Efficient parameter calibration and real-time simulation of large scale spiking neural networks with GeNN and NEST

Felix Johannes Schmitt[1], Vahid Rostami[1], and Martin Paul Nawrot[1,*]

[1] *Computational Systems Neuroscience, Institute of Zoology, University of Cologne, Cologne , Germany*
[*] *Corresponding author: Martin Paul Nawrot, martin.nawrot@uni-koeln.de*

Abstract

Spiking neural networks (SNN) represent the state-of-the-art approach to the biologically realistic modeling of nervous system function. The systematic calibration for multiple free model parameters to achieve robust network function demands high computing power and large memory resources. Special requirements arise from closed-loop model simulation in virtual environments, and from real-time simulation in robotic application. Here, we compare two complementary approaches to efficient large scale and real-time SNN simulation. The widely used NEural Simulation Technology (NEST) parallelizes simulation across multiple CPU cores. The GPU-enhanced Neural Network (GeNN) simulator uses the highly-parallel GPU-based architecture to gain simulation speed. We quantify fixed and variable simulation costs on different hardware configurations. As benchmark model we us a spiking cortical attractor network with a topology of densely connected excitatory and inhibitory neuron clusters. We show that simulation time scales linear with the simulated biological time and, for large networks, approximately linear with the model size as dominated by the number of synaptic connections. Additional fixed costs with GeNN for model compilation are independent of model size while fixed costs with NEST for model building increase linear with model size. We demonstrate how GeNN can be used for simulating networks with up to $3,5$ million neurons ($> 3 \times 10^{12}$ synapses) on a high-end GPU, and up to 250 thousand neurons (25 billion synapses) on a low-cost GPU. Real-time simulation was achieved for networks of more than 90 k neurons. We conclude that GeNN is in advantage for large networks and real-time applications while NEST plays out its strengths in a high degree of flexibility ideal for prototyping, and easy access for non-expert programmers.

Keywords:

spiking neural network, neocortex, computational neuroscience, attractor neural network, metastability, real-time simulation, computational neuroethology, artificial agents

Introduction

Information processing in animal nervous systems is highly efficient and robust. The vast majority of nerve cells in invertebrates and vertebrates are action potential generating (aka spiking) neurons. It is thus widely accepted that neural computation with action potentials in recurrent networks forms the basis for sensory processing, sensory-to-motor transformations and higher brain function (Abeles, 1991; Singer and Gray, 1995). The availability of increasingly detailed anatomical, morphological and physiological data allows for well-defined functional SNNs of increasing complexity that are able to generate testable experimental predictions at physiological and/or behavioral levels. SNNs have thus become a frequent tool in basic (Brunel, 2000; Van Vreeswijk and Sompolinsky, 1996), translational (McIntyre and Hahn, 2010; Eliasmith et al., 2012), and clinical (Kasabov and Capecci, 2015; Hammond et al., 2007) neuroscience research. At a more abstract modeling level, brain-inspired SNNs have the potential to shape future solutions for intelligent systems (Neftci et al., 2013; Chicca et al., 2014). This specifically includes spike-based approaches to machine learning (Indiveri et al., 2010; Schmuker et al., 2014; Zenke and Ganguli, 2018; Gütig and Sompolinsky, 2006; Gütig, 2016; Pfeiffer and Pfeil, 2018; Tavanaei et al., 2019), to reservoir computing (Büsing et al., 2010; Tanaka et al., 2019), and brain-inspired control architectures for artificial agents and robots (Rapp and Nawrot, 2020; Rapp et al., 2020; Bartolozzi et al., 2022). Computation with attractor networks has been hypothesized as one hallmark of brain-inspired computation (Hopfield, 1982; Amit and Brunel, 1997) and, with increasing evidence, has been implicated in decision making (Finkelstein et al., 2021), working memory (Sakai and Miyashita, 1991; Inagaki et al., 2019), and sensory-motor transformation (Mazzucato et al., 2019; Wyrick and Mazzucato, 2021; Rostami et al., 2022; Mazzucato, 2022).

SNNs have two basic elements: mathematical models of neurons and synaptic connections between neurons. Elaborate models can involve complex models of neurons and synapses as well as a variety of additional elements, which may considerably increase model complexity. Network size defined by the number of neurons N and network connectivity quantified by the number of synapses M are thus major determinants of the computational demand, while the available hardware memory is a major limiting factor on network size and connectivity. We here consider large-scale SNNs in the approximate size range from one thousand to a few million neurons. This range covers the complete nervous system of many invertebrate and of small vertebrate species as shown in Fig. 1 (Table 1), including for instance the adult fruit fly *Drosophila melanogaster* with ≈ 100 k neurons in the central brain (Raji and Potter, 2021), the European honeybee *Apis melifera* (≈ 900k) (Witthöft, 1967; Godfrey et al., 2021; Menzel, 2012) and the zebrafish (≈ 10 million). In mammals it covers a range of subsystems from a single cortical column in the range of approximately $30-80$ k neurons (Markram et al. (2015); Boucsein et al. (2011); Potjans and Diesmann (2012) to the complete neocortex of the mouse (≈ 5 million neurons, Herculano-Houzel et al. (2013)); cf.). Models exceeding this scale are currently still an exception (Eliasmith and Trujillo, 2014; Van Albada et al., 2015; Igarashi et al., 2019; Yamaura et al., 2020) and typically require the use of a supercomputer.

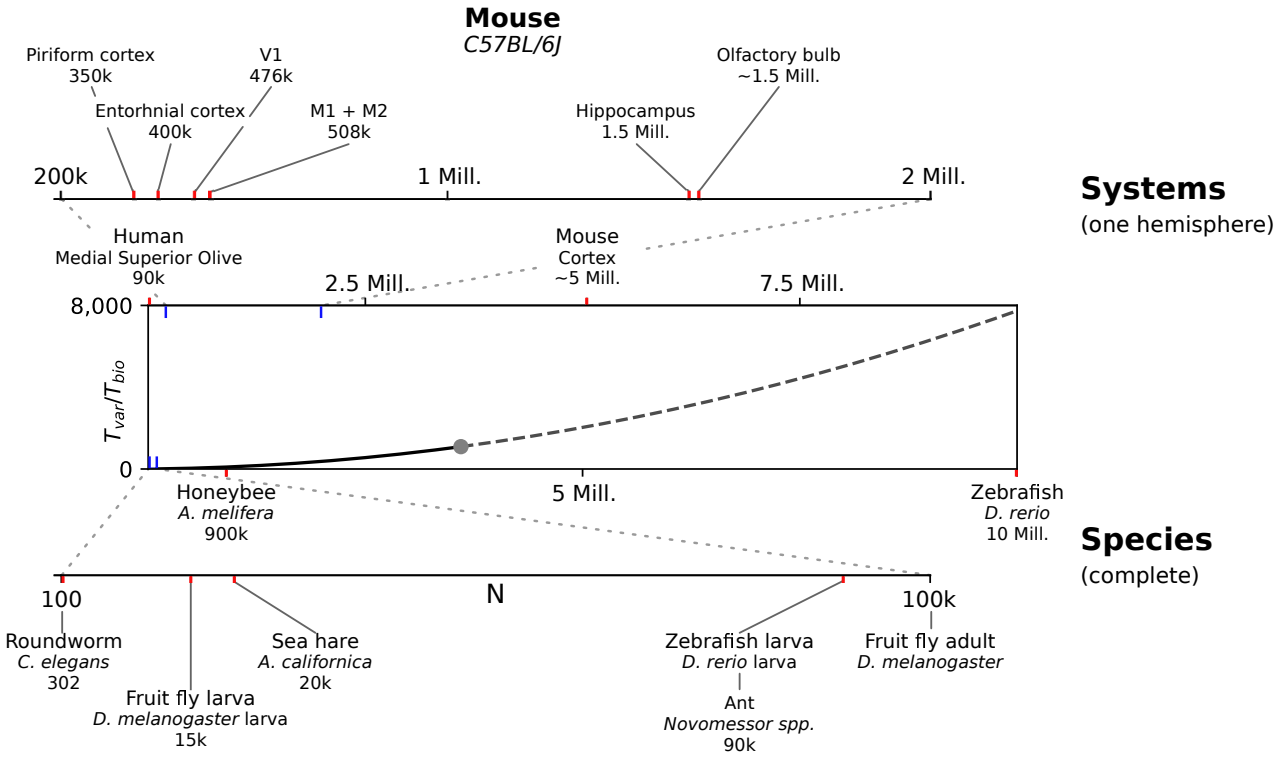

Figure 1: **Full system modeling with spiking neurons.** Simulation time relative to simulated biological time over a realistic range of network sizes for full spiking simulation. Network size is projected to the total number of neurons per hemisphere for subsystems in the mouse (top), and in the CNS of invertebrate and vertebrate model species (bottom), see also Table 1. The graph expresses the quadratic dependence of eqn. 1 fitted for the procedural simulation method with GeNN. Full line shows the range of network sizes covered in our simulations using a high-end GPU (S3, Table 2, dashed line extrapolates to a maximum size of 10 million neurons covering the complete CNS of the zebrafish.

