LETTER

https://doi.org/10.1038/s41586-019-1424-8

Towards artificial general intelligence with hybrid Tianjic chip architecture

Jing Pei[1,2,13], Lei Deng[1,13], Sen Song[3,4,13], Mingguo Zhao[5,13], Youhui Zhang[6,13], Shuang Wu[1,2,13], Guanrui Wang[1,2,13], Zhe Zou[1,2], Zhenzhi Wu[7], Wei He[1,2], Feng Chen[5], Ning Deng[8], Si Wu[9], Yu Wang[10], Yujie Wu[1,2], Zheyu Yang[1,2], Cheng Ma[1,2], Guoqi Li[1,2], Wentao Han[6], Huanglong Li[1,2], Huaqiang Wu[8], Rong Zhao[11], Yuan Xie[12] & Luping Shi[1,2]*

There are two general approaches to developing artificial general intelligence (AGI)[1]: computer-science-oriented and neuroscience-oriented. Because of the fundamental differences in their formulations and coding schemes, these two approaches rely on distinct and incompatible platforms[2–8], retarding the development of AGI. A general platform that could support the prevailing computer-science-based artificial neural networks as well as neuroscience-inspired models and algorithms is highly desirable. Here we present the Tianjic chip, which integrates the two approaches to provide a hybrid, synergistic platform. The Tianjic chip adopts a many-core architecture, reconfigurable building blocks and a streamlined dataflow with hybrid coding schemes, and can not only accommodate computer-science-based machine-learning algorithms, but also easily implement brain-inspired circuits and several coding schemes. Using just one chip, we demonstrate the simultaneous processing of versatile algorithms and models in an unmanned bicycle system, realizing real-time object detection, tracking, voice control, obstacle avoidance and balance control. Our study is expected to stimulate AGI development by paving the way to more generalized hardware platforms.

The neuroscience-oriented approach to AGI attempts to closely mimic the cerebral cortex, being based on observations of a tight interaction between memory and computing, rich spatiotemporal dynamics, spike-based coding schemes and various learning rules[9–12], which are normally represented as spiking neural networks (SNNs). By contrast, the computer-science-oriented approach mainly involves explicit algorithms that are executed on computers[13]. Of these algorithms, the prevailing non-spiking artificial neural networks (ANNs)—inspired in part by the cortex in terms of spatial complexity[14]—have made substantial progresses in dealing with specific tasks[15,16], such as image classification[17], speech recognition[18], language processing[19] and game playing[20].

Although both approaches can solve subproblems in specialized domains where data are abundant, it remains difficult to solve complex dynamic problems with the uncertain or incomplete information that is associated with many systems[21]. To further improve the intelligence capability needed to achieve AGI, there is an increasing trend to incorporate more biologically inspired models or algorithms into the prevailing ANNs, resulting in a more explicit dialogue between the two approaches[22–29]. Given current progress in machine learning and neuroscience, an AGI system should have at least the following features: first, support for vast and complex neural networks that can represent rich spatial, temporal and spatiotemporal relationships; second, support for hierarchical, multigranular and multidomain network topologies, but without being limited to a specialized network structure; third, support for a wide range of models, algorithms and coding schemes; and fourth, support for the intertwined cooperation of multiple specialized neural networks that are designed for different tasks in parallel processing. This requires a general platform to effectively support these features in a unified architecture that can implement the prevailing ANNs as well as neuroscience-inspired models and algorithms.

To support these features, we developed a cross-paradigm computing chip that can accommodate computer-science-oriented and neuroscience-oriented neural networks (Fig. 1). Designing a general platform that is compatible with diverse neural models and algorithms is a fundamental challenge, especially for distinct ANN and biologically inspired (for example, SNN) primitives. Usually, ANNs and SNNs have different modelling paradigms in terms of information representation, computation philosophy and memory organization (Fig. 2a). Among these differences, the biggest is that an ANN processes information in precise multibit values, while an SNN uses binary spike trains. To implement both models on one platform, the spikes need to be represented as digital sequences (1 or 0) so that they are compatible with the ANN coding format of digital number. Several other key points also need to be considered carefully. First, an SNN operates in spatiotemporal domains, which requires the memorization of historical membrane-potential and spike patterns within a certain duration, while an ANN accumulates the weighted activations intermediately and refreshes the information every cycle. Second, the computation of an SNN includes membrane-potential integration, threshold crossing and potential reset, which is driven by spike events. By contrast, an ANN is related mainly to dense multiply-and-accumulate (MAC) operations and activation transformations. Third, the processing of spike patterns in SNNs requires a bit-programmable memory and extra high-precision memories to store the membrane potential, firing threshold and refractory period, whereas an ANN needs only byte-wise memories for activation storage and transformation. The implementation comparisons between an ANN neuron and an SNN neuron are illustrated in Fig. 2b. On the other hand, there are some similarities between ANN and SNN neurons, which leaves room to fuse the model implementations.

By compiling various neural network models in both domains, we were able to carry out a detailed comparison to align the model dataflow, with one-to-one correspondence, to relevant building blocks—namely axon, synapse, dendrite, soma and router (Extended Data Table 1). On the basis of this unified abstraction, we built a cross-paradigm neuron scheme (Fig. 2c). Overall, we designed the synapse and dendrite to be shared, while the axon and soma can be reconfigured independently.

[1]Department of Precision Instruments, Center for Brain-Inspired Computing Research (CBICR), Optical Memory National Engineering Research Center, Tsinghua University, Beijing, China. [2]Beijing Innovation Center for Future Chip, Tsinghua University, Beijing, China. [3]Laboratory of Brain and Intelligence, Department of Biomedical Engineering, CBICR, Tsinghua University, Beijing, China. [4]IDG/McGovern Institute for Brain Research, Tsinghua University, Beijing, China. [5]Department of Automation, CBICR, Tsinghua University, Beijing, China. [6]Department of Computer Science and Technology, CBICR, Tsinghua University, Beijing, China. [7]Lynxi Technologies, Beijing, China. [8]Institute of Microelectronics, CBICR, Tsinghua University, Beijing, China. [9]State Key Laboratory of Cognitive Neuroscience and Learning, Beijing Normal University, Beijing, China. [10]Department of Electronic Engineering, CBICR, Tsinghua University, Beijing, China. [11]Engineering Product Development Pillar, Singapore University of Technology and Design, Singapore, Singapore. [12]Department of Electrical and Computer Engineering, University of California Santa Barbara, Santa Barbara, CA, USA. [13]These authors contributed equally: Jing Pei, Lei Deng, Sen Song, Mingguo Zhao, Youhui Zhang, Shuang Wu, Guanrui Wang. *e-mail: lpshi@tsinghua.edu.cn

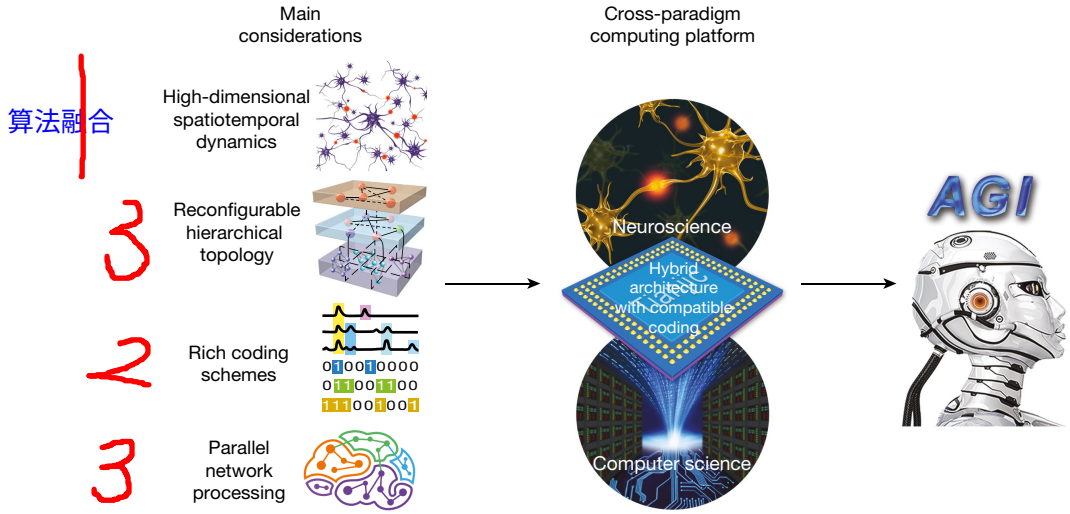


Fig. 1 | The hybrid approach to the development of AGI. The hybrid approach combines the advantages of neuroscience-oriented and computer-science-oriented approaches (shown on the left), aiming to develop a cross-paradigm computing platform with several broad features identified from human brains and prevailing machine-learning algorithms.

In the axon block, we deployed a small buffer memory to memorize the historic spike patterns in the SNN mode. This buffer memory supports a reconfigurable spike-collection duration and bit-wise access through shift operations. In ANN mode, the same memory can be reorganized as ping-pong chunks for buffering input and output data; this decouples the computation and data transfer for parallel processing. Here, the synaptic weights and neuronal parameters are pinned into on-chip memories, which enables localized high-throughput computation by minimizing data movement between the processing unit and the memory. In the dendrite block, the membrane-potential integration in SNN mode and MACs in ANN mode share the same calculators, reunifying high-level abstraction of SNNs and ANNs during processing. Specifically, in ANN mode the MAC units are used to perform multiplication and accumulation; in SNN mode, a bypassing mechanism is provided to skip the multiplication in order to allow energy reduction under a temporal window of length one. The soma can be reconfigured to be either a spike generator with potential storage, threshold comparison, deterministic or probabilistic fire, and potential reset in SNN mode; or a simple activation function block in ANN mode. The leaky function of membrane potential can reduce the potential value through fixed or adaptive leakage. The activation function in ANN mode relies on a reconfigurable look-up table (LUT) that provides arbitrary function.

By combining the axon, synapse, dendrite and soma blocks, we designed a unified functional core (FCore) (Fig. 2d; for more details, see Extended Data Fig. 1). To achieve deep fusion, nearly the whole of the FCore is reconfigurable for high utilization in different modes. The dendrite and soma were divided into multiple groups during operation. The computation within each group is parallelized (with 16 MACs per dendrite for each clock cycle), while the intergroup execution is serialized. The FCore is able to cover the linear integration and nonlinear transformation operations used by most ANNs and SNNs. In addition, to transfer information among neurons, we built a router to receive and send messages. Because the messages can be encoded in either ANN or SNN format depending on the configuration, we designed a unified format for the routing packet and a shared routing infrastructure to transfer both message types. The routing packet usually contains control, address and data segments, where the data segment can be either multibit activation values in ANN mode, or nothing in SNN mode, since the routing packet itself acts as a spike event. Depending on need, a pre-soma can package the output into either an SNN or an ANN packet according to the soma configuration, and the post-axon parses the routing packet into either SNN or ANN format according to its axon configuration.

Because of the fully independent configurability for axon (input) and soma (output), along with the shared dendrite (computation), FCore provides great flexibility for building homogenous or heterogeneous networks by appropriately wiring many cores. If we configure all of the units in the same mode, a homogeneous paradigm of an SNN or an ANN network primitive can be achieved that supports many single-paradigm models, including SNNs and ANNs (for example, multilayer perceptrons (MLPs), convolutional neural networks (CNNs), recurrent neural networks (RNNs), and rate-based biologically inspired neural networks). Furthermore, the FCore allows the construction of a heterogeneous network for exploring hybrid modelling. By independently configuring the axon and soma in different modes, we can easily realize a hybrid network primitive with 'ANN-input and SNN-output' or 'SNN-input and ANN-output' (Fig. 2e). In other words, the FCore can act as an ANN/SNN converter (see Methods for details). This cross-paradigm scheme opens up the possibility of designing innovative hybrid models, providing an efficient platform for cross-modelling exploration.

