nature portfolio

Peer Review File

Adaptive spatiotemporal neural networks through complementary hybridization

REVIEWER COMMENTS

Reviewer #1 (Remarks to the Author):

Short Summary: The paper proposes a three-step approach to creating a hybrid network comprising of artificial and spiking neurons named HSTNNs. The network creation process is based on a unified surrogate gradient learning framework and a Hessian based neuron selection strategy. To show the effectiveness of this three-step approach, the authors show improved performance on N-MNIST, S-MNIST, PTB and DVS-Gesture datasets.

Strengths:

The idea of using hybrid models is interesting. But the work needs better motivation for using such complex search methodology for hybridization since the results does not show sufficient improvements.

Weaknesses:

Paper is not well written. For instance, acronyms like HSTNNs, HTSNNs is used interchangeably. Additionally, the most important parameter i.e. Hybrid-ratio has been referenced as "the ratio between two types of neurons" without clarity on whether this ratio is between spiking neurons and artificial neurons or vice versa.

The paper's methodology for selecting neurons is far from clear. For instance, during the Hessian-based neuron selection process, are we considering the local minima (positive definite hessian) or the local maxima (negative definite hessian)? This fundamental oversight raises questions about the rigor of the research.

The rationale for hybridization at the neuron level lacks sufficient explanation. The authors fail to justify why this approach is superior to hybridization at the layer level [1]. Furthermore, the paper lacks a clear motivation for employing hybrid models. It does not adequately explain whether hybrid models address the limitations of traditional SNNs or RNNs.

Claims have been made about the robustness of SNNs against variable hyperparameters and [2] has been reference. However, [2] only talks about robustness to changes in temporal resolution alone. How does this claim hold with respect to training hyperparameters?

The paper doesn't have enough experiments to show that the proposed method can be used effectively. Results are only shown on small datasets like N-MNIST, S-MNIST, PTB and DVS-Gesture. Hence, the scalability of the proposed method is not conclusive. The authors need to show experiments on more complex tasks like optical flow estimation.

Neural network architecture for SNNs, RNNs and hybrid models is not mentioned in the paper.

The experiments are not strong enough to prove that hybridization at neuron level performs better. From the results it can be seen that the accuracy difference between Directly-hybrid and HSTNNs is negligible.

Results reported are on MLP layers only, do the authors have results after incorporating convolution operations and deeper networks?

References

Negi S, Sharma D, Kosta AK, Roy K. Best of Both Worlds: Hybrid SNN-ANN Architecture for Event-based Optical Flow Estimation. arXiv preprint arXiv:2306.02960. 2023 Jun 5.

W. He, Y. Wu, L. Deng, G. Li, H. Wang, Y. Tian, W. Ding, W. Wang, and Y. Xie, "Comparing SNNs and RNNs on neuromorphic vision datasets: Similarities and differences," Neural Networks, vol. 132, pp. 108–120, 2020.

Reviewer #2 (Remarks to the Author):

In this work, Wu and colleagues explore a hybrid approach of integrating RNNs and SNNs for processing spatio-temporal data. Their study aims to demonstrate a comprehensive solution that combines the advantages of RNNs and SNNs, achieving enhanced adaptability in terms of accuracy, robustness, and computational costs, and boosting practical neuromorphic applications. Their results are demonstrated with various spatio-temporal datasets, robot navigation tasks, and hardware simulation tests.

Overall, this is a valuable study with compelling results. There is an increasing inclination towards hybrid analog and digital computing in recent neuromorphic chips, such as Loihi 2, SpiNNaker 2, and Tianjic (shown in this manuscript). However, to the best of my knowledge, there appears to be a gap in solutions for effectively combining the strengths of diverse recurrent networks. The methodology employing the classical OBS method for determining neuron importance is sound and the presented findings are important for the neuromorphic computing community in advancing towards more integrated solutions that leverage various recurrent neural networks. However, there is a need for further exploration in assessing the scalability of the proposed model in more standard networks, providing more implementation details, and improving the overall presentation (see below). With proper improvements, this manuscript can be suitable for publication.

Major issues
1. Comparison with related work: I would like to see a clearer comparison to the following references on hybrid modelling methods:
a) Zhao R, Yang Z, Zheng H, et al. A framework for the general design and computation of hybrid neural networks[J]. Nature communications, 2022, 13(1): 3427.
b) Negi S, Sharma D, Kosta A K, et al. Best of Both Worlds: Hybrid SNN-ANN Architecture for Event-based Optical Flow Estimation[J]. arXiv preprint arXiv:2306.02960, 2023.
2. Network structure and performance: Although Figure 2 shows better accuracy for hybrid methods, the overall performance does not match the state-of-the-art in the same tasks. The authors should consider using a more standard and larger structure commonly used in previous studies for comparison.
3. Network training: the restorage stages only preserve half of hybrid neurons from the two pools of neurons. What's the impact of the sizes of the two pools?
4. Hardware implementations: I appreciated the inclusion of the hardware validation for the proposed method, which is useful for indicating the practical advantages of the hybrid approach. However, some details and necessary explanations are still missing. In Figure 3c, it is unclear why the hybrid model can achieve a lower latency than the pure SNN. Given that there are multiple mapping and scheduling strategies (like fixed-core and variable-core mapping strategies), The rationale behind the chosen mapping strategy needs more clarifications.
5. Robot application: The demonstrated robot navigation is of great interest. However, some details regarding the model's implementation are missing, including the exact size of the camera outputs and the sequence length of each sequence used in training and testing. Also, were the pre-trained CNNs and SNNs models fine-tuned for the tasks?

Minor issues
1. I am curious if the hybrid model can support the combination of more complex biological models like the adaptive LIF or Hodgkin-Huxley model. It would be very useful to show the potential of proposed methods for gaining more benefits from biological models.
2. The mathematical formulation for the storage is somewhat complex. Please clarify the dimensions of each tensor, like m^{n,r} and m^{n,s}
3 In Equations 23-25, the approximate loss in the cost assessment is not clear. The authors should elaborate the terms ignored in these calculations.

Point-by-Point Response to Reviewers' Comments

We would like to thank the reviewers for spending valuable time and raising insightful comments that truly helped improve our manuscript. The point-by-point responses are provided as follows to address the questions raised by the reviewers. All revisions in the manuscript are marked in blue for your convenience.

Reviewer #1

General Comment:

The paper proposes a three-step approach to creating a hybrid network comprising of artificial and spiking neurons named HSTNNs. The network creation process is based on a unified surrogate gradient learning framework and a Hessian based neuron selection strategy. To show the effectiveness of this three-step approach, the authors show improved performance on N-MNIST, S-MNIST, PTB and DVS-Gesture datasets. The idea of using hybrid models is interesting. But the work needs better motivation for using such complex search methodology for hybridization since the results does not show sufficient improvements.

General Response:

We appreciate the reviewer's insightful comments for helping us enhance the overall presentation and sharpening our key contributions.

First, regarding your concerns about the significance of the improvement, we guess that you are referring to the comparison between HSTNNs and the directly-hybrid models according to Comment 5. We would like to point out that the directly-hybrid model is, in fact, also a simple form of hybridization methods proposed by our work rather than an existing method. Furthermore, to better address your concern, we will show in the following responses that *consistent and more significant improvements can be achieved by HSTNNs when handling more complicated tasks, particularly considering constraints of reduced network sizes*, critical for deployment on resource-limited devices (see our response to Comment 5).

In response to your other comments, we have accordingly included five newly-added experiments using deeper convolutional neural networks and more complicated benchmarks (including CIFAR10-DVS, WikiText-2 and optical flow estimation, see Tables R1-R4 and Figures R1-R3) to demonstrate the effectiveness and scalability of HSTNNs.

In addition, we would like to summarize several main advantages of the proposed hybrid methodology as follows for providing a better overview of our motivation.

(1) Integrating diverse coding schemes with robust performance in multiple task scenarios

HSTNNs enable the integration of neural coding schemes in non-spiking and spiking neural networks, such as rate-based coding and temporal coding. The integration yields a more comprehensive representation of spatio-temporal data, facilitating overcoming the limitations of single coding schemes adopted by existing single-paradigm neural networks.

For instance, in comparison to typical coding schemes of RNNs, hybrid coding avoids network sensitivity to the input at the last time step by integrating the spike rate coding, improving robustness against noisy inputs. This feature is particularly advantageous in diverse realistic tasks with highly noisy neuromorphic vision inputs, as evidenced by the results in our control experiments with noisy inputs (see Figure 3(c)) as well as the consistent accuracy improvement over RNNs on multiple neuromorphic datasets.

On the other hand, hybrid coding improves the representation precision of single-paradigm SNNs, providing a simple solution for extending SNNs' capability to handle complicated tasks which require high-precision representation, as demonstrated by the results in the PTB task and the newly-added WikiText-2 task (Table R3).

(2) Enhancing the learning nonlinearity and the sequence information processing capability, demonstrating superior accuracy than single-paradigm networks and the state-of-the-art

Various neuronal models, including non-spiking vanilla RNNs and LSTMs, as well as spike-based LIF, and spike-response models, exhibit different temporal dynamics features and various typical nonlinear activation functions, which leads to varied task performances in traditional sequence tasks and neuromorphic tasks [1, 2]. HSTNNs provide a solution to combine these features and enhance nonlinearity in network computation, which has been recognized as key to enhancing the learning capability [3].

In this response letter, our newly-added experiments further show that HSTNNs are applicable for deep recurrent convolution neural networks (see Tables R1&R2). Importantly, integrating more diverse neuronal computation features enhances the HSTNN performance (see Figure R5 in response to Reviewer 2's Comment 5), improving task performance compared to single-paradigm networks and achieving comparable accuracy to state-of-the-art models (see Table R4 in response to Reviewer 2's Comment 2). These results highlight the scalability and effectiveness of our hybrid approach.

(3) Providing better comprehensive performance in terms of accuracy, robustness, and computational cost, validated by hardware implementation

It is important to note that one of our main motivations for developing HSTNNs is to facilitate realistic applications. The emerging trend in neuromorphic hardware with hybrid architectures, supported by

platforms like Loihi2 from Intel and Tianjic from our team, highlights the need for integrated solutions that combine the advantages of different neuron models. Our comprehensive validations (see Figures. 2-4) indicate that HSTNNs maintain the advantages of single-paradigm neural networks in terms of accuracy, robustness, computational cost, and representation precision, and even demonstrate better performance under optimal configurations. In this way, HSTNNs enable flexible controllability of the comprehensive performance by tuning the SNN ratio, which provides sufficient flexibility to adapt to different performance requirements in variable application scenarios. To better show hardware deployability, we have made extensive efforts in hardware implementation and evaluation (see Figure 5). These results offer an integrated solution that utilizes complementary features of diverse neural network models for processing real-world applications.

References

[1] Zhao R, Yang Z, Zheng H, et al. A framework for the general design and computation of hybrid neural networks[J]. Nature communications, 2022, 13(1): 3427.

[2] Negi S, Sharma D, Kosta A K, et al. Best of Both Worlds: Hybrid SNN-ANN Architecture for Event-based Optical Flow Estimation[J]. arXiv preprint arXiv:2306.02960, 2023.

[3] Chen H, Wang Y, Guo J, et al. VanillaNet: the Power of Minimalism in Deep Learning[J]. arXiv preprint arXiv:2305.12972, 2023.

Comment 1:

Paper is not well written. For instance, acronyms like HSTNNs, HTSNNs is used interchangeably. Additionally, the most important parameter i.e., Hybrid-ratio has been referenced as "the ratio between two types of neurons" without clarity on whether this ratio is between spiking neurons and artificial neurons or vice versa.

Response and revision:

We appreciate the reviewer's kind reminder of the inconsistent acronyms and the definition of the hybrid ratio. First, we have carefully revised our manuscript to ensure the consistent use of the acronym "HSTNNs". Second, to avoid ambiguity, we have replaced the hybrid ratio with the SNN ratio and provided its definition in Section "*Three-stage hybrid learning"* of the *Results* part: the ratio of the number of spiking neurons to the total number of hidden neurons after the three-stage learning. As last, we have improved our presentation throughout the revised manuscript to enhance accuracy and clarity.

Comment 2:

The paper's methodology for selecting neurons is far from clear. For instance, during the Hessian-based neuron selection process, are we considering the local minima (positive definite hessian) or the local maxima (negative definite hessian)? This fundamental oversight raises questions about the rigor

of the research.

Response:

We would like to point out that these information had indeed been provided in the Section "*Details of the Three-Stage Learning for HSTNN*" in the *Methods* part of the previous manuscript, particularly in the original paragraph preceding Equation (10) (i.e., Equation (8) in the revised manuscript), where we specified the focus on local minima and the positive semidefinite property of the Hessian matrix *H*. Considering your feedback, we have added more descriptions in the main context of the revised manuscript (see the revised Section *"Three-stage hybrid learning"* in the *Results* part) for convenient reading.

Revision:

Included more descriptions in the revised Section "*Three-stage hybrid learning"* of the *Results* part, i.e., "…We use the second-order gradient information of synaptic connections around the local minimum of the loss function as the basic measurement for importance, collect the importance scores of both feedforward and recurrent connections for each neuron, and finally evaluate the neuron importance according to the accumulated gradients spanning the temporal domain."

Comment 3:

The rationale for hybridization at the neuron level lacks sufficient explanation. The authors fail to justify why this approach is superior to hybridization at the layer level [1]. Furthermore, the paper lacks a clear motivation for employing hybrid models. It does not adequately explain whether hybrid models address the limitations of traditional SNNs or RNNs.

References

[1] Negi S, Sharma D, Kosta AK, Roy K. Best of Both Worlds: Hybrid SNN-ANN Architecture for Event-based Optical Flow Estimation. arXiv preprint arXiv:2306.02960. 2023 Jun 5.

Response:

For the two questions, we provide answers respectively as follows.

(1) We appreciate this insightful comment for distinguishing our key distinctions from previous work. Considering your concern, we outline the following three key features of our neuron-wise hybrid approach compared to the previous layer-wise approach exemplified in [1]:

(a) Avoiding the communication conversion bottleneck among different neuronal modules inherited by layer-wise methods and supporting the intra-layer hybrid coding

The previous work [1] was designed to combine non-recurrent ANNs and SNNs for addressing

specific optical flow tasks. To enable the information conversion between ANNs and SNNs, it processes the accumulated spike train of SNNs into non-recurrent ANNs, potentially leading to latency bottlenecks and increased memory costs proportional to the encoding windows of spike modules. Even if one considers extending the method to recurrent network models, such layer-wise hybridizations would still present challenges in information conversion bottlenecks and losing the event-driven computing advantages inherent to SNNs.

Contrastively, our method, with a different goal of developing a general framework designed for integrating neuron models with diverse temporal dynamics, adopts a decoupling neuron-wise hybridization strategy. It supports the real-time interaction of different neuron outputs in a finer time resolution and can maintain different neuronal dynamics, thereby *supporting the intra-layer hybrid coding and avoiding* the information loss inherited by [1].

(b) Providing broader hybridization search space including the layer-wise scheme

Please note that the Selection stage in HSTNN construction evaluates different neuron types across layers. This has considered the layer-wise architecture in [1] as a special case where all significant neurons of the same type are selected within the same layer. Owning to the larger search space, our hybrid method potentially benefits for better task performance.

(c) Providing greater flexibility and finer-grained configuration for real-world applications

The layer-wise hybridization provides a limited selection for the hybrid combinations since different layers are designed for different neuron types. Without such limitation, the neuron-wise method enables optimizing hybrid configurations more precisely with finer grain, thus better accommodating diverse performance and resource requirements in practical applications.

(2) As mentioned in the responses to the General Comment, the primary motivations for developing our hybrid model and addressing the potential limitations of single-paradigm neural networks include (1) overcoming the limitations of single-type coding schemes in realistic tasks by employing richer hybrid coding schemes; (2) enhancing the learning nonlinearity and the sequence information processing capability, demonstrating superior accuracy than single-paradigm networks; (3) achieving better comprehensive performance compared to single-paradigm models, which provides sufficient flexibility to adapt to different performance requirements in variable application scenarios. For more detailed explanations, please refer to the general response to the General Comment.

Revision:

Added discussions of comparing our hybridization method with the layer-wise method [1] (i.e., ref. [24] in the revised manuscript) in the revised *Discussion* section to clarify our contributions clearer and more accurately.

In the revised *Discussion* section, we included a comparison with references [24]: "…Several advantages of layer-wise hybridization have been demonstrated in references [23, 24]. These layer-wise hybridization approaches focus on integrating non-recurrent ANN and SNN modules using layer-wise strategies, providing efficient solutions for practical applications such as optical flow estimation and high-speed tracking tasks. In contrast, the proposed neuron-wise hybridization approach in this work offers a finer-grained method of hybridization, enabling real-time interaction of the coding and computational features of different types of neurons. Moreover, the neuron selection strategy employed in the Selection stage represents a more general solution that encompasses layer-wise hybridization as a specific case."

