nature communications

Article

https://doi.org/10.1038/s41467-025-64470-3

Modeling macroscopic brain dynamics with brain-inspired computing architecture

Received: 15 March 2025

Accepted: 18 September 2025

Published online: 24 October 2025

Check for updates

Zhong Zheng[1,5], Jing Wei[2,5], Yaru Xu[3], Chenhua Li[1], Tianying Lu[3], Qili Guo[3], Xueyao Ji[4], Hao Guo[3] ✉, Gang Wang[4] ✉ & Lei Deng[1] ✉

Understanding the brain requires modeling large-scale neural dynamics, where coarse-grained modeling of macroscopic brain behaviors is a powerful paradigm for linking brain structure to function with empirical data. However, the model inversion process remains computationally intensive and time-consuming, limiting research efficiency and medical deployment. In this work, we present a pipeline bridging coarse-grained brain modeling and advanced computing architectures. We introduce a dynamics-aware quantization framework that enables accurate low-precision simulation with maintained dynamical characteristics, thereby addressing the precision challenges inherent in the brain-inspired computing architecture. Furthermore, to exploit hardware capabilities, we develop hierarchical parallelism mapping strategies tailored for brain-inspired computing chips and GPUs. Experimental results demonstrate that the deployed low-precision models maintain high functional fidelity while achieving tens to hundreds-fold acceleration over commonly used CPUs. This work provides essential computational infrastructures for modeling macroscopic brain dynamics and extends the application of brain-inspired computing to scientific computing in neuroscience and medicine.

Brain functions like cognition and perception emerge from the collective activities of numerous neurons. Modeling and simulating large-scale neural activities is thus crucial for understanding brain behaviors[1]. There are different granularities of modeling methods that depend on the building blocks of the model. The most well-known and widely adopted method is fine-grained modeling, which utilizes a large number of microscopic neuron models[2–4] as nodes to construct neural circuits[5–8]. Theoretically, this method can simulate brain activities in great detail, but the vast number of neurons results in an enormous number of parameters that need to be determined[9]. Current brain imaging techniques struggle to acquire such large-scale microscopic dynamic data, making the neuron-based fine-grained modeling of the whole brain quite challenging.

In contrast, the coarse-grained modeling, based on the macroscopic dynamical models of collective neural behaviors, is more suitable for large-scale brain simulation at the current stage. Each node in this method represents the dynamics of a neuron population or a brain region, allowing the whole-brain modeling with significantly fewer nodes and parameters than those in the fine-grained modeling. More importantly, this modeling method can integrate macroscopic empirical data from multiple modalities[10–14], such as functional magnetic resonance imaging (fMRI), diffusion MRI (dMRI), T1-weighted MRI (T1w MRI), and electroencephalography (EEG), into a unified framework through model inversion (also termed as model identification, that is, model fitting of empirical data). In this way, it enables the construction of data-driven brain models that effectively bridge the gap between brain structures and functionalities, allowing for more precise interpretation and prediction of the brain. The mean field approximation, using closed-form equations to describe the brain dynamics, is the most common coarse-grained approach[15].

[1]Department of Precision Instrument, Center for Brain Inspired Computing Research (CBICR), Tsinghua University, Beijing, China. [2]School of Information, Shanxi University of Finance and Economics, Taiyuan, China. [3]College of Computer Science and Technology (College of Data Science), Taiyuan University of Technology, Taiyuan, China. [4]Beijing Institute of Basic Medical Sciences, Beijing, China. [5]These authors contributed equally: Zhong Zheng, Jing Wei.
✉e-mail: guohao@tyut.edu.cn; g_wang@foxmail.com; leideng@mail.tsinghua.edu.cn

Representative models include the Wilson-Cowan Model[16,17], the Kuramoto Model[18], the Hopf Model[19,20], the dynamic mean-field (DMF) model[21], and so forth. These models have significantly contributed to exploring neural mechanisms[22,23], understanding neural oscillation[24], investigating abnormal mechanisms of brain diseases and predicting therapeutic effects[25–27], and optimizing deep brain stimulation[28].

However, identifying macroscopic models of brain dynamics remains time-consuming due to heavy computational demands. The process requires continuous parameter adjustments and repeated simulations of long-duration brain dynamics. Moreover, current identification methods rely on general-purpose processors, but even high-end CPUs offer insufficient computing power. This is a common issue in the field of biophysics, where traditional processors struggle to meet the demands of the large-scale model inversion process. For the models of brain dynamics discussed here, the computational constraints not only hinder the research efficiency in the laboratory but also impact the potential for its medical applications in hospitals, such as brain disorder understanding and therapeutic interventions based on individualized brain models.

In recent years, advanced computing architectures have developed rapidly. Brain-inspired computing chips (also called neuromorphic computing chips) are the most influential ones. Representative brain-inspired computing chips include TrueNorth[29], SpiNNaker[30–34], BrainScaleS[35,36], Loihi[37,38], Darwin[39,40], Tianjic[41,42], etc. Drawing inspiration from the human brain, they typically adopt a decentralized many-core architecture, which can offer a large number of parallel computing resources, extremely high local memory bandwidth, and higher efficiency than intelligent accelerators based on the von Neumann architecture (such as general-purpose GPUs and neural processing units[43–48]). However, accelerating the aforementioned model inversion process on brain-inspired computing chips is not straightforward. First, brain-inspired computing chips are increasingly oriented to intelligent computing workloads[49,50] (i.e., artificial intelligence (AI) tasks such as pattern recognition, computer vision, and natural language processing, typically involving neural network computation) rather than scientific computing ones (e.g., numerical simulation, mathematical modeling, requiring high-precision floating-point computation for solving differential equations), so they prioritize low-precision computing to reduce hardware resource costs and power consumption[30,38,51,52]. Unfortunately, there have not yet been any attempts to implement such low-precision macroscopic brain models to our best knowledge. Existing low-precision quantization methods for neural networks cannot be directly applied to these dynamical models, which are characterized by large temporal variations in state variables, complex spatiotemporal heterogeneity, and the need for numerical stability across long-duration simulation periods. Second, brain-inspired computing architectures, influenced by fine-grained neuron models, are typically organized with neurons as the basic unit. Their functionality is primarily designed to support various neural network models, and most existing works only use them for neuron- or sub-neuron-level simulation[53–57]. The challenge of leveraging highly parallel architectural resources for coarse-grained brain modeling remains largely unexplored.

To bridge the gap between the intelligent computing and scientific computing paradigms, we propose a comprehensive pipeline enabling macroscopic brain dynamics models to benefit from modern computing architectures. Our pipeline addresses the challenges from two complementary perspectives. From the computational methodology perspective, we analyze the characteristics of brain dynamics models to design a dynamics-aware quantization framework for low-precision implementation. With the semi-dynamic quantization strategy, we can address the large temporal variations during the transient phase and achieve stable long-duration simulation of dynamic models using low-precision integers once the numerical ranges stabilize. We also observe that the state variables exhibit pronounced spatial heterogeneity across brain regions, while their temporal evolution demonstrates heterogeneity characterized by distinct timescales. Therefore, we further propose range-based group-wise quantization and multi-timescale simulation, enhancing the accuracy of low-precision models under such heterogeneity. In this way, we enable the majority of the model simulation process to be deployed on low-precision platforms. From the system engineering perspective, we propose architecture-aware hierarchical parallelism mapping strategies to exploit the parallel resources of advanced computing architectures. We start by introducing a population-based metaheuristic optimization algorithm to improve the parallelization potential of the costly model inversion. Then we map the multi-level parallelism of the algorithm to hierarchically parallel computing and memory resources on the architectures. Beyond brain-inspired computing chips, GPUs are also our target platform. Despite the widespread application of GPUs in medical image processing, such as MRI[58–62], the implementation for macroscopic brain dynamics is rare and has not fully exploited the architectural potential[63]. We develop specialized acceleration methods for model inversion based on the distinct parallelism organization of the two architectures. With analyses of the execution time on the GPU and the brain-inspired computing architecture, we aim to gain deeper insights into how architectural choices impact the execution performance of macroscopic brain dynamics.

Our experiments demonstrate that low-precision brain dynamics models, under the dynamics-aware quantization framework, maintain various characteristics close to their floating-point counterparts on multimodal neuroimaging datasets. The distribution of goodness-of-fit indicators in the parameter space obtained from low-precision models tightly matches that of full-precision ones. And we validate that the proposed low-precision methods can achieve reliable parameter estimation. We also evaluate the execution performance across different platforms, finding that both the GPU and the brain-inspired computing chip outperform the CPU, with the brain-inspired computing chip demonstrating the best results. Compared to high-precision simulations on the baseline CPU, our approach accelerates the parallel model simulation by 75–424 times on the TianjicX brain-inspired computing chip, reducing the entire identification time to only 0.7–13.3 min. The brain-inspired computing architecture also shows better scalability than the GPU, so it has greater potential for future macroscopic or mesoscopic models with potentially larger numbers of nodes. Our proposed pipeline promises the reduction of the research period in modeling macroscopic brain dynamics and paves the way for potential medical applications. Meanwhile, we also expand a broader application scope for brain-inspired computing, especially to scientific computing in neuroscience and medicine.

Results

Framework for accelerating macroscopic brain modeling with brain-inspired computing

Coarse-grained brain models based on macroscopic neural dynamics enable data-driven brain modeling using empirical data. The core workload in data-driven modeling is the model inversion. The macroscopic brain models typically contain multiple parameters, and the model inversion aims to find the parameter set that best matches empirical data. The model inversion process generally proceeds as illustrated in Fig. 1a. First, the empirical structural data are integrated into the model for simulation to generate simulated functional signals. Second, comparing these signals with empirical functional data allows evaluation of the current fit quality. Then, parameters are adjusted based on the current fit results, and the process returns to the simulation step. The entire model inversion process typically requires numerous iterations to find a near-optimal solution, making it extremely time-consuming.

We aim to accelerate the costly model inversion process using advanced computing architectures such as brain-inspired computing

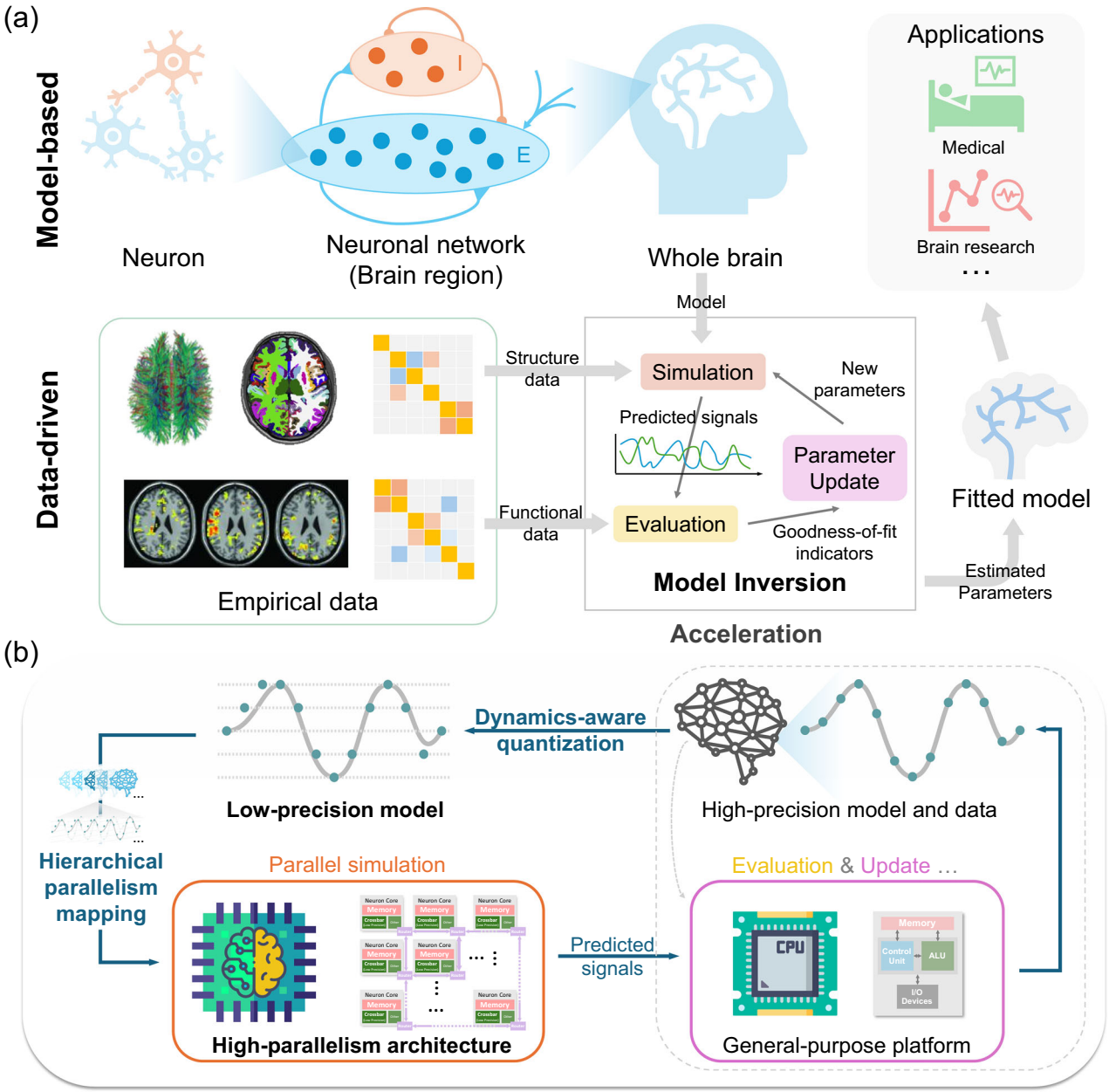


Fig. 1 | Framework of macroscopic brain dynamics modeling. a Coarse-grained modeling with multimodal data for empirically-grounded brain models. The macroscopic brain model, derived from population activities of microscopic neurons, integrates the brain structure data to simulate and predict functional signals. Model parameters are fitted to empirical functional data using inversion algorithms, producing data-driven personalized models for research and medical applications. The MRI example images are visualized using the MRtrix3 package[108], FreeSurfer[113], and Resting-State fMRI Data Analysis Toolkit (REST)[114], respectively. The brain icons were designed by meaicon from Flaticon. **b** Proposed pipeline for accelerating model inversion using brain-inspired computing architectures with low precision and high parallelism. The solution in the dashed box represents the current common practice, which is to perform model inversion using high-precision brain models based on the general-purpose platforms. The chip icon, the brain icon on the chip, and the CPU icon were designed by Freepik from Flaticon. The brain network icon was designed by imaginationlol from Flaticon.

chips and GPUs with high parallelism, thereby significantly improving the execution efficiency and expanding the applicability of macroscopic brain models. Our proposed acceleration pipeline is illustrated in Fig. 1b. First, at the model level, considering the precision constraints of brain-inspired computing architectures, we quantize high-precision models and data into low-precision ones. In this step, we comprehensively consider the characteristics of the dynamical models and propose a dynamics-aware quantization framework. Our methods ensure that the simulation results of low-precision models closely approximate those of high-precision counterparts. Second, at the hardware level, we accelerate the model simulation, the most time-consuming component of model inversion, by deploying it onto the highly parallel architectures. We design hierarchical parallelism mapping strategies, which parallelize the simulation workloads with a metaheuristic algorithm and map them onto the computing and memory resources based on architectural features. This approach achieves significant speed improvement compared to simulation on mainstream general-purpose platforms such as CPUs that are widely used in conventional brain modeling. Finally, we transfer the simulation results back to general-purpose platforms to complete steps that require less computation but higher generality, such as evaluation and parameter updates. Such a pipeline enables efficient execution of the

complete model inversion, thus greatly accelerating the macroscopic brain modeling on top of it.

Dynamics-aware quantization for macroscopic brain models

Brain-inspired computing chips favor low-precision integer types for multiplication and memory access to achieve higher efficiency. In recent years, model quantization methods have been developed to represent full-precision neural network models with integers[64–70]. However, such methods are not effective for dynamic systems. The AI-oriented quantization process usually focuses on outcomes rather than internal computational processes, contrasting with the simulation of dynamic systems like the brain models that emphasize precise calculation throughout the entire computational process. Since most neural networks are memoryless and static, with no or only a few state variables in the model, existing quantization methods may struggle to handle dynamical models with complex temporal dependencies and extremely long simulation sequences. Therefore, low-precision modeling is the primary challenge to enable macroscopic brain dynamics models to benefit from brain-inspired computing platforms. Our aim is to develop a dynamics-aware framework for low-precision implementation of macroscopic brain dynamics models. We specifically focus on the simulation of the DMF model and the hemodynamic model for the resting-state brain. The choice of the DMF model as the primary case study is based on its widespread use in whole-brain dynamics research[71–73] and its biophysically grounded foundation derived from microscopic spiking neuron models[74]. To further validate the generalization of our dynamics-aware quantization framework, we also conduct experiments over the Hopf model[19,20], which represents a phenomenological approach with different oscillatory dynamics compared to the fixed-point attractor behavior of the DMF model.

