A neuromorphic network for generic multivariate data classification

Michael Schmuker[a,b,1], Thomas Pfeil[c], and Martin Paul Nawrot[a,b]

[a]Neuroinformatics and Theoretical Neuroscience, Institute for Biology, Department of Biology Chemistry and Pharmacy, Freie Universität Berlin, 14195 Berlin, Germany; [b]Bernstein Center for Computational Neuroscience Berlin, 10119 Berlin, Germany; and [c]Kirchhoff-Institute for Physics, Heidelberg University, 69120 Heidelberg, Germany

Edited by Terrence J. Sejnowski, Salk Institute for Biological Studies, La Jolla, CA, and approved December 23, 2013 (received for review February 20, 2013)

Computational neuroscience has uncovered a number of computational principles used by nervous systems. At the same time, neuromorphic hardware has matured to a state where fast silicon implementations of complex neural networks have become feasible. En route to future technical applications of neuromorphic computing the current challenge lies in the identification and implementation of functional brain algorithms. Taking inspiration from the olfactory system of insects, we constructed a spiking neural network for the classification of multivariate data, a common problem in signal and data analysis. In this model, real-valued multivariate data are converted into spike trains using "virtual receptors" (VRs). Their output is processed by lateral inhibition and drives a winner-take-all circuit that supports supervised learning. VRs are conveniently implemented in software, whereas the lateral inhibition and classification stages run on accelerated neuromorphic hardware. When trained and tested on real-world datasets, we find that the classification performance is on par with a naïve Bayes classifier. An analysis of the network dynamics shows that stable decisions in output neuron populations are reached within less than 100 ms of biological time, matching the time-to-decision reported for the insect nervous system. Through leveraging a population code, the network tolerates the variability of neuronal transfer functions and trial-to-trial variation that is inevitably present on the hardware system. Our work provides a proof of principle for the successful implementation of a functional spiking neural network on a configurable neuromorphic hardware system that can readily be applied to real-world computing problems.

bioinspired computing | spiking networks | machine learning | multivariate classification

The remarkable sensory and behavioral capabilities of all higher organisms are provided by the network of neurons in their nervous systems. The computing principles of the brain have inspired many powerful algorithms for data processing, most importantly the perceptron and, building on top of that, multilayer artificial neural networks, which are being applied with great success to various data analysis problems (1). Although these networks operate with continuous values, computation in biological neuronal networks relies on the exchange of action potentials, or "spikes."

Simulating networks of spiking neurons with software tools is computationally intensive, imposing limits to the duration of simulations and maximum network size. To overcome this limitation, several groups around the world have started to develop hardware realizations of spiking neuron models and neuronal networks (2–10) for studying the behavior of biological networks (11). The approach of the *Spikey* hardware system used in the present study is to enable high-throughput network simulations by speeding up computation by a factor of 10^4 compared with biological real time (12, 13). It has been developed as a reconfigurable multineuron computing substrate supporting a wide range of network topologies (14).

In addition to providing faster tools for neurosimulation, high-throughput spiking network computation in hardware offers the possibility of using spiking networks to solve real-world computational problems. The massive parallelism is a potential advantage over conventional computing when processing large amounts of data in parallel. However, conventional algorithms are often difficult to implement using spiking networks for which many neuromorphic hardware substrates are designed. Novel algorithms have to be designed that embrace the inherent parallelism of a brain-like computing architecture.

A common problem in data analysis is classification of multivariate data. Many problems in artificial intelligence relate to classification in some way or the other, such as object recognition or decision making. It is the basis for data mining and, as such, has widespread applications in industry. We interact with classification systems in many aspects of daily life, for example in the form of Web shop recommendations, driver assistance systems, or when sending a letter with a handwritten address that is deciphered automatically in the post office.

In this work, we present a neuromorphic network for supervised classification of multivariate data. We implemented the spiking network part on a neuromorphic hardware system. Using a range of datasets, we demonstrate how the classifier network supports nonlinear separation through encoding by virtual receptors, whereas lateral inhibition transforms the input data into a sparser encoding that is better suited for learning.

Results

We first outline our spiking neural network design and show examples of the network activity during operation in supervised classification of multivariate data. Then we analyze the temporal dynamics of the classification process and compare the network classification performance against the performance of a naïve Bayes (NB) classifier. We show that the network tolerates the

Significance

One primary goal of computational neuroscience is to uncover fundamental principles of computations that are performed by the brain. In our work, we took direct inspiration from biology for a technical application of brain-like processing. We make use of neuromorphic hardware—electronic versions of neurons and synapses on a microchip—to implement a neural network inspired by the sensory processing architecture of the nervous system of insects. We demonstrate that this neuromorphic network achieves classification of generic multidimensional data—a widespread problem with many technical applications. Our work provides a proof of concept for using analog electronic microcircuits mimicking neurons to perform real-world computing tasks, and it describes the benefits and challenges of the neuromorphic approach.

Author contributions: M.S. designed research; M.S. and T.P. performed research; M.S. analyzed data; and M.S., T.P., and M.P.N. wrote the paper.

The authors declare no conflict of interest.

This article is a PNAS Direct Submission.

Freely available online through the PNAS open access option.

[1]To whom correspondence should be addressed. E-mail: m.schmuker@fu-berlin.de.

This article contains supporting information online at www.pnas.org/lookup/suppl/doi:10.1073/pnas.1303053111/-/DCSupplemental.

neuronal variability that is present on the hardware through leveraging a population code. Finally, we demonstrate that the network design is generic and can be applied, without reparameterization, to different multivariate problems. We used the PyNN software package for network implementations on the *Spikey* neuromorphic hardware system (15, 16). For simplicity, all temporal parameters are specified in the biological time domain throughout this study. The actual time values referring to the spiking network execution on the hardware are 10^4 times smaller due to the speedup factor of the accelerated *Spikey* system.

A Spiking Network for Supervised Learning of Data Classification. In multivariate classification problems, data are typically organized as observations of a number of variables arranged in a matrix $\mathbf{X}$, with rows corresponding to observations and columns to real-valued features. Each observation has an associated class label stored in a binary matrix $\mathbf{Y}$, with $Y_{i,j} = 1$ if the observation i belongs to class j. The aim is to find a mapping $\mathbf{A}$ such that $argmax(\mathbf{X}\cdot\mathbf{A}) = \mathbf{Y}$, with *argmax* returning 1 for the maximal value in each row and 0 otherwise. The classes of new observations $\mathbf{X}'$ can then be predicted by applying the transformation $argmax(\mathbf{X}'\cdot\mathbf{A}) = \mathbf{Y}'$. The architecture of the insect olfactory system maps well on this task (17–19).

We designed a classifier network that approximates the basic blueprint of the insect olfactory system, without claiming to be an exact model of the biological reality. Its three-stage architecture consists of an input layer, a decorrelation layer, and an association layer (Fig. 1*A*). We provide a detailed description in *SI Materials and Methods* and a parameter list in Table S1.

In the input layer, real-valued multidimensional data are transformed into bounded and positive firing rates. The data enter the network via ensembles of receptor neurons (RNs). RNs fire spikes at specified rates which are computed from the real-valued input data using "virtual receptors" (VRs) (17) (see also *SI Materials and Methods, VRs* for details). A VR corresponds to the center of a linear (cone-shaped) radial basis function in feature space. The magnitude of its response to a data point (a "stimulus") depends on the distance between the VR and the stimulus. Hence, the VR response is large for small distances between stimulus and receptor, and vice versa. VRs are placed in data space in a self-organized manner using the neural gas algorithm (20).

RN ensembles project onto projection neurons (PNs) in the decorrelation layer, which are grouped in ensembles that represent the so-called glomeruli in the insect antennal lobe. Each RN ensemble targets one glomerulus, which thus receives excitatory input that represents the activation of one VR. The PNs project to local inhibitory neurons (LNs), which laterally inhibit other glomeruli. Moderate lateral inhibition between glomeruli reduces correlations between the variables they represent without degrading the encoding to a fully orthogonalized representation (14, 21–23).

The output of the decorrelation layer is projected to the association layer, in which supervised learning for data classification is realized. Association neurons (ANs) are grouped in as many populations as there are classes in the dataset. Each population in the association layer is assigned one label from the dataset (for example, "choice A" and "choice B" as indicated in Fig. 1*A*). The AN populations project onto associated populations of inhibitory neurons. The strong inhibition between AN populations induces a soft winner-take-all (sWTA) behavior in the association layer. The synaptic weights from PNs to ANs are initialized randomly. An activity pattern presented to the network will thus by chance deliver more input to one of the "choice" populations than to the others, resulting in higher firing rate of that population (the "winner population"). If the label of the winner population matches the one of the stimulus, the network performed a correct classification. We used a 50% connection probability from RNs to PNs, from PNs to LNs and to ANs, and from excitatory to inhibitory neurons in the sWTA circuit (Table S1). Inhibitory populations are fully connected to excitatory populations.

We train the network in a supervised fashion by presenting stimuli with known class labels. If classification was correct, active synapses from PNs to the winner population are potentiated. If classification was incorrect, active synapses are depressed (see *Materials and Methods* for a detailed description of the algorithm). This learning rule is derived from the delta rule for perceptron training (24, 25). Network training leads to an optimized set of synaptic weights for classification of the dataset. After successful training, the winner population in the association layer indicates which class a stimulus belongs to, and it can predict the class adherence for unseen stimuli.

