and Maryam M Shanechi. Dissociative and prioritized modeling of behaviorally relevant neural dynamics using recurrent neural networks. *Nature neuroscience*, 27(10):2033–2045, 2024.

[38] Parsa Vahidi, Omid G. Sani, and Maryam Shanechi. BRAID: Input-driven nonlinear dynamical modeling of neural-behavioral data. In *The Thirteenth International Conference on Learning Representations*, 2025.

[39] Chethan Pandarinath, Daniel J O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D Stavisky, Jonathan C Kao, Eric M Trautmann, Matthew T Kaufman, Stephen I Ryu, Leigh R Hochberg, et al. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature methods*, 15(10):805–815, 2018.

[40] Felix Pei, Joel Ye, David Zoltowski, Anqi Wu, Raeed H Chowdhury, Hansem Sohn, Joseph E O'Doherty, Krishna V Shenoy, Matthew T Kaufman, Mark Churchland, et al. Neural latents benchmark'21: evaluating latent variable models of neural population activity. *arXiv preprint arXiv:2109.04463*, 2021.

[41] Yizi Zhang, Yanchen Wang, Donato Jiménez-Benetó, Zixuan Wang, Mehdi Azabou, Blake Richards, Renee Tung, Olivier Winter, Eva Dyer, Liam Paninski, et al. Towards a" universal translator" for neural dynamics at single-cell, single-spike resolution. *Advances in Neural Information Processing Systems*, 37:80495–80521, 2024.

[42] Jingyuan Li, Leo Scholl, Trung Le, Pavithra Rajeswaran, Amy Orsborn, and Eli Shlizerman. Amag: Additive, multiplicative and adaptive graph neural network for forecasting neuron activity. *Advances in Neural Information Processing Systems*, 36:8988–9014, 2023.

[43] Andrew Wagenmaker, Lu Mi, Marton Rozsa, Matthew Bull, Karel Svoboda, Kayvon Daie, Matthew Golub, and Kevin G Jamieson. Active learning of neural population dynamics using two-photon holographic optogenetics. *Advances in Neural Information Processing Systems*, 37:31659–31687, 2024.

[44] Antonis Antoniades, Yiyi Yu, Joe S Canzano, William Yang Wang, and Spencer Smith. Neuroformer: Multimodal and multitask generative pretraining for brain data. In *The Twelfth International Conference on Learning Representations*, 2024.

[45] A. Carolina Filipe and Il Memming Park. One model to train them all: A unified diffusion framework for multi-context neural population forecasting, 2025.

[46] Alexander Immer, Jan-Matthis Lueckmann, Alex Bo-Yuan Chen, Peter H Li, Mariela D Petkova, Nirmala A Iyer, Aparna Dev, Gudrun Ihrke, Woohyun Park, Alyson Petruncio, et al. Forecasting whole-brain neuronal activity from volumetric video. *arXiv preprint arXiv:2503.00073*, 2025.

[47] Yu Duan, Hamza Tahir Chaudhry, Misha B Ahrens, Christopher D Harvey, Matthew G Perich, Karl Deisseroth, and Kanaka Rajan. Poco: Scalable neural forecasting through population conditioning. *arXiv preprint arXiv:2506.14957*, 2025.

[48] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

[49] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. 2024.

[50] A Aldo Faisal, Luc PJ Selen, and Daniel M Wolpert. Noise in the nervous system. *Nature reviews neuroscience*, 9(4):292–303, 2008.

[51] Tilmann Gneiting and Matthias Katzfuss. Probabilistic forecasting. *Annual Review of Statistics and Its Application*, 1(1):125–151, 2014.

[52] Quanxin Wang, Song-Lin Ding, Yang Li, Josh Royall, David Feng, Phil Lesnar, Nile Graddis, Maitham Naeemi, Benjamin Facer, Anh Ho, et al. The allen mouse brain common coordinate framework: a 3d reference atlas. *Cell*, 181(4):936–953, 2020.

[53] Robert H Shumway and David S Stoffer. *Time series analysis and its applications: with R examples*. Springer, 2006.

[54] Chang-Jin Kim and Charles R Nelson. *State-space models with regime switching: classical and Gibbs-sampling approaches with applications*. MIT press, 2017.

[55] Vassilis Assimakopoulos and Konstantinos Nikolopoulos. The theta model: a decomposition approach to forecasting. *International journal of forecasting*, 16(4):521–530, 2000.

[56] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3):1181–1191, 2020.

[57] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.

[58] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.

[59] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

[60] Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan Mathur, Rajat Sen, and Rose Yu. Long-term forecasting with tide: Time-series dense encoder. *arXiv preprint arXiv:2304.08424*, 2023.

[61] Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, et al. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 12:1, 2016.

[62] Rob J. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, Melbourne, Australia, 3rd edition, 2021. Accessed on September 1, 2025.

[63] Chris Chatfield. *Time-series forecasting*. Chapman and Hall/CRC, 2000.

[64] Zhiwen Ye, Alexander Ladd, Nancy MacKenzie, Ljuvica Kolich, Anna J. Li, Daniel Birman, Matthew S. Bull, Tanya L. Daigle, Bosiljka Tasic, Hongkui Zeng, and Nicholas A. Steinmetz. Brain-wide topographic coordination of traveling spiral waves. *bioRxiv*, 2025.

[65] Rob J. Hyndman. Derivations of forecast variance for benchmark methods. `https://robjhyndman.com/hyndsight/forecasting_variances.html`, 2022. Accessed: 2025-08-25.

[66] Peter J Brockwell and Richard A Davis. *Introduction to time series and forecasting*. Springer, 2002.

[67] Jose A Fiorucci, Tiago R Pellegrini, Francisco Louzada, Fotios Petropoulos, and Anne B Koehler. Models for optimising the theta method and their relationship to state space models. *International journal of forecasting*, 32(4):1151–1161, 2016.

[68] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C. Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, Lorenzo Stella, Ali Caner Türkmen, and Yuyang Wang. GluonTS: Probabilistic and Neural Time Series Modeling in Python. *Journal of Machine Learning Research*, 21(116):1–6, 2020.

[69] Youngsuk Park, Danielle Maddix, François-Xavier Aubet, Kelvin Kan, Jan Gasthaus, and Yuyang Wang. Learning quantile functions without quantile crossing for distribution-free time series forecasting. In *International conference on artificial intelligence and statistics*, pages 8127–8150. PMLR, 2022.

[70] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[71] Hanna Pankka, Jaakko Lehtinen, Risto J Ilmoniemi, and Timo Roine. Enhanced eeg forecasting: a probabilistic deep learning approach. *Neural Computation*, 37(4):793–814, 2025.

