Multiscale Modeling of Spiking Attractor Neural-Network with Coarse-Grained Representations and U-Net Scaling

Foo Tun Jing

Department of Computer Science and Technology, Tsinghua University

Author Note

Data collection and preliminary analysis were supported by the Computational Neuroscience course at Peking University. Portions of these findings were presented by the course lecturers in Peking, online resources, and large language models. We have no conflicts of interest to disclose.

Correspondence concerning this article should be addressed to Foo Tun Jing, Department of Computer Science and Technology, Tsinghua University, Beijing, China. Email: tunjingfoo@hotmail.com

Abstract

In computational neuroscience, bridging multiple scales of neural activity is essential for understanding large-scale brain dynamics. This paper introduces a multiscale modeling framework for a recurrent, excitatory‑inhibitory leaky integrate‑and‑fire (LIF) spiking neural network with local Gaussian connections and attractor dynamics. The framework employs a coarse-graining method by aggregating fine-grained details of a microscopic Spiking Attractor Neural Network (SANN) into a simpler, coarser representation, which is fed to the macroscopic SANN model that operates at lower resolution (downscaling). After downscaling, the macroscopic SANN model can then be upscaled using a trained 4-level U-Net model. This dual scaling mechanism maintains fidelity between microscopic and macroscopic representations while ensuring computational efficiency. Scaling accuracy is validated using normalization and cosine similarity metrics, showcasing the potential of this framework for accurate and efficient modeling of spiking attractor networks across scales. The framework is designed to facilitate the study of large-scale neural dynamics, providing a computationally efficient tool for simulating the behavior of complex neural systems.

Keywords: Multiscale modeling, Spiking attractor neural networks, Leaky integrate-and-fire (LIF) neurons, U-Net scaling, Computational neuroscience, Neural dynamics, Downscaling, Upscaling

1. Introduction

The study of neural networks and their dynamics is a central focus in computational neuroscience, as it provides valuable insights into how large-scale brain activity emerges from the interaction of individual neurons. While advances in computational power have enabled the development of detailed, biologically realistic models of neural circuits, these models often require substantial computational resources, limiting their application to smaller regions of the brain or less complex networks. In contrast, scalable models that can bridge multiple levels of neural organization, from individual neurons to larger-scale brain regions, remain a significant challenge.

Attractor networks, which are networks of neurons with excitatory interconnections that can settle into stable patterns of firing, offer an elegant approach to modeling neural dynamics. These networks have been widely studied for their role in memory, decision-making, and pattern recognition, leveraging attractor dynamics to form stable activity patterns (Rolls ET., 2010). In this context, we make use of a spiking attractor neural network (SANN), which is characterized by its recurrent excitatory-inhibitory interactions and local Gaussian connectivity (Eliasmith C., 2005), build using leaky integrate-and-fire (LIF) neuron layers which provide a simplified yet powerful representation of neural behavior, and are commonly used in the construction of spiking attractor networks (Burkitt, Anthony., 2006). However, simulating the full dynamics of such networks at the level of individual spiking neurons often comes at a high computational cost, making them less feasible for large-scale simulations.

One promising solution to this problem is the use of coarse-graining modelling, a statistical approach that aggregates fine-grained details (microscopic) into a simpler, coarser representation (macroscopic). In this paper, we introduce a multiscale modeling framework that integrates coarse-graining via block averaging and U-Net scaling to bridge the gap between microscopic and macroscopic SANNs. Only excitatory potentials are concerned in this study. The accuracy of the scaling processes is evaluated using cosine similarity and normalization techniques to ensure fidelity between microscopic and macroscopic representations. This dual scaling approach maintains computational efficiency while preserving the essential dynamics of the original SANN. It is also noted that the use of mean field theory might be plausible, which models the collective activity of large neural populations (La Camera G., 2022). In short, mean-field models reduce the dimensionality of the system, making it feasible to simulate the behavior of large-scale networks with fewer computational resources.

By introducing a scalable solution for spiking attractor networks, this framework enables researchers to model large-scale neural dynamics more effectively, paving the way for improved understanding of neural computation and its implications for brain function.

2. Method

2.1 Microscopic Spiking Attractor Network

2.1.1 LIFlayer

Firstly, we define the LIF membrane potential dynamics as such:

$$C\frac{dV}{dt} = -g_L * (V - V_{rest}) + I_{input}$$

where

$$I_{\text{input}} : \text{Including external and synaptic input}$$

$$g_L : \text{Leak conductance}$$

$$C : \text{Membrane capacitance}$$

The spiking of the neuron is defined as such (H is the Heaviside step function):

$$S = H(V - V_{threshold}) * H(t - t_{spike} - \tau)$$

where

$$\tau : \text{Absolute refractory period, membrane potential will remain at } V_{\text{rest}}$$

This layer is constructed with a 128*128 grid of LIF neurons, threshold of -50mV, reset_value of -70mV, membrane capacitance of 1nF and a refractory period of 5ms.

2.1.2 Synaptic Layer

Both AMPA and GABA synapses are modeled using exponential decay synapses (ExpSynapse). The simulation includes the processes of synaptic current and synaptic conductance, and defines a Gaussian wave packet (Lu W, Rossoni E, Feng J., 2010):

$$I = g * (V - Vrest)$$

$$\tau \frac{dg}{dt} = -g + W * S$$

PyTorch does not provide built-in support for periodic boundary conditions. Therefore, a custom function is defined to handle periodic boundary conditions. Through the scale_up and scale_down functions, scaling between the inhibitory (I) and excitatory (E) networks is achieved, enabling the handling of networks of different sizes within a single framework.