Added more detailed descriptions for [24] in the revised Section "*Creating HSTNNs"* of the *Results* part: "…Several studies [23, 24] have explored layer-wise hybrid models integrating non-recurrent ANN and SNN modules at the layer level. However, elaborating on specifying fixed heterogeneous networks in advance for each specific task is required, and a hybrid approach supporting effectively integrating diverse temporal dynamics and handling spatio-temporal data flows is still lacking."

Comment 4:

Claims have been made about the robustness of SNNs against variable hyperparameters and [2] has been reference. However, [2] only talks about robustness to changes in temporal resolution alone. How does this claim hold with respect to training hyperparameters?

References

[2] W. He, Y. Wu, L. Deng, G. Li, H. Wang, Y. Tian, W. Ding, W. Wang, and Y. Xie, "Comparing SNNs and RNNs on neuromorphic vision datasets: Similarities and differences," Neural Networks, vol. 132, pp. 108–120, 2020.

Response:

We are sorry for the inaccurate descriptions for ref. [2] (i.e., ref. [12] in the current manuscript). The reviewer is correct that the cited reference is mainly addressing the robustness when varying the temporal solution. We have accordingly revised our claims and descriptions of this work in the revised Introduction section to reflect the existing studies on SNNs' advantages more accurately.

Revision:

Modified the descriptions in the revised *Introduction* section, i.e., "Owing to the natural filtering effect of the membrane potential leakage along with the spike firing and reset mechanism of spiking neurons, SNNs have demonstrated strong robustness against variations in temporal resolution [12] and against adversarial attack [14]."

Comment 5:

The experiments are not strong enough to prove that hybridization at neuron level performs better. From the results it can be seen that the accuracy difference between Directly-hybrid and HSTNNs is negligible.

Response:

First, as mentioned in the response to the General Comment, we would like to point out that the directly-hybrid approach is also a simple form of hybridization methods proposed by our work rather than an existing method.

Then, for the comparison between HSTNNs and the directly-hybrid models, we acknowledge that the significance of performance improvements by HSTNNs depends on the task type and varies with the average accuracy level that can be achieved by single types of networks. However, as noted in Figure 2, there is a consistent trend of HSTNNs outperforming directly-hybrid models.

To better address your concern on the need for stronger experiment results, we have further analyzed the impact of hybridization, particularly under typical resource constraints on edge devices. In the following we will show that the improvement by HSTNNs can be more significant when creating a more compact network structure.

To quantify the effect of HSTNNs on creating reduced-size compact networks, we introduce a metric termed the "remaining neuron ratio". This ratio represents the number of hidden neurons in the Restoration stage over those in the Adaptation stage. We validate the effects of this remaining neuron ratio on the S-MNIST dataset with the network structure of [28-800-800-10] and on the N-MNIST dataset with [2312-800-800-10]. The vanilla RNN and LIF-based SNN pools are used in the Adaptation stage.

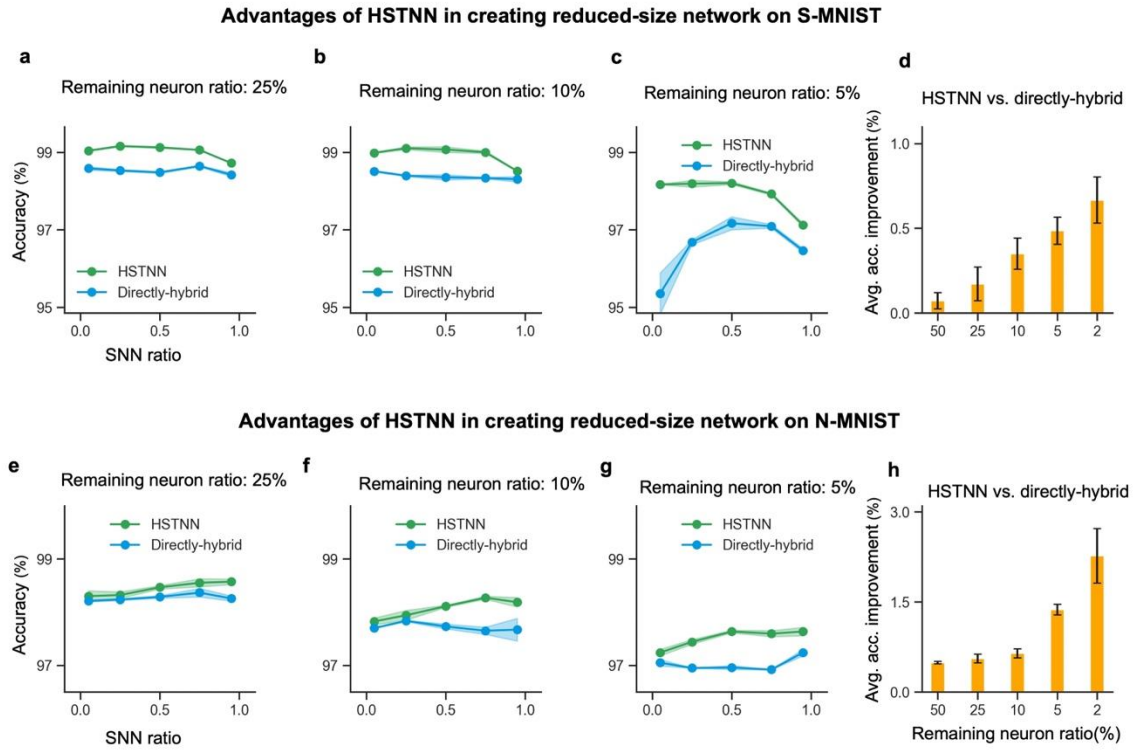


Figure R1. Advantages of HSTNNs in creating reduced-size compact networks on S-MNIST (upper) and N-MNIST (lower). The remaining neuron ratio refers to the ratio of the number of remaining hidden neurons in the Restoration stage over those in the Adaptation stage. The network structure of [28-800-800-10] are used for S-MNIST and [2312-800-800-10] for N-MNIST. Error bars represent the standard variance over five trials.

Results depicted in Figure R1(a)-(c) and Figure R1(e)-(g) evidence a consistent improvement by HSTNNs. Importantly, the average improvement of HSTNNs across five SNN ratios (i.e., {0.05,0.25,0.5,0.75,0.95}) over those trained by the directly-hybrid method, as shown in Figure R1(d)&(h), becomes more pronounced as the remaining neuron ratio decreases, i.e., smaller network sizes. It suggests that our model offers a more stable solution for the requirements in real-world edge systems where resources are usually limited.

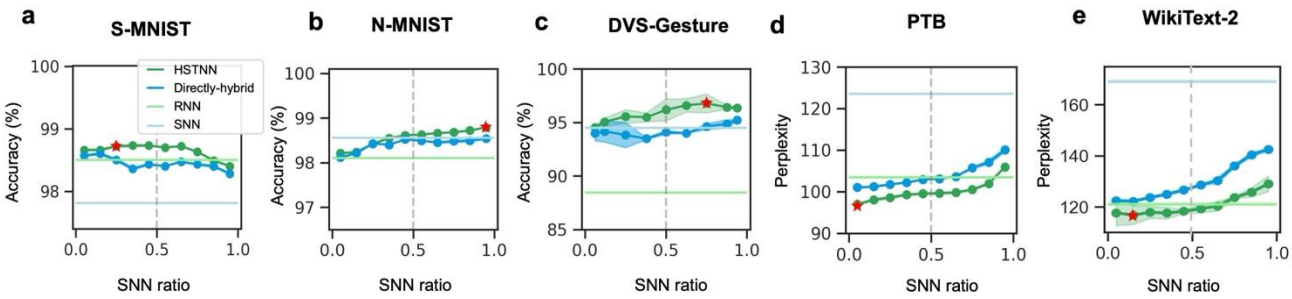


Figure R2. Comparison of HSTNNs to directly-hybrid models on (**a**) S-MNIST, (**b**) N-MNIST, (**c**) DVS-Gesture, (**d**) PTB and (**e**) WikiText-2 datasets, where larger and deeper convolutional structures are used. The network structure of [input-400-400-10] is used for S-MNIST, [input-800-800-10] for N-MNIST, [input-650-650-output] for both PTB and WikiText-2, and a nine-layer convolutional network structure of [input-128C3-AP2-256C3-AP2-384C3-AP2-256-11] for DVS-Gesture. Error bars represent the standard variance over five trials. The red star denotes

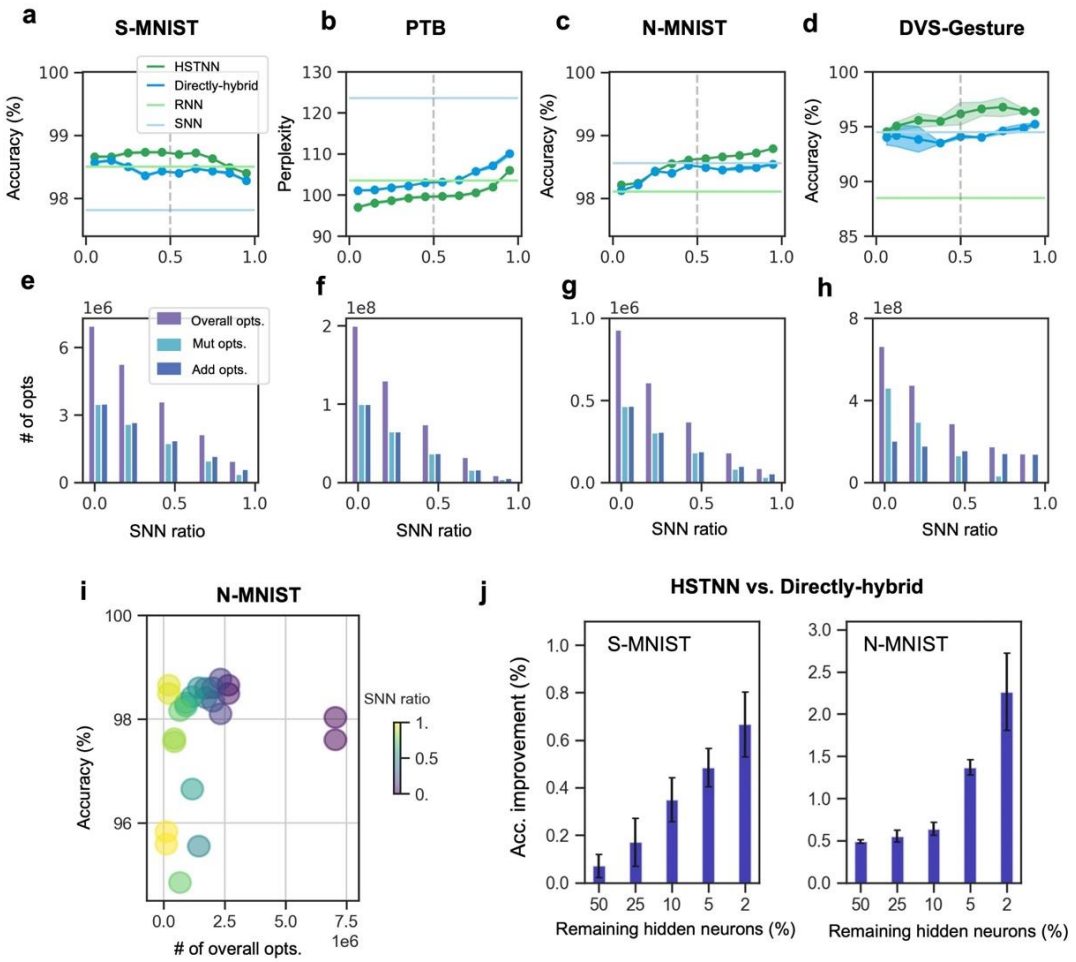
the best performance achieved by HSTNNs. For PTB and WikiText-2, the performance is measured by perplexity and lower perplexity indicates better performance.

Moreover, compared to directly-hybrid models, we observe that HSTNNs exhibit more significant improvement when using a deeper and more complex network structure (see Figure R2 and the responses to Reviewer 2's Comment 2) and can benefit from adopting a larger initial pool size in the Adaptation stage (see the responses to Reviewer 2's Comment 3). This robust and consistent improvement is also validated on more challenging text analysis tasks, such as WikiText-2 (see Figure R2(e)). These findings support the effectiveness and scalability of our hybridization methodology.

Revision:

Included Figure 2(j) and added analyses of the effectiveness of HSTNNs compared to directly-hybrid models. We have attached the revised Figure 2 and the relevant descriptions for Figure 2(j) below for your convenience:

"Furthermore, as observed in Figure 2(j), the improvement of HSTNNs over directly-hybrid models becomes more pronounced when constructing smaller-size hybrid neural networks, a practical constraint commonly considered in real-world edge systems with limited resources. These comparisons reflect the effectiveness of the elaborate three-stage learning methodology: expanding the representation dimension initially and then meticulously selecting important neurons from the hybrid redundant populations, rather than directly training a compact hybrid network from scratch."



Revised Figure 2. **Comprehensive analysis of the adaptive trade-off between accuracy and the computational cost.** Impact of the SNN ratio on accuracy (upper) for (**a**) S-MNIST, (**b**) PTB, (**c**) N-MNIST, and (**d**) DVS-Gesture datasets. Impact of the SNN ratio on the number of operations (lower) for (**e**) S-MNIST, (**f**) PTB, (**g**) N-MNIST, and (**h**) DVS-Gesture datasets. A two-hidden-layer network structure is adopted for PTB, S-MNIST, and N-MNIST, and a convolutional structure for DVS-Gesture. Note that accuracy on the PTB dataset is indicated by perplexity (ppl), where lower is better. (**i**) Comprehensive analysis of the trade-off between accuracy and the computational cost for the N-MNIST dataset, where the computational cost is measured by the total number of multiplication and addition operations. (**j**) Analysis of average accuracy improvement by HSTNNs compared to directly-hybrid models, measured by the ratio of remaining hidden neurons in the Restoration stage to those in the Adaptation stage. Error bars represent the standard deviation over five repeated trials.

Comment 6:

Results reported are on MLP layers only, do the authors have results after incorporating convolution operations and deeper networks?

Response:

Yes, our methodology can be extended to convolutional operations and deeper networks. In the

previous manuscript, we focused on MLP layers as our primary goal to develop a hybrid modelling approach for handling complex spatiotemporal data. To address your concern, we have expanded our hybrid methodology to Recurrent Convolutional Neural Networks (RCNN) and Spiking Convolutional Neural Networks (SCNN). Also, we have applied this approach to a deeper network per your suggestion and validated it on neuromorphic CIFAR10-DVS and DVS-Gesture datasets. As below, we detail the main modifications for the convolutional structure, outline our key findings, and attach our revisions of the manuscript.

(1) Expansion to convolutional neural networks. The primary modification for RCNNs and SCNNs involves computing the neuron importance within the convolutional layers using the Hessian matrix of convolutional parameters. Specifically, the neuron computations in RCNNs can be formulated as

$$\begin{cases} a_t = W_{in}^r * x_t + g(W_{rec}^r * r_t) \\ r_t = f(a_t) \end{cases},$$

where * denotes the convolutional operation, $x_t = [r_t, s_t]$ denotes the inputs that combines the non-spiking and spiking signal from the last layer at the t-th time step, a_t denotes the neuronal states and r_t denotes the outputs. W_{in}^r and W_{rec}^r represent the feedforward and recurrent convolutional weights, respectively, with g and f are activation functions.

We model the spiking neuron in SCNNs receiving convolution-based inputs as

$$\begin{cases} u_t = e^{-\frac{dt}{\tau_u}} u_{t-1}(1 - s_{t-1}) + W_{in}^s * x_t, \\ s_t = H(u_t) \end{cases}$$

where u_t denotes the membrane potential, H denotes the Heavisde function, W_{in}^r represents the feedforward convolutional weights, and $x_t = [r_t, s_t]$ shares the same definition as that in RCNNs.

Given the above expression for RCNNs and SCNNs, we perform the Adaption stage for creating large neuron pools incorporating recurrent convolutional neurons and spiking convolutional neurons. In the Selection stage, we calculate the importance of each neuron in RCNN and SCNN populations based on the Hessian traces of their afferent feedforward weights. Then we adopt a structural grouping strategy that selects the most important output feature map channels based on cumulative importance scores across all neurons within the same feature map. These selected feature maps are then maintained to create a reduced network structure for retraining in the Restoration stage.

