PNAS
www.pnas.org

Supplementary Information for

Neuronal spike-rate adaptation supports working memory in language processing

Hartmut Fitz, Marvin Uhlmann, Dick van den Broek, Renato Duarte, Peter Hagoort and Karl Magnus Petersson

Hartmut Fitz.
E-mail: hartmut.fitz@mpi.nl

This PDF file includes:

www.pnas.org/cgi/doi/10.1073/pnas.2000222117

Supporting Information Text

Methods and Materials

Neuron and synapse model. Integrate-and-fire neurons had a fixed voltage threshold with conductance-based mechanisms for refractoriness and spike-rate adaptation (1). The sub-threshold membrane dynamics is described by the equation

$$C_m \frac{dV(t)}{dt} = \frac{1}{R_m}(V_{\rm rest} - V(t)) + I(t) - (g_{\rm sra}(t) + g_{\rm ref}(t))(V(t) - E_K) \quad [1]$$

where $V_{\rm rest}$ is the resting potential, $R_{\rm m}$ denotes the leakage resistance, $C_{\rm m}$ the membrane capacitance and $I(t)$ the total current flowing into the neuron at time t. When the membrane potential reached the threshold $V_{\rm th}$, a spike occurred and V was reset to $V_{\rm rest}$. Spike-rate adaptation was modeled as a K^+-conductance $g_{\rm sra}$ with reversal potential E_K. Following a spike, this conductance was increased by a $g_{\rm sra} \leftarrow g_{\rm sra} + \Delta g_{\rm sra}$ and it decayed back to 0 exponentially with time constant $\tau_{\rm sra}$ otherwise.

$$\tau_{\rm sra} \frac{dg_{\rm sra}(t)}{dt} = -g_{\rm sra}(t) \quad [2]$$

Another conductance $g_{\rm ref}$ generated a relative refractory period during which neurons were prevented from spiking immediately following an action potential. Its dynamics was also modeled as an exponential decay with time constant $\tau_{\rm ref}$.

$$\tau_{\rm ref} \frac{dg_{\rm ref}(t)}{dt} = -g_{\rm ref}(t) \quad [3]$$

Both conductances modeled spike aftereffects and acted homeostatically to prevent runaway activity in the network. While $g_{\rm ref}$ had a strong, short-term impact on the neuron, $g_{\rm sra}$ was weaker but decayed more slowly (e.g., $\tau_{\rm sra} = 200$ ms, $\tau_{\rm ref} = 2$ ms).

Neurons were interconnected through synapses to transmit signals. For simplicity, current-based synapses were used. The shape of synaptic currents $I_{ij}(t)$ was modeled as an instantaneous rise triggered by a pre-synaptic spike, followed by an exponential decay with time constant $\tau_{\rm syn}$,

$$\frac{dI_{ij}(t)}{dt} = -\frac{I_{ij}(t)}{\tau_{\rm syn}} + w_{ij} \sum_{t_j} \delta(t - t_j) \quad [4]$$

where w_{ij} is the synaptic weight from the pre-synaptic neuron j to the post-synaptic neuron i, $\delta(.)$ is the Dirac delta, and t_j are the spike times of the pre-synaptic neuron j. Synaptic weights w_{ij} were drawn uniformly from the real interval [0,1] $\subset \mathbb{R}$. To globally balance excitation and inhibition, inhibitory synapses were scaled to be five times stronger on average than excitatory ones. Weights were kept constant throughout the simulations, i.e., networks implemented no form of synaptic plasticity. The total current into each neuron was the sum of the individual contributions of excitatory and inhibitory synaptic currents from within the network, currents generated by the adaptive conductances, and the external drive due to language input.

Network graphs. Networks were composed of 1,000 neurons of the above type, with 80% excitation (E) and 20% inhibition (I). To eliminate potential memory contributions from recurrent connectivity, networks had feed-forward graphs. In this way, it was guaranteed that network memory was due to neuronal adaptation, and memory differences between models with distinct parameter values for τ_{sra} and Δ_{sra} were not confounded by connectivity effects due to feedback. Feed-forward graphs were obtained by the following procedure: a synapse w_{ij} was inserted between randomly chosen pairs of neurons j and i and the resulting directed graph was tested for cycles. If the synapse created a cycle, it was discarded, else it was retained. This procedure was iterated until the target connection density was reached. Thus, feed-forward graphs contained no cycles within E or I, nor between E and I. Networks were simulated with a temporal resolution of 0.2 ms and Euler's method was used for numerical integration.

