Forecasting Whole-Brain Neuronal Activity from Volumetric Video

Alexander Immer[1], Jan-Matthis Lueckmann[1], Alex Bo-Yuan Chen[2], Peter H. Li[1], Mariela D. Petkova[2], Nirmala A. Iyer[3], Aparna Dev[3], Gudrun Ihrke[3], Woohyun Park[3], Alyson Petruncio[3], Aubrey Weigel[3], Wyatt Korff[3], Florian Engert[2], Jeff W. Lichtman[2], Misha B. Ahrens[3], Viren Jain[1*], Michał Januszewski[1*]

[1]Google Research [2]Harvard University [3]HHMI Janelia

[*]Correspondence: `viren@google.com`, `mjanusz@google.com`

Abstract

Large-scale neuronal activity recordings with fluorescent calcium indicators are increasingly common, yielding high-resolution 2D or 3D videos. Traditional analysis pipelines reduce this data to 1D traces by segmenting regions of interest, leading to inevitable information loss. Inspired by the success of deep learning on minimally processed data in other domains, we investigate the potential of forecasting neuronal activity directly from volumetric videos. To capture long-range dependencies in high-resolution volumetric whole-brain recordings, we design a model with large receptive fields, which allow it to integrate information from distant regions within the brain. Our model outperforms trace-based forecasting approaches on ZAPBench, a benchmark on whole-brain activity prediction in zebrafish, demonstrating the advantages of preserving the spatial structure of neuronal activity. We perform extensive model selection, analyze effects of input resolution and spatio-temporal trade-offs, explore the impact of pre-training, provide comprehensive controls, and propose complementary performance metrics.

1 Introduction

Recent advances in imaging techniques have enabled the recording of neuronal activity at unprecedented resolution and scale. Light-sheet imaging allows recording of whole-brain activity for small animals, such as the larval zebrafish [1]. Raw recordings are in the form of volumetric videos, with hundreds of millions voxels per time step, recorded over hours. Typically, heavy postprocessing is applied to reduce dimensionality of this data down to 1D time traces of activity for distinct regions of interest representing individual neurons or clusters of cells [2]. Inspired by the success of deep learning models in analyzing minimally processed data in other fields, such as weather and climate forecasting [3, 4], we explore the potential of building predictive models directly on such volumetric videos, avoiding any information loss.

The ability to predict future behavior based on past observations is a cornerstone of scientific modeling across a diverse range of domains, ranging from physics to social sciences. The recently introduced Zebrafish Activity Prediction Benchmark (ZAPBench) [5] aims to apply this principle in the context of whole-brain activity in a vertebrate, taking advantage of datasets that can now be acquired with modern microscopy techniques. By comparing forecasts to actual measurements, ZAPBench provides a rigorous evaluation of predictive models of brain activity. ZAPBench [5] provides whole-brain larval zebrafish light-sheet microscopy recordings, in both trace (preprocessed) and volumetric (raw) formats. It poses the forecasting problem at the scale of tens of thousands of neurons.

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

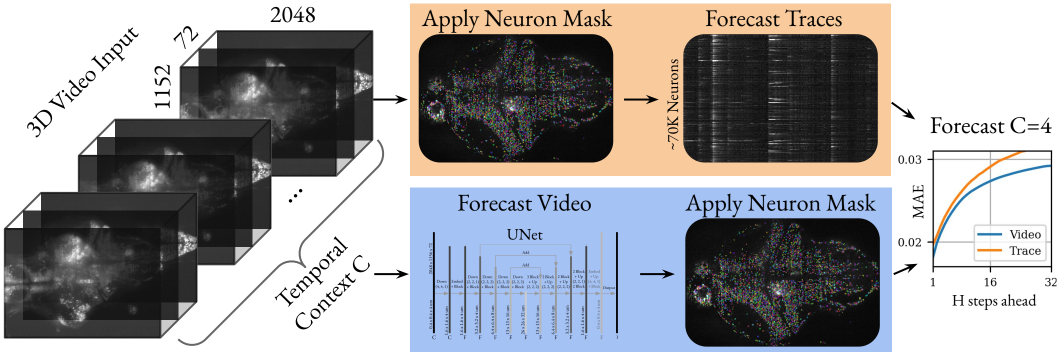

Figure 1: We propose to model light-sheet microscopy recordings of neural activity directly as volumetric video for forecasting instead of extracting and modelling neuron traces. Specifically, we train a model directly on the video and mask the output to optimize the per-neuron mean absolute error (MAE). We find that a UNet performs particularly well for small temporal context and can more effectively utilize spatial contextual information than trace-based time series models.

To test the viability of end-to-end forecasting on such data, we propose to use a video model based on techniques that have not been applied to this domain previously. Since processing in the brain is highly distributed [6, 7], we hypothesize that large receptive fields are important. Furthermore, in comparison to models applied to activity traces, we expect the following advantages. First, by utilizing the entire video as input, a video-based model is not reliant on the precision of neuron segmentation masks. Second, the inherent grid structure of the video preserves the spatial relationships between neurons, information that is otherwise lost during trace extraction. Finally, such a model can leverage potentially relevant visual cues present in the voxels between segmented cells or within the voxels of individual cell masks to enhance forecasting accuracy.

Building a model for this problem poses fundamental engineering challenges. Standard video models operate on 2D frames, and the presence of the additional Z dimension naturally complicates scaling. In ZAPBench, a single XY slice of a 3D frame has a native size of 2048×1328 pixels, and is thus comparable to a frame of a natural 1080p video. Every 3D frame is composed of 72 such slices, increasing the volume of the input data by up to two orders of magnitude relative to such videos and resulting in several hundreds of megabytes per frame.

For our model, we choose a variant of the UNet [8] and adapt it to 4D data. We develop a pipeline where both input and model are spatially sharded across multiple hosts and accelerators. To maintain a manageable size of the intermediate activations, we represent temporal input frames as channels. This approach allows us to explore the impact of varying spatial context by manipulating the receptive field while keeping computational cost (FLOPS) roughly constant.

We conduct extensive experiments to develop an effective video-based model for neuronal activity forecasting on ZAPBench. In our investigations, we analyze the impact of input resolution, spatial and temporal context on forecast accuracy. Notably, reducing input resolution by up to 4x can improve performance, and we observe a clear trade-off between spatial and temporal context. By exploiting spatial relationships in the video domain, our models achieve better forecast accuracy than trace-based approaches, particularly when only short temporal context is available.

On ZAPBench, we find that existing multivariate trace-based models do not significantly outperform univariate baselines, despite their ability to learn interactions between cells. In contrast, our proposed model, which is also briefly reported in ZAPBench [5], is the only multivariate approach that consistently outperforms univariate models on this benchmark. We further explore the effects of additional data for pre-training, which has shown to be effective in other domains [9, 10]: while pre-training on more specimens has negligible impact on accuracy, increasing the amount of data from the *same* animal yields measurable improvements. This insight informs the design of future datasets. Finally, we conduct detailed control experiments to understand model behavior and introduce a complementary, correlation-based performance metric for ZAPBench.