The conventional tool for the simulation of SNNs are CPU-based simulation environments. Several well-adopted simulators are in community use (Brette et al., 2007; Tikidji-Hamburyan et al., 2017), each of which has typically been optimized for specific purposes such as the simulation of complex neuron models with extended geometry and detailed biophysics (Hines and Carnevale, 2001), the convenient implementation of neuron dynamics by means of coupled differential equations ((Stimberg et al., 2019), or the implementation of the Neural engineering framework (Eliasmith and Anderson, 2003; Bekolay et al., 2014). The NEural Simulation Technology (NEST, `https://www.nest-simulator.org/`, Gewaltig and Diesmann (2007)) that we consider here was designed for the parallelized simulation of large and densely connected recurrent networks of point neurons. It has been under continuous development since its invention under the name of SYNOD (Diesmann et al., 1995, 1999; Rotter and Diesmann, 1999; Morrison et al., 2005, 2007; Schmidt et al., 2018) and enjoys a stable developer and a large user community. More recently, new initiatives have

Table 1: Number of neurons of relevant model organisms and subsystems.

	Neurons	**Citation**
Invertebrate		
Round worm (*Caenorhabditis elegans*)	302	White et al. (1986)
Fruit fly larva (*Drosophila melanogaster*)	15k	Eschbach and Zlatic (2020)
California sea hare (*Aplysia californica*)	20k	Zhao and Wang (2009)
Fruit fly adult (*Drosophila melanogaster*)	70k∼200k	Godfrey et al. (2021) Raji and Potter (2021)
Ant (*Novomessor spp.*)	90k	Godfrey et al. (2021)
European honeybee, worker (*Apis mellifera*)	600∼900k	Witthöft (1967) (Godfrey et al., 2021)
Vertebrate		
Human: Medial Superior olive	15.5k	Kulesza (2007)
Zebrafish larva (*Danio rerio*)	90k	Bruzzone et al. (2021)
Sprague Dawley rat: Basal ganglia	2,900k	Oorschot (1996)
Zebrafish adult (*Danio rerio*)	10,000k	Hinsch and Zupanc (2007)
Smoky shrew (*Sorex fumeus*)	39,490k	Sarko et al. (2009)
Mouse (C57BL/6J) per hemisphere		
Piriform cortex	350k	Herculano-Houzel et al. (2013)
Entorhinal cortex	400k	Herculano-Houzel et al. (2013)
Visual cortex V1	476k	Herculano-Houzel et al. (2013)
Motor cortex M1+M2	508k	Herculano-Houzel et al. (2013)
Hippocampus	1,500k	Herculano-Houzel et al. (2013)
Olfactory bulb	1,520k	Parrish-Aungst et al. (2007)
Cortex	5,049k	Herculano-Houzel et al. (2013)

formed to harness GPU-based simulation speed for the modeling of SNNs (Nageswaran et al., 2009; Fidjeland et al., 2009; Mutch et al., 2010; Brette and Goodman, 2012; Florimbi et al., 2021; Ben-Shalom et al., 2022). The GPU-enhanced Neural Network (GeNN) simulation enviroment (`https://genn-team.github.io/genn/`) developed by Thomas Nowotny and colleagues (Yavuz et al., 2016; Knight et al., 2021; Knight and Nowotny, 2021) is a code generation framework (Blundell et al., 2018) for SNNs targeted towards computational neuroscience research as well as machine learning (Knight and Nowotny, 2022). Neuromorphic hardware provides an alternative substrate for the simulation of SNNs and is not considered here.

Results

Spiking neural attractor network as benchmark model.

We performed simulations of the spiking cortical attractor network model established by (Rostami et al., 2022). This network inherits the overall network structure of the random balanced network (Van Vreeswijk and Sompolinsky, 1996; Brunel, 2000) with random recurrent connections among excitatory and inhibitory neurons. Introducing a topology of strongly interconnected pairs of excitatory and inhibitory neuron populations (clusters, Fig. 2 A) in terms of their synaptic weights to the connectivity matrix (see Material and Methods) results in a complex pattern of spontaneous network activity where each cluster can switch between a state of low (baseline) activity and states of increased activity. The resulting network dynamics is visible in the network simulation of Fig. 2 B. This network behavior marks the desired feature of metastability (Mazzucato et al., 2019; Rost et al., 2018) where the network as a whole cycles through different attractors (or network-wide states) that are defined by the possible cluster activation patterns.

The random connectivity scheme between neurons implies that the number of synapses scales quadratic with the number of neurons as $M = pN^2$. The clustered network topology in our benchmark model implies a dense connectivity scheme within the pair of clusters and a lower connectivity in terms of their relative weight between these clusters (Fig. 2 A). For the chosen network parameters this results in an overall connectivity parameter of $p \approx 0.3$ (see Materials and Methods). This compartmentalized architecture suits well for our bench-marking purpose because it is reminiscent for whole-system or multi-area modeling in large scale models that involve several different neuropiles or brain areas (Rapp and Nawrot, 2020; Schmidt et al., 2018). For bench-marking purposes, the combined excitatory-inhibitory clusters may thus be considered as different system parts. We kept the number of clusters fixed to $N_Q = 20$.

Benchmark approach and quantification of simulation costs.

For benchmarking performance we quantify the total costs of simulation in terms of total time required to complete a simulation and to obtain the simulated data. We distinguish fixed costs and variable costs of a simulation:

$$T_{sim} = T_{fixed} + T_{var}.$$

The fixed costs involve all steps necessary to set up and prepare the model before the first simulation step. The variable costs involve the actual propagation of the network state during each time step and the overall time used for data collection during a simulation (cf. Materials and Methods).

We use two different server configurations, S2 (20 cores, 2.5 GHz; 192 GB RAM) and S3 (24 cores, 3.0 GHz; 128 GB RAM) for the simulations with NEST (cf. Table 3). For simulations with GeNN we use one configuration with a low-cost GPU (NVIDIA GeForce

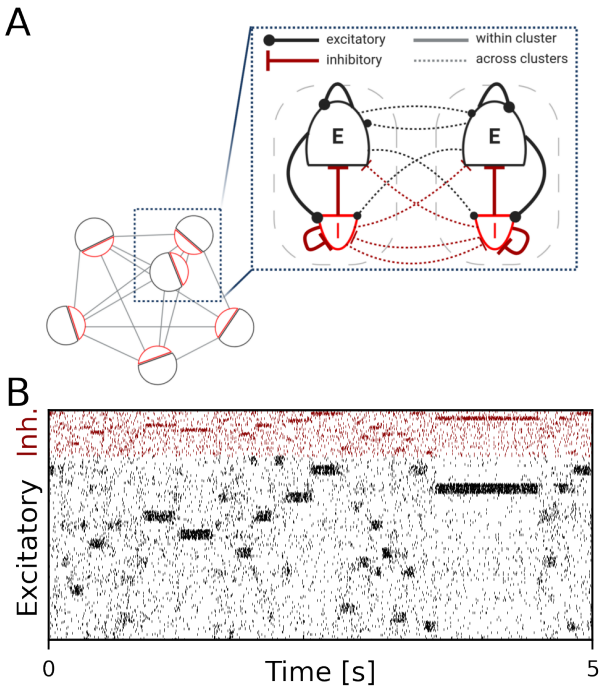

Figure 2: **Metastable activity in spiking cortical attractor network model with excitatory-inhibitory cluster topology.** (A) The network is tiled into strongly interconnected neuron populations (clusters). Each excitatory cluster E (black) is strongly and reciprocally connected (strong lines) to its corresponding inhibitory cluster I (red), forming an E/I cluster. Each neuron forms weak connections (dashed lines) to the excitatory and inhibitory neurons outside of its cluster. E neurons make up for 80% and I neurons for 20% of the total neuron number N. The model uses exponential leaky I&F neurons and all neurons receive weak constant current input. (B) Raster display of excitatory (black) and inhibitory (red) spiking activity in a network of N=$25,000$ neurons during 5 s of spontaneous activity shows the desired metastable behavior where different individual clusters can spontaneously assume states of high and low activity. Shown is a portion of 8% of neurons from each of the 20 clusters.

GTX 970, S1), and a second configuration with a high-end state-of-the-art GPU (NVIDIA Quadro RTX A6000, S3).

With GeNN we test two different approaches to store the connectivity matrices of the model Knight and Nowotny (2021). The sparse matrix format (Sp) stores all matrices in a sparse representation. They are initialized when loading the model. The procedural connectivity (Pr) stores the random seeds used for constructing the connectivity matrices and regenerates those on demand, i.e. after a spike occurred. We use the Python interfaces PyNEST (Eppler et al., 2009) and PyGeNN (Knight et al., 2021) to interact with the simulators.

Fixed costs for GeNN are high but independent of network size.

We find that for NEST the overall fixed costs scale approximately linear with the network connectivity as expressed in the total number of connections $M \propto N^2$ (Fig. 3) while the fixed costs stay approximately constant for GeNN and essentially over the complete range of tested network sizes. The fixed costs add up three contributions as shown in Fig. 3 A.