Language environment. As input, networks received word sequences generated from argument-structure templates with fixed patterns of function words and variation in content words (2). These constructions could take one, two or three verbal arguments (e.g., intransitive, transitive and dative, respectively) and included syntactic alternations when permissible. For example, the transfer dative X VERBS Y TO Z could occur in prepositional or ditransitive form, in active or passive voice (e.g., *"apples are being given to a woman by the boys"*). Concepts in the grammar were organized into several categories, for example LIVING (MAN, CAT, GIRL, etc.), and OBJECT (BALL, TOY, CAKE, etc.). There were six action categories, each containing four action concepts. One of these action categories corresponded to the unergative intransitive verb class (VERB-UNERG, e.g., JUMP, DANCE), another one to the transitive class (VERB-TRANS, e.g., PUSH, HIT), and so forth. For each construction, a message template specified the set of roles that should be filled and the type of category that should fill them. The dative template, for instance, had role slots for AGENT, THEME and RECIPIENT, with AGENT and RECIPIENT roles selected from the LIVING category, and the THEME role from the OBJECT category. To create a specific message, each role was instantiated by randomly selecting a concept from the appropriate category. For example, when a message stipulated AGENT = LIVING, the AGENT role might be filled with the concept BOY. The sentence template that was paired with the message determined how that message mapped onto words and in which order. The transitive template, for example, specified that the AGENT came first, followed by an ACTION, then the PATIENT. Table S1 shows all constructions in the grammar; these sentence types and their alternations form a core subset of English. Similar to natural language, the grammar had various features such as

Table S1. Constructions in the grammar that was used to generate network input sequences

Construction	Regular pattern	Syntactic alternation
Inanimate Intransitive		
Message template	ACTION = VERB-ERG; PATIENT = OBJECT	
Semantic roles	PATIENT ACTION	–
Example sentence	*A cup was break -ing.*	–
Animate Intransitive		
Message template	ACTION = VERB-UNERG; AGENT = LIVING	
Semantic roles	AGENT ACTION	–
Example sentence	*Old cat -s jump -ed.*	–
Transitive		
Message template	ACTION = VERB-TRAN; AGENT = LIVING; PATIENT = OBJECT	
Semantic roles	AGENT ACTION PATIENT	PATIENT ACTION AGENT
Example sentence	*She kick -ss the small apple.*	*The small apple is kick -par by her.*
Theme-Experiencer		
Message template	ACTION = VERB-THEX; THEME = OBJECT; EXPERIENCER = LIVING	
Semantic roles	THEME ACTION EXPERIENCER	EXPERIENCER ACTION THEME
Example sentence	*They scare -ed the girl -s.*	*The girl -s were scare -par by them.*
Prepositional Dative		
Message template	ACTION = VERB-DAT; AGENT = LIVING; THEME = OBJECT; RECIPIENT = LIVING	
Semantic roles	AGENT ACTION THEME RECIPIENT	THEME ACTION RECIPIENT AGENT
Example sentence	*The man throw -s the stick to a dog.*	*The stick was throw -par to a dog by the man.*
Ditransitive Dative		
Message template	ACTION = VERB-DAT; AGENT = LIVING; THEME = OBJECT; RECIPIENT = LIVING	
Semantic roles	AGENT ACTION RECIPIENT THEME	RECIPIENT ACTION THEME AGENT
Example sentence	*The man throw -s a dog the stick.*	*A dog was throw -par the stick by the man.*
Caused Motion		
Message template	ACTION = VERB-TRAN; AGENT = LIVING; THEME = OBJECT; GOAL = OBJECT	
Semantic roles	AGENT ACTION THEME GOAL	THEME ACTION GOAL AGENT
Example sentence	*The boy -s were push -ing it on the cake.*	*It was push -par on the cake by the boy -s.*
Locative		
Message template	ACTION = VERB-INTR; AGENT = LIVING; GOAL = OBJECT	
Semantic roles	AGENT ACTION GOAL	GOAL ACTION AGENT
Example sentence	*A nice woman was drive -ing to it.*	*It was drive -par to by a nice woman.*