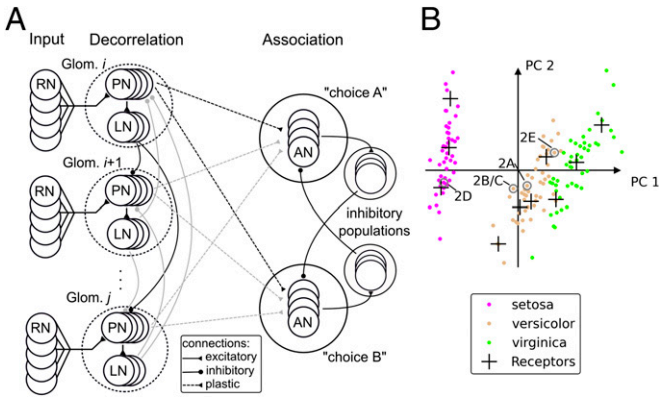

Fig. 1. Network architecture and real-world classification problem. (*A*) Schematic of the generic network. AN, association neuron; LN, local inhibitory neuron; PN, projection neuron; RN, receptor neuron. (*B*) Projection of the complete iris dataset to the first two principal components (97.7% variance explained) and locations of 10 VRs. Annotations refer to data points presented in Fig. 2.

Application of the Neuromorphic Classifier Network to a Real-World Dataset. We implemented the classifier network on the *Spikey* hardware system, which has been described in detail previously (14). We assessed its performance using Fisher's iris dataset (26) as a benchmark. The iris dataset is a four-dimensional dataset describing features of the blossom leaves for three species of the iris flower, *Iris setosa*, *Iris virginica*, and *Iris versicolor*. This dataset is particularly well suited for this study for two reasons. First, it contains only 150 data points, which makes rapid prototyping of the network feasible. Second, the constellation of the data points allows for a fine-grained interpretation of the classifier capabilities: The *I. setosa* class is well separated from the other two, making learning the classification boundary easy (Fig. 1*B*). Separation of the *I. virginica* and *I. versicolor* classes is more difficult because they partly overlap in feature space. Classifier performance on this separation indicates how well the classifier copes with more challenging problems. Separating such overlapping data classes typically requires supervised learning methods, because there is no clear "gap" between the classes in data space that would allow an unsupervised method to detect class boundaries.

We used 10 VRs to encode the dataset. They represented the data points by firing intensities, which were used to generate the RN spike trains in the input layer using a gamma point process. The number of VRs determines the number of glomeruli, and thus the total number of neurons required for the network. The specific choice of 10 VRs was a compromise between choosing a number as high as possible while staying within the maximal neuron count of 192 on the present neuromorphic hardware system (see *SI Results, Number of Glomeruli* for a detailed explanation).

The spiking activity of the classifier network is depicted in Fig. 2. Fig. 2*A* shows the activity of all neurons in all three layers in the beginning of the training phase when stimulated with the data point annotated as "2A" in Fig. 1*B*. The activity pattern across the

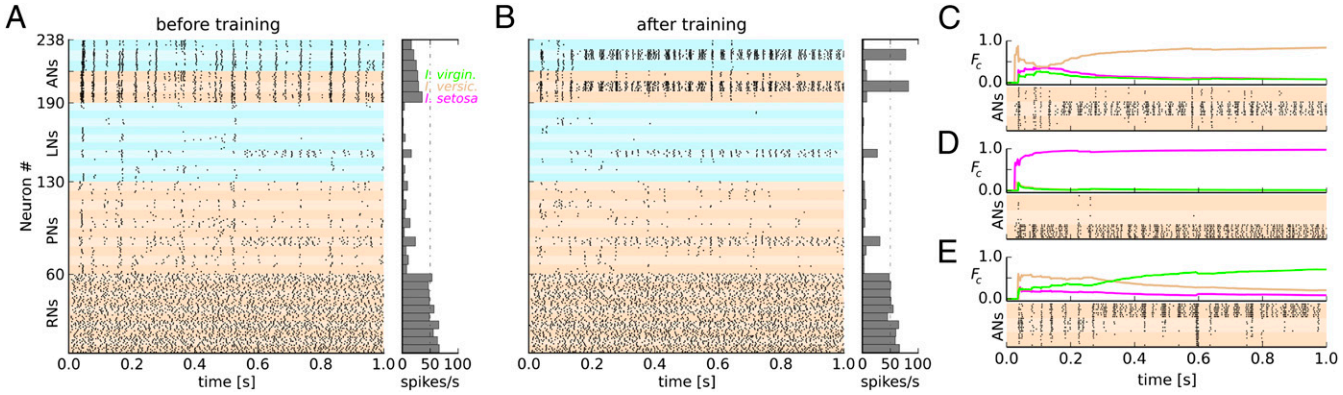

Fig. 2. Network activity during stimulus presentation before and after training. (*A*) Untrained network. Spike raster display and population spike count of all neuron populations in the network in response to the presentation of one data point from *I. versicolor* as indicated in Fig. 1*B*. Distinct neuronal populations are labeled by alternating color saturation. Warm/cool color, excitatory/inhibitory population. The stimulus was applied at time $t = 0$ s for the duration of 1 s. (*B*) Network activity after training during 1 s of stimulation with a test sample from *I. versicolor* as labeled in Fig. 1*B*. (*C–E*) Spiking activity and temporal evolution of $F_c(t)$ (Eq. **1**) for all three excitatory AN populations in response to three different data samples as labeled in Fig. 1*B*. Color of $F_c(t)$ trace indicates the *Iris* species associated to the respective AN population (color code as in Fig. 1*B*). Only spiking activity from excitatory ANs is shown.

RN population expresses the activation level of the VRs. PNs exhibited sparser activity compared with RNs, largely due to lateral inhibition from LNs. All three populations of ANs responded with approximately the same intensity because the weights from decorrelation layer to the association layer are initially random. Due to the strong lateral inhibition in the association layer, all three populations showed synchronized and oscillating activity. The population associated to the *I. setosa* class emitted a slightly higher number of spikes than the others during the 1-s stimulus presentation. Because the presented data point belonged to the *I. versicolor* class, this association was wrong, and hence the weights of synapses targeting the *I. setosa* population were reduced after this presentation as part of the training procedure. During the training phase, 80% of all data points were presented and the weights adjusted according to the learning rule after each presentation. Fig. 2*B* shows network activity in response to a sample from the *I. versicolor* class in the test phase. The AN population activity rapidly converged to a representation that indicated the correct association after only a few spikes and maintained this state throughout the duration of the stimulus presentation.

To assess the convergence of the association layer activity to a winner population, we calculated the cumulative fraction of spikes $F_c(t)$ from each population c at time t as follows:

$$F_c(t) = \frac{I_c(t)}{I_{all}(t)}, \qquad [1]$$

where $I_c(t)$ indicates the number of spikes emitted by population c within the interval $(0,t]$, whereas $I_{all}(t)$ refers to the total number of spikes from all AN populations. $F_c(t)$ thus reflects, at each time point t, the integrated activity of one AN population compared with the total AN activity up to that point in time. Fig. 2*C* shows the resulting population dynamics for the example in Fig. 2*B*, together with spike trains in the AN populations. For this data point, it took about 150–200 ms before the network activity converged toward a stable state with the *I. versicolor* population having the highest activity, indicating the correct association. This convergence happened faster for data samples from the well-separated *I. setosa* class (Fig. 2*D*). In a third example from the *I. versicolor* class close to the class boundary, the network first showed a slightly higher firing rate for the correct class, but eventually converged to a wrong decision (Fig. 2*E*).

Time to Decision and Classification Performance. We used Gorodkin's K-category correlation coefficient R_K to measure classification performance (27) (see *SI Materials and Methods, Evaluation of Classifier Performance* for a formal definition and rationale behind our preference of R_K over more frequently used performance measures like "percent correct").

In our network, each data class is represented by a different AN population. For each presentation of a test stimulus, the population that generated the most spikes within a certain observation time window is the winner population, indicating either a correct or incorrect classification. We computed the R_K across all test samples in a time-resolved manner by varying the time t after stimulus onset that was used to count AN spikes. As shown in Fig. 3*A*, R_K rapidly approaches a stable maximum indicating a time-to-decision of less than 100 ms in biological time corresponding to 10 μs of real time with the *Spikey* chip.

We next compared the absolute classification performance with that of the NB classifier, which we use here as a benchmark for conventional machine learning methods. We chose NB because it is a linear classifier without any free parameters, so it delivers robust classification without the need for parameter tuning. We evaluated R_K across the entire 1-s stimulus presentation. For the iris dataset, the NB classifier yields an average R_K of 0.89 ($P^{20} = 0.88$, $P^{80} = 0.90$) in 50 repetitions of fivefold cross-validation and thus slightly outperforms the neuromorphic classifier with $R_K = 0.87$ ($P^{20} = 0.85$, $P^{80} = 0.89$, 50 repetitions) (Fig. 3*B*). The performance evaluation is described in detail in *SI Materials and Methods*.