Quantization parameters are key to the accuracy of the low-precision model. For the most commonly used uniform quantization, we need to determine the scaling factor and zero point for each variable mapped to low-precision integers. The scaling factors on brain-inspired computing chips are usually static. This means that the parameters have to be determined before on-chip computation, and the quantized integers need to represent all possible values that may arise during the computation. Intuitively, the selection of the scaling factor is a trade-off between the range and the precision of the representation. Integers quantized with larger scaling factors can represent values of a wider range. But larger scaling factors may also lead to lower precision, as values within a larger interval might be mapped to the same integer. For brain simulation, it is difficult to balance the range and precision for the whole process due to the large temporal variations of the variable. Taking the resting-state DMF simulation as an example, the variable values can differ significantly between the initial and stable states. Additionally, the models with different parameters yield substantially diverse variable distributions. Hence, it is impossible to establish a single static set of quantization parameters that works well throughout the entire simulation process of a model, let alone across different models.

We introduce semi-dynamic quantization to address this problem. Most current macroscopic whole-brain models are developed based on the resting-state data. The simulation of the brain's resting-state activity can be divided into two stages. In the warm-up stage, the variables may start from an arbitrary state and experience a transient period with significant changes before converging to a relatively stable state. In the simulation stage, the model typically runs in a relatively stable state where the variables fluctuate randomly or oscillate within a certain range. These stages represent a common practice in the model simulation scenario, regardless of which specific model is employed. So we can just quantize the simulation stage to reduce the data range for higher integer precision, leaving the warm-up stage with high-precision floating-point values. A quantization parameter selection (QPS) stage can be added between the warm-up stage and the simulation stage, as shown in Fig. 2a. During this stage, the model should have already converged near a fixed point, but the computation is still performed using floating-point data. Based on the ranges of variables in this stage, we can calculate the scaling factor and the zero point (see "Methods"). In this way, the computation in the simulation stage can be "statically" quantized on the brain-inspired computing chip.

The applicability of semi-dynamic quantization depends on the numerical range of variables. As long as the ranges of variables and their derivatives of the model do not undergo substantial changes over time, low-precision quantization can remain effective. Previous literature empirically indicates that the resting-state functional connectivity (FC) does not exhibit extremely drastic transient changes (although some time-varying information might be extractable)[75]. Therefore, we believe that semi-dynamic quantization should be capable of meeting the computational demands of resting-state modeling of brain dynamics across various model types. It is also worth noting that the semi-dynamic quantization method partially depends on general-purpose processors for floating-point computation and quantization parameter calculation, but the overheads are acceptable, as the warm-up stage and the QPS stage are both much shorter than the simulation stage.

The spatial heterogeneity and temporal heterogeneity of brain models pose additional challenges in low-precision simulation. For example, in the hemodynamic model, variable distributions vary across regions spatially (Fig. 2b), while state variables fluctuate on different timescales temporally (Supplementary Fig. S1). To address spatial heterogeneity, we focus on improving the quantization method. The varying signal distributions across brain regions indicate different quantization parameter requirements for different regions, making heterogeneous quantization parameters a natural choice. However, most brain-inspired computing architectures only support coarse-grained quantization parameters (such as per-tensor quantization), as the computational overhead of element-wise quantization is prohibitive. Therefore, we propose a range-based group-wise quantization method. As illustrated in Fig. 2b, brain regions in a model are grouped according to their ranges of variable values, with each group using a separate set of quantization parameters. This approach balances the accuracy of the quantized model and the computational efficiency on specialized platforms.

The temporal heterogeneity problem is more complex. On one hand, the magnitude of the variation in a variable at each time step determines the upper bound of the quantization scaling factor, as the changes must be representable by at least an integer 1. On the other hand, the variable distribution determines the lower bound of the scaling factor, as the quantized variables must fit within the limited range of a fixed-width integer. For many variables in the brain dynamics model, the required upper bound may be lower than the lower bound, which means that no scaling factor s can be found to achieve effective quantization. However, we find that this problem can be approached from the simulation respective. In fact, as long as the Euler integration time step used in simulation exceeds a certain lower bound (determined jointly by the variable distribution, variable derivative distribution, and quantization bitwidth), the variable can be properly quantized (see "Methods"). Meanwhile, the stability condition of Euler integration constrains the upper bound of the time step. We find that using a homogeneous time step for different variables makes it difficult to meet these requirements. Therefore, we adopt a multi-timescale simulation approach, setting different simulation time step sizes for different variables, as shown in Fig. 2c. As long as the dependencies between variables are properly handled, this approach can improve quantization accuracy while reducing the computational load in simulation.

The low-precision DMF model preserves bifurcation characteristics

To validate the effectiveness of the low-precision model simulation, we first qualitatively compare the dynamical characteristics between the

https://doi.org/10.1038/s41467-025-64470-3

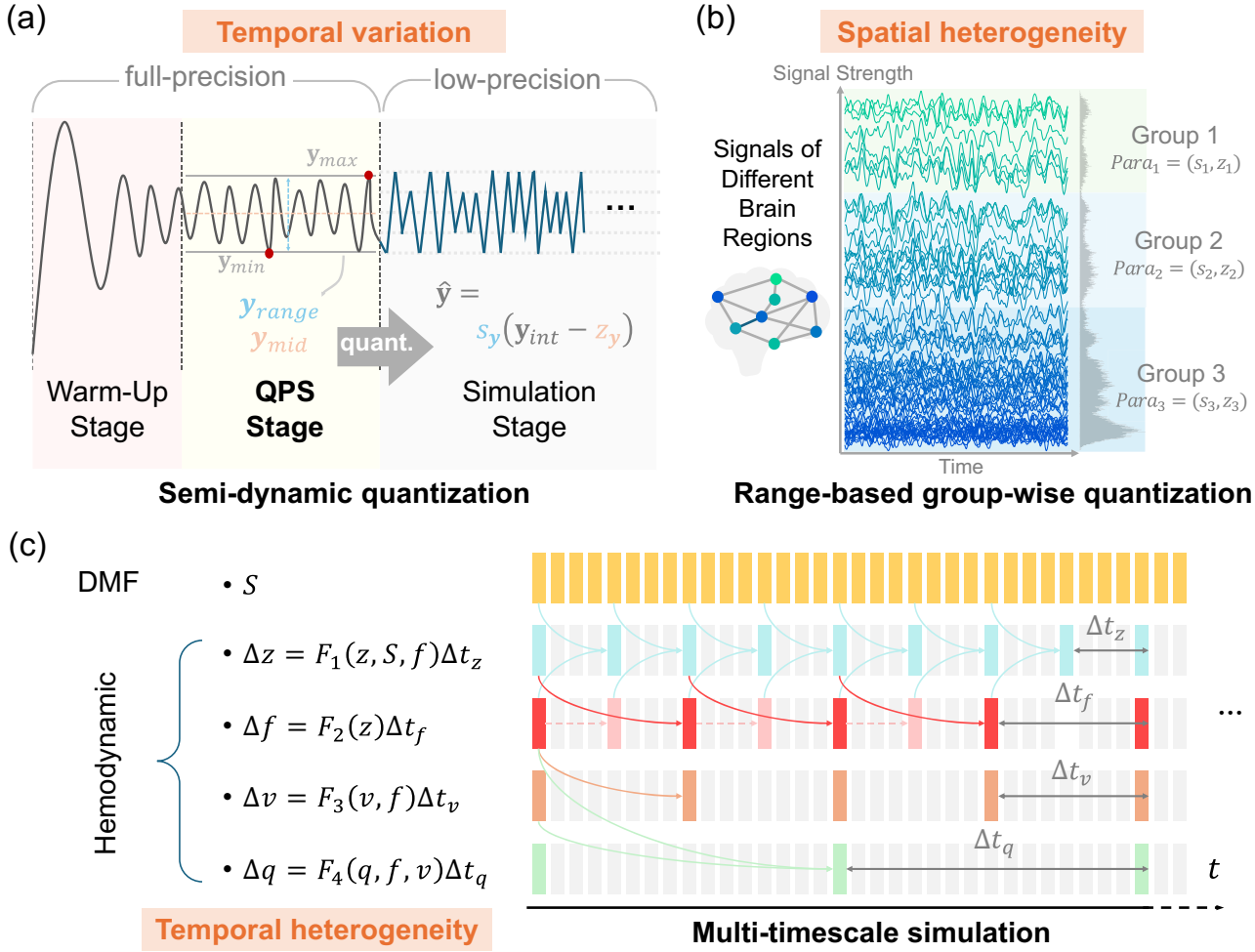


Fig. 2 | Dynamics-aware quantization for low-precision brain dynamics modeling and simulation. a The process of semi-dynamic quantization. The models are first simulated using floating-point data at the warm-up stage and the QPS stage. The quantization parameters are determined based on the distribution of the variables at the QPS stage (see "Methods" for details). The quantized model is then used at the simulation stage. **b** Some signals in the model do not follow the normal distribution. Signals from different brain regions show certain clustering characteristics. We group the brain regions based on their signal strength range and then determine the quantization parameters separately for each group. The brain icon was designed by meaicon from Flaticon. **c** Multi-timescale simulation of the hemodynamic model. Each row represents the iterative updating process of a state variable. Each state variable uses a different simulation time step size, represented by the spacing between two colored rectangles. Arrows illustrate data dependencies between variables. The dependency of each state variable on its previous time step has not been shown here for clarity. **d** Bifurcation diagram of the DMF model with respect to the global information strength parameter G. The region between the two curves represents the range of population firing rate H in the DMF model. **e** Temporal evolution of population firing rates across the whole brain. Each row in the heatmap represents the time series of average firing rates in a brain region. Blue represents lower firing rates, while redder colors indicate higher firing rates. The data shown are from high-precision simulations, and the heatmaps from low-precision models are too similar to distinguish visually. **f** Firing rate signals of three selected brain regions from the whole-brain network. Mid-range normalization is used to highlight signal fluctuations.

low-precision and high-precision models. Bifurcation is one of the most important characteristics of the brain model, which means that a small change in the system parameters could lead to a qualitative change in the system behaviors[12]. There have been studies that explore the mechanisms of seizures through bifurcation analysis of brain models[26,76–78]. Meanwhile, there may exist correlations between bifurcation point parameters and optimal fitting parameters of the model[79,80]. Thus, the preservation of bifurcation characteristics serves as a crucial criterion for validating low-precision models.

We can characterize the behaviors of the DMF model using the range of population firing rates H. Figure 2d is the bifurcation diagram of the DMF model with respect to the global scaling factor of information strength G. We fix other parameters, including w and I_0 of the model. When G is small, the firing rates of all brain regions are close to zero (state type I). When G exceeds the first bifurcation point, firing rates in some brain regions jump to a higher value, while others maintain low firing rates (state type II). When G exceeds the second bifurcation point, firing rates in all brain regions become significantly greater than zero (state type III). The firing rates of each brain region under different G values can be observed more clearly from Fig. 2e. A normal brain at rest typically operates in the type-II state, which is similar to the "$G = 2$" case. Moreover, under certain w and I_0 parameters, the DMF model may not exhibit the second bifurcation point and the type-III state.

We find that the low-precision model obtained through semi-dynamic quantization naturally maintains nearly identical bifurcation characteristics as the high-precision counterpart. This is because both the warm-up stage and the QPS stage use high-precision variables. If the model can converge to a certain type of state during these two stages, the quantization parameters of the low-precision model can be determined based on the range of variables in this state, thus having a high probability of maintaining the firing rate behaviors during the simulation stage. Under identical parametric conditions, both high-precision and low-precision models exhibit nearly identical relative firing rate patterns across brain regions. Furthermore, the model's bifurcation is reflected not only in the distinct temporal means of the regional firing rates but also in their temporal variance. In simpler terms, the amplitude of temporal fluctuations in each region's firing rate is also related to bifurcation characteristics. We select 3 brain regions and plot their firing rate signals under different G values in Fig. 2f. We align the median values of all signals to focus solely on their fluctuations. Despite the obvious discrete nature of signal changes, the amplitude patterns of firing rate fluctuations in these 3 brain regions in the low-precision model are identical to those in the high-precision model. Before the first bifurcation point, signals from all brain regions are relatively "steady" with small fluctuations; between the two bifurcation points, signals in some brain regions become more "volatile"; and after the second bifurcation point, firing rates from all brain regions show significant fluctuations. This result further validates that the low-precision model is capable of manifesting effective bifurcation.

Static and dynamic characteristics of the low-precision model

In addition to bifurcation, other dynamic and static characteristics of the model are closely related to the inversion algorithm. Dynamic characteristics describe how the brain changes and functions over time. Here, we look at the global coherence of the brain, which can be evaluated using the order parameter[81]. The order parameter at each moment represents the instantaneous synchronization between different brain regions at that time, where 1 indicates complete synchronization and 0 indicates the opposite. Studies[82,83] have shown that the resting brain may frequently switch between high and low synchronization states, and this can be captured by the order parameter. Besides, the mean and variance of the order parameter can represent synchrony and metastability, respectively, which are commonly used as dynamic indicators in model inversion algorithms. The order parameter simulated with both high-precision and low-precision models is illustrated in Supplementary Fig. S4a, with a duration of approximately 12 s. Despite having lower order parameter values and some differences in detail, the overall trend of the low-precision model aligns well with the high-precision model, demonstrating that it can effectively capture the dynamic synchronization patterns between brain regions. The lower order parameter might be due to the random perturbations introduced by low-precision quantization, which can reduce the overall coherence between brain regions.

Beyond dynamic characteristics, brain models must also capture static characteristics, including hierarchical organization and inter-regional connectivity. Resting-state FC is one of the most important static characteristics in macroscopic brain models. The similarity or correlation between the FC matrices obtained with empirical data and simulated data is usually used as the primary indicator in the inversion algorithm. Supplementary Fig. S4b displays the FC matrices from both empirical and simulated group-averaged models. Although both high-precision and low-precision models deviate from empirical data when comparing absolute values, they capture similar patterns observed in the empirical matrix in terms of relative connectivity strengths. The FC matrix generated by the low-precision model shows strong concordance with the high-precision result, exhibiting overall similarity but with some local discrepancies in detailed patterns, potentially also due to the noise arising from the discretization of temporal signals.

Qualitatively, the low-precision model simulation yields results very close to the high-precision model, reproducing the bifurcation phenomenon as well as the fundamental dynamic and static characteristics of the brain. Applying the low-precision model in inversion algorithms requires further investigation, including examining evaluation metrics and comparing low- and high-precision models across the parameter space. The results are detailed in the next section.

Consistency of goodness-of-fit indicators between low- and high-precision models

The low-precision model's effectiveness in model inversion fundamentally depends on the distribution of the goodness-of-fit indicators within the parameter space. Specifically, the inversion algorithm can find a sufficiently good solution with the low-precision model only when the extrema points and monotonicity features of its indicators are close to those of the corresponding high-precision model across the parameter space. To assess this, we first randomly sample 2000 parameter points in the (w, G, I_0) parameter space to evaluate the indicators (see Supplementary Table S2 for experimental details), and illustrate the overall distribution of a static indicator (Pearson correlation coefficient of FC) and a dynamic indicator (metastability) in Fig. 3a. Generally, the indicator distributions of low- and high-precision models are remarkably close. Most of the "good" points (the yellow, orange, or red points) cluster around the plane of $I_0 = 0.3$. The points with the highest dynamic and static indicators largely overlap, suggesting that the optimal solution is likely nearby. In the models of both precision levels, the number of points with a high dynamic indicator is noticeably fewer than the points with a high static indicator, which indicates that the dynamic indicator may be more sensitive to parameters.