[72] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[73] Abdul Fatir Ansari, Caner Turkmen, Oleksandr Shchur, and Lorenzo Stella. Fast and accurate zero-shot forecasting with chronos, bolt, and autogluon. https://aws.amazon.com/blogs/machine-learning/fast-and-accurate-zero-shot-forecasting-with-chronos-bolt-and-autogluon/, 2024. Accessed: 2025-07-29.

[74] Federico Garza, Max Mergenthaler Canseco, Cristian Challú, and Kin G. Olivares. StatsForecast: Lightning fast forecasting with statistical and econometric models. PyCon Salt Lake City, Utah, US 2022, 2022.

[75] Scott W. Linderman, Peter Chang, Giles Harper-Donnelly, Aleyna Kara, Xinglong Li, Gerardo Duran-Martin, and Kevin Murphy. Dynamax: A python package for probabilistic state space modeling with jax. *Journal of Open Source Software*, 10(108):7069, 2025.

[76] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The journal of machine learning research*, 13(1):281–305, 2012.

[77] Oleksandr Shchur, Caner Turkmen, Nick Erickson, Huibin Shen, Alexander Shirkov, Tony Hu, and Yuyang Wang. AutoGluon-TimeSeries: AutoML for probabilistic time series forecasting. In *International Conference on Automated Machine Learning*, 2023.

A Appendix

A.1 Data collection and preprocessing

All experimental protocols were conducted according to US National Institutes of Health guidelines for animal research and approved by the Institutional Animal Care and Use Committee at the University of Washington.

We analyzed five sessions of widefield imaging experiments collected from five male and female mice from multiple GCaMP-expressing genotypes: two expressed GCaMP8s (tetO-GCaMP8s), one expressed GCaMP6s (tetO-GCaMP6s), and two expressed jGCaMP7f (Ai210 and Ai210 triple). Some of the data has been used in [18] or [64], and the readers should refer to them for more experimental details. Briefly, images were collected with a CMOS camera with 17.3 um/pixel resolution (Basler acA2440-75m) and 560×560 pixel frame size. The camera was fitted with a 0.63x objective lens (Leica Planapo 0.63x). The true frame rate is 70 Hz, but the effective rate is 35 Hz due to alternating 405 nm and 470 nm excitation for hemodynamic correction. Mice were head-fixed facing a dark screen, and were free to turn a wheel. In two of the five sessions, 10% sucrose rewards were given in random 2-5 seconds intervals to keep mice at an awake and alert state. Three of the five sessions were solely spontaneous activity. In the other two sessions, receptive field mapping experiments were performed prior to spontaneous activity recording, and we analyzed spontaneous activity starting at least 2 minutes after the receptive field mapping experiments were turned off. Non-neuronal signals arising from hemodynamic changes were removed by subtracting the violet-light evoked signals from the blue-light evoked signals with linear regression, as previously described in [17, 64]. To store and process the data, we compressed the widefield data (D) into spatial components (U) and temporal components ($SV^\top$) with singular value decomposition in the form $D = USV^\top$. For each of the four brain regions (SS, MO, RSP, VIS), activity was reconstructed using the top 500 (four sessions) or 200 (one session) SVD components and was averaged across pixels within that region. It was reported in [18] and [64] that over 97% of the total variance before SVD compression was captured within the top 50 components. As we have used hundreds of components here, the information lost in compression and reconstruction should be negligible.

A.2 Univariate time series models

A.2.1 Baselines [65]

1. **Naive (repeat-last-step).** Let y_t denote the time series at timestep t, and $\{\epsilon_t\}$ be white noise with mean 0 and variance σ^2. The Naive baseline assumes y_t follows a random walk: $y_t = y_{t-1} + \epsilon_t$. Thus, to forecast from y_T, we have

$$\begin{aligned} y_{T+1} &= y_T + \epsilon_{T+1} \\ y_{T+2} &= y_{T+1} + \epsilon_{T+2} = y_T + \epsilon_{T+1} + \epsilon_{T+2} \\ &\cdots \\ y_{T+h} &= y_T + \sum_{i=1}^{h} \epsilon_{T+i} \end{aligned}$$

   Let $y_{T+h|T}$ denote the forecast of y_{T+h} using previous observations up to (and including) y_T. It follows that

$$\begin{aligned} \hat{y}_{T+h|T} &:= \mathbb{E}[y_{T+h|T}] = y_T, \\ \hat{\sigma}_h^2 &:= \mathrm{Var}(y_{T+h|T}) = \mathrm{Var}(\sum_{i=1}^{h} \epsilon_{T+i}) = \sum_{i=1}^{h} \mathrm{Var}(\epsilon_{T+i}) = h\sigma^2, \end{aligned}$$

   where σ^2 can be estimated from fitted residuals:

$$\sigma^2 = \frac{1}{T-1} \sum_{i=2}^{T} (y_i - y_{i|i-1})^2 = \frac{1}{T-1} \sum_{i=2}^{T} (y_i - y_{i-1})^2.$$

   Assuming ϵ_t is normal, then a $100(1-\alpha)\%$ prediction interval is

$$\left[\hat{y}_{T+h|T} - z_{\frac{\alpha}{2}} \hat{\sigma}_h, \hat{y}_{T+h|T} + z_{\frac{\alpha}{2}} \hat{\sigma}_h\right],$$

where $z_{\frac{\alpha}{2}}$ is the value above which a fraction of $\frac{\alpha}{2}$ of the data in a standard normal distribution falls.

2. **Average.** Again let y_t denote the time series at timestep t, and $\{\epsilon_t\}$ be white noise with mean 0 and variance σ^2. The Average baseline assumes y_t randomly fluctuates around some mean value: $y_t = c + \epsilon_t$, where c is some constant to be estimated from the past observations. To forecast from y_T, we first compute the least-squares estimate of c:

$$\hat{c} = \frac{1}{T}\sum_{j=0}^{T-1} y_{T-j},$$

then for $i = 1, 2, \ldots,$

$$y_{T+i} = \hat{c} + \epsilon_{T+i}.$$

It follows that

$$\hat{y}_{T+h|T} := \mathbb{E}[y_{T+h|T}] = \hat{c},$$
$$\hat{\sigma}_h^2 := \mathrm{Var}(y_{T+h|T}) = \mathrm{Var}(\hat{c}) + \mathrm{Var}(\epsilon_{T+i}) = \frac{1}{T}\sigma^2 + \sigma^2,$$

where σ^2 can be estimated from fitted residuals:

$$\sigma^2 = \frac{1}{T-1}\sum_{j=0}^{T-1}(y_{T-j} - \hat{c})^2.$$

The prediction interval can be constructed in the same manner as in the Naive method. Note that $\hat{\sigma}_h$ is actually independent of h, and thus the width of prediction interval is constant through all forecasting steps.