For a thorough examination of the classification outcome, we compared the confusion matrix produced by the classifier network (Table S2 and *SI Results, Per-Class Classification Performance*). The classifier only produced errors on the more challenging separation of *I. versicolor* and *I. virginica*, whereas it always succeeded to

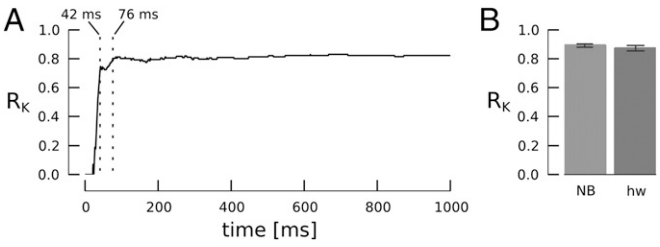

Fig. 3. Classification performance. (*A*) R_K obtained for decision time points between 0 and 1,000 ms from a single cross-validation run. The vertical dotted lines indicate when R_K first exceeds values of 0.7 (42 ms) and 0.8 (76 ms). (*B*) Classification performance of the hardware classifier network (hw) at decision time of 1 s compared with a naïve Bayes classifier (NB) in fivefold cross-validation. Error bars indicate 20th/80th percentile from 10 repetitions.

separate *I. setosa*. This observation indicates that the classifier network is capable of delivering reliable classification not only of well-separable data, but also in cases where samples from different classes overlap in feature space.

Tolerance Against Neuronal Variability. The analog circuits used to represent neurons in the *Spikey* system exhibit inherent variability that the classifier network must tolerate to be useful in practice. Two sources of variability on the hardware system can be distinguished: "temporal noise" and "fixed-pattern noise." Temporal noise (including thermal noise and other sources of stochastic variability) affects the circuits on short timescales in an unpredictable fashion. In contrast, fixed-pattern noise is caused by device mismatch. Device mismatch describes the deviance of an electronic component from its specification due to inevitable variations in the manufacturing process. The variations of neuron parameters due to device mismatch occur on much slower timescales and can be regarded as constant for our use case. They introduce heterogeneity across all analog components—neurons and synapses—according to a fixed pattern (hence the term "fixed-pattern noise"). The individual variation can be measured and calibrated for. The integrated development environment of the *Spikey* system contains calibration methods that reduce the amount of fixed-pattern variability. However, such generic calibration methods cannot account for all network configurations in an efficient manner, because calibration at the neuron level does not take into account network effects. This is particularly relevant for the *Spikey* system, which was designed to accommodate a wide variety of network topologies (14). In our case, the fixed-pattern variation that remains after built-in calibration manifests itself in variability of the neurons' transfer functions that relate input rate to output rate. Both maximal output rate and slope of the transfer functions varied considerably across PNs and LNs (Fig. 4*A*).

Due to its stochastic nature, temporal noise cannot be avoided by systematic measures such as calibration of synaptic weights. We quantified the variation in spike count caused by temporal noise by measuring the variability of the spike count in all 192 hardware neurons across 50 repetitions with identical stimuli. For this purpose, we generated input spike trains only once and used them repeatedly as input to all 192 neurons ("frozen input"). We used six gamma processes of order five and mean rate of 25 spikes/s to mimic the inputs that PNs receive in the classifier network. We adjusted the weights of the neurons to yield a mean output frequency of 25.4 spikes/s. The neurons exhibited moderate trial-to-trial variability under these conditions. Fig. 4*B* shows the distribution of spike counts for one exemplary neuron that produced 25.3 spikes on average, with a variance of 1.0. This amount of variability is also reflected when considering the total population (Fig. 4*C*). On average, the individual spike trains from the same neuron varied with a Fano factor (28) of 0.083, which is smaller than the variability inherent to the gamma process used for the generation of the RN spike trains ($\gamma = 5$, Fano factor = 0.2). Thus, the trial-to-trial variability due to temporal noise intrinsic to the neuromorphic hardware is small compared with those variations imposed by the biologically realistic stochastic generation of input spike trains.

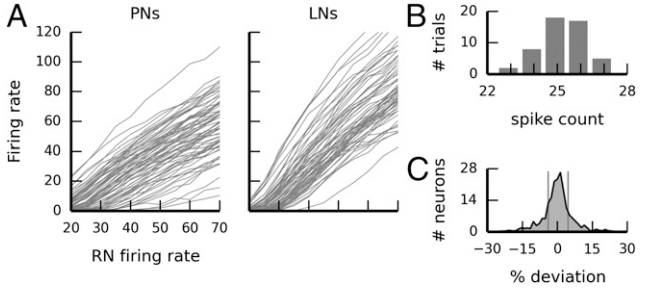

Fig. 4. Neuromorphic hardware variability. (*A*) Variability of transfer functions across all PNs and LNs on the hardware chip. Each line corresponds to one neuron. (*B*) Hardware trial-to-trial variability: Histogram of spike counts for one example neuron across 50 repeated stimulations with identical "frozen input" of 1-s duration (average spike count, 25.3; variance, 1.0). (*C*) Histogram of per-neuron spike counts relative to the individual average spike count emitted by each of the 192 neurons across 50 repeated identical 1-s stimulations. The vertical lines indicate 20th/80th percentile ($P^{20} = 3.99$, $P^{80} = 4.58$).

The classifier network achieved the reported performance despite transfer function variability caused by fixed-pattern noise and trial-to-trial variability caused by temporal noise and by the stochasticity of the input. This robustness is the result of considerable efforts to optimize network topology. Essentially, the key to achieve robustness in our network was to leverage population coding. Two network properties proved essential to ensure a valid population code. First, synchronization of neurons within a population should be avoided because it violates the rate code assumption of independent neurons within each population. We achieved this by sparsifying the input to individual neurons, i.e., using 50% connection probability instead of full connectivity. Second, population sizes must be sufficiently large to reduce the variance of the population transfer function. We provide a detailed explanation of how these properties affect network operation in *SI Results* (Figs. S1 and S2, and *SI Results, Network Optimization for Robustness Against Neuronal Variability*).

General Applicability to Other Datasets. As a demonstration for the ability of the network to solve nonlinear problems, we applied the network to classification of a 2D "Ring" dataset. This simple dataset consists of two classes, one class situated in a cluster centered at the origin and a second class surrounding it (Fig. 5*A*). It has skewed class proportions with sevenfold more data points in the surround than in the center class. In addition, the arrangement of data points requires a nonlinear separation between the center and surround classes. Our network achieves this separation through the VR trick: By using 10 VRs to represent a 2D dataset, we transform the data into a higher-dimensional space in which linear separation is possible. The classifier network running on the *Spikey* system achieved an average performance of $R_K = 0.96$ on the Ring dataset (NB: $R_K = 0.98$; Fig. 5*C*, *Left*).

The Mixed National Institute of Standards and Technology (MNIST) database is a commonly used high-dimensional benchmark problem with practical relevance (http://yann.lecun.com/exdb/mnist/). The database contains images of handwritten digits from 0 to 9, digitized to 28 × 28 pixels. Hence, each observation has 28·28 = 768 dimensions. The dataset is divided into a training and a test set to enable reproducible benchmarking. We picked a subset of this dataset consisting of the digits "5" and "7," using 2,000 samples from the training set and 1,920 samples from the test set (Fig. 5*B*). On the MNIST dataset, the spiking network outperformed the NB classifier by a large margin (hardware network: mean $R_K = 0.94$; NB: 0.82; Fig. 5*C*, *Right*). Interestingly, when training the NB classifier on the spike counts produced by PNs in the network (that is, after the lateral inhibition stage in the decorrelation layer), its performance increases to similar levels as obtained with the classifier network (mean $R_K = 0.96$). This observation is in line with a previous study which demonstrated that lateral inhibition increases classifier performance on a 184-dimensional odor dataset (17).

The reason for this effect lies in the fact that lateral inhibition transforms the broad, overlapping receptive fields of VRs (and in consequence RNs) into more localized representations of input space. In other words, receptive fields of PNs are narrower than those of VRs, and they overlap to a lesser degree. As a result, PN activity is also sparser than VR activity, that is, only few PN populations respond to a particular stimulus. This behavior can be observed, for example, when comparing the spike counts of RNs and PNs in Fig. 2 *A* and *B*. Sparser activity and more local receptive fields simplify the training process, because it

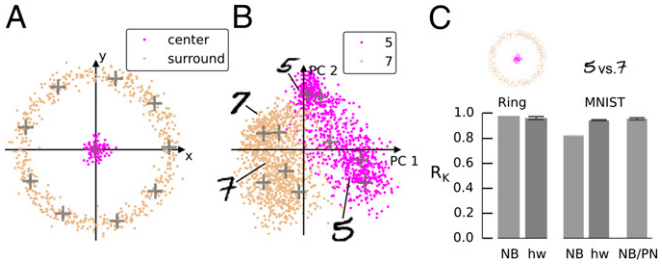

Fig. 5. Application to generic classification problems. (*A*) Ring dataset (training samples) and locations of 10 VRs. (*B*) First two principal components of MNIST digits "5" (893 samples) and "7" (1,107 samples) from the training set (2,000 samples total) and VR locations. (*C*) Performance comparison of the naïve Bayes classifier (NB) vs. the classifier network on hardware (hw), and the NB classifier trained on PN firing rates (NB/PN). Error bars indicate 20th/80th percentile from 10 repetitions. When trained on the VR responses, the NB performance is deterministic for these datasets because the training and test datasets are fixed. For NB/PN, we extracted the PN firing rates from the 10 repeated network runs that we used to assess the spiking network. Hence, the NB classification performance varies.

becomes easier to identify the input units that are relevant to discriminate data points in a particular region of input space (the "credit assignment problem"). We explain the soft partitioning effect provided by lateral inhibition in detail in *SI Results, Effect of Lateral Inhibition on Classification Performance*, including an illustrated example (Fig. S3).