(2) Details of the establishment of deep networks on CIFAR10-DVS and DVS-Gesture. For DVS-Gesture, we built a network with the same structure of [input-128C3-AP2-256C3-AP2-384C3-AP2-256-output] for both RCNN and SCNN parts, where the outputs from RCNN and SCNN pools in each layer were concatenated and fed into the next layer. We followed by selecting half of the important output channels in the Selection stage and retraining the shrunk network structure in

the Restoration stage. We used $tanh$ for $g(x)$ and $ReLU$ for $f(x)$ for RCNNs. The decay and threshold for SCNN were set to 0.3 and 0.5, respectively. For DVS-Gesture, we used Adam as the optimizer and MSE as the loss function, with 320 training epochs for both the Adaption stage and the Restoration stage. The learning rates were set to 0.0005 and 0.0003 for the Adaption stage and the Restoration stage, respectively. For CIFAR10-DVS, we adhered to the same modelling strategy and constructed our hybrid model on a deeper structure of [input-128C3-128C3-AP2-128C3-128C3-AP2-256C3-256C3-AP2-512C3-512C3-512C3-512C3-10]. Stochastic Gradient Descent (SGD) was selected as the optimizer, incorporating a momentum of 0.9. The loss function, a weighted average of Cross-Entropy (CE) and MSE functions, was adopted, setting the weight factors for the CE and MSE components at 0.95 and 0.05, respectively, as outlined in setup [1]. The training regimen for both Adaptation and Restoration stages was uniform, comprising 300 epochs each, with an initial learning rate of 0.05.

(3) Main results of applying HSTNNs to convolutional neural networks. Our methodology showed that for the convolutional structures, HSTNNs still achieve better accuracy compared to single-paradigm neural networks. Additionally, we found that our models can achieve state-of-the-art accuracy compared to other advanced works on the two neuromorphic datasets. We have added the extra results (see Tables R1&R2) in the revised manuscript.

Table R1. Results of HSTNNs and single-paradigm NNs (i.e., RCNNs and SCNNs) on CIFAR10-DVS. For each accuracy, we showed the mean average ± standard deviation over five repeated trials.

SNN ratio	0.0	0.125	0.25	0.50	0.75	0.875	1.0
Single-paradigm NN	72.25±0.22	N/A	N/A	N/A	N/A	N/A	76.10±0.79
HSTNN		74.5±0.70	76.27±0.41	77.47±0.81	77.47±0.47	**77.60±0.36**	

Table R2. Results of HSTNNs and single-paradigm NNs (i.e., RCNNs and SCNNs) on DVS-Gesture. For each accuracy, we showed the mean average ± standard deviation over five repeated trials.

SNN ratio	0.0	0.125	0.25	0.50	**0.75**	0.875	1.0
Single-paradigm NN	87.93±0.5	N/A	N/A	N/A	**N/A**	N/A	94.79±0.18
HSTNN		95.05±0.38	95.57±0.63	96.18±1.02	**96.79±0.87**	96.44±0.20	

References

[1]. Fang W, Yu Z, Chen Y, et al. Incorporating learnable membrane time constant to enhance learning of spiking neural networks[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2021: 2661-2671.

Revisions:

Included extra results on deeper convolutional neural networks and more complex datasets in the revised Figures 2&3, Table 1 and the relevant descriptions in the revised main text and *Methods*, to demonstrate the effectiveness and scalability of our methodology.

Comment 7:

Neural network architecture for SNNs, RNNs and hybrid models is not mentioned in the paper.

Response:

We would like to draw attention to Supplementary Table S1, where we had provided in the previous manuscript the structures of SNNs, RNNs, and hybrid models for the tasks. Given your concern, we have expanded the descriptions in the *Methods* part to detail the architectures more comprehensively. Here we copied the main revisions below for your reference.

Revision:

Expanded the descriptions in the revised Section "*Details of parameter configurations and model comprehensive evaluation*" of the *Methods* part, i.e., "We used consistent network structures for SNNs, RNNs, directly-hybrid models, and HSTNNs in our comparisons. On N-MNIST, S-MNIST, and PTB datasets, the network structures of [input-800-800-10], [input-400-400-10], and [input-650-650-10,000] were employed, respectively. On above three datasets, the HSTNNs were built based on vanilla RNN and LIF models. …On DVS-Gesture, the network structure of [input-128C3-AP2-256C3-AP2-384C3-AP2-256-11] was adopted, using recurrent convolutional neural network (RCNN) and LIF-based spiking convolutional neural network models for HSTNN construction."

Expanded the descriptions in the revised Section "*Experimental parameter setting*" of *Supplementary Information*, i.e., "For the deep-learning-oriented datasets, including PTB and S-MNIST, we establish HSTNNs with two hybrid hidden layers based on vanilla RNNs and LIF SNNs and use Cross Entropy (CE) as the loss function. For PTB, the word embedding size is 650, and the output dimension is 10000. During the training process, we halve the learning rate every 25 epochs. The SGD optimizer is applied for PTB with the momentum equal to 0.9. For S-MNIST, images are fed into networks in a column-wise manner. Note that the output spikes of the last hidden layer are accumulated across time steps.

Neuromorphic datasets comprise event-driven spike trains. To accelerate the training, we follow the setup in prior work (reference [1-3]) and preprocess the spike inputs by compressing the events in a time interval (i.e., temporal resolution, dt) and reducing the input size (i.e., spatial resolution, ds). In our experiments, $dt = 3$ms and $ds = 1$ are adopted on N-MNIST, $dt = 20$ms and $ds = 4$ on DVS-Gesture.

A convolutional neural network with the structure of [input-128C3-AP2-256C3-AP2-384C3-AP2-256-output] is built for the two datasets. The 128C3 denotes the convolutional layer with the kernel size of 3 and the number of output channels of 128, AP2 denotes the average pooling with the pooling size of 2, and FC denotes the fully connected layer. Besides, an FC-based network architecture with two hybrid hidden layers is also deployed on these two datasets. The loss functions are all set as Mean Square Error (MSE). For CIFAR10-DVS, we follow the high-performance architecture from reference [4]. It integrates the event data into 10 frames per sample and downscales each event image to 48×48×2 before being fed into the network. In addition, a 13-layer convolutional neural network with the structure of [input-128C3-128C3-AP2-128C3-128C3-AP2-256C3-256C3-AP2-512C3-512C3-512C3-512C3-10] is applied for CIFAR10-DVS. The SGD optimizer, with a momentum of 0.9, and a loss function comprising a weighted average of CE and MSE functions, with factors for the CE and MSE components set at 0.95 and 0.05, respectively, are employed. For the LIF-based SCNN models, we adhere to the modelling approach of reference [4], where the membrane decay time constant τ is set to 1.1, and a triangular surrogate function is utilized."

References

[1] Bellec G, Salaj D, Subramoney A, et al. Long short-term memory and learning-to-learn in networks of spiking neurons[J]. Advances in neural information processing systems, 2018, 31.

[2] Zhu A Z, Yuan L, Chaney K, et al. Unsupervised event-based learning of optical flow, depth, and egomotion[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019: 989-997.

[3] Wu, Y., Deng, L., Li, G., Zhu, J., Xie, Y., & Shi, L. (2019, July). Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 33, No. 01, pp. 1311-1318).

[4] Fang W, Yu Z, Chen Y, et al. Incorporating learnable membrane time constant to enhance learning of spiking neural networks[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2021: 2661-2671.

Comment 8:

The paper doesn't have enough experiments to show that the proposed method can be used effectively. Results are only shown on small datasets like N-MNIST, S-MNIST, PTB and DVS-Gesture. Hence, the scalability of the proposed method is not conclusive. The authors need to show experiments on more complex tasks like optical flow estimation.

Response:

We appreciate this insightful feedback regarding the need for more extensive experiments to demonstrate the effectiveness and scalability of our hybrid methodology. In previous responses, we have extended our approach to convolutional and deeper networks. To address the concerns in this comment, we have expanded our experiments to larger-scale datasets and more complex tasks, including a neuromorphic dataset, CIFAR10-DVS, a text dataset, WikiText-2, and the standard optical flow estimation task on the Multi Vehicle Stereo Event Camera (MVSEC) dataset. The results from these experiments, presented in Tables R1 and R3, as well as in Figure R3, can demonstrate the effectiveness and scalability of our approach. Below are more details on the experimental setups and results.

(1) Results of applying our hybrid approach on larger datasets (CIFAR10-DVS and WikiText-2)

CIFAR10-DVS: We reported the performance on CIFAR10-DVS in Table R1. Similarly, we also validated the effectiveness of our method on this dataset with a deeper convolutional network structure. Please refers to the response to Comment 6.

WikiText-2: We applied our hybrid method to the WikiText-2 dataset. Compared to the PTB dataset, WikiText-2 is more than twice as large. The token counts for the training, validation, and testing sets in WikiText-2 are [(2,088,628), (217,646), (245,569)], respectively. The vocabulary size for WikiText-2 is 33,278. We constructed a network structure of [input-650-650-(33,728)] for this dataset. We utilized SGD as the optimizer, with each of the Adaptation and Restoration stages spanning 200 training epochs. The learning rates were set to 0.5 and 0.1 for the Adaptation and Restoration stages, respectively. The dropout technique was applied with a rate of 0.5. The truncated length of the BPTT was set to 35, and batch size was set to 25.

Table R3. Results of HSTNNs and single-paradigm NNs (i.e., RNNs and SNNs) on WikiText-2. Perplexity (ppl) was used here as a measurement. We showed the mean average ± standard deviation over five repeated trials.

SNN Ratio	0.0 (RNN)	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0 (SNN)
Single-Paradigm NN	121.81±1.25	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	169.07±0.53
HSTNN		117.04 ±0.72	**116.80 ±1.00**	117.28 ±0.67	117.52 ±0.80	118.56 ±1.07	118.97 ±0.64	120.35 ±0.80	123.08 ±0.21	128.43 ±0.28	

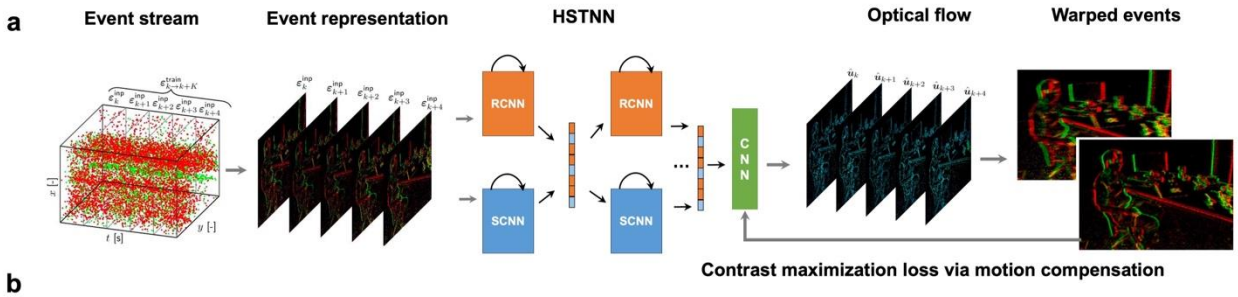
The results evidence that our proposed hybrid method is capable of handling larger datasets, showcasing superior performance compared to single-paradigm neural network models.

(2) **Results of applying our hybrid approaches to the optical flow estimation task**

We have applied our hybrid approach to the optical flow estimation, suggested by the reviewer, on the MVSEC dataset, following the experimental setups and parameter configurations in [1]. Note that previous work adapts the FireNet architecture [2] to single-paradigm RNN and SNN versions for optical flow estimation. As illustrated in Figure R3, the event stream is segmented into small partitions, each containing an equal number of events. These partitions are formed into event-based images with a size of 256×256×2 and sequentially injected into the network. The network predicts an optical flow map for every pixel and every polarity (+1, -1) in each partition, correlating every input event with a motion vector. Following sufficient event processing, a backward pass is conducted using the contrast maximization loss as in [3]. In the following we would like to sequentially illustrate the network architecture, main results and the details of the training process for your reference.

Network architecture for hybrid models. Our implementation retains the base network architecture adopted by recent studies [1, 4], consisting of three recurrent (or spiking) convolutional layers, two recurrent (or spiking) ResNet layers, and a final prediction layer, as shown in Figure R3(a). Each layer uses a single stride, has 32 output channels, and employs 3×3 kernel encoders. The spiking and recurrent convolutional layers are designed in accordance with Equations (6), (7), and (15) in [1]. The prediction layer integrates spike representations with non-spiking components to yield motion estimation.

Measurement and main results. We utilize the average endpoint error (AEE), the percentage of points with AEE greater than 3 pixels and 5% of the magnitude of the optical flow vector, denoted by %Outlier, as performance metrics, as per [1, 4]. The lower values of AEE and %Outlier indicate better performance. Quantitative results of our best performing networks in this task are shown in Figure R3(b) and the qualitative results of the warped input events by the motion compensation obtained from the outputs of HSTNNs are shown in Figure R3(c). As can be seen from Figure R3(b), the proposed models generally outperform the single-paradigm RNNs and the spiking LIF networks under the same network structure (i.e., FireNet), demonstrating the applicability of the proposed hybrid approach.



b

$dt = 1$	Indoor_flying1		Indoor_flying2		Indoor_flying3	
	AEE	$\%_{Outlier}$	AEE	$\%_{Outlier}$	AEE	$\%_{Outlier}$
RNN FireNet	0.96	2.60	1.77	17.55	1.48	13.6
Spiking LIF FireNet	0.98	2.48	1.77	16.4	1.50	12.81
Our work	0.845	2.48	1.18	14.95	1.15	14.95

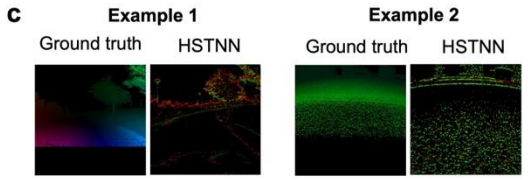


Figure R3. **Illustration of applying the proposed hybrid model for the optical flow estimation task.** (**a**) Optical flow pipeline for construction of hybrid neural networks, adapted from Figure 1 in reference [1]. (**b**) Quantitative results, including the average endpoint error (AEE), the percentage of points with AEE greater than 3 pixels and 5% of the magnitude of the optical flow vector (%Outlier), are presented for our hybrid model alongside results from the single-paradigm models in reference [1]. We use the same network structure (i.e., FireNet) and neuron models as described in Equations (6), (7) and (15) in reference [1], and construct the hybrid FireNet with an SNN ratio of 0.5 for comparison. (**c**) Qualitative evaluation of the warped input events by the motion compensation obtained from the outputs of the proposed hybrid model.

Details of the training process. Per the steup of [1], the hybrid model first compresses a fixed number of events, N_e (1,000 followed by [1]), containing per-pixel and per-polarity information, into a tensor-represented event image. This tensor is then fed into the RCNN and SCNN modules. The outputs of RCNN and SCNN modules in each layer are concatenated (see Figure R3(a)) and fed into the different network modules in the next layer. The network parameters are then optimized by minimizing the contrast maximization proxy loss following [1]. The basic idea of the contrast maximization proxy loss is to train the network to align events more accurately by minimizing the amount of motion blur after deblurring with the predicted motion to produce the deblurred image. To obtain the expression of this loss function, an image of the average timestamp at each pixel for each polarity $p \in \{+1, -1\}$ is first constructed. Specifically, knowing the per-pixel optical flow,

$$\hat{\boldsymbol{u}}(\boldsymbol{x}) = \big(u(\boldsymbol{x}), v(\boldsymbol{x})\big)^T,$$

the events are propagated to a reference time t_{ref} through

$$\boldsymbol{x}_i' = \boldsymbol{x}_i + \big(t_{ref} - t_i\big)\hat{\boldsymbol{u}}(\boldsymbol{x}_i).$$

Then, the Image of Warped Events (IWE) is generated [1, 5] by

$$T_p\big(\boldsymbol{x}; \hat{\boldsymbol{u}} | t_{ref}\big) = \frac{\sum_j k\big(x - x_j'\big) k\big(y - y'_j\big) t_j}{\sum_j k\big(x - x_j'\big) k\big(y - y'_j\big) + \varepsilon,}, \quad j = \{i | p_i = p\}, p \in P\{+1, -1\}, \ \varepsilon \approx 0,$$

where $k(x) \coloneqq \max(0,1-|x|)$. Given the images, the loss function is defined following

$$L_{contrastive}(t_{ref}) = \frac{\Sigma_x T_+(\boldsymbol{x}; \hat{\boldsymbol{u}}|t_{ref})^2 + T_-(\boldsymbol{x}; \hat{\boldsymbol{u}}|t_{ref})^2}{\Sigma_x[n(x')>0]+\varepsilon},$$

where $n(x')$ denotes a per-pixel event count of IWE. Finally, the overall function to train our hybrid model is optimized through a compound loss as in [1, 5], combining the above $L_{contrastive}$ and a local smoothness regularizer L_{smooth} which minimizes the difference in optical flow between neighboring pixels by

$$L_{smooth} = \sum_{\boldsymbol{x}} \sum_{\boldsymbol{x}' \in \boldsymbol{N}(\boldsymbol{x})} \rho(\boldsymbol{u}(\boldsymbol{x}) - \boldsymbol{u}(\boldsymbol{x}')) + \rho(\boldsymbol{v}(\boldsymbol{x}) - \boldsymbol{v}(\boldsymbol{x}')),$$

where $\rho(*)$ denotes the L2 norm and $\boldsymbol{N}(\boldsymbol{x})$ is the 4-connected neighborhood around $\boldsymbol{x}$. Finally, the expression can be described as

$$L_{overall} = L_{contrastive} + \lambda L_{smooth}$$

where λ (0.001 in our experiment) is a scalar balancing the effect of the two losses. Further details on the modelling process can be found in [1].