Additionally, Gaussian wave packet is defined for connection weights as such:

$$W_{ij} = W^{\lambda} e^{-\frac{(x_i - x_j)^2 + (y_i - y_j)^2}{\sigma}}$$

where

W^{λ} represents the excitatory or inhibitory output, and σ is the decay factor.

2.1.3 Construction of Spiking Attractor Neural Neural Network

The SpikingAttractorNetwork class models a recurrent excitatory-inhibitory spiking neural network using Leaky Integrate-and-Fire (LIF) neurons and synaptic layers. It initializes two neuronal layers: an excitatory neuron layer (E_neurons) and an inhibitory neuron layer (I_neurons), each represented as grids of neurons governed by LIF dynamics. Four synaptic layers handle connections between these populations: synapsesEE (excitatory-to-excitatory), synapsesEI (excitatory-to-inhibitory), synapsesIE (inhibitory-to-excitatory), and synapsesII (inhibitory-to-inhibitory), with parameters such as connection strength, decay factor, and resting potential tailored for excitatory and inhibitory interactions.

Specifically for synapsesIE and synapsesII, resting potential is set to -80mV and decay factor to 400. In this experiment, bump state is achieved via We=0.23, Wi=0.06.

The update method simulates one timestep of the network's dynamics, updating the membrane potential and spikes of the excitatory and inhibitory layers based on their respective inputs, which combine external stimuli and synaptic currents.

Excitatory Neuron Update

Updates the membrane potential and spikes for the excitatory neurons:

$$E_{\text{potential}}, E_{\text{spike}} = \text{self.E\_neurons.update}(I_E)$$

where I_E is the total input current:

$$I_E = \text{inputE} + \sum(\text{self.synapsesEE.i}) + \sum(\text{self.synapsesIE.i})$$

Inhibitory Neuron Update

Updates the membrane potential and spikes for the inhibitory neurons:

$$I_{\text{potential}}, I_{\text{spike}} = \text{self.I\_neurons.update}(I_I)$$

where I_I is the total input current:

$$I_I = \text{inputI} + \sum(\text{self.synapsesII.i}) + \sum(\text{self.synapsesEI.i})$$

It also updates the synaptic layers to compute the influence of spiking activity on connected neurons.

Updates the synaptic layers by computing new synaptic currents:

Excitatory to Excitatory (EE):

$$\text{self.synapsesEE.update}(E_{\text{spike}}, E_{\text{potential}})$$

Excitatory to Inhibitory (EI):

$$\text{self.synapsesEI.update}(E_{\text{spike}}, I_{\text{potential}})$$

Inhibitory to Excitatory (IE):

$$\text{self.synapsesIE.update}(I_{\text{spike}}, E_{\text{potential}})$$

Inhibitory to Inhibitory (II):

$$\text{self.synapsesII.update}(I_{\text{spike}}, I_{\text{potential}})$$

Finally, the method returns the membrane potentials and spiking statuses for both neuron populations, enabling step-by-step simulation of network activity. This design captures the essential dynamics of spiking attractor networks, including excitatory-inhibitory interactions and recurrent connections, suitable for modeling neural processes like memory and decision-making.

2.2 Coarse-Grained Representation and Macroscopic SANN

The first step in building the macroscopic SANN involves downscaling the spiking activity of the microscopic network. A factor ds has to be chosen beforehand. The original network consists of high-resolution grids of excitatory and inhibitory neurons, each representing detailed spiking activity at the individual neuron level. To transition to the macroscopic scale, the spiking activities of these neurons are aggregated using a process called block averaging. This approach divides the high-resolution grid into smaller, non-overlapping blocks and computes the average activity within each block. By replacing the detailed activity of individual neurons with these averaged values, the network is reduced to a coarser grid. For example, a dense grid of excitatory neurons is downscaled into a smaller grid, allowing the macroscopic network to simulate population-level dynamics instead of individual neuron behavior.

Once the spiking activity has been downscaled, the synaptic layers of the macroscopic SANN are adapted to operate at the reduced resolution. In the original microscopic network, synaptic connections are defined with fine-grained spatial precision, allowing detailed interactions between neurons. In the macroscopic network, these connections are modified to reflect the coarser grid. The Gaussian weight kernels used to define the connectivity patterns are resized to match the reduced resolution, ensuring that the spatial extent of connections remains proportional to the grid size. Additionally, scaling mechanisms are introduced to handle transitions between different resolutions, ensuring compatibility between excitatory and inhibitory neuron grids of varying sizes. These modifications allow the macroscopic synaptic layers to capture the essential dynamics of the original network while reducing computational complexity.

To maintain consistency between the microscopic and macroscopic networks, the leaky integrate-and-fire (LIF) dynamics are preserved in the macroscopic neurons. These dynamics govern the membrane potentials and spiking behavior of the neurons, ensuring that the fundamental characteristics of the original network are retained. Although the macroscopic neurons simulate population-level activity rather than individual spiking, their behavior reflects the same principles as their microscopic counterparts. The refractory mechanism and spike threshold are also retained, ensuring that the firing patterns remain biologically realistic.