On the other hand, the results also reveal some differences between models with different precisions, primarily manifested in the numerical ranges of indicators. Low-precision models generally exhibit smaller indicators overall, particularly in metastability, with more transparent points in the visualization. These phenomena are closely associated with the dynamic and static characteristics mentioned in the previous section. For the static indicators, we unfold the values in the simulated FC matrix into a vector and then calculate the Pearson correlation coefficient with the FC vector of empirical data. In the previous section, we observed that the FC matrices from the low-precision model show overall similarity but local differences compared

https://doi.org/10.1038/s41467-025-64470-3

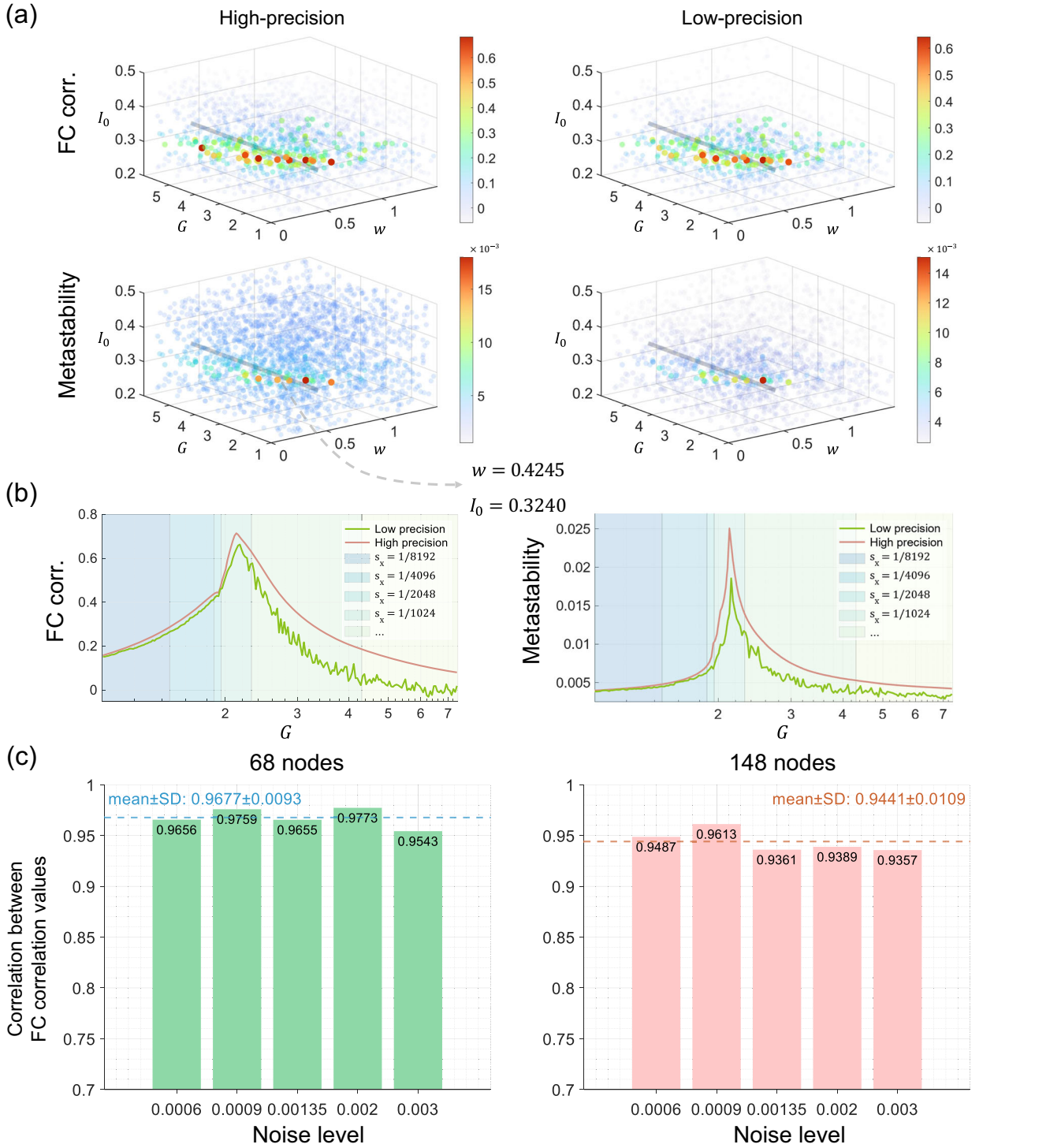


Fig. 3 | Distribution of goodness-of-fit indicators from models with different precision levels. a Distributions of indicators in the three-dimensional parameter space. Lower transparency indicates higher indicators, meaning better fit quality. **b** Curves of indicators with respect to G, while keeping w and I_0 fixed. The values of w and I_0 correspond to the gray lines in (**a**). Different background colors represent different values of the quantization parameter s_x. **c** Robustness analysis of the quantization method across different noise conditions and model scales. Each bar represents the Pearson correlation coefficient calculated from FC correlation values obtained from 1200 parameter sets sampled across the parameter space. All correlation results are statistically significant with $p < 10^{-4}$.

to those from the high-precision model. This explains why their static indicators are relatively close. The lower indicators in the low-precision model also suggest that quantization-induced FC discrepancies cause greater deviation from empirical data. For the dynamic indicators, we calculate the metastability values by computing the variance of the order parameter. Although the order parameter in the low-precision model exhibits similar local extrema to the high-precision model, its overall fluctuation range is smaller, potentially resulting in significantly lower variance. This aligns with the indicator differences we observe here.

Despite lower values, the distribution pattern of the goodness-of-fit indicators from the low-precision model appears almost identical to

that of the high-precision model, preliminarily validating the feasibility of parameter space exploration using the low-precision model. To further examine its extrema points and monotonicity features, we focus on the metric distribution along a line segment in the parameter space by observing how metrics vary with one parameter (G) while keeping the other two (w and I_0) fixed. The result curves are shown in Fig. 3b. Similar to what we observe in the three-dimensional distribution, the static and dynamic indicators obtained using models of different precision levels show the same trend as the parameter G changes, and the positions of the optimal points are almost consistent. It is worth noting that the inferred optimal G value from the low-precision model is slightly larger than that from the high-precision model. We provide detailed analyses and discussions of this phenomenon in the Supplementary Fig. S12. Moreover, the indicator values obtained from the low-precision model are also relatively smaller, particularly noticeable in the dynamic indicators. However, more details can be observed from the curves. The changes in the indicators obtained using the low-precision model are not as smooth as those from the high-precision model. When G is small, the indicators from the low-precision model increase smoothly as G increases, but when G exceeds the optimal point, both static and dynamic indicator curves exhibit significant local fluctuations in addition to the overall downward trend.

Due to the apparent regularity in these local fluctuations, we hypothesize that they might be related to the quantization process. To verify this hypothesis, we visualize in Fig. 3b the scaling coefficient s_x of the total input current variable x in the DMF quantization process. Different background colors represent different values of s_x. First, blocks of the same background color are continuous, indicating that under the fixed w and I_0, there is a clear piecewise correspondence between s_x and G, i.e., G values within a certain range correspond to the same s_x. Since s_x is determined by the range of x, it is very likely that the range of x changes monotonically with G. Second, s_x increases as G increases, showing that the range of x expands with increasing G. We notice that the background block where $s_x = 1/2048$ is notably narrow compared to other blocks, suggesting a significant change in the range of x at this point, possibly corresponding to a bifurcation point of the model. Finally, after marking the quantization parameters, we can clearly see that the severe local fluctuations in the indicator curves all occur in regions where $s_x > 1/1024$, indicating that quantization performance may deteriorate when the range of x is large. We believe that it stems from the nonlinear relationship between the population firing rate H and the total input current x in the DMF model (see Eq. 1). When x is very small, H is also close to 0; and when x increases slightly, H increases rapidly (for example, when x increases from 0.3 to 0.4, H increases more than tenfold). Therefore, as the range of x expands with increasing G, the range of H expands at an even faster rate, which significantly reduces the quantization precision of H, leading to unstable indicators from the low-precision model. Meanwhile, we also quantitatively assess the impact of quantization parameters on indicator distributions in Supplementary Fig. S9.

To further quantitatively evaluate the consistency of goodness-of-fit indicators between low-precision and high-precision models, we calculate the Pearson correlation for the indicator values (FC correlation) obtained from the models across random sample points in the parameter space. To test the robustness of our results, we conduct additional experiments under different noise intensities and across different model scales. Specifically, we vary the additive noise amplitude σ in the stochastic differential equations from 0.0006 to 0.003 and run simulations with both 68-node and 148-node connectomes. The noise amplitude range encompasses the majority of scenarios encountered in the parameter search and optimization procedures. As shown in Fig. 3c, the low-precision simulations preserve significantly positive correlations with high-precision counterparts under all tested scenarios, indicating robust performance under varying noise levels and model complexities. Meanwhile, we also observe that the positive correlation coefficient shows some degradation when the noise amplitude is high (e.g., $\sigma = 0.003$). On one hand, larger noise may render the dynamic system more unstable, and if significant range variations occur, the effectiveness of low-precision simulation would deteriorate. On the other hand, the complex dynamics may amplify the noise effects, especially in the DMF model, where the functional relationship between the total input current and the population firing rate is highly sensitive to numerical variations. This accuracy degradation could potentially be mitigated by appropriately increasing the quantization precision.

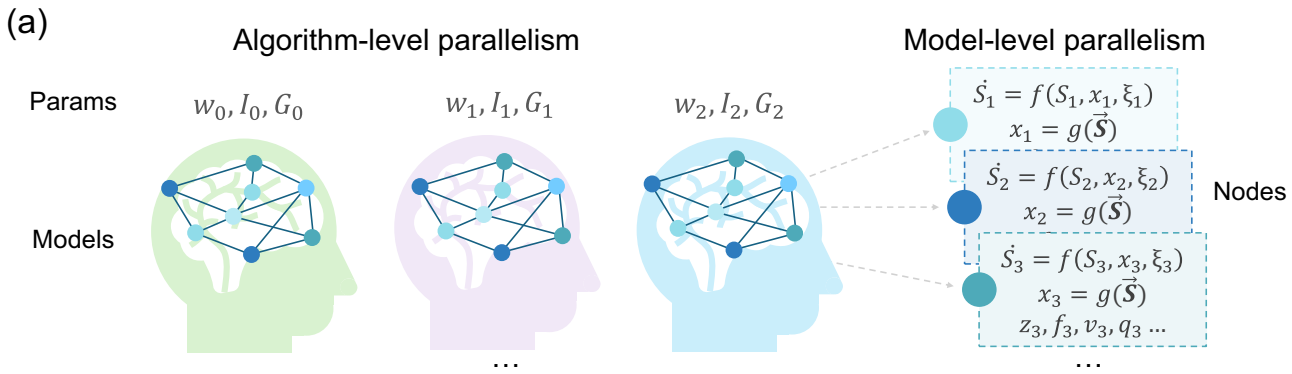
To directly validate the applicability of low-precision models in practical scenarios, we use the particle swarm optimization (PSO) algorithm to search for optimal parameters in the parameter space based on the low-precision model. Unlike the experiments above, here we also include the noise amplitude as an optimizable parameter. We conduct 30 repeated simulations using both high-precision and low-precision models with the searched parameters. The correlation between low-precision simulated and empirical FC results is, on average, 0.7018 ($p < 10^{-4}$), with an SD of 0.0103. The correlation between high-precision simulated and empirical FC results also achieves 0.6908 ($p < 10^{-4}$), with an SD of 0.0130. Meanwhile, the correlation between the simulated FC results from the two precisions reaches 0.9675 on average ($p < 10^{-4}$). As a baseline, the correlation between the empirical SC and FC is 0.4545. These results demonstrate that using the low-precision model for model inversion is effective.

In addition to the overall goodness-of-fit indicators of the model, we also conduct a quantitative analysis of the topological properties. We characterize the functional properties of each node using local topological properties derived from group-level functional networks. Our results in Supplementary Fig. S5 demonstrate that the low-precision model is capable of capturing key topological properties of the empirical networks and exhibits significant positive correlations with empirical data. Low-precision models exhibit a similar correlation distribution to high-precision models, despite differences in metrics such as participation coefficient and betweenness centrality.

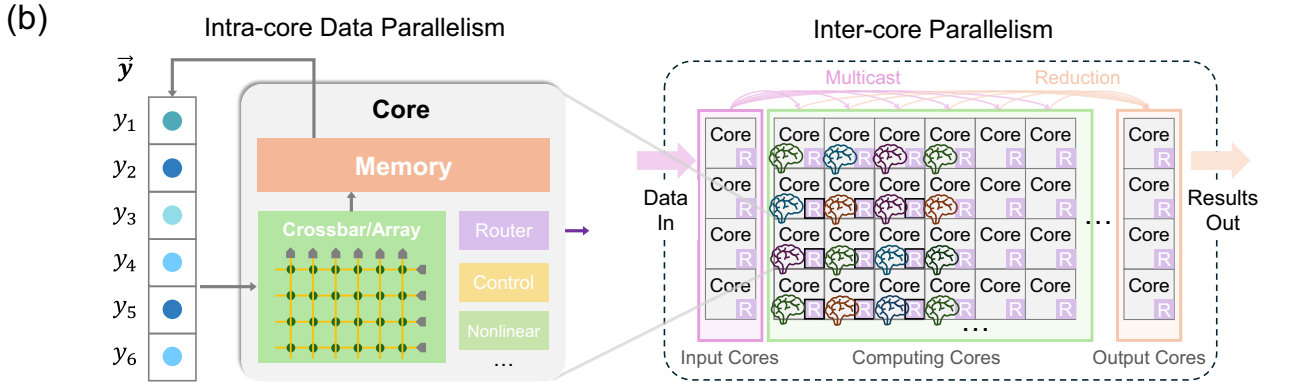
Furthermore, we conduct temporal domain analyses to examine the impact of the quantization approach on temporal domain characteristics. We first compare the power spectral density of time series from high-precision simulation, low-precision simulation, and empirical rs-fMRI data. As shown in Supplementary Fig. S11a, both high-precision and low-precision simulations exhibit power spectral profiles that closely match the empirical data in terms of peak locations and overall curve shapes, although specific amplitude values show slight differences. The low-precision simulation results demonstrate high similarity to high-precision counterparts. Notably, the low-precision results show reduced peak power intensity while exhibiting elevation in other frequency components. This phenomenon is theoretically expected, as the quantization process introduces noise-like artifacts that elevate the overall power spectrum while reducing the signal-to-noise ratio. We also calculate the sample entropy for each brain region's simulated time series, and compute correlations with the empirical regional sample entropy. As demonstrated in Supplementary Fig. S11b, both precision levels show significant correlations with empirical data. Furthermore, the correlation between high-precision and low-precision sample entropy values reaches $r = 0.9482$ ($p < 10^{-4}$), indicating strong consistency. The detailed description of the indicators and metrics we use is presented in "Methods".

Mapping hierarchical parallelism of the model inversion onto parallel computing architectures

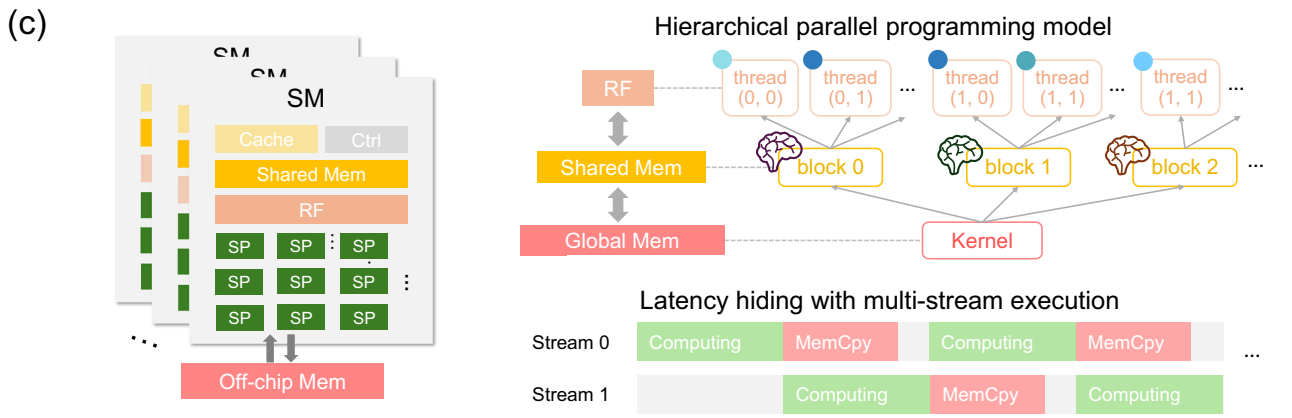
The parallelization opportunities of the inversion process originate in the whole-brain model and the inversion algorithm, as shown in Fig. 4a. At the model level, each node (i.e., a neuronal population or a brain region) has its own set of differential equations, which can be



Hierarchical parallelization opportunities of model inversion



Hierarchical parallelism of the brain-inspired computing chip



Hierarchical parallelism of general-purpose GPU

Fig. 4 | Accelerating multi-level parallelizable model inversion via hierarchical architectural parallelism. a The two-level parallelism in the algorithm and models. The brains in different colors represent the whole-brain dynamical models with different parameters, and the blue dots of varying shades represent different nodes in a model. **b** Illustration of the inversion process on brain-inspired computing chips. Parallel models (marked as brains) are allocated to different cores and accelerated by leveraging inter-core parallelism. The cores are grouped into input cores, computing cores, and output cores. They execute in a pipelined manner for hiding latency. The variables within different nodes make up vectors and are computed in the crossbar or MAC array in each core, leveraging the intra-core data parallelism. **c** Illustration of the inversion process on GPUs. The left section depicts the typical GPU architecture. The upper right section presents the hierarchical programming model, including the memory hierarchy and the thread model. The blue dots and brain diagrams marked on the "thread" and "block" represent the mapping of parallel models and nodes to the thread model. The lower right section illustrates the latency hiding strategy, highlighting how pipelined computation and memory access are achieved through multi-stream concurrent execution. The brain icons were designed by meaicon from Flaticon.

simulated concurrently. At the algorithm level, multiple models with distinct parameter settings can also be evaluated in parallel if using a population-based metaheuristic algorithm (see "Methods"). The core of hardware deployment is mapping these two-level parallel workloads onto the computing and memory resources of the computing platforms. Limited parallel computing resources on general-purpose processors hinder the parallelization of the models and nodes, particularly when dealing with algorithms containing numerous nodes and models, making the inversion process time-consuming. In contrast, brain-inspired computing chips and GPUs, as representative advanced computing architectures, both possess highly parallel computing capabilities, thus having the potential to accelerate the inversion

process by leveraging their hierarchical parallelism. However, they differ fundamentally in architecture organization: brain-inspired computing chips adopt a non-von Neumann many-core architecture, parallelizing both computing and memory resources, while GPUs have massively parallel computing elements with shared memory, which is closer to the traditional von Neumann architecture. To fully leverage the computational capabilities, we design different mapping strategies for different architectures.