Speed Considerations. The major advantage in using accelerated neuromorphic hardware for spiking neuronal simulations is its potentially fast execution time. On the neuromorphic hardware system used in this study, simulations run with a speedup factor of 10^4. Hence, presenting all 150 iris data points for 1 s (biological time) each to the hardware network takes 150 s/10^4 = 15 ms pure network run time. Practical applications require data transfer for spikes and synaptic weights to and from the system as well as the parameterization of the hardware network, which adds to the pure network run time (for details, see *SI Materials and Methods* and Fig. S4). These factors depend on the efficiency of the software interface for the hardware system. Because we are working with a prototype setup, its interface is under constant development and improvement. At the time of writing, the hardware system effectively achieved an overall 13-fold speedup compared with biological real time. We want to stress that this number may improve as the software interface is continuously optimized.

Discussion

We demonstrated the implementation of a spiking neuronal network for classification of multidimensional data on a neuromorphic hardware system. The network is capable of separating data in a nonlinear fashion through encoding by VRs. The transformation by lateral inhibition increases classification performance. It performed robustly in the presence of stochastic trial-to-trial variability inherent to the hardware system. The network is not restricted to any specific kind of data, but is capable of classifying arbitrary real-valued, multidimensional data, and hence universally suited for all kinds of classification tasks. It achieved performance values comparable to a standard machine learning classifier, which points out the network's wide applicability to real-world problems. The present network implementation is a proof of concept that can serve as a building block for classifier tasks on neuromorphic hardware. Together with the high speedup factor of the neuromorphic hardware system, our universal classification network is an important step toward high-performance neurocomputing.

We verified the capability of our implementation of VRs to transform data into a higher-dimensional space in which linear separation is possible. The network we presented contains a linear classifier, with the additional constraint that the separating hyperplane must pass through the origin (29). As such, it is limited to separating linear problems. We overcame this limitation through the VR approach, which provides a higher-dimensional representation of the data. Our results on the MNIST dataset point out that the lateral inhibition step is crucial for successful classification of real-world, high-dimensional datasets. Although more complex machine learning algorithms like support vector machines or restricted Boltzmann machines may allow for better classification performance directly on the VR data, the strength of our approach lies in the simplicity of a linear classifier combined with appropriate filtering of input data through the lateral inhibition step, which is very efficiently carried out in a massively parallel neuromorphic hardware network.

Lateral inhibition provides a soft partitioning of input space that facilitates classifier training. Note that this circumstance also points out a limitation of the presented classifier network, because class boundaries in data space can only be optimally represented if they coincide with partition borders. A straightforward way to deal with this problem is to increase the number of VRs and glomeruli, resulting in a more fine-grained partitioning of data space. Such an approach will be possible using emerging large-scale neuromorphic hardware systems supporting tens of thousands of neurons (8, 30).

VRs depend on a self-organizing process that is trained in data space. A particularly interesting prospect is to implement this process on the neuronal substrate. Spiking self-organizing maps have been described in the literature (31–33), suggesting that, in principle, it is possible to implement a self-organizing process on a neuromorphic hardware system. However, the learning rules used in these studies would require sophisticated control logic, which makes it difficult to implement them on the *Spikey* system. A more straightforward and mathematically well-founded approach has recently been put forward by Nessler et al. (34). They suggested a probabilistic, self-organizing mechanism to learn prototypes in feature space using spike timing-dependent plasticity (STDP) and a winner-take-all circuit, which is suited to represent the VR encoding. An integrated implementation of this encoding together with the classifier network we present here will likely require a much higher neuron count and more flexible plasticity mechanisms compared with what is available on the *Spikey* system (13). On-chip implementations may become feasible considering the BrainScaleS wafer-scale hardware system that extends the number of available neurons by up to several orders of magnitude and provides more sophisticated plasticity mechanisms (35, 36). In that system, multiple identical neuromorphic modules may be implemented on a single silicon wafer and communicate through high-bandwidth connections. Moreover, advanced control logic for on-chip implementation of elaborate STDP rules is under development (36), which is designed to be compatible with the self-organized prototype learning mechanisms described by Nessler et al. (34). In addition, the deterministic connectivity structure of the glomerular classifier network presented here facilitates splitting the network across different neuromorphic modules. The increased neuron count available in a large-scale system would allow for a larger number of VRs to solve more complex problems and enables scaling the network to larger population sizes to support robustness against noise.

Analysis of the dynamic network activation in response to the onset of a stimulus presentation revealed a fast decision time where the average performance reached its maximum within less than 100 ms in biological time (Fig. 3*A*). This is in good agreement with recent measurements in insects. In the honeybee, a prominent animal model for studying learning and memory, it was shown that the encoding of the identity of an olfactory stimulus at the level of PNs evolved rapidly within tens of milliseconds (37, 38). Neuronal populations at the output of the mushroom body encode odor–reward associations. These neuronal populations fulfill a similar function like the ANs in our network. In a classical conditioning paradigm, they indicated the

classification of the conditioned stimulus (an odor that was previously paired with a sugar reward) within less than 200 ms (39).

Our network proved to be robust against neuronal variability, which is an important factor in the design of neuromorphic algorithms. Biological neuronal networks face a similar challenge. The study of neuronal variance is an integral part of today's neuroscience ever since the seminal study by Mainen and Sejnowski (40). Many neural properties are stochastic in nature, like neurotransmitter release or spike initiation, so a certain amount of variability is inevitable in biological neuronal networks (41). In the same vein, the analog nature of the circuits in the hardware enables the massive speedup and integration density, but unavoidably entails variability. In our case, we achieved tolerance against variability by using a population code. Generally, accelerated analog neurocomputing requires models that can cope with and, ideally, make use of variability. The design of these models will benefit greatly from a deep understanding of biological circuits, interpreted in the light of variability. Likewise, creating functional networks on an analog neuromorphic substrate provides insight into critical properties that networks must possess to operate under noisy conditions.

Materials and Methods

Stimuli were presented to the classifier network in a sequential manner. For each stimulus i, the corresponding feature vector $\mathbf{x}_i$ was obtained from the observation matrix $\mathbf{X}$, converted into a firing-rate presentation with VRs, from which spike trains were generated by a gamma point process (42). Each stimulus was presented for 1 s of biological time. Synaptic weights that fulfilled a Hebbian eligibility constraint were updated after each stimulus presentation. A synaptic weight was eligible for updating if the target neuron was a member of the winner population, and if the spike count emitted by the presynaptic neuron during the previous stimulus presentation exceeded a threshold (fixed to 35 spikes in the 1-s stimulus interval). Eligible synapses were potentiated by a fixed amount if classification was correct, or depressed by a fixed amount if classification was incorrect. A formal description of the training algorithm is available in *SI Materials and Methods, Network Training and Supervised Learning Rule*.

Network training was implemented in an interactive chip-in-the-loop fashion: Stimuli were processed by the network on the chip. After each stimulus, the network response was evaluated on the host computer where the weight changes are calculated. The network was then reconfigured and the next stimulus presented.

ACKNOWLEDGMENTS. We are deeply indebted to Daniel Brüderle for his support in using the *Spikey* system, and we thank Eric Müller for technical assistance and Mihai A. Petrovici and Emre Neftci for comments and discussion. We also thank the two anonymous reviewers for insightful comments that stimulated new experiments that led to great improvements of the manuscript. This work was supported by Deutsche Forschungsgemeinschaft Grants SCHM2474/1-1 and 1-2 (to M.S.), a grant from the German Ministry of Education and Research [Bundesministerium für Bildung und Forschung (BMBF)], Grant 01GQ1001D [to M.S. (Bernstein Center for Computational Neuroscience Berlin, Project A7) and M.P.N. (Project A9)], and a grant from BMBF [Bernstein Focus Neuronal Learning: Insect Inspired Robots; Grant 01GQ0941 (to M.P.N.)]. T.P. has received funding by the European Union Seventh Framework Programme under Grant 243914 (Brain-i-Nets) and Grant 269921 (BrainScaleS).