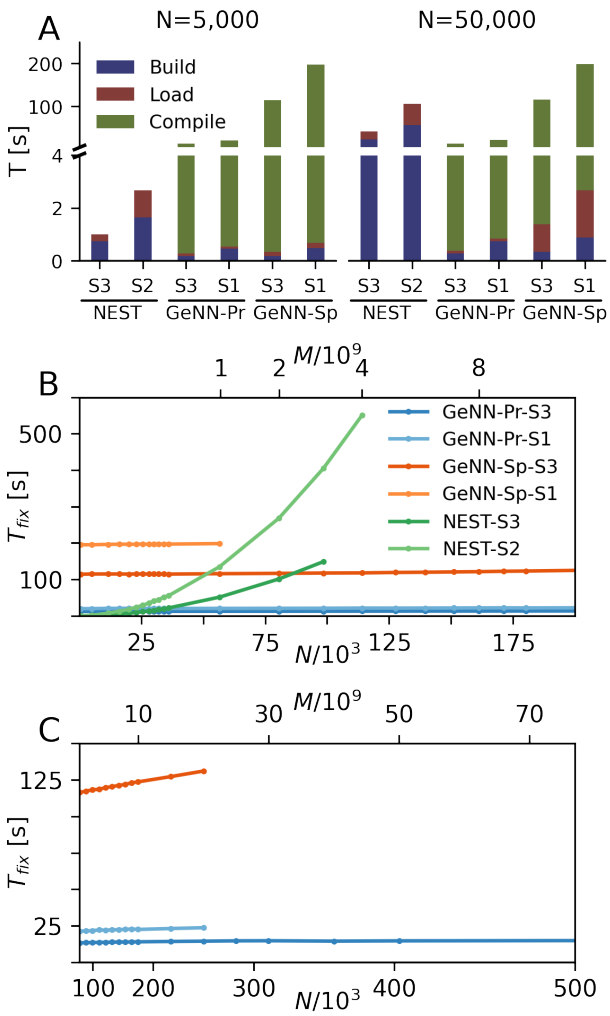

Figure 3: **Fixed costs of simulation.** (A) Individual costs for building the model, for laoding the model into RAM, and for compiling the model (GeNN only) for two network sizes of $N = 5,000$ and $N = 50,000$ neurons. The GPU-based simulation requires expensive model compilation. NEST uses pre-compiled neuron models. Note that the y-axes has different scales for low (≤ 4) and high (≥ 4) values of fixed costs. (B) Total fixed costs increase rapidly with network size in NEST. (B,C) Total fixed costs are approximately constant across different network sizes for the GPU-based simulation with the procedural connectivity, while they have a small slope for the sparse method.

These are building, loading, and compilation of the model (see Materials and Methods for details). For NEST, the cost for model compilation is essentially zero because it uses pre-compiled neuron and synapse models (Diesmann and Gewaltig, 2002) in combination with exact integration (Rotter and Diesmann, 1999). Compilation in GeNN, however, requires considerable time and accounts for most of the fixed costs. Thus, for a small network size of $N = 5,000$ neurons overall fixed costs amount to ≈ 1 min and ≈ 3 min for the Procedural and Sparse approach, respectively, compared to only 3 s with NEST. This picture changes for a ten times larger network with $N = 50,000$ neurons. Now the time for building the model with NEST is more costly than compilation with GeNN (procedural method). The fixed costs for NEST increase quadratic in N (linear in M) on both hardware configurations (Fig. 3 B) and eventually exceed fixed costs with GeNN.

The maximal network size that we were able to simulate (cf. Materials and Methods) is indicated by the end points in the benchmark graphs in Fig. 3 B and C. It is bound by the available memory, which is required for storing the network model and the data recorded during simulation. In the case of simulation with NEST, this bound is determined by the RAM configuration (Table 2). The larger RAM size of 192 GB on S2 allowed for a maximum network connectivity of $M = 4$ billion synapses and $N \approx 114$ k neurons, while on the faster server configuration S3 with only 128 GB RAM the limit was reached earlier (Fig. 3 B). With GeNN the limiting factor for the network size and connectivity is the hardware memory on the GPU itself. The Procedural approach allows for a more efficient usage of the GPU memory at the expense of simulation speed and allowed for a network size of > 3.5 million neurons and $> 3,000$ billion synapses on the high-end GPU (S3) and a respectable size of $N \approx 250$ k neurons ($M \approx 20$ billion synapses) on the low-cost GPU (S1).

Variable costs scale linear with biological time and approximate linear with network connectivity.

We first quantify simulation time T_{var} in dependence on the simulated biological time T_{biol} for a fix network size of $N = 50,000$ (Fig. 4 A). As to be expected, simulation time grows approximately linear with the number of simulated samples S and thus, for a pre-defined simulation time constant $1/\Delta t$, is proportional to the simulated time with

$$T_{var} \propto \frac{1}{\Delta t} \cdot T_{biol}$$

for all tested hardware platforms. We see considerably faster simulations with the sparse approach compared to the procedural connectivity in our network with a total connectivity of $p \approx 0.308$. Interestingly, a previous publication by Knight and Nowotny (2021) considering the balanced random network with a lower connectivity density of $p = 0.1$ found that procedural connectivity performs equally fast or even faster, which could be due to their lower connectivity density.

Next, we analyzed the relation between simulation time and network size. To this end we determined the proportionality factor T_{var}/T_{biol} as a function of network size. As shown in Fig. 4 B, the simulation time for NEST shows an approximate linear dependence on the

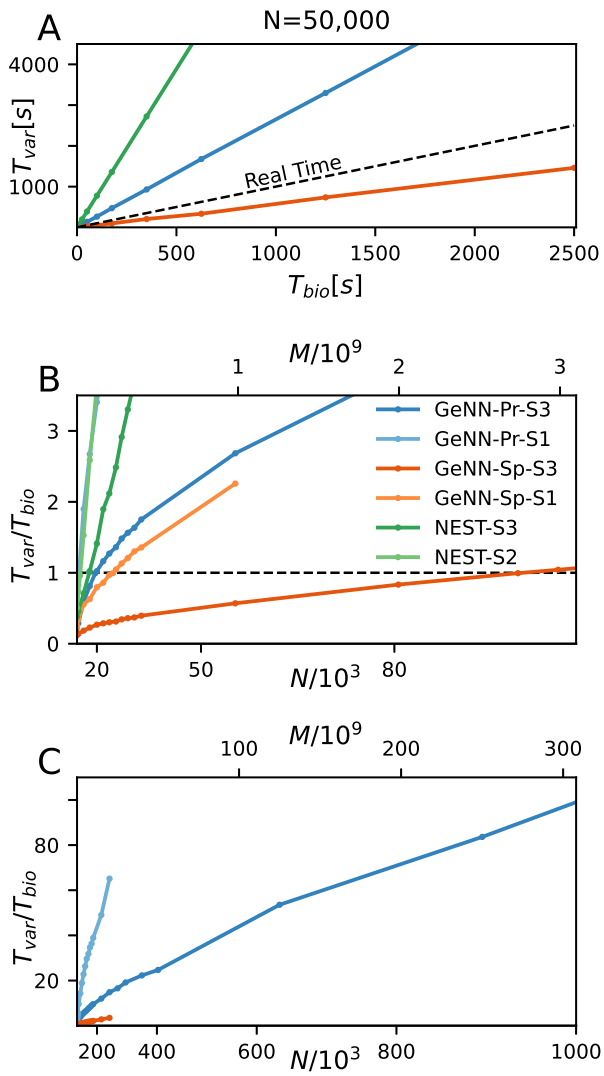

Figure 4: **Variable costs of simulation and real-time limitation.** (A) Simulation time T_{var} scales linearly with simulated biological time T_{biol}. The variable cost factor directly scales with the number of simulation steps, determined by $1/\Delta t$ and T_{biol}. (B) Real-time capability requires a variable cost factor that remains below $RT = T_{dyn}/T_{bio} = 1$ (dashed line). On the high-end GPU, simulation faster than real-time can be achieved for a network size of $N \approx 94k$ neurons and $M \approx 2.7e9$ synapses. With the best performing NEST configuration a network of $N \approx 16k$ neurons and $M \approx 0.1e9$ synapses can be simulated in real-time. (C) Simulation time scales almost linear with the total number of synapses M (and thus with N^2) for large network sizes. All simulations have been conducted with a time resolution of $\Delta t = 0.1$ ms in biological time.

total number of synapses M for large network sizes and is thus proportional to the squared

number of neurons $T_{var} \propto pN^2 \cdot T_{biol}$ with linear scale factor p, denoting the network-specifc connectivity.

For GeNN the graph shows a convex relation between T_{var}/T_{biol} and lower range network sizes that can be approximated as (cf. Table 2)

$$T_{var} \approx (\alpha_1 \cdot N + \alpha_2 \cdot pN^2 + \alpha_0) \cdot \frac{1}{\Delta t} \cdot T_{biol} \tag{1}$$

determined by the model-specific factors of neuron number, synaptic connectivity and the desired temporal resolution of model simulation. For increasing network size N^2 dominates and for large networks and for $N \gtrsim 100$ k the relation may be considered approximate linear in N^2. Additional model-related factors may contribute, in particular the average neural spiking activity and resulting spike traffic (see Discussion).