Brain-inspired computing chips typically adopt a many-core near-memory or in-memory computing architecture, with each core containing a large number of arithmetic circuits organized in a multiplier-and-accumulator (MAC) array with local on-chip memory or an in-memory computing crossbar. So the hierarchical parallelism of brain-inspired architectures comes from two sources: many-core parallelism and data parallelism within each core. As shown in Fig. 4b, we leverage inter-core parallelism to accelerate the evaluation of multiple models in the algorithm and employ intra-core data parallelism to speed up solving differential equations of multiple nodes within each model. In fact, the simulation of a model can also take advantage of multi-core parallelism on the brain-inspired computing chips with spatiotemporal elasticity, such as TianjicX[42]. We will discuss the mapping flexibility in the Discussion section. As the distributed local memory lacks hierarchy, we put all the data of a model, including the parameters, the SC matrix, and the variables of nodes, in the local memory of the corresponding core(s). In this way, the overhead of accessing data during the evaluation process of each model can be minimized. Furthermore, both model inputs and simulation outputs need to be transferred from or to the host through inter-chip interfaces, and then routed to certain computing cores through the on-chip network composed of routers in cores. The time overhead of off-chip communication is non-negligible in certain cases. Therefore, we have designated certain cores in the chip as data forwarding cores, i.e., input cores and output cores in Fig. 4b, which are dedicated to receiving data from or sending data to the host. These cores can operate in a pipelined manner with the computing cores to hide latency effectively. In addition, to accelerate the inversion algorithm on ASICs (application-specific integrated circuits) such as brain-inspired computing chips, we also need to implement various operations in the model using the intrinsic functions on the chip. The details are provided in "Methods".

By contrast, GPUs feature a hierarchical design of multiprocessors containing multiple stream processors (SPs), with each level having dedicated memory. All SPs can compute simultaneously, but memory resources are not fully parallel. Larger memory is shared by more computing resources, resulting in lower average access bandwidth. Specifically, the fastest register files (RFs) are accessible only to their local SP, shared memory can be accessed by all SPs within a multiprocessor, and the slower and larger-capacity global memory is accessible to all multiprocessors. In correspondence with the hardware structure, GPUs typically adopt a hierarchical thread model as their programming model at the software level. Individual threads run on SPs and combine into thread blocks. Multiprocessors execute these blocks in the granularity of warps. The key to GPU deployment lies in matching the two-level parallelism of the inversion algorithm with the architectural hierarchy. Intuitively, we map the model-level nodes onto the threads on SPs, and map the multiple models at the algorithm level onto the thread blocks on multiprocessors, as illustrated in Fig. 4c. We have carefully orchestrated the locations of different data during the computation process to reduce memory access overhead as much as possible (see "Methods").

After the model simulation on the GPU, the resulting data (e.g., the state variables in the DMF model at each time step) also need to be moved to the host for post-processing, such as metrics calculation and comparison. The data transfer overhead between the CPU and the GPU cannot be ignored compared to the computation and memory access within the GPU. We propose a time-segmented pipeline method for hiding latency. Implementing this type of task-level parallelism on the GPU is not as straightforward as the many-core parallelism in brain-inspired computing chips. Modern GPUs with multi-stream support allow concurrent operations across streams if their resource dependencies do not conflict. We divide tens of thousands of time steps in the simulation into N segments and alternately assign each segment (including computation and data transfer) to two streams. For example, assume that the ith segment is executed in stream 0, which means that stream 0 will first perform simulation computation and then copy the output data from the global memory to the host. We will initiate the execution of the $(i+1)$th segment in stream 1 at the start of the data transfer in stream 0. The process is depicted in the lower right area of Fig. 4c. Thus, data communication is overlapped with computation, leading to higher throughput and hardware resource utilization.

We deploy the inversion process using the aforementioned mapping methods onto a brain-inspired computing chip, TianjicX[42], and a commercial GPU, NVIDIA RTX 4060 Ti, with comparable computing power. We measure and estimate the execution time of the inversion process on three different architectures. Detailed measurement and estimation methods are provided in "Methods". We use the same identification algorithm across different architectures, with fixed iteration counts and the number of models evaluated per iteration, only varying the number of nodes in models. The results are presented in Table 1. Both parallel computing architectures demonstrate significant advantages over the CPU in terms of overall inversion algorithm runtime and simulation time alone, and the acceleration effect becomes increasingly pronounced as the model size grows. For example, the parallel simulation speed of the commonly used 148-node model on the brain-inspired computing chip is 90.1 times faster than the CPU, while this difference increases to 424.4 times with the synthesized 512-node model. Between the two parallel computing architectures, the brain-inspired computing chip consumes less time in model simulation, and this performance gap also widens as the model size increases. However, the speed difference in executing the overall model inversion is not so pronounced. This is because the semi-dynamic quantization on the brain-inspired computing chip relies on the host (CPU) to complete the warm-up stage and the QPS stage, which can only leverage the limited parallel resources of multi-core

Table 1 | Computational performance of brain modeling across different platforms

#Node	Platform	$T_{simulation}$*	$T_{overall\_inversion}$*
68	CPU	1978.85	2074.64
	GPU (ours)	31.10 (63.6×)	52.46 (39.5×)
	Brain-inspired (ours)	**26.32** (**75.2**×)	**42.33** (**49.0**×)
148	CPU	4246.34	4365.02
	GPU (ours)	109.35 (38.8×)	154.51 (28.3×)
	Brain-inspired (ours)	**47.14** (**90.1**×)	**77.11** (**56.6**×)
512[a]	CPU	92426.65	93534.92
	GPU (ours)	2007.15 (46.0×)	2184.16 (42.8×)
	Brain-inspired (ours)	**217.80** (**424.4**×)	**796.43** (**117.4**×)

The GPU and the brain-inspired computing chip can handle most simulation tasks, but require a host platform for completing the entire inversion process. $T_{overall\_inversion}$ is the total time consumed during the entire model inversion process, while $T_{simulation}$ represents the model simulation time. For the GPU and the brain-inspired computing chip, the $T_{simulation}$ results include both computation time and intra-device data transfer time. The remaining time in the entire process is used for data transfer between the host and device, metric calculations, search algorithm iterations, memory management, and other workloads.

*All time units in the table are in seconds. Numbers in the parentheses represent speedup factors compared to CPU execution. The bold numbers represent the minimum computation time and the maximum speedup ratio across different platforms for a given number of nodes.

[a]Measured and estimated using synthetic data.

CPUs for acceleration. In contrast, the GPU implementation does not require low-precision computation, and the GPU itself can accelerate the warm-up process.

Even with this limiting factor, the brain-inspired computing chip still maintains the fastest among all three architectures regardless of the model size. This further highlights the inherent advantages of the brain-inspired computing architecture, including faster low-precision computing units and efficient distributed near-memory computing circuits with reduced redundancy. Additionally, we also explore using a GPU to perform the warm-up phase while maintaining the main simulation on the brain-inspired computing chip. When the model scale is small, this approach does not provide particularly significant improvements, as the time spent on warm-up accounts for a relatively small proportion of the total time consumption. However, when the model scale is large (e.g., 512 nodes), GPU-based warm-up acceleration can significantly improve overall performance. Specifically, for the 512-node model, the hybrid pipeline achieves a 2.1× speedup over the brain-inspired computing chip pipeline (see Supplementary Table S4). This hybrid pipeline can leverage the best of both worlds, and it is worth exploring in future work.

Scalability of model inversion on different architectures

Current coarse-grained brain dynamics models are relatively small in scale (e.g., 68 nodes, 148 nodes) due to limitations in brain imaging resolution and other factors. With technological advancement and a deeper understanding of the brain, future dynamics models may incorporate more nodes for more detailed brain modeling, leading to increased computational costs. Therefore, we further explore how computational performance varies with model size. We evaluate inversion runtime on the GPU and the brain-inspired computing chip using synthetic data from 64-node to 512-node models, with 128 models per iteration, measuring time consumption for each processing step. Based on the platform dependencies of these steps, we divided the total runtime into three parts: host-side time, device-side time, and data transfer time. Figure 5a shows the results. Across all model sizes evaluated, the total runtime of model inversion based on the brain-inspired computing chip is shorter than the GPU-based implementation. This performance gap becomes more pronounced with larger model sizes. The runtime breakdown shows that the data transfer time is similar between the two architectures, and the host-side time when using brain-inspired computing chips is comparable to or greater than that when using the GPU. Therefore, the overall runtime advantage primarily stems from the significantly shorter device-side time on the brain-inspired computing chip compared to the GPU. Regardless of whether the GPU or the brain-inspired computing chip is used for computation, the host (general-purpose platform) is necessary to perform operations such as data I/O, optimization algorithms, pre-processing, post-processing, and so on. Therefore, the host-side time consumption is non-negligible during the inversion process. In the brain-inspired computing chip implementation, host operations actually consume the majority of the runtime when model size is large, primarily due to the host-dependent warmup and quantization parameter determination in the semi-dynamic quantization method. We observe a sudden jump in the host-side time consumption when scaling from 320 to 384 nodes, likely caused by factors such as decreased cache hit rates due to insufficient cache size, memory paging, or BLAS library switching. We can predict that the host time would become the dominant bottleneck in the brain-inspired computing chip pipeline as the model size increases. This highlights a significant opportunity for future optimization. Performance might be further improved through more efficient quantization algorithms, or by offloading portions of the warm-up process to a GPU (like what we do in Supplementary Table S4) or other specific hardware accelerators.

We present a detailed runtime breakdown in Fig. 5b for the three representative model sizes from Table 1. In the GPU-based implementation, as the model size increases, the proportion of device-side time (i.e., GPU execution time) grows accordingly. For brain-inspired computing chip implementation, the proportion of device-side time remains relatively stable at smaller scales (from 68 to 148 nodes), but decreases significantly for the 512-node model due to the substantial growth in host-side time. In GPU-based inversion, the data transfer latency is almost completely hidden by the device-side time. However, the situation is more complex on the brain-inspired computing chip. The device-side time on the brain-inspired chip actually consists of two components: computation time and on-chip data communication time. We configure data forwarding cores and computing cores on the brain-inspired computing chip for pipelined execution, but these two types of cores need to synchronize during on-chip communication. Consequently, on-chip communication cannot occur simultaneously with computation or inter-chip data transfer, and only the computation time and the inter-chip data transfer time can overlap. For smaller model sizes, the computation time is shorter than the inter-chip data transfer time, thus only partially hiding the data transfer latency. Only when the model size becomes larger, with significantly increased computation time, can the inter-chip data transfer time be almost completely covered.

Beyond the model size, the inversion process can also be expanded in terms of the number of parallel models. For the same optimization algorithm, evaluating more models (parameter sets) per iteration would increase the likelihood of finding better solutions. Therefore, the scaling of the model count is crucial for both applications and research. The scalability of inversion time across both dimensions, including the model size and the model count, is illustrated in Fig. 5c. Here, we primarily focus on data transfer time and device-side time, both of which are architecture-sensitive. We exclude the steps such as warm-up and metric calculations to minimize the influence of the host. The GPU architecture consistently consumes more time than the brain-inspired computing architecture, and this gap widens with both increasing model size and model count. In some large-scale cases, the GPU execution time can reach up to 10 times longer than that of the brain-inspired computing chip. Notably, the GPU execution time shows clear discontinuities with an increasing number of parallel models, likely stemming from the scheduling constraints where each SM can accommodate a maximum of 32 thread blocks. In contrast, the brain-inspired chip shows a small variation in execution time with an increasing number of parallel models. This is because we allocate one core per model, and the brain-inspired chip we used has sufficient cores to avoid time-division multiplexing. This enables fully parallel simulation computation of different models, with only the on-chip communication time increasing as the number of models grows.

Discussion

We propose an accelerated pipeline for coarse-grained modeling of brain dynamics based on advanced computing architectures. We primarily address two key challenges. The first is implementing macroscopic brain dynamics on specialized computing architectures like brain-inspired computing chips, which offer high efficiency but low precision. This challenge needs computational algorithmic innovation. Through theoretical and experimental analyses on the DMF model, we discover that macroscopic brain dynamics models exhibit complex spatiotemporal heterogeneity during resting-state simulation. State variables typically undergo transient variations initially before converging to relatively stable states, after which they demonstrate differences in numerical distributions across brain regions and require distinct time step sizes among different variables.

These characteristics differ markedly from normal neural networks in AI. Building upon existing methods on neural network quantization, we propose dynamics-aware approaches, including semi-dynamic quantization, range-based group-wise quantization, and

https://doi.org/10.1038/s41467-025-64470-3

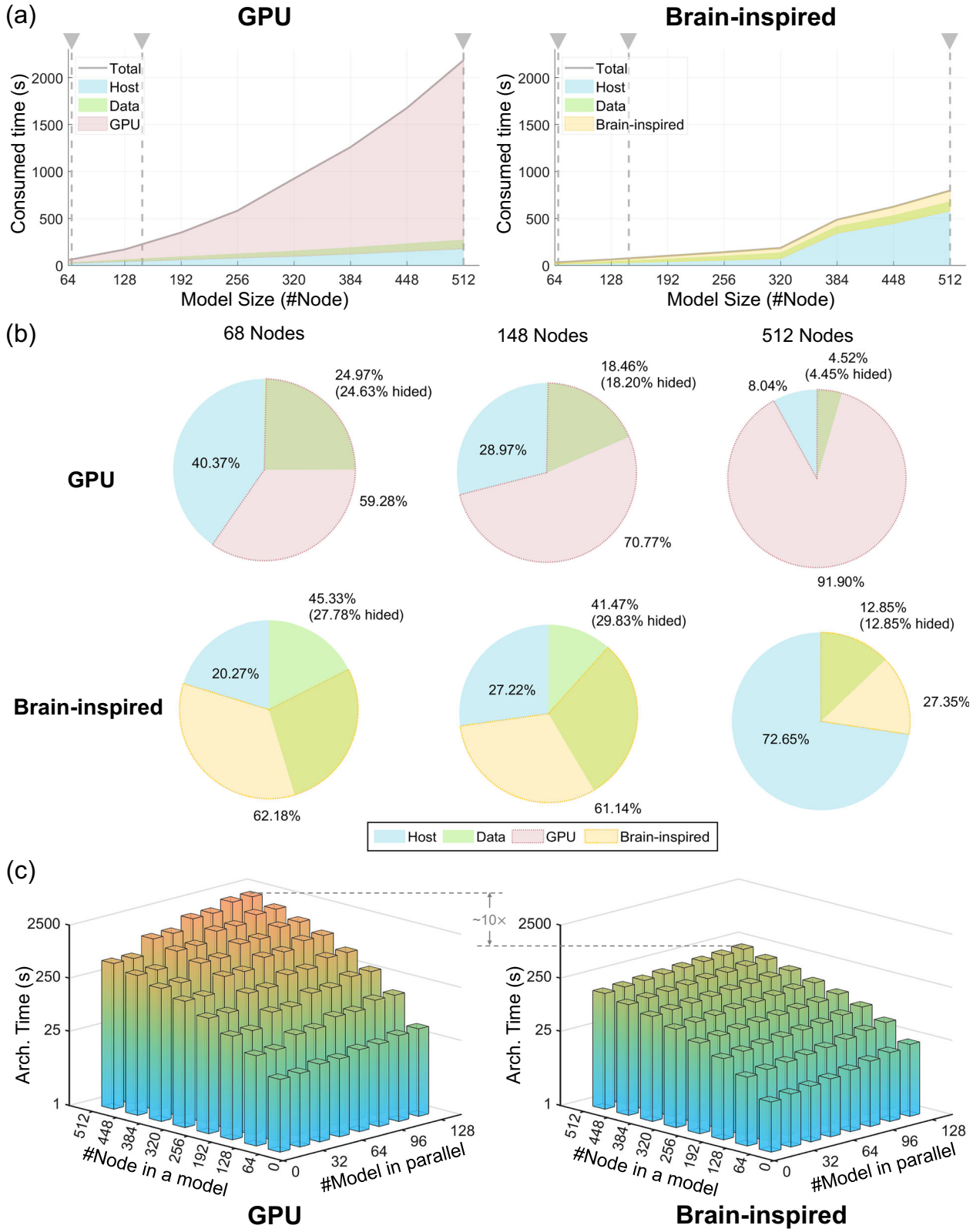


Fig. 5 | Performance analysis of model inversion across different scales. **a** Runtime breakdown comparison between the GPU and the brain-inspired computing chip for the inversion process with 128 parallel models at different model scales (number of brain region nodes in each model). Note that the green area representing data transfer time is partially overlapped by the transparent red/yellow area representing device-side computation time. **b** Detailed runtime breakdown at three representative model scales corresponding to Table 1 and the gray dashed lines in (**a**). The data transfer segment is also partially overlapped by the computation segment in each pie chart. **c** Consumed time on parallel computing architectures during the inversion process under different algorithm scales and model scales, excluding pre-processing and post-processing time.

multi-timescale simulation. These approaches preserve essential dynamical properties in the quantized models. We find that the low-precision models maintain bifurcation characteristics and retain similar static and dynamic properties to their high-precision counterparts. Experimental results show that fitting indicator distributions of low-precision models in the parameter space closely match those of high-precision models, and model inversion based on low-precision models can achieve reliable parameter estimation. We also validate the quantization approaches on the Hopf model, demonstrating a certain degree of generalizability across different dynamical models.

