# DO SPIKING NEURAL NETWORKS LEARN SIMILAR REPRESENTATION WITH ARTIFICIAL NEURAL NETWORKS? A PILOT STUDY ON SNN REPRESENTATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Spiking Neural Networks (SNNs) have recently driven much research interest owing to their bio-plausibility and energy efficiency. The biomimicry spatial-temporal communication and computation mechanisms are the key differences that set SNNs apart from current Artificial Neural Networks (ANNs). However, some essential questions exist pertaining to SNNs and yet are little studied: *Do SNNs learn similar representation with ANN? Does the time dimension in spiking neurons provide additional information?* In this paper, we aim to answer these questions by conducting a representation similarity analysis between SNNs and ANNs using Centered Kernel Alignment (CKA). We start by analyzing the spatial dimension of the networks, including both the width and the depth. Furthermore, our analysis of residual connection shows that SNN learns a periodic pattern, which rectifies the representations in SNN to ANN-like. We additionally investigate the effect of the time dimension on SNN representation, finding that deeper layers encourage more dynamics along the time dimension. Other aspects like potential improvement in terms of accuracy, efficiency, and adversarial robustness are also analyzed using CKA. We hope this work will inspire future research to fully comprehend the representation of SNNs.

## 1 INTRODUCTION

Lately, Spiking Neural Networks (SNNs) (Tavanaei et al., 2019; Roy et al., 2019; Deng et al., 2020; Panda et al., 2020; Christensen et al., 2022) have received increasing attention thanks to their biology-inspired neuron activation and efficient neuromorphic computation. SNNs process with binary spike representation and therefore avoid the need for multiplication operations during inference. Neuromorphic hardware such as TrueNorth (Akopyan et al., 2015) and Loihi (Davies et al., 2018) demonstrate that SNNs can save energy by orders of magnitude compared to ANNs. Hybrid architecture like Tianjic (Pei et al., 2019) suggests its potential power for general intelligence when integrated with traditional artificial infrastructure.

Despite increasing interest in SNNs, there is limited understanding of how spiking neurons affect the representation learned in SNNs *beyond its efficiency and accuracy.* Investigating this fundamental question is critical, especially with the consensus that spiking neurons learn distinct spatial-temporal information compared to ANNs (Tavanaei et al., 2019; Ghosh-Dastidar & Adeli, 2009). Understanding the representation learned in SNN can also promote further research developments, e.g., designing spiking-oriented architectures (Kim et al., 2022; Na et al., 2022).

More concretely, we ask, does the leaky integrate-and-fire (LIF) neuron (Burkitt, 2006) in SNNs learn distinct non-linear activations than the rectified linear unit (ReLU) (Krizhevsky et al., 2012) neuron in ANNs? How do the width and the depth of the neural network affect the representation learned in SNNs and ANNs? Does the extra temporal dimension in SNNs yields unique intermediate features? On the neuromorphic dataset, how does the SNN process event-based data? How are SNNs and ANNs differ in terms of robustness? In this paper, we study these core questions, through a detailed analysis of ResNets (He et al., 2016a) and VGG-series (Simonyan & Zisserman, 2015) models using a representation similarity analysis tool. In particular, we utilize the popular Centered Kernel Alignment (CKA) (Kornblith et al., 2019) to measure the similarity between SNN and ANN.