Real-time simulation is defined as $T_{sim} = T_{biol}$. For all practical purposes we may require that the simulation runs equally fast or faster than real time. For a given hardware configuration and the pre-defined time resolution Δt we may approximate the corresponding maximum network size as

$$N_{RT} \lesssim \sqrt{\frac{\Delta t}{p \cdot \alpha}}. \tag{2}$$

For our attractor network model (Table 2) we determined a maximum network size of $N_{RT} = 94,300$ using GeNN on the high-end GPU. We verified this result by simulating this network size as shown in Fig. 4 B.

Table 2: Estimated coefficients in eqn. 1 for the different hardware configurations. Maximum network size N_{RT} within real-time limit was determined in steps of 100 neurons. Simulations were 10 seconds long with a $\Delta t = 0.1ms$. The parameter $\boldsymbol{\alpha_2}$ measures the influence of the number of synapses, $\boldsymbol{\alpha_1}$ the influence of the number of neurons and $\boldsymbol{\alpha_0}$ is independent on the size.

Server	**Simulator**	$\alpha_2/10^{-12}$	$\alpha_1/10^{-6}$	$\alpha_0/10^{-3}$	N_{RT}	$M/10^6$
S1	GeNN (Sparse)	748	22.9	225	26,900	223
S3	GeNN (Sparse)	77	7.1	112	94,300	2,739
S1	GeNN (Proc)	1,324	145.1	170	6,400	13
S3	GeNN (Proc)	239	38.9	109	19,200	114
S2	NEST	41,108	-76.3	623	7,900	19
S3	NEST	11,821	-21.7	421	15,700	76

Efficient approach to parameter grid search.

Spiking neural networks have a large set of parameters. These include parameters of the individual neuron models, of synaptic elements and connectivity parameters. In our bench-

mark spiking attractor model we have 21 independent parameters (cf. Tables 4 and 5). Thus, both in scientific projects and for the development of real-world applications of SNNs, model tuning through parameter search creates the highest demand in computing time. Currently available methods including the standard grid search, Bayesian optimization (Parsa et al., 2019), and machine learning approaches (Carlson et al., 2014) require extensive sampling of the parameter space. Generally, the total costs for a parameter optimization directly scale with the number of samples tested for the considered parameter combinations. We here propose a batch-mode approach for efficient parameter search with GeNN (see Material and Methods) that economizes on reduced fixed costs and maximizes GPU utilization. To this end we perform a 2 D grid search investigating the average firing rate across the entire population of excitatory neurons in dependence on two parameters: the constant background stimulation currents I_x^E and I_x^I measured in multiples of the rheobase current $I_x^X = I_{thX} \cdot I_{rheo}^X$ (Fig. 5). The grid has a dimension of 40×40 and for each point our benchmarking model is simulated for 10 seconds biological time. This results in a total simulation time of 16,000 seconds. Using a network size of $N = 25,000$ neurons and $M = 193e6$ connections we obtain plausible spike rasters for different combinations of the background stimulation of excitatory and inhibitory neuron populations (Fig. 5 A). The activity of the excitatory populations increases along the I_{thE}-axis and decreases by larger stimulation of the inhibitory populations. Metastability emerges in a large parameter regime. The comparison of NEST(Fig. 5 B) and GeNN (Fig. 5 C) shows only negligible differences due to the random network structure. The grid search with NEST on S3 took 45,918 seconds compared to 2,309 seconds with GeNN by utilizing a batch size of 40. This is a speed-up of

$$S = \frac{T_{NEST}}{T_{GeNN40}} = 19.9.$$

By using a batch size of 1, GeNN needs 4,432 seconds. The speed-up by fully utilizing the GPU with a batch size of 40 compared to the batch size of 1 is

$$S = \frac{T_{GeNN1}}{T_{GeNN40}} = 1.9.$$

GeNN reduces the memory consumption in the batch mode by using the same connectivity matrices for all network instances. Identical connectivity matrices introduce correlations across network instances within a batch.

Metastability emerges robustly in attractor networks with large clusters

We here address the scientific question of whether the proposed network architecture with the excitatory-inhibitory cluster topology supports metastability for large cluster sizes. Hitherto we had analyzed metastability in detail for the network size of $N = 5,000$ neurons and found that it can be robustly achieved for a varying number of clusters size (Rostami et al., 2022). In our calibration approach of Fig. 5 we simulated five-fold larger networks

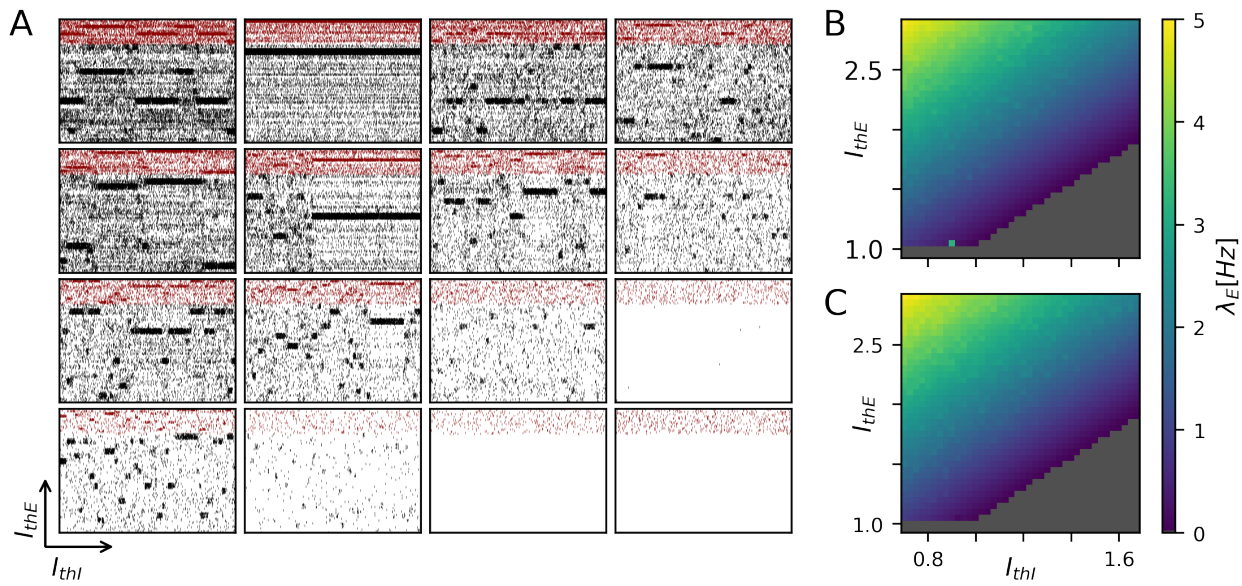

Figure 5: **Two-dimensional parameter grid-search with GeNN and NEST**. Average spontaneous firing rates of excitatory neurons (20,000, inhibitory 5,000) during 10s of biological time in dependence on constant excitatory and inhibitory input currents (40×40 parameter values) to individual neurons with $I_x^X = I_{thX} \cdot I_{rheo}^X$, I_{thX} denoting the proportionality constant with the rheobase current. (A) Raster displays of excitatory (black) and inhibitory (red) spiking activity along the grid. The network shows metastability in a large subset of the parameter space with a varying stability of the attractors. Shown is every 8th sample from the whole grid d a portion of 8% of neurons from each of the 20 clusters. (B) With NEST all parameter combinations are simulated with independent model instances. The whole simulation of the grid takes 45,918 seconds (1h 45m) on server 3. (C) The gridsearch takes 2,309 seconds (39m) on server 3 with the high-end GPU. The GPU is fully utilized by using a batch size of 40 at the cost of using the same connectivity matrix per batch. This is a speed-up of $S \approx 20$.

($N = 25,000$) for a fix number of 20 clusters. We then consistently varied the excitatory and inhibitory background input to all neurons and kept all other network parameters fixed. As a result we observe that metastability emerges also for the large network and cluster sizes (see also Fig. 2). We propose that the E/I topology affords metastability for large network sizes.

Conclusion

We approached the present study from a practical perspective having in mind that part of the every-day challenge in many computational neuroscience labs is to efficiently conduct

spiking neural network simulations. We have identified two limiting factors in our scientific work. First, with increasing network size in terms of neuron and synapse numbers, the exploration of model behavior and in particular the calibration of model performance and robustness in dependence on multiple parameters is severely hampered by the computational load. In practice this often requires to strictly limit network size. Second, modeling functional spiking neural network of sensory-motor transformation to simulate animal behavior can require long simulation times over minutes or even hours in biological time. Closing the loop in the spiking control of artificial agents in a dynamic environment further increases computational load and thus simulation time, and real-time applications impose strict limitations on network size. The overall spiking activity in the network is an additional factor that determines the computational load. While we did not attempt to determine the dependence of simulation time on firing activity, we found that in our network architecture the average firing rate per neuron of ≈ 8.5 Hz stayed approximately constant over the complete range of tested network sizes and is in a similar range as the average firing rate of 5.7 Hz in (Knight and Nowotny, 2021, data regenerated by us).