The macroscopic SANN is implemented using a framework that integrates these downscaled components into a unified system. During simulation, the macroscopic network receives inputs derived from the downscaled spiking activities of the microscopic network. These inputs drive the dynamics of the excitatory and inhibitory neuron populations, which are updated based on the modified synaptic layers and LIF equations. The macroscopic network outputs coarse-grained excitatory and inhibitory spiking activities, which approximate the behavior of the original microscopic network at a larger scale.

To validate the accuracy of the macroscopic model, the output spiking activity of the microscopic SANN is downscaled by the same factor ds and compared with the output of the macroscopic model. The downscale function is discussed in section 2.3, and the evaluation result is shown in section 3.2 (cosine similarity).

2.3 Downscaling and Upscaling of Spiking Activity

The **downscale_activity** function reduces the resolution of a 2D tensor by a specified factor (ds) using block averaging. This approach divides the input tensor into non-overlapping blocks of size ds × ds and computes the average value within each block, thereby aggregating the local activity into a coarser grid. For instance, a tensor of size N×N is transformed into a smaller tensor of size (N//ds)×(N//ds). The function first ensures that the tensor can be evenly divided by the downscaling factor, throwing an assertion error if this condition is not met. It uses PyTorch's F.avg_pool2d method to perform the block averaging efficiently, applying it to a tensor reshaped to include batch and channel dimensions for compatibility. Once processed, the batch and channel dimensions are removed, returning the downscaled tensor. This function is essential for extracting macroscopic activity from a high-resolution representation, effectively capturing population-level behavior.

The **upscale_activity** function restores the resolution of a 2D tensor to its original size by nearest-neighbor interpolation. Given a downscaled tensor, this function replicates each value within the smaller grid to reconstruct a higher-resolution tensor, thereby expanding the spatial dimensions by a factor of ds. For example, an input tensor of size (N, N) is upscaled to size (N×ds, N×ds). The function uses PyTorch's F.interpolate method with scale_factor=ds and mode='nearest' to achieve this replication, preserving the relative simplicity of the input data. The tensor is processed similarly to downscale_activity, with additional dimensions for batch and channel added temporarily for compatibility. After interpolation, the batch and channel dimensions are removed, and the upscaled tensor is returned. This function is crucial for comparing macroscopic activity with its original microscopic representation and for restoring finer details in neural activity models.

2.4 U-Net

The exact U-Net architecture used in our implementation is a 4-level deep convolutional neural network designed to process 2D inputs as followed (Seo, Junhyeon & Kapania, Rakesh., 2023):

```
UNet2D(
  (enc1): DoubleConv(
    (conv): Sequential(
      (0): Conv2d(1, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU(inplace=True)
      (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (5): ReLU(inplace=True)
    )
  )
  (down1): DownBlock(
    (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (double_conv): DoubleConv(
      (conv): Sequential(
        (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU(inplace=True)
        (3): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (5): ReLU(inplace=True)
      )
    )
  )
  (down2): DownBlock(
    (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (double_conv): DoubleConv(
      (conv): Sequential(
        (0): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU(inplace=True)
        (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (5): ReLU(inplace=True)
      )
    )
  )
  (down3): DownBlock(
    (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (double_conv): DoubleConv(
      (conv): Sequential(
        (0): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU(inplace=True)
        (3): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (4): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (5): ReLU(inplace=True)
      )
    )
  )
  (up1): UpBlock(
    (up): ConvTranspose2d(512, 256, kernel_size=(2, 2), stride=(2, 2))
    (double_conv): DoubleConv(
      (conv): Sequential(
        (0): Conv2d(512, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU(inplace=True)
        (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (5): ReLU(inplace=True)
      )
    )
  )
  (up2): UpBlock(
    (up): ConvTranspose2d(256, 128, kernel_size=(2, 2), stride=(2, 2))
    (double_conv): DoubleConv(
      (conv): Sequential(
        (0): Conv2d(256, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU(inplace=True)
        (3): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (5): ReLU(inplace=True)
      )
    )
  )
  (up3): UpBlock(
    (up): ConvTranspose2d(128, 64, kernel_size=(2, 2), stride=(2, 2))
    (double_conv): DoubleConv(
      (conv): Sequential(
        (0): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU(inplace=True)
        (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
```

```
        (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (5): ReLU(inplace=True)
      )
    )
  )
  (final_conv): Conv2d(64, 1, kernel_size=(1, 1), stride=(1, 1))
)
```

This U-Net is trained with num_epochs=10, batch_size=4 and learning_rate of 1e-3. Final loss at epoch 10 manages to reduce until 21.402557.

This U-Net configuration is well-suited for inputs with spatial dimensions ranging from 100x100 to 256x256 pixels or larger, depending on available computational resources. It is employed to refine upscaled macroscopic neural activity by learning to map coarse representations to a fine-grained resolution, effectively restoring details that might be lost during downscaling. This design leverages the symmetric encoder-decoder structure and skip connections to ensure accurate and high-quality output reconstruction.

2.5 Evaluation metrics

Given two image tensors, min-max normalisation is first performed on the two tensors before computing their similarity metrics. We use cosine similarity to evaluate their similarity and effectiveness of multiscale approximation.