References

[1] Hagenaars J, Paredes-Vallés F, De Croon G. Self-supervised learning of event-based optical flow with spiking neural networks[J]. Advances in Neural Information Processing Systems, 2021, 34: 7167-7179.

[2] Cedric Scheerlinck, Henri Rebecq, Daniel Gehrig, Nick Barnes, Robert Mahony, and Davide Scaramuzza. Fast image reconstruction with an event camera. In IEEE Winter Conf. Appl. Comput. Vis., pages 156–163, 2020.

[3] Zhu A Z, Yuan L, Chaney K, et al. Unsupervised event-based learning of optical flow, depth, and egomotion[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019: 989-997.

[4] Alex Z. Zhu and Liangzhe Yuan. EV-FlowNet: Self-supervised optical flow estimation for event-based cameras. In Robot.: Science and Syst., 2018

[5] Guillermo Gallego, Mathias Gehrig, and Davide Scaramuzza. Focus is all you need: Loss functions for event-based vis. In IEEE Conf. Comput. Vis. Pattern Recog., pages 12280–12289, 2019.

Revision:

Incorporated the results of optical estimation task in the newly added Section '*Applying the hybrid model to optical flow estimation*' of *Supplementary Information*.

Reviewer #2

General Comment:

In this work, Wu and colleagues explore a hybrid approach of integrating RNNs and SNNs for processing spatio-temporal data. Their study aims to demonstrate a comprehensive solution that combines the advantages of RNNs and SNNs, achieving enhanced adaptability in terms of accuracy, robustness, and computational costs, and boosting practical neuromorphic applications. Their results are demonstrated with various spatio-temporal datasets, robot navigation tasks, and hardware simulation tests.

Overall, this is a valuable study with compelling results. There is an increasing inclination towards hybrid analog and digital computing in recent neuromorphic chips, such as Loihi 2, SpiNNaker 2, and Tianjic (shown in this manuscript). To the best of my knowledge, there appears to be a gap in solutions for effectively combining the strengths of diverse recurrent networks. The methodology employing the classical OBS method for determining neuron importance is sound and the presented findings are important for the neuromorphic computing community in advancing towards more integrated solutions that leverage various recurrent neural networks.

However, there is a need for further exploration in assessing the scalability of the proposed model in more standard networks, providing more implementation details, and improving the overall presentation (see below). With proper improvements, this manuscript can be suitable for publication.

General Response:

We greatly appreciate your such positive feedback and the right recognition of our potential impacts on neuromorphic computing. For your comments, we have provided point-by-point responses as follows.

Comment 1:

Comparison with related work: I would like to see a clearer comparison to the following references on hybrid modelling methods:

[1] Zhao R, Yang Z, Zheng H, et al. A framework for the general design and computation of hybrid neural networks[J]. Nature communications, 2022, 13(1): 3427.

[2] Negi S, Sharma D, Kosta A K, et al. Best of Both Worlds: Hybrid SNN-ANN Architecture for Event-based Optical Flow Estimation[J]. arXiv preprint arXiv:2306.02960, 2023.

Response:

Refs. [1, 2] focus on integrating the non-recurrent ANN and spiking network in a layer-wise structure, i.e., combining non-spiking and spiking modules in different layers. While they have shown better task

performance compared to single SNN or ANN models, their methods do not support information conversion at the intra-layer level (i.e., neuron level) and face challenges in effectively integrating various neural networks, such as recurrent neural networks, thus limiting the ability to process complex spatio-temporal data.

To overcome the limitations, in this work, we develop neuron-wise hybrid approaches, aiming to unleash the potential of hybrid spatio-temporal networks for processing spatio-temporal data. To provide a clearer comparison, we summarize our key features compared to refs. [1] and [2], as outlined below.

1. **Comparison with Zhao et al. (Ref. [1])**

   Zhao et al. provide a general framework for a hybrid computation framework combining non-recurrent ANNs and SNNs. Despite the general framework provided in their work, it lacks a clear and feasible approach for building hybrid spatio-temporal neural networks, making it difficult to handle complicated spatio-temporal data directly.

   Our hybrid models offer a general method supporting the combination of diverse types of non-spiking recurrent neuronal models and spiking neuronal models, and demonstrate high scalability of diverse network architectures and superior performance over single-paradigm networks, paving the way for applying such hybridization in performing real-world spatio-temporal computing tasks.

2. **Comparison with Negi S et al. (Ref. [2])**

   Negi S et al. develop a layer-wise hybrid model based on the non-recurrent ANN and SNNs for addressing the specific task for optical flow estimation. To achieve information conversion between ANNs and SNNs, it requires first accumulating the spike train of SNNs and then feeding it into ANNs, leading to latency bottlenecks and increased memory costs.

   Our work, with a different goal, aims to develop a general hybrid approach for processing diverse spatio-temporal data flows. Our demonstrations on diverse neuromorphic and traditional tasks show that the proposed method enables a more efficient intra-layer information conversion between different types of neural network modules and interacts with finer time resolution. This feature allows fully leveraging temporal dynamic features and richer spike coding schemes within a layer, thereby avoiding potential information loss and supporting real-time interaction. Furthermore, our optimization process can be formalized as a more general hybridization method, because the optimization space incorporates such layer-wise solution as a special case where all significant neurons of the same type are selected within the same layer. This larger search space potentially leads to better task performance.

Considering this comment, we have provided more discussions on the comparison with previous hybrid methods in the revised manuscript to make our contributions clearer and more accurate.

Revision:

Added discussions on comparing our model with the layer-wise methods [1, 2] (refs. [23, 24] in the revised manuscript) to make our contribution clearer and more accurate.

"In the revised *Discussion* section, included a comparison with references [23, 24]: "…Several advantages of layer-wise hybridization have been demonstrated in references [23, 24]. These layer-wise hybridization approaches focus on integrating non-recurrent ANN and SNN modules using layer-wise strategies, providing efficient solutions for practical applications such as optical flow estimation and high-speed tracking tasks. In contrast, the proposed neuron-wise hybridization approach in this work offers a finer-grained method of hybridization, enabling real-time interaction of the coding and computational features of different types of neurons within a layer. Moreover, the neuron selection strategy employed in the Selection stage represents a more general solution that encompasses layer-wise hybridization as a specific case."

In the revised *Introduction* section, added more accurate descriptions: "…Several studies [23, 24] have explored layer-wise hybrid models integrating non-recurrent ANNs and SNN modules at the layer level. However, elaborating on specifying fixed heterogeneous networks in advance for each task is required, and a neuron-wise hybrid approach designed for integrating diverse temporal dynamics and handling spatio-temporal data flows is still lacking."

Comment 2:

Network structure and performance: Although Figure 2 shows better accuracy for hybrid methods, the overall performance does not match the state-of-the-art in the same tasks. The authors should consider using a more standard and larger structure commonly used in previous studies for comparison.

Response:

According to your suggestions, we have extended our approach to larger and more standard structures and datasets in the updated Figure 2 and newly-added Table 1 in the revised manuscript. Specifically, for the updated Figure 2, we adopt a larger network structure of [input-400-400-10] for S-MNIST, a more standard and larger network structure of [input-650-650-output] for both PTB and WikiText-2, a nine-layer convolutional network structure of [input-128C3-AP2-256C3-AP2-384C3-AP2-256-11] for DVS-Gesture, and a larger network structure of [input-800-800-10] for N-MNIST, following previous work [1-3]. As illustrated in the following Figure R2, our hybrid approach consistently outperforms the direct-hybrid models. Furthermore, our model with an optimal SNN ratio can demonstrate higher performance than single-paradigm neural networks, indicating the effectiveness and scalability of our methodology.

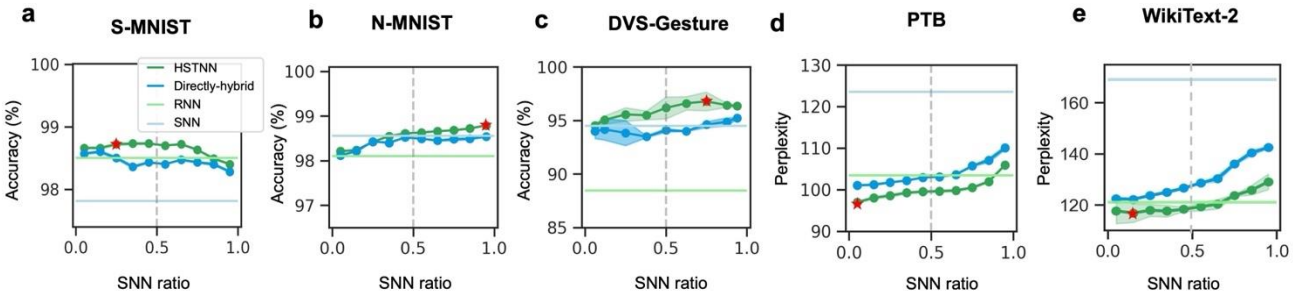


Figure R2. Comparison of HSTNNs to directly-hybrid models on (**a**) S-MNIST, (**b**) N-MNIST, (**c**) DVS-Gesture, (**d**) PTB and (**e**) WikiText-2 datasets, where larger and deeper convolutional structures are used. The network structure of [input-400-400-10] is used for S-MNIST, [input-800-800-10] for N-MNIST, [input-650-650-output] for both PTB and WikiText-2, and a nine-layer convolutional network structure of [input-128C3-AP2-256C3-AP2-384C3-AP2-256-11] for DVS-Gesture. Error bars represent the standard variance over five trials. The red star denotes the best performance achieved by HSTNNs. For PTB and WikiText-2. The performance is measured by perplexity and lower perplexity indicates better performance.

Table R4 Comparison with advanced models on neuromorphic datasets. Best accuracy, mean, and standard deviations were computed for each single-paradigm model and our hybrid model with the optimal SNN ratio over five trials. References in the table pertain to those in the revised manuscript.

Model	Dataset	Network Structure	Best Acc. (%)	Mean ± SD (%)
SNN model[8]	N-MNIST	5-layer SCNN	99.35	N/A
SNN model[30]		7-layer SCNN	99.40	N/A
RNN model (our work)		6-layer RCNN	99.55	99.51 ± 0.04
SNN model (our work)		6-layer SCNN	99.56	99.54 ± 0.02
Hybrid model (our work)		6-layer hybrid CNN	99.61	99.63 ± 0.02
SNN model[31]	DVS-Gesture	14-layer SCNN	97.22	N/A
SNN model[32]		14-layer SCNN	97.57	N/A
RNN model (our work)		9-layer RCNN	88.54	87.93 ± 0.50
SNN model (our work)		9-layer SCNN	95.49	94.79 ± 0.18
Hybrid model (our work)		9-layer hybrid CNN	98.26	96.79 ± 0.87
SNN model[33]	CIFAR10-DVS	9-layer SCNN	N/A	67.22 ± 0.43
SNN model[34]		ResNet-19	67.80	N/A
SNN model[32]		VGG11	74.80	N/A
RNN model (our work)		13-layer RCNN	72.50	72.23 ± 0.22
SNN model (our work)		13-layer SCNN	77.00	76.10 ± 0.79
Hybrid model (our work)		13-layer hybrid CNN	78.00	77.60 ± 0.36

Moreover, in the newly-added Table 1 (referred to as the above Table R4 for your convenience), we extended our model to convolutional layer neural networks and conducted comparisons with single-paradigm networks of the same structure and other state-of-the-art works on N-MNIST, DVS-Gesture, and CIFAR10-DVS. We utilized a network structure of [input-128C3-AP2-256C3-AP2-384C3-256-10] for N-MNIST, [input-128C3-128C3-AP2-128C3-128C3-AP2-256C3-256C3-AP2-512C3-512C3-512C3-512C3-10] for CIFAR10-DVS. The results indicate that employing a more larger network

structure enables the proposed hybrid model not only significantly enhancing the accuracy over single-paradigm networks but also outperforming other advanced models across multiple neuromorphic datasets, which demonstrate the effectivness of our hybrid approach.

References

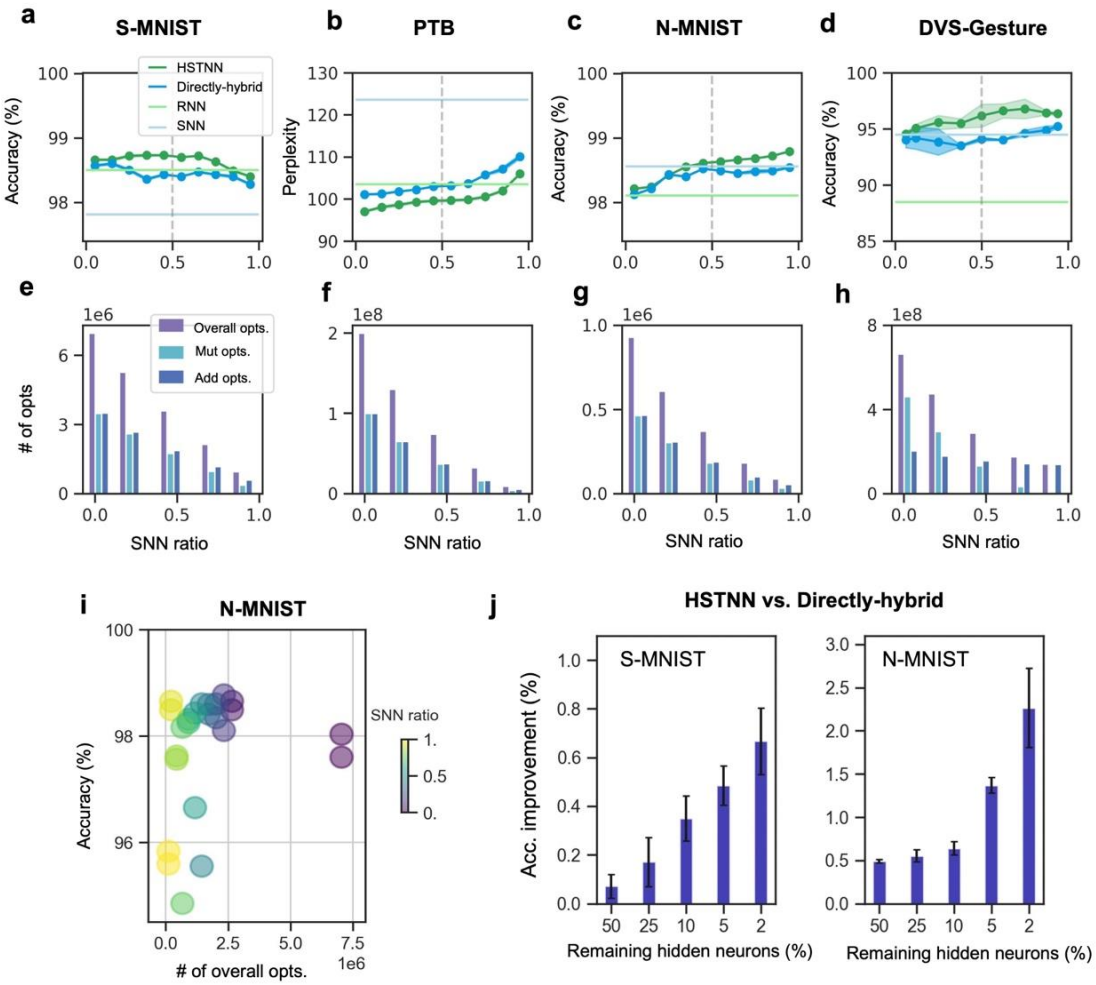
[1] Lee J H, Delbruck T, Pfeiffer M. Training deep spiking neural networks using backpropagation[J]. Frontiers in neuroscience, 2016, 10: 508.