Our primary goal was thus to evaluate whether and in which ways GPU based simulation can effectively support conventional CPU based simulation, extending our tool set. For this we used a benchmark model that employs a dense recurrent connectivity and we used hardware configurations (Table 3) that we can consider standard equipment in computational labs. We deliberately did not attempt comparison with supercomputing facilities as access is typically restricted and computing time limited.

The NEST simulation environment is widely used in neuroscience research (Tikidji-Hamburyan et al., 2017). Supported by a sustained developers community it has seen continuous development and extension such as e.g. parallelization across multiple cores (Morrison et al., 2005), and optimization for very large brain-scale networks (Kunkel et al., 2012) using supercomputers (Helias et al., 2012). Curated documentation, a plethora of NEST-based publications and abundant resources of publicly accessible code add to its force. These features make NEST user-friendly and an excellent choice specifically for prototyping and efficient use with high performance computing. Our results show that GPU based simulation with GeNN can be beneficial for large networks and long-term biological simulation. In our hands the speed-up factor for calibrating a network of $25,000$ neurons for two independent parameters was on the order of $10 - 20$. Even a low-budget GPU allows to conduct scientific projects with networks of considerable size, which may be specifically usefull for teaching and training purposes (Currin et al., 2019). To our experience, and in comparison to NEST, GeNN requires more advanced programming expertise. Documented code solutions for all steps of simulation and analysis are available in a dedicated code repository (`https://github.com/nawrotlab/SNN_GeNN_Nest`).

We conclude that NEST is flexible, provides abundant documentation and many implemented neuron and synapse models and thus is ideal for prototyping while GeNN provides a highly efficient approach to the simulation of large scale SNN over long biological times even on a low budget that will see a growing user community. Combining both approaches will be our choice henceforth.

Materials and Methods

Hardware configurations

We perform benchmark simulations on hardware systems that can be considered as standard equipment in a computational research lab. We do not attempt to use High Performance Computing facilities that for most users, are available only for highly limited computing time and require an overhead in scheduling simulation jobs. We employ three compute server systems specified in Table 3 that are between 6 years and six months old. The amount of investment at the time of purchase has been fairly stable on the order of $\$7,000 - 10,000$ depending on whether a state of the art GPU was included. Servers 1 and 3 are equipped with GPUs. The GeForce GTX 970 (Server 1; NVIDIA, Santa Clara, USA) can now be considered a low-cost GPU in the price range of $\$ \, 300$. The Quadro RTX A6000 (Server 3; NVIDIA, Santa Clara, USA) is one example of current state of the art high-end GPUs, for which prices are typically in the range of $\$ \, 3,000 - 5,000$.
We use the job scheduler HTCondor (Thain et al., 2005) on all servers independently with one job scheduler per server. We ensure acquisition of all cores and the whole GPU to a running job and prevent other jobs from execution till the running job is finished.

Table 3: Hardware configurations and benchmark setups

CPU			**GPU**			
#cores	Clock speed [GHz]	Memory [GB]	Architecture	#CUDA-cores	Memory [GB]	Performance (single) [TFLOPS]
Server 1 Ubuntu 16.04 LTS						
Dual AMD Opteron 6380			GeForce GTX 970			
2 x 16	2.5	128	Maxwell	1,664	4	3.5
Server 2 Ubuntu 16.04 LTS						
Dual Intel Xeon E5-2630 v4			-			
2 x 10	2.2	192	-	-	-	-
Server 3 Ubuntu 20.04 LTS						
Intel Xeon Gold 6248R			Quadro RTX A6000			
24	3.0	128	Ampere	10,752	48	38.7

Simulators

We benchmark with the Neural Simulation Tool (NEST) (Gewaltig and Diesmann, 2007) and the GPU-enhanced Neuronal Networks (GeNN) framework (Yavuz et al., 2016), two very different simulators which follow different design principles and target different hardware platforms.

The Neural Simulation Tool (NEST) (Gewaltig and Diesmann, 2007) is targeted towards computational neuroscience research and provides various biologically realistic neuron and synapse models. With the introduction of NESTML (Plotnikov et al., 2016) it also allows custom model definitions for non-expert programmers. NEST uses the so called simulation language interpreter (SLI) to orchestrate the simulation during run-time as a stack machine. This allows a modular design of the whole simulator and thus the usage of precompiled models. NEST supports parallization across multiple threads as well as multiple processes which can be distributed across multiple nodes via the message passing interface (MPI). It is suitable for the whole range of desktop workstations to whole multi-node High-Performance clusters. We use Nest version 3.1 (in its standard cmake setting) with the Python interface PyNEST (Eppler et al., 2009) and Python version 3.8 to define our model and control the simulation.

GeNN (GPU-enhanced Neuronal Networks) is a C++ library to generate code from a high level description of the model and simulation procedure. It employs methods to map the description of the neuron models, the network topology as well as the design of the experiment to plain C++ code which then is compiled for a specific target. GeNN supports single-threaded CPUs or GPU with a CPU as host. The scope of GeNN is broader as NEST's. With features like a batch-mode which allows for inference of multiple inputs, GeNN becomes especially usefull for machine learning tasks with SNNs as well as general research in computational neuroscience. GeNN is more rigid in the network topology as well as in its parameters after the code generation is finished. It does not support general reconfiguration of the network during the simulation. The code generation framework in GeNN allows for a heavily optimized code depending on the usecase. One of these optimization possibilities is the choice of the connectivity matrices. Here we utilize only the sparse format and the procedural connectivity as explained bellow.

Sparse matrix format The connectivity matrix is generated on the GPU during the loading / re-initialization of the model. It is persistent during the simulation. No additional computational load is generated during the simulation.

Procedural matrix format Single elements of the connectivity matrix are generated on demand during simulation for the spike propagation. Only fixed random seeds are saved during the loading /re-initialization of the model. An additional computational load is present during the simulation but the memory consumption is low.

We use GeNN version 4.6.0 and 4.7.0 with its Python interface PyGeNN (Knight et al., 2021) and Python version 3.6.

Neuron model and network architecture

We use the biologically realistic spiking neural network described by (Rostami et al., 2022) as benchmarking model. This model uses the leaky integrate-and-fire neurons with exponential shaped postsynaptic currents as follows,

$$\begin{aligned} \frac{dV}{dt} &= \frac{-(V - E_L)}{\tau_m} + \frac{I_{syn} + I_x}{C_m} \\ I_{syn} &= \sum_i I_{syn}^i \\ \tau_{syn} \frac{dI_{syn}^i}{dt} &= \sum_j J_{ij} \sum_k \delta\left(t - t_k^j\right) \end{aligned} \tag{3}$$

Equations 3 describe the dynamics of the neurons. NEST implements an exact integration scheme as introduced by (Rotter and Diesmann, 1999) to solve these equations efficiently for two different synaptic input ports. We implement the same integration scheme but modify it to fit only one synaptic input port while treated as piece-wise linear to combine different synapse types in terms of their time constant.

We follow the widely-used assumption that 80% of neurons in the neocortex are excitatory while 20% are inhibitory to build the network model, which is based on the statistical work in the mouse neocortex by (Braitenberg and Schüz, 1998). Connections between neurons are established with the probabilities $p_{EE} = 0.2$, $p_{EI} = p_{IE} = p_{II} = 0.5$. Autapses and multapses are excluded. Excitatory and inhibitory populations are divided into Q clusters by increasing weights of intra-cluster connections while decreasing weights of inter-cluster connections. Weights are calculated to ensure a local balance between excitation and inhibition as introduced before for binary networks (Rost et al., 2018). Parameters are given in Tables 4 and 5.

In our model each neuron can be presynaptic and/or postsynaptic partner to all other neurons. As a result the number of synapses M scales quadratically with the neuron number N. We calculate the expectation of the number of synapses M by using the assumption of the portioning into 80% excitatory and 20% inhibitory neurons and then calculating the expected number of synapses between the combinations of both by using the connection probabilities. Due to our exclusion of autapses we have to reduce the number of postsynaptic possibilities by one for the connections between the excitatory and excitatory neurons as well as between inhibitory neurons and inhibitory neurons.

$$\begin{aligned} \mathbb{E}(M) &= (N \cdot 0.8) \cdot (p_{EE} \cdot (N \cdot 0.8 - 1) + p_{IE} \cdot (N \cdot 0.2)) \\ &\quad + (N \cdot 0.2) \cdot (p_{EI} \cdot (N \cdot 0.8) + p_{II} \cdot (N \cdot 0.2 - 1)) \\ &= 0.308 \cdot N^2 - 0.25 \cdot N \approx 0.308 \cdot N^2. \end{aligned} \tag{4}$$

Formula 4 determines the overall network connectivity parameter $p = 0.308$.

Table 4: Constant model parameters for all simulation modalities.