3. Results

We begin the development and validation of our multiscale modeling framework by setting some experiment configurations. We make use of random seed 42 to ensure liability and reconstructability of the experiment results. We use 30 ms of 5mV external current input followed by 200ms of no current input, with a time step of 0.5ms. We set We=0.23, Wi=0.06 to achieve a stable bump state from the SANN. Finally, we chose a downscaled factor of 16 (ds), so the original scale of microscopic SANN is 128*128, whereas the downscaled macroscopic SANN has a scale of 8*8.

3.1 Microscopic SANN Excitatory Potentials

Fig 3.1a shows the output of the microscopic Spiking Attractor Neural Network at time 230ms, where a stable bump state pattern is displayed. The elapsed time with GPU is 32s.

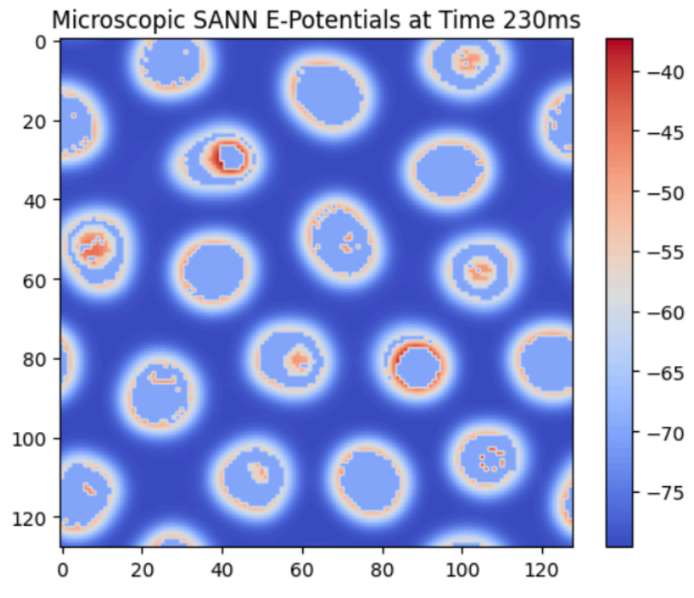


Fig 3.1a

Figure 3.1b shows the downscaled version of Fig 3.1a, with a factor of 16.

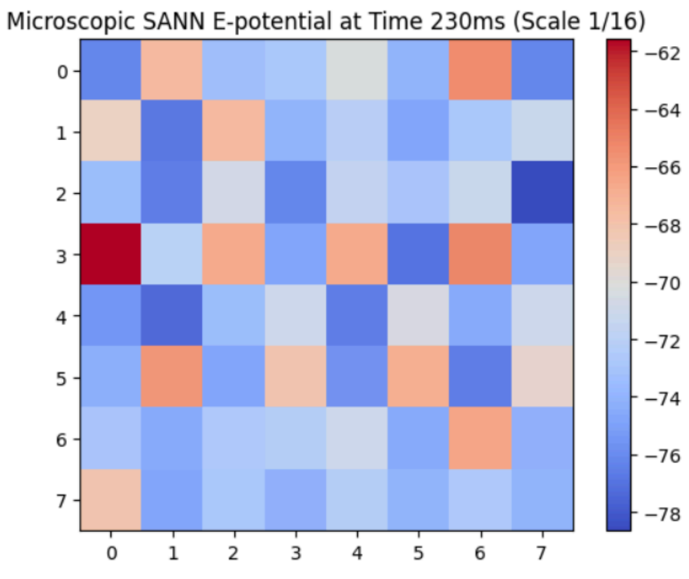


Fig 3.1b

3.2 Downscaled Microscopic SANN Excitatory Potentials vs Macroscopic SANN Excitatory Potentials

Fig 3.2a shows the output of the macroscopic Spiking Attractor Neural Network at time 230ms. The elapsed time with GPU is only 1s! One can also see some resemblance between Fig 3.1a and Fig 3.2a (slight difference in colour due to difference in min-max value)