In summary, our contributions are as follows:

1. We propose to forecast zebrafish neuronal activity recorded using light-sheet microscopy directly in the native domain as volumetric video (3D + time).
2. We perform extensive model selection. We empirically show that input resolution can be reduced without sacrificing performance and quantify the impact of spatial (XYZ) and temporal context size for activity forecasting accuracy, finding a clear trade-off.
3. We show that on ZAPBench our model is the only approach that consistently benefits from multivariate information, and achieves leading performance for short temporal context.
4. We explore pre-training on additional volumetric videos finding that more data from other specimen has negligible impact on the results while more data from the same animal improves forecast accuracy.
5. We provide extensive controls to understand the performance of our model and propose complementary correlation-based metrics for the ZAPBench forecasting tasks.

2 Forecasting Neuronal Activity from Video

We propose to forecast neuronal activity in the ZAPBench dataset [5] directly in the volumetric video domain. Specifically, we utilize a temporal context of C video frames to predict the subsequent H frames. Per-neuron forecasts and loss are then computed by applying the segmentation mask to the predicted video frames. This contrasts with the traditional approach, which applies the segmentation mask to the original video data to extract activity traces before performing any forecasting. See Figure 1 for a comparison of these two approaches.

The ZAPBench dataset comprises high-resolution, whole-brain activity recordings of a larval zebrafish engaged in various behavioral tasks. Data was acquired using light-sheet fluorescence microscopy, enabling real-time imaging of neuronal activity at cellular resolution. This was made possible by using an animal genetically modified to express GCaMP [11], a fluorescent calcium sensor, in the nuclei of its neurons. ZAPBench provides both preprocessed activity traces for approximately 70,000 neurons and the corresponding raw volumetric video data. This raw data, denoted as $\mathbf{Y}$, has dimensions of $2048 \times 1152 \times 72 \times 7879$ (XYZT) and a resolution of $406\,\mathrm{nm} \times 406\,\mathrm{nm} \times 4\,\mu\mathrm{m} \times 914\,\mathrm{ms}$. We use a center crop of 1328 voxels in Y due to negligible cell activity in the border regions. Models forecasting $H = 32$ steps are benchmarked using short ($C = 4$) or long ($C = 256$) temporal context.

ZAPBench [5] preprocesses the raw volumetric video by aligning each frame to a reference volume for stabilization so that the neuron segmentation masks can be statically applied throughout the experiment. Further, a standard "$\Delta F/F$" normalization scheme is applied to the voxel intensities, with F denoting a baseline value [12, 13]. The normalized signal is in the $[-0.25, 1.5]$ range.

The neuron segmentation model is specifically trained for the dataset and yields 71,721 neurons. Formally, the segmentation mask can be considered as a mapping $\mathrm{seg} \colon \mathcal{N} \to 2^{\mathcal{S}}$ from integer identity of a neuron $\mathcal{N} = [71721]$ to a set of three-dimensional spatial indices, which is an element of the power set of index locations $\mathcal{S} = [2048] \times [1328] \times [72]$. The neuron activity at an arbitrary timestep t is then given by *averaging* the activity over spatial locations associated with each cell, i.e.:

$$\mathrm{y}_n(t) = \frac{1}{|\,\mathrm{seg}(n)|} \sum_{s \in \mathrm{seg}(n)} \mathbf{Y}_s(t). \tag{1}$$

While this is a natural choice, it loses information related to cell size, position and spatial distribution of intensities within it, and completely discards voxels that are not part of any segmentation mask or incorrectly segmented. Figure 2 depicts these potential issues.

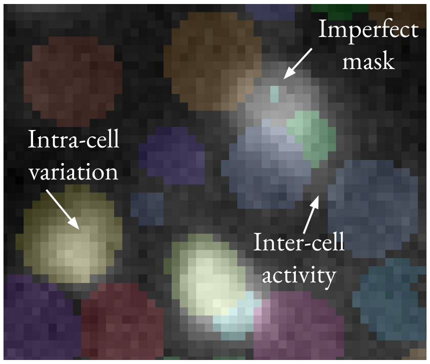

Figure 2: Illustration of potential loss of information when segmenting neurons. The colors show segmentation masks. A fragment of a 2d slice of the activity video is shown in greyscale.

We instead apply a video model to the raw input frames and directly forecast volumetric frames while optimizing and measuring the mean absolute error (MAE) on the segmented neurons. Prior work in neural response prediction [14, 15] has proposed additional metrics that explicitly take

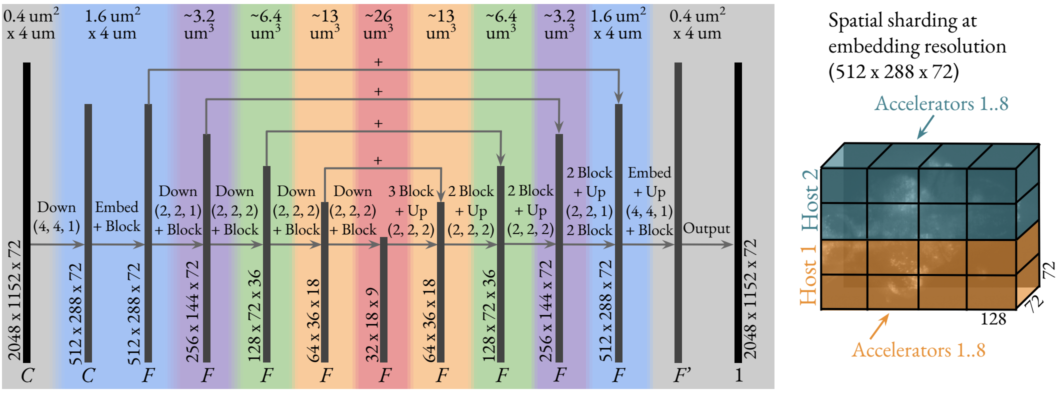

Figure 3: Architecture and input sharding overview. **A**: We use a variation of the UNet architecture [8] with 3D spatial input and treat the C input frames as channels. Further, we use a fixed number of features at every resolution to improve scalability. The network is conditioned on the time horizon H and outputs a single volumetric frame at a time, similar to MetNet-3 [4]. To control for spatial context at constant FLOPS, four blocks at the lowest resolution can be replaced by one block of higher resolution. **B**: Data loading and the network are spatially sharded and allow for flexible scaling to full resolution inputs.

trial-to-trial variability into account. The experimental setting used in ZAPBench did not allow for sufficiently numerous trial repetitions to make these metrics applicable, but we note them as an interesting direction to explore in future work if calcium recordings made with increased number of trials become available. In addition to MAE, we also report correlations between the predicted and actual activity in Sec. 4.6.