1. Hertz JA, Krogh AS, Palmer RG (1991) *Introduction to the Theory of Neural Computation* (Addison-Wesley, Redwood City, CA), Vol 1.
2. Mahowald M, Douglas RJ (1991) A silicon neuron. *Nature* 354(6354):515–518.
3. Indiveri G, Chicca E, Douglas R (2009) Artificial cognitive systems: From VLSI networks of spiking neurons to neuromorphic cognition. *Cognit Comput* 1:119–127.
4. Yu T, Sejnowski TJ, Cauwenberghs G (2011) Biophysical neural spiking, bursting, and excitability dynamics in reconfigurable analog VLSI. *IEEE Trans Biomed Circuits Syst* 5(5):420–429.
5. Indiveri G, et al. (2011) Neuromorphic silicon neuron circuits. *Front Neurosci* 5:73.
6. Renaud S, et al. (2010) PAX: A mixed hardware/software simulation platform for spiking neural networks. *Neural Netw* 23(7):905–916.
7. Brüderle D, et al. (2011) A comprehensive workflow for general-purpose neural modeling with highly configurable neuromorphic hardware systems. *Biol Cybern* 104(4-5):263–296.
8. Furber SB, et al. (2013) Overview of the SpiNNaker System Architecture. *IEEE Trans Comput* 62(12):2454–2467.
9. Choudhary S, et al. (2012) Silicon neurons that compute. *Artificial Neural Networks and Machine Learning–ICANN 2012*, eds Villa AEP, Duch W, Érdi P, Masulli F, Palm G (Springer, Berlin), pp 121–128.
10. Merolla P, et al. (2011) A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm. *Custom Integrated Circuits Conference (CICC), 2011 IEEE* (IEEE, Piscataway, NJ), pp 1–4.
11. Imam N, et al. (2012) Implementation of olfactory bulb glomerular-layer computations in a digital neurosynaptic core. *Front Neurosci* 6:83.
12. Brüderle D, et al. (2010) Simulator-like exploration of cortical network architectures with a mixed-signal VLSI system. *Proceedings of 2010 IEEE International Symposium on Circuits and Systems* (IEEE, Piscataway, NJ), pp 2784–2787.
13. Schemmel J, Gruebl A, Meier K, Mueller E (2006) Implementing synaptic plasticity in a VLSI spiking neural network model. *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN)* (IEEE, Vancouver), pp 1–6.
14. Pfeil T, et al. (2013) Six networks on a universal neuromorphic computing substrate. *Front Neurosci* 7:11.
15. Davison AP, et al. (2008) PyNN: A common interface for neuronal network simulators. *Front Neuroinform* 2:11.
16. Brüderle D, et al. (2009) Establishing a novel modeling tool: A Python-based interface for a neuromorphic hardware system. *Front Neuroinform* 3:17.
17. Schmuker M, Schneider G (2007) Processing and classification of chemical data inspired by insect olfaction. *Proc Natl Acad Sci USA* 104(51):20285–20289.
18. Huerta R, Nowotny T, García-Sanchez M, Abarbanel HDI, Rabinovich MI (2004) Learning classification in the olfactory system of insects. *Neural Comput* 16(8):1601–1640.
19. Huerta R, Nowotny T (2009) Fast and robust learning by reinforcement signals: Explorations in the insect brain. *Neural Comput* 21(8):2123–2151.
20. Martinetz T, Schulten K (1991) A "neural-gas" network learns topologies. *Artificial Neural Networks*, eds Kohonen T, Mäkisara K, Simula O, Kangas J (Elsevier B.V., North-Holland, Amsterdam), pp 397–402.
21. Linster C, Sachse S, Galizia CG (2005) Computational modeling suggests that response properties rather than spatial position determine connectivity between olfactory glomeruli. *J Neurophysiol* 93(6):3410–3417.
22. Wick SD, Wiechert MT, Friedrich RW, Riecke H (2010) Pattern orthogonalization via channel decorrelation by adaptive networks. *J Comput Neurosci* 28(1):29–45.
23. Schmuker M, Yamagata N, Nawrot MP, Menzel R (2011) Parallel representation of stimulus identity and intensity in a dual pathway model inspired by the olfactory system of the honeybee. *Front Neuroeng* 4:17.
24. Brader JM, Senn W, Fusi S (2007) Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural Comput* 19(11):2881–2912.
25. Soltani A, Wang X-J (2010) Synaptic computation underlying probabilistic inference. *Nat Neurosci* 13(1):112–119.
26. Fisher R (1936) The use of multiple measurements in taxonomic problems. *Ann Eugen* 7(2):179–188.
27. Gorodkin J (2004) Comparing two K-category assignments by a K-category correlation coefficient. *Comput Biol Chem* 28(5-6):367–374.
28. Nawrot MP, et al. (2008) Measurement of variability dynamics in cortical spike trains. *J Neurosci Methods* 169(2):374–390.
29. Häusler C, Nawrot MP, Schmuker M (2011) A spiking neuron classifier network with a deep architecture inspired by the olfactory system of the honeybee. *2011 5th International IEEE/EMBS Conference on Neural Engineering* (IEEE, Piscataway, NJ), pp 198–202.
30. Schemmel J, et al. (2010) A wafer-scale neuromorphic hardware system for large-scale neural modeling. *Proceedings of the 2010 International Symposium on Circuits and Systems (ISCAS)* (IEEE, Paris), pp 1947–1950.
31. Ruf B, Schmitt M (1998) Self-organization of spiking neurons using action potential timing. *IEEE Trans Neural Netw* 9(3):575–578.
32. Choe Y, Miikkulainen R (1998) Self-organization and segmentation in a laterally connected orientation map of spiking neurons. *Neurocomputing* 21(1-3):139–158.
33. Behi T, Arous N (2011) Word recognition in continuous speech and speaker independent by means of recurrent self-organizing spiking neurons. *Signal Processing* 5(5):215–226.
34. Nessler B, Pfeiffer M, Buesing L, Maass W (2013) Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS Comput Biol* 9(4):e1003037.
35. Pfeil T, et al. (2012) Is a 4-bit synaptic weight resolution enough?—constraints on enabling spike-timing dependent plasticity in neuromorphic hardware. *Front Neurosci* 6:90.
36. Friedmann S, Frémaux N, Schemmel J, Gerstner W, Meier K (2013) Reward-based learning under hardware constraints-using a RISC processor embedded in a neuromorphic substrate. *Front Neurosci* 7:160.
37. Krofczik S, Menzel R, Nawrot MP (2008) Rapid odor processing in the honeybee antennal lobe network. *Front Comput Neurosci* 2:9.
38. Strube-Bloss MF, Herrera-Valdez MA, Smith BH (2012) Ensemble response in mushroom body output neurons of the honey bee outpaces spatiotemporal odor processing two synapses earlier in the antennal lobe. *PLoS One* 7(11):e50322.
39. Strube-Bloss MF, Nawrot MP, Menzel R (2011) Mushroom body output neurons encode odor-reward associations. *J Neurosci* 31(8):3129–3140.
40. Mainen ZF, Sejnowski TJ (1995) Reliability of spike timing in neocortical neurons. *Science* 268(5216):1503–1506.
41. Boucsein C, Nawrot MP, Schnepel P, Aertsen A (2011) Beyond the cortical column: Abundance and physiology of horizontal connections imply a strong role for inputs from the surround. *Front Neurosci* 5:32.
42. Tuckwell HC (1988) *Introduction to Theoretical Neurobiology: Volume 2, Nonlinear and Stochastic Theories* (Cambridge Univ Press, Cambridge, UK).

Supporting Information

Schmuker et al. 10.1073/pnas.1303053111

SI Results

Number of Glomeruli. We used 10 virtual receptors (VRs) and thus 10 glomeruli in the network. The specific choice of 10 was made as a compromise to use as many VRs as possible to encode the data while staying within the maximal neuron count of 192 on the present hardware system. In the following, we describe this circumstance in more detail. First, each VR requires one glomerulus. One glomerulus consists of 6 input channels [receptor neurons (RNs)] and 13 neurons [7 projection neurons (PNs) and 6 local inhibitory neurons (LNs)]. Ten glomeruli thus require 130 neurons and 60 additional synapse line drivers for input spikes (see ref. 1 for a detailed technical explanation of the hardware system with regard to synapse line drivers). In addition, each association neuron (AN) population consists of 16 neurons (8 excitatory and 8 inhibitory neurons). For the iris dataset, we require three AN populations, or a total of 48 neurons. Because each neuron requires a synapse line driver, we thus require a total of 130 + 60 + 48 = 238 synapse line drivers. An additional VR would require 6 RNs + 7 PNs + 6 LNs = 19 additional line drivers, totaling to 257 and exceeding the maximum number of 256 synapse line drivers. Although it might be possible to gain space for one or two additional glomeruli by tuning the network to use fewer neurons per glomerulus, we do not expect a significantly different network behavior from a small increase in glomerulus number (see also ref. 2 for an analysis of how VR count affects the performance of a naïve Bayes classifier). Large-scale neuromorphic hardware systems that are under development (e.g., refs. 3 and 4) will overcome this limitation and support thousands of neurons.

Per-Class Classification Performance. For a thorough examination of the classification outcome, we depict the confusion matrix produced by the classifier network (Table S2). The classifier only produced errors on the separation of *Iris versicolor* and *Iris virginica*, whereas it always succeeded to correctly separate *Iris setosa* ($R_K = 0.87$, $P^{20} = 0.85$, $P^{80} = 0.89$). *I. setosa* is well separated from the other classes in feature space. The classifier network achieved perfect separation in cross-validated training in all 50 repetitions. *I. versicolor* and *I. virginica* overlap in feature space, providing a harder challenge to the classifier that is reflected in the higher error rate for that particular separation.