A.2.2 Local models

The following models are "local" in the sense that we need to fit separate models to each series if there are multiple time series in one dataset.

1. **Autoregressive integrated moving average (ARIMA) family [53].** Let y_t denote the time series at timestep t, and $\{\epsilon_t\}$ be white noise. An ARIMA(p, q, d) model is of the form

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t$$

where y'_t is a d-th order differenced version of y_t. We applied both the KPSS test and the augmented Dickey–Fuller test to each time series here and found that they all satisfy the stationarity condition. So we take $d = 0$, $y'_t = y_t$, and the ARIMA model simplifies to an ARMA model. The **autoregressive (AR) model** can be seen as a special case of ARIMA, where $d = 0$, and the $\theta' s$ are taken to be zero. During fitting, for fixed p, q, the parameters c, ϕ_i, θ_i are determined by maximizing the likelihood of observed data under the assumption that $\{\epsilon_t\}$ is Gaussian

$$L(c, \{\phi_i\}_{i=1}^p, \{\theta_i\}_{i=1}^q) = \prod_{t=1}^{n} f(y'_t | y'_{t-1}, \ldots, y'_1, c, \{\phi_i\}_{i=1}^p, \{\theta_i\}_{i=1}^q).$$

To find the most appropriate value of (p, q), a common approach is to compute the likelihood for various values of (p, q), and choose the pair achieving the minimum AICC[66]. Alternatively, we can evaluate fitted models on a validation set, and selected the (p, q) pair achieving the best score on the validation set.
The **autoregressive hidden Markov model (AR-HMM)** extends the AR model, allowing the parameters to switch among different values. It can be written as

$$y_t = c^{S_t} + \phi_1^{S_t} y_{t-1} + \cdots + \phi_p^{S_t} y_{t-p} + \epsilon_t^{S_t},$$

where S_t denotes the state (assumed to be hidden, unobserved) at timestep t, and $\epsilon_t^{S_t} \sim \mathcal{N}(0, (\sigma^{S_t})^2)$. S_t switches among a set of discrete values following a first-order Markov chain. Since it is possible to transition into several different states at each step (although some states are more probable than others), AR-HMM can make probabilistic forecasts by sampling multiple trajectories from the forecast start.

2. **Theta family [55, 67].** The general idea of Theta model is to decompose the time series into multiple components with different curvatures (as controlled by second derivatives), extrapolate each component separately into the future, and then combine their extrapolations. In its most basic form proposed in [55], two Theta lines are drawn from the original time series: one with second derivative equal to 0 ($\Theta = 0$) for extracting long-term linear trend, and the other with second derivative double that of the original series ($\Theta = 2$) for extracting local fluctuations. During prediction, the first Theta line ($\Theta = 0$) continues to follow the linear trend, and the second Theta line ($\Theta = 2$) is extrapolated via simple exponential smoothing. A simple average of their extrapolations is taken to be the final forecast. Methods to further improve Θ-parameter selection and extrapolations were later proposed in [67]. In a recent benchmark study [14], the Theta family demonstrates competitive performance on time series with high sampling frequency (10 sec). Nonetheless, as those series are from a very different domain (cloud operations) than neuroscience, it is unclear whether the Theta family is also effective for forecasting widefield imaging data.

A.2.3 Global models

The following models are "global" in the sense that one model can simultaneous fit all time series in a dataset.

1. **DeepAR[56].** DeepAR model makes probabilistic forecasts with an autoregressive recurrent neural network (RNN). The value of the time series at the next timestep is assumed to be drawn from some probability distribution, which is usually taken to be Gaussian or Student-t for continuous data. The parameters of the distribution (e.g., the mean and variance for Gaussian distribution) are determined by the output of a multi-layer RNN: Let y_t denote time series observations and x_t denote some known covariate. At timestep t, the RNN computes $\boldsymbol{h}_t$ from $\boldsymbol{h}_{t-1}, y_{t-1}, x_t$, and assumes $y_t \sim \theta(\boldsymbol{h}_t)$, where $\theta(\cdot)$ is some fixed distribution. In practice, it is often advantageous to also incorporate lagged versions of y_{t-1} and x_t (e.g. y_{t-2}, x_{t-1}) when computing $\boldsymbol{h}_t$. During inference, to forecast L steps from $t = T + 1$, the model will first run from $t = T - H$ (where H is the relevant history context length) to T to generate $\boldsymbol{h}_{T+1}$, sample one $\hat{y}_{T+1}$ from $\theta(\boldsymbol{h}_{T+1})$, and then use $\hat{y}_{T+1}, \boldsymbol{h}_{T+1}, x_{T+1}$ to compute $\boldsymbol{h}_{T+2}$ and generate a sample trajectory in this autoregressive form. To make probabilistic forecasts, multiple sample trajectories will be generated, and the prediction interval at each time step will be estimated from the sample trajectories.
2. **DLinear[57].** DLinear model is a one-layer linear feedforward network for direct multi-step forecasting. It was originally proposed to showcase that the advantage of many transformer-based models in time series forecasting mainly comes from the direct multi-step-ahead forecasting strategy (as opposed to autoregressive one-step-ahead), instead of the transformer architecture. Suppose we want to forecast $\boldsymbol{y}^{\text{future}} = [y_T, \ldots, y_{T+L}]$ from $\boldsymbol{y}^{\text{history}} = [y_{T-H}, \ldots, y_T]$. Using a moving average smoother, $\boldsymbol{y}^{\text{history}}$ is first decomposed into a trend and a remainder component: $\boldsymbol{y}^{\text{history}} = \boldsymbol{y}^{\text{history, trend}} + \boldsymbol{y}^{\text{history, remainder}}$. Each of the two components is passed through a one-layer linear feedforward network, and then summed together as the future forecast: $\hat{\boldsymbol{y}}^{\text{future}} = \boldsymbol{W}_1 \boldsymbol{y}^{\text{history, trend}} + \boldsymbol{W}_2 \boldsymbol{y}^{\text{history, remainder}}$, where $\boldsymbol{W}_1, \boldsymbol{W}_2$ are $L \times H$ matrices. If the time series is multivariate, i.e., for the i-th variate, $\boldsymbol{y}^{(i),\text{future}} = \left[y_T^{(i)}, \ldots, y_{T+L}^{(i)}\right]$, $\boldsymbol{y}^{(i),\text{history}} = \left[y_{T-H}^{(i)}, \ldots, y_T^{(i)}\right]$, then $\boldsymbol{W}_1, \boldsymbol{W}_2$ will be shared across all variates and no cross-variate relationship will be modeled, i.e., $\hat{\boldsymbol{y}}^{(i),\text{future}} = \boldsymbol{W}_1 \boldsymbol{y}^{(i),\text{history, trend}} + \boldsymbol{W}_2 \boldsymbol{y}^{(i),\text{history, remainder}}$ for all i.