One frame of the volumetric video can be described as $\mathbf{Y}(t) \in \mathcal{V}$ with $\mathcal{V} = \mathbb{R}^{2048 \times 1152 \times 72}$. A video model with a P-dimensional weight vector $\mathbf{w} \in \mathbb{R}^P$ can then be denoted as $\mathbf{f} : \mathcal{V}^C \times \mathbb{R}^P \rightarrow \mathcal{V}^H$. That means the video model $\mathbf{f}$ receives a 4D volumetric input with C frames, outputs H frames, and is parameterized with weights $\mathbf{w}$. We obtain the prediction of the h-th frame as

$$\hat{\mathbf{Y}}(t, h) = \mathbf{f}_h(\mathbf{Y}(t - C + 1), \ldots, \mathbf{Y}(t), \mathbf{w}), \tag{2}$$

and denote by $\hat{\mathbf{y}}(t, h)$ the corresponding 1D trace vector computed using Eq. 1. For a fair comparison with trace-based models in ZAPBench we optimize the trace-based MAE $\mathcal{L}$ over all training timesteps T_{train} with respect to the model parameters $\mathbf{w}$

$$\mathcal{L}(\mathbf{w}) = \frac{1}{|T_{\text{train}}| \, H \, |\mathcal{N}|} \sum_{t \in T_{\text{train}}} \sum_{h \in [H]} \sum_{n \in \mathcal{N}} |\mathbf{y}_n(t + h) - \hat{\mathbf{y}}_n(t, h)| \, . \tag{3}$$

If we instead optimize the voxel-wise MAE, the models perform relatively worse when evaluated on the trace-based MAE because it corresponds to a different weighting of neurons by their size expressed in number of voxels.

3 Scalable Volumetric Video Architecture

Efficiently training models consuming high-resolution volumetric video of varying input context sizes C requires a scalable architecture and data loading system. We achieve this by extending a standard UNet architecture [8] to 4D by mapping temporal input context to features of the first convolutional layer, conditioning on lead-time to predict only single frames, and sharding both the model and the data loading process. Figure 3 shows the intermediate resolutions and representation sizes. The network comprises a series of pre-activation residual convolutional blocks [16] with fixed feature size $F = 128$, each with two group normalization layers [17] using 16 groups, Swish activation [18], and 3^3 convolutions for XYZ throughout.

3.1 Temporal Input Context as Features

Typically, video UNet variations use color channels as input features [19, 20] and convolve over frames using a temporal convolution [21]. This approach is intractable in our case because of the additional Z dimension. Instead, we treat the temporal input context of C frames as input features to the UNet. This confers the following advantages: 1) the temporal sizes of the input and output are decoupled, 2) the network parameter count is easily controlled, 3) representation sizes and computation requirements are reduced while using more features, and 4) early layers of the network have access to long-range temporal dependencies. Our model is similar to architectures used in standard time series models, which often treat temporal context as features [22, 23].

3.2 Varying the Receptive Field

We design a flexible UNet architecture that can adapt the receptive field while keeping the computational cost (FLOPS) fixed. We find that full native resolution is not necessary for optimal prediction accuracy (see Sec. 4.1), and thus downsample the input by a factor of 4 in XY using averaging. The first resampling block then uses a factor 2 in XY to achieve roughly isotropic resolution in XYZ, while the following ones downsample equally in all dimensions. We always use four residual blocks at the lowest resolution, and three at all other resolutions. This allows us to change the receptive field while keeping the FLOPS roughly fixed by removing the four lowest resolution blocks and instead adding one block to the respective next higher resolution. This is because one block at the higher resolution requires as many FLOPS as four blocks after downsampling by a factor of two in X and Y. In an ablation, we show that controlling for FLOPS is sensible because increasing the parameter count does not increase performance further (see Figure 7). A detailed calculation of the receptive field can be found in App. A.1.2.

3.3 Lead-time Conditioning

Rather than autoregressive or one-shot forecasting, we condition the network on a lead-time h, predicting a single frame, as proposed by [4] for weather forecasting. This reduces data I/O, avoids overfitting, and outperforms autoregression. Conditioning is applied via FiLM layers [24] with sinusoidal lead-time embeddings [25]. Figure 4 shows that directly predicting all H frames tends to overfit while lead-time conditioning performs equally well with both MAE and HL-Gauss [26], a distributional regression objective that results in slightly faster model convergence. However, in our experiments we use the conditioned MAE for its simplicity and because it does not require binning.

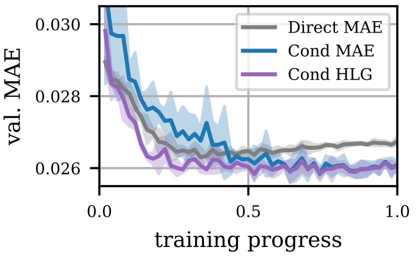

Figure 4: Comparison of direct MAE and lead-time conditioned variants.

3.4 Sharded Data Loading and Model

Despite the scalability features of the proposed UNet model for volumetric video, in practice applying it requires distributing the input and hidden representations across accelerators and machines. We train all models in Sec. 4 using a single sample per batch, noting that this can already correspond to several GBs of input data. We use spatial sharding in XY using the `jax.Array` API [27] so that each box in Figure 3B is handled by an individual accelerator. We also implement a custom data loader that distributes data loading across hosts so that each machine only loads the necessary subvolumes. To achieve this, we chunk our data in the `zarr3` format [28] and use the TensorStore API [29] to load and collate chunks. Our data loader follows the jax sharding automatically.

4 Experimental Results

We present experimental results evaluating the proposed volumetric video model on ZAPBench, a benchmark for whole-brain neuronal activity prediction for a larval zebrafish [5]. Uniquely, ZAPBench provides the raw volumetric recordings for most of the neurons in the brain enabling data-driven approaches like ours.

First, we empirically select and validate the architecture variant used for the benchmark. In particular, we investigate the effect of input resolution (Sec. 4.1), as well as the trade-off between temporal context C and spatial context in the form of the receptive field to assess the need for multivariate

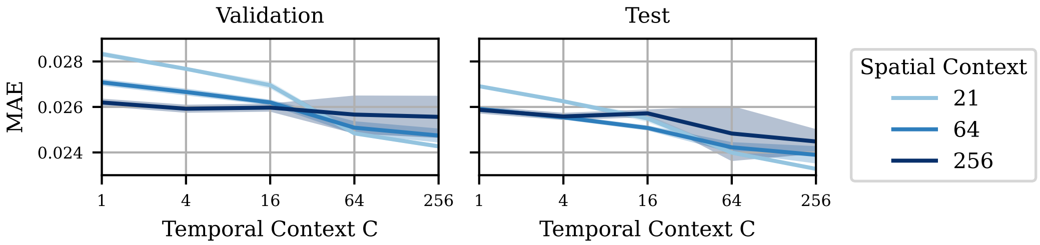

Figure 5: Validation and test performance for varying temporal context sizes C and spatial context sizes S with networks having comparable FLOPS. We find that there is a trade-off between spatial and temporal context with a cross-over point between $C = 16$ and $C = 64$, where spatial context stops being useful and leads to overfitting. The periodicity of many conditions is roughly 64, which might explain spatial context becoming redundant. We report the mean and two standard errors.

models (Sec. 4.2). We identify the model depicted in Figure 3 as a strong model for the short context size $C = 4$, where we achieve the best performance across the benchmark, as presented in Sec. 4.3. For the long temporal context $C = 256$, we only see an improvement of forecast accuracy in specific cases. We explore pre-training as a potential avenue to further improve results (Sec. 4.4), and provide in-depth analyses of the model's performance through extensive controls (Sec. 4.5). Finally, we propose a complementary, correlation-based metric (Sec. 4.6). More details are in App. A.2.