[2] Zaremba W, Sutskever I, Vinyals O. Recurrent neural network regularization[J]. arXiv preprint arXiv:1409.2329, 2014.

[3] Wu Y, Zhao R, Zhu J, et al. Brain-inspired global-local learning incorporated with neuromorphic computing[J]. Nature Communications, 2022, 13(1): 65.

Revision:

Added Table 1 and updated Figure 2 with the results obtained using the larger and more standard structures and datasets. The updated Figure 2 is copied below for your convenience.



Revised Figure 2. **Comprehensive analysis of the adaptive trade-off between accuracy and the computational cost.** Impact of the SNN ratio on accuracy (upper) for (**a**) S-MNIST, (**b**) PTB, (**c**) N-MNIST, and (**d**) DVS-Gesture datasets. Impact of the SNN ratio on the number of operations (lower) for (**e**) S-MNIST, (**f**) PTB, (**g**) N-MNIST, and

(**h**) DVS-Gesture datasets. A two-hidden-layer network structure is adopted for PTB, S-MNIST, and N-MNIST, and a convolutional structure for DVS-Gesture. Note that accuracy on the PTB dataset is indicated by perplexity (ppl), where lower is better. (**i**) Comprehensive analysis of the trade-off between accuracy and the computational cost for N-MNIST, where the computational cost is measured by the total number of multiplication and addition operations. (**j**) Analysis of average accuracy improvement by HSTNNs compared to directly-hybrid models, measured by the ratio of remaining hidden neurons in the Restoration stage to those in the Adaptation stage. Error bars represent the standard deviation over five repeated trials.

Comment 3:

Network training: the Restorage stage only preserves half of hybrid neurons from the two pools of neurons. What's the impact of the sizes of the two pools?

Response:

We appreciate your insightful comment, which has led us to explore the benefits of using an increased size of neuron pools in the Adaption stage. We have conducted a control experiment that varies the size of the two neuron pools in the Adaptation stage and fixes the resulting pool size at 100. We utilize the same network structures in the Restoration stage as in the previous manuscript and assesse the impact of varying the initial neuron pool size, as shown in Figure R4. The results in Figure R4(a)&(b) demonstrate that using a larger neuron pool size for pre-training can enhance task accuracy over different SNN ratios (as compared to the accuracy in Figure R4(d)). Notably, the constructed neuron pools in the Adaptation stage are applicable across different SNN ratios required in the following stages, i.e., reusable for reconfigurations. Thus, adjusting the size of initial neuron pools has emerged as an additional effective hyper-parameter for controlling the task performance of HSTNNs.

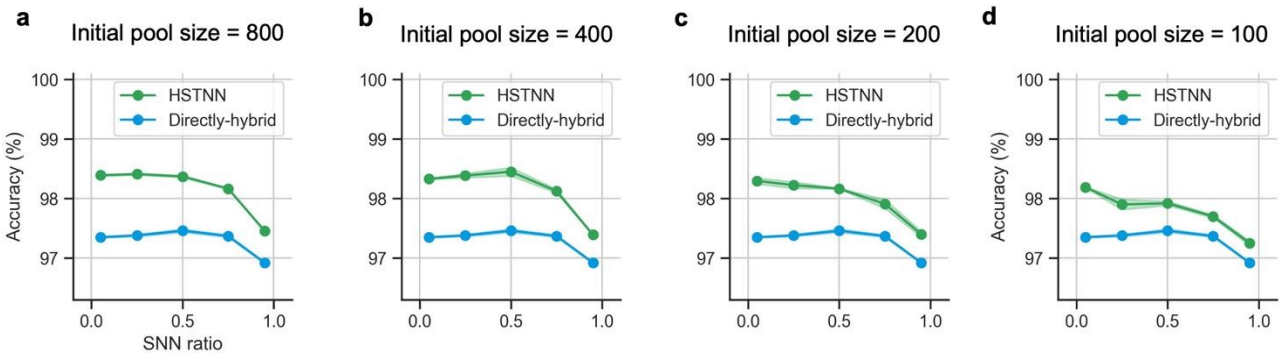


Figure R4. Adoption of a larger size of neuron pools in the Adaptation stage can improve the performance of resulting HSTNNs on S-MNIST. The network structure of [input-100-100-10] and default parameters as used in the previous manuscript are adopted for validation. Note that here we fix the pool size in the Restoration stage and only vary the pool size in the Adaptation stage.

Comment 4

Hardware implementations: I appreciated the inclusion of the hardware validation for the proposed

method, which is useful for indicating the practical advantages of the hybrid approach. However, some details and necessary explanations are still missing. In Figure 3c, it is unclear why the hybrid model can achieve a lower latency than the pure SNN. Given that there are multiple mapping and scheduling strategies (like fixed-core and variable-core mapping strategies), the rationale behind the chosen mapping strategy needs more clarifications.

Response:

Regarding the choice of mapping strategies: In our implementation, we utilized two mapping strategies, including fixed-core and variable-core mapping, to adapt to different computational workloads. In fixed-core mapping, each layer's SNN part is computed by one core and the RNN part by another core. This method is suited for layers with smaller computational workloads. Conversely, variable-core mapping, where the computational resources for SNN and RNN parts are proportional to the number of neurons, is suited for layers with larger computational workloads.

For the models on S-MNIST with smaller computational workloads, the fixed-core mapping is better because it can prevent resource waste and excessive power consumption by avoiding the inefficiencies of layer separation which underutilizes multi-core parallelism and increases data transfer. For the models on N-MNIST with larger computational demands in the first layer, variable-core mapping is preferable for effectively utilizing multiple cores.

Latency Explanation in Figure 5c: Regarding the networks used for S-MNIST, the adopted fixed-core mapping strategy separates the core allocation for SNN and RNN parts. It is observed that although a sole SNN (with the SNN ratio of 1) has the least computational workload, it doesn't achieve minimal latency owing to the utilization of only a single core. Conversely, for hybrid models, as the SNN ratio decreases, the latency of the SNN core shortens, and that of the RNN core lengthens. The total latency is the maximum of the latencies consumed by two cores, thus usually achieving the minimum value when their latencies are equal.

Regarding the networks used for N-MNIST, it is observed that the latencies at SNN ratios of 0.75 and 1 were almost identical due to two possible reasons. First, the minimal additional latency caused by the small RNN module in the first layer at the SNN ratio of 0.75 may be offset by the data transfer latency. Second, we use an oscilloscope to measure the running time by recording the trigger signals on the circuit board, which is not precise to the internal clock, potentially introducing latency errors.

Revision:

Included additional explanations in the revised manuscript to clarify our hardware implementation, which are detailed as follows.

"We employed two mapping strategies, namely fixed-core mapping and variable-core mapping, for

different hybrid layers, as presented in Figure 5(b). The small layers are mapped to two fixed cores respectively used for computing SNN and RNN modules, i.e., the fixed-core mapping strategy, while the larger layers are mapped to more cores where the computational resources for SNN and RNN modules are proportional to the number of neurons, i.e., the variable-core mapping strategy. The choice of the strategy depends on the potential for parallel execution of the layer across multiple cores, which will be detailed in *Methods*." in the Section "*Hardware deployability of HSTNNs*" of the *Results* part.

"We applied different mapping strategies for different layers considering the layer size. In fixed-core mapping, a layer is mapped onto a core group containing a small fixed number of cores dedicated to computing the RNN or SNN module, respectively. The workload for each core varies according to the SNN ratio. We used this strategy for small layers, such as those on S-MNIST, and fixed both the core numbers for SNN or RNN modules to one. This is because partitioning a small layer cannot fully utilize the parallelism of multiple cores but bring additional data transfer, which can result in resource wastage and excessive power consumption. In contrast, larger layers can utilize the resources of multiple cores better, so we used more cores with a fixed workload for each and allocated them for the SNN or RNN module according to the SNN ratio." in the Section "*Details of implementation on neuromorphic hardware*" of the *Methods* part.

"The execution latency of the sole SNN on S-MNIST is significantly higher than that of the HSTNN at the SNN ratio of 0.75 because the sole SNN only utilizes a single SNN core for computation in the fixed-core mapping strategy while the hybrid HSTNN can use both cores." in the Section "*Hardware deployability of HSTNNs*" section of the *Results* part.

"It is worth mentioning that the execution latency of HSTNNs can be shorter than that of the sole SNN. For the networks on S-MNIST, we noticed that although a sole SNN (with the SNN ratio of 1) has the least computational workload, it doesn't achieve minimal latency due to the utilization of only a single core. Conversely, for hybrid models, as the SNN ratio decreases, the latency of the SNN core shortens while that of the RNN core lengthens. The total latency is the maximum of the latencies consumed by the two cores, thus achieving the minimum value when their latencies are equal. In the case of the networks on N-MNIST, we found that the latencies at SNN ratios of 0.75 and 1 were almost identical, possibly because the minimal additional latency of the small RNN module in the first layer at the SNN ratio of 0.75 may be offset by the data transfer latency and the off-chip measurement we adopted might introduce errors." in the Section "*Details of implementation on neuromorphic hardware*" of the *Methods* part.

Comment 5:

Robot application: The demonstrated robot navigation is of great interest. However, some details

regarding the model's implementation are missing, including the exact size of the camera outputs and the sequence length of each sequence used in training and testing. Also, were the pre-trained CNNs and SNNs models fine-tuned for the tasks?

Response:

The camera outputs are of size 240×180 pixels, incorporating both positive and negative polarity information. This dimension means that each frame from the camera is a 240 (width) by 180 (height) sized pixel image. Each sequence used in our experiments has a length of 9 frames, in line with the experimental setup. For the tasks demonstrated, we utilize the pre-trained CNN and SNN models in the front network, which are fixed in our tasks.

Revision:

In the revised Section "*Details of experiments on the robot place recognition tasks*" of the *Methods* part, included more detailed descriptions regarding the model's implementation: "The CNN, used for RGB images, processed inputs of size 240×180×3 to include three RGB channels. The SCNN processed event images with an input size of 240×180 pixels, incorporating both positive and negative polarity information. The parameters of both pre-trained CNN and SCNN were fixed in our simulations. Outputs from the CNN and SCNN models were combined and fed into a three-layer HSTNN with a network structure of [input-500-500-100]. Due to the different resolutions of DVS and RGB cameras, we used nine consecutive event images and three corresponding RGB images as a training sample. The HSTNN was then constructed through our three-stage hybrid approach, learning to recognize the correct place from among 100 candidates. The training process involved 150 epochs for the Selection stage and 100 epochs for the Restoration stage. We employed the Adam optimizer and a cross-entropy loss function for the three-stage learning process."

Minor issues

(1) *I am curious if the hybrid model can support the combination of more complex biological models like the adaptive LIF or Hodgkin-Huxley model. It would be very useful to show the potential of proposed methods for gaining more benefits from biological models.*

(2) *The mathematical formulation for the storage is somewhat complex. Please clarify the dimensions of each tensor, like m^{n,r} and m^{n,s}*

(3) *In Equations 23-25, the approximate loss in the cost assessment is not clear. The authors should elaborate the terms ignored in these calculations.*

Response and revisions:

(1) Our methodology can adapt to the integration of different types of neuron models. Following your suggestion, we have applied our method to the integration of the adaptive LIF model and validated

its performance on the DVS-Gesture dataset. In this experiment, we adopted the same structure of [input-400-400-10] for both vRNN&LIF, vRNN&ALIF, and LSTM&LIF models. As shown in Figure R5, the inclusion of the adaptive threshold improved task performance from 86.7% to 89.67%. Encouraged by this feedback, we further explored the integration of LSTNN&LIF with the same network structure, and the integration of SCNN&RCNN with a recurrent convolutional neural network structure of [input-128C3-AP2-256C3-AP2-384C3-AP2-256-11]. These experiments demonstrated further enhancement in task performance, indicating the benefits of incorporating more learning features of bio-inspired and artificial neurons.

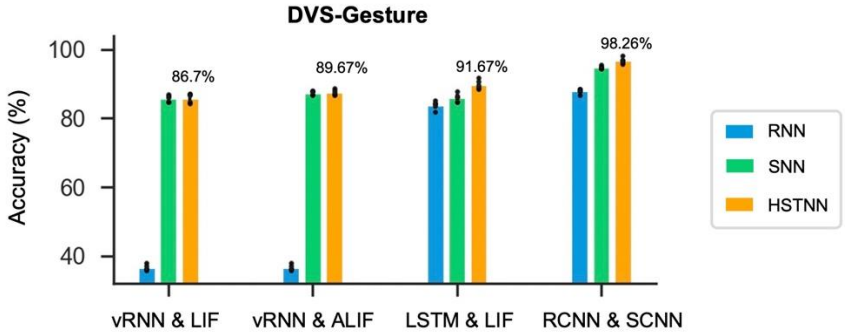


Figure R5. Results of HSTNNs on the DVS-Gesture dataset by integrating different types of neuron models. The HSTNNs with the optimal SNN ratios in each configuration are reported here for comparison. The error bars represent the standard deviation and the numbers above the yellow bars denote the best accuracies of HSTNNs over five runs.

According to this feedback, we have included the results of integrating richer learning features of bio-inspired and artificial neurons in the revised Figure 3(d) to illustrate the effectiveness of our methodology.

(2) We have clarified the dimensions of these tensors and polished our presentations on the mathematical formulation in the revised Section "*Details of the three-stage learning for HSTNN*" of the *Methods* part for greater clarity.

(3) We have elaborated on the approximate terms in Equations (23)-(25). We have included the relevant descriptions in the revised Section "*Details of parameter configurations and model evaluation*" of the *Methods* part: *"...In order to provide an intuitive and concise comparison, here we mainly estimate the computational cost of matrix operations, which produces a great impact on the hardware execution energy, and ignore the computation of the vector or scalar operations..."; "As with RNNs, the computational cost of the vector or scalar operations is omitted for clarity."*

REVIEWER COMMENTS

Reviewer #1 (Remarks to the Author):

Review: Adaptive spatiotemporal neural networks through complementary hybridization:

1. Input to RNN and SNN Layers: In Figure 1(b), it appears that the output of the RNN layer is fed into the SNN layer. However, it's important to note that SNN layers typically receive spiking inputs. If the inputs to RNN and SNN layers are indeed independent, this should be properly depicted in the diagram. Description in “Details of parameter configurations and model comprehensive evaluation” section doesn’t explain the input flow in the hybrid model.

2. Concatenation of Outputs: Equation (1) suggests that outputs from the RNN and SNN layers are concatenated and fed into the network at each timestep. If the inputs to these layers are independent, it raises questions about why their outputs are combined. This might result in a loss of important spiking information if not handled properly.

3. The addition of a comprehensive figure illustrating the HSTNN architecture, including clear flow of input and output dimensions for both MLP and CNN components, would significantly enhance the clarity and comprehensibility of the paper and address the points mentioned above better.

4. The paper mentions the hybrid coding scheme on page 5 but does not provide much clarity. Explain the neural coding schemes, particularly how rate-based and temporal coding are integrated, as mentioned in the rebuttal on page 2.

5. HSTNN architecture in Figure S3 is not clear.

6. The authors say that their framework is a general framework for hybrid models. Can they show some results of hybridization at layer level and compare with hybridization at neuron level? Looking at hybridization results at layer level for both MLP and CNN will be interesting.

7. The values in Table 1 show low accuracy for the RNN model compared to SNN, this brings us back to our previous comment on the need for a complex search process to find a hybrid model when the contribution seems to be mostly coming from the spiking neurons. Also, if RNN is performing worse compared to SNN why do we need a hybrid model?

8. In equation (9) how is $\Delta w = -wu$.? Is it because we want the unimportant weights to be set to zero? If so, how is equation 12 used to prune unimportant neurons. Please explain clearly with a figure or proper equations with a description. Last paragraph on page 15 under Adaptation stage is not clear. How is $(H-1)-1=Tr(H(u,u))$ in equation 12? Please explain clearly which equation out of 8-12 is the score for the neuron?

Reference:
1. J. Hagenaars, F. Paredes-Valles, and G. De Croon, “Self-supervised learning of event-based optical flow with spiking neural networks,” Advances in Neural Information Processing Systems, vol. 34, pp. 7167–7179, 2021.

Reviewer #2 (Remarks to the Author):

I sincerely appreciate the tremendous efforts made by the authors, and I am pleased to see that my concerns have been thoroughly addressed. In particular, I would like to emphasize the commendable efforts made by the authors in conducting a substantial number of new experiments to establish the efficacy of the proposed hybrid strategy. This work undoubtedly holds great potential and I believe it will make a significant contribution to the field of hybrid neural network design and its implementation in edge computing scenarios.