To support the parallel processing of large networks or multiple networks concurrently, our Tianjic chip adopts a many-core architecture with scattered localized memory for timely and seamless communication. The FCores on this chip are arranged in a two-dimensional (2D) mesh manner, as shown in Fig. 2e, f. A reconfigurable routing table in the router of each FCore allows arbitrary connection topology. By configuring the routing table, we can connect a neuron to any other neuron inside or outside an FCore or even outside the chip, which helps to build multigranular network topologies (for example, feedforward or recurrent). Furthermore, besides the normal point-to-point (P2P) routing, Tianjic also encompasses several special strategies to increase its fan-in capability (the number of inputs a neuron can handle) and fan-out capability (the number of outputs a neuron can drive). For a typical neuromorphic core, the numbers of fan-ins and fan-outs are normally limited by the memory and interface, which restrains the model scale. In Tianjic, the fan-ins and fan-outs of each FCore can be extended easily by designing interneuron lateral cooperation, interFCore hierarchical accumulation, intraFCore/interFCore neuron copy, or interFCore multicast routing (see Methods for details). Together with 2D mesh pile-up at chip level, Tianjic exhibits strong scalability to ultralarge neural networks, while still maintaining seamless communication among deeply intertwined neural networks during concurrent processing.

The layout and a physical view of the Tianjic chip and the testing boards are shown in Fig. 3a and Extended Data Fig. 2. The chip consists of 156 FCores, containing approximately 40,000 neurons and

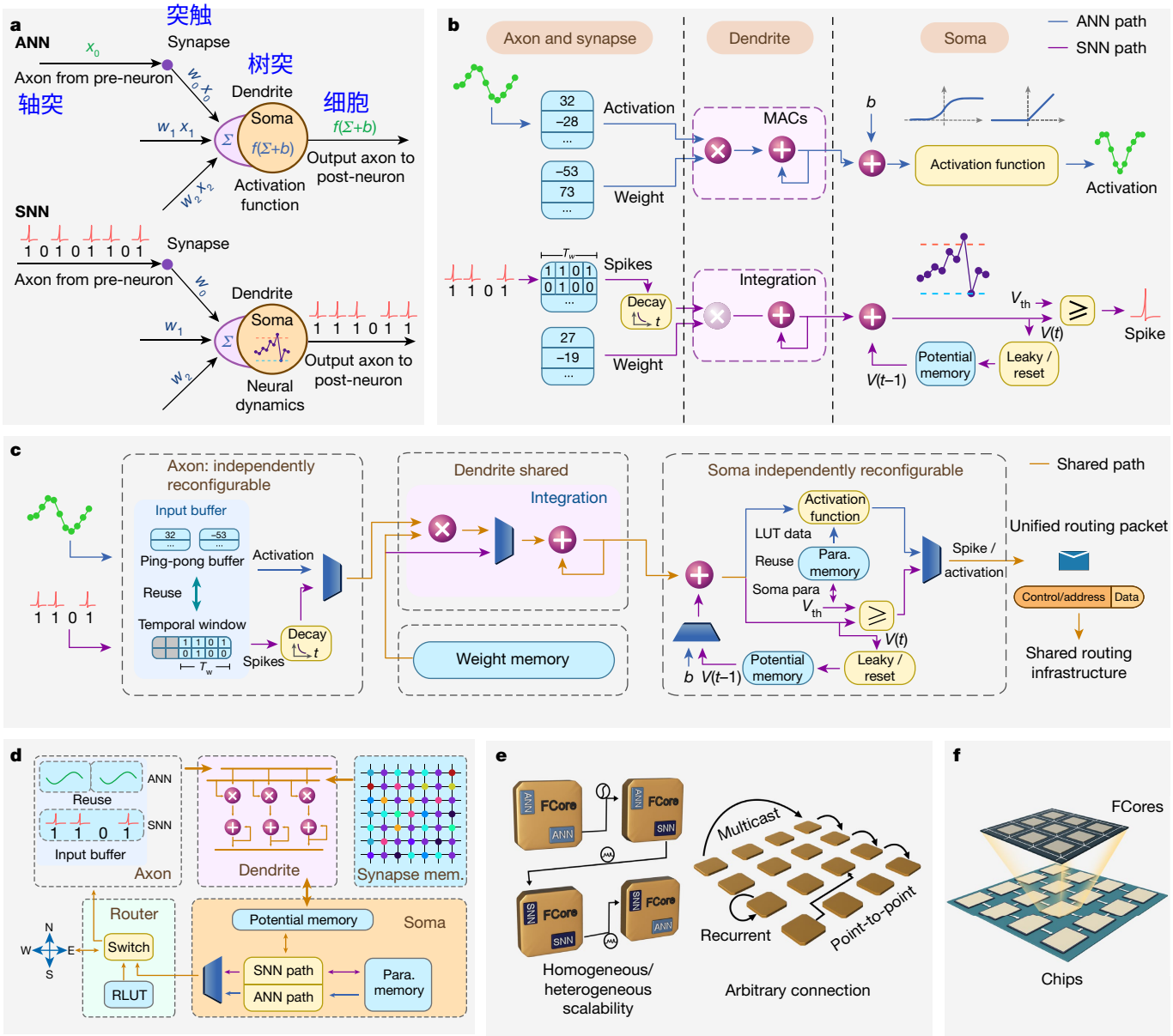


Fig. 2 | Design of the Tianjic chip. a, Computational model of an ANN or biologically inspired (for example, SNN) neuron. w_0, w_1, w_2 are synaptic weights; x_0, x_1, x_2 are input activations; Σ is the dendritic integration; f is the activation function; and b is the bias. **b**, Implementation diagrams for an ANN or SNN neuron. $V(t)$ is the neuronal membrane potential at time step t, and V_{th} is the firing threshold. Numbers in blue boxes are examples of input activation/spike and weight values. The faded purple multiplier in the SNN path indicates that the dendrite can possibly bypass the multiplication (for example, in the case that the time window length equals one). **c**, Diagram of a hybrid circuit, showing a cross-paradigm neuron with fused ANN and SNN components. Para. memory, parameter memory. **d**, Diagram of a unified functional core (FCore). Each FCore includes axon, synapse, dendrite, soma and router building blocks. Synapse mem., synapse memory. **e**, Flexible modelling configuration and connection topology of FCores. The coding schemes can be freely transformed between ANN and SNN modes, enabling heterogeneous neural networks. The scheme also allows flexible connections for the implementation of arbitrary network topology. **f**, Illustration of the hierarchy of 2D mesh architecture at the core and chip levels, demonstrating the ability to scale up the technique. RLUT, routing look-up table.

10 million synapses. Fabricated using 28-nm processing technology, Tianjic occupies a die area of 3.8 × 3.8 mm^2 (Extended Data Table 2). The chip area occupied by each individual block, including axon, dendrite, soma, router, controller and other chip overheads, is shown in Fig. 3b. Because of resource reuse, an area only slightly larger than that of a single paradigm (roughly 3%) was used to fuse the SNN and ANN modes. The power breakdown of FCore is given in Fig. 3c. During operation, the dendrite module (here including the involved axon and synapse blocks when performing dendritic integration) consumes the most power (63%). With its distributed on-chip memory and decentralized many-core architecture, Tianjic provides an internal memory bandwidth of more than 610 gigabytes (GB) per second, and yields an effective peak performance of 1.28 tera operations per second (TOPS) per watt in ANN mode when running at 300 MHz. In SNN mode, synaptic operations are usually used to bench the chips, and Tianjic achieves an effective peak performance of about 650 giga synaptic operations per second (GSOPS) per watt. Details can be found in Extended Data Table 2.

Tianjic is able to support diverse neural network models, including neuroscience-inspired networks (for example, SNNs and rate-based biologically inspired neural networks) and computer-science-oriented networks (for example, MLP, CNNs and RNNs). Figure 3d shows the results of testing different network models on the Tianjic chip versus a general processing unit (GPU; see Methods for model details).

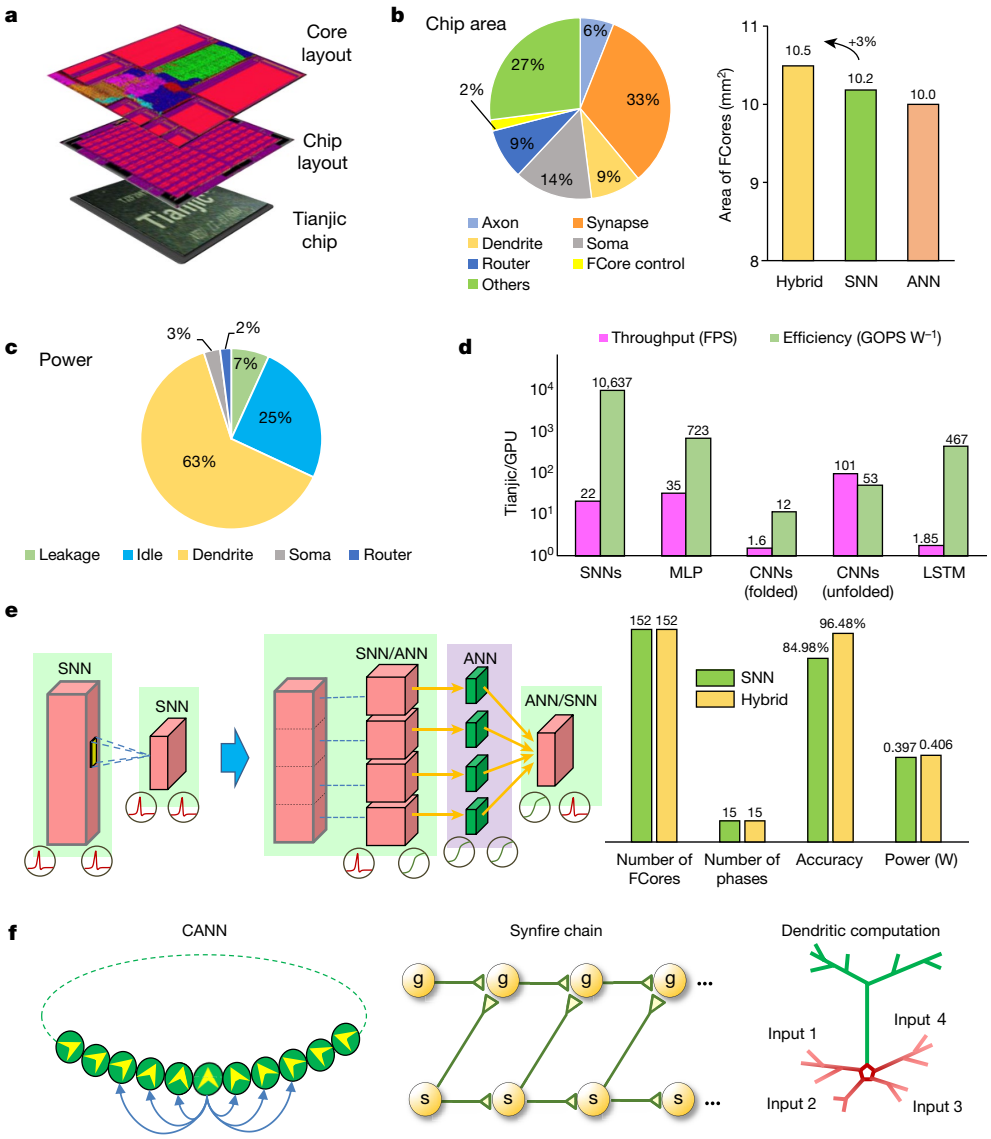


Fig. 3 | Summary of chip evaluation and modelling. a, Integrated layout and packaging of the Tianjic chip. **b**, Left, percentage of the chip area that is occupied by different features (axons, dendrites, routers and so on). Right, owing to the high level of resource sharing and reconfigurability, only a small area increase (roughly 3%) is needed to fuse the two paradigms. **c**, Power breakdown for FCore. **d**, Evaluation of FCore performance in various single-paradigm models, including SNNs, MLPs, CNNs (under folded or unfolded mapping) and long short-term memory networks (LSTMs). GOPS, giga operations per second; FPS, frames per second. **e**, Left, example of the implementation of a large-scale SNN with ANN dendritic relay. Right, with the help of ANN relays to transfer intermediate membrane potentials with high precision, a hybrid device was able to achieve higher recognition accuracy than an SNN alone, with negligible hardware overheads. **f**, The Tianjic chip can also support more biologically plausible neural network models (for example, CANN; a temporal-coding-based synchronous firing (synfire) neural chain; and dendritic multicompartment models). g, graded; s, synfire.