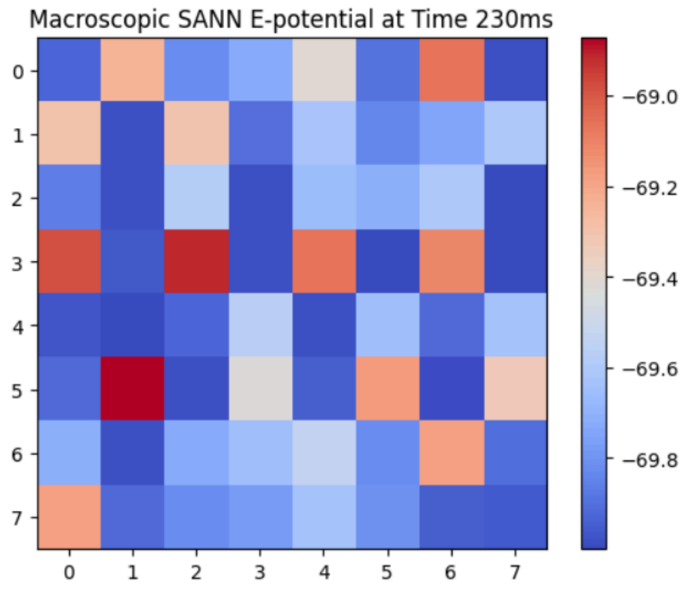
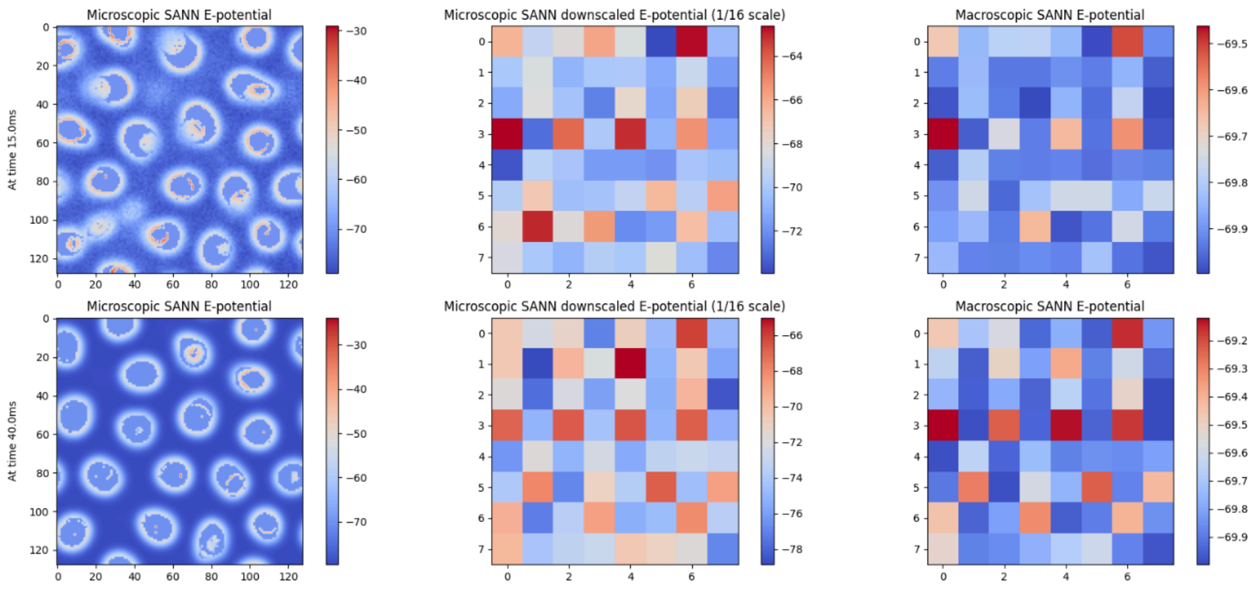


Fig 3.2a

Fig 3.2b shows comparison between microscopic SANN downscaled E-potential vs macroscopic SANN E-potentials at different times.