4.1 Effect of Input Resolution

We assess the relevance of input resolution when forecasting neuronal activity, and find that, surprisingly, predicting from a lower resolution performs best. We compare three variants of our model: a model that predicts from data 4× downsampled in XY, as depicted in Figure 3, one that downsamples only by factor 2, and one that is parameterized at full native resolution. Therefore, the full-resolution model loads and processes 16× more data. We achieve almost perfectly linear scaling by using proportionally more compute resources, maintaining the same throughput thanks to the sharded data input pipeline and model (see Sec. 3). In all cases, we scale the field of view of the network so that its size in physical units remains constant between experiments (see App. A.1.1).

For $C = 4$, Table 1 shows that the model with the lowest input resolution obtains a trace-based test MAE that is statistically identical with that of the model using intermediate resolution inputs. However, the full resolution model performs significantly worse. This suggests that, despite the short temporal context, input resolution does not play a major role in improving performance, and that the intracellular voxel-to-voxel variations in the recorded images do not carry information useful for forecasting, which might have applications to the design of future zebrafish activity recording experiments. We suspect that the decreased performance of the full resolution model could be caused by the significantly increased input voxel-to-parameter ratio while keeping the number of training examples fixed.

INPUT	TEST MAE
Downsample 4×	0.0268 ± 0.0002
Downsample 2×	0.0268
Full resolution	0.0273

Table 1: Increasing input resolution does not improve performance, and decreases it slightly at full resolution.

We thus downsample the volumetric frames by a factor of four in XY using averaging to 512×288×72. The segmentation mask is downsampled to the same shape using striding.

4.2 Spatial vs. Temporal Context

We use UNets with different numbers of downsampling blocks to vary the spatial context but keep the FLOPS fixed (see Sec. 3), and find that there is a trade-off between the spatial (S) and temporal (C) context. We compare models without any downsampling blocks, with two downsampling blocks, and with four. The models have spatial contexts S of 21, 64, and 256 in XY, respectively. Details on the architecture and computation of the receptive field can be found in App. A.1.3. Also note that the spatial context at full resolution of these models would be 4× higher. Figure 5 shows that a short temporal context requires larger spatial context to obtain optimal performance. For long temporal context, however, the models with large spatial context start to overfit and underperform. The effect becomes apparent between a temporal context of 16 and 64. This result suggests that video models are able to exploit multivariate information for short temporal context but provide little benefit for long context, where univariate models perform equally well [5].

4.3 Evaluation on ZAPBench

We evaluate the best-performing architectures on ZAPBench for both short and long context settings. The comparisons in this section are also included as a reference result in ZAPBench [5]. Any results outside this section, including the preceding model selection experiments, and all follow-up analyses and experiments in the next sections, are exclusive to this submission, significantly extending upon previously published work.

In Figure 6, we report the trace-based MAE versus forecasting steps in comparison to the best-performing trace-based models [5]. We average performance across test sets of different stimulus conditions. For trace-based models, TSMixer [23] achieves the best overall performance. We use the video-based architecture depicted in Figure 3 for the short context that has a spatial context of $1024 \times 1024 \times 72$ in XYZ, which is global except in X where it covers half the voxels. For the long temporal context, we use a model that does not downsample further than $(4, 4, 1)$ at the input, which we found in Sec. 4.2 to work best for this case. This model has a spatial context of $64 \times 64 \times 21$, which corresponds to $26\,\mu\mathrm{m} \times 26\,\mu\mathrm{m} \times 84\,\mu\mathrm{m}$ in XYZ.

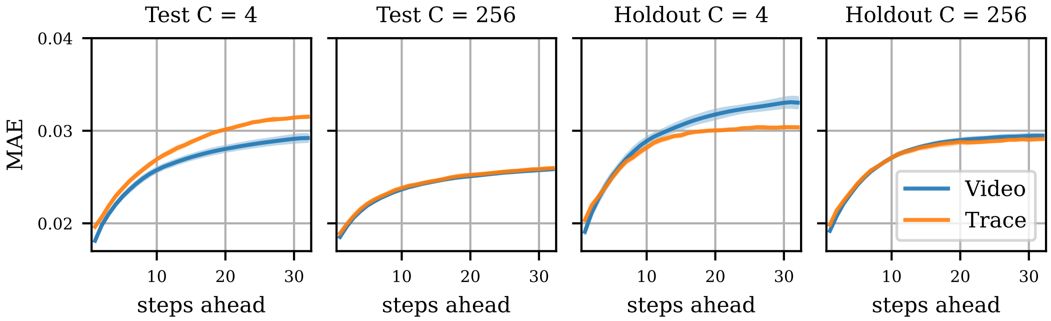

Figure 6: Comparison of volumetric video model with best-performing trace-based model for short (left) and long (right) context on the benchmark test set (averaged over eight conditions) and the experimental condition held out from the training data. We report the mean and two standard errors.

We find that the volumetric video models achieve the best performance in the short context $C = 4$ setting. For $C = 256$, there is no significant difference between the univariate trace-based model and the video model on the test set when evaluated with MAE, but the video model does improve correlation metrics (Sec. 4.6). This aligns with our observation in Sec. 4.2, where longer temporal context requires less spatial context for the same forecast accuracy. ZAPBench also holds out one stimulus condition entirely from training. We find that video models generalize better on this holdout condition for one-step-ahead forecasts but not for longer horizons. In App. A.3, we further show model performance separately for each experimental condition. For the short context, we find that the video model performs better in five, equally well in one, and worse in three out of the nine conditions.

More precisely, when evaluated with a context length $C = 4$ on both the test and holdout sets, the video model demonstrates a significant improvement in one-step-ahead forecasting accuracy, achieving about 8 (test) and 6 (holdout) percentage point reduction in error compared to the best performing trace-based model. With $C = 256$, the video model exhibits marginally superior performance in the first few forecasting steps, achieving up to a 2 percentage point reduction in MAE. Beyond the initial steps, both models demonstrate comparable accuracy on the test set.

4.4 Pre-Training on Other Specimens

We attempt to pre-train a model on other specimens recorded and preprocessed in a similar way to the zebrafish used for ZAPBench. We pre-train the model either on two additional specimens recorded in the same experimental session, or on these two and six more from two other sessions.