By forming a parallel on-chip memory hierarchy and organizing the dataflow in a streaming fashion, the Tianjic chip can provide improved throughput (1.6 to 10^2 times) and power efficiency (12 to 10^4 times) over the GPU. A detailed mapping of these networks onto the chip is illustrated in Extended Data Fig. 3.

Moreover, Tianjic enables the concurrent deployment of multiple expert networks within one chip, including most types of SNNs and ANNs. With the help of flexibly reconfigurable coding schemes, it supports heterogeneous neural networks with a deep fusion of the two paradigms. For example, Tianjic can easily deploy a large-scale SNN with numerous dendritic branches. Conventionally, the number of allowed synaptic inputs is identical for each FCore (for example, fewer than 256 inputs), which normally limits the model accuracy if only binary spike signals are used for interFCore communication, owing to the severe loss of precision during dendritic integration that faces oversized neuronal fan-ins. By configuring some of the FCores in ANN mode to accumulate membrane potentials in higher precision (acting as a relay for dendritic trees) rather than just communicating via spikes in SNN mode, Tianjic enables the implementation of large-scale SNNs with a high accuracy that stems from the direct transfer of intermediate membrane potentials. As shown in Fig. 3e, the hybrid neural network with dendritic relays shows an improvement in accuracy of 11.5% over that of an SNN-only configuration, without needing to increase the number of neurons. The extra overheads caused by this hybrid paradigm are negligible because Tianjic can naturally implement heterogeneous conversion within the FCore. More comprehensive analysis can be found in the Methods.

The use of Tianjic also enables the exploration of more biologically plausible cognitive models. For instance, Tianjic can implement the continuous attractor neural network (CANN)[30], the synfire chain[31] and dendritic multicompartment models[32] (Fig. 3f). We also developed a software tool that converts the multimodal and hybrid networks to meet the hardware constraints of the Tianjic chip automatically. In general, Tianjic adopts a non-von Neumann paradigm with hybrid compatibility, many-core architecture, localized memory and streamlined dataflow, which is able to support cross-paradigm modelling, maximize parallelism, and improve power efficiency.

To demonstrate the utility of building a brain-like cross-paradigm system, we designed an unmanned bicycle experiment by deploying multiple specialized networks in parallel within one Tianjic chip. Equipped with versatile algorithms and models, the bicycle was able to perform real-time object detection, tracking, voice-command recognition, riding over a speed bump, obstacle avoidance, balance control and decision making (Fig. 4a). Realizing these functions involved three major challenges: first, detecting and smoothly tracking a moving human in an outdoor natural environment, riding over a speed bump, and automatically avoiding obstacles when necessary; second, generating real-time motor control signals in response to balance control, voice commands and visual perception to keep the bicycle moving in the correct direction; and third, realizing

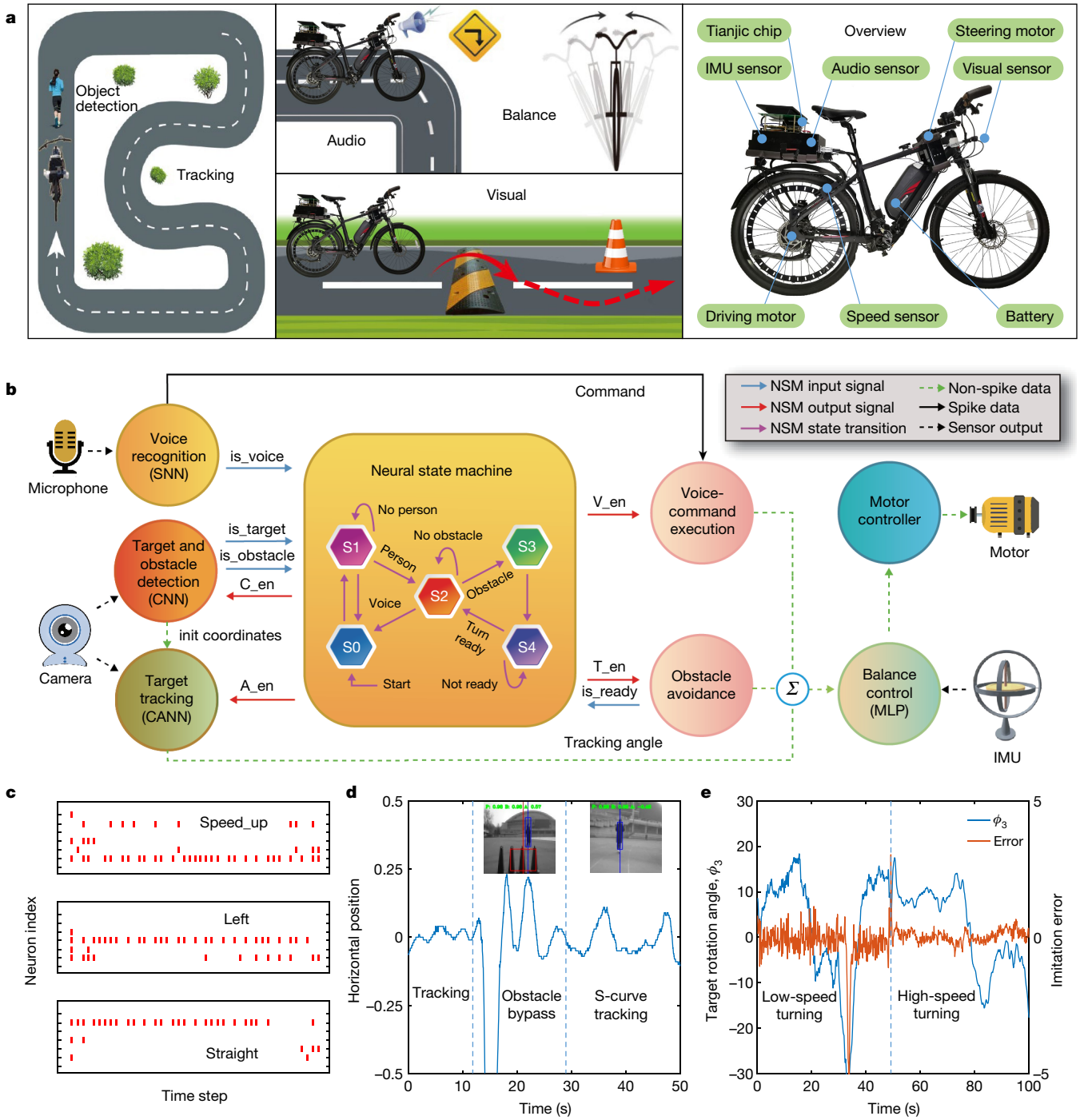


Fig. 4 | Demonstration of multimodal integration on a Tianjic chip for an unmanned bicycle. a, Left and centre, illustration of the tasks conducted in the bicycle experiment, including real-time object detection, tracking, voice perception, riding over a speed bump, automatic obstacle avoidance and balance of attitude. Right, the bicycle was equipped with a camera, gyroscope, speedometer, motors and a Tianjic chip. IMU, inertial measurement unit. **b**, Diagram of the multiple neural networks used in the unmanned bicycle experiment. The states inside the NSM diagram were defined as: voice command execution (S0), human detection (S1), human tracking (S2), a start of obstacle avoidance (S3), and a wait of avoidance completion (S4). An init coordinate is an initialization coordinate. C_en, A_en, V_en and T_en denote enabling signals for CNN, CANN, voice control and turning control, respectively. **c**, SNN voice-command-recognition test. The neuron producing the most spikes indicates the resulting classification. **d**, Tracking test. The *y* axis shows the relative horizontal position of the human in the frame. The bicycle automatically avoided an obstacle, then followed an instructor who ran in an S-curve route. **e**, Balance control and turning by an MLP network, which was trained by imitating the outputs of several well tuned controllers using the proportional-integral-derivative algorithm at different speeds (low to high).

multimodal information integration and prompt decision-making. To accomplish this task, we developed several neural networks, including a CNN for image processing and object detection, a CANN for human target tracking, an SNN for voice-command recognition, and an MLP for attitude balance and direction control (Fig. 4b). Here, the CANN utilizes the membrane-potential normalization mechanism implemented via nonlinear dendritic operations. To integrate these networks and achieve high-level decision-making, we developed an SNN-based neural state machine (NSM). The NSM receives the inputs from other networks (CNN, SNN), and outputs enabling signals (CNN, CANN) and action signals (for example, forced turn, obstacle avoidance) to downstream FCores for bicycle motor control.

Five discrete states were trained for the experiment (see Methods and Extended Data Table 3).

Before the road test, the CNN, CANN, SNN and MLP networks were pretrained and programmed onto the Tianjic chip. Because of its decentralized architecture and arbitrary routing topology, Tianjic allowed all of the neural network models to operate in parallel and realized seamless communication across the models, enabling the bicycle to accomplish these tasks smoothly (see Supplementary Video). Figure 4c visualizes the output spike signals in response to different voice commands; Fig. 4d presents the tracking performance during obstacle avoidance and the S-curve path; and Fig. 4e illustrates the learning of attitude and steering control at different speeds on the basis of physical measurements. This demonstration provides an excellent experimental platform with which to study the key issues in the iterative evolution of AGI. For example, problems of high spatiotemporal complexity can be generated by randomly introducing new variables into the environment in real time, such as different road conditions, noises, weather factors, multiple languages, more people and so on. By exploring solutions that allow adaptation to these environmental changes, issues critical to AGI—such as generalization, robustness and autonomous learning—can be examined.

In summary, we have developed the Tianjic chip, which supports both computer-science-based, machine-learning algorithms and neuroscience-based, biologically inspired models simultaneously. Various neural networks and hybrid coding schemes can be freely integrated, allowing for seamless communication among multiple networks, including SNNs and ANNs. Our research has examined a novel neuromorphic architecture that offers flexibility by integrating cross-paradigm models and algorithms onto a single platform; we hope that our findings will accelerate the development of AGI, with many possible real-world applications.

Online content

Any methods, additional references, Nature Research reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41586-019-1424-8.

Received: 20 May 2018; Accepted: 7 May 2019;
Published online 31 July 2019.