Our proposed methods not only enable the deployment of macroscopic brain dynamics models on low-precision architectures but also show promise for extending to lightweight implementations of dynamical models in bio-physical or other scientific domains, allowing broader scientific computing tasks to benefit from advanced computing architectures. However, the proposed method also suffers inherent limitations: When a dynamical model exhibits frequent or unpredictable large-magnitude fluctuations in its numerical range, the semi-dynamic quantization would likely become unsuitable. There exists a fundamental trade-off between quantization precision and generality: higher quantization precision enables support for a broader variety of dynamical models with better simulation fidelity, but may result in reduced computational efficiency. Future work should explore adaptive quantization schemes that can dynamically balance this trade-off based on the specific characteristics of the target dynamic system and the available computational resources. Additionally, the computational operators supported by existing brain-inspired computing chips are limited and may not be capable of implementing arbitrary dynamic systems. These constraints highlight the need for future research in developing more flexible quantization strategies and expanding the computational capabilities of brain-inspired computing architectures to accommodate a broader range of scientific computing applications. Furthermore, the proposed low-precision methods are currently tested only with the Euler integration scheme. However, the field might require other integration methods for specific applications in the future, such as higher-order Runge-Kutta methods for enhanced accuracy or adaptive time-stepping algorithms for complex dynamical behaviors. The application of low-precision data types and computation to different integration schemes might be a valuable direction for future research.

In addition, we reveal the underlying mechanisms that determine how quantization affects the consistency of models with different precisions. First, under certain model parameters, variables associated with nonlinear transformations may exhibit significantly wider value ranges, affecting the selection of quantization parameters and increasing the rounding errors in low-precision models, thus degrading quantization performance. While this effect does not significantly impact the validity of the low-precision models in this work, it might limit the low-precision implementation of other dynamic systems. An interesting topic in future work is to further explore this effect and seek possible solutions. Second, the quantization of variables, especially the SC, may introduce a slight bias in the inferred optimal parameters. This bias might be addressed by either improving the quantization precision of SC as much as hardware permits or by compensating for the inferred parameters based on changes in SC before and after quantization.

The second challenge is to fully utilize the highly parallel computing and memory resources of brain-inspired computing chips and GPUs to accelerate the costly model inversion process, which requires the development of new systems that leverage heterogeneous computing platforms along with considerable engineering optimization efforts. Existing whole-brain modeling tools, such as TVB[84], neurolib[85], and BrainPy[86], typically rely on high-performance computing platforms and run on general-purpose CPUs or GPUs. They leverage thread- or process-level parallelism to accelerate the simulation of brain regions, connectivities, or model instances. However, they do not finely control the resource scheduling to fully utilize the hardware capabilities. And the parallel efficiency is constrained by their underlying hardware architectures: CPUs possess limited parallel computing resources, while GPUs, despite having abundant computational units, may suffer from memory bandwidth bottlenecks. Both platforms employ considerably complex circuit designs to ensure generality, which introduces heavy overhead. By contrast, our work focuses on detailed mapping between algorithmic components and architectural resources, and explores the application of the non-von Neumann brain-inspired computing architectures, which feature tightly coupled computation and memory with specialized low-precision circuit designs. Specifically, we adopt a population-based metaheuristic inversion algorithm that is more amenable to parallelization and analyze multiple levels of parallelism in model inversion. We develop tailored mapping strategies that distribute multi-level parallel tasks across the computing and memory resources of both architectures. These strategies enable fine-grained and low-overhead scheduling, particularly for the brain-inspired computing architecture, providing more efficient parallelism. According to our evaluation, model inversion running on these advanced computing architectures can achieve significant speedup compared to general-purpose CPU platforms. Further analyses of runtime breakdown and scalability reveal that, despite the brain-inspired computing chip's greater dependence on the general-purpose host due to specialized functionality, its superior performance in parallel model simulation makes it a better choice over GPUs, especially for larger inversion tasks.

This work extends brain-inspired computing chips, originally designed for fine-grained neuronal networks, to simulate macroscopic brain models, broadening their application scope while further leveraging their architectural advantages. Meanwhile, for the commonly used brain dynamics models with fewer nodes, GPUs can also achieve considerable acceleration. Notice that the proposed pipeline cannot be implemented solely on brain-inspired computing chips or GPUs, but requires a heterogeneous system composed of a host. This aligns with the current trend in heterogeneous platform development. Future work could explore deploying brain dynamics modeling on systems that integrate brain-inspired computing architectures with CPUs on a single chip for potentially enhanced computational performance.

In addition, deploying brain dynamics also provides insights for future design of brain-inspired computing architectures. First, existing brain-inspired computing chips typically support only 8-bit or lower precision data types, with limited mixed-precision or dynamic quantization capabilities. This constrains the selection of quantization methods and degrades the effectiveness of low-precision models, while simultaneously increasing the workloads on the host and compromising the overall performance. Our analysis indicates that the host-side warm-up process becomes a bottleneck for the entire pipeline as the model size increases. While using a GPU for warm-up can effectively alleviate this bottleneck, it inevitably increases system complexity. We believe the inversion time could be more efficiently reduced by either adding high-precision computation paths within cores or incorporating some high-precision computing cores on the chip. Second, the brain-inspired computing chips we use require host control for time-step iterations in dynamics simulation, necessitating frequent accelerator-host interactions. Enabling a flexible on-chip control flow might improve simulation efficiency, especially in complex scenarios like multi-timescale simulation. Third, aside from the warm-up phase, other essential pipeline components, including search algorithms and data I/O operations, are all handled by the host. In most complex applications and deployment scenarios, these advanced computing architectures cannot independently satisfy all functional requirements and cannot operate completely standalone without CPUs or FPGAs, unless we design specialized circuits for each specific

scenario. We believe this is a universal limitation of current computing systems. One of the most promising approaches to improve the system integration under this constraint is System-on-Chip or System-in-Package designs. These designs not only reduce the physical size but also potentially minimize the data transfer between the host and the accelerators, thus significantly enhancing the deployability of the proposed pipeline. Last, we deploy model inversion on brain-inspired computing chips by simply mapping each model to a single core in experiments. Actually, the simulation of a model does not necessarily rely solely on a single core; it can also take advantage of the multi-core parallelism of the brain-inspired computing chip. When the resources of a single core cannot meet the requirements of a model, we can split the simulation of a model across multiple cores. We can divide a large model into several smaller parts, for example, splitting a model with N nodes equally between two cores, with each core responsible for executing $\frac{N}{2}$ nodes. Alternatively, we can divide T time steps of a model simulation across multiple cores, each responsible for a segment of the temporal evolution. This forms a temporal pipeline where the simulation is partitioned into sequential segments (e.g., time steps 1–100, 101–200, 201–300, etc.), with each core handling one segment. Within a single simulation run, adjacent segments cannot execute in parallel due to the causal dependency between consecutive time steps. For example, core 1 must complete processing time steps 1–100 before core 2 can begin processing time steps 101–200 using the results from core 1. However, this pipelining approach enables different simulation instances to overlap their execution: while core 1 processes time steps 1–100 of simulation instance A, core 2 can simultaneously process time steps 101–200 of a previously started simulation instance B, and core 3 can process time steps 201–300 of an even earlier simulation instance C. Despite introducing additional communication overhead for data transfer between cores, it provides considerable flexibility for resource mapping. This may help improve resource utilization and overall throughput. The complex spatiotemporal mapping problem is beyond the scope of this work and thus left for future work.

In short, macroscopic brain dynamics modeling on brain-inspired computing architectures demonstrates the potential of applying existing intelligence-oriented computing architectures to scientific computing in neuroscience and medicine, while bringing macroscopic brain models into the realm of brain-inspired computing. This work fosters deeper interdisciplinary collaboration between brain science and computational science, driving mutual advancement. Looking ahead, it opens up promising directions for future exploration, such as lightweight brain dynamics, heterogeneous computing architectures, and efficient medical systems.

Methods

Dynamic mean-field model

In this work, we use the DMF model proposed in prior work[21] to simulate the brain dynamics. It describes the dynamics of the network containing excitatory and inhibitory populations of spiking neurons interconnected via NMDA synapses. The global brain dynamics of the interconnected local networks can be expressed by the following nonlinear differential equations:

$$\begin{cases} \dot{S}_i = -\frac{S_i}{\tau_S} + r(1-S_i)H(x_i) + \sigma\xi_i(t) \\ x_i = wJS_i + GJ\Sigma_j C_{ij}S_j + I_0 \\ H(x_i) = \frac{ax_i - b}{1-\exp(-d(ax_i-b))} \end{cases} \tag{1}$$

where S_i denotes the average synaptic gating variable at the ith local cortical area, which is the state variable with time dependency; x_i and $H(x_i)$ are intermediate variables denoting the total input current and the population firing rate at the ith area, respectively. The range of i varies along with the number of areas in the dataset. The local excitatory recurrence w, the external input I_0, and the global scaling factor of information strength G are the parameters we need to tune through model inversion. C_{ij} denotes the structural connectivity (SC) between the ith and the jth areas, which can be derived from dMRI data. $\xi_i(t)$ corresponds to the Gaussian noise in the dynamic system, and σ is the amplitude, which can also be tuned. Here, we fix σ to 0.001 for simplicity since it typically falls around the value after identification. The other variables in the equations are all constant parameters. The value of synaptic coupling J is set to be 0.2609 (nA). The parameter values for the input-output functions $H(x_i)$ are $a = 270$(n/C), $b = 108$(Hz), and $d = 0.154$(s). The kinetic parameters for synaptic activity are $r = 0.641$ and $\tau_S = 0.1$(s).

To identify the DMF model with empirical data, we need to simulate the blood-oxygen-level-dependent (BOLD) fMRI signals further with the Balloon-Windkessel hemodynamic model[87]. The dynamic process linking synaptic activities to BOLD signals is governed by

$$\begin{cases} \dot{z}_i = S_i - \kappa z_i - \gamma(f_i - 1) \\ \dot{f}_i = z_i \\ \tau\dot{v}_i = f_i - v_i^{1/\alpha} \\ \tau\dot{q}_i = \frac{f_i}{\rho}[1-(1-\rho)^{1/f_i}] - q_i v_i^{1/\alpha - 1} \end{cases} \tag{2}$$

where S_i is the same variable as in the DMF model. The vasodilatory signal z_i, the inflow f_i, the blood volume v_i, and the deoxyhemoglobin content q_i are four state variables. The BOLD signal follows

$$BOLD_i = V_0[k_1(1-q_i) + k_2(1-\frac{q_i}{v_i}) + k_3(1-v_i)]. \tag{3}$$

The principles of the model and parameter values can be found in prior work[87], and are not repeated here. Combining DMF and the hemodynamic model, we can simulate the brain activity as well as the BOLD signals, which can also be measured using fMRI empirically.

Semi-dynamic quantization for brain dynamics

The commonly used data types on intelligent computing platforms can be divided into two categories: floating-point types and integer types. GPUs usually possess large capacities for floating-point computing, supporting full-precision formats like Float32 (FP32) and half-precision formats like Float16 (FP16) or BFloat16 (BF16). Implementing dynamical models with these data types is not difficult because their range and precision are sufficient to represent the model variables in most cases. In contrast, the most popular low-precision integer formats in brain-inspired computing chips are signed 8-bit integers (INT8) and unsigned 8-bit integers (UINT8), both of which are limited to 256 distinct values. Here, we introduce the approach to simulate brain activities using the integer format.

The mapping of values from a continuous set to a countable, smaller set is called quantization in the signal processing and AI domains. The principle of uniform quantization can be written as

$$x \approx \hat{x} = s(x_{\text{int}} - z), \tag{4}$$

where x and x_{int} are the original real-valued variable and the quantized variable, respectively. $\hat{x}$ is the approximate value represented by the low-precision data. The quantization is primarily a simple affine transformation with two parameters, namely the scaling factor s and the zero point z. Thus, the quantization function is defined as

$$x_{\text{int}} = \text{clamp}(\lceil\frac{x}{s}\rfloor + z; q_{\min}, q_{\max}) \tag{5}$$

where $\lceil\cdot\rfloor$ corresponds to the round function. The equation rounds the scaled real-valued variable to an integer and then clamps it to the range of the integer format ($[q_{\min}, q_{\max}]$).

We use semi-dynamic quantization to determine the scaling factor s and zero point z for both the DMF model and the hemodynamic model. During the warm-up stage, the model quickly converges near the fixed point. Then we determine the quantization parameters based on the variable ranges in the QPS stage. We can calculate the power-of-two scaling factor for each of them using

$$\begin{cases} s_{\mathbf{y}} = 2^{-\lfloor \log_2(q_{\max}/y_m) \rfloor} \\ y_m = \begin{cases} \max(|\mathbf{y}|) & \text{symmetric quantization} \\ \frac{\max(\mathbf{y}) - \min(\mathbf{y})}{2} & \text{asymmetric quantization} \end{cases} \end{cases} \tag{6}$$

where $\lfloor \cdot \rfloor$ represents the floor function, and $|\cdot|$ represents the absolute value. q_{max} is the maximum positive value of the signed integer type, and it is 127 for INT8 here. The bold $\mathbf{y}$ is the variable that needs to be quantized. As most brain-inspired computing chips only support per-tensor quantization, we treat the same variable within each node of a model as a vector and apply uniform quantization parameters to all elements in it. Thus, the shape of $\mathbf{y}$ here is (N, T_{QPS}), where N denotes the number of nodes in a model and T_{QPS} is the number of time steps in the QPS stage. y_m represents the unilateral maximum range that needs to be expressed by the integer type, which can be obtained using the extreme values of the vector in the QPS stage. Similarly, the zero point z can also be calculated by

$$z_{\mathbf{y}} = \begin{cases} 0 & \text{symmetric quantization,} \\ \frac{\max(\mathbf{y}) + \min(\mathbf{y})}{2} & \text{asymmetric quantization.} \end{cases} \tag{7}$$

In the subsequent simulation stage, all variables are represented as integers. We can envision scenarios where the ranges of variables change significantly during model execution. If such changes are predictable, semi-dynamic quantization could still be employed. For instance, when simulating brain lesions[88], perturbations are typically introduced to a baseline resting-state brain model. These perturbations can cause the model to transfer from one stable state to another. Since the timing of these perturbations is known as a priori, high-precision floating-point numbers could be used for the transition phase, while a new low-precision quantization scheme could be employed once another stable state is reached. By iteratively applying the warm-up and simulation stages, the semi-dynamic quantization method could be potentially extended to more complex scenarios.

Influences of range-based group-wise quantization

We propose a range-based group-wise quantization method to address the spatial heterogeneity of the hemodynamic model. Figure 2b shows that some variables exhibit a multimodal distribution across the whole-brain model, where the values from different brain regions tend to cluster within several distinct ranges. The most straightforward way to reduce quantization errors from using a single set of parameters across brain regions is to group regions by distribution peaks and calculate parameters for each group separately. However, the variable distributions vary significantly across different empirical data and model parameters, leading to unstable results with inconsistent numbers of groups and varying group sizes. Therefore, we adopt a simpler grouping method: we evenly divide the entire range of variables across the whole-brain model into multiple intervals and sequentially group the regions whose maximum variable values fall within the same interval, from the lowest to the highest. The number of groups (i.e., the number of intervals) can be determined based on the model size (i.e., the number of brain regions), preventing excessive computational overhead from too many groups.