Because there is no segmentation available for the other specimens, the model is pre-trained using voxel-based MAE for 800k steps, and then fine-tuned on the ZAPBench dataset for 200k steps using the trace-based MAE. We use three different learning rates, 10^{-4}, 10^{-5}, and 10^{-7}, for fine-tuning, and select the best model by validation performance, which was obtained by fine-tuning with the lowest learning rate. Table 2 shows that pre-training with fine-tuning does not improve performance over standard training. However, training on $\sim$14% more data from the same specimen does improve performance significantly, suggesting that it may be useful to work towards prolonging activity recordings.

SETTING	TEST MAE $\pm$ 2 SE
Train	0.02573 ± 0.00005
Pre-train +2	0.02590 ± 0.00005
Pre-train +8	0.02591 ± 0.00001
Train + Val	0.02534 ± 0.00010

Table 2: Training on more data from the same specimen (“Train + Val”) improves performance more than pre-training and finetuning on others.

4.5 Impact of Unsegmented Voxels and Spatial Distribution of Calcium Signal

In contrast to video models, which analyze all voxels of the calcium movie, the trace extraction process ignores voxels that do not correspond to segmented cells. This potentially discards information that could be useful in forecasting. To test to which degree this is indeed the case, we trained the $C = 4$ video forecasting model with the unsegmented voxels set to constant value (0). The grand average test MAE for that model (0.0267 ± 0.0001) was not significantly different from that of the video model processing the complete volume (0.0267 ± 0.0003). This indicates that the unsegmented voxels are unlikely to contain information that could improve forecasts and that any gains relative to the trace-based models can be attributed to the utilization of the spatial distribution of the underlying calcium signals within the segmented cells.

The results in Figure 5 suggest that it is specifically the correlations between cells in the recorded fluorescence signals, rather than the distribution of signals within individual cells, that drives these improvements. To further test this hypothesis we rendered two "synthetic calcium movies": one ("rendered traces") with the voxels of the segmented cells set to the corresponding trace value (uniformly throughout each cell), and one ("shuffled traces") with the traces randomly reassigned to different cells. Training video models on this data, we observed that the model using "rendered traces" performs equivalently to the ones using the full $\Delta F/F$ and the segment-masked $\Delta F/F$ volumes (test MAE of 0.0267 ± 0.0001). The "shuffled traces" variant however showed significantly worse test MAE (0.0272 ± 0.0001). This provides additional evidence that the distribution of fluorescent signal within individual cells has negligible impact on forecasting accuracy, and that the additional accuracy of the video model stems for the utilization of multivariate, cross-cell information – precisely the type of information that trace-based models in ZAPBench have difficulty using.

4.6 Complementary Correlation-based Metrics

To better measure the quality of the temporal structures predicted by the model, we also computed two types of correlation metrics $\mathrm{Corr_W}$ and $\mathrm{Corr_H}$, which compare recorded and predicted activity over $H = 32$ steps, with the predictions assembled at constant lead time h or from a specific starting point t, respectively (see Figure 13 in the Appendix for explanation of these metrics). These metrics are complementary to MAE/MSE by being more sensitive to the temporal structure of the underlying activity and having a bounded range of $[-1, +1]$. The correlation metrics are broadly consistent with the MAE, except in the long-context regime $C = 256$ where the video model outperfoms the trace-based models.

Context	**Video**	**Trace**
$C = 4$	$\mathbf{0.1495 \pm 0.0018}$	0.1155 ± 0.0056
$C = 256$	$\mathbf{0.1853 \pm 0.0003}$	0.1645 ± 0.0007

Table 3: Test set $\mathrm{Corr_H} \pm 2$ SE.

5 Conclusion

We presented a volumetric video approach for forecasting neuronal activity. Compared to trace-based baselines, our model leverages spatial relationships for more accurate short-context forecasts. Surprisingly, higher resolution and cross-specimen pretraining showed no gains, while more data from the same specimen improved performance. Future work should explore probabilistic and latent-space models to further enhance video forecasting.

References

[1] Elizabeth MC Hillman, Venkatakaushik Voleti, Wenze Li, and Hang Yu. Light-sheet microscopy in neuroscience. *Annual review of neuroscience*, 42(1):295–313, 2019.

[2] Waseem Abbas and David Masip. Computational methods for neuron segmentation in two-photon calcium imaging data: a survey. *Applied Sciences*, 12(14):6876, 2022.

[3] Stephan Rasp, Peter D Dueben, Sebastian Scher, Jonathan A Weyn, Soukayna Mouatadid, and Nils Thuerey. Weatherbench: a benchmark data set for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems*, 12(11):e2020MS002203, 2020.

[4] Marcin Andrychowicz, Lasse Espeholt, Di Li, Samier Merchant, Alexander Merose, Fred Zyda, Shreya Agrawal, and Nal Kalchbrenner. Deep learning for day forecasts from sparse observations. *arXiv preprint arXiv:2306.06079*, 2023.

[5] 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, Alyson Petruncio, Aubrey Weigel, Wyatt Korff, Florian Engert, Jeff Lichtman, Misha Ahrens, Michal Januszewski, and Viren Jain. ZAPBench: A benchmark for whole-brain activity prediction in zebrafish. In *The Thirteenth International Conference on Learning Representations*, 2025.

[6] Anne E. Urai, Brent Doiron, Andrew M. Leifer, and Anne K. Churchland. Large-scale neural recordings call for new insights to link brain and behavior. *Nature Neuroscience*, 25(1):11–19, 2022.

[7] Eva A Naumann, James E Fitzgerald, Timothy W Dunn, Jason Rihel, Haim Sompolinsky, and Florian Engert. From whole-brain data to functional circuit models: the zebrafish optomotor response. *Cell*, 167(4):947–960, 2016.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[9] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

[11] Hod Dana, Yi Sun, Boaz Mohar, Brad K Hulse, Aaron M Kerlin, Jeremy P Hasseman, Getahun Tsegaye, Arthur Tsang, Allan Wong, Ronak Patel, et al. High-performance calcium sensors for imaging activity in neuronal populations and microcompartments. *Nature methods*, 16(7):649–657, 2019.

[12] Yu Mu, Davis V Bennett, Mikail Rubinov, Sujatha Narayan, Chao-Tsung Yang, Masashi Tanimoto, Brett D Mensh, Loren L Looger, and Misha B Ahrens. Glia accumulate evidence that actions are futile and suppress unsuccessful behavior. *Cell*, 178(1):27–43, 2019.

[13] Yan Zhang, Márton Rózsa, Yajie Liang, Daniel Bushey, Ziqiang Wei, Jihong Zheng, Daniel Reep, Gerard Joey Broussard, Arthur Tsang, Getahun Tsegaye, et al. Fast and sensitive gcamp calcium indicators for imaging neural populations. *Nature*, 615(7954):884–891, 2023.

[14] Oliver Schoppe, Nicol S Harper, Ben DB Willmore, Andrew J King, and Jan WH Schnupp. Measuring the performance of neural models. *Frontiers in computational neuroscience*, 10:10, 2016.