1. Goertzel, B. Artificial general intelligence: concept, state of the art, and future prospects. *J. Artif. Gen. Intell.* **5**, 1–48 (2014).
2. Benjamin, B. V. et al. Neurogrid: a mixed-analog-digital multichip system for large-scale neural simulations. *Proc. IEEE* **102**, 699–716 (2014).
3. Merolla, P. A. et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* **345**, 668–673 (2014).
4. Furber, S. B. et al. The SpiNNaker project. *Proc. IEEE* **102**, 652–665 (2014).
5. Schemmel, J. et al. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Proc. 2010 IEEE Int. Symposium on Circuits and Systems* 1947–1950 (IEEE, 2010).
6. Davies, M. et al. Loihi: a neuromorphic manycore processor with on-chip learning. *IEEE Micro* **38**, 82–99 (2018).
7. Chen, Y.-H. et al. Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE J. Solid-State Circuits* **52**, 127–138 (2017).
8. Jouppi, N. P. et al. In-datacenter performance analysis of a tensor processing unit. In *2017 ACM/IEEE 44th Annual Int. Symposium on Computer Architecture* 1–12 (IEEE, 2017).
9. Markram, H. The blue brain project. *Nat. Rev. Neurosci.* **7**, 153–160 (2006).
10. Izhikevich, E. M. Simple model of spiking neurons. *IEEE Trans. Neural Netw.* **14**, 1569–1572 (2003).
11. Eliasmith, C. et al. A large-scale model of the functioning brain. *Science* **338**, 1202–1205 (2012).
12. Song, S., Miller, K. D. & Abbott, L. F. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nat. Neurosci.* **3**, 919–926 (2000).
13. Gusfield, D. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology* (Cambridge Univ. Press, 1997).
14. Qiu, G. Modelling the visual cortex using artificial neural networks for visual image reconstruction. In *Fourth Int. Conference on Artificial Neural Networks* 127–132 (Institution of Engineering and Technology, 1995).
15. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
16. Russell, S. J. & Norvig, P. *Artificial Intelligence: A Modern Approach* (Pearson Education, 2016).
17. He, K. et al. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition* 770–778 (IEEE, 2016).
18. Hinton, G. et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Process. Mag.* **29**, 82–97 (2012).
19. Young, T. et al. Recent trends in deep learning based natural language processing. *IEEE Comput. Intell. Mag.* **13**, 55–75 (2018).
20. Silver, D. et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).
21. Lake, B. M. et al. Building machines that learn and think like people. *Behav. Brain Sci.* **40**, e253 (2017).
22. Hassabis, D. et al. Neuroscience-inspired artificial intelligence. *Neuron* **95**, 245–258 (2017).
23. Marblestone, A. H., Wayne, G. & Kording, K. P. Toward an integration of deep learning and neuroscience. *Front. Comput. Neurosci.* **10**, 94 (2016).
24. Lillicrap, T. P. et al. Random synaptic feedback weights support error backpropagation for deep learning. *Nat. Commun.* **7**, 13276 (2016).
25. Roelfsema, P. R. & Holtmaat, A. Control of synaptic plasticity in deep cortical networks. *Nat. Rev. Neurosci.* **19**, 166–180 (2018).
26. Ullman, S. Using neuroscience to develop artificial intelligence. *Science* **363**, 692–693 (2019).
27. Xu, K. et al. Show, attend and tell: neural image caption generation with visual attention. In *Int. Conference on Machine Learning* (eds Bach, F. & Blei, D.) 2048–2057 (International Machine Learning Society, 2015).
28. Zhang, B., Shi, L. & Song, S. in *Brain-Inspired Robotics: The Intersection of Robotics and Neuroscience* (eds Sanders, S. & Oberst, J.) 4–9 (Science/AAAS, 2016).
29. Sabour, S., Frosst, N. & Hinton, G. E. Dynamic routing between capsules. *Adv. Neural Inf. Processing Syst.* **30**, 3856–3866 (2017).
30. Mi, Y. et al. Spike frequency adaptation implements anticipative tracking in continuous attractor neural networks. *Adv. Neural Inf. Processing Syst.* **27**, 505–513 (2014).
31. Herrmann, M., Hertz, J. & Prügel-Bennett, A. Analysis of synfire chains. *Network* **6**, 403–414 (1995).
32. London, M. & Häusser, M. Dendritic computation. *Annu. Rev. Neurosci.* **28**, 503–532 (2005).

METHODS

Description of the unified model. Tianjic is a specialized platform that supports most of today's neural network models across neuroscience and computer-science domains, which normally use distinct ways to represent information. We have re-examined most widely used neural network models in the neuroscience domain (for example, SNN, rate-based bio-inspired neural networks and so on) and computer-science domain (for example, MLP, CNN and RNN), and we have proposed a unified description to align the model implementations to axon, synapse, dendrite, soma and router compartments. We identified the similarities and differences between ANN and SNN neurons, and arranged the operations and transformations into these compartments according to their respective functions (see Extended Data Table 1). By aligning the dataflow, Tianjic can flexibly implement various models in a single or hybrid paradigm.

Design philosophy. Tianjic adopts a many-core architecture with massive parallelism. Each FCore includes several blocks: axon, dendrite (with synapse), soma and router. Three key designs that enable us to achieve the hybrid paradigm are:

Independently reconfigurable axon and soma. The axon and soma can be configured into different modes independently. The axon receives and organizes SNN inputs or ANN inputs according to its mode configuration. Similarly, the soma generates SNN outputs or ANN outputs according to its mode configuration. When the axon and soma are configured into the same operating mode (either ANN or SNN), the FCore acts in pure ANN or SNN mode, respectively, which we call a single-paradigm FCore. When the axon and soma are configured into different operating modes, the FCore processes ANN inputs and fires SNN outputs, or processes SNN inputs and generates ANN outputs; we call this a hybrid FCore.

Shared dendritic integration. The dendritic integrations for processing SNN inputs and ANN inputs share the same calculators (multipliers and accumulators), although they have different processing operations and style. At each time phase, the dendrite performs intense MACs when processing ANN inputs. When processing SNN inputs, the dendrite also performs MACs when the length of the integration-time window is greater than one; if this time window is less than one, the dendrite performs only addition operations and bypasses multipliers; if no spike is received, the dendrite skips all operations.

Unified routing infrastructure. The interconnections of FCores share the same routing infrastructure to transfer routing packets in a unified format. The pre-soma can package the output into either an SNN or an ANN packet according to the soma configuration, and the post-axon parses the routing packet into either SNN or ANN format on the basis of its axon configuration.

Hybrid configuration. At the network level, besides constructing conventional single-paradigm ANNs or SNNs, Tianjic offers hybrid modelling at two levels of granularity. At the coarse-grain level, we can configure some FCores in ANN mode and other FCores in SNN mode to perform ANNs and SNNs individually. In this way, we can realize both ANNs and SNNs on the same hardware platform, operating concurrently without the need to build a heterogeneous system using various chips. This also simplifies the design of off-chip communication interfaces on the circuit board, because Tianjic uses a unified packet format and routing infrastructure. At the fine-grain level, we can configure FCores working in hybrid mode (axon in ANN and soma in SNN, or vice versa) to build a network with hybrid signal coding.

Note that, to achieve a particular target application, the working modes of axon and soma are fixed after the initial configuration, without any mode switching during the consequent execution process. No extra switching circuits and overheads are required for signal conversion in the hybrid mode, because it is naturally performed on the critical data path of FCore.

Architecture design. A schematic view of the FCore architecture is shown in Extended Data Fig. 1. The cross-paradigm computing of Tianjic is realized through shared dendrites, independently reconfigurable axons and somas, and a unified interconnection infrastructure.

In the dendrite block, the processing neurons are divided into multiple groups (for example, 16 groups). The groups run in series and the neurons within the group are executed in parallel during operation. In the dendrite module, there are 16 8-bit multipliers and 16 24-bit accumulators to support the vector-matrix multiplication (VMM) operations, which are shared by ANN and SNN modes. In ANN mode, the dendrite module reads one input from the axon module and 16 8-bit weights from the synapse memory simultaneously at each clock cycle, then concurrently executes 16 MACs for 16 neurons that share the same axon data. Each neuron completely performs 256 multiplications with 8-bit precision, and the same amount of accumulations with 24-bit precision. In SNN mode, when the temporal window length (T_w) for controlling the duration of historical spike integration is larger than one, the multipliers and accumulators are used in the same way as in ANN mode; otherwise, the axon module outputs only 1-bit spikes and the dendrite module skips the multipliers.

The axon and the soma modules are essential for the hybrid operation. They can be flexibly and independently reconfigured between ANN and SNN modes as discussed. In ANN mode, the axon memory is divided into two chunks, working as ping-pong buffers. In SNN mode, the two buffers are merged as a complete chunk to store spike patterns within a historical temporal window with tunable duration. Extra memories are used to buffer the latest spikes in a ping-pong manner. In addition, a timing-factor calculator is implemented for the time decay.

In ANN mode, the data in soma flow in a 'bias, activation function, output transmission' fashion. The 25-bit biased activation values are truncated by a reconfigurable 10-bit sliding window. Then a reconfigurable LUT with 10-bit entries and 8-bit outputs is applied in each FCore for arbitrary activation function. In SNN mode, the dataflow changes to 'potential leakage, spike generation, output transmission'. The 25-bit membrane potential (with fixed or adaptive leakage) is truncated and then compared with a 24-bit threshold to determine whether a spike is fired or not. If a spike is fired, multiple modes of membrane potential reset are supported. All memories are shared between the two modes for storing inputs/outputs or parameters such as synaptic weights, biases, LUTs, leakages and thresholds.

The soma outputs are packaged into routing packets. For interFCore transmission, ANN and SNN modes share the same routing packet format, consisting of control, address and data segments. The control segment determines whether the input is an inhibitory signal in the SNN mode. The address segment contains the destination FCore and memory-cell addresses. The data segment conveys either 8-bit activation in the ANN mode or nothing in the SNN mode (the packet itself represents a spike event[33]). Once received, the post-axon parses the packet as ANN or SNN signal according to the axon configuration. The control and address segments of neurons are stored in a reconfigurable 1-kilobyte (KB) routing LUT of each router with five communication channels: local, eastern, western, southern and northern.

Tianjic adopts a 2D-mesh many-core architecture to construct large networks hierarchically (core-chip-board-system). Besides the normal P2P routing scheme[3], an adjacent multicast (AMC) routing scheme is designed to expand the fan-outs. When an FCore is configured into the AMC routing mode, the received packets are delivered to the next FCore according to the configured multicast direction and distance. In this way, multiple FCores can receive duplications through routing relays.

Tianjic also flexibly supports more special operations. (1) Nonlinear integration: in addition to the MACs between the dynamic input vector and static weight matrix, the dendrite module can also perform operations between dynamic input vectors. (2) Somatic cooperation: adjacent somas can be merged into a stronger soma that accumulates all integrations of every soma, thus expanding fan-ins without using additional FCores. (3) Ternary synapse: Tianjic supports the ternary neural networks[34] by reducing the bit width of each synapse to two, and then the number of synapses increases accordingly. (4) Connection extensions for scaling up networks, including, first, fan-in extension through the abovementioned intraFCore soma cooperation or extra interFCore hierarchical integration; and second, fan-out extension through neuron copy or multicast routing scheme.

Chip specification. The Tianjic chip was fabricated using 28-nm high performance low power (HLP) technology. The layout, physical picture, and testing boards are shown in Extended Data Fig. 2. One Tianjic chip consists of 156 FCores and the weight-sharing technique[3,35] was leveraged. The number of weight indexes (M) and fan-ins/fan-outs (N) are set as 32 and 256, respectively, leading to a total number of roughly 22 KB static random-access memory (SRAM) in each FCore. The configuration and performance of the Tianjic chip are summarized in Extended Data Table 2 and compared with existing neural network platforms. At a clock frequency of 300 MHz and supplied voltage of 0.85 V, the Tianjic chip typically consumes 6.1 mW and 5.5 mW per FCore in ANN mode and SNN mode ($T_w = 1$), respectively. Tianjic requires 5,050 clock periods (a clock period being 16.8 μs at 300 MHz) to complete a round of computations and communications, which reflects the minimum phase latency.