Parameter	Unit	Value
E_L	mV	0
V_{th}	mV	20
V_r	mV	0
C	pF	1
${\tau_m}^E$	ms	20
${\tau_m}^I$	ms	10
${\tau_{syn}}^E$	ms	3
${\tau_{syn}}^I$	ms	2
τ_r	ms	5
p_{EE}	-	0.2
p_{EI}, p_{IE}, p_{II}	-	0.5
g	-	1.2
N_Q	-	20
R_J	-	$3/4$

Simulations

We perform two different sets of simulations with all tested combinations of a server and a simulator. We enforce a maximum run-time of 5 hours per simulation. The first set contains simulations of two networks with the sizes of 5,000 and 50,000 neurons and simulation times of 0; 1; 5; 25; 50; 100; 175; 350; 625; 1,250 and 2,500 seconds (zero seconds are used to determine the simulation preperation). We execute each simulation 5 times.
All simulations in the second set are 10 seconds long and we used network sizes between 500 neurons and $\approx 3,6$ million neurons. We execute each simulation 10 times. Based on the recorded simulation times we estimate the network size for each configuration which fullfills real-time requirements and test this network size.

For GeNN we use the spike-recorder, which saves the spikes during the simulation in the RAM of the GPU and allows a block transfer of all spikes for a given number of simulation steps. The consumption of GPU-RAM is composed of the model with all its state variables, connectivity matrices and the memory for the spike recorder. The memory consumption of the model is independent of the simulation time but heavily dependent on the network size and the choice of the connectivity matrix type. The size of the memory of the spike-recorder depends on the network size and the simulation steps.

We implement a partitioning of simulations into sub-simulations, if the GPU-RAM is too

Table 5: Variable model parameters for different simulation modalities.

		Value			
Parameter	**Unit**	**5k**	**50k**	**Var N**	**Grid**
N_E	-	4k	50k	$0.8 \cdot N$	20k
N_I	-	1k	10k	$0.2 \cdot N$	5k
I_x^E	pA	2.13	2.13	2.13	0.95 - 2.9
I_x^I	pA	1.24	1.24	1.24	0.7 - 1.675
J_{E+}	-	10.0	10.0	10.0	2.75
J_{EE}	pA	0.329	0.104	$f(N)$	0.147
J_{IE}	pA	0.250	0.079	$f(N)$	0.112
J_{EI}	pA	-0.877	-0.277	$f(N)$	-0.392
J_{II}	pA	-1.337	-0.423	$f(N)$	-0.598

small, to fit the model and the spike-recorder in one simulation. We determine the maximum number of simulation steps for a large network which fits in the GPU-RAM and use this number as a heuristic. If the product of network size and simulation steps exceeds the heuristic we divide the simulation into two iteratively till the product of both is smaller than the heuristic. As a fallback mechanism we include an error handling to divide the simulation further if the model and the spike-recorder don't fit the GPU-RAM. We transfer the spikes from the GPU to the host after each simulation-partition and process them into one list containing the neuron-ID and the spiking time by 16 workers. A greater number of workers exceeded the available RAM on our servers for large networks.
Simulations are performed one time with the `SPARSE` format of the connectivity matrices and one time with the `PROCEDURAL` connectivity. We delete the folder with the generated and compiled code before we submit the first job to the scheduler to not use any code of previous code generation runs, which would consumed time by this process. We run one complete sequence of all different simulation times before switching to the other matrix type. This ensures that the code is generated 5 and 10 times respectively for the two different simulation sets.

We use the same order of execution for the simulations in NEST. NEST does not generate any files which are reused for following simulations.

Gridsearch

We implement a general framework to perform a grid search. The framework takes multiple parameters and generates a regular mesh grid of the parameter values. It simulates for each point in the grid, analyzes the network activity in the simulation and saves the result together with its position in the grid to a pickle file.

We implement the gridsearch in GeNN by utilizing the batch mode with N_{Batch} network instances in each run. We add global parameters to the model description. Global parameters are not translated into constants during compilation but can be set independently for each instance in a batch and can be modified during run-time. The instances share all other parameters and the specific connectivity matrices. We use the `SPARSE` format of the connectivity matrices and generate the simulation code only one time and then reinitialize the network after each batch and set the global parameters to their respective values. The re-initialization regenerates the connectivity matrix and resets all state variables as well as the spike buffer. The parameters as well as the batch size N_{batch} can only be changed by recompilation. Each time a number of samples equal to N_{batch} is drawn without replacement from the grid till all points in the grid are simulated. If less samples as N_{batch} are left, the free slots are filled by setting the global parameters to 0; the results of these slots are ignored. $N_W = 24$ workers process the spiketimes into a representation equals to NESTs representation after a block transfer at the end of each simulation. Afterwards we analyze the spiketimes serially for each instance with the specific analyzes. The process writes the results afterwards to the pickle file.

We implement the gridsearch in NEST by creating a list of all parameters in the grid and then parallizing the simulations with Pathos to $N_W = 6$ workers. The workers import NEST independently, thus each simulation is completely independent. Each worker utilizes $N_T = 4$ threads. We match the number of cores available on the system: $N_{cores} = N_W \cdot N_T$. The workers write the results to one pickle file, which is protected by a lock to ensure data integrity by only allowing one worker to write at a time. This pickle file does not contain by default the spiketimes but instead only the result of the specific analyzes applied.

Definition of fixed and variable costs

We break the simulation cost into fixed and variable costs:

$$T_{sim} = T_{fixed} + T_{var}.$$

The fixed costs involve all steps necessary to set up and prepare the model before the first simulation step. They are independent of the time span to be simulated. The variable costs involve the actual propagation of the network state during each time step and the overall time used for data collection during a simulation. We include timestamps in the simulation

scripts to access the runtime of different execution phases. Due to the different design principles not all phases of GeNN can be mapped to NEST. Table 6 shows the defined phases and the commands which are executed in the phase.

Table 6: Execution phases of simulation

		GeNN	**NEST**
Build	**Fixed**	*model.add...*	*nest.create...* *nest.connect...* *nest.setstatus...*
Compile		*model.build*	
Load		*model.load*	*nest.prepare*
Simulate	**Variable**	*model.step_time*	*nest.run*
Download		*model.pull_recording_buffers* + conversion to NEST data format	*nest.getstatus*(*spike_recorder*)

Data analysis

We estimate the median of the simulation-speed (T_{var}/T_{bio}) per network size for each configuration (server + simulator) individually. On this result we perform weighted least squares with three modifications: *i*) We scale the quatratic term by the connection density $p = 0.308$ to relate this term to the number of synapses M, *ii*) we use the relative error and *iii*) we weight the samples by the inverse of the density along the independent variable N. This let us balance the influence of the large networks and the larger number of samples for small networks with the factor λ (0.75). We estimate the density $F(N)$ by a kernel density estimation with a Gaussian kernel ($\sigma = 10,000$ neurons). $\frac{\varphi(N,a)_i - y_i}{y-i}$ and weight it by the factor simulated

$$\begin{aligned} \varphi(N, \boldsymbol{\alpha}) &= \alpha_2 \cdot 0.308 N^2 + \alpha_1 \cdot N + \alpha_0 \\ \ell &= \sum_{i=1}^{n} \left(\frac{\lambda}{F(N)} + (1-\lambda) \right) (\varphi(N, \boldsymbol{\alpha}) - y_i)^2 \\ \boldsymbol{\alpha} &= \underset{\boldsymbol{\alpha}}{\arg\min} \, (\ell) \end{aligned} \tag{5}$$

We use equation 5 for the regression.

Conflict of Interest Statement

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Author Contributions

FS and MN designed the research. FS carried out simulations and analysis of results. VR and MN supervised project. FS and MN wrote the manuscript.

Funding

This research is supported by the German Research Foundation through the collaborative research center *Key Mechanismsm of Motor Control in Health and Disease* (DFG-CRC 1451, grant no. 431549029 to MN, https://www.crc1451.uni-koeln.de/). FS is funded through the DFG research training group *Neural Circuit Analysis* (DFG-RTG 1960, grant no. 233886668).

Acknowledgments

We are grateful to James Knight for his support with GeNN.

Data Availability Statement

The generated data and the source code to produce this data as well to analyze it for this study can be found in the GitHub repository `https://github.com/nawrotlab/SNN_GeNN_Nest`.

References

Abeles, M. (1991). *Corticonics: Neural circuits of the cerebral cortex*. Cambridge University Press.

Amit, D. and Brunel, N. (1997). Model of global spontaneous activity and local structured activity during delay periods in the cerebral cortex. *Cerebral Cortex*, 7(3):237–252.

Bartolozzi, C., Indiveri, G., and Donati, E. (2022). Embodied neuromorphic intelligence. *Nature Communications*, 13:1024.

Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Stewart, T., Rasmussen, D., Choo, X., Voelker, A., and Eliasmith, C. (2014). Nengo: a Python tool for building large-scale functional brain models. *Frontiers in Neuroinformatics*, 7.