tense (past and present), aspect (simple and progressive), and noun-verb number agreement (e.g., *"boy runs"* versus *"boys run"*). Each noun phrase was either a pronoun (e.g., *she, they, him*) or a lexical noun preceded by a determiner (optional with plurals) and an adjective (25% of the time). These features created considerable positional variation in the input sentences. Inflectional morphemes were represented as separate words (e.g., *"they were chase -ing big dog -s"*), participles and past tense were distinguished by the morphemes *-par* and *-ed*. As model input, a set of ∼1500 unique sentences was generated from the grammar and concatenated into a sequence of 12,500 words. This corresponded to half an hour of language exposure in real time. Constructions were randomly selected with the same probability and instantiated from 75 lexical items in 9 word categories. Sentences were between 2 and 17 words long and the mean utterance length was 8.6 words. The construction grammar generated $\sim 1.67 \times 10^9$ distinct utterances and network input contained less than 0.0001% of the total number of sentences licensed by the grammar. Due to this expressivity of the language, substantial generalization was required from the network and processing strategies based on list memory could be excluded. Prior tests had also shown that the model was relatively insensitive to input size and the number of content words in the language, indicating robust scaling properties.

Comprehension task. The network's task was to incrementally interpret word sequences in terms of semantic roles (*who does what to whom*). Thus, comprehension was modeled as the instantaneous transduction from word sequences to sequences of appropriate semantic categories. There were 6 thematic roles (AGENT, PATIENT, THEME, EXPERIENCER, RECIPIENT, and GOAL), a predicate label ACTION, and an end-of-sentence marker EOS. Roles had to be assigned to all content and function words within a phrase. For example, the target role sequence for the input sentence *"the dog gave the shiny toy to him"* was AGENT AGENT ACTION THEME THEME THEME RECIPIENT RECIPIENT. Role assignment was challenging for a number of reasons. Words were mostly ambiguous; apart from auxiliaries and verbs, all other lexical items could occur in at least two distinct roles. The presence of syntactic alternations created role inversions and temporary ambiguity. Solving the task required the unification of multiple cues (e.g., noun animacy, verb identity, morphology, function words) which could occur anywhere in the preceding context, and non-adjacent to the target word. Early in sentences, role assignment was probabilistic and became increasingly certain towards the end. No input sentence was globally ambiguous. We assessed the model's role assignment on all words in a sentence (overall accuracy) and, separately, on the sentence-final noun phrase (NP) only, where memory demands

were typically the highest and target roles unique. Given the temporary ambiguity in various sentence positions, we estimated that an optimal system could achieve 88% (100%) accuracy overall (final NP) when exposed to the same input and test items as our networks. Although the language generator contained information about syntactic categories, verb classes, structural types, and word order differences between alternations, the network was not given this information. It only experienced lexical sequences and had to preserve all contextual information in its generated state-space trajectories in order to make appropriate categorical generalizations available to the downstream readout.

Input encoding and state collection. Words in the lexicon were encoded into random subsets of 5% of the neurons in the network such that each input word stimulated 50 neurons on average. Populations of neurons representing different words could partially overlap and, statistically, each neuron participated in the representation of 3.75 words. Thus, words were spatially coded by sparse, distributed representations but every neuron in the network was targeted by multiple words. Word input was presented to the network by injecting a step current into the target population, scaled by the corresponding input weight for each neuron. These afferent connections followed an exponential distribution with mean 0.4, and external input was assumed to be excitatory. Word duration varied between 50 ms and 0.45 s, and exposure was proportional to orthographic length with a letter duration of 50 ms. For example, the word *boy* was presented for a duration of $3 \times 50 = 150$ ms while *beautiful* had a duration of $9 \times 50 = 450$ ms. Unless stated otherwise, language input was presented as a continuous word stream without pauses or resetting of the network between sentences. While this sequence was filtered through the model, internal network states were recorded. States were defined as vectors of membrane potentials V with each component corresponding to the current membrane voltage of one neuron. To keep simulations manageable, states were sampled at a constant rate of 200 Hz and averaged within words for each neuron. The collection of states was split into input and validation sets (10,000 and 2,500 words, respectively), and standardized before entering into a maximum entropy classifier (see details below).