Network Optimization for Robustness Against Neuronal Variability. Constructing a heterogeneous network under constraints of limited neuron count and bounded synaptic weights imposes a tradeoff in connectivity: The number of neurons in a population that project on postsynaptic neurons (the postsynaptic "fan-in") must be sufficiently large to be able to drive the postsynaptic neuron to spiking. At the same time, individual populations must be kept small to accommodate many populations on the chip. In a previous version of the network, we achieved the maximum possible postsynaptic fan-in by using all-to-all connectivity (connection probability $p_{\text{conn}} = 100\%$) between all connected populations. In addition, that network contained only three LNs per glomerulus (instead of six in the current, optimized network). Because the fan-in of LNs on PNs is large anyway, this decision seemed a viable way to reduce the total neuron count. Achieving good classification performance with that network required a network-specific calibration routine (*SI Materials and Methods, Application-Specific Calibration of the Neuromorphic Hardware System*). The calibration improved the homogeneity of the transfer functions (Fig. S1*A*) and classification performance improved from R_K around 0.75 to values around 0.86 (Fig. S1*B*).

Analyzing the operation of the fully connected network in detail, we found that neurons in PN, LN, and AN populations were highly synchronized (Fig. S2). This synchronization at the population level was a direct consequence of full connectivity, which entails that all neurons of a population receive the same input. For example, all PNs in a glomerulus received input from the same set of RNs, and their spiking activity was highly correlated as a consequence of the common input. Under these conditions, the assumption of a population rate code with independent neurons is violated—the whole population of n neurons behaves like a single neuron with n times the synaptic weight. Clearly, the postsynaptic interaction of excitatory and inhibitory inputs in this regime is impaired, because synchronous PN spikes lead to synchronized inhibitory LN spikes with a short delay. In contrast, if all n neurons fire independently, the postsynaptic cell receives the same total number of spikes, but distributed more evenly in time. Hence, the chance that excitatory and inhibitory postsynaptic potentials overlap is considerably higher in the asynchronous case. In the optimized network, we reduced the synchronization within relevant populations by sparsifying the connectivity from RNs to PNs and from PNs to LNs and ANs by 50% and readjusting the synaptic weights accordingly.

As an additional step to increase the robustness against transfer function variability, we increased the number of LNs from three to six. Because the individual transfer functions of LNs underlie variability on the hardware, the total transfer function of an LN population will vary according to σ^2/n, with n the population size and σ^2 the variance of the individual transfer functions. Thus, increasing LN population size decreases variability of LN population transfer functions. In consequence, the inhibition strength that a PN population receives from other glomeruli becomes more homogeneous. In other words, increasing LN population size decreases the likelihood that a particular glomerulus may exert significantly higher inhibition than the others and thus alleviates the impact of transfer function variability.

Taken together, we achieved robustness to transfer function variability by two measures: First, we improved population rate coding by making the connectivity sparser, thus alleviating strong coupling on the postsynaptic side. Second, we increased the size of LN populations, thus reducing the variance of the population transfer functions of the LN groups. These steps resulted in the present network that is robust against variability in the transfer functions of individual neurons.

Effect of Lateral Inhibition on Classification Performance. The result that the naïve Bayes classifier's performance increases if trained on the PN firing rates compared with training on the VR responses (Fig. 5*C* in the main text) points out the beneficial effect of lateral inhibition in the presented network. Lateral inhibition transforms the broad, overlapping receptive fields of VRs into localized and more selective receptive fields on the PN level. This step facilitates the "credit assignment problem," that is, the identification of the PNs (or more precisely the PN–AN synapses) that are most responsible for the classification outcome. This information is necessary to select the correct synapses to be potentiated or depressed during classifier training (the "credit assignment problem").

Fig. S3 shows a sketch to illustrate this circumstance. Consider the VR "R_2" in Fig. S3*A*. Because the distances d_1 and d_2 are equal, the response of R_2 to the respective points will be equal, since it depends linearly on these distances (Eq. **S1**). Thus, the response magnitude of this particular VR provides ambiguous information with regard to class adherence, which complicates the learning process. Moreover, because VR receptive fields are

broad, there is considerable overlap in the receptive fields of R_1 and R_2 (Fig. S3*B*). One could now simply reduce the receptive field size of VRs (Fig. S3*C*). However, this approach would cause many data points not to be covered by any receptive field—the network would be "blind" toward these data points (Fig. S3*D*). They could neither be used for training, nor could the trained network achieve correct classification to any data point in the "blind" areas; these data points would simply produce no input to the network. Moreover, as the density of VRs in different regions of data space may be different, choosing one RF size for all VRs is clearly not optimal.

Lateral inhibition solves this problem in an elegant way: The response of a VR to a data point will be attenuated by lateral inhibition on the PN level if another VR is closer. Every PN thus has an "authoritative" region in data space where it provides the highest response and the responses of PNs in other glomeruli are attenuated. This region is equivalent to the Voronoi partitioning of input space with the VRs as generators (symbolized by the dotted lines in Fig. S3*E*). The resulting PN receptive fields become narrower in regions where there is overlap, but retain their full extent in regions where no other PN competes (Fig. S3*F*). Hence, lateral inhibition between PNs optimally and efficiently partitions data space on the PN level. Each PN thus represents a region in input space for which it is authoritative, considerably simplifying the credit assignment problem.

Why does the naïve Bayes classifier benefit from lateral inhibition? This classifier estimates the mean μ and the variance σ^2 of each class along each dimension in its input space. Classification is then achieved by comparing the (naively) estimated probability of adherence to class 1 vs. class 2. These probabilities are computed from the multivariate normal distributions $N(\boldsymbol{\mu},\boldsymbol{\sigma}^2)$, with $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ the means and variances along each dimension of input space. Broad VR receptive fields entail high variance of VR responses; thus, the estimated variance of the multivariate response distribution will also be high. In contrast, PN responses exhibit smaller variance because their receptive fields are narrower. Thus, the estimated variance of the PN response distribution will be smaller, and in consequence the naïve Bayes estimate of class adherence will exhibit lower variance, allowing for a better discrimination of classes in data space.

SI Materials and Methods

Network Parameters. Each glomerulus was driven by six RNs and contained seven PNs and six LNs. Each population in the associative layer comprised eight excitatory and eight inhibitory neurons. Connectivity and synaptic weights are described in detail Table S1. For a schematic overview of the general network architecture, see Fig. 1*A* in the main text. Time constants in the table refer to the biological value they model. The actual values on the hardware are 10^4 times smaller, due to the 10^4 speedup factor at which the hardware operates (5, 6). The weights are specified as fractions of the maximal weight $w_{\max}^{\mathrm{hw}\ \{\mathrm{inh,exc}\}}$ for excitatory and inhibitory synapses in the hardware system, where $w_{\max}^{\mathrm{hw\ inh}} \sim 4 \cdot w_{\max}^{\mathrm{hw\ exc}}$. Neurons were implemented as standard integrate-and-fire models (see ref. 1 for details).

VRs. The response r of a VR with coordinates $\mathbf{p}$ to the stimulus $\mathbf{s}$ is given by Eq. **S1** as follows:

$$r = 1 - \frac{d(\mathbf{s},\mathbf{p}) - d_{\min}}{d_{\max} - d_{\min}}, \qquad \textbf{[S1]}$$

with $d(\mathbf{s},\mathbf{p})$ the Manhattan distance (Minkowski metric with $k = 1$, sum of absolute coordinate differences) between $\mathbf{s}$ and $\mathbf{p}$; $d_{\min}$ and $d_{\max}$ denote the minimum and maximum distance observed in the dataset. Hence, the receptor response is a value in [0, 1], and it is inversely proportional to the distance between stimulus and receptor.

The receptive fields implemented by Eq. **S1** are equivalent to linear radial basis functions representing cones. They extend over the entire space that is covered by the data ("broadly tuned"). Their receptive fields are largely overlapping. This guarantees that there are no "blind spots" in data space that are not covered by any receptive field.

VRs were placed in data space using a self-organizing process. In this study, we used the neural gas algorithm (7), as implemented in the MDP toolkit (8). The neural gas learns to represent the distribution of data in the original coordinate space, thus ensuring that the VRs cover data space appropriately. Each node in the neuronal gas corresponds to one VR. Using n VRs, a stimulus will thus evoke a response vector $\mathbf{r} = (r_1, \ldots, r_n)$. The elements of response vector r_i are then converted into firing rates ρ_i using Eq. **S2** as follows:

$$\rho_i = r_i \cdot (\rho_{\max} - \rho_{\min}) + \rho_{\min} \quad \text{for} \quad i = (1, \ldots, n), \qquad \textbf{[S2]}$$

with $\rho_{\min}$ and $\rho_{\max}$ the minimal and maximal firing rate, set to 20 and 70 spikes/s, respectively. Firing rates were transformed into spike trains using a gamma process of order five. The waiting time between stimulus onset and the first RN spike was drawn from the appropriate waiting time distribution, in our case a gamma distribution of order six, to prevent synchronization of RNs at stimulus onset. We chose a gamma process to generate spike times because its spiking statistics compares realistically to biological neurons (see, e.g., ref. 9). In addition, the increased regularity of a gamma process of order five [Fano factor (FF) = 0.2] compared with a Poisson process (FF = 1.0) reduces the spike count variability and thus yields a more reliable encoding of input firing rates.

VRs were implemented in software as a convenient approach to convert numerical data into a spiking format. The VR approach satisfies the need for dimensionality reduction due to limited neuron counts and provides a generic approach to convert real-valued data into bounded firing rate intervals.

Network Training and Supervised Learning Rule. The classifier network was trained using a supervised learning algorithm. Only synapses between PNs and excitatory association layer neurons were subject to learning.