Ben-Shalom, R., Ladd, A., Artherya, N. S., Cross, C., Kim, K. G., Sanghevi, H., Korngreen, A., Bouchard, K. E., and Bender, K. J. (2022). NeuroGPU: Accelerating multi-compartment, biophysically detailed neuron simulations on GPUs. *Journal of Neuroscience Methods*, 366:109400.

Blundell, I., Brette, R., Cleland, T. A., Close, T. G., Coca, D., Davison, A. P., Diaz-Pier, S., Fernandez Musoles, C., Gleeson, P., Goodman, D. F. M., Hines, M., Hopkins, M. W., Kumbhar, P., Lester, D. R., Marin, B., Morrison, A., Müller, E., Nowotny, T., Peyser, A., Plotnikov, D., Richmond, P., Rowley, A., Rumpe, B., Stimberg, M., Stokes, A. B., Tomkins, A., Trensch, G., Woodman, M., and Eppler, J. M. (2018). Code generation in computational neuroscience: A review of tools and techniques. *Frontiers in Neuroinformatics*, 12.

Boucsein, C., Nawrot, M., Schnepel, P., and Aertsen, A. (2011). Beyond the Cortical Column: Abundance and Physiology of Horizontal Connections Imply a Strong Role for Inputs from the Surround. *Frontiers in Neuroscience*, 5.

Braitenberg, V. and Schüz, A. (1998). *Cortex: Statistics and Geometry of Neuronal Connectivity*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Brette, R. and Goodman, D. F. (2012). Simulating spiking neural networks on GPU. *Network: Computation in Neural Systems*, 23(4):167–182.

Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J. M., Diesmann, M., Morrison, A., Goodman, P. H., Harris, F. C., Zirpe, M., Natschläger, T., Pecevski, D., Ermentrout, B., Djurfeldt, M., Lansner, A., Rochel, O., Vieville, T., Muller, E., Davison, A. P., El Boustani, S., and Destexhe, A. (2007). Simulation of networks of spiking neurons: A review of tools and strategies. *Journal of Computational Neuroscience*, 23(3):349–398.

Brunel, N. (2000). Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *Journal of computational neuroscience*, 8(3):183–208.

Bruzzone, M., Chiarello, E., Albanesi, M., Miletto Petrazzini, M. E., Megighian, A., Lodovichi, C., and Dal Maschio, M. (2021). Whole brain functional recordings at cellular resolution in zebrafish larvae with 3d scanning multiphoton microscopy. *Scientific Reports*, 11.

Büsing, L., Schrauwen, B., and Legenstein, R. (2010). Connectivity, dynamics, and memory in reservoir computing with binary and analog neurons. *Neural computation*, 22(5):1272–1311.

Carlson, K., Nageswaran, J., Dutt, N., and Krichmar, J. (2014). An efficient automated parameter tuning framework for spiking neural networks. *Frontiers in Neuroscience*, 8.

Chicca, E., Stefanini, F., Bartolozzi, C., and Indiveri, G. (2014). Neuromorphic Electronic Circuits for Building Autonomous Cognitive Systems. *Proceedings of the IEEE*, 102(9):1367–1388.

Currin, C. B., Khoza, P. N., Antrobus, A. D., Latham, P. E., Vogels, T. P., and Raimondo, J. V. (2019). Think: Theory for africa. *PLOS Computational Biology*, 15(7):1–5.

Diesmann, M. and Gewaltig, M.-O. (2002). NEST: An Environment for Neural Systems. *Forschung und Wisschenschaftliches Rechnen Beiträge zum Heinz-Billing-Preis*, 58.

Diesmann, M., Gewaltig, M.-O., and Aertsen, A. (1995). Synod: An environment for neural systems simulations language interface and tutorial.

Diesmann, M., Gewaltig, M.-O., and Aertsen, A. (1999). Stable propagation of synchronous spiking in cortical neural networks. *Nature*, 402(6761):529–533.

Eliasmith, C. and Anderson, C. H. (2003). *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press.

Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., and Rasmussen, D. (2012). A large-scale model of the functioning brain. *Science*, 338(6111):1202–1205.

Eliasmith, C. and Trujillo, O. (2014). The use and abuse of large-scale brain models. *Current Opinion in Neurobiology*, 25:1–6.

Eppler, J., Helias, M., Muller, E., Diesmann, M., and Gewaltig, M.-O. (2009). PyNEST: a convenient interface to the NEST simulator. *Frontiers in Neuroinformatics*, 2.

Eschbach, C. and Zlatic, M. (2020). Useful road maps: studying Drosophila larva's central nervous system with the help of connectomics. *Current Opinion in Neurobiology*, 65:129–137.

Fidjeland, A. K., Roesch, E. B., Shanahan, M. P., and Luk, W. (2009). NeMo: a platform for neural modelling of spiking neurons using GPUs. In *2009 20th IEEE international conference on application-specific systems, architectures and processors*, pages 137–144. IEEE.

Finkelstein, A., Fontolan, L., Economo, M. N., Li, N., Romani, S., and Svoboda, K. (2021). Attractor dynamics gate cortical information flow during decision-making. *Nature Neuroscience*, 24(6):843–850.

Florimbi, G., Torti, E., Masoli, S., D'Angelo, E., and Leporati, F. (2021). Granular layEr Simulator: Design and Multi-GPU Simulation of the Cerebellar Granular Layer. *Frontiers in Computational Neuroscience*, 15.

Gewaltig, M.-O. and Diesmann, M. (2007). Nest (neural simulation tool). *Scholarpedia*, 2(4):1430.

Godfrey, R. K., Swartzlander, M., and Gronenberg, W. (2021). Allometric analysis of brain cell number in Hymenoptera suggests ant brains diverge from general trends. *Proceedings of the Royal Society B: Biological Sciences*, 288(1947):20210199.

Gütig, R. (2016). Spiking neurons can discover predictive features by aggregate-label learning. *Science*, 351(6277):aab4113.

Gütig, R. and Sompolinsky, H. (2006). The tempotron: a neuron that learns spike timing–based decisions. *Nature neuroscience*, 9(3):420–428.

Hammond, C., Bergman, H., and Brown, P. (2007). Pathological synchronization in Parkinson's disease: networks, models and treatments. *Trends in neurosciences*, 30(7):357–364.

Helias, M., Kunkel, S., Masumoto, G., Igarashi, J., Eppler, J., Ishii, S., Fukai, T., Morrison, A., and Diesmann, M. (2012). Supercomputers Ready for Use as Discovery Machines for Neuroscience. *Frontiers in Neuroinformatics*, 6.

Herculano-Houzel, S., Watson, C., and Paxinos, G. (2013). Distribution of neurons in functional areas of the mouse cerebral cortex reveals quantitatively different cortical zones. *Frontiers in neuroanatomy*, 7:35.

Hines, M. L. and Carnevale, N. T. (2001). NEURON: a tool for neuroscientists. *The neuroscientist*, 7(2):123–135.

Hinsch, K. and Zupanc, G. (2007). Generation and long-term persistence of new neurons in the adult zebrafish brain: A quantitative analysis. *Neuroscience*, 146(2):679–696.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8):2554–2558.

Igarashi, J., Yamaura, H., and Yamazaki, T. (2019). Large-Scale Simulation of a Layered Cortical Sheet of Spiking Network Model Using a Tile Partitioning Method. *Frontiers in Neuroinformatics*, 13.

Inagaki, H. K., Fontolan, L., Romani, S., and Svoboda, K. (2019). Discrete attractor dynamics underlies persistent activity in the frontal cortex. *Nature*, 566(7743):212–217.

Indiveri, G., Stefanini, F., and Chicca, E. (2010). Spike-based learning with a generalized integrate and fire silicon neuron. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 1951–1954.

Kasabov, N. and Capecci, E. (2015). Spiking neural network methodology for modelling, classification and understanding of EEG spatio-temporal data measuring cognitive processes. *Information Sciences*, 294:565–575.

Knight, J. C., Komissarov, A., and Nowotny, T. (2021). PyGeNN: A Python Library for GPU-Enhanced Neural Networks. *Frontiers in Neuroinformatics*, 15.

Knight, J. C. and Nowotny, T. (2021). Larger GPU-accelerated brain simulations with procedural connectivity. *Nature Computational Science*, 1(2):136–142.

Knight, J. C. and Nowotny, T. (2022). Efficient GPU Training of LSNNs Using EProp. In *Neuro-Inspired Computational Elements Conference*, NICE 2022, page 8–10, New York, NY, USA. Association for Computing Machinery.

Kulesza, R. J. (2007). Cytoarchitecture of the human superior olivary complex: Medial and lateral superior olive. *Hearing Research*, 225(1):80–90.

Kunkel, S., Potjans, T., Eppler, J., Plesser, H. E., Morrison, A., and Diesmann, M. (2012). Meeting the Memory Challenges of Brain-Scale Network Simulation. *Frontiers in Neuroinformatics*, 5.