Spike rate tuning. Before membranes states were recorded during processing, networks were tuned to exhibit spiking activity at a target rate of 3, 5 or 10 Hz within a 10% margin, respectively. This was done to ensure that networks with different neuronal parameters τ_{sra} and Δ_{sra} showed comparable levels of activity. To do this, all network-internal synapses were disconnected and synapses for external input were scaled such that networks spiked at an evoked rate of 2 Hz. Then, internal synapses were tuned to reach the overall target rate: on the first 1,000 words of the input set, the average instantaneous firing rate was determined and internal connectivity strength was iteratively adjusted up or down using a global synaptic scaling parameter R_{app} (Table S2), until the network target firing rate was achieved. To see how processing memory was modulated by spike rates, we compared networks with different rates, crossed with connection density, on the semantic role assignment task of Figure 2a (main text). The networks had fixed $\tau_{sra} = 200$ ms and $\Delta_{sra} = 4$ nS and were trained and tested as before. The results from this comparison are shown in Figure S1 There was a main effect of rate ($\beta = -0.19$, $\chi^2(1) = 20.8$, $p < 0.001$), a

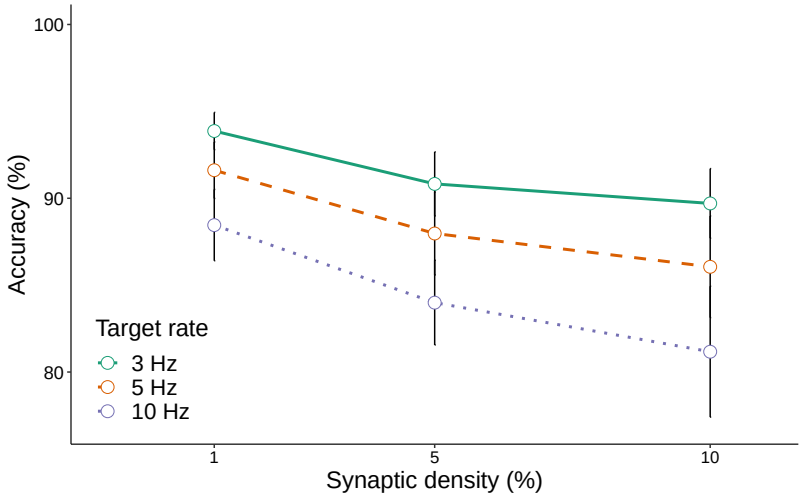

Fig. S1. Network comparison on sentence-final noun phrases for different target spike rates and synaptic connection densities. Error bars show 95% confidence intervals.

main effect of density ($\beta = -0.16$, $\chi^2(1) = 19.9$, $p < 0.001$) and no interaction between rate and density ($\beta = -0.01$, $\chi^2(1) = 1.6$, $p = 0.20$). Thus, in the absence of synaptic learning, neuronal working memory favors low spike rates and sparse network connectivity. This is explained by the fact that higher spike rates and denser connectivity lead to more dispersion and dilute the spatial coding of input words.

Decoding and evaluation. A decoder was calibrated by means of logistic regression to map network states onto target semantic roles (readout neurons). A readout is a projection of states onto the output dimensions. Typically, the projected variables are low-pass filtered spike trains. However, the filter can introduce a time constant which might distort estimation of the network's memory characteristics (3). Therefore, we used filters corresponding to neuronal membrane properties and the simplest way

to implement this protocol is to readout from membrane potentials. It is easily shown that this is equivalent to a real-time readout of filtered spike trains from the presynaptic neurons and hence every neuron in the network. Thus, at each point in time during online processing, reading out from membrane potentials is tantamount to reading out filtered spike trains. The decoder's objective was to maximize the conditional data log-likelihood

$$W_{\text{out}} = \arg\max_{W_{\text{out}}} \sum_k \ln P\left(Y^k | X^k, W_{\text{out}}\right) \quad [5]$$

where W_{out} is an $M \times N$ matrix of estimated readout weights for M labels and N neurons, X^k is the observed network state resulting from processing the kth word in the training data, and Y^k is the target semantic role for X^k (4). The log-likelihood has a unique maximum but there is no closed-form solution to finding W_{out}. To estimate these parameters, conjugate gradient descent was used with 100 iterations, and a regularization parameter of $\lambda = 0.05$. For discrete semantic role labels $Y \in \{y_1, \ldots, y_{M-1}\}$, the probabilities $P(Y = y_m|X)$ are then explicitly given as

$$P(Y = y_m|X) = \frac{\exp\left(w_{m0} + \sum_{i=1}^N w_{mi} X_i\right)}{1 + \sum_{j=1}^{M-1} \exp\left(w_{j0} + \sum_{i=1}^N w_{ji} X_i\right)} \quad [6]$$

with $P(Y = y_M|X) = 1 - \sum_{m=1}^{M-1} P(Y = y_m|X)$. Prior comparisons had shown that logistic regression was superior to ordinary least-squares, PCA regression, and Ridge regression.