Point-by-Point Response to Reviewers' Comments

We would like to thank the reviewers for spending valuable time and raising insightful comments that truly helped improve our manuscript. The point-by-point responses are provided as follows to address the questions raised by the reviewers. All revisions in the manuscript are marked in blue.

Comment 1:

Input to RNN and SNN Layers: In Figure 1(b), it appears that the output of the RNN layer is fed into the SNN layer. However, it's important to note that SNN layers typically receive spiking inputs. If the inputs to RNN and SNN layers are indeed independent, this should be properly depicted in the diagram. Description in "Details of parameter configurations and model comprehensive evaluation" section doesn't explain the input flow in the hybrid model.

Concatenation of Outputs: Equation (1) suggests that outputs from the RNN and SNN layers are concatenated and fed into the network at each timestep. If the inputs to these layers are independent, it raises questions about why their outputs are combined. This might result in a loss of important spiking information if not handled properly.

The addition of a comprehensive figure illustrating the HSTNN architecture, including clear flow of input and output dimensions for both MLP and CNN components, would significantly enhance the clarity and comprehensibility of the paper and address the points mentioned above better.

Response:

We appreciate your instructive comments. To address your concerns more effectively, we would like to first clarify two key designs and then combine them with a comprehensive figure.

Our main goal is to synergistically leverage different neural dynamics for processing spatiotemporal data with better comprehensive performance in terms of accuracy, robustness, and the computational cost. Considering the distinct differences between RNNs and SNNs, two specialized designs have been made in this work, aiming at facilitating simple yet efficient modelling for HSTNNs:

(1) **Independent neural dynamics within RNN and SNN neuron groups:** To fully leverage the information representation and processing characteristics of RNNs and SNNs, each layer in the HSTNN remains two neuron groups: the RNN group and the SNN group, whose model equations are performed independently for keeping the integrity of their respective neural dynamics. This is in the independent sense.

(2) **Hybrid inputs for RNN and SNN neuron groups:** To combine the complementary information from two neuron groups, the outputs from the RNN and SNN neuron groups are concatenated

before fed into the next layer. In other words, each neuron group in the next layer receives hybrid inputs including both non-spiking activations and spiking events. In this way, each neuron group can observe all features from both neuron groups of the previous layer, enabling deep information fusion. This is in the hybrid sense.

With the above design principles of HSTNNs in mind, we try our best to address your concerns:

Regarding the inputs to RNNs and SNNs: As aforementioned, the information processing within RNN and SNN neuron groups are independent, and the inputs to the two neuron groups are not independent but combined. You are correct that the SNN neuron group receives non-spiking inputs from the RNN neuron group of the previous layer. This is reasonable because we do not intend to build a pure SNN model whose inputs in each layer should be spike events. On the contrary, the goal of this work is to construct a hybrid model which can combine the complementary advantages of both RNNs and SNNs through the above design principles. Such a way of using hybrid inputs for SNNs can also be found in prior work on neuromorphic computing [1, 2] and supported by some advanced neuromorphic chips featuring hybrid architectures [3, 4].

Regarding the concern on losing spiking information and concatenation of outputs: It is important to note that RNNs are simulated with synchronized time as SNNs in our model. While the proposed hybridization methodology in principle allows RNNs and SNNs to update with independent simulation steps, due to the current lack of suitable software frameworks, synchronized updates are used in establishing HSTNNs, which might be the simplest yet effective method for building such hybrid models on existing programming platforms. In short, the outputs of RNNs and SNNs are updated synchronously and combined before fed into the next layer at each time step.

Compared to traditional single-paradigm SNNs, the SNN neuron group receives complete spike events from the previous layer, where the spiking information is not lost. Besides, the SNN neuron group further receives non-spike activations from the RNN neuron group in the previous layer, thus the dynamics of the SNN neuron group can be influenced by RNN neuron groups in pre-layers, which is just the information fusion feature of our hybrid model. Altogether, there is no loss of spiking information.

Regarding the architecture illustration: Per your suggestion, we have plotted a comprehensive illustration of the hybrid information flow of input and output dimensions in both fully connected (FC) and convolutional (Conv) architectures, as shown in Figure R1. Particularly, in Figure R1(a), we show four spiking inputs, two non-spiking inputs, and two neurons per hidden layer, with one in each different neuron group for simple illustration. The architecture for convolutional neural networks (CNNs) is generally rather complicated than that of the FC-based architectures. Please note that the main difference between the two architectures lies in the computational operations, where FC layers employ matrix multiplications (denoted by $\times$) and Conv layers employ convolutional operations

(denoted by $*$). Since the neural dynamics of RNN and SNN groups are independent as aforementioned, the internal network structures and models can be configured flexibly, the selection of FC and Conv layers in each neuron group is not restricted. For the input manner, the use of hybrid inputs in each layer is the same for both FC and Conv architectures.

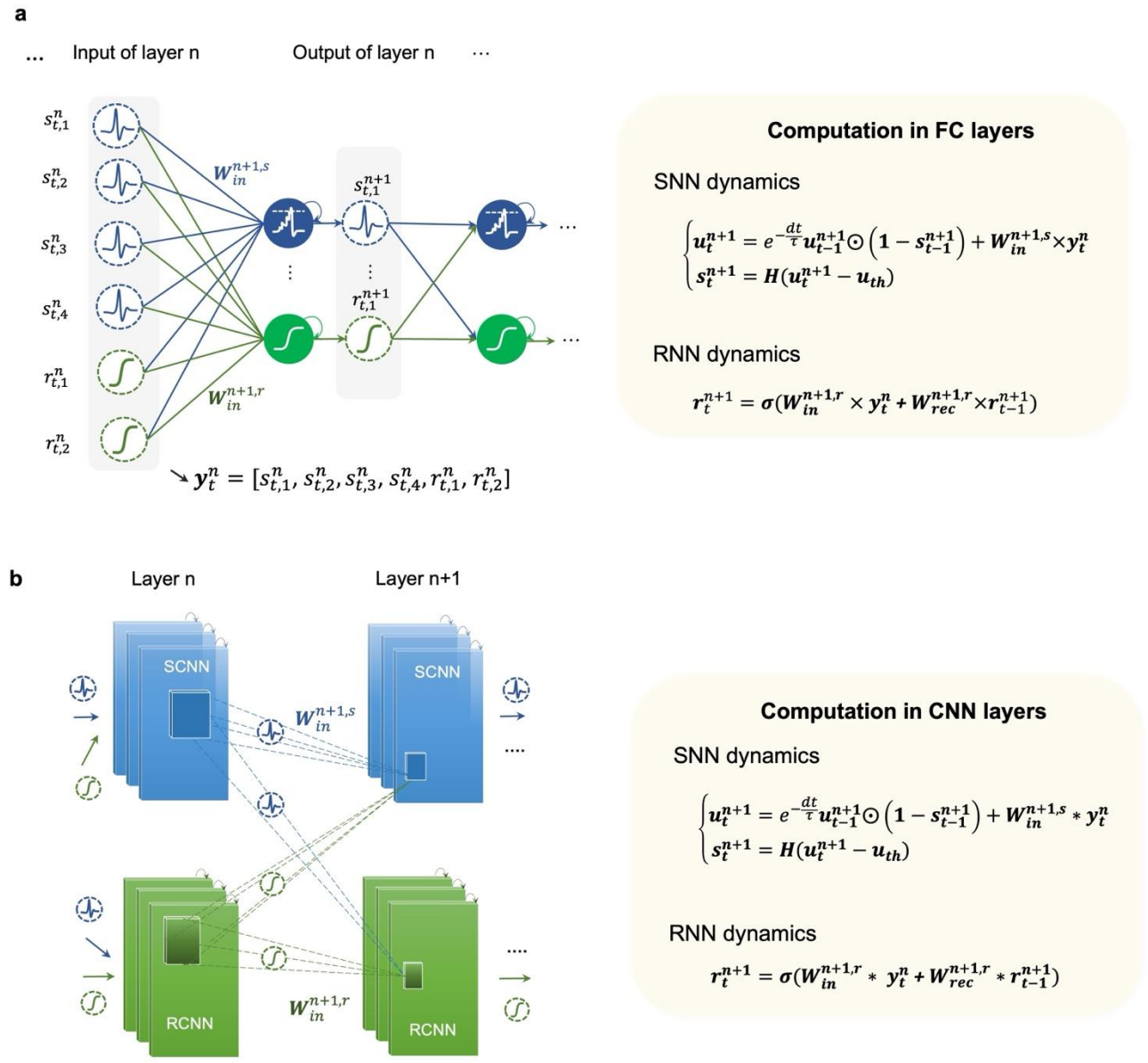


Figure R1. A comprehensive illustration of the hybrid information flow in a fully connected (FC) architecture (a) and the convolutional (Conv) architecture (b). For demonstration, we consider four spiking inputs and two non-spiking inputs from the previous layer in panel (a). The main difference between FC and Conv architectures lies in the computational operations, where FC layers employ weight matrix multiplications denoted by $\times$ and Conv layers employ convolutional operations denoted by $*$. For the input manner, the use of hybrid inputs in each layer is the same for different network architectures.

Finally, we would like to summarize the following three innovative contributions of this work again for your reference:

(1) Hybrid neuron-wise modelling framework for processing variable spatiotemporal data:
We propose a hybrid modelling framework that can synergistically combine RNNs and SNNs and employ a hybrid information representation and processing mechanism to tackle the complexities of spatiotemporal data. Our methodology supports the integration of various RNN modules (e.g., RNN and LSTM) and spiking modules (e.g., LIF and ALIF), demonstrating high efficiency in creating compact hybrid networks under the same modelling framework for variable application scenarios.

(2) Superior accuracy, stronger robustness, and better comprehensive performance:
Our empirical analyses on seven multifaceted datasets show that the HSTNNs can outperform single-paradigm RNNs and SNNs. This includes four neuromorphic classification datasets (N-MNIST, DVS-Gesture, DVS-CIFAR10, MVSEC) and three non-spiking sequential datasets (S-MNIST, PTB, and WikiText-2). Furthermore, our comprehensive analyses validate that HSTNNs exhibit stronger robustness and enable a better balance between accuracy and the computational cost.

(3) Validation of hybrid models by hardware simulation and robot navigation tasks:
Our extensive validation efforts, involving in hardware implementation and challenging robotic navigation tasks, confirm the practical deployability and robust applicability of our hybrid models. The outcomes offer an energy-efficient integrated solution utilizing the complementary features of diverse neural network models for processing real-world applications.

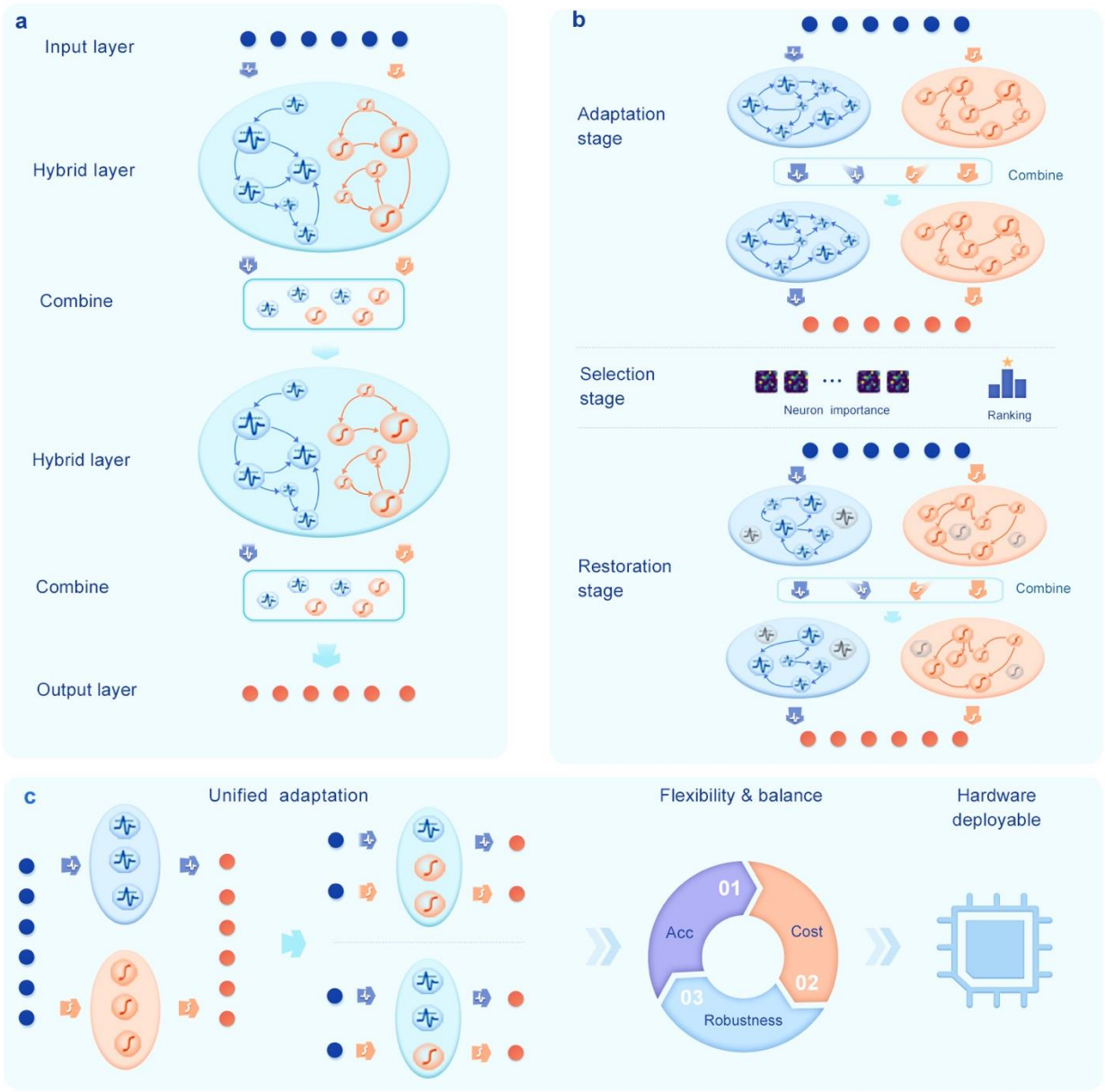
References:
[1] Fang W, Chen Y, Ding J, et al. SpikingJelly: An open-source machine learning infrastructure platform for spike-based intelligence[J]. Science Advances, 2023, 9(40): eadi1480.
[2] Chen X, Yang Q, Wu J, et al. A hybrid neural coding approach for pattern recognition with spiking neural networks[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2024, 46(05): 3064-3078.
[3] Orchard G, Frady E P, Rubin D B D, et al. Efficient neuromorphic signal processing with Loihi 2[C]//2021 IEEE Workshop on Signal Processing Systems (SiPS). IEEE, 2021: 254-259.
[4] Pei J, Deng L, Song S, et al. Towards artificial general intelligence with hybrid Tianjic chip architecture[J]. Nature, 2019, 572(7767): 106-111.

Revision:
Added more descriptions of the hybrid information flow in the revised Section "*Establishment of HSTNNs*", adjusted Figure 1, and added Supplementary Figure S1 to enhance the clarity.
"To facilitate efficient training of hybrid models, non-spiking recurrent neurons are simulated with synchronized timing as spiking neurons. At each time step, neurons in the two populations receive the same hybrid inputs from the previous layer and their outputs in the previous time step, and then update

the neural dynamics and generate their respective outputs ($\boldsymbol{r}_t$ or $\boldsymbol{s}_t$). These outputs are combined before being forwarded to the next processing layer.
…A detailed illustration of the hybrid information flow is provided in Supplementary Figure S1."

Added a supplementary section including a comprehensive figure (i.e., Figure R1) to illustrate the hybrid information flow of HSTNNs per your suggestion.

Improved the clarity of Figure 1(b) and (c) by adding blue arrows and boxes, and relevant texts.



Revised Figure 1. Methodology of creating HSTNNs. (a) The HSTNN architecture. Each hidden layer of an HSTNN contains two types of neurons and their outputs are combined before being injected into the next layer. (b) The three-stage learning process for creating an HSTNN. In the Adaptation stage, we create two redundant populations of neurons in each hidden layer and apply the unified BPTT learning algorithm to warm the

connection weights. In the Selection stage, we propose a neuron-aware selection mechanism to measure the importance of neurons and select important neurons. In the Restoration strategy, we shrink the network by invalidating unselected neurons and their connections and retrain the compact model until convergence. (c) HSTNNs exhibit several advantageous features: an HSTNN can be conveniently initialized under a unified *Adaption* stage no matter what the specific task scenario is; it enables a flexible balance between accuracy, cost, and robustness to satisfy variable performance requirements in practice; it can be deployed on neuromorphic hardware for constructing an efficient application system.