We apply such a range-based group-wise quantization method to three variables in the hemodynamic model: the blood flow f, the blood volume v, and the deoxyhemoglobin content q, as they exhibit the most pronounced spatial heterogeneity. In Supplementary Fig. S3a, we show the goodness-of-fit indicators obtained from simulations at the same sampling points as in Fig. 3a, using low-precision models without group-wise quantization. The distributions are close to those from the high-precision models and the low-precision models with group-wise quantization in Fig. 3a. We further test the distribution of three indicators along a line in the parameter space using different models. The low-precision models without group-wise quantization show satisfactory results in FC correlation and metastability, differing from their group-wise quantized counterparts mainly in slightly lower indicator values and increased G-dependent fluctuations. However, they are almost unusable if we measure the similarity between simulated and empirical functional connectivity dynamics (FCD) matrices by calculating the Kolmogorov–Smirnov (KS) distance between their probability distribution functions (pdfs). This demonstrates that group-wise quantization significantly aids low-precision models in reproducing the spatiotemporal dynamics of the brain network. Nevertheless, since model inversion does not require all indicators for parameter evaluation, we believe that low-precision models without group-wise quantization still retain a certain practical value.

Determining the time steps for multi-timescale simulation

The state variables within a dynamic system may fluctuate on different timescales temporally. The brain dynamics model serves as a good example of such temporal heterogeneity. As demonstrated in Supplementary Fig. S1, some state variables (f, v, and q in the hemodynamic model) exhibit much slower temporal dynamics compared to the others such as S in the DMF model and z in the hemodynamic model. In other words, the difference in some variables between two adjacent time steps ($\Delta x[t]$) is smaller. Therefore, it is necessary to use a smaller scaling factor ($\frac{\Delta x}{s} > 1$) to distinguish consecutive values of such "slower" variables ($x[t]$ and $x[t+1] = x[t] + \Delta x[t]$). However, as aforementioned, a smaller scaling factor will also sacrifice the representation range of the integer, resulting in substantial truncation errors. To resolve this dilemma, we increase the simulation time step size Δt for the simulation because a larger Δt may lead to a larger temporal difference Δx with the same differential equation, which may relax the requirement for the scaling factor s. Meanwhile, a larger time step can also reduce the number of iterations needed for a given simulation time, thus accelerating the entire simulation process.

However, this does not address all challenges, as another dilemma emerges in the selection of the time step size: if only the "slower" variables are considered and a large time step is used for all variables, it may affect the accuracy of the Euler integration; while if a smaller time step is employed, we will again encounter the scaling factor problem where the "slower" variables cannot be correctly quantized to lower precision. Supplementary Fig. S1 shows the simulated brain dynamics signals using different global time step sizes. When using the same time step as the high-precision simulation (e.g., 0.01 s), the "slower" variables show such small temporal variations that they cannot be represented by integers, resulting in quantized variables remaining constant over time (issue A). When the time step is increased to 0.18 s, z exhibits normal temporal evolution, but the variables f, v, and q still show periods of constant values, with the "slowest" q being notably more affected than f and v (issue B). As the time step further increases (e.g., 0.36 s and 0.54 s), while all variables no longer show significant value stagnation, the oscillation amplitudes of the "faster" variable (z) significantly exceed those of the high-precision model, indicating stability issues in the dynamical model due to excessive time steps (issues C and D). Overall, it is difficult to find a single global time step that meets the requirements of all variables.

In fact, for a state variable x with a time derivative $\dot{x}$, the low-precision simulation is constrained by

$$\frac{x_{\mathrm{range}}}{\min_{\dot{x}\neq 0}|\dot{x}|} \leq 2^{b-1}\Delta t \tag{8}$$

where $\min_{\dot{x}\neq 0}|\dot{x}|$ denotes the minimum effective absolute value (i.e., the smallest absolute value that needs to be distinguished from zero) of the time derivative, x_{range} is the maximum unilateral range that must be represented for variable x when quantized to lower precision, b represents the bitwidth of the low-precision integer, and Δt is the simulation time step size. Note that $\min_{\dot{x}\neq 0}|\dot{x}|\Delta t$ represents the minimum effective change in this variable, which needs to be represented by integer 1 in the low-precision simulation; while x_{range} needs to be represented by the maximum integer value (2^{b-1}). Therefore, this equation implies that both ranges of the variable and its time derivative must fit into the integer representation, establishing a lower bound for the simulation time step size. It merits clarification that both x_{range} and $\min_{\dot{x}\neq 0}|\dot{x}|$ are not necessary to be the actual bounds of the high-precision values, as the quantization process tolerates certain levels of rounding and truncation errors. We visualize the relationship between the variable ranges and the time derivatives in Supplementary Fig. S2. The distributions of the variables and their absolute time derivatives are shown in Supplementary Fig. S2a. We calculate a temporal scale index τ by using $(x_{\max} - x_{\min})/2$ to represent the range of the variable x and characterizing its minimum effective non-zero absolute derivative value with the first quartile. From the indices in Supplementary Fig. S2b, we can observe that z requires a much larger simulation time step size compared to S, f, and v demand larger time step sizes over z, and q exhibits the highest lower bound for time step requirements. It should be noted that the indexes only represent the temporal characteristics of the variables and do not necessarily correlate numerically with Δt.

Prior studies[63,73,89] employed a uniform time step across the entire model. However, the above results and analyses reveal that distinct variables necessitate different time step sizes in low-precision simulation. Therefore, we propose a multi-timescale simulation approach for the low-precision brain dynamics model, allowing each variable to update at different time intervals. The term “multi-timescale simulation" is used because the method stems from the different evolution rates among variables in the dynamic system. However, it is important to clarify that this is a computational strategy and is distinct from the neuroscientific concept of temporal scales. Figure 2c illustrates an example of this method. We first set the smallest time step size $\Delta t_{\min}$ for the “fastest” variable (e.g., Δt_S in Fig. 2c), and then set the time steps of other variables to be multiples of $\Delta t_{\min}$. Compared to normal simulation, multi-timescale simulation needs to address the interdependencies among variables with different time steps. We establish two rules to solve the problem. First, if a variable x with a time step of t_x is updated to a value $x[t]$ at time t, its value remains $x[t]$ during the interval from t to $t + t_x$. This rule ensures that the value of each variable is defined at every moment during the simulation. Second, when a variable x with a time step of t_x is updated at time t and depends on a variable y with a different time step of t_y, it needs to use the historical value of y at time $t–t_x$ (i.e., the time of the last update of x) rather than at time $t–t_y$ (i.e., the time of the last update of y). In this way, when the update of a “fast” variable depends on “slow” variables or vice versa, we can select reasonable values for differential equations during the simulation process. Based on the observations from Supplementary Fig. S2 and empirical tuning, we determine the specific time steps for each variable in our simulation. Our experimental results demonstrate that the proposed discretization method is effective, at least within the context of resting-state whole-brain model simulation. However, we acknowledge that this approach has inherent limitations: when a dynamical model cannot achieve effective low-precision simulation with uniform time steps and exhibits high sensitivity to temporal resolution, higher data precision may be the only viable solution to maintain adequate numerical accuracy.

Metaheuristics for model inversion

Model inversion aims to fit the DMF model to empirical FC data obtained from fMRI by tuning the parameters. Due to the excessive number of time steps in the simulation process, it is impossible to use gradient descent based on backpropagation, which is the dominant training method in deep learning. The simplest methods are grid search and random search, which can almost uniformly explore the parameter space. The advantage of these methods is that they can find relatively good solutions on a global scale, and the search algorithm can be easily parallelized, while the disadvantage is that the search process lacks guidance, possibly wasting a lot of time on meaningless search points. Currently, the most commonly used inversion algorithm is based on the expectation-maximization algorithm[90], which guides the parameter update by a Gauss–Newton search for the maximum of the conditional or posterior density. This optimization method exhibits high efficiency like gradient descent, as it can converge rapidly, but it might get stuck in a local optimum and lacks sufficient parallelism for acceleration.

We turn to the metaheuristics algorithms to balance search efficiency, exploration, and parallelism. We adopt a new inversion algorithm based on the PSO[91] in the experiments. In this algorithm, the candidate solutions are represented as a population of particles. Each coordinate of a particle in the space corresponds to a parameter that needs to be optimized, so the position of the particle corresponds to a set of model parameters. The search process is conducted by iteratively moving these particles around in the parameter space. Each iteration consists of two steps: particle movement and solution evaluation. Particle movement is influenced by its local best-known position and the global best-known positions, which may balance the “exploitation” and “exploration” in the search space. After the particle positions are updated, the corresponding parameters need to be input into the model for simulation to evaluate the quality of the solution. The details are demonstrated in Supplementary Algorithm S1. This algorithm can utilize the computing capacities of parallel architectures better, as the simulation and evaluation of each candidate solution can be computed in parallel.

In fact, most population-based optimization algorithms can be readily integrated into our proposed pipeline. This includes CMA-ES algorithms in[89] and grid search methods in[85]. These approaches all require evaluating large numbers of parameter sets (e.g., populations in evolutionary algorithms, particle swarms in PSO, and grid points in grid search), where the evaluation process can be fully parallelized. Therefore, they can take full advantage of the parallelism offered by advanced computing architectures. In these algorithms, both high-precision and low-precision simulations serve as evaluation or cost functions. As long as the goodness-of-fit between low-precision simulation results and empirical data shows similar distribution patterns as high-precision simulations across the parameter space, these search algorithms should achieve valid results with low-precision simulations. This similarity in distribution is what we want to demonstrate in Fig. 3.

We test our low-precision simulation approach with two different algorithms: PSO and CMA-ES, comparing all results against high-precision models. Unlike previous experiments, here we use the larger Human Connectome Project (HCP) dataset containing 1004 participants and employed a validation procedure similar to that described in[89]. We first performed an algorithmic search on the training set, then selected optimal parameters using the validation set, and finally conducted repeated simulations on the test set to calculate FC correlations. The results in Supplementary Table S7 demonstrate that the FC correlations achieved by different algorithm-precision combinations are similar. No significant differences were observed between PSO and CMA-ES results, indicating that the choice of optimization algorithm does not substantially impact the effectiveness of our low-precision approach. The FC correlations are not as high as those reported in[89] because we employ a distinct model with fewer parameters.

Beyond population-based metaheuristics, the proposed pipeline has certain limitations when integrating with other algorithms. First,

algorithms with lower parallelism, such as some Bayesian estimation methods, may not fully utilize the parallel resources of advanced computing architectures, resulting in less pronounced acceleration benefits. Second, the goodness-of-fit obtained from low-precision simulations varies less smoothly with parameters compared to high-precision models (as shown in Fig. 3b), which may make the approaches less suitable for some local optimization algorithms.

Implementation of model operations on brain-inspired computing chips

On brain-inspired computing chips, the cores do not contain fine-grained computing units like the SPs on GPUs; instead, they only have larger and more complex dedicated circuits, such as crossbars or MAC arrays. Therefore, the cores can only support certain predetermined complex functions at matrix or vector granularity, and do not support fine-grained arithmetic operations like those on GPUs. Correspondingly, when programming each core, we can only use coarse-grained instructions such as matrix-vector multiplication. Therefore, to map the model onto the cores, we need to treat the variables of all nodes in the model as vectors, and then decompose the vector-form computation into instructions that the cores support. These steps are closely related to the quantization process, where we select hardware-constrained quantization methods and data types with per-vector parameters. We use a brain-inspired computing chip, TianjicX[42], as the validation platform. It is a programmable platform that executes different operations based on configured instructions as mentioned above. Although TianjicX is a prototype chip designed for academic research, it also has a toolchain that allows us to generate chip instructions using Python programming.

Supplementary Fig. S6 shows the mapping from the DMF model to specialized instructions in more detail. The TianjicX chip only supports about 15 types of instructions, and the DMF model uses 5 of them. The input and output data types are fixed for operations as they are implemented by dedicated circuits. We first introduce several constraints on the quantized model to ensure compatibility with the supported operations. First, the unary operations must use a look-up table (LUT), and the input type must be 8-bit integers. Second, all the operations involving multiplication must use INT8 as the input data type, and the output data type can be either INT8 or high-precision integers (i.e., INT32). Third, the addition operations can use either INT8 or INT32 for the input and output data types, and all the inputs must use the same data type. We then detail the specific implementation with these constraints (see Supplementary Fig. S6). To keep the accuracy as high as possible, we use high-precision integers for all inputs of addition operations (operations ③ and ⑦). We also fuse consecutive unary operations such as constant scaling and nonlinear functions into one LUT operation to reduce computation overhead and mitigate accuracy decline (operation ④). In addition, we utilize asymmetric quantization for the variables whose distribution ranges are far from 0. Despite the potential additional computational overhead of non-zero z, asymmetric quantization can make better use of the limited integer range in this case, thereby significantly reducing the errors caused by quantization.

When model parameters, connectomes, or equations change, code modifications are required as follows: When the numerical values in the structural connectome change, we only need to adjust the connection matrix data configured in the chip; when the connectome scale (e.g., number of brain regions) changes, we need to modify corresponding variables in the code to adjust instruction fields; when firing rate differential equations change, the code needs to be adjusted according to the new equations, which may require longer time for implementation.

Looking at the broader landscape of neuromorphic chips, not all neuromorphic chips share the same programmable characteristics, nor do all specialized neuromorphic chips support the full range of functionalities required for various whole-brain models. However, programmability and configurability has emerged as a dominant trend in the neuromorphic computing field[50]. Therefore, we are highly optimistic about the applicability and flexibility of current and future neuromorphic platforms.

Optimizing data orchestration on GPUs

GPUs employ a hierarchical memory design similar to CPUs. Different memory levels have varying capacities, bandwidths, and latencies, and are shared by different scales of computing resources. The GPU programming model allows control over data placement in the memory hierarchy to some extent. Proper data organization can significantly reduce memory access overhead during program execution. In the parallel simulation of brain dynamics models, different types of variables and data are allocated in the memory hierarchy based on their capacity requirements and access patterns. First, all variables are explicitly stored in the on-chip memory (including the RF and the shared memory) of the GPU whenever possible, and they are only copied to global memory when the on-chip space is insufficient or when the variables need to be transferred to the host (i.e., general-purpose processors). Second, most of the state variables and the intermediate variables in each node such as H_i, x_i, z_i, and f_i are kept in the RF for local calculation in SPs. This is because these values need to be frequently read, modified, and written during each iteration, and they do not need to be shared with other nodes. Third, the average synaptic gating variable, S_i, as a special case of state variables, is stored not only in the local RF but also in the shared memory. As S_i influences the activities of other nodes through the SC (see Eq. 1), it needs to be stored in the shared memory for access by all nodes in the model. The shared memory copy is updated to match the local copy after each iteration. Last, static data at the model level, including the SC matrix and parameters that need to be tuned, will be stored in the shared memory if possible, as they are frequently used by all nodes. If the model is too large for the SC to fit in the shared memory, we will store it in the global memory, which is automatically determined at compile time based on the model size.

Generalization of the proposed pipeline to other brain dynamics models

Different whole-brain models employ distinct differential equations to describe brain regions, and the signals may exhibit different characteristics[92]. The DMF model we primarily focus on evolves from random initial states to fixed points during simulation and exhibits random fluctuations around these fixed points driven by noise. In contrast, oscillator-based models such as the Hopf and Kuramoto models demonstrate pronounced periodic oscillation over time, with variables showing regular temporal patterns rather than the stochastic fluctuations observed in the DMF model. Therefore, to demonstrate the generalization of our proposed pipeline, we conduct additional validation experiments over the Hopf model[19,20].

The whole-brain dynamics of the Hopf model can be described by

$$\frac{dz_j}{dt} = (a + i\omega_j)z_j - |z_j|^2 z_j + g\sum_{k=1}^{N} C_{jk}(z_k - z_j) + \eta_j, \tag{9}$$

where $z_j = x_j + iy_j$ is the complex state variable of the brain region j, $|z_j|$ is the module of z_j, the parameter a is the bifurcation parameter, which we make constant across nodes, ω_j is the intrinsic angular frequency, g is a global scaling of the connectivity C, and η_j is uncorrelated white noise. From the modeling foundation perspective, the Hopf model represents a phenomenological approach to whole-brain dynamics, in contrast to the DMF model, which is a biophysically grounded model derived from microscopic spiking neuron networks. From the dynamical behavior perspective, the Hopf model under certain parameter regimes exhibits oscillatory dynamics mentioned in

ref. 93, fundamentally different from the DMF model, which converges to fluctuations around a fixed-point attractor. The difference allows us to validate our approach across distinct dynamical regimes.