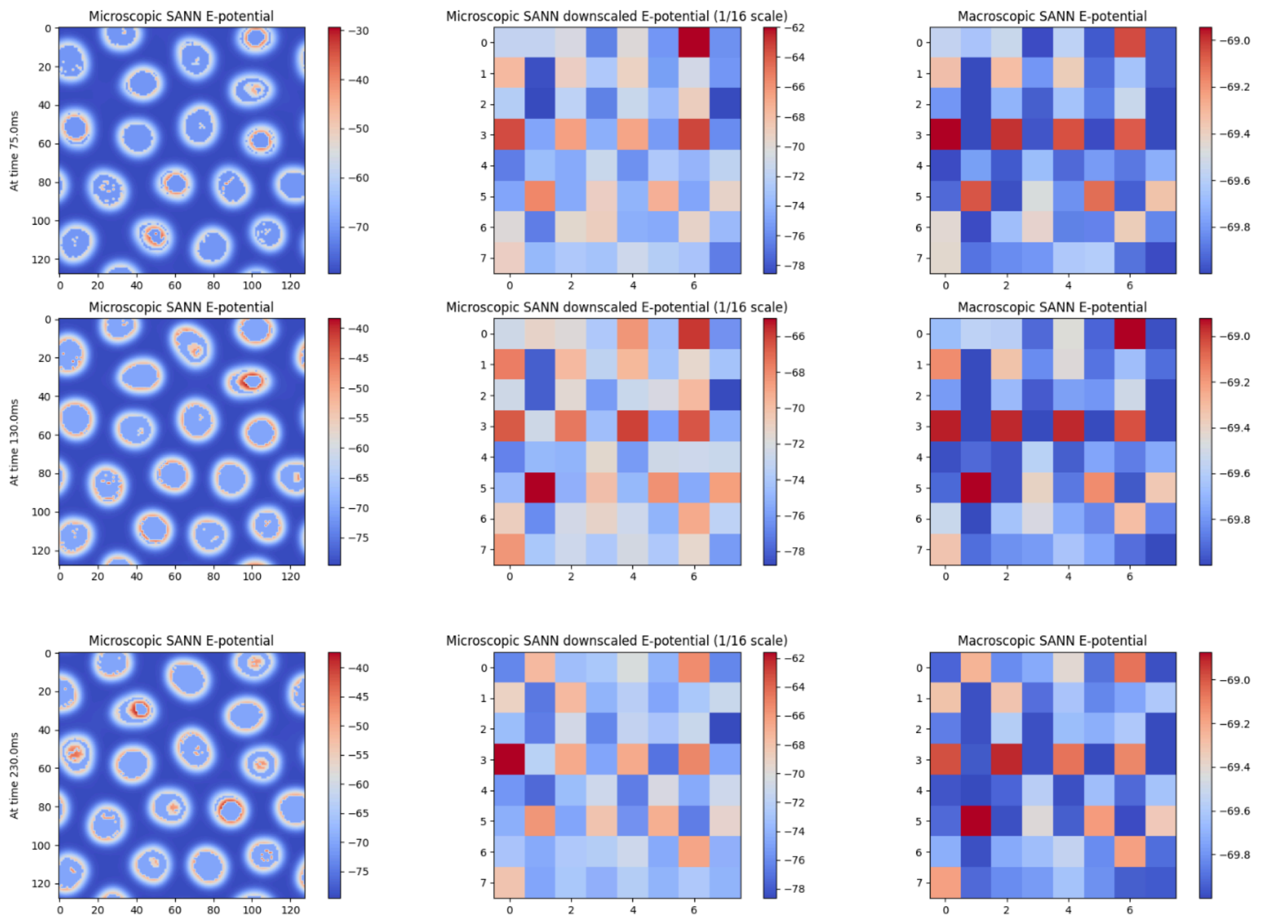


Fig 3.2b

To further prove with statistics that our macroscopic SANN model gives good approximations of the microscopic SANN, below is the calculated metrics at different times:

```
Cosine Similarity of downscaled Microscopic SANN and Macroscopic SANN E-Potentials:
Time 15.0ms
Cosine Similarity: 0.9400

Time 40.0ms
Cosine Similarity: 0.9490

Time 75.0ms
Cosine Similarity: 0.9522

Time 130.0ms
Cosine Similarity: 0.9298

Time 230.0ms
Cosine Similarity: 0.9424
```

3.3 U-Net upscaled Macroscopic SANN Excitatory Potentials vs Microscopic SANN Excitatory Potentials

To upscale the macroscopic SANN output, its excitatory potential (8*8) is first fed to the upscale_activity() function. Fig 3.3a shows an example of the resulting upscaled spiking activity (128*128).

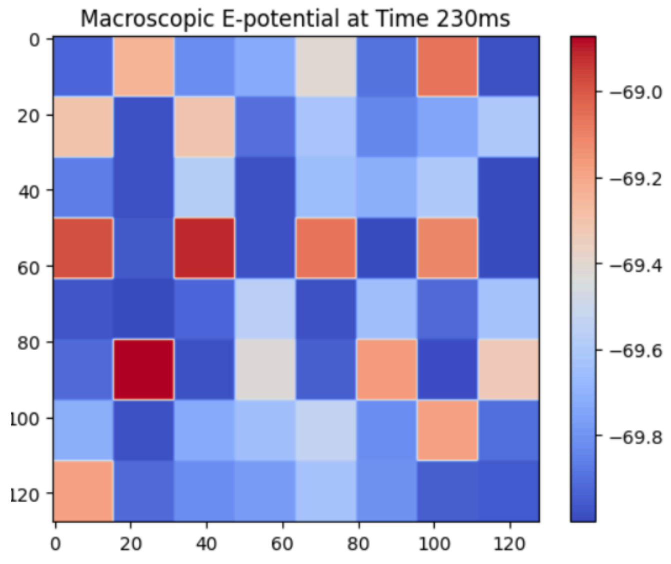
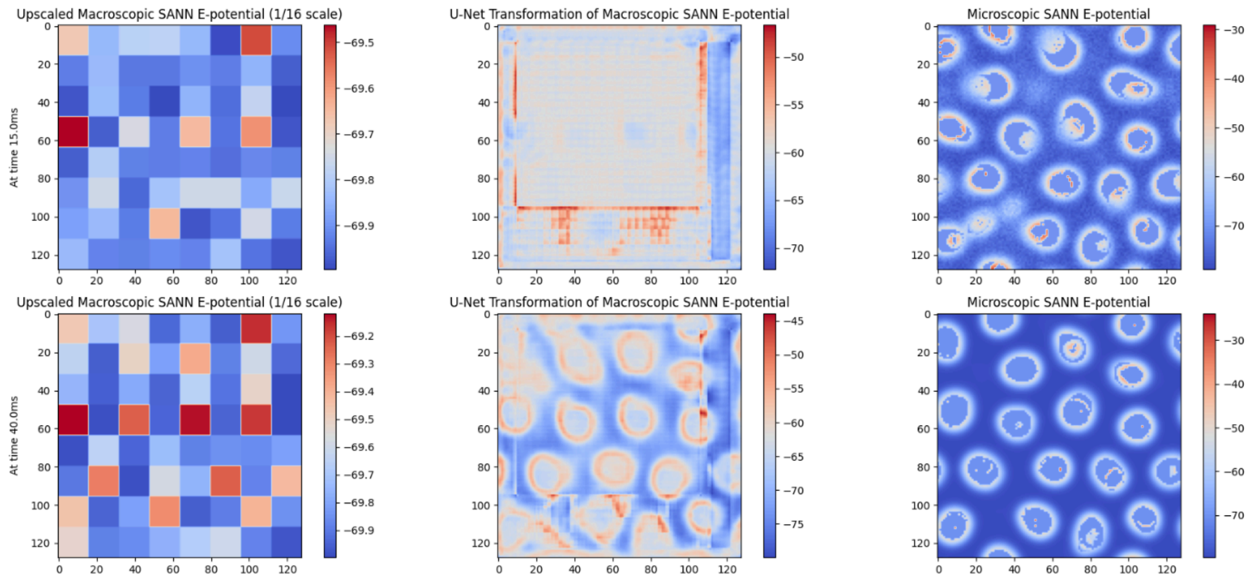


Fig 3.3a

This upscaled spiking activity is then fed to the U-Net to add resolution to the graph. Fig 3.3b shows the comparison between U-Net Transformed and Microscopic SANN E-potentials at different times.