Comment 2:

The paper mentions the hybrid coding scheme on page 5 but does not provide much clarity.

Response:

We apologize for our previous unclear descriptions, which may have caused your misunderstanding. By 'hybrid coding scheme', we referred to a hybrid information representation that combines both non-spiking and spiking neural networks. Specifically, the output of each layer includes both features produced by the RNN and SNN neuron groups, indicating a hybrid information representation. This integration enhances the representation of spatiotemporal data and overcomes the individual limitations of traditional single-paradigm neural networks, as evidenced by various sequence tasks demonstrated in Figure 2 of our manuscript. To improve clarity, we have replaced the 'hybrid coding scheme' with 'hybrid information representation' throughout our manuscript.

Revision:

Replaced 'hybrid coding strategy' with more accurate descriptions in the revised *Results*:

'This improvement is likely due to the hybrid information representation of non-spiking and spiking neural networks with richer neuronal computation mechanisms, which increases learning nonlinearity and integrates the complementary strengths of SNNs and RNNs on addressing specific tasks. For instance, such integration can increase the representation precision of single-paradigm SNNs on the non-spiking PTB dataset and enhancing the robustness of single-paradigm RNNs against input fluctuations on the spiking N-MNIST dataset.'

Comment 3:

Explain the neural coding schemes, particularly how rate-based and temporal coding are integrated, as mentioned in the rebuttal on page 2.

Response:

The core mechanism of the HSTNN for integrating rate-based and temporal-based coding involves a neuron-wise hybridization strategy designed for RNNs and SNNs, which is a prominent feature distinct from previous methods [1, 2]. The strategy ensures the preservation of information integrity from the

input to the output for both RNN and SNN paths. For the SNN path, this allows for the complete transmission of information encoded in spikes across the network.

To elucidate how the HSTNN utilizes rate-based or temporal-based information from spike trains, we first describe the general decoding scheme adopted by HSTNNs:

$$\boldsymbol{y}_t^N = \boldsymbol{W}^N Concat\left(\boldsymbol{r}_t^N, d(\boldsymbol{s}_1^N, \dots, \boldsymbol{s}_t^N | \boldsymbol{\omega})\right)$$

where vector $\boldsymbol{r}_t^N$ represents the output of the RNN path at the t-th time step in the last layer, vector $\boldsymbol{s}_t^N$ denotes the output of the SNN path at the t-th time step in the last layer, and matrix $\boldsymbol{W}^N$ denotes the linear projection weight. $d(\cdot|\boldsymbol{\omega})$ is a parametric decoder that can be designed to decode the rate-based or temporal information from the spike train into a vector representation using parameter matrix $\boldsymbol{\omega}$. The matrix $\boldsymbol{\omega}$ can be specifically tailored according to the chosen decoding scheme as indicated by Figure R2. Next, we show how different neural coding schemes can be used for HSTNNs.

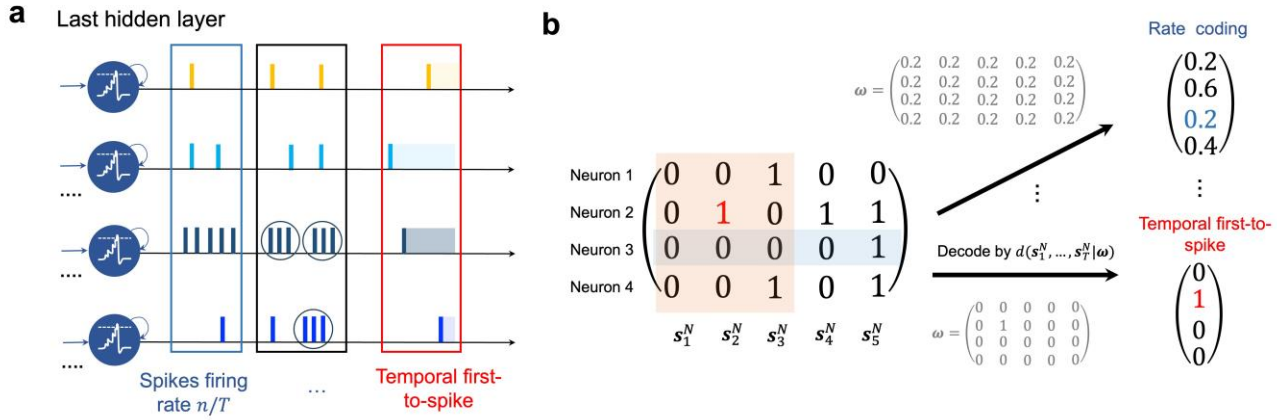


Figure R2. Illustration of decoding the output hybrid information flow of the HSTNN. (**a**) The output spike train can encode different types of spatiotemporal patterns. (**b**) These patterns can be formulated as a binary weight matrix (left matrix) where each row records the firing behaviors of one neuron across different time steps, and each column represents the firing behavior at a specific time step across neurons. By using the coding-specific parametric function $d(\cdot|\boldsymbol{\omega})$, an HSTNN can integrate rate-based or timing-based spiking coding schemes. For illustration, we show that an HSTNN can decode rate-based information by employing an equal-weight matrix; it can implement a straightforward first-to-spike decoding [2, 3] by leveraging a binary weight matrix.

(1) Integration of rate coding: If we conceptualize

$$d(s_{i1}^N, \dots, s_{it}^N | \boldsymbol{\omega}) = \sum_{t'=1}^{t} \omega_{it'} \boldsymbol{s}_{it'}^N$$

and set an equal weight parameter $\omega_{i1} = \omega_{i2} = \cdots = \omega_{it} = 1/t, \forall i$, the HSTNN decodes the rate-based information from the output spike.

(2) Integration of temporal coding: Now, we showcase that the construction of $d(\cdot)$ allows us to decode temporal information and leverage this feature into the gradient-based learning framework.

Taking the first-to-spike coding scheme as an example, it utilizes the temporal order of output spikes among a neuron population to reduce the latency of neural processing [1]. Let's consider a commonly used implementation scheme [2, 3] in which the output neuron with the earliest activation, resulting in a first-spike response, determines the network's output information. Specifically, the network monitors the output spike trains for the earliest spike event. Upon detecting the first spike, the network immediately halts further inference and returns the output result.

This process can be effectively implemented by identifying the index of the first-firing neuron, for example, the neuron i firing at the t-th time step, and configuring the corresponding parameter matrix $\boldsymbol{\omega}$ by

$$\omega_{jt'} = \begin{cases} 1, & j = i \text{ and } t' = t, \\ 0, & others. \end{cases}$$

By doing so, $d(\cdot)$ decodes the earliest firing timing and returns a binary vector representing the index of the first-firing neuron. Note that the depression components—the zero entries of $\boldsymbol{\omega}$—can be further designed, such as setting them to exponentially decay to reflect the firing time delays. However, for clarity in this illustration, we have simply set them to zero. Then the temporal coding result from the firing pattern (see Figure R2(b)) is used to calculate the loss function. Throughout minimizing the loss function, potentiating the firing of the target neuron at that specific time step while depressing others (see Eq. (10) of [1] as a training example), the network can learn to optimize its parameters and leverage the earliest output spike for fast decision.

References:

[1] Chen, Xinyi, et al. "A hybrid neural coding approach for pattern recognition with spiking neural networks." IEEE Transactions on Pattern Analysis and Machine Intelligence 46.05 (2024): 3064-3078.

[2] Thorpe S, Delorme A, Van Rullen R. Spike-based strategies for rapid processing[J]. Neural networks, 2001, 14(6-7): 715-725.

[3] Gardner B, Grüning A. Supervised learning with first-to-spike decoding in multilayer spiking neural networks[J]. Frontiers in computational neuroscience, 2021, 15: 617862.

Comment 4:

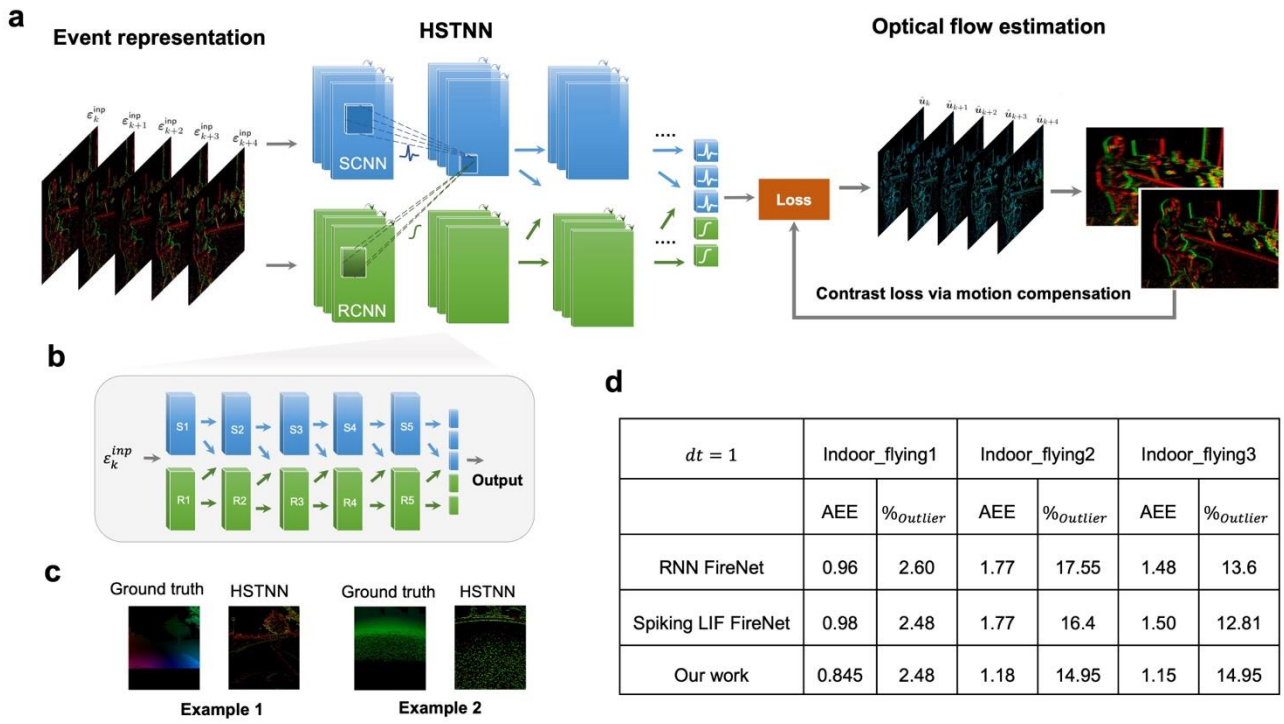
HSTNN architecture in Figure S3 is not clear.

Response:

Considering your feedback, we have enhanced the clarity of Supplementary Figure S3 by explicitly illustrating the network architecture as well as the information flow and providing a more detailed description for the employed architecture in the caption.

Revision:

Polished Supplementary Figure S3 (copied as Figure R3 below for your convenience) and added more accurate descriptions in the revised manuscript.



d

$dt = 1$	Indoor_flying1		Indoor_flying2		Indoor_flying3	
	AEE	$\%_{Outlier}$	AEE	$\%_{Outlier}$	AEE	$\%_{Outlier}$
RNN FireNet	0.96	2.60	1.77	17.55	1.48	13.6
Spiking LIF FireNet	0.98	2.48	1.77	16.4	1.50	12.81
Our work	0.845	2.48	1.18	14.95	1.15	14.95

Figure R3. **Illustration of applying the proposed hybrid model for the optical flow estimation task.** (**a**) Optical flow pipeline for constructing the hybrid model, adapted from Figure 1 in reference [1]. (**b**) The network structure was adapted from the original FireNet [2], which is composed of three spiking and non-spiking recurrent convolutional layers (S1&R1, S2&R2 and S4&R4), two spiking and non-spiking recurrent ResNet blocks (S3&R3 and S5&R5), and a final prediction layer. Each spiking or non-spiking layer has a single stride, 32 output channels, and employs 3×3 kernel encoders. Neurons in these layers follow the corresponding neural dynamics described by Equations (6), (7), and (15) in reference [1]. Each spiking (or non-spiking) residual block [2] contains two spiking (or non-spiking) convolutional layers with skip connections, aiming to preserve information and prevent gradient vanishing in the motion estimation. The final prediction layer has a single stride and two output channels, which integrates spiking representation with non-spiking components to estimate motion. (**c**) Qualitative evaluation of warped input events by the motion compensation from outputs of the HSTNN. (**d**) Quantitative results including average endpoint error (AEE), and the percentage of points with AEE greater than 3 pixels and 5\% of the optical flow vector magnitude (\%Outliers), alongside results from single-paradigm models [1].

References:

[1] Hagenaars J, Paredes-Vallés F, De Croon G. Self-supervised learning of event-based optical flow with spiking neural networks[J]. Advances in Neural Information Processing Systems, 2021, 34: 7167-7179.

[2] Scheerlinck C, Rebecq H, Gehrig D, et al. Fast image reconstruction with an event camera[C]//Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2020: 156-163.

Comment 5:

The authors say that their framework is a general framework for hybrid models. Can they show some results of hybridization at layer level and compare with hybridization at neuron level? Looking at hybridization results at layer level for both MLP and CNN will be interesting.

Response:

There seems to be a misunderstanding regarding our statement about the 'general framework'. We indeed mentioned 'general framework', stated as '*our method, with a different goal of developing a general framework designed for integrating neuron models with diverse temporal dynamics' (*noted on page 5 of our last round of responses*).* It means that that our approach can generally support the integration of different types of neuronal dynamics, rather than producing any predetermined structure.

Nevertheless, to address your comment, we have tried our best to conduct comparative studies between HSTNNs and layer-wise hybridization. We establish two layer-wise models: [input-RNN-SNN-decoder] (termed as LWR) and [input-SNN-RNN-decoder] (termed as LWS), by employing a direct hybridization method as used in Figure 2, and validate their performance on two datasets, S-MNIST and DVS-Gesture. Below, we first present the main results from these comparative studies, followed by detailed descriptions of the modelling approaches.

Table R1: Comparison of HSTNNs with layer-wise hybridizations using MLP architectures on S-MNIST. Mean and std are calculated over three repeated trials.

SNN Ratio	0.05	0.25	0.50	0.75	0.95
LWR (Inp-RNN-SNN)	98.12±0.02	98.00±0.07	97.80±0.06	97.39±0.04	94.51±0.14
LWS (Inp-SNN-RNN)	96.21±0.22	97.14±0.24	95.77±0.20	91.36±0.85	62.89±5.02
HSTNN	**98.16±0.09**	**98.02±0.11**	**97.91±0.02**	**97.64±0.10**	**97.22±0.08**

Table R2: Comparison of HSTNNs with layer-wise hybridizations using CNN architectures on DVS-gesture. Mean and std are calculated over three repeated trials.

SNN Ratio	0.125	0.25	0.50	0.75	0.875
LWR (Inp-RNN-SNN)	91.09±1.34	91.90±0.43	92.59±0.71	92.01±1.99	93.06±0.57
LWS (Inp-SNN-RNN)	87.38±3.38	85.65±2.44	87.15±2.96	84.03±0.75	80.44±0.33
HSTNN	**92.59±0.87**	**92.48±0.66**	**93.29±1.00**	**93.75±0.49**	**93.75±0.49**

(1) Main results of comparing HSTNNs with layer-wise hybridizations:

We apply the HSTNN and two layer-wise hybridization models to S-MNIST and DVS-gesture using MLP architectures with two hidden layers and CNN architectures with two convolutional layers, respectively. To ensure fairness, we maintain the same overall number of hidden neurons and the same number of output channels for the three models in our experiments.

Tables R1 and R2 demonstrate that HSTNNs perform better than the two layer-wise hybridization methods on these datasets. Such a superior performance is likely due to the proposed three-stage optimization process, as the *Selection* stage in principle has considered the layer-wise architecture as a special case where all significant neurons of the same type are selected within the same layer. By doing so, HSTNNs can search for an architecture with potentially better task performance from a redundant network model created in the *Adapation* stage.

(2) Details to experimental setups and modelling:

Signal propagation for LWS and LWR: In the LWS model, the input first passes through an SNN layer, followed by an RNN layer, with the output decoded using the RNN output at the last time step. Conversely, in the LWR model, the input first passes through an RNN layer, then an SNN layer, with the output decoded using the average number of spikes over a given time window. At each time step, an RNN or SNN layer can receive non-spiking or spiking signals from its preceding layer, respectively, depending on the signal type configured in different hybrid models. Other network parameters, including the optimizer, preprocessing of DVS data, and training epochs, are consistent across all comparative models and follow the setup of relevant tasks in our manuscript, as detailed in Supplementary Table S1.