Network deployment. Tianjic provides great flexibility in network deployment. Specifically, most spiking and non-spiking neural networks can be constructed from the same basic topological layers, including fully connected, convolutional, pooling and recurrent layers. During network deployment, the weights are partitioned and pinned into the synapse memories of FCores, which remain unchanged and do not need reloads given fixed working modes and static network topologies after initialization. To balance processing throughput and resource overhead, Tianjic supports two mapping schemes: unfolded mapping and folded mapping (Extended Data Fig. 3).

The unfolded mapping scheme converts all topologies into fully connected structures without resource reuse. The FCores performing VMM operations share the inputs through the multicast routing strategy, and other FCores for accumulating the partial sums (called reduce operations) are additionally required. For CANN, the differential dynamics are converted to difference equations, similar to the iterative format of LSTM. Besides matrix operations, vector operations (for example, generating spike rates in CANN or updating cell/hidden states in LSTM) can be realized by the nonlinear dendritic integration. On the other hand,

a folded mapping scheme is supported to reduce the resource overhead of non-spiking convolutions with massive data reuse and independency of historical information. It requires two types of FCores: buffer FCores and computation FCores. The weights are shared in convolutions along the row dimension of feature maps, which reside in the synapse memory for reuse. The inputs are accordingly arranged row by row using buffer FCores before each convolution operation performed in the consequent computation FCores. In this way, the convolutions are pipelined by generating one output row at each phase, and such row-wise streaming can tolerate CNNs with excess rows.

System support. Inspired by the hierarchy of computer systems, Tianjic's software tool chain contains similar levels to the host computer to facilitate applications, such as a unified abstraction for programming and an automatic compiler for mapping. The software supports applications in both SNN and ANN modes. For SNNs, it supports two training methods: indirect and direct training. The indirect training uses popular deep-learning frameworks to train an ANN model using back propagation, and then converts it into its SNN counterpart[36]. The direct training uses the emerging spatiotemporal back-propagation algorithm[37] to directly train the SNN model. For the non-spiking ANNs, besides the common workflow that modifies and maps existing networks manually, our compiler can also automatically transform a pretrained model into an equivalent network that meets the Tianjic hardware constraints, thus decoupling the applications from the target hardware.

Single-paradigm evaluation. Here we detail the benchmarks used to evaluate single-paradigm neural networks in Fig. 3d. For SNN, we used a fully connected network with the NMNIST dataset[38]. The dimensions of input data are $(34 \times 34) \times 2 = 2{,}312$ for both ON-type and OFF-type spikes, and the network structure was 2,312-800-10. For MLP, we used the fully connected layers of AlexNet (9,216-4,096-1,000)[39] and VGG16 (25,088-4,096-1,000)[40] on the ImageNet dataset[41]. For CNN, we used the networks of LeNet-variant[34] on the MNIST dataset[42], VGG8[43] on the CIFAR10 dataset[44], and AlexNet/VGG16/ResNet18[17] on the ImageNet dataset. For LSTM, we used two customized networks with one hidden layer of 1,024 cells on the WikiText-2 dataset[45] and two hidden layers of 512 cells for each on the Tiny-Shakespeare dataset[46]. The average performance across multiple networks under the same structure category was reported. Direct training with a spike-rate coding scheme was used to train the spiking model, and quantized training[47] with 8-bit weights and activations was used to train the non-spiking models.

For small-scale networks, including the ANN/SNN hybrid example in Fig. 3e (with a structure of input-20C3-AP2-20C2-AP2-10C2-10) and all models in the bicycle experiment, we directly measured the accuracy, latency and power on the testing board. We ran large networks that exceed present resources, such as those in Fig. 3d, in a cycle-accurate simulator whose accuracy and latency can correspond one-to-one with the hardware. The power estimation in these cases was extrapolated on the basis of measured results from a single chip, including both leakage and active power. In the simulation, we relaxed two fabrication cost constraints: we removed the weight-sharing technique for a full degree of synapses, and ignored the interchip communication overhead because it is easy to integrate more FCores into one single chip to accommodate larger networks[3]. The programming frameworks for the Titan-Xp GPU results were Pytorch[48] (SNN, MLP, CNN) and Torch (LSTM). For LSTM, we used the benchmarks of DeepBench[49] to measure the running performance. The batch size was set to one as suggested[50] for inference tasks.

Implementation of a hybrid network. We rigorously compared the performance between single-paradigm (ANN-only or SNN-only) and fine-grained cross-paradigm (ANN/SNN hybrid) implementations using the same network structure as in Fig. 3e. The benchmark dataset was MNIST. As well as using the afore-mentioned training methods for ANN-only and SNN-only models, we used a hybrid model adapted from a pretrained SNN model. For the SNN-only and hybrid modes, the original pixel values were converted to spike events through Bernoulli sampling. In the hybrid-mode network, the hybrid layer used 'SNN-input and ANN-output' FCores to integrate SNN spikes and generate ANN signals (high-precision intermediate membrane potentials), and then used 'ANN-input and SNN-output' FCores to accumulate these ANN signals and fire SNN spikes again. In this example, the ANN-only, SNN-only and hybrid models all used the unfolded mapping scheme, and for simplicity only the last convolution layer was configured into hybrid mode.

Extended Data Figs. 4, 5 present comprehensive comparisons among different configuration modes and detailed routing profiling. The extra hardware cost introduced by the hybrid model beyond the SNN-only model was negligible, and the costs of both the hybrid model and the SNN-only model were much lower than that of the computation-intensive ANN-only model. Moreover, because it avoided the loss of precision during communication that is induced by binary spikes when the number of inputs exceeds the fan-in limitation of neurons, the hybrid model inherited the accuracy advantage of the ANN-only model. In this example, the hybrid paradigm provides an efficient way to combine the advantages of both SNN-only and ANN-only models, leading to an overall high performance. In short, the hybrid paradigm of Tianjic can be used to construct various ANN and SNN combinations according to the requirements of practical applications, such as low power consumption, high speed, high accuracy or overall high performance.

Unmanned bicycle demonstration. The detailed network topologies are listed in Extended Data Table 3. The model sizes were trimmed and optimized for the demonstration of system functionality on a single Tianjic chip. Data preprocessing was performed on an ARM processor, and all of the models were trained with quantization before they were implemented on Tianjic. The mapping results of these networks and the measured power consumption are presented in Extended Data Fig. 6.

Each frame from a web camera was rescaled to a 70×70 greyscale image as CNN inputs, and clipped before being injected into CANN as inputs according to the initial bounding boxes produced by the CNN model. For CANN, we applied a modified model that reduces remote connections outside the scope of the Gaussian bump. We converted the audio signals obtained from the microphone into 51-dimension features with different frequencies using the Mel-frequency cepstral coefficient (MFCC)[51] method. We then applied the Gaussian population coding strategy[52] to encode each frequency feature into ten spiking units. We adopted the fully connected structure based on leaky integrate-and-fire (LIF)[53] neurons to construct the SNN model trained by the direct method as above. The seven output neurons represented six instruction commands and a noise case. For attitude and motion control, we selected five signals produced by the inertial measurement unit (IMU). The signals during the last sixth of the time steps were concatenated to form a 30-dimension vector as the input of MLP. We trained the MLP network to imitate the outputs of several proportional-integral-derivative controllers under different speeds (low to high). The output target rotation angle was sent to the motor controller to tune the steering for balance keeping. In addition, the voltage coefficient was sent directly to another motor (back wheel) controller to adjust the speed.

Inspired by prior work[54], we designed an SNN-based finite neural state machine (NSM; Extended Data Fig. 7) to integrate the outputs of the above diverse neural networks. The FCores between the decision-making module (that is, NSM) and action module (that is, MLP) worked in a hybrid mode (SNN-input and ANN-output), which contained trajectory patterns in the soma LUT to convert action signals (spikes) into target inclination angle sequences (real values). In this way, the NSM could guide the bicycle to achieve forced turn (voice command) or obstacle avoidance. The weights of the NSM model were trained following the offline spike-timing-dependent plasticity (STDP)-like rule[12].

Data availability

The datasets that we used for benchmarks are publicly available, as described in the text and the relevant references[38,41,42,44–46]. The training methods are provided in the relevant references[36,37,47,54]. The experimental setups for simulations and measurements are detailed in the text. Other data that support the findings of this study are available from the corresponding author on reasonable request.

Code availability

The codes used for the software tool chain and the bicycle demonstration are available from the corresponding author on reasonable request.

33. Imam, N. & Manohar, R. Address-event communication using token-ring mutual exclusion. In *2011 17th IEEE Int. Symposium on Asynchronous Circuits and Systems* 99–108 (IEEE, 2011).
34. Deng, L. et al. GXNOR-Net: training deep neural networks with ternary weights and activations without full-precision memory under a unified discretization framework. *Neural Netw.* **100**, 49–58 (2018).
35. Han, S. et al. EIE: efficient inference engine on compressed deep neural network. In *2016 ACM/IEEE 43rd Annual Int. Symposium on Computer Architecture* 243–254 (IEEE, 2016).
36. Diehl, P. U. et al. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 Int. Joint Conference on Neural Networks* 1–8 (IEEE, 2015).
37. Wu, Y. et al. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Front. Neurosci.* **12**, 331 (2018).
38. Orchard, G. et al. Converting static image datasets to spiking neuromorphic datasets using saccades. *Front. Neurosci.* **9**, 437 (2015).
39. Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **25**, 1097–1105 (2012).
40. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *Int. Conference on Learning Representations*; preprint at https://arxiv.org/pdf/1409.1556.pdf (2015).
41. Deng, J. et al. ImageNet: a large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* 248–255 (IEEE, 2009).

42. LeCun, Y. et al. Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998).
43. Courbariaux, M., Bengio, Y. & David, J.-P. BinaryConnect: training deep neural networks with binary weights during propagations. *Adv. Neural Inf. Processing Syst.* **28**, 3123–3131 (2015).
44. Krizhevsky, A. & Hinton, G. *Learning Multiple Layers of Features from Tiny Images*. MSc thesis, Univ. Toronto (2009).
45. Merity, S. et al. Pointer sentinel mixture models. In *Int. Conference on Learning Representations*; preprint at https://arxiv.org/abs/1609.07843 (2017).
46. Krakovna, V. & Doshi-Velez, F. Increasing the interpretability of recurrent neural networks using hidden Markov models. Preprint at https://arxiv.org/abs/1606.05320 (2016).
47. Wu, S. et al. Training and inference with integers in deep neural networks. In *Int. Conference on Learning Representations*; preprint at https://arxiv.org/abs/1802.04680 (2018).
48. Paszke, A. et al. Automatic differentiation in Pytorch. In *Proc. NIPS Autodiff Workshop* https://openreview.net/pdf?id=BJJsrmfCZ (2017).
49. Narang, S. & Diamos, G. Baidu DeepBench. https://github.com/baidu-research/DeepBench (2017).
50. Fowers, J. et al. A configurable cloud-scale DNN processor for real-time AI. In *2018 ACM/IEEE 45th Annual Int. Symposium on Computer Architecture* 1–14 (IEEE, 2018).
51. Xu, M. et al. HMM-based audio keyword generation. In *Advances in Multimedia Information Processing – PCM 2004*, Vol. 3333 (eds Aizawa, K. et al.) 566–574 (Springer, 2004).
52. Mathis, A., Herz, A. V. & Stemmler, M. B. Resolution of nested neuronal representations can be exponential in the number of neurons. *Phys. Rev. Lett.* **109**, 018103 (2012).
53. Gerstner, W. et al. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition* (Cambridge Univ. Press, 2014).
54. Liang, D. & Indiveri, G. Robust state-dependent computation in neuromorphic electronic systems. In *IEEE Biomedical Circuits and Systems Conference* 1–4 (IEEE, 2017).
55. Akopyan, F. et al. TrueNorth: design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Trans. Comput. Aided Des. Integrated Circ. Syst.* **34**, 1537–1557 (2015).
56. Han, S. et al. ESE: efficient speech recognition engine with sparse LSTM on FPGA. In *Proc. 2017 ACM/SIGDA Int. Symposium on Field-Programmable Gate Arrays* 75–84 (ACM, 2017).