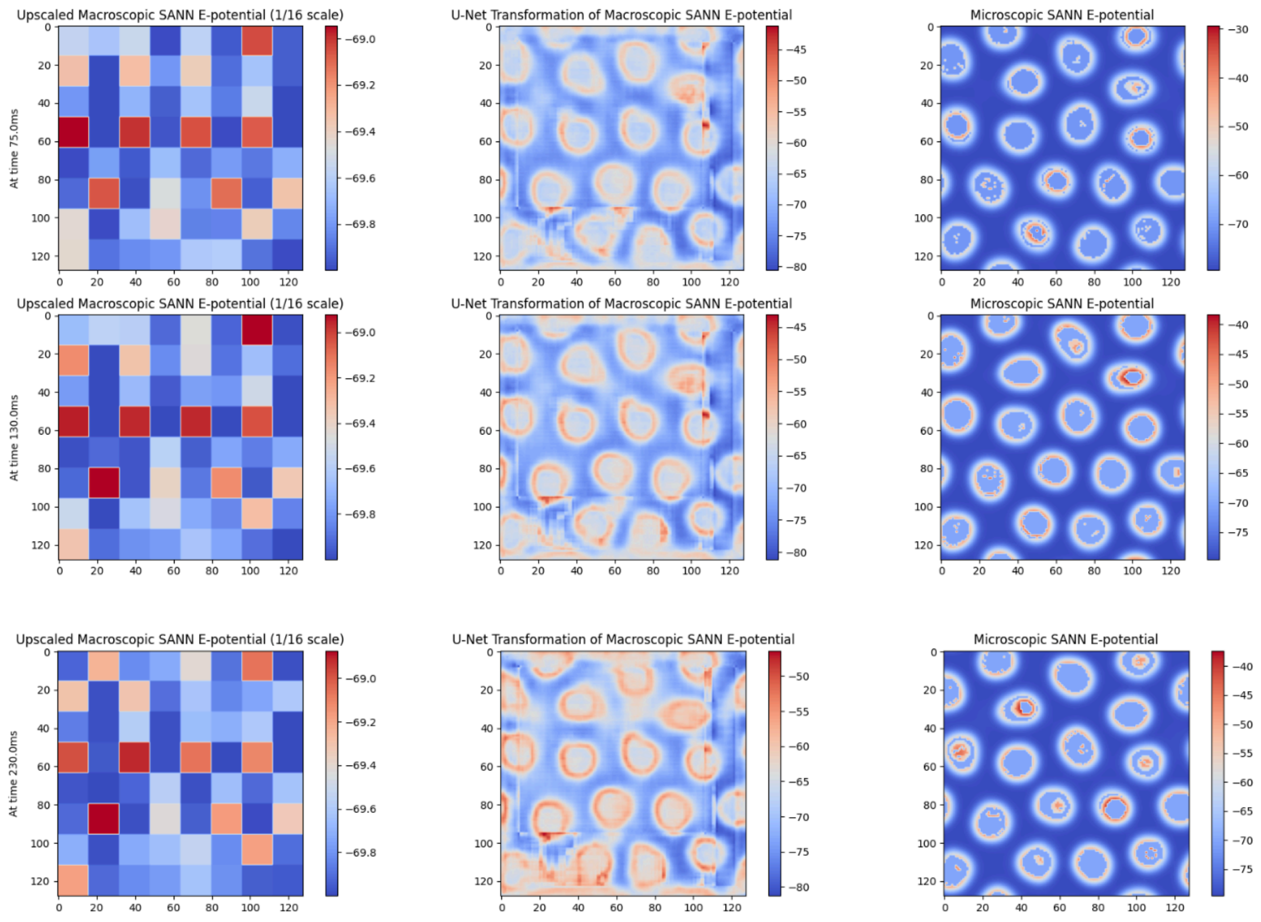


Fig 3.3b

It is visually evident that the upscaled macroscopic SANN effectively captures the dynamics and provides a strong representation of the microscopic SANN's neural activity as the network approaches a stable state. We then further prove our arguments with cosine similarity that shows improvement of macroscopic representations after transformation using U-Net:

```
Cosine Similarity of Microscopic SANN and upscaled Macroscopic SANN E-Potentials
Before -> After U-Net Transformation:
Time 15.0ms
Cosine Similarity: 0.7168 -> 0.8027

Time 40.0ms
Cosine Similarity: 0.6941 -> 0.7520

Time 75.0ms
Cosine Similarity: 0.6929 -> 0.7601

Time 130.0ms
Cosine Similarity: 0.6747 -> 0.7933

Time 230.0ms
Cosine Similarity: 0.6985 -> 0.7925
```

4. Discussion

Our results demonstrate the feasibility and effectiveness of the proposed multiscale modeling framework in simulating large-scale spiking attractor neural networks (SANNs) while maintaining computational efficiency and fidelity. The framework's ability to transition between microscopic and macroscopic scales using downscaling, upscaling, and a trained U-Net model is validated through qualitative visual comparisons and quantitative similarity metrics, such as cosine similarity. These metrics consistently show that the macroscopic SANN model closely approximates the dynamics of the microscopic SANN at various time points, with high similarity scores (>0.94) achieved for downscaled comparisons and significant improvements (>0.75) after U-Net transformation.