We first validate the low-precision simulation of the Hopf model. Since the Hopf model we employ has only two parameters (the bifurcation parameter a and the global scaling parameter g), we directly computed the correlation between simulated and empirical FC results across the two-dimensional parameter space, and compared the correlation distributions of high-precision and low-precision models. Supplementary Fig. S10 demonstrates that the quantization approach can maintain strong correspondence with high-precision simulations for oscillatory dynamics. Specifically, the correlation between high-precision and low-precision FC results across the parameter space can reach $r = 0.9968$ ($p < 10^{-4}$), indicating that the low-precision model can effectively capture the parameter-dependent FC patterns observed in the high-precision model. Our results are also consistent with previous findings[20,94], demonstrating that the fitting performance improves (with higher correlation) when the scaling factor is sufficiently large. Additionally, due to differences in the empirical data and the computational method for scaling factor normalization, the specific optimal regime we identify shows some variation from previous work. Furthermore, we note that our analysis in Supplementary Fig. S10 does not distinguish between stable and unstable regions[20] within the parameter space. If we focus exclusively on the stable dynamical regions, the correlation between low-precision and high-precision simulation results would likely be even stronger.

We also validate the acceleration potential of the Hopf model simulation on the advanced computing architectures. We conduct experiments over the Hopf model with 68, 148, and 512 nodes, and compare the performance of CPU and brain-inspired computing chip implementations. The results are summarized in Supplementary Table S6. Unlike the DMF model, the Hopf model requires smaller time steps for simulation, resulting in significantly longer simulation times on CPU, with simulation time accounting for a higher proportion of the total inversion time. Additionally, the warm-up phase in the brain-inspired computing chip pipeline also consumes more time. Despite these challenges, the brain-inspired computing chip pipeline still demonstrates significant acceleration compared to the CPU implementation, achieving speedup factors of up to 90.3 × for simulation time and 30.4 × for overall inversion time in the 512-node case.

Scalability to larger numbers of parameters

Recent work emphasizes the need for inferring whole-brain models with larger numbers of parameters[73,89], including region-specific parameters. The proposed pipeline can scale effectively to accommodate such increased parameter complexity. From the quantization perspective, the number of model parameters does not fundamentally change the basic quantization workflow. Regardless of how many tunable parameters exist in the model, the quantization process relies on determining quantization parameters based on floating-point results during the warm-up stage and the QPS stage. We explicitly perform model inversion using the proposed low-precision simulation to train a model with region-specific local excitatory recurrence parameters (w) on empirical data. We use the PSO algorithm to search for optimal parameters. Here, the noise amplitude is also included as an optimizable parameter. We conducted 30 repeated simulations using both high-precision and low-precision models with the searched parameters. The correlation between low-precision simulated and empirical FC results is, on average, 0.7299 ($p < 10^{-4}$), with an SD of 0.0115. The correlation between high-precision simulated and empirical FC results also reaches 0.7228 ($p < 10^{-4}$), with an SD of 0.0093. Meanwhile, the correlation between the FC results from the two precisions reaches 0.9603 on average ($p < 10^{-4}$). As a baseline, the correlation between the empirical SC and FC is 0.4545. The results demonstrate that our quantization approach maintains high fidelity even with increased parameter complexity. In addition, we find that region-specific parameters lead to larger difference of the numerical ranges across brain regions. The proposed group-wise quantization strategy can effectively handle this heterogeneity.

From the hardware acceleration perspective, the primary impact of increasing the parameter number is on the population size and iteration count of the population-based optimization algorithms like PSO and CMA-ES, as higher-dimensional search spaces demand greater search efforts. However, these algorithmic changes do not affect the applicability of our proposed method. In fact, larger population sizes increase the parallelism potential of the optimization algorithm, allowing better utilization of more parallel hardware resources and potentially increasing the advantage of GPUs and brain-inspired computing chips over CPUs. Even when the parallel architectures cannot simultaneously evaluate the entire population, all computations can be completed through simple time-division multiplexing. The number of iterations clearly does not impact the hardware deployment feasibility.

In summary, the proposed approaches can scale to larger parameter spaces, with the quantization strategy adapting naturally to increased model complexity while the parallel hardware architectures providing enhanced benefits as computational demands grow.

Generalization to diverse neuromorphic platforms

Brain-inspired computing chips are inherently advantageous for this neuroscience application due to common architectural features inspired by the brain, like data locality, massive parallelism, and specialized circuit efficiency[50]. Our proposed methods are designed to leverage these characteristics, making neuromorphic platforms a natural fit for modeling macroscopic brain dynamics.

However, significant differences between platforms pose adaptation challenges. Many neuromorphic platforms, especially analog designs, are highly optimized for the efficient simulation of spiking neurons[95]; however, the specialization makes them unable to directly support the non-spiking models we use. Drawing from the Neural Engineering Framework[93,95,96], the primary hurdle is “Representation,” i.e., the whole-brain models use continuous signals, not spikes. The “Transformation” and “Dynamics" principles, conversely, are fundamentally similar and may require only implementation adjustments.

Adapting our framework to digital chips is straightforward once they support non-spiking data types. Analog circuits, in contrast, require deeper modifications at both the hardware and algorithmic levels. Hardware adjustments would involve re-purposing neuron circuits for non-spiking differential equations, or alternatively, designing spike-based encoding schemes. Synaptic circuits would also need to process these new signal types and support more complex recurrent dynamics. Algorithmically, quantization methods would need to be adapted for the target hardware, for instance, by replacing discrete integer constraints with noise constraints for analog circuits.

Fortunately, the evolution of neuromorphic computing is likely to simplify this process. A trend towards hybrid, configurable architectures that support both spiking and non-spiking models is evident in platforms like the TianjicX chip[42], Loihi 2[38], SpiNNaker 2[30], BrainScaleS 2[36], and PAICORE[52]. This shift toward mixed-signal and reconfigurable designs[50] will broaden the applicability of our pipeline, making future platforms increasingly suitable for macroscopic brain modeling.

Estimating the computational performance of different architectures

To compare the performance of model inversion implementations across different architectures, we need to measure or estimate the time consumed during the inversion process. We use an 8-core AMD Ryzen 7 7800X3D as the baseline CPU, paired with 64GB DDR5 memory to form the general-purpose platform, running Windows 11

24H2. All experiments involving the general-purpose platform (including the CPU-based model inversion and the host-dependent steps of model inversion based on parallel computing platforms) are conducted with this setup. We implement the models and algorithms and measure their execution times on the general-purpose platform with MATLAB, as it is currently one of the most commonly used tools in this field and can easily utilize the CPU's multi-core parallelization and vectorization acceleration.

We select TianjicX[42] and NVIDIA RTX 4060 Ti as the experimental platforms for the brain-inspired computing chip and the GPU, respectively. Detailed specifications for both platforms are provided in Supplementary Table S1. Their computing power is relatively comparable. On the GPU platform, we implement parallel simulation of coarse-grained brain dynamics models with CUDA, and use Nsight System to measure the kernel execution time, data transfer time between host and device, and latency hiding. The total time consumption for the entire inversion process is calculated based on the GPU-based inversion workflow (detailed in Supplementary Fig. S8). As mentioned above, the times consumed by host-dependent steps, such as pre-processing and post-processing, are measured on the general-purpose platform.

The situation with the brain-inspired computing chip is different. TianjicX is a prototype chip with a less comprehensive interface and software support compared to the commercial GPUs. We implement model simulation and on-chip data transfer using specialized instructions, and evaluate the time consumption with a cycle-accurate chip simulator. For fair comparison, we assume the chip has the same PCIe 4.0 x8 interface as the GPU, and estimate the data transfer time between the host and device at 80% bandwidth utilization. The inversion workflow based on the brain-inspired computing chip is illustrated in Supplementary Fig. S7. Similarly, we calculate the total time consumption based on the duration of each step. In addition, we did not consider group-wise quantization when measuring the quantization overhead on the host, as this step is not always necessary for model inversion. Other experimental details can be found in Supplementary Table S2.

Goodness-of-fit indicators and other quality metrics

To validate the performance of low-precision models in reproducing empirical brain network characteristics and to compare their performance with high-precision models, we employ a comprehensive set of methods to quantify and contrast the properties of these models. Firstly, we calculate the FC matrix for each subject based on the Pearson correlation coefficient between the BOLD time series. The simulated time series length strictly corresponds to the time series length provided in the dataset (i.e., 14.4 min). The FC matrices are then averaged across subjects to obtain group-level connectivity patterns for model validation. To capture the spatiotemporal dynamics of FC, we compute the FCD matrix, which represents the similarity of FC matrices over time. For each participant, the FC within each sliding time window is calculated, and the similarity between FC matrices of each window is measured using Pearson's correlation to generate the FCD matrix. The pdf of the FCD matrix is obtained by folding the upper triangular terms of the matrix into a histogram and normalizing it to an area of one[97]. We assess the similarity between simulated and empirical FCD matrices by calculating the KS distance between their pdfs, with a smaller KS distance indicating better model performance in reproducing empirical data.

A hallmark of brain activities is the metastable neural dynamics, which can flexibly reconfigure distributed functional sub-networks to balance the competing demands of functional segregation and integration, thereby supporting the brain's diverse and complex functions[11,98]. Research has shown that brain activities typically operate near a critical point[80]. This organizational state provides the system with essential stability while still allowing for efficient information transmission and processing, which corresponds to a high degree of neural metastability. Thus, in this study, metastable neural dynamics also serves as a crucial criterion for model validation. To quantify neuronal population dynamics, we use the Kuramoto order parameter to measure phase coherence. Synchrony indicates average instantaneous phase coherence, while metastability, defined as the standard deviation of the order parameters, reflects the temporal variability in synchrony.

To evaluate the ability of low-precision models to reproduce the spatial topology of empirical data, we utilize the Brain Connectivity Toolbox[99] to characterize the functional properties of each node using local topological properties derived from group-level functional networks. These properties include degree centrality, clustering coefficient, betweenness centrality, participation coefficient, Z-score of within-module degree, local efficiency, and node strength. These metrics quantify the role of nodes in promoting network integration across multiple brain regions and efficient processing within local regions[100,101].

For temporal complexity measures, we employ the sample entropy as our primary metric. Sample Entropy quantifies the regularity of a time series by computing the negative natural logarithm of the conditional probability that two similar sequences will maintain their similarity when extended by one data point[102]. A higher sample entropy value reflects more complex nonlinear dynamics (e.g., chaotic or stochastic behaviors). This metric is widely employed in biomedical signal analysis, particularly for characterizing dynamic properties of neuroimaging data such as fMRI and EEG[103–105].

Through these methods, we comprehensively evaluate the ability of low-precision models to capture both the dynamic and static aspects of brain network organization, providing critical insights into their applicability in brain network modeling.

Multimodal neuroimaging dataset and data processing

We utilize T1w MRI, dMRI, and rs-fMRI from 45 subjects from the HCP Retest data release[106]. The subject population appears to be relatively small, as the primary objective of using the data in our study is to validate the effectiveness and accuracy of the framework, rather than conducting extensive statistical analyses of population features. Further details regarding data collection can be found on the HCP website (https://www.humanconnectome.org/study/hcp-young-adult). All MRI data are pre-processed according to the HCP minimal pre-processing pipeline[107]. After the HCP dMRI step, fiber tracking is performed using the MRtrix3 package[108]. Details of the tractography procedure can be found elsewhere[109]. Following the data pre-processing, cortical parcellation is defined using the Desikan–Killiany atlas[110] and Destrieux atlas[111]. Subsequently, we extract weighted connectivity matrices for each individual, including 68×68 matrices based on the Desikan–Killiany atlas and 148×148 matrices based on the Destrieux atlas, for both SC and FC. In addition to individual SC matrices, a group-averaged SC matrix is generated by retaining connections that are present in at least 50% of the subjects within the group. To ensure consistency across models, the SC matrices are normalized such that their maximum entry is set to 0.2. These matrices are then used to construct individual and group-level brain models. Further details regarding the data processing and feature extraction can be found elsewhere[88]. In the indicator distribution experiments, we primarily employ the 68-node group-averaged model. For computational performance evaluation, we utilize both 68-node and 148-node models, and synthesize random SC and FC data ranging from 64 to 512 nodes to explore scalability. We also use a larger HCP 1004 dataset, which has been used in previous work[89], to present the key inversion results in Supplementary Table S7.

Data availability

The HCP T1w MRI, dMRI, and rs-fMRI data used in this study are available at https://www.humanconnectome.org/study/hcp-young-adult. The processed individual data, group-averaged data, and the synthetic data are available at https://github.com/GnohzZ/Brain-Dynamics-Modeling-Acceleration.

Code availability

Source codes for reproducing the results in this paper are available at https://github.com/GnohzZ/Brain-Dynamics-Modeling-Acceleration, with a Zenodo link https://zenodo.org/records/17104941[112].

References

1. Hoel, E. P., Albantakis, L. & Tononi, G. Quantifying causal emergence shows that macro can beat micro. *Proc. Natl. Acad. Sci. USA* **110**, 19790–19795 (2013).
2. Hodgkin, A. L. & Huxley, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.* **117**, 500 (1952).
3. Izhikevich, E. M. Simple model of spiking neurons. *IEEE Trans. Neural Netw.* **14**, 1569–1572 (2003).
4. Maass, W. & Bishop, C. M. *Pulsed Neural Networks* (MIT Press, 2001).
5. Hong, C. et al. Spaic: a spike-based artificial intelligence computing framework. *IEEE Comput. Intell. Mag.* **19**, 51–65 (2024).
6. Fang, W. et al. SpikingJelly: an open-source machine learning infrastructure platform for spike-based intelligence. *Sci. Adv.* **9**, eadi1480 (2023).
7. Eppler, J. M., Helias, M., Muller, E., Diesmann, M. & Gewaltig, M.-O. PyNEST: a convenient interface to the nest simulator. *Front. Neuroinform.* **2**, 363 (2009).
8. Stimberg, M., Brette, R. & Goodman, D. F. Brian 2, an intuitive and efficient neural simulator. *eLife* **8**, e47314 (2019).
9. Chen, G., Scherr, F. & Maass, W. A data-based large-scale model for primary visual cortex enables brain-like robust and versatile visual processing. *Sci. Adv.* **8**, eabq7592 (2022).
10. Kong, X. et al. Sensory-motor cortices shape functional connectivity dynamics in the human brain. *Nat. Commun.* **12**, 6373 (2021).
11. Vohryzek, J., Cabral, J., Vuust, P., Deco, G. & Kringelbach, M. L. Understanding brain states across spacetime informed by whole-brain modelling. *Philos. Trans. R. Soc. A* **380**, 20210247 (2022).
12. Breakspear, M. Dynamic models of large-scale brain activity. *Nat. Neurosci.* **20**, 340–352 (2017).
13. Deco, G. & Kringelbach, M. L. Great expectations: using whole-brain computational connectomics for understanding neuropsychiatric disorders. *Neuron* **84**, 892–905 (2014).
14. Deco, G., Tononi, G., Boly, M. & Kringelbach, M. L. Rethinking segregation and integration: contributions of whole-brain modelling. *Nat. Rev. Neurosci.* **16**, 430–439 (2015).
15. Siettos, C. & Starke, J. Multiscale modeling of brain dynamics: from single neurons and networks to mathematical tools. *Wiley Interdiscip. Rev. Syst. Biol. Med.* **8**, 438–458 (2016).
16. Wilson, H. R. & Cowan, J. D. Excitatory and inhibitory interactions in localized populations of model neurons. *Biophys. J.* **12**, 1–24 (1972).
17. Wilson, H. R. & Cowan, J. D. A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue. *Kybernetik* **13**, 55–80 (1973).
18. Cumin, D. & Unsworth, C. Generalising the Kuramoto model for the study of neuronal synchronisation in the brain. *Phys. D Nonlinear Phenom.* **226**, 181–196 (2007).
19. Deco, G., Kringelbach, M. L., Jirsa, V. K. & Ritter, P. The dynamics of resting fluctuations in the brain: metastability and its dynamical cortical core. *Sci. Rep.* **7**, 3095 (2017).
20. Ponce-Alvarez, A. & Deco, G. The Hopf whole-brain model and its linear approximation. *Sci. Rep.* **14**, 2615 (2024).
21. Deco, G. et al. Resting-state functional connectivity emerges from structurally and dynamically shaped slow linear fluctuations. *J. Neurosci.* **33**, 11239–11252 (2013).
22. Schirner, M., McIntosh, A. R., Jirsa, V., Deco, G. & Ritter, P. Inferring multi-scale neural mechanisms with brain network modelling. *eLife* **7**, e28927 (2018).
23. Kringelbach, M. L. et al. Dynamic coupling of whole-brain neuronal and neurotransmitter systems. *Proc. Natl. Acad. Sci. USA* **117**, 9566–9576 (2020).
24. Freyer, F. et al. Biophysical mechanisms of multistability in resting-state cortical rhythms. *J. Neurosci.* **31**, 6353–6361 (2011).
25. Siddiqi, S. H. et al. Brain stimulation and brain lesions converge on common causal circuits in neuropsychiatric disease. *Nat. Hum. Behav.* **5**, 1707–1716 (2021).
26. Jirsa, V. K. et al. The virtual epileptic patient: individualized whole-brain models of epilepsy spread. *Neuroimage* **145**, 377–388 (2017).
27. Patow, G. et al. Whole-brain modeling of the differential influences of amyloid-beta and tau in alzheimer's disease. *Alzheimer's Res. Ther.* **15**, 210 (2023).
28. Sermon, J. J. et al. Sub-harmonic entrainment of cortical gamma oscillations to deep brain stimulation in parkinson's disease: model based predictions and validation in three human subjects. *Brain Stimul.* **16**, 1412–1424 (2023).
29. Akopyan, F. et al. TrueNorth: design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip. In *Proc. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* Vol. 34, 1537–1557 (IEEE, 2015).
30. Höppner, S. et al. The SpiNNaker 2 processing element architecture for hybrid digital neuromorphic computing. Preprint at *arXiv* https://doi.org/10.48550/arXiv.2103.08392 (2021).
31. Liu, C. et al. Memory-efficient deep learning on a SpiNNaker 2 prototype. *Front. Neurosci.* **12**, 840 (2018).
32. Mayr, C., Hoeppner, S. & Furber, S. SpiNNaker 2: a 10 million core processor system for brain simulation and machine learning-keynote presentation. in *Communicating Process Architectures 2017 & 2018* (Pedersen, J. B. et al.) 277–280 (IOS Press, 2019).
33. Furber, S. B. et al. Overview of the spinnaker system architecture. *IEEE Trans. Comput.* **62**, 2454–2467 (2012).
34. Furber, S. B., Galluppi, F., Temple, S. & Plana, L. A. The spinnaker project. *Proc. IEEE* **102**, 652–665 (2014).
35. Schemmel, J., Fieres, J. & Meier, K. Wafer-scale integration of analog neural networks. In *Proc. 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)* 431–438 (IEEE, 2008).
36. Pehle, C. et al. The BrainScaleS-2 accelerated neuromorphic system with hybrid plasticity. *Front. Neurosci.* **16**, 795876 (2022).
37. Davies, M. et al. Loihi: a neuromorphic manycore processor with on-chip learning. *IEEE Micro* **38**, 82–99 (2018).
38. Orchard, G. et al. Efficient neuromorphic signal processing with Loihi 2. In *Proc. 2021 IEEE Workshop on Signal Processing Systems (SiPS)* 254–259 (IEEE, 2021).
39. Ma, D. et al. Darwin: a neuromorphic hardware co-processor based on spiking neural networks. *J. Syst. Archit.* **77**, 43–51 (2017).
40. Ma, D. et al. Darwin3: a large-scale neuromorphic chip with a novel ISA and on-chip learning. *Natl. Sci. Rev.* **11**, nwae102 (2024).
41. Pei, J. et al. Towards artificial general intelligence with hybrid Tianjic chip architecture. *Nature* **572**, 106–111 (2019).
42. Ma, S. et al. Neuromorphic computing chip with spatiotemporal elasticity for multi-intelligent-tasking robots. *Sci. Robot.* **7**, eabk2948 (2022).
43. Chen, T. et al. DianNao: a small-footprint high-throughput accelerator for ubiquitous machine-learning. *ACM SIGARCH Comput. Archit. News* **42**, 269–284 (2014).
44. Chen, Y.-H., Krishna, T., Emer, J. S. & Sze, V. Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE J. Solid State Circuits* **52**, 127–138 (2016).
45. Jouppi, N. P. et al. In-datacenter performance analysis of a tensor processing unit. In *Proc. 44th Annual International Symposium on Computer Architecture* 1–12 (IEEE, 2017).