Acknowledgements We thank B. Zhang, R. S. Williams, J. Zhu, J. Guan, X. Zhang, W. Dou, F. Zeng and X. Hu for thoughtful discussions; L. Tian, Q. Zhao, M. Chen, J. Feng, D. Wang, X. Lin, H. Cui, Y. Hu and Y. Yu contributing to experiments; H. Xu for coordinating experiments; and MLink for design assistance. This work was supported by projects of the National Natural Science Foundation of China (NSFC; 61836004, 61327902 and 61475080); the Brain-Science Special Program of Beijing (grant Z181100001518006); and the Suzhou-Tsinghua innovation leading program (2016SZ0102).

Author contributions J.P., L.D., S.S., M.Z., Y.Z., Shuang Wu and G.W. were in charge of, respectively, the principles of chip design, chip design, the principles of neuron computing, the unmanned bicycle system, software, implementation of Tianjic in the unmanned bicycle system, and chip testing. J.P., L.D., G.W., Z.W. and Y.Z. carried out chip development. Shuang Wu, G.W., Z.Z., Z.Y. and Yujie Wu worked on the unmanned bicycle experiment. Y.Z. and W. Han worked on software development. Yujie Wu, Shuang Wu and G.L. developed the algorithm. J.P., L.D., S.S., Si Wu, C.M., F.C., W. He, R.Z. and L.S. contributed to the analysis and interpretation of results. All of the authors contributed to discussion of architecture design principles. L.D., W. He, R.Z., S.S., Z.W. and L.S. wrote the manuscript with input from all authors. L.S. proposed the concept of hybrid architecture and supervised the whole project.

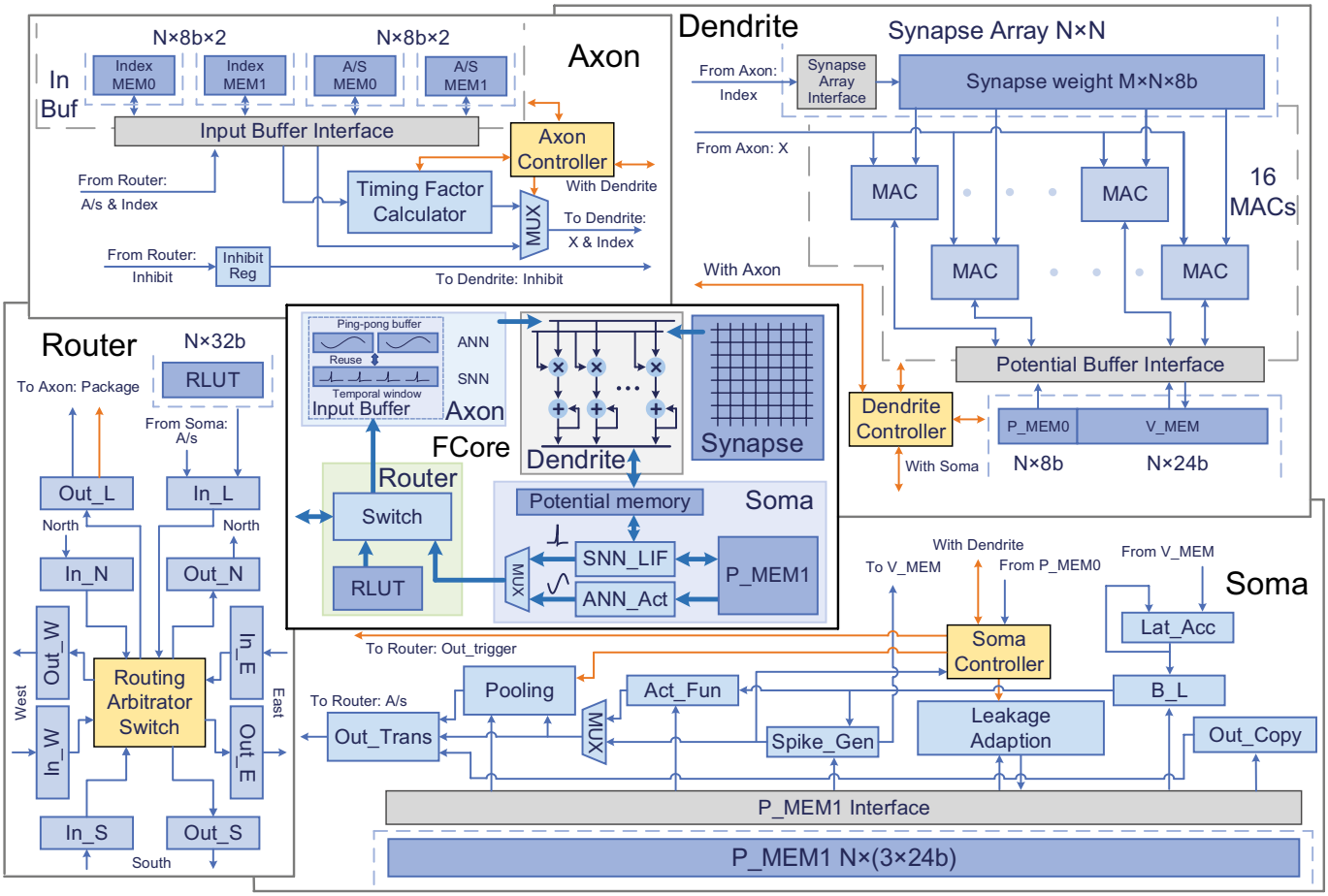
Competing interests The authors declare no competing interests.

Additional information

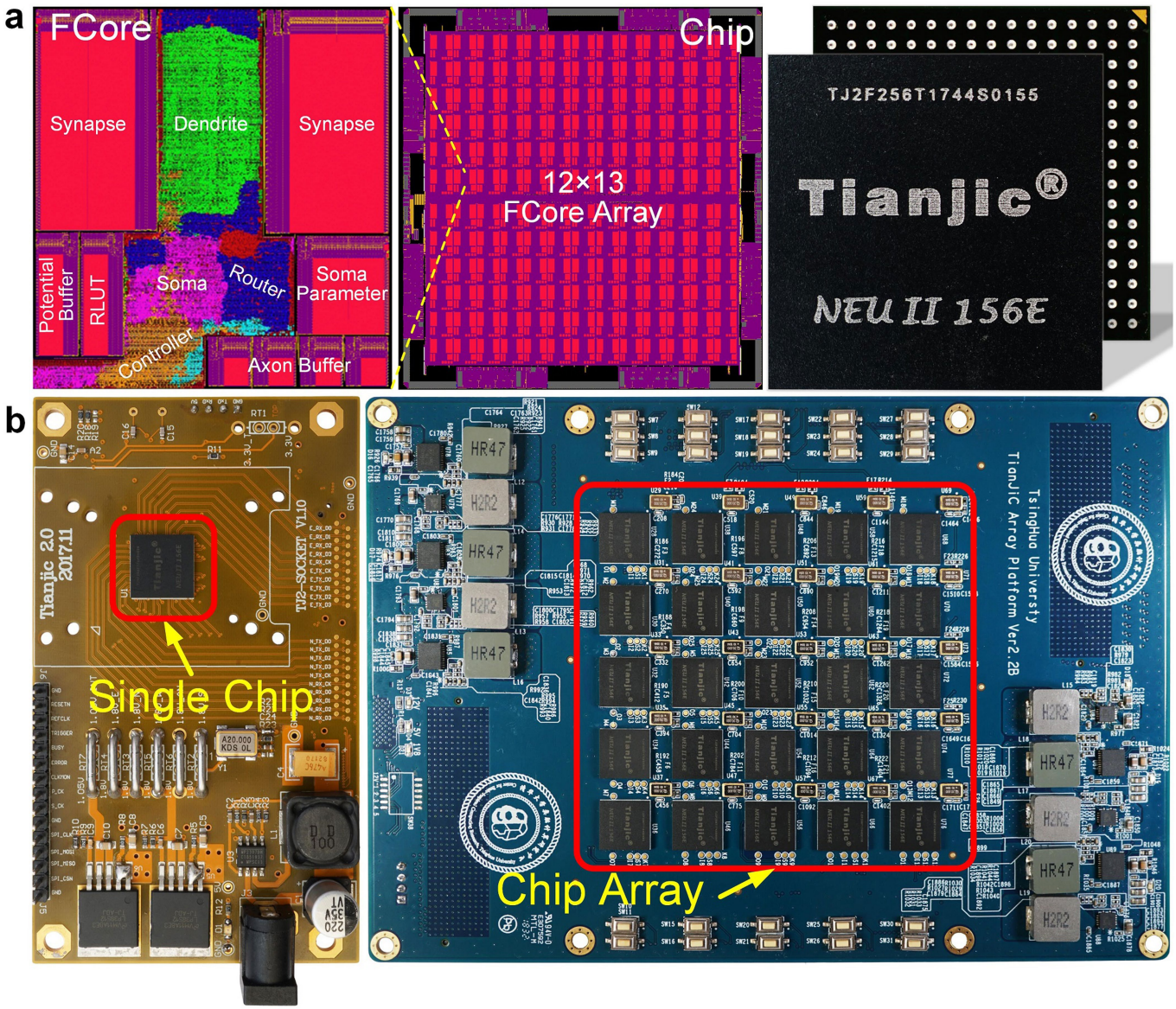
Supplementary information is available for this paper at https://doi.org/10.1038/s41586-019-1424-8.

Correspondence and requests for materials should be addressed to J.P.

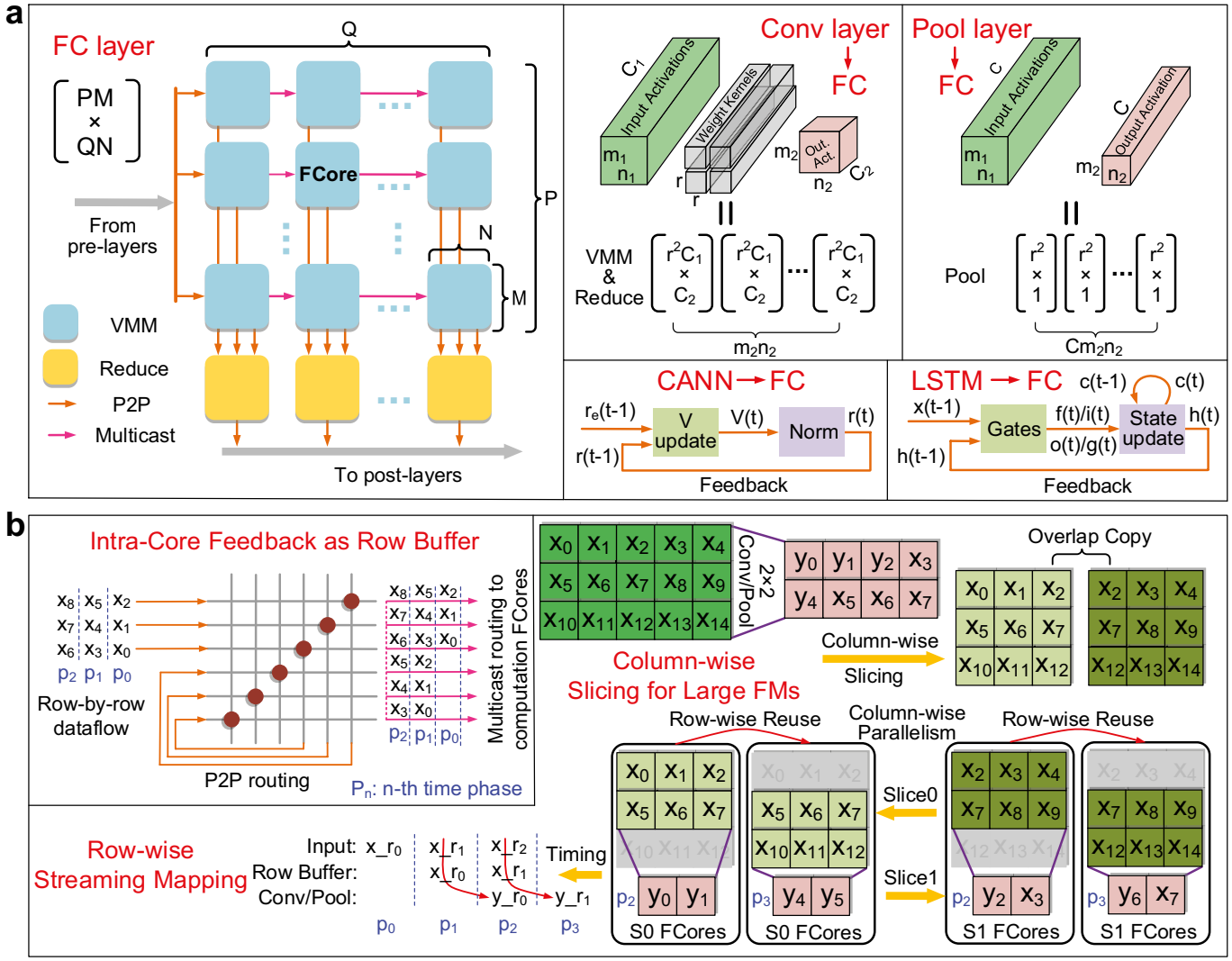
Peer review information *Nature* thanks Meng-Fan (Marvin) Chang and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.