   The original DLinear model only produces point forecasts, but the GluonTS package[68] adds an additional simple transformation step to make it output the parameters of the desired distribution and thus make the forecasts probabilistic. For example, for the Gaussian distribution, $\hat{y}_t^{\text{future}} \sim \mathcal{N}(\mu(\boldsymbol{y}^{\text{feature}}), \sigma(\boldsymbol{y}^{\text{feature}}))$, where $\boldsymbol{y}^{\text{feature}} \in \mathbb{R}^D$. $\boldsymbol{y}^{\text{feature}} = \boldsymbol{W}_1 \boldsymbol{y}^{\text{history, trend}} + \boldsymbol{W}_2 \boldsymbol{y}^{\text{history, remainder}}$ as in the original DLinear model. $\mu(\boldsymbol{y}^{\text{feature}}) = \boldsymbol{w}_\mu^\top \boldsymbol{y}^{\text{feature}} + \boldsymbol{b}_\mu$, and $\sigma(\boldsymbol{y}^{\text{feature}}) = \log(1 + \exp(\boldsymbol{w}_\sigma^\top \boldsymbol{y}^{\text{feature}} + \boldsymbol{b}_\sigma))$.
3. **TFT[58].** Temporal fusion transformer (TFT) is a versatile model that can incorporate static covariates, covariates that are only known in the past, and covariates that are known for the forecasting horizon. It first uses recurrent layers to extract local temporal information that is shared across neighboring time steps, and then uses the attention mechanism to process global temporal information that are spread across a wider time frame. It performs

probabilistic forecasting by optimizing the quantile loss. However, one caveat is that it does not address the problem of quantile crossing[69]. For instance, the predicted 0.5-quantile may be larger than the predicted 0.9-quantile.

4. **PatchTST[59].** PatchTST model uses the vanilla transformer encoder[70] as backbone, together with two key designs: channel-independence and patching. Suppose we want to forecast a M-dimensional multivariate time series $\boldsymbol{y}^{\text{future}} = [\boldsymbol{y}_T, \ldots, \boldsymbol{y}_{T+L}]$ from $\boldsymbol{y}^{\text{history}} = [\boldsymbol{y}_{T-H}, \ldots, \boldsymbol{y}_T]$, where $\boldsymbol{y}_t \in \mathbb{R}^M$. PatchTST models each variate individually using a shared model $\mathcal{F}$ ("channel-independence"): $\boldsymbol{y}^{(i),\text{future}} = \mathcal{F}(\boldsymbol{y}^{(i),\text{history}})$, where $\boldsymbol{y}^{(i),\text{future}} = \left[y_T^{(i)}, \ldots, y_{T+L}^{(i)}\right]$, $\boldsymbol{y}^{(i),\text{history}} = \left[y_{T-H}^{(i)}, \ldots, y_T^{(i)}\right]$. For each variate i, neighboring timesteps are grouped together as one "patch" (or token), and then fed into the transformer encoder. The patches may or may not be overlapping, and the number of patches is typically smaller than the number of history context timesteps. Each patch will be linearly mapped to a high-dimensional vector, and go through nonlinear transformations (such as self-attention and feed-forward networks) in the transformer encoder. $\hat{\boldsymbol{y}}^{(i),\text{future}}$ is computed from linear transformation of the transformer encoder output. Similar to DLinear, the original PatchTST model only produces point forecasts, but is adapted for probabilistic forecasting by the GluonTS[68] package.

5. **TiDE[60].** Time-series Dense Encoder (TiDE) can be thought as a nonlinear extension of the DLinear model. While DLinear only accounts for the linear relationship, TiDE uses the classical multi-layer perceptron and residual connections to capture the potentially nonlinear relationship between future and observed activity. When there are multiple time series in the dataset, it operates in a channel-independent manner, as PatchTST, and does not model the relationship between different time series. Similar to DLinear and PatchTST, the original TiDE model only produces point forecasts, but is adapted for probabilistic forecasting by the GluonTS[68] package.

6. **WaveNet[61].** WaveNet is a convolutional neural network-based autoregressive model. For a sequence $[y_1, \ldots, y_T]$, it learns $p(y_t|y_{t-1}, \ldots, y_1)$ for all t. WaveNet is originally proposed for modeling audio signals, which are usually stored as 16-bit integers so only take a finite number of values. Thus WaveNet can learn $p(y_t|\cdots)$ as categorical distributions and can be trained using the cross-entropy loss. GluonTS[68] adapts WaveNet to model general time series by discretizing the real-valued observations into a fixed number of bins so they can be modeled as categorical distributions too. Similar to DeepAR, WaveNet generates probabilistic forecasts by autoregressively sampling different trajectories. In a recent study[71], WaveNet was applied to forecast resting-state EEG signal and was observed to outperform the classical AR model.

A.2.4 Foundation models

1. **Chronos[48].** Chronos is a large language model (LLM)-based method for forecasting univariate time series, using the T5 model[72] as backbone. The idea is that, upon proper tokenization, time series can be forecast in the same way as text using LLM. Since LLMs work with a finite dictionary of tokens (i.e., words), Chronos discretizes a real-valued time series into a fixed number of bins, and is trained using the cross-entropy loss (similar to the adapted WaveNet). During prediction, the original Chronos model proposed in [48] follows the autoregressive one-step-ahead forecasting strategy, similar to DeepAR. The later version, Chronos-Bolt[73], adopts the direct multi-step-ahead strategy, which generates estimates of quantiles $0.1, \ldots, 0.9$ of all predicted steps simultaneously using a feedforward network applied to decoder output. However, similar to TFT, Chronos-Bolt also faces the problem of quantile crossing[69].

   As with other LLMs, Chronos has been trained on a huge amount of data. Its training set consists of 28 real-world datasets with about 890K univariate time series spanning multiple domains (weather, finance, transportation, etc). It is trained with maximum context length of 512 and maximum forecast horizon of 64 time steps. Because of its rich training history, Chronos may be capable to perform zero-shot forecasting on time series that have never been seen by the model before. Nonetheless, since the majority of the time series in the training set of Chronos is of low frequency (sampled at hourly or lower rate), it is unclear whether the pretraining of Chronos can actually benefit it in forecasting widefield data.