[15] Santiago A Cadena, George H Denfield, Edgar Y Walker, Leon A Gatys, Andreas S Tolias, Matthias Bethge, and Alexander S Ecker. Deep convolutional models improve predictions of macaque v1 responses to natural images. *PLoS computational biology*, 15(4):e1006897, 2019.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 630–645. Springer, 2016.

[17] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[18] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[19] Zhangyang Gao, Cheng Tan, Lirong Wu, and Stan Z Li. Simvp: Simpler yet better video prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3170–3180, 2022.

[20] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

[21] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

[22] 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.

[23] Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O Arik, and Tomas Pfister. TSMixer: An All-MLP architecture for time series forecasting. *Transactions on Machine Learning Research*, 2023.

[24] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[26] Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taiga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, et al. Stop regressing: Training value functions via classification for scalable deep rl. In *Forty-first International Conference on Machine Learning*, 2024.

[27] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[28] Alistair Miles, Jonathan Striebel, and Jeremy Maitin-Shepard. Zarr v3. `https://zarr.readthedocs.io/en/stable/spec/v3.html`, 2023.

[29] TensorStore developers. Tensorstore: Library for reading and writing large multi-dimensional arrays., 2024.

[30] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. In *European conference on computer vision*, pages 459–479. Springer, 2022.

[31] Ilya Loshchilov, Frank Hutter, et al. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 5, 2017.

[32] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

A Appendix

A.1 Architectural Details

Every network has an embedding 3^3 convolutional layer mapping from temporal context C to F features, and an output convolutional layer mapping from F' (when upsampling) or F to 1 feature, giving the single lead-time conditioned forecast frame. In the downsampling pathway, we apply one convolutional block at every resolution. During symmetric upsampling, we use three convolutional blocks at the lowest resolution, and two for all higher resolutions. Before upsampling to super-resolution (i.e., resolution that is higher than that of the input), we use a convolution to map from F to F' features to reduce the size of intermediate representations. During super-resolution upsampling we use one convolutional block per resolution. Each convolutional block has a pre-activation residual design with the following chained layers: group normalization, swish activation, 3^3 convolution, group normalization, conditioning on lead time using a FiLM layer, swish activation, optional feature dropout (only used with rate 0.1 for $C = 256$), and lastly the second 3^3 convolution. The UNet-structure is realized by adding the representations obtained during sequential downsampling to the upsampled representation. The number of features at every resolution is fixed to $F = 128$, except for the super-resolution upsampling, where it is $F' = 32$.

A.1.1 Models for Different Input Resolutions

To investigate the influence of input resolution on model performance, we conducted a comparative analysis. We compared our primary model, which operates on data downsampled by a factor of 4 in the XY plane, with two alternative configurations: one employing a downsampling factor of 2, and another utilizing full-resolution input.

To ensure equitable comparison, the architectures of these models were kept broadly consistent, with necessary adjustments to accommodate the differing input resolutions, while maintaining a consistent full-resolution output frame. Specifically, for the model operating on $2\times$ downsampled input, we augmented the architecture with three additional blocks at the input resolution and removed one upsampling block, relative to the architecture depicted in Figure 3. In contrast, the model utilizing full-resolution input incorporated two initial downsampling blocks with factors $(2, 2, 1)$ and omitted any super-resolution components, resulting in a conventional UNet architecture.

A.1.2 Receptive Field Calculation

The receptive field along a dimension depends on the cumulative product of the downsampling factors and the number of convolutions at the lowest resolution,

$$\texttt{receptive\_field}_{\texttt{dim}} = \texttt{cum\_downsampling\_factor}_{\texttt{dim}} \times \texttt{num\_blocks} \times 4,$$

where the factor 4 is because every block has two convolutions, each of which increase the receptive field by two. For a network that does not downsample at all, as for example used in Sec. 4.2, to account for the input and output convolutions and the center voxel, we have to increase the receptive field size by five. Therefore, the architecture depicted in Figure 3 has a receptive field of $(1024, 1024, 128)$ in XYZ comparable to the size of the complete frame. We tried to further enhance the receptive field to cover the whole frame using a multi-axis vision transformer [30] at the lowest resolution, but did not observe any accuracy gains. For the output, we upsample twice to obtain the original resolution, and use one residual block per resolution, but with a reduced feature dimension of $F' = 32$ to keep hidden representations at a manageable size.

A.1.3 Spatial vs. Temporal Models

This study employs three distinct models based on the aforementioned design.

The first model, maintaining a consistent spatial dimension of 512×288×72, forgoes downsampling and upsampling blocks. It incorporates four processing blocks at this resolution, along with two convolutional layers at the input and output stages. The receptive field, calculated as $S = 1 + (4 \times 2 + 2) \times 2 = 21$, is determined by considering the central voxel and adding 2 for each 3^3 convolution.

The second model, downsamples the input data to 64×64×32 and has a receptive field of $S = 64$. This is derived from the cumulative downsampling factors of (4, 4, 2) in the X, Y, and Z dimensions, respectively, and applying Equation 4.

Similarly, the third model employs downsampling factors of $(16, 16, 8)$, resulting in a $256\times256\times128$ receptive field. This translates to a global receptive field along the Z-axis, a near-global receptive field along the Y-axis, and a receptive field encompassing half of the total extent along the X-axis.

A.1.4 Lead-Time Conditioning

For the results shown in Figure 4, we use three different losses: direct MAE, conditioned MAE, and conditioned HL-Gauss. Apart from the FiLM layers to condition on lead-time, the architecture is the same in all cases, with the exception of the last layer which maps from F to the output dimensionality. The output dimensionality for the direct MAE is the number of forecast timesteps H. For the conditioned MAE, it is simply 1, as also described in Figure 3. For the conditioned HL-Gauss loss, it is 32, which is equal to H, and each output corresponds to a discretized bin of the data range. The HL-Gauss loss transforms a real value by representing it as a weighted average of bin mean-values, for details see [26].

A.2 Hyperparameters

Unless stated otherwise, we train every model for 250k to 500k steps by optimizing the trace-based MAE with a batch size of 1 using the AdamW optimizer [31] using an initial learning rate of 10^{-4} decayed using a cosine schedule [32] to 10^{-7} and a weight decay factor of 10^{-5}. Due to their tendency to overfit, we use a dropout rate of 0.1 on the features for long-context models with $C = 256$. These hyperparameters were optimized on the validation set during development. We choose checkpoints based on the validation performance monitored during training. We present experimental results in terms of mean performance and report two standard errors over three random seeds that control data loading and parameter initialization. The only exception to this are the high resolution results presented in Sec. 4.1, where we only report a single result because of their compute requirements. Most individual training experiments use 16 A100 40GB GPUs.

A.3 Additional Experimental Results

On the right in Figure 7, we further show an ablation to confirm that the improvement of multivariate video models is due to increased receptive field and not because of using more parameters. In particular, in our experimental setup in Sec. 4.2 we keep FLOPS fixed instead of number of parameters. In the example on the right, we instead increase the width by a factor of two leading to an increase in FLOPS by a factor of 4 while keeping the receptive field fixed. We observe that increasing FLOPS at the same spatial context leads statistically to the same performance. Therefore, the performance improvement observed in Figure 5 is likely due to the increased receptive field, especially for short context. The example on the right is for the case of $C = 4$.