In testing, the decoder assigned probabilities to the six semantic role labels AGENT, PATIENT, THEME, RECIPIENT, EXPERIENCER, GOAL and the ACTION label for each word in the input sentence. The role with the highest score was selected as the response and compared to the target label. Performance was quantified by means of a conservative kappa statistic κ for multinomial classification with $\kappa = (acc - rand)/(1 - rand)$ where acc was the raw labeling accuracy, and $rand$ the expected accuracy of a random classifier obtained through permutation of the semantic roles assigned by the decoder. Thus, the κ measure factors out what could be achieved by chance. We assessed role comprehension on all words in each sentence and, separately, on the sentence-final noun phrase only where memory demands on context-dependent processing were typically the highest. Model subjects were evaluated with 5-fold cross-validation on novel sets of test items. The decoder itself had no memory, it could only access traces of past input that persisted in the membrane dynamics. Hence, it provided an unbiased measure of the network's memory capacity.

Estimated readout weights were normally distributed around mean zero, with a few large, negative values (Figure 2e, main text). When these large weights were removed, accuracy declined ∼5% overall and ∼1% on sentence-final noun phrases, revealing a low degree of neuronal specialization for particular semantic roles.

Spike regularity and synchrony. The degree of network spiking regularity (Figure 2e, main text) was determined by the coefficient of variation of the inter-spike intervals (ISI).

$$\text{CV ISI} = \left\langle \frac{\sigma_i^{\text{ISI}}}{\mu_i^{\text{ISI}}} \right\rangle$$

where μ_i^{ISI} and σ_i^{ISI} denote the mean and standard deviation of the ISIs of neuron i and $\langle . \rangle$ the average over all neurons. The CV ISI measures spike train variability around the mean ISI. Regular spiking is indicated by a CV ISI close to 0, irregular spiking occurs for values close to 1. Values much larger than 1 indicate bursting activity. The degree of spike synchrony was quantified as the average correlation coefficient over 1,000 randomly sampled pairs of neurons

$$\text{CC} = \left\langle \frac{\text{Cov}(C_i, C_j)}{\text{Var}(C_i)\text{Var}(C_j)} \right\rangle$$

where C_i and C_j represent the number of spikes of neurons i and j, counted within successive temporal bins of 10 ms.

Model comparison. In Figure 2a (main text), the spiking network was compared to two other models, one without memory of sentence context, and one with perfect memory. In the *memoryless* model, target semantic roles were regressed directly on all the words in the training set, employing the same logistic regression classifier that was used to decode the spiking network states. The estimated model was then applied to predict the semantic roles of the words within the novel test sentences of the validation set (5-fold cross-validation). The *perfect memory* model was a back-off n-gram learner. It was first segmenting each input sentence S into all possible contiguous chunks of n words with $1 \leqslant n \leqslant N$, where N was the number of words in S. Each chunk was labeled with the target semantic role of the chunk-final word and these associations were stored in a hash table. Since chunks could be associated with several different roles (e.g., the word sequence *"the big cat"* could occur as an AGENT, EXPERIENCER or RECIPIENT in the training items), the learner also recorded the frequency of all chunk→role associations. When processing a test sentence $S^* = w_1 w_2 \ldots w_N$ from the validation set, the model sequentially stepped through each word w_k of S^* and looked for the largest lexical chunk $w_1 \ldots w_k$ in memory. If there was a match, it assigned the highest frequency role associated with this chunk to the word w_k. If $w_1 \ldots w_k$ was not matched in memory, the model would back-off to try $w_2 \ldots w_k$, and so forth. In case no contextual chunk could be found in memory, the most frequent semantic role attached to the single word w_k was selected as the response. Thus, the n-gram learner had perfect memory of all sentence contexts experienced in the training environment and used a frequency-based heuristic to deal with residual ambiguity.

Network queries. To test the binding of words to semantic roles in neuronal memory space (Figure 3, main text), the network was queried with semantic role labels for noun phrases after every input sentence. For instance, there were two possible queries for the item *"the mouse eats the apple"*, i.e., AGENT→*mouse* and PATIENT→*apple*, and one role was randomly selected per sentence. Role queries were encoded the same way as any other lexical item in the language, with an input duration of 100 ms. The network states that resulted from processing the query were recorded for 20,000 training items, and a readout was estimated to map these states onto target words in the lexicon that filled the queried role in each of the sentences (e.g., Query = AGENT, Target = *mouse* in the example above). Then, binding accuracy was tested on 5,000 novel and unique sentences from the validation set. The neuronal time constant τ_{sra} was set to 1 s and memory was cleared after each sentence to minimize interference. When the same lexical noun occurred twice in a sentence (so called *Problem of Two* sentences 5), a second, parallel readout was estimated to indicate the ordinal number of the target noun. For instance, in the test sentence *"the small boy gave a book to the big boy"*, the correct readout response to the query RECIPIENT would be: *boy* (lexical readout) and *2nd position* (ordinal readout).