2. **Moirai[49].** Moirai is another LLM-based time series forecasting model. Its main differences from Chronos are that it can handle time series with covariates and does not require discretizing time series into categorical variables. To achieve the former, it flattens the multivariate time series (e.g., a matrix of shape $d \times T$) into one sequence (e.g. a vector of length $d \times T$), while keeping track of the variate identities and time steps. Also, similar to PatchTST, instead of defining each time step as a separate token input to the underlying transformer model, Moirai forms "patches" by grouping nearby time steps of the same variate, and take each patch as a token. The patch size is set based on the frequency of the data. To model time series with diverse distributions of values without mapping them to categorical variables, Moirai is trained by maximizing the log-likelihood with respect to a mixture of probability distributions:

$$l(\boldsymbol{\theta}) = \log p(y_{T+1}, \ldots, y_{T+h} | f_{\boldsymbol{\theta}}(y_{T-C}, \ldots, y_T)), \text{with } p(\cdot) = \sum_{i=1}^{4} w_i p_i(\cdot),$$

where the $w_i's$ are also learnable parameters and the $p_i's$ are taken to be the Student's t, negative binomial, log-normal, and normal distributions. Probabilistic forecasts can be generated by sampling from this mixed distribution. Just like Chronos, Moirai has also been trained on a huge amount of data with a total of 27 billion observations, and thus may be capable of zero-shot forecasting. During training, Moirai uses samples with randomized history and prediction length to further improve its applicability to diverse forecasting problems.

A.3 Metrics

Let $y_{i,t}$ denote the true activity at t-th forecast step in the i-th test sample, $y_{i,t} \sim D_{i,t}$, $t = 1, \ldots, H, i = 1, \ldots, N$. Let $f^q_{i,t}$ denote the predicted q-th quantile of $D_{i,t}$.

1. **Mean Weighted Quantile Loss (MWQL):** When aggregating across all test samples and prediction steps (Figure 1c), we computed

$$\text{MWQL} = \frac{\sum_{i=1}^{N} \sum_{t=1}^{H} \frac{1}{Q} \sum_q \rho_q(y_{i,t}, f^q_{i,t})}{\sum_{i=1}^{N} \sum_{t=1}^{H} |y_{i,t}|}.$$

When aggregating across all test samples for each prediction steps t (Figure 1d), we computed

$$\text{MWQL}_t = \frac{\sum_{i=1}^{N} \frac{1}{Q} \sum_q \rho_q(y_{i,t}, f^q_{i,t})}{\sum_{i=1}^{N} |y_{i,t}|}.$$

In both cases, $q = 0.1, 0.2, \ldots, 0.9$, $Q = 9$. ρ_q is the quantile score[62]:

$$\rho_q(y_{i,t}, f^q_{i,t}) = \begin{cases} 2(1-q)(f^q_{i,t} - y_{i,t}), \text{ if } y_{i,t} < f^q_{i,t} \\ 2q(y_{i,t} - f^q_{i,t}), \text{ if } y_{i,t} \geq f^q_{i,t} \end{cases}$$

MWQL has been popularly used for evaluating probabilistic forecasts, e.g., in [69, 48, 49].

2. **Mean Scaled Interval Score (MSIS):**

$$\text{MSIS} = \frac{1}{N} \sum_{i=1}^{N} \frac{\sum_{t=1}^{H} W^{\alpha}_{i,t}(y_{i,t}, u^{\alpha}_{i,t}, l^{\alpha}_{i,t})}{\sum_{t=1}^{H} |y_{i,t}|}$$

Here $[l^{\alpha}_{i,t}, u^{\alpha}_{i,t}]$ define a $100(1-\alpha)\%$ prediction interval for $y_{i,t}$, typically $l^{\alpha}_{i,t} = f^{\frac{\alpha}{2}}_{i,t}$, $u^{\alpha}_{i,t} = f^{1-\frac{\alpha}{2}}_{i,t}$. W^{α} is the Winkler score[62], which combines penalties for the width of the interval and for lack of coverage:

$$W^{\alpha}(y_{i,t}, u^{\alpha}_{i,t}, l^{\alpha}_{i,t}) = (u^{\alpha}_{i,t} - l^{\alpha}_{i,t}) + \frac{2}{\alpha}(l^{\alpha}_{i,t} - y_{i,t}) \mathbb{1}(y_{i,t} < l^{\alpha}_{i,t}) + \frac{2}{\alpha}(y_{i,t} - u^{\alpha}_{i,t}) \mathbb{1}(y_{i,t} > u^{\alpha}_{i,t})$$

In this paper we took $\alpha = 0.2$. [69, 49] used a similar metric to evaluate prediction intervals, where the denominator is the absolute seasonal error. Since seasonality is not obvious in neural time series, we used the absolute target value instead.

3. **Mean Absolute Error (MAE) and Mean Squared Error (MSE):** When aggregating across all test samples and prediction steps (Figure 2), we computed

$$\text{MAE} = \frac{\sum_{i=1}^{N}\sum_{t=1}^{H}|y_{i,t}-\hat{y}_{i,t}|}{\sum_{i=1}^{N}\sum_{t=1}^{H}|y_{i,t}|},\ \text{MSE} = \frac{\sum_{i=1}^{N}\sum_{t=1}^{H}|y_{i,t}-\hat{y}_{i,t}|^2}{\sum_{i=1}^{N}\sum_{t=1}^{H}|y_{i,t}|^2}$$

When aggregating across all test samples for each prediction steps t (Figures 3,4), we computed

$$\text{MAE}_t = \frac{\sum_{i=1}^{N}|y_{i,t}-\hat{y}_{i,t}|}{\sum_{i=1}^{N}|y_{i,t}|},\ \text{MSE}_t = \frac{\sum_{i=1}^{N}|y_{i,t}-\hat{y}_{i,t}|^2}{\sum_{i=1}^{N}|y_{i,t}|^2}$$

Here $\hat{y}_{i,t}$ is a point forecast of $y_{i,t}$, and is taken to be the median value of probabilistic forecasts, i.e., $f_{i,t}^{0.5}$. MAE was also used in [56], where it was called Normalized Deviation.

4. **Correlation:** We computed the Pearson's correlation coefficient r between $y_i = \{y_{i,t}\}_{t=1,\dots,H}$ and $\hat{y}_i = \{\hat{y}_{i,t}\}_{t=1,\dots,H}$ for each test sample. The median r across of all test samples is reported is Figure 2.

A.4 Model implementation and training details

For the **Naive** and **Average** baselines, we used the implementations from the StatsForecast package [74]. These models have no trainable parameters or hyperparameters.