Markram, H., Muller, E., Ramaswamy, S., Reimann, M. W., Abdellah, M., Sanchez, C. A., Ailamaki, A., Alonso-Nanclares, L., Antille, N., Arsever, S., Kahou, G. A. A., Berger, T. K., Bilgili, A., Buncic, N., Chalimourda, A., Chindemi, G., Courcol, J.-D., Delalondre, F., Delattre, V., Druckmann, S., Dumusc, R., Dynes, J., Eilemann, S., Gal, E., Gevaert, M. E., Ghobril, J.-P., Gidon, A., Graham, J. W., Gupta, A., Haenel, V., Hay, E., Heinis, T., Hernando, J. B., Hines, M., Kanari, L., Keller, D., Kenyon, J., Khazen, G., Kim, Y., King, J. G., Kisvarday, Z., Kumbhar, P., Lasserre, S., Le Bé, J.-V., Magalhães, B. R., Merchán-Pérez, A., Meystre, J., Morrice, B. R., Muller, J., Muñoz-Céspedes, A., Muralidhar, S., Muthurasa, K., Nachbaur, D., Newton, T. H., Nolte, M., Ovcharenko, A., Palacios, J., Pastor, L., Perin, R., Ranjan, R., Riachi, I., Rodríguez, J.-R., Riquelme, J. L., Rössert, C., Sfyrakis, K., Shi, Y., Shillcock, J. C., Silberberg, G., Silva, R., Tauheed, F., Telefont, M., Toledo-Rodriguez, M., Tränkler, T., Van Geit, W., Díaz, J. V., Walker, R., Wang, Y., Zaninetta, S. M., DeFelipe, J., Hill, S. L., Segev, I., and Schürmann, F. (2015). Reconstruction and simulation of neocortical microcircuitry. *Cell*, 163(2):456–492.

Mazzucato, L. (2022). Neural mechanisms underlying the temporal organization of naturalistic animal behavior. *arXiv*.

Mazzucato, L., La Camera, G., and Fontanini, A. (2019). Expectation-induced modulation of metastable activity underlies faster coding of sensory stimuli. *Nature Neuroscience*, 22:787–796.

McIntyre, C. C. and Hahn, P. J. (2010). Network perspectives on the mechanisms of deep brain stimulation. *Neurobiology of disease*, 38(3):329–337.

Menzel, R. (2012). The honeybee as a model for understanding the basis of cognition. *Nature reviews. Neuroscience*, 13:758–68.

Morrison, A., Mehring, C., Geisel, T., Aertsen, A., and Diesmann, M. (2005). Advancing the Boundaries of High-Connectivity Network Simulation with Distributed Computing. *Neural Computation*, 17(8):1776–1801.

Morrison, A., Straube, S., Plesser, H. E., and Diesmann, M. (2007). Exact subthreshold integration with continuous spike times in discrete-time neural network simulations. *Neural computation*, 19(1):47–79.

Mutch, J., Knoblich, U., and Poggio, T. (2010). CNS: a GPU-based framework for simulating cortically-organized networks. *MIT CSAIL*, 17:2013.

Nageswaran, J. M., Dutt, N., Krichmar, J. L., Nicolau, A., and Veidenbaum, A. V. (2009). A configurable simulation environment for the efficient simulation of large-scale spiking neural networks on graphics processors. *Neural Networks*, 22(5):791–800.

Neftci, E., Binas, J., Rutishauser, U., Chicca, E., Indiveri, G., and Douglas, R. J. (2013). Synthesizing cognition in neuromorphic electronic systems. *Proceedings of the National Academy of Sciences of the United States of America*, 110(37).

Oorschot, D. E. (1996). Total number of neurons in the neostriatal, pallidal, subthalamic, and substantia nigral nuclei of the rat basal ganglia: A stereological study using the cavalieri and optical disector methods. *Journal of Comparative Neurology*, 366(4):580–599.

Parrish-Aungst, S., Shipley, M., Erdelyi, F., Szabó, G., and Puche, A. (2007). Quantitative analysis of neuronal diversity in the mouse olfactory bulb. *The Journal of comparative neurology*, 501:825–36.

Parsa, M., Mitchell, J. P., Schuman, C. D., Patton, R. M., Potok, T. E., and Roy, K. (2019). Bayesian-based Hyperparameter Optimization for Spiking Neuromorphic Systems. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4472–4478.

Pfeiffer, M. and Pfeil, T. (2018). Deep Learning With Spiking Neurons: Opportunities and Challenges. *Frontiers in Neuroscience*, 12.

Plotnikov, D., Rumpe, B., Blundell, I., Ippen, T., Eppler, J. M., and Morrison, A. (2016). NESTML: a modeling language for spiking neurons. *In Modellierung 2016, March 2-4 2016*.

Potjans, T. C. and Diesmann, M. (2012). The Cell-Type Specific Cortical Microcircuit: Relating Structure and Activity in a Full-Scale Spiking Network Model. *Cerebral Cortex*, 24(3):785–806.

Raji, J. I. and Potter, C. J. (2021). The number of neurons in Drosophila and mosquito brains. *PLOS ONE*, 16(5):1–11.

Rapp, H. and Nawrot, M. P. (2020). A spiking neural program for sensorimotor control during foraging in flying insects. *Proceedings of the National Academy of Sciences*, 117(45):28412–28421.

Rapp, H., Nawrot, M. P., and Stern, M. (2020). Numerical Cognition Based on Precise Counting with a Single Spiking Neuron. *iScience*, 23(2):100852.

Rost, T., Deger, M., and Nawrot, M. P. (2018). Winnerless competition in clustered balanced networks: inhibitory assemblies do the trick. *Biological cybernetics*, 112(1-2):81–98.

Rostami, V., Rost, T., Riehle, A., van Albada, S. J., and Nawrot, M. P. (2022). Excitatory and inhibitory motor cortical clusters account for balance, variability, and task performance. *bioRxiv*.

Rotter, S. and Diesmann, M. (1999). Exact digital simulation of time-invariant linear systems with applications to neuronal modeling. *Biological Cybernetics*, 81(5):381–402.

Sakai, K. and Miyashita, Y. (1991). Neural organization for the long-term memory of paired associates. *Nature*, 354(6349):152–155.

Sarko, D., Catania, K., Leitch, D., Kaas, J., and Herculano-Houzel, S. (2009). Cellular scaling rules of insectivore brains. *Frontiers in Neuroanatomy*, 3.

Schmidt, M., Bakker, R., Shen, K., Bezgin, G., Diesmann, M., and van Albada, S. J. (2018). A multi-scale layer-resolved spiking network model of resting-state dynamics in macaque visual cortical areas. *PLOS Computational Biology*, 14(10):1–38.

Schmuker, M., Pfeil, T., and Nawrot, M. P. (2014). A neuromorphic network for generic multivariate data classification. *Proceedings of the National Academy of Sciences*, 111(6):2081–2086.

Singer, W. and Gray, C. M. (1995). Visual feature integration and the temporal correlation hypothesis. *Annual review of neuroscience*, 18(1):555–586.

Stimberg, M., Brette, R., and Goodman, D. (2019). Brian 2, an intuitive and efficient neural simulator. *eLife*, 8.

Tanaka, G., Yamane, T., Héroux, J. B., Nakane, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D., and Hirose, A. (2019). Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100–123.

Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., and Maida, A. (2019). Deep learning in spiking neural networks. *Neural Networks*, 111:47–63.

Thain, D., Tannenbaum, T., and Livny, M. (2005). Distributed computing in practice: the Condor experience. *Concurrency - Practice and Experience*, 17(2-4):323–356.

Tikidji-Hamburyan, R. A., Narayana, V., Bozkus, Z., and El-Ghazawi, T. A. (2017). Software for Brain Network Simulations: A Comparative Study. *Frontiers in Neuroinformatics*, 11.

Van Albada, S. J., Helias, M., and Diesmann, M. (2015). Scalability of asynchronous networks is limited by one-to-one mapping between effective connectivity and correlations. *PLoS computational biology*, 11(9):e1004490.

Van Vreeswijk, C. and Sompolinsky, H. (1996). Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science*, 274(5293):1724–1726.

White, J. G., Southgate, E., Thomson, J. N., and Brenner, S. (1986). The structure of the nervous system of the nematode Caenorhabditis elegans. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 314(1165):1–340.

Witthöft, W. (1967). Absolute Anzahl und Verteilung der Zellen im Hirn der Honigbiene. *Zeitschrift für Morphologie der Tiere*, 61(1):160–184.

Wyrick, D. and Mazzucato, L. (2021). State-dependent regulation of cortical processing speed via gain modulation. *Journal of Neuroscience*, 41(18):3988–4005.

Yamaura, H., Igarashi, J., and Yamazaki, T. (2020). Simulation of a Human-Scale Cerebellar Network Model on the K Computer. *Frontiers in Neuroinformatics*, 14.

Yavuz, E., Turner, J., and Nowotny, T. (2016). GeNN: a code generation framework for accelerated brain simulations. *Scientific reports*, 6(1):1–14.

Zenke, F. and Ganguli, S. (2018). SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks. *Neural Computation*, 30(6):1514–1541.

Zhao, Y. and Wang, Dan O.and Martin, K. C. (2009). Preparation of Aplysia sensory-motor neuronal cell cultures. *Journal of visualized experiments : JoVE*.