After stimulus presentation, a synapse was eligible for weight update if it fulfilled a Hebbian eligibility constraint. A synaptic weight was eligible for updating if the target neuron $\upsilon_{\mathrm{target}}$ was a member of the winner population $\Upsilon_{\mathrm{winner}}$, and if the firing rate ρ_{pre} of the presynaptic neuron during the previous stimulus presentation exceeded a threshold θ (fixed to 35 spikes/s in this study). The eligibility constraint ε can thus be formalized as follows:

$$\varepsilon = \begin{cases} 1, & \text{if } \rho_{\mathrm{pre}} > \theta \text{ and } \upsilon_{\mathrm{target}} \in \Upsilon_{\mathrm{winner}}, \\ 0, & \text{otherwise.} \end{cases} \qquad \textbf{[S3]}$$

The change of the weight $\Delta w_{\mathrm{PN}\to\upsilon}$ between any PN and target neuron υ in the association layer was governed by Eq. **S4** as follows:

$$\Delta w_{\mathrm{PN}\to\upsilon} = \begin{cases} \varepsilon \cdot c, & \text{if classification was correct,} \\ -\varepsilon \cdot c, & \text{if classification was incorrect,} \end{cases} \qquad \textbf{[S4]}$$

with c a constant value determined by the granularity of synaptic weights on the hardware (1). The new weight w_{new} was computed from w_{old} as in Eq. **S5**:

$$w_{\mathrm{new}} = w_{\mathrm{old}} + \Delta w_{\mathrm{PN}\to\upsilon}. \qquad \textbf{[S5]}$$

Synaptic weights were bounded in the interval $[w_{\min}, w_{\max}]$ by the constraints of the hardware. Thus, the final value of the synaptic weight was given by Eq. **S6** as follows:

$$w_{\text{final}} = \begin{cases} w_{\max}, & \text{if } w_{\text{new}} > w_{\max}, \\ w_{\min}, & \text{if } w_{\text{new}} < w_{\min}, \\ w_{\text{new}}, & \text{otherwise.} \end{cases} \quad \textbf{[S6]}$$

Evaluation of Classifier Performance. Classifier performance was evaluated from fivefold cross-validation (CV). The data were split into five equal parts, and four parts were used in training and one part was used to test the classifier predictions in each CV run. After five runs, each data point was once in the test set, allowing computing a single performance value for all five CV runs. CV was repeated multiple times with different random splitting of the data into five equal parts.

Classifier performance (i.e., prediction accuracy) was assessed using Gorodkin's R_K correlation coefficient for discrete multicategory data (10). The aim is to compare a prediction $\mathbf{Y}_{\text{pred}}$ to the true target values $\mathbf{Y}$, with $Y_{n,k} \in \{0,1\}$ for n predictions of k classes. The $K \times K$ confusion matrix $\mathbf{C}$ contains the number of correctly and falsely predicted data instances per class. $C_{k,k}$ contains the number of correctly predicted instances of class k, and off-diagonal elements contain the number of falsely predicted instances. For example, $C_{1,2}$ contains the number of instances predicted to belong to class 1, but actually belonging to class 2. The K-category correlation coefficient computes as in Eq. **S7**:

$$R_K = \frac{\sum_{klm} c_{k,k} c_{l,m} - c_{k,l} c_{m,k}}{\sqrt{\sum_k \left(\sum_l c_{k,l}\right)\left(\sum_{l',k' \neq k} c_{k',l'}\right)} \sqrt{\sum_k \left(\sum_l c_{l,k}\right)\left(\sum_{l',k' \neq k} c_{k',l'}\right)}}. \quad \textbf{[S7]}$$

Compared with other frequently used performance measures like "percent correct," R_K is more sensitive to small performance differences when overall performance is already high and thus better suited for benchmarking. In addition, R_K is corrected for the bias introduced by skewed class proportions. For example, if 90% of the data are of one class and 10% the other class, we could yield "90% correct" classification by simply assigning all data samples to the first class. In contrast, R_K would report a value of zero, which is intuitively more accurate.

Application-Specific Calibration of the Neuromorphic Hardware System. The network-specific calibration for the previous version of the network with 100% connectivity (*SI Results, Network Optimization for Robustness Against Neuronal Variability*) consisted of two steps. We first calibrated the PNs for homogeneous rate response, before calibrating the LNs. Calibration was carried out with the weight of all inhibitory synapses set to zero. We first measured PN firing rates in response to a 1-s stimulation with nominal intensity, formed the median from all PN rates and used this as target firing rate. The "fitness" of the rate distribution was assessed by mean square deviation (MSD) of PN firing rates from the targeted PN firing rate as follows:

$$\text{MSD} = \frac{1}{n} \sum_1^n \left(\rho_{\text{goal}}^{\text{PN}} - \rho_{\text{i}}^{\text{PN}}\right)^2, \quad \textbf{[S8]}$$

with n the number of PNs, $\rho_{\text{i}}^{\text{PN}}$ the firing rate of the ith PN, and $\rho_{\text{goal}}^{\text{PN}}$ the targeted firing rate. The weights w_i from the RNs to the i^{th} PN were then updated according to the following:

$$w_i^{\text{new}} = w_i \cdot \frac{\rho_{\text{goal}}^{\text{PN}}}{\rho_{\text{i}}^{\text{PN}}}. \quad \textbf{[S9]}$$

In this case, we relied on the automatic conversion of the *Spikey* control software that mapped the weight values into the discrete distribution required by the hardware (1).

When the MSD failed to decrease over five iterations, optimization was terminated and the set of weights that yielded the best MSD until then was used. After the weights from RNs to PNs were optimized, we adjusted the weights between PNs and LNs using the same algorithm.

Speed Considerations for the Neuromorphic Hardware System. The execution of the network on the accelerated hardware happens extremely fast: A simulation lasting for 150-s biological time is executed in 15 ms (a 10^4 speedup factor). However, the total run time of the classifier network is mainly determined by other factors, which we describe in the following.

A typical CV run requires 150 stimulus presentations of 1-s duration. Before starting such a simulation session, generic calibration data must be loaded and applied. The network connectivity as well as synaptic weights must be encoded and transferred, and subsequently be mapped from their specification in biologically realistic physical units to the appropriate hardware parameters. In addition, for each of the 150 simulations, spike data need to be sent to and received from the hardware, including transfer, encoding, and decoding of spike times and neuron IDs. During the training phase of the classifier, synaptic weights also have to be updated before every stimulus presentation.

The absolute duration of these additional factors depends heavily on the efficiency of the software interface that links the hardware with the host system. Because it is a prototype system, this software interface is constantly developed and improved. It is therefore difficult to state an absolute number for the effective speedup achieved by offloading network simulations to the hardware. To give the reader the opportunity of an informed estimate, we analyzed how much time is required by each of the above steps (Fig. S4).

Several of these steps still bear potential for optimization. For example, the time required for weight update could be drastically shortened by differential configuration, i.e., updating only those hardware weights that have changed, instead of overwriting all weights as in the current implementation. In addition, on the current system all spike times produced in the network are being transferred back to the host system during training and testing phases of the classifier network. The interface can be improved to only transfer those spikes that are necessary for the off-chip calculation of the weight change, namely PNs and excitatory ANs, and not transferring spike times from LNs and inhibitory ANs. When the network is completely trained, only the spike times from excitatory ANs are needed, further reducing the overhead due to handling spike data. We plan to implement these optimizations in future versions of the software interface.

1. Pfeil T, et al. (2013) Six networks on a universal neuromorphic computing substrate. *Front Neurosci* 7:11.
2. Schmuker M, Schneider G (2007) Processing and classification of chemical data inspired by insect olfaction. *Proc Natl Acad Sci USA* 104(51):20285–20289.
3. Furber SB, et al. (2013) Overview of the SpiNNaker System Architecture. *IEEE Trans Comput* 62(12):2454–2467.
4. Schemmel J, et al. (2010) A wafer-scale neuromorphic hardware system for large-scale neural modeling. *Proceedings of the 2010 International Symposium on Circuits and Systems (ISCAS)* (IEEE, Paris), pp 1947–1950.
5. Brüderle D, et al. (2010) Simulator-like exploration of cortical network architectures with a mixed-signal VLSI system. *Proceedings of 2010 IEEE International Symposium on Circuits and Systems* (IEEE, Piscataway, NJ), pp 2784–2787.
6. Schemmel J, Gruebl A, Meier K, Mueller E (2006) *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN)* (IEEE, Vancouver), pp 1–6.
7. Martinetz T, Schulten K (1991) A "neural-gas" network learns topologies. *Artificial Neural Networks*, eds Kohonen T, Mäkisara K, Simula O, Kangas J (Elsevier B.V., North-Holland, Amsterdam), pp 397–402.

8. Zito T, Wilbert N, Wiskott L, Berkes P (2008) Modular toolkit for Data Processing (MDP): A Python data processing framework. *Front Neuroinform* 2:8.
9. Nawrot MP, et al. (2008) Measurement of variability dynamics in cortical spike trains. *J Neurosci Methods* 169(2):374–390.
10. Gorodkin J (2004) Comparing two K-category assignments by a K-category correlation coefficient. *Comput Biol Chem* 28(5-6):367–374.