For the **AR** model, we considered two order–selection strategies:

1. AICC-based (**AR(aicc)**): we used `AutoARIMA` from StatsForecast, which by default uses the Hyndman-Khandakar stepwise search algorithm to find the model order and parameter achieving the lowest AICC[62]. Models were fitted using both training and validation sets. We set the limit on the number of models to explore to 100.
2. Validation-based (**AR(valQL)**): we used `AutoRegressive` from StatsForecast. Starting from lag order 1, we fitted the model on the training set and computed its MWQL on the validation set. The lag order was increased by one until validation MWQL failed to improve for 10 consecutive orders; the best-performing model so far was then chosen and evaluated on the test set.

For **ARIMA**, we again used `AutoARIMA` from StatsForecast. The AR order (p) was initialized with the optimal order found by **AR(aicc)**, and the MA order (q) was initialized to 0. Models were fitted using both training and validation sets. We set the limit on the number of models to explore to 100.

For **Theta**, we fitted four variants provided in StatsForecast (standard, optimized, dynamic standard, and dynamic optimized) on the training set. The variant with the lowest validation MWQL was evaluated on the test set.

For **AR-HMM**, we used `LinearAutoregressiveHMM` from the Dynamax package[75]. We fitted models with number of states $S \in \{2,3,4,5,6,7,8,9,10\}$ and number of lags $L \in \{1,2,4,6,\dots,40,44,48,\dots,80\}$ on the training set, and evaluated their log likelihoods of the validation data (LL_{val}). The optimal S was chosen as the smallest value whose LL_{val} was within 0.1% of the maximum across all models (i.e., $\geq 1.001 \times \max \text{LL}_{\text{val}}$, since LLs are negative). The optimal L was chosen as the smallest value whose LL_{val} was within 0.1% of the maximum across all lags at the selected S. We then fitted models with the optimal (S, L) with 5 random seeds on the training set and evaluated them on the test set. For each test sample, we generated 100 forecast trajectories to compute prediction intervals.

For **Global** models, we used the implementation from GluonTS[68]. Some models (**DLinear**, **PatchTST**, **TiDE**) was originally proposed for point forecast, but were adapted to generate probabilistic forecasts by GluonTS. See the **DLinear** section in A.2 for an example of such adaption. All models assumed Student-t output distributions. During training, all models employed early stopping based on validation loss, terminating if no improvement occurred for 10 epochs. Since widefield data does not clear seasonality, we disabled automatically added time features (e.g., hour of day, day of week) in **TiDE** and **WaveNet**. Hyperparameters were tuned by random search[76]: for each model, 40 hyperparameter configurations were randomly sampled from a grid, and the optimal configuration was selected as the one with the lowest validation MWQL. Finally, we trained models

with the optimal hyperparameter configuration using 5 random seeds, and evaluated them on the test set. Table 1 lists the names and candidate values of the hyperparameters for each model. Note that the values of `context_length` include all candidate values for all prediction lengths, but the candidates may vary across prediction lengths: for each session, we first performed hyperparameter search for forecasting 35 steps. Depending on the validation results, we may adjust the candidates for forecasting 70 and 18 steps. The candidate values of all other hyperparameters were the same across all sessions and prediction lengths.

For **Chronos**, we used `chronos-bolt-base` from AutoGluon[77]. For fine-tuning, we fixed `fine_tune_batch_size` $= 128$, `fine_tune_lr` $= 10^{-5}$, and varied `fine_tune_steps` $\in \{100, 200, 400, 600, 800, 1000\}$. The configuration achieving the lowest validation MWQL was then evaluated on the test set. Fine-tuning Chronos yielded identical results across random seeds, likely due to an issue in the AutoGluon implementation.

For **Moirai**, we used `moirai-1.0-R-base` from the official implementation of [49]. Since **Moirai** randomly samples prediction and context length during training, for zero-shot evaluation, we performed inference tuning as in [49]. Specifically, we varied context length $C \in \{50, 100, 250, 500, 750, 1000, 2000, 3000, 4000, 5000\}$ and patch size $P \in \{8, 16, 32, 64, 128\}$, and evaluated all pairs with the desired prediction horizon on the validation set. The pair with the lowest validation MWQL was then evaluated on the test set. Note that there is not a default patch size for the sampling frequency of widefield data, so we swept through all possible patch sizes of **Moirai**. 32 and 64 generally work the best. For fine-tuning, we fixed `batch_size` $= 64$ (due to memory constraint) and varied `learning_rate` $\in \{10^{-3}, 5 \times 10^{-4}, 10^{-4}, 5 \times 10^{-5}, 10^{-5}, 5 \times 10^{-6}, 10^{-6}\}$, while keeping other setting as default. The optimal learning rate was chosen as the one with the lowest validation loss when evaluated with the specific prediction length, `patch_sizes` $= [32, 64]$, and `context_lengths` $= [1000, 2000, 3000, 4000, 5000]$. The best hyperparameter configuration was then fine-tuned with 5 different random seeds. For each seed, we computed the validation MWQL across multiple (C, P) pairs as in the zero-shot setting. The pair with the lowest validation MWQL across 5 seeds was evaluated on the test set.

A.5 Supplementary figures

Table 1: Hyperparameter search space for GluonTS models.

Model	Hyperparameter	Values
DeepAR	lr	{2.5e-05, 5.0e-05, 7.5e-05, 1.0e-04, 7.5e-04}
	batch_size	{32, 64, 128}
	hidden_size	{64, 128, 256}
	num_layers	{1, 2, 3}
	context_length	{35, 70, 140, 175, 210, 245}
	lags_seq	{1,2, …, L-1}, where $L \in \{5, 10, 20, 30, 40\}$
PatchTST	context_length	{35, 70, 140, 175, 210, 245}
	patch_len	{24, 32, 40}
	stride	$s \times$ patch_len, where $s \in \{0.25, 0.5, 1\}$
	d_model	{32, 64, 128, 256}
	nhead	{1, 4}
	dim_feedforward	$r \times$ d_model, where $r \in \{1, 2\}$
	activation	'relu'
	num_encoder_layers	{1, 2, 3}
	lr	{0.0001, 0.00025, 0.0005, 0.00075, 0.001}
	batch_size	{128, 256, 512}
DLinear	context_length	{140, 175, 210, 245, 280, 315}
	hidden_dimension	{16, 32, 64, 128, 256}
	lr	{0.0001, 0.0005}
	kernel_size	{5}
	batch_size	{128, 256, 512}
TiDE	context_length	{35, 70, 105, 140, 175, 210}
	feat_proj_hidden_dim	4
	encoder_hidden_dim	{64, 128, 256, 512, 1024}
	decoder_hidden_dim	same as encoder_hidden_dim
	temporal_hidden_dim	{64, 128}
	distr_hidden_dim	{4, 8, 16, 32, 64, 128}
	num_layers_encoder	{1, 2}
	num_layers_decoder	{1, 2, 3}
	decoder_output_dim	{4, 8, 16, 32}
	dropout_rate	{0.3, 0.5}
	num_feat_dynamic_proj	2
	layer_norm	False
	lr	{1.e-05, 5.e-05, 1.e-04, 5.e-04}
	batch_size	{128, 256, 512}
TFT	context_length	{70, 105, 140, 175, 210, 245, 280, 315}
	quantiles	[0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95]
	num_heads	{1, 2, 4}
	hidden_dim	{64, 128, 256}
	variable_dim	same as hidden_dim
	dropout_rate	{0.2, 0.4, 0.6}
	lr	{0.00075, 0.001, 0.0025}
	batch_size	{128, 256, 512}
WaveNet	num_bins	{1024, 2048}
	num_residual_channels	{4, 8, 12, 24, 36, 48}
	num_skip_channels	{4, 8, 16, 32, 48, 64}
	dilation_depth	{2, 4, 8, 10}
	num_stacks	{1, 2, 3}
	lr	{1.e-05, 5.e-05, 1.e-04, 5.e-04, 1.e-03}
	batch_size	{32, 64, 128, 256}