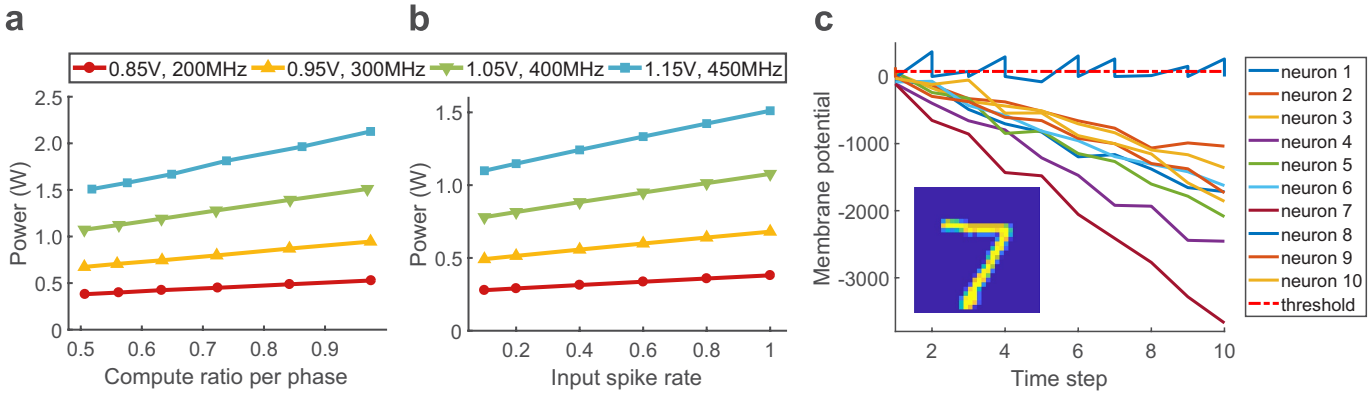
Extended Data Fig. 1 | Overview of the FCore architecture. We adopted a fully digital design. The axon module acts as a data buffer to store the inputs and the outputs. Synapses are designed to store on-chip weights and are pinned close to the dendrite for better memory locality. The dendrite is an integration engine that contains multipliers and accumulators. The soma is a computation unit for neuronal transformations. IntraFCore and interFCore communications are wired by a router, which supports arbitrary topology. Act_Fun, activation function; A/s, activation (ANN mode)/spike (SNN mode); B_L, bias/leakage; In BUF, input buffer; Inhibit Reg, inhibition register; In(Out)_L/E/W/S/N, local/eastern/western/southern/northern input(output); Lat_Acc, lateral accumulation; MEM, memory; MUX, multiplexer; Out_Copy, output copy; Out_Trans, output transmission; P_MEM, parameter memory; Spike_Gen, spike generator; V_MEM, membrane potential memory; X & Index, axon output and weight index. The numbers in or above memories indicate memory size; 'b' represents bit(s).



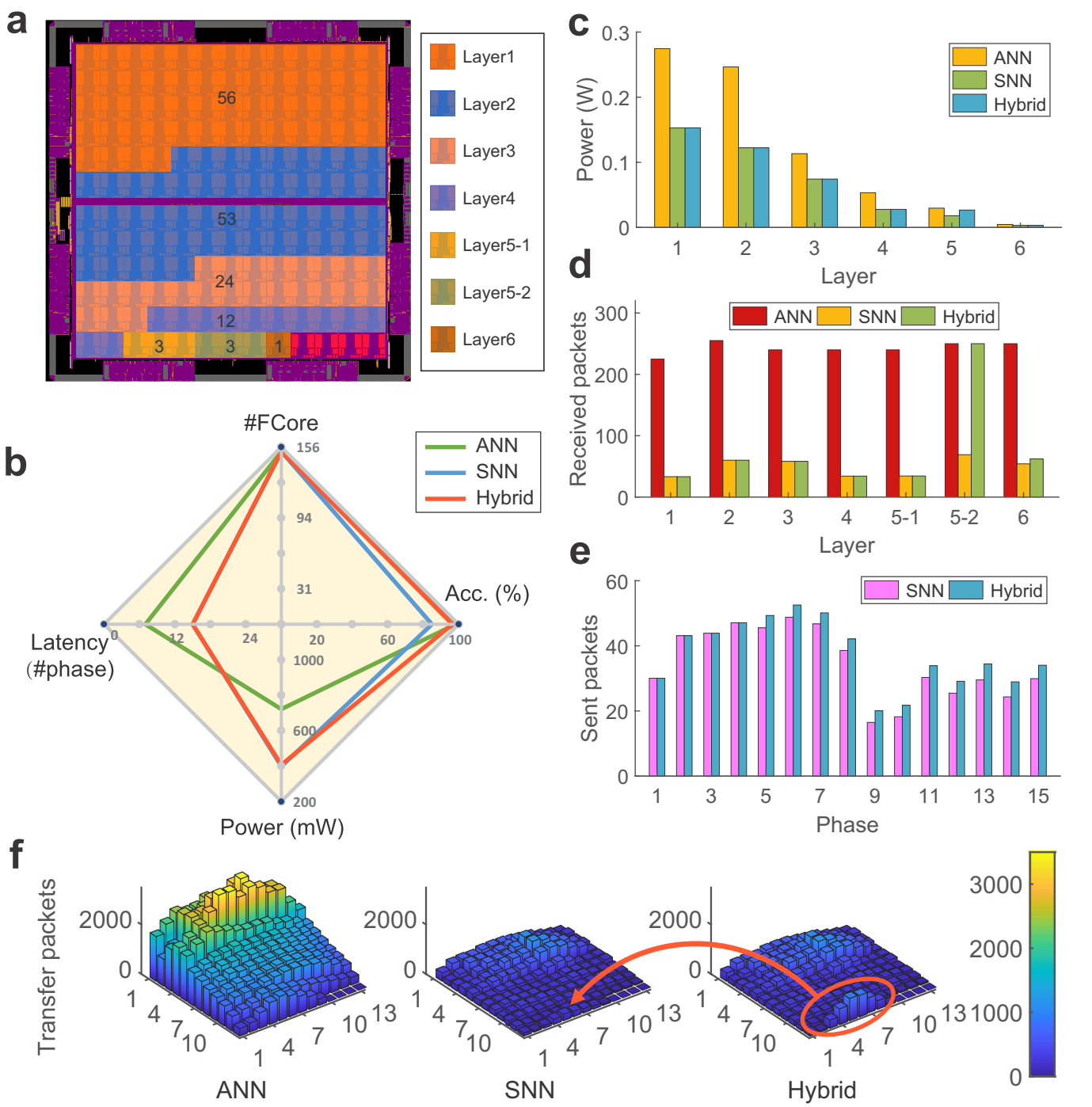
Extended Data Fig. 2 | Fabrication of the Tianjic chip and testing boards. a, Chip layout and images of the Tianjic chip. **b**, Testing boards equipped with a single Tianjic chip or a chip array (5 × 5 size).



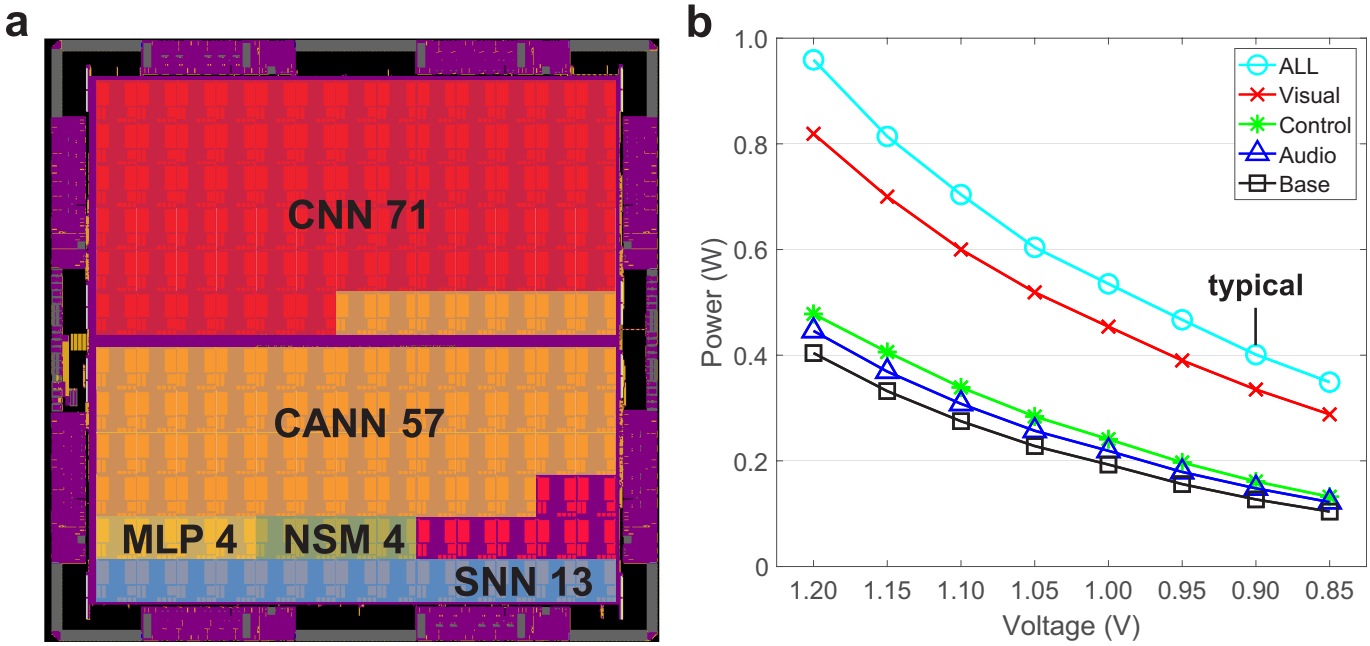
Extended Data Fig. 3 | Throughput-aware unfolded mapping and resource-aware folded mapping. a, Unfolded mapping converts all topologies into a fully connected (FC) structure without reusing data. In CANN: Norm, normalization; r, firing rate; V, membrane potential. In LSTM: f/i/o, forget/input/output gate output; g, input activation; h/c, hidden/cell state; t, time step; x, external input. **b**, Folded mapping folds the network along the row dimension of feature maps (FMs) for resource reuse. We note that the weights are still unfolded along the column dimension to maintain parallelism, and wide FMs can be split into multiple slices, which are allocated into different FCores for concurrent processing. $r_{0/1/2}$, row 0/1/2.