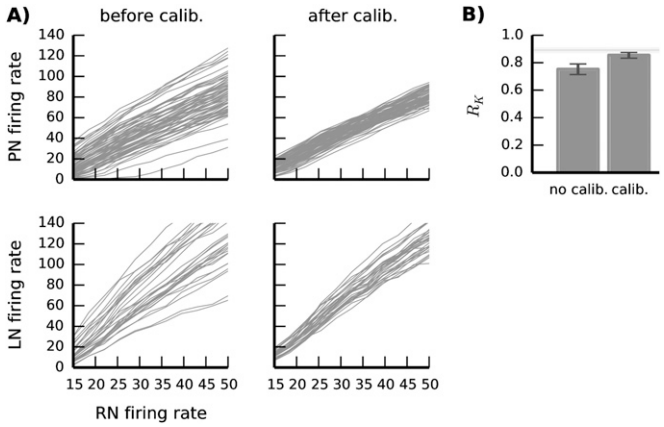

Fig. S1. Neuronal variability on the hardware system and impact of calibration on classifier performance using a previous version of the network with 100% connectivity. (*A*) Rate-response functions of the hardware neurons, before (*Left*) and after (*Right*) calibration (5-s stimulation duration). Upper row, PNs; lower row, LNs. (*B*) Classifier performance in the iris benchmark before and after network specific calibration. Error bars denote P^{20} and P^{80}. The horizontal gray bar indicates naïve Bayes performance.

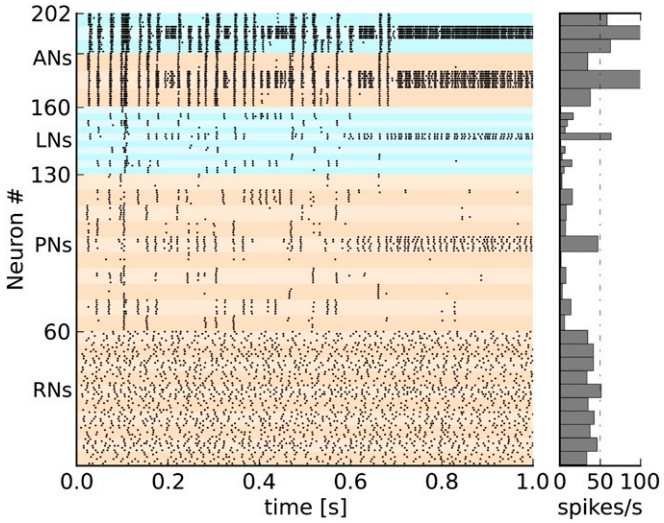

Fig. S2. Synchronized spiking activity in PNs, LNs, and ANs in a previous version of the network with p_{conn} = 100%. The total neuron count in the previous network is lower than in the version presented in the main text due to different per-population neuron counts for LNs (three in the previous network vs. six in the main text) and inhibitory ANs (six vs. eight).

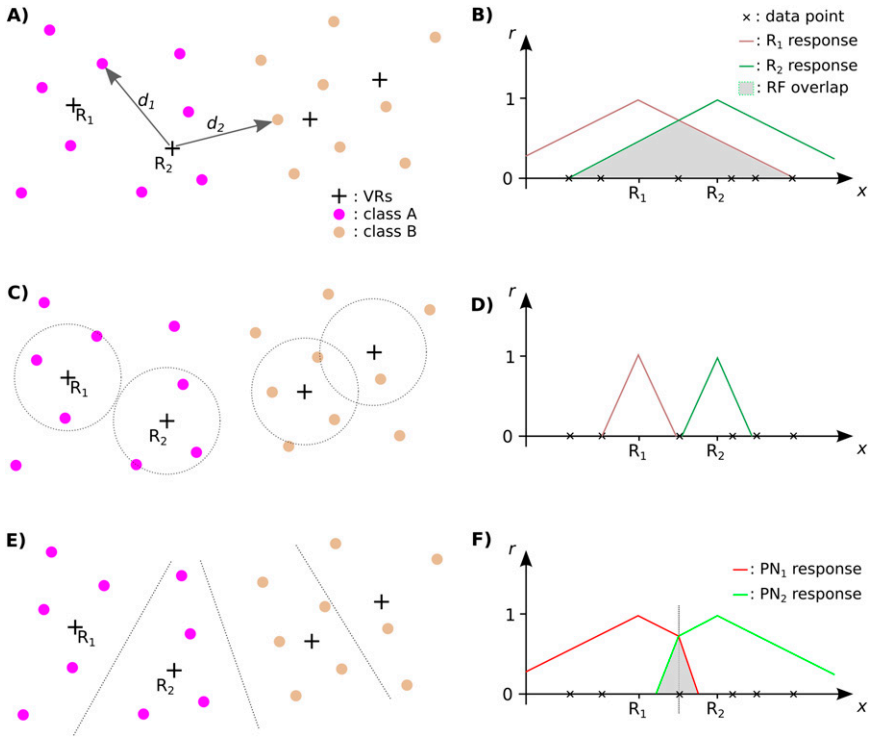

Fig. S3. Illustration of the credit assignment problem and the effect of lateral inhibition. (*A*) Cartoon of a hypothetical two-class, 2D classification problem with VRs. The distances d_1 and d_2 are equal. (*B*) One-dimensional sketch of the response profile of the two VRs, R_1 and R_2. (*C*) Effect of reducing VR receptive field size in data space. (*D*) Effect of reducing VR receptive field size on the response profiles. (*E*) Voronoi partitioning of input space with VRs as generators. (*F*) Effect of lateral inhibition on PN receptive field size.

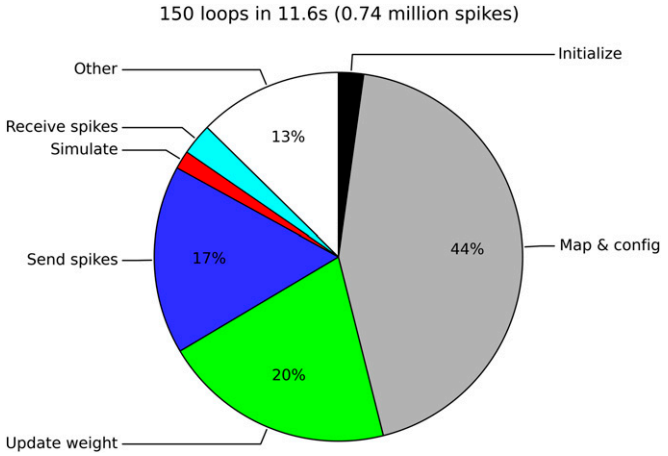

Fig. S4. Simulation time for one CV run (150 simulations) broken down into discrete steps. The largest fraction of the time is required by mapping the simulation parameters to hardware-compatible values and configuring the hardware network. Some of these tasks have to be repeated for every simulation, adding up to a substantial amount of total time. The second largest chunk is taken up by updating weights. The actual simulation requires less than 2% of the total time. "Other" encompasses numerous small tasks like handling of spike data and network configuration in the PyNN interface code. All numbers are subject to change as the software interface evolves.

Table S1. Network parameters

Type of neuron	Parameters
Receptor neurons (RNs)	
Type	Gamma process ($\gamma = 5$)
Count	Six RNs per VR
Outgoing connectivity	Each RN projects on the PNs in one glomerulus; connection probability $p_{\text{conn}} = 50\%$
Outgoing weights	RN to PN: $0.5 \cdot w_{\text{max}}^{\text{hw exc}}$
Projection neurons (PNs)	
Type	Leaky integrate-and-fire
Count	Seven PNs per glomerulus
Outgoing connectivity	Excitatory synapses on LNs in the same glomerulus ($p_{\text{conn}} = 50\%$) and on excitatory ANs ($p_{\text{conn}} = 50\%$)
Outgoing weights	PN to LN: $0.7 \cdot w_{\text{max}}^{\text{hw exc}}$ PN to AN: initially random between $0.2 \cdot w_{\text{max}}^{\text{hw exc}}$ and $0.66 \cdot w_{\text{max}}^{\text{hw exc}}$ (adjusted in training)
Local inhibitory neurons (LNs)	
Type	Leaky integrate-and-fire
Count	Six LNs per glomerulus
Outgoing connectivity	Inhibitory synapses on all PNs in all other glomeruli ($p_{\text{conn}} = 100\%$)
Outgoing weights	LN to PNs: $0.133 \cdot w_{\text{max}}^{\text{hw inh}}$
Excitatory neurons in association layer (ANs)	
Type	Leaky integrate-and-fire
Count	Eight per association population
Outgoing connectivity	Excitatory synapses on adjoint inhibitory population ($p_{\text{conn}} = 50\%$)
Outgoing weights	AN to adjoint inhibitory population: $0.5 \cdot w_{\text{max}}^{\text{hw exc}}$
Inhibitory neurons in association layer	
Type	Leaky integrate-and-fire
Count	Eight per association population
Outgoing connectivity	Inhibitory synapses on excitatory neurons of all other association populations ($p_{\text{conn}} = 100\%$)
Outgoing weights	Inhibitory neuron to ANs in different association populations: $1.0 \cdot w_{\text{max}}^{\text{hw inh}}$

Table S2. Average count of predicted vs. actual class adherence (columns vs. rows) obtained across 50 repetitions of fivefold CV

	I. setosa	*I. versicolor*	*I. virginica*
I. setosa	50.0	0.0	0.0
I. versicolor	0.0	47.1	10.7
I. virginica	0.0	2.9	39.3