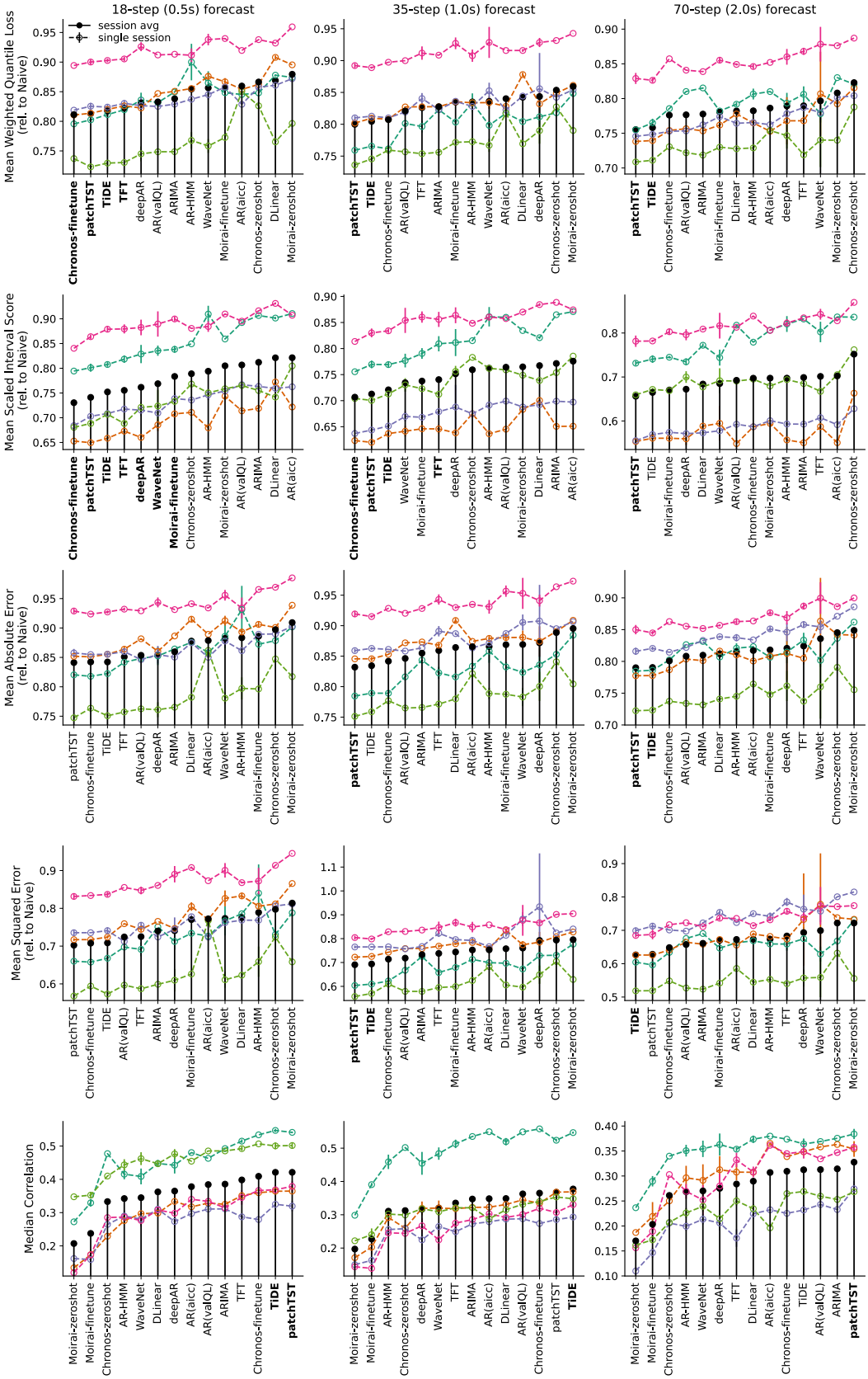

Figure 2: Evaluation on additional metrics. Models sorted by mean performance across sessions. Models in bold significantly outperform AR(valQL) (one-sided paired t-test, $p < 0.05$). For Chronos and Moirai, -FT indicates the finetuned version, -ZS indicates zeroshot version.