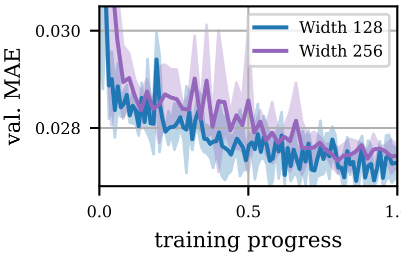

Figure 7: Ablation on increasing parameter count instead of receptive field.

In Figure 9 and 10, we show a fine-grained version of the benchmark of the trace and video-based models in the main text (Figure 6). The figures show the performance per experimental condition the fish was exposed to. For more details on these conditions, we refer to ZAPBench [5]. For short context, $C = 4$, we observe that the video-based model performs better on six experimental conditions, and worse for many steps ahead on the “dots”, “taxis”, and “open loop” conditions. For long context, the video-based model performs almost identically. As in the main text, we display two standard errors about the mean with shaded regions.

Figure 11 reports performance relative to four trace-based models included in ZAPBench. Figure 12 illustrates MAE differences for a few example frames.

Figure 8 further shows the performances in terms of the correlation-based metric, which is mentioned in Sec. 4.6 and visually explained in Figure 13.

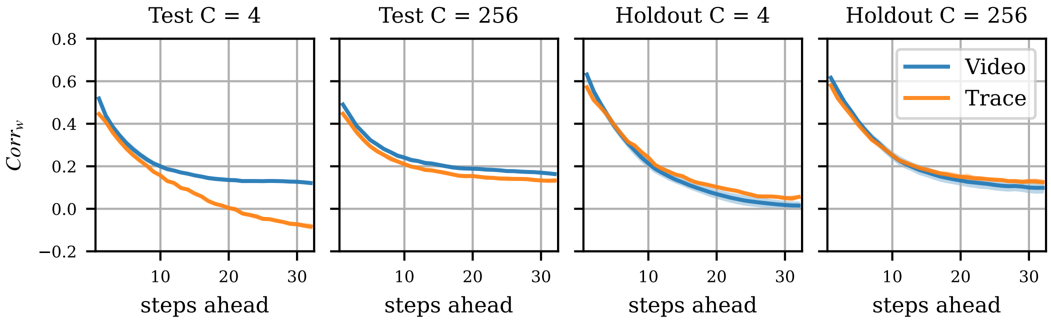

Figure 8: Comparison of volumetric video model with best-performing trace-based model in terms of Corr_W for short and long context on the benchmark test set (averaged over eight conditions) and holdout, higher is better. We report the mean and two standard errors.

A.4 Computational cost estimates

The loss ablation in Figure 4 required around 5k GPU hours, pre-training and fine-tuning as shown in Sec. 4.4 around 14k GPU hours, comparing spatial to temporal context in Sec. 4.2 around 50k GPU hours, and the final results including the ablation on input resolution another 30k GPU hours. This makes a total of roughly 100k GPU hours used for the experiments presented in the paper.

A single training run of the best performing video model for $C = 4$ required $36\,\mathrm{h}$, whereas the model for $C = 256$ required $120\,\mathrm{h}$, both using 16 A100 GPUs. This compute cost is two to three orders of magnitude higher than that incurred by training the baseline trace-based models, which require about $2\,\mathrm{h}$ on a single A100 GPU. However, video models require less raw data preprocessing relative to time series models, partially offsetting the increased cost.

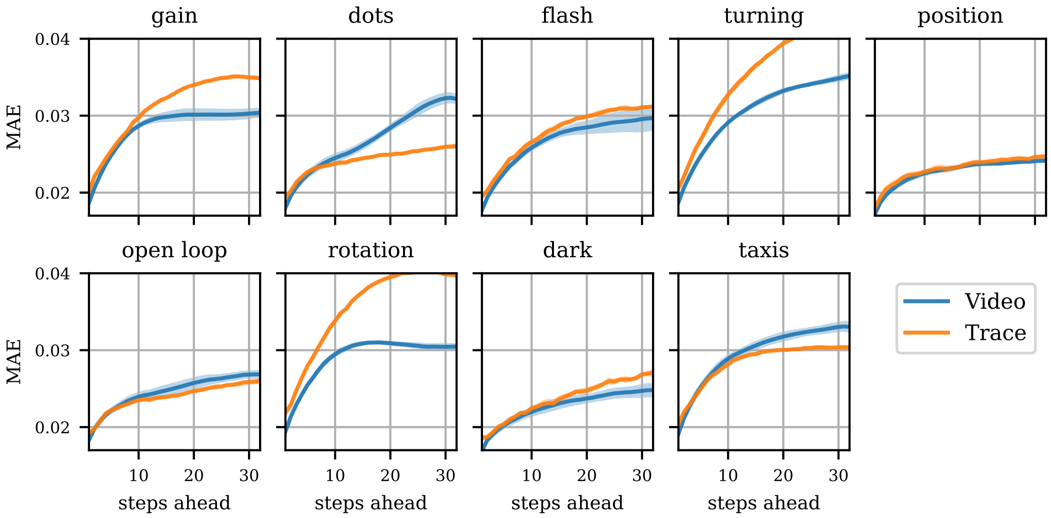

Figure 9: Comparison of volumetric video to best trace-based model on all conditions **for short context,** $C = 4$.

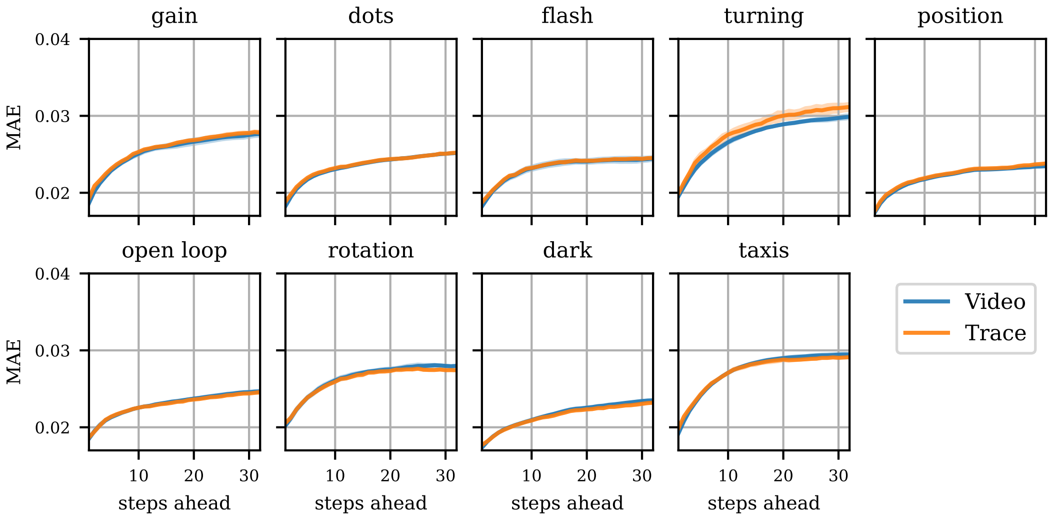

Figure 10: Comparison of volumetric video to best trace-based model on all conditions **for long context,** $C = 256$.