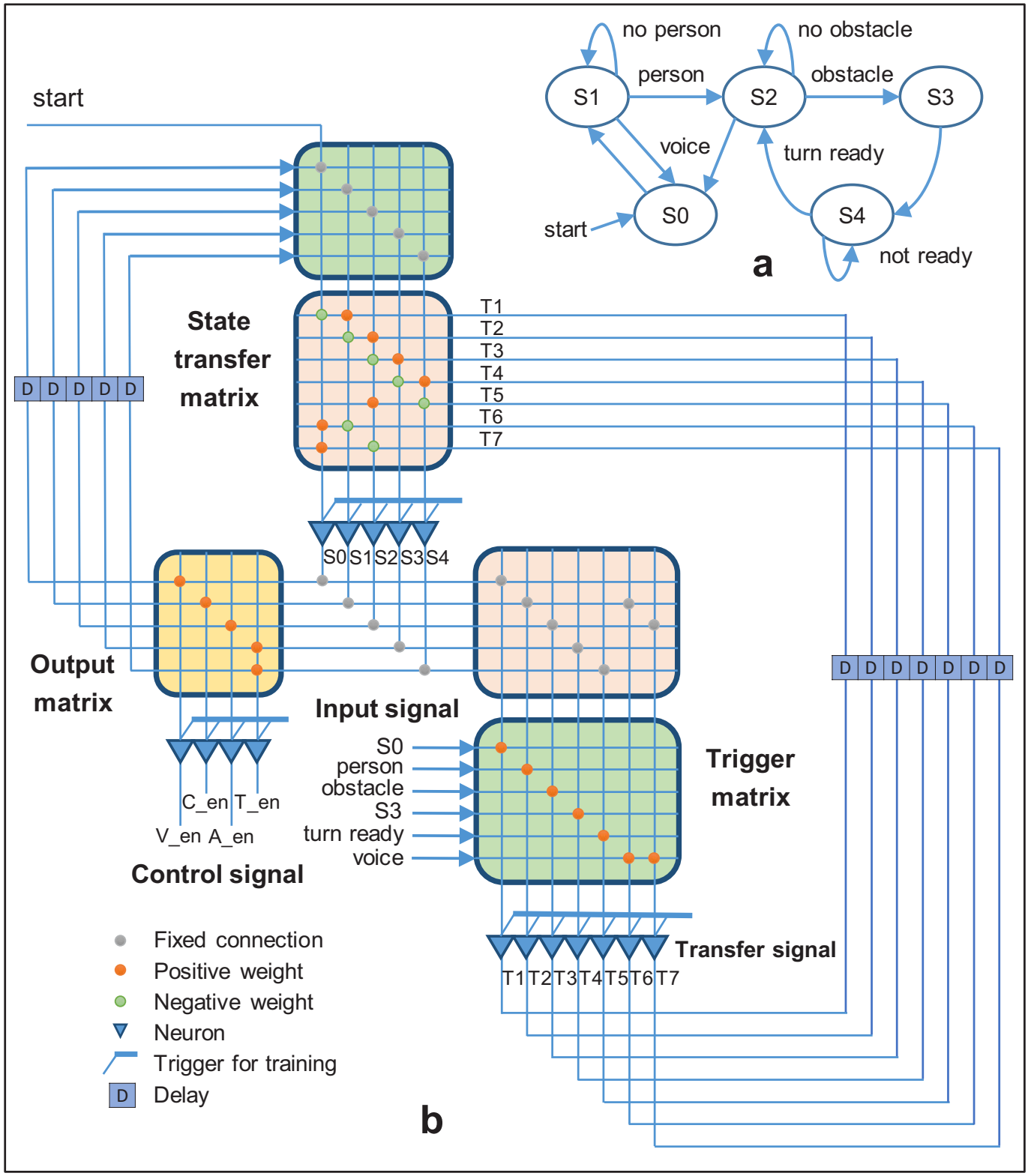
Extended Data Fig. 4 | Chip measurements in different modes. a, Power consumption in ANN-only mode at different voltages and frequencies. Here the 'compute ratio' is the duty ratio for computation, that is, the ratio of computation time/(computation time + idle time). The phase on the *x*- axis denotes the execution time phase of FCore. **b**, Power consumption in SNN-only mode with different rates of input spikes. **c**, Membrane potential of output neurons in SNN mode. Information was represented in a rate-coding scheme by counting the number of spikes during a given time period.



Extended Data Fig. 5 | Performance comparison and routing profiling. **a**, FCore placements in six layers (split into seven execution layers); the numbers within the image denote the numbers of FCores used. **b**, Comparison of the performance of different neural network modes. Acc., accuracy. **c**, Power consumption for each layer. **d**, Average number of received routing packets per FCore in each layer. **e**, Average number of sent packets per FCore across time phases. **f**, Distribution of total transfer packets for each FCore. The oval with the arrow emphasizes the difference in packet amount between the SNN-only mode and the hybrid mode.



Extended Data Fig. 6 | Overheads of the Tianjic chip during the bicycle experiment. a, Placement of FCores in different network models. Numbers refer to the number of FCores used. **b**, Measured power consumption under different tasks and at different voltages. The Tianjic chip typically worked at 0.9 V during the bicycle demonstration, and the power consumption was about 400 mW.



Extended Data Fig. 7 | Neural state machine. a, State transition in the bicycle task. **b**, NSM architecture. The NSM is composed of three subgroups of neurons: state, transfer and output neurons. There are three matrices that determine the connections between different neurons: the trigger, state-transfer and output matrices.

Extended Data Table 1 | A unified description of neural network models

	Neuroscience-oriented model		Computer-science-oriented model	
	SNN	**Rate-based (e.g. CANN)**	**MLP/CNN**	**RNN (e.g. LSTM)**
Axon	$s_e(t)$, S^{T_w}, $s(t)$	$r_e(t)$, $r(t-1)$, $V(t)$, $r(t)$	x, y	$x(t)$, $h(t-1)$, $c(t-1)$, $f/i/o/g(t)$, c(t), $h(t)$
Synapse	Weight matrix	Weight matrix	**Conv:** Weight kernels **Pool:** Bypass **FC:** Weight matrix	**Gates:** Weight matrices **Cell/Hidden state:** Intermediate variables
Dendrite	$\sum_j w_{ij} \sum_{t_j^k \in S_j^{T_w}} K(t - t_j^k) + w_{ie} s_{ie}(t)$	$V_i(t) = \sum_j w_{ij} r_j(t-1) + w_{ie} r_{ie}(t)$ **Parts of spike_rate:** Norm(·)	**Conv:** $\sum_m \mathbf{FM}_m^{in} \circledast \boldsymbol{W}(m,n)$ **Pool:** Bypass **FC:** $\sum_j x_j w_{ij}$	**Gates:** $\sum_p x_p(t) w_{kp}^{f/i/o/gx} + \sum_q h_q(t-1) w_{kq}^{f/i/o/gh}$ **Cell state:** $\boldsymbol{c}(t) = \boldsymbol{c}(t-1) \odot \boldsymbol{f}(t) + \boldsymbol{g}(t) \odot \boldsymbol{i}(t)$ **Hidden state:** $\boldsymbol{h}(t) = \tanh(\boldsymbol{c}(t)) \odot \boldsymbol{o}(t)$
Soma	**Leakage:** $\tau \frac{dV_i(t)}{dt} = -[V_i(t) - V_{r1}] + V_\Sigma$ Thresh_comp, V_update | (Spike_fire & V_reset)	**Parts of spike_rate:** Norm(·)	**Conv:** $\varphi(b_n + \mathbf{FM}_n^\Sigma)$ **Pool:** Max/Average **FC:** $\varphi(b_i + x_i^\Sigma)$	**Gates** $\boldsymbol{f/i/o}$: sigmoid$(b + (x \,\&\, h)^\Sigma)$ **Gate** $\boldsymbol{g}$: $\tanh(b + (x \,\&\, h)^\Sigma)$ **Cell state**: tanh(·)
	Op: B_L, Spike_Gen, Out_Trans	**Op:** Act_Fun, Out_Trans (Lat_Acc)	**Op:** B_L, Act_Fun, Out_Trans (Pooling, Out_copy)	**Op:** B_L, Act_Fun, Out_Trans
Router	Feedforward/Recurrent	Recurrent (Feedforward)	Feedforward (Recurrent)	Recurrent (Feedforward)
mode	Spiking representation (SNN mode)	Non-spiking representation (ANN mode)		

The vector/matrix operations and transformations are assigned into several compartments, including axon, synapse, dendrite, soma and router. Each compartment supports the functions shown; by combining these, various neuroscience-oriented and computer-science-oriented models can be realized. In SNN: *s*, spike; s_e, external spike; Spike_fire, spike firing; Thresh_comp, threshold comparison; τ, time constant; *t*, time step; *V*, membrane potential; V_{r1}, rest potential; V_reset, membrane potential reset; V_update, membrane potential update. In MLP/CNN: *x*/*y*, input/output activation; *b*, bias; $\circledast$, convolution. For other abbreviations and variables, please refer to Extended Data Figs. 1, 3.

Extended Data Table 2 | Comparison of the Tianjic chip with existing specialized platforms

Platform	TrueNorth[3,55]	Loihi[6]	EIE[35]	Eyeriss[7]	ESE[56]	**Tianjic**
Model	SNN	SNN	MLP	CNN	LSTM	**Hybrid**
BitWidth	9W-1S	9W-1S	4W-4A	16W-16A	12W-16A	**8W-8A/1S**
Memory	SRAM	SRAM	SRAM	DRAM	DRAM	**SRAM**
Technology	28 nm	14 nm	45 nm	65 nm	22 nm	**28 nm**
Clock (MHz)	Async	Async	800	100-250	200	**300**
Area (mm^2)	430	60	40.8	16	N. A.	**14.44**
Power (W)	0.063-0.3	N. A.	0.59	0.2-0.3	41	**0.95**
GOPS/W	N. A.	N. A.	174	246	6.9	**1278**
GSOPS/W	400	N. A.	N. A.	N. A.	N. A.	**649**

Tianjic is a specialized platform that can simultaneously support most of today's neural network models across both neuroscience and computer-science domains. For a single chip, the effective peak power efficiency could reach 1.28 TOPS W^{-1} (in ANN mode) and 649 GSOPS W^{-1} (in SNN mode with T_w values of less than 1). DRAM, dynamic random access memory. References cited are refs [3,6,7,35,55,56].

Extended Data Table 3 | Model topologies and input/output descriptions for networks applied in the bicycle demonstration

Network	Topology	Input	Output
CNN	6C3-MP2-12C3-MP2-12C3-MP2-FC192-10	70×70 grayscale images	Confidences and coordinates of objects (human, obstacle)
CANN	20×24 recurrent	Images clipped by the initial coordinates of the human from CNN	Coordinates of tracked human
SNN	510-256-7	510 converted voice features from microphone	Classification results of 7 commands (one for noise)
MLP	30-256-32-1	Target and adjusted angle (φ_1, φ_2), angular velocity (ω), velocity (v), voltage coefficient (k_V)	Target rotation angle (φ_3) sent to the motor controller
NSM	16 neurons 5 states 7 transitions	Signals from CNN, SNN and hybrid-mode FCores	Enable signals to CNN and CANN, and action signals to hybrid-mode FCores