4.1 Computational Efficiency

One of the most notable achievements of this framework is the drastic reduction in computational time. The microscopic SANN model required 32 seconds to simulate 230 ms of neural activity, whereas the macroscopic SANN completed the same simulation in just 1 second, reflecting a substantial improvement in efficiency. This efficiency gain makes it possible to simulate larger networks or longer time scales, expanding the applicability of spiking attractor models to broader neuroscientific questions and practical applications.

4.2 Fidelity of Multiscale Representations

The fidelity of the macroscopic SANN model is evident from both the visual and statistical comparisons. The high cosine similarity values between the microscopic and macroscopic SANN excitatory potentials confirm that the essential dynamics of the network are preserved even after downscaling. Furthermore, the integration of the U-Net model to upscale macroscopic representations significantly enhances the resolution and quality of the reconstructed neural activity. The improvement in cosine similarity before and after U-Net transformation (e.g., from 0.6985 to 0.7925 at 230 ms) underscores the capability of this approach to restore finer details that might be lost during downscaling.

4.3 Biological Relevance

By preserving the dynamics of leaky integrate-and-fire neurons and incorporating biologically inspired mechanisms such as Gaussian connectivity and synaptic decay, the framework maintains its relevance for modeling realistic neural activity. The ability to simulate attractor dynamics at both microscopic and macroscopic levels without compromising accuracy provides a robust tool for investigating neural processes such as memory, decision-making, and pattern recognition.

4.4 Limitations and Future Directions

Despite its strengths, the proposed framework has some limitations. First, the current implementation focuses solely on excitatory potentials, which may limit its generalizability to networks with more complex inhibitory interactions or diverse neural types. Second, while the U-Net model effectively enhances resolution, its reliance on pretraining introduces potential biases that may affect its performance in novel scenarios. Finally, the framework's dependence on GPU resources, though mitigated at the macroscopic scale, still poses a barrier for researchers without access to high-performance computing infrastructure.

Future work could address these limitations by extending the framework to incorporate inhibitory dynamics more comprehensively, exploring alternative upscaling models beyond U-Net, and optimizing the framework for deployment on less powerful hardware. Additionally, integrating mean-field theory into the modeling pipeline could provide an analytical foundation for understanding the emergent dynamics of large-scale networks.

5. Conclusion

This study introduces a novel multiscale modeling framework for spiking attractor neural networks that bridges microscopic and macroscopic scales while ensuring computational efficiency and accuracy. By employing a combination of downscaling, upscaling, and U-Net transformation, the framework enables detailed neural dynamics to be simulated at reduced computational costs. The high cosine similarity scores and significant improvements after U-Net transformation validate the fidelity of the framework, demonstrating its potential for advancing our understanding of large-scale neural systems.

The results highlight the promise of multiscale approaches in computational neuroscience, offering a scalable and efficient tool for modeling complex neural dynamics. By providing a means to simulate large-scale networks with realistic attractor dynamics, this framework paves the way for new insights into the mechanisms underlying brain function and offers a foundation for future advancements in neural modeling.

Code Availability

The code conducting the experiment in this study will be made available at https://github.com/cybercrazetech/ComputationalNeuroscience/blob/main/Multiscale%20Modeling%20of%20Spiking%20Attractor%20Neural%20Network%20with%20Coarse%20Graining%20and%20U-Net%20Scaling.ipynb

Acknowledgment

LLMs are used for language polishing in our paper, but we did not rely on LLMs to generate the core research content. Additionally, careful review has been done on the polished content.

References

Rolls ET. Attractor networks. Wiley Interdiscip Rev Cogn Sci. 2010 Jan;1(1):119-134. doi: 10.1002/wcs.1. Epub 2009 Dec 17. PMID: 26272845.

Eliasmith C. A unified approach to building and controlling spiking attractor networks. Neural Comput. 2005 Jun;17(6):1276-314. doi: 10.1162/0899766053630332. PMID: 15901399.

Burkitt, Anthony. (2006). A Review of the Integrate-and-fire Neuron Model: I. Homogeneous Synaptic Input. Biological cybernetics. 95. 1-19. 10.1007/s00422-006-0068-6.

La Camera G. The Mean Field Approach for Populations of Spiking Neurons. Adv Exp Med Biol. 2022;1359:125-157. doi: 10.1007/978-3-030-89439-9_6. PMID: 35471538; PMCID: PMC10317473.

Seo, Junhyeon & Kapania, Rakesh. (2023). Topology optimization with advanced CNN using mapped physics-based data. Structural and Multidisciplinary Optimization. 66. 10.1007/s00158-022-03461-0.

Lu W, Rossoni E, Feng J. On a Gaussian neuronal field model. Neuroimage. 2010 Sep;52(3):913-33. doi: 10.1016/j.neuroimage.2010.02.075. Epub 2010 Mar 10. PMID: 20226254.