MLP architecture: The LWS and LWR models are trained from scratch with a predefined network configuration. Assuming the total number of hidden neurons is N and the intended SNN ratio among all hidden neurons is r, the LWS is configured with *round(Nr)* SNN neurons in the first hidden layer and *round(N(1-r))* RNN neurons in the second layer. The LWR configures the number of different neurons in an opposite way. On S-MNIST, we set $N=200$. Thus, the MLP architecture for LWS is [input-*round(200r)*-*round(200(1-r))*-FC10] and for LWR is [Input-*round(200(1-r))*-*round(200r)*-FC10]. For HSTNNs, we employ an architecture of [input-200-200-10] during the *Adaptation* stage and prune both spiking and non-spiking neurons during the *Selection* stage so as to maintain the final number of hidden neurons as the same as that used in LWR and LWS, ensuring a fair comparison.

CNN architecture: We follow a similar strategy as that of MLP and manipulate the hybridization on feature maps (FMs). The backbone network structure for both LWR and LWS is [input-C*x*-AP2-C*x*-AP2-decoder], where different models configure different number of neurons, x, in the first and second convolutional layers and an MLP decoder, [-FC256-FC11] with the ReLU activation, is used here for decoding the outputs of the second pooling layer. We assume the overall number of FMs in the two convolutional layers is 32, which remains the same for different models. Then, for a given SNN ratio r, the LWS configures *round(32r)* FMs in the first convolutional layer and *round(32*(1-r))* in the second layer; the LWR vice versa. HSTNNs employ an architecture of [input-C32-AP2-C32-AP2-FC256-FC11] in the *Adaptation* stage and then prune the *Selection* stage to maintain the same number of FMs (i.e., 32) as LWR and LWS for a given SNN ratio in the comparison.

Comment 6:

The values in Table 1 show low accuracy for the RNN model compared to SNN, this brings us back to our previous comment on the need for a complex search process to find a hybrid model when the contribution seems to be mostly coming from the spiking neurons. Also, if RNN is performing worse compared to SNN why do we need a hybrid model?

Response:

You are correct that Table 1 demonstrates the lower accuracy of RNNs compared to SNNs on three typical neuromorphic datasets. However, RNNs also achieve higher accuracy compared to SNNs on non-spiking datasets such as S-MNIST, PTB, and Wiki2 (see Figure 2). Such distinct task performances of RNNs and SNNs in different tasks, in our view, justify the importance of a hybrid model, as it can help find a better tradeoff between accuracy and the computational cost by tuning the hybridization ratio between RNNs and SNNs, and can usually achieve better performance over sole RNNs and SNNs (see Table 1 and Figure 2). Notably, in many complicated real-world environments, task features are agnostic, and we cannot know which models among RNNs and SNNs are better in advance. Thus, employing hybrid models provides a stable and superior option for these cases.

In addition, please note that Table 1 merely measured task accuracy. By assessing the robustness against diverse noise and the computational cost (see Figures 3 and 5), we demonstrate that hybrid models further enable improving overall performance in terms of accuracy, robustness, and the computational cost.

Comment 7:

- *Please explain clearly which equation out of 8-12 is the score for the neuron?*
- *In equation (9) how is $\Delta w = -wu.$? Is it because we want the unimportant weights to be set to zero? If so, how is equation 12 used to prune unimportant neurons. Please explain clearly with a figure or proper equations with a description.*
- *How is (H-1)-1=Tr(H(u,u)) in equation 12?*
- *Last paragraph on page 15 under Adaptation stage is not clear.*

Response:

Equations (9)-(12) mainly employed on a parameter saliency measure used in the seminal pruning framework [1-3] to evaluate the importance score of a neuron. Below, we first explain how Equations (8)-(12) calculate the importance score and then address your concerns about Equations (9) and (12). Please note that in the previous manuscript, we intended to provide a brief description of the key equations and left the details in relevant references to keep succinctness and sharpen our contributions. In response to your last comment, we have significantly revised the presentation of this part by adding more descriptions to enhance its readability and completeness.

(1) **Using Equations (8)-(12) to calculate the importance scores of neurons:**

The importance score of a neuron was calculated by aggregating scores of its afferent weight parameters. The importance score of a weight parameter is based on a parameter saliency measure [1, 2], which calculates the smallest change of the loss function caused by the perturbation of the weight parameter.

Equation (8) begins with a general description of the above loss change due to the perturbation. Let Δw denote the pruning perturbation such that the corresponding weight becomes zero (i.e., $\Delta w + w = 0$). We denote the corresponding change of the loss as ΔL:

$$\Delta L = \nabla_{\mathrm{w}} L(w)^T \Delta w + \frac{1}{2}\Delta w^T H \Delta w + O(\|\Delta w\|^3)\,.$$

Following previous studies [1, 3], we assume that a pre-trained neural network (i.e., the HSTNN established by the *Adaptation* stage) has converged to a local minimum of the loss function L, where the gradient yields $\nabla_{\mathrm{w}} L(w) = 0$. Hence, Equation (8) indicates that the perturbation can be primarily associated with the second-order term $\Delta w^T H \Delta w$, where H denotes the Hessian matrix.

Equation (9), detailed below, further defines the measure as finding the smallest perturbation of L, under the constraint of removing the specific weight (i.e., setting it to zero by $\Delta w_u = -w_u$). Note that the perturbation can apply to multiple weight parameters; for generality, we denote w_u and Δw_u as vectors in the derivation:

$$\min_{\Delta \mathrm{w}} \frac{1}{2}\Delta w^T H \Delta w = \frac{1}{2}\begin{pmatrix}\Delta w_u \\ \Delta w_i\end{pmatrix}^T \begin{pmatrix}H_{u,u} & H_{u,i} \\ H_{i,u} & H_{i,i}\end{pmatrix}\begin{pmatrix}\Delta w_u \\ \Delta w_i\end{pmatrix},\ s.t. \Delta w_u = -w_u.$$

Equations (10)-(11) use the Lagrangian method to solve the optimization problem in Equation (9), and derive a final solution:

$$\frac{1}{2}\Delta w^T H \Delta w = \frac{1}{2} w_u^T \left(H_{u,u} - H_{u,i} H_{i,i}^{-1} H_{i,u}\right) w_u.$$

By employing the Schur complement of the inverse matrix, we derive the left part of Equation (12):

$$\frac{1}{2} w_u^T \left(H_{u,u} - H_{u,i} H_{i,i}^{-1} H_{i,u}\right) w_u = \frac{1}{2} w_u^T [H^{-1}]_{u,u}^{-1} w_u.$$

Note that in the original OBS, one has to measure the perturbations for all the parameters separately and needs to calculate the inverse of Hessian matrix. Directly estimating $w_u^T [H^{-1}]_{u,u}^{-1} w_u$ causes a high computational cost, especially for a very large neural network model with many parameters. On the other hand, in the context of our hybrid model, we are more interested in the comprehensive impact of a group of afferent weights connecting to the same neuron. Hence, we use a structural pruning strategy [2] that groups several weight parameters to compute the corresponding perturbation when that weight group is pruned. Additionally, following previous studies [1, 3, 4], we assume that the diagonal elements (blocks) contain the main saliency features. Then, we handle the Hessian matrix H by approximating each block with a diagonal operator:

$$\frac{1}{2}w_u^T[H^{-1}]_{u,u}^{-1}w_u \approx \frac{1}{2}w_u^T H_{u,u} w_u \approx \frac{1}{2u}w_u^T Tr(H_{u,u})w_u = \frac{\|w_u\|_2^2}{2u}Tr(H_{u,u})$$

where u denotes the dimention of w_u and $Tr(H_{u,u})$ denotes the trace of the block diagonal Hessian matrix of the unimportant group. The last approximation holds from an expectation property that if different entries of w_u are independent or off-diagonal elements of H are relatively small compared to diagonal elements (as assumed by previous studies [1, 2, 4]), we have $E(w_u^T H_{u,u} w_u) \approx 1/u\, E(w_u^T tr(H_{u,u})w_u)$.

(2) **Explanation of $\Delta w_u = -w_u$**:
Setting $\Delta w_u = -w_u$ is a strategy to zero out the evaluated weight and assess its significance by directly observing how its removal influences the loss change. This process can be formulated by Equation (9) as discussed above.

(3) **How is Equation (12) used to prune unimportant neurons:**
As indicated above, the importance score is measured by how its removal influences the smallest change of the loss. However, directly estimating the relevant perturbation ΔL involves the computation of the inverse matrix of H. To derive a numerically tractable solution, Equation (12) employs the Hessian trace method to approximate the smallest perturbation, as formulated by Equation (9). By doing so, the importance score *S* of a neuron, measured by the above smallest perturbation, can be evaluated using the Hessian trace:

$$S \equiv \frac{\|w_v\|_2^2}{2v}Tr(H_{v,v}),$$

where w_v denotes a vector consisting of all afferent weight parameters of the neuron and v denotes the dimmention of w_v.

(4) **How is (H-1)-1=Tr(H(u, u)) in Equation (12)**:
There seems to be a misunderstanding in the interpretation of this equation from the reviewer's question. In fact, we use $\frac{1}{u}\Delta w_u^T Tr(H_{u,u})\Delta w_u$ to approximate the quadratic form in the original equation (i.e., $\Delta \boldsymbol{w}_u^T[H^{-1}]_{u,u}^{-1}\Delta \boldsymbol{w}_u$), rather than estimating $[H^{-1}]_{u,u}^{-1} = Tr(H_{u,u})$.

As indicated above, this approximation is based on a structural pruning strategy[1, 3, 4] that groups several weight parameters to compute the corresponding perturbation when that weight group is pruned. On this basis, the Hessian matrix H can be handled by approximating each block with a diagonal operator and then the estimation of $\frac{1}{2}w_u^T[H^{-1}]_{u,u}^{-1}w_u$ can be simplified by $\frac{1}{u}\Delta w_u^T Tr(H_{u,u})\Delta w_u$. More detailed derivations and justifications are provided in our response to the point (1) of this comment.

(5) Last paragraph on page 15 under Adaptation stage is not clear:

We guess your mention of the '*last paragraph on page 15 under Adaptation stage*' actually refers to the '*last paragraph on page 15 under Selection stage*,' as all descriptions of the Adaptation stage appear on page 14. In response, we have reorganized this part by providing more detailed derivations, clarifying the approximation, and enhancing the descriptions of the selection process.

Reference

[1] LeCun Y, Denker J, Solla S. Optimal brain damage[J]. Advances in neural information processing systems, 1989, 2.

[2] Yu S, Yao Z, Gholami A, et al. Hessian-aware pruning and optimal neural implant[C] //Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2022: 3880-3891.

[3] Hassibi B, Stork D. Second order derivatives for network pruning: Optimal brain surgeon[J]. Advances in neural information processing systems, 1992, 5.

[4] Liu C, Zhang Z, Wang D. Pruning deep neural networks by optimal brain damage[C] // Interspeech. 2014, 2014: 1092-1095.

Revisions:

Clarified the computation of the Hessian matrix and added more detailed derivations in the revised *Methods*.

"…The neuronal importance is evaluated by aggregating the importance scores of its afferent weights. The weight score is evaluated based on a classical parameter saliency measure [16, 43], which accesses the saliency of a parameter through calculating the smallest change of the loss function ΔL caused by perturbing the specific parameter.

Next, we formulate the smallest change of the loss caused by the perturbation as an optimization problem and employ a neuron-wise pruning strategy to adapt the saliency measure for the hybrid model. A key relationship exists between the parameter perturbation and neuron pruning: pruning an unimportant neuron can be formalized as perturbing the model such that all weights connecting to the unimportant neuron become zero (i.e., $\Delta w = -w$, where Δw denotes the weight perturbation). On this basis, the loss change ΔL, expressed in the Taylor expansion form, is governed by

$$\Delta L = \nabla_{\mathrm{w}} L(w)^T \Delta w + \frac{1}{2} \Delta w^T H \Delta w + O(\|\Delta w\|^3).$$

Based on the OBS, we assume that a trained neural network model (i.e., the HSTNN established by the *Adaptation* stage) has converged to a local minimum of the loss function L, where the gradient yields $\nabla_{\mathrm{w}} L(w) = 0$, and the Hessian matrix H is positive semidefinite. Thus, ΔL can be primarily associated with the second-order term containing the Hessian matrix $\Delta w^T H \Delta w$.

We then formulate the process of finding the smallest change of the loss function ΔL while removing the specific weight parameter w_u as an optimization problem:

$$\min_{\Delta w} \frac{1}{2}\Delta w^T H \Delta w = \frac{1}{2}\begin{pmatrix}\Delta w_u \\ \Delta w_i\end{pmatrix}^T \begin{pmatrix}H_{u,u} & H_{u,i} \\ H_{i,u} & H_{i,i}\end{pmatrix}\begin{pmatrix}\Delta w_u \\ \Delta w_i\end{pmatrix},\ s.t.\ \Delta w_u = -w_u\ .$$

…By employing the Schur complement of the inverse matrix, we have

$$\frac{1}{2} w_u^T \left(H_{u,u} - H_{u,i} H_{i,i}^{-1} H_{i,u}\right) w_u = \frac{1}{2} w_u^T [H^{-1}]_{u,u}^{-1} w_u.$$

The original OBS requires measuring the perturbation estimation for all parameters separately and calculating the inverse of the Hessian matrix, which leads to an intolerable computational cost in large-scale neural networks. Instead, we focus on evaluating the comprehensive impact of a group of afferent weights connecting to the same neuron. Therefore, we develop a neuron-wise strategy based on the structural pruning method [17]. Specifically, we first group all weight parameters connecting to one target output neuron and compute the corresponding perturbation when this group of weights is pruned. Additionally, following previous studies [17, 43, 44], we assume the main saliency features are contained within the diagonal blocks. Therefore, the Hessian matrix H can be approximated by a diagonal block matrix where each block includes a diagonal operator:

$$\frac{1}{2} w_u^T [H^{-1}]_{u,u}^{-1} w_u \approx \frac{1}{2} w_u^T H_{u,u} w_u \approx \frac{1}{2u} w_u^T Tr\left(H_{u,u}\right) w_u = \frac{\|w_u\|_2^2}{2u} tr\left(H_{u,u}\right).$$

where $Tr\left(H_{u,u}\right)$ denotes the trace of the block diagonal Hessian of the unimportant group.

The above equation effectively avoids the computation of the inverse of the Hessian matrix by using the trace $Tr\left(H_{u,u}\right)$. Furthermore, we employ the Hutchinson method for calculating the trace, which employs stochastic vectors to effectively estimate the Hessian operator (see Equations (6)-(7) in reference [46] for implementation).

Consequently, the neuronal importance score, measured by the above smallest perturbation, can be evaluated by Hessian trace estimation with only a moderate computational cost. Considering the distinct neuronal dynamics and representation manners between RNNs and SNNs, we rank the importance scores of spiking and non-spiking neurons separately. Specifically, we collect the same types of neurons from all layers and uniformly sort them according to the important scores.”

Reviewer 2

Comment:

I sincerely appreciate the tremendous efforts made by the authors, and I am pleased to see that my concerns have been thoroughly addressed. In particular, I would like to emphasize the commendable efforts made by the authors in conducting a substantial number of new experiments to establish the efficacy of the proposed hybrid strategy. This work undoubtedly holds great potential and I believe it will make a significant contribution to the field of hybrid neural network design and its implementation in edge computing scenarios.

Response:

We are very pleased to see that we were able to address the reviewer's previous comments. We also thank the reviewer for taking the time to evaluate our paper and for the insightful comments, which have significantly contributed to enhancing the quality of our work.

REVIEWERS' COMMENTS

Reviewer #2 (Remarks to the Author):

Minor points:
- In the caption of Figure 1, 'Adaption' should be replaced by 'Adaptation'.
- In line 328, I think 'd_i()' should be clearly defined as the i-th component of 'd()'.
- In the caption of Figure S4, 'motion compensation from outputs of the HTSNNs' should be replaced by 'motion compensation using outputs of the HTSNN'”

Point-by-Point Response to Reviewers' Comments

We would like to thank the reviewers for spending valuable time and raising insightful comments that truly helped improve our manuscript. The point-by-point responses are provided as follows to address the questions raised by the reviewers.

Reviewer 2

Comment:

Minor points:

- In the caption of Figure 1, 'Adaption' should be replaced by 'Adaptation'.
- In line 328, I think 'd_i()' should be clearly defined as the i-th component of 'd()'.
- In the caption of Figure S4, 'motion compensation from outputs of the HTSNNs' should be replaced by 'motion compensation using outputs of the HTSNN'

Response:

We have corrected these descriptions accordingly in the revised manuscript.