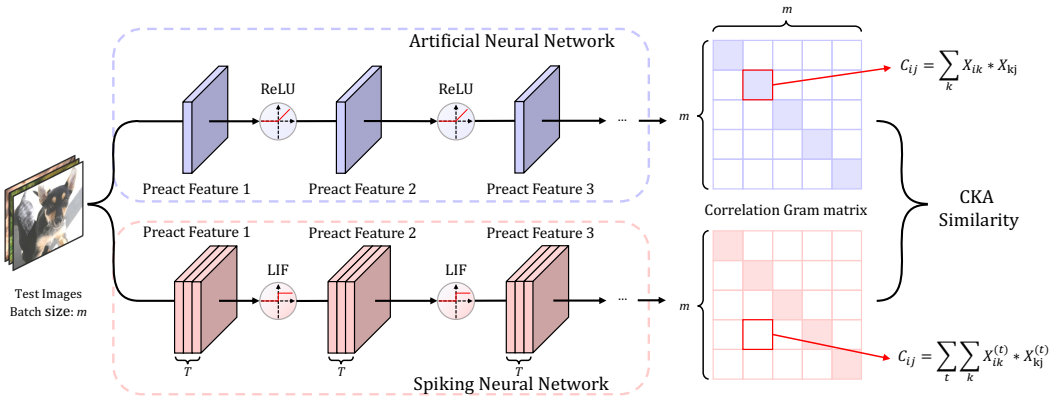


Figure 1: **The representation similarity analysis workflow.** The test images are fed into both ANN and SNN, then we record the intermediate feature for computing the correlation matrix, which is used for inferring the CKA similarity (Kornblith et al., 2019).

Fig. 1 demonstrates the overall workflow of our representation similarity analysis framework. Our analysis spans both spatial and temporal dimensions of SNN, as well as the impact of network architecture and input data. Our contributions can be summarized as follows:

- We analyze the representation similarity between SNNs and ANNs using the centered kernel alignment, in order to determine whether SNNs bring distinct features and differ from ANN.
- Various aspects of representation similarity between SNNs and ANNs are examined, including spatial and temporal dimensions, the type of input data, and network architecture.
- Our analysis of representation similarity could inspire future research directions. We also provide two prototype methods to improve the temporal dynamics and efficiency in SNNs using CKA.

## 2 Related Work

**Spiking Neural Networks (SNNs).** SNNs are recognized as a candidate for next-generation artificial intelligence. Generally, the SNN algorithms to obtain high-performance can be divided into two categories: (1) ANN-SNN conversion (Rueckauer et al., 2016; 2017; Han et al., 2020; Sengupta et al., 2019; Han & Roy, 2020) and (2) direct training SNN from scratch (Wu et al., 2018; 2019). Conversion-based methods utilize the knowledge from ANN and convert the ReLU activation to a spike activation mechanism. This type of method can produce an SNN in a short time. For example, in Rueckauer et al. (2017), one can find the percentile number and set it as the threshold for spiking neurons. Authors in Deng & Gu (2021) and Li et al. (2021) decompose the conversion error to each layer and then propose to reduce the error by calibrating the parameters. However, achieving near-lossless conversion requires a considerable amount of time steps to accumulate the spikes. Direct training from scratch allows SNNs to be operated in extremely low time steps, even less than 5 (Zheng et al., 2020). To enable gradient-based learning, direct training leverages surrogate gradient to compute the derivative of the discrete spiking function. This also benefits the hyperparameters' choice in spiking neurons. Recent works (Fang et al., 2021; Rathi & Roy, 2020; Kim & Panda) co-optimize parameters, firing threshold, and leaky factor together via gradient descent. Our analysis is mostly based on directly trained SNN, as converted SNN contains ANN features only, which may be misleading for representation comparison.

**Representation Similarity Analysis (RSA).** RSA (Kriegeskorte et al., 2008) was not originally designed for analyzing neural networks only. Rather, it is used for representation comparison between any two computational models. Prior works like Khaligh-Razavi & Kriegeskorte (2014); Yamins et al. (2014) use RSA to find the correlation between the visual cortex features and the convolutional neural network features. Authors of Seminar (2016); Raghu et al. (2017); Morcos et al. (2018); Wang et al. (2018) studied the RSA between different neural networks. However, recent work (Kornblith et al., 2019) argues that none of the above methods for studying RSA can

yield high similarity even between two different initialization of the same architecture. They further propose CKA which has become a powerful evaluation tool for