Dimensionality reduction. In order to visualize neural trajectories in the section on Feature Binding (Figure 3, main text), neuronal states were recorded for every millisecond during sentence processing. The dimensionality of this data was reduced using the R implementation of t-Distributed Stochastic Neighbor Embedding (tSNE; 6) with 500 initial Principal Components for 1000 iterations, and parameter values of perplexity = 30 and theta = 0.5. tSNE is a non-linear method that attempts to faithfully preserve the distance between points in high-dimensional space based on a similarity metric.

Statistical analysis. Throughout, linear mixed effects models were applied to logit-transformations of the various dependent measures described in the main text. Unless stated otherwise, categorical predictors were effect-coded, numerical ones were scaled and centered. All statistical models included the maximal random effects structure that still converged (7) and p-values were obtained through likelihood-ratio tests (χ^2). These mixed models were run in R using the lme4 package (v. 1.1–17) with Nelder-Mead optimizer.

Modeling parameters. We summarize network and simulation parameters in Table S2 below to enable replication. Values refer to the model detailed in Figure 2a (main text), parametric variations (e.g., neuronal time constants and excitability) are described in the main text.

References

1. Dayan P, Abbott LF (2005) *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems.* (The MIT Press, Cambridge, MA).
2. Goldberg A (2006) *Constructions at Work: The Nature of Generalization in Language.* (Oxford University Press).
3. van den Broek D, et al. (2017) The best spike filter kernel is a neuron (Extended abstract *Cognitive Computational Neuroscience* conference, NYC, September 6–8).
4. Mitchell T (2017) Generative and discriminative classifiers: Naive Bayes and logistic regression.
5. Jackendoff R (2002) *Foundations of Language: Brain, Meaning, Grammar, Evolution.* (Oxford University Press).
6. van der Maaten LJP, Hinton GE (2008) Visualizing data using t-SNE. *Journal of Machine Learning Research* 9:2579–2605.
7. Barr DJ, Levy R, Scheepers C, Tily HJ (2013) Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language* 68:255–278.

Table S2. Network and simulation parameters

Model summary	
Populations	Three main: Excitatory (E), inhibitory (I), external (X)
Connectivity	Feed-forward (acyclic graph)
Neuron model	Leaky integrate-and-fire, fixed voltage threshold, spike-rate adaptation, relative refractory period
Synapse model	Current-based, exponentially decaying post-synaptic currents
Word input	Step-current, delivered to random, non-disjoint subpopulations $S_k \subset E \times I$ for each word k
State variable	Neuronal membrane potentials
Network decoder	Logistic regression classifier with conjugate gradient descent

Network		
Parameter	Value	Description
N^E	800	Number of excitatory neurons
N^I	200	Number of inhibitory neurons
N^X	50	Word-specific input population size
ϵ	1%	Internal connection probability

Neuron and synapse		
Parameter	Value	Description
τ_m	10 ms	Membrane time constant
R_m	15 mΩ	Membrane resistance
V_{th}	–54 mV	Fixed firing threshold
V_{rest}	–70 mV	Resting membrane potential
τ_{ref}	2 ms	Time-constant refractory conductance
Δg_{ref}	200 nS	Increase in refractory conductance per spike
τ_{sra}	0 ms–1 s	Time-constant spike-rate adaptation conductance (see main text)
Δg_{sra}	4 nS–500 nS	Increase in spike-rate conductance per spike (see main text)
E_K	–80 mV	Spike-rate adaptation reversal potential
τ_{syn}	10 ms	Decay time constant for synaptic currents

Simulation		
Parameter	Value	Description
dt	0.2 ms	Discretization time step
S_{rate}	200 Hz	Network state sampling rate
T_{rate}	5 Hz	Target network activity rate
Inp_{app}	3e-9	Initial input current scaling factor (tunable)
Res_{app}	4e-9	Initial internal current scaling factor (tunable)
trainSize	10,000	Number of input words
testSize	2,500	Number of tested words