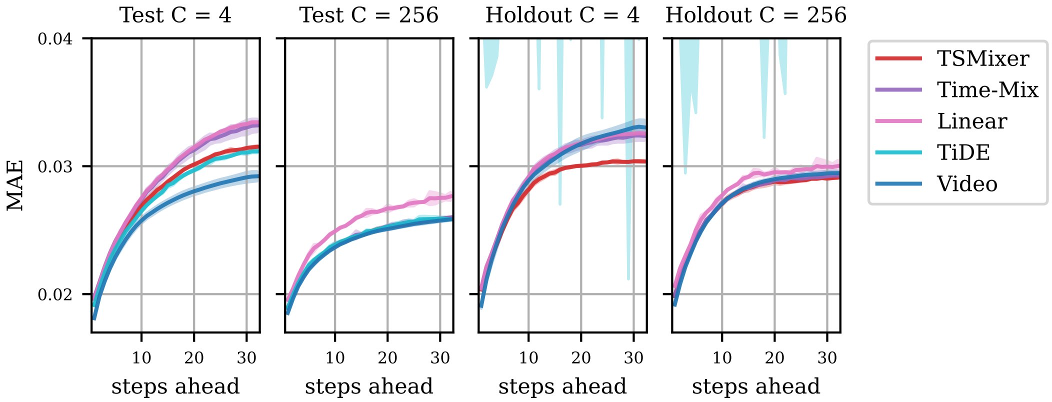

Figure 11: Comparison of volumetric video model with four trace-based models from ZAPBench [5] for short (left) and long (right) context on the benchmark test set (averaged over eight conditions) and the experimental condition held out from the training data. In remaining figures, we report performance relative to TSMixer. Note that MAEs of TiDE on the holdout are higher than the axis limits, which is due to its reliance on stimulus covariates. We report the mean and two standard errors.

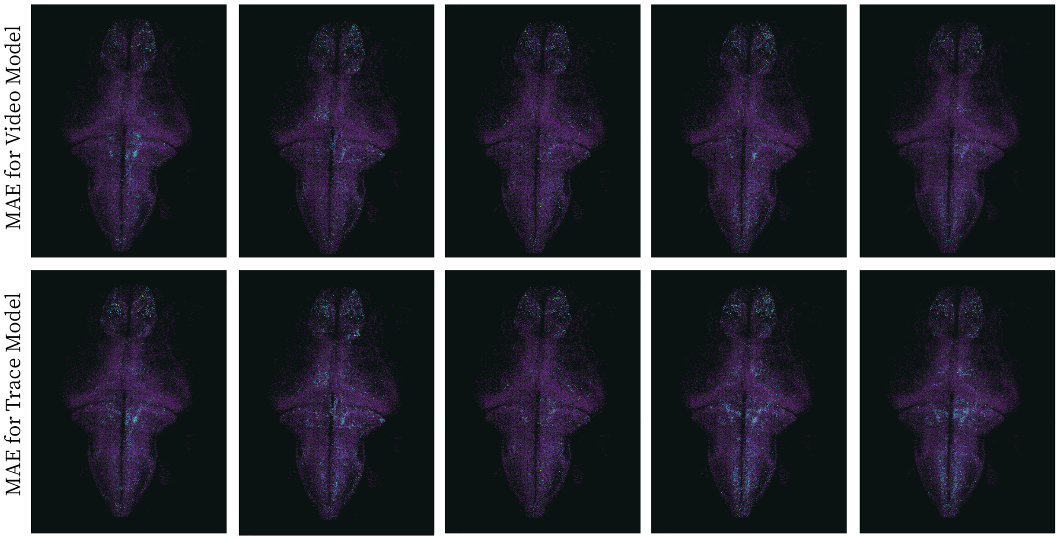

Figure 12: Illustration of MAE differences. Top row shows the MAE between target and predicted activity for a video model on five test set frames for the gain condition, $C = 4$, at 32 steps predicted ahead, with brighter colors indicating higher error. Bottom row shows corresponding MAEs on these frames for a trace-based model. When MAEs are averaged across all test set frames and neurons for this condition, the MAE difference between these models is approximately 0.005.

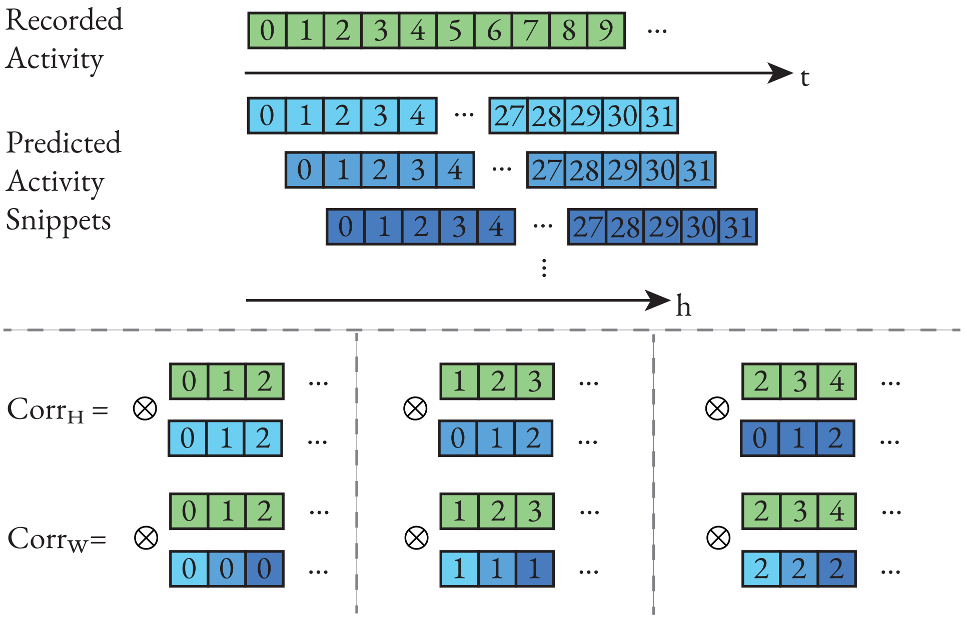

Figure 13: Illustration of the two types of correlation metrics (for a single neuron). Top: actually recorded activity (green) is aligned in experiment time t with predicted snippets (blue) of activity of length $H = 32$ starting from various offsets. Bottom: In $\mathrm{Corr_H}$, complete predicted snippets are correlated with the recordings, and then averaged over starting points. In $\mathrm{Corr_W}$, snippets are assembled from predictions at a specific lead time h, and correlated with the corresponding recordings. Reported metrics are averaged over all neurons.