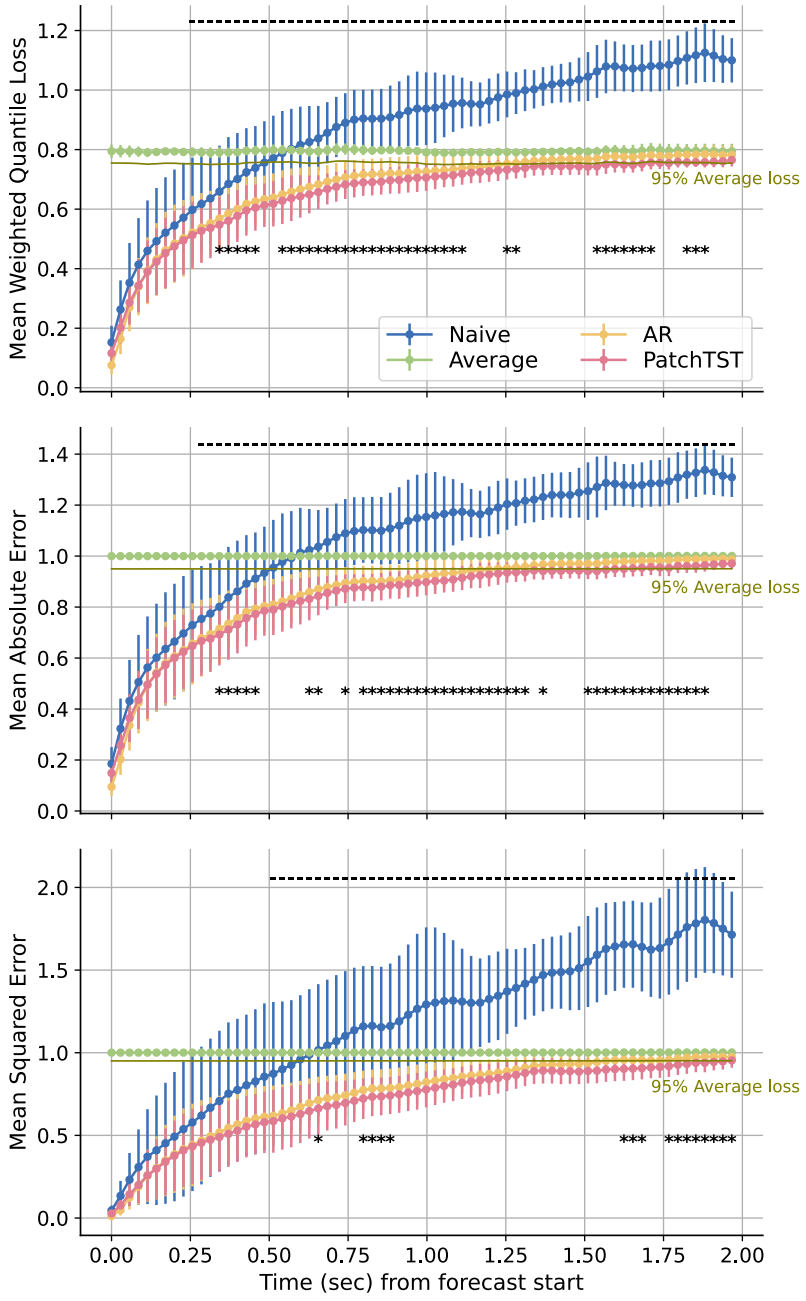

Figure 3: Additional metrics over prediction steps. Bars and stars indicate steps where PatchTST significantly outperforms Naive and AR, respectively (one-sided t-test, $p < 0.05$). PatchTST loss exceeds 95% of the Average model loss (solid line) after 1.80s (top), 1.65s (middle), 1.97s (bottom). AR loss exceeds 95% of the Average model loss after 1.28s (top), 1.23s (middle), 1.60s (bottom).

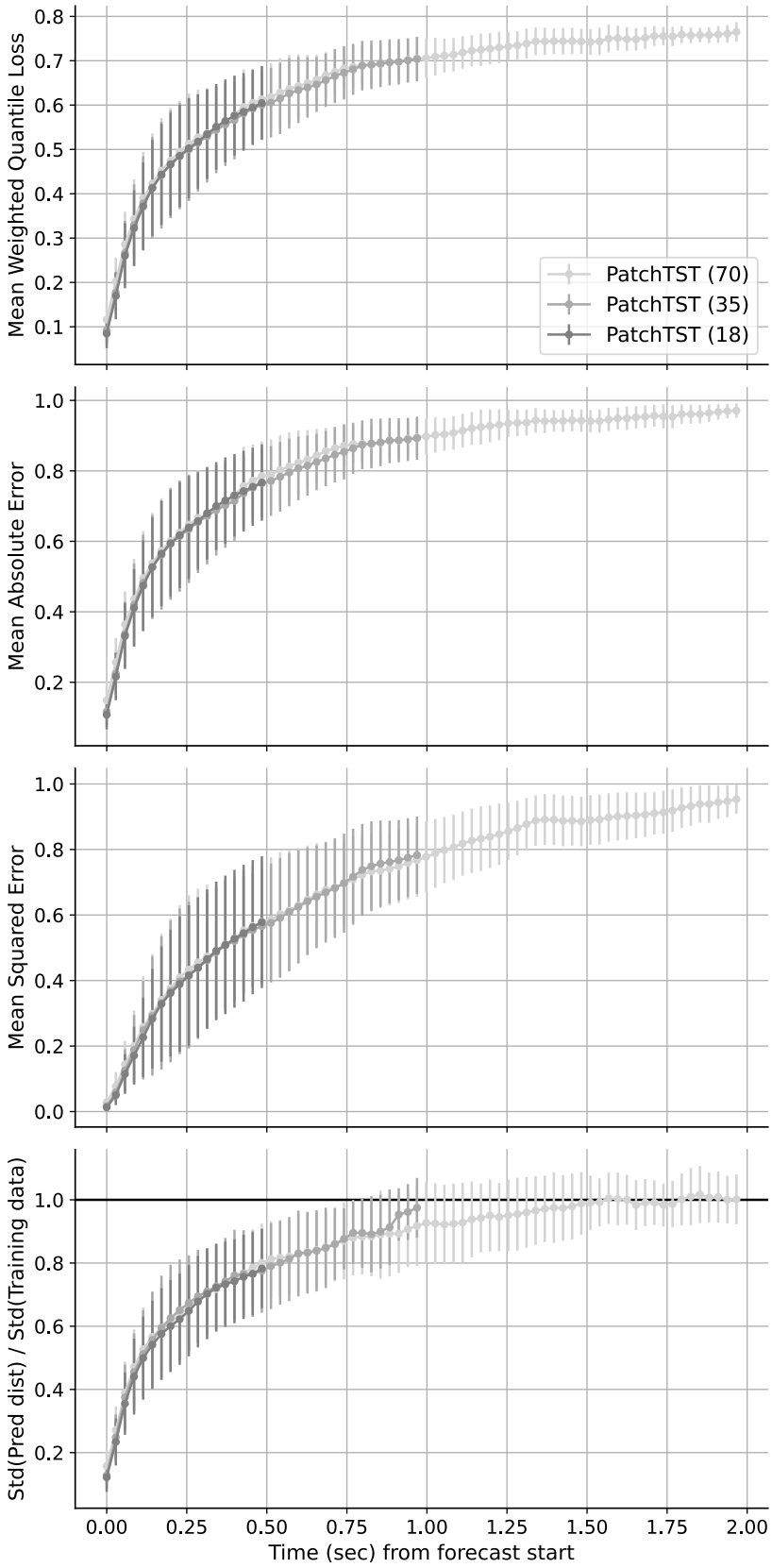

Figure 4: Performance across PatchTST models trained with different forecast horizons. PatchTST (18), PatchTST (35), PatchTST (70): PatchTST trained with forecast horizon of 18, 35, and 70 steps.