46. Chen, Y.-H., Yang, T.-J., Emer, J. & Sze, V. Eyeriss v2: a flexible accelerator for emerging deep neural networks on mobile devices. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **9**, 292–308 (2019).
47. NVIDIA Deep Learning Accelerator. https://nvdla.org/index.html.
48. Liao, H. et al. Ascend: a scalable and unified architecture for ubiquitous deep neural network computing: industry track paper. In *Proc. 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* 789–801 (IEEE, 2021).
49. Li, G. et al. Brain-inspired computing: a systematic survey and future trends. *Proc. IEEE* **112**, 544–584 (2024).
50. Kudithipudi, D. et al. Neuromorphic computing at scale. *Nature* **637**, 801–812 (2025).
51. Zhang, Y. et al. A system hierarchy for brain-inspired computing. *Nature* **586**, 378–384 (2020).
52. Zhong, Y. et al. PAICORE: a 1.9-million-neuron 5.181-tsops/w digital neuromorphic processor with unified SNN-ANN and on-chip learning paradigm. *IEEE J. Solid State Circuits* **60**, 651–671 (2024).
53. Manna, D. L., Vicente-Sola, A., Kirkland, P., Bihl, T. J. & Di Caterina, G. Simple and complex spiking neurons: perspectives and analysis in a simple stdp scenario. *Neuromorphic Comput. Eng.* **2**, 044009 (2022).
54. Dey, S. & Dimitrov, A. Mapping and validating a point neuron model on Intel's neuromorphic hardware Loihi. *Front. Neuroinform.* **16**, 883360 (2022).
55. Bian, S. & Magno, M. Evaluating spiking neural network on neuromorphic platform for human activity recognition. In *Proc. 2023 ACM International Symposium on Wearable Computers* 82–86 (Association for Computing Machinery, 2023).
56. Zins, N. & An, H. Reproducing fear conditioning of rats with unmanned ground vehicles and neuromorphic systems. In *Proc. 2023 24th International Symposium on Quality Electronic Design (ISQED)* 1–7 (IEEE, 2023).
57. Zheng, H. et al. Temporal dendritic heterogeneity incorporated with spiking neural networks for learning multi-timescale dynamics. *Nat. Commun.* **15**, 277 (2024).
58. Wang, H., Peng, H., Chang, Y. & Liang, D. A survey of GPU-based acceleration techniques in MRI reconstructions. *Quant. Imaging Med. Surg.* **8**, 196 (2018).
59. Xanthis, C. G., Venetis, I. E., Chalkias, A. & Aletras, A. H. Mrisimul: a GPU-based parallel approach to MRI simulations. *IEEE Trans. Med. Imaging* **33**, 607–617 (2013).
60. Stone, S. S. et al. Accelerating advanced MRI reconstructions on GPUs. In *Proc. 5th Conference on Computing Frontiers* 261–272 (Association for Computing Machinery, 2008).
61. Xanthis, C. G., Venetis, I. E. & Aletras, A. H. High performance MRI simulations of motion on multi-GPU systems. *J. Cardiovasc. Magn. Reson.* **16**, 48 (2014).
62. Eklund, A., Dufort, P., Forsberg, D. & LaConte, S. M. Medical image processing on the GPU–past, present and future. *Med. Image Anal.* **17**, 1073–1094 (2013).
63. Kong, X. et al. Anatomical and functional gradients shape dynamic functional connectivity in the human brain. Preprint at *bioRxiv* https://doi.org/10.1101/2021.03.15.435361 (2021).
64. Zhou, S. et al. DoReFa-Net: training low bitwidth convolutional neural networks with low bitwidth gradients. *CoRR.* **abs/1606.06160** (2016).
65. Courbariaux, M., Bengio, Y. & David, J.-P. BinaryConnect: training deep neural networks with binary weights during propagations. In *Proc. Advances in Neural Information Processing Systems* **28** (Curran Associates Inc., 2015).
66. Lin, Z., Courbariaux, M., Memisevic, R. & Bengio, Y. Neural networks with few multiplications. In *International Conference on Learning Representations* (2016).
67. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R. & Bengio, Y. Quantized neural networks: training neural networks with low precision weights and activations. *J. Mach. Learn. Res.* **18**, 1–30 (2018).
68. Zhou, A., Yao, A., Guo, Y., Xu, L. & Chen, Y. Incremental network quantization: towards lossless CNNs with low-precision weights. In *International Conference on Learning Representations* (2017).
69. Deng, L., Li, G., Han, S., Shi, L. & Xie, Y. Model compression and hardware acceleration for neural networks: a comprehensive survey. *Proc. IEEE* **108**, 485–532 (2020).
70. Nagel, M. et al. A white paper on neural network quantization. Preprint at *arXiv* https://doi.org/10.48550/arXiv.2106.08295 (2021).
71. Deco, G., Cruzat, J. & Kringelbach, M. L. Brain songs framework used for discovering the relevant timescale of the human brain. *Nat. Commun.* **10**, 583 (2019).
72. Demirtaş, M. et al. Hierarchical heterogeneity across human cortex shapes large-scale neural dynamics. *Neuron* **101**, 1181–1194 (2019).
73. Wang, P. et al. Inversion of a large-scale circuit model reveals a cortical hierarchy in the dynamic resting human brain. *Sci. Adv.* **5**, eaat7854 (2019).
74. Wong, K.-F. & Wang, X.-J. A recurrent network mechanism of time integration in perceptual decisions. *J. Neurosci.* **26**, 1314–1328 (2006).
75. Laumann, T. O. et al. On the stability of bold fMRI correlations. *Cereb. Cortex* **27**, 4719–4732 (2017).
76. El Houssaini, K., Ivanov, A. I., Bernard, C. & Jirsa, V. K. Seizures, refractory status epilepticus, and depolarization block as endogenous brain activities. *Phys. Rev. E* **91**, 010701 (2015).
77. Richardson, M. P. Large scale brain models of epilepsy: dynamics meets connectomics. *J. Neurol. Neurosurg. Psychiatry* **83**, 1238–1248 (2012).
78. Breakspear, M. et al. A unifying explanation of primary generalized seizures through nonlinear brain modeling and bifurcation analysis. *Cereb. Cortex* **16**, 1296–1313 (2006).
79. Deco, G. & Jirsa, V. K. Ongoing cortical activity at rest: criticality, multistability, and ghost attractors. *J. Neurosci.* **32**, 3366–3375 (2012).
80. Liang, J., Wang, S.-J. & Zhou, C. Less is more: wiring-economical modular networks support self-sustained firing-economical neural avalanches for efficient processing. *Natl Sci. Rev.* **9**, nwab102 (2022).
81. Váša, F. et al. Effects of lesions on synchrony and metastability in cortical networks. *Neuroimage* **118**, 456–467 (2015).
82. Deco, G. & Kringelbach, M. L. Metastability and coherence: extending the communication through coherence hypothesis using a whole-brain computational perspective. *Trends Neurosci.* **39**, 125–135 (2016).
83. Tognoli, E. & Kelso, J. S. The metastable brain. *Neuron* **81**, 35–48 (2014).
84. Sanz Leon, P. et al. The virtual brain: a simulator of primate brain network dynamics. *Front. Neuroinform.* **7**, 10 (2013).
85. Cakan, C., Jajcay, N. & Obermayer, K. neurolib: a simulation framework for whole-brain neural mass modeling. *Cogn. Comput.* **15**, 1132–1152 (2023).
86. Wang, C. et al. Brainpy, a flexible, integrative, efficient, and extensible framework for general-purpose brain dynamics programming. *eLife* **12**, e86365 (2023).
87. Friston, K. J., Harrison, L. & Penny, W. Dynamic causal modelling. *Neuroimage* **19**, 1273–1302 (2003).
88. Wei, J. et al. Effects of virtual lesions on temporal dynamics in cortical networks based on personalized dynamic models. *NeuroImage* **254**, 119087 (2022).
89. Zhang, S. et al. In vivo whole-cortex marker of excitation-inhibition ratio indexes cortical maturation and cognitive ability in youth. *Proc. Natl. Acad. Sci. USA* **121**, e2318641121 (2024).
90. Friston, K. J. Bayesian estimation of dynamical systems: an application to fMRI. *NeuroImage* **16**, 513–530 (2002).

91. Kennedy, J. & Eberhart, R. Particle swarm optimization. In *Proc. ICNN'95—International Conference on Neural Networks* Vol. 4, 1942–1948 (IEEE, 1995).
92. Cabral, J., Kringelbach, M. L. & Deco, G. Functional connectivity dynamically evolves on multiple time-scales over a static structural connectome: models and mechanisms. *Neuroimage* **160**, 84–96 (2017).
93. Tsur, E. E. *Neuromorphic Engineering: The Scientist's, Algorithms Designer's and Computer Architect's Perspectives on Brain-Inspired Computing* (CRC Press, 2021).
94. Sanz Perl, Y., Escrichs, A., Tagliazucchi, E., Kringelbach, M. L. & Deco, G. Strength-dependent perturbation of whole-brain model working in different regimes reveals the role of fluctuations in brain dynamics. *PLOS Comput. Biol.* **18**, e1010662 (2022).
95. Hazan, A. & Ezra Tsur, E. Neuromorphic neural engineering framework-inspired online continuous learning with analog circuitry. *Appl. Sci.* **12**, 4528 (2022).
96. Eliasmith, C. & Anderson, C. H. *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems* (MIT Press, 2003).
97. Hansen, E. C., Battaglia, D., Spiegler, A., Deco, G. & Jirsa, V. K. Functional connectivity dynamics: modeling the switching behavior of the resting state. *Neuroimage* **105**, 525–535 (2015).
98. Shine, J. M. et al. Human cognition involves the dynamic integration of neural activity and neuromodulatory systems. *Nat. Neurosci.* **22**, 289–296 (2019).
99. Rubinov, M. & Sporns, O. Complex network measures of brain connectivity: uses and interpretations. *Neuroimage* **52**, 1059–1069 (2010).
100. Sporns, O. Network attributes for segregation and integration in the human brain. *Curr. Opin. Neurobiol.* **23**, 162–171 (2013).
101. Lord, L.-D., Stevner, A. B., Deco, G. & Kringelbach, M. L. Understanding principles of integration and segregation using whole-brain computational connectomics: implications for neuropsychiatric disorders. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.* **375**, 20160283 (2017).
102. Richman, J. S. & Moorman, J. R. Physiological time-series analysis using approximate entropy and sample entropy. *Am. J. Physiol. Heart Circ. Physiol.* **278**, H2039–H2049 (2000).
103. Sokunbi, M. O. et al. Resting state fMRI entropy probes complexity of brain activity in adults with ADHD. *Psychiatry Res. Neuroimaging* **214**, 341–348 (2013).
104. Yentes, J. M. et al. The appropriate use of approximate entropy and sample entropy with short data sets. *Ann. Biomed. Eng.* **41**, 349–365 (2013).
105. Abásolo, D., Hornero, R., Espino, P., Alvarez, D. & Poza, J. Entropy analysis of the EEG background activity in alzheimer's disease patients. *Physiol. Meas.* **27**, 241 (2006).
106. Van Essen, D. C. et al. The WU-Minn Human Connectome Project: an overview. *Neuroimage* **80**, 62–79 (2013).
107. Glasser, M. F. et al. The minimal preprocessing pipelines for the Human Connectome Project. *Neuroimage* **80**, 105–124 (2013).
108. Tournier, J.-D. et al. MRtrix3: a fast, flexible and open software framework for medical image processing and visualisation. *Neuroimage* **202**, 116137 (2019).
109. Proix, T. et al. How do parcellation size and short-range connectivity affect dynamics in large-scale brain network models? *Neuroimage* **142**, 135–149 (2016).
110. Desikan, R. S. et al. An automated labeling system for subdividing the human cerebral cortex on MRI scans into gyral based regions of interest. *Neuroimage* **31**, 968–980 (2006).
111. Destrieux, C., Fischl, B., Dale, A. & Halgren, E. Automatic parcellation of human cortical gyri and sulci using standard anatomical nomenclature. *Neuroimage* **53**, 1–15 (2010).
112. Zheng, Z. et al. Released code for modeling macroscopic brain dynamics with brain-inspired computing architecture. https://doi.org/10.5281/zenodo.17104941 (2025).
113. Fischl, B. FreeSurfer. *Neuroimage* **62**, 774–781 (2012).
114. Song, X.-W. et al. REST: a toolkit for resting-state functional magnetic resonance imaging data processing. *PLoS ONE* **6**, e25031 (2011).

Acknowledgements

This work was partially supported by STI 2030—Major Projects 2021ZD0200300, National Natural Science Foundation of China (No. 62276151), and Chinese Institute for Brain Research, Beijing.

Author contributions

Z.Z., J.W., and L.D. conceived the work. Z.Z., Y.X., T.L., Q.G., and X.J. carried out the brain dynamics simulation experiments. Z.Z. and C.L. carried out the architecture implementation and time estimation. Z.Z., J.W., Y.X., and L.D. contributed to the analyses of experimental results. All of the authors contributed to the discussion of quantization, architecture mapping, and experiment design. H.G., G.W., and L.D. led the discussion. Z.Z., J.W., Y.X., and L.D. contributed to the writing of the paper. H.G., G.W., and L.D. supervised the whole project.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41467-025-64470-3.

Correspondence and requests for materials should be addressed to Hao Guo, Gang Wang or Lei Deng.

Peer review information *Nature Communications* thanks Trang-Anh Nghiem, Anand Subramoney, and the other anonymous reviewer(s) for their contribution to the peer review of this work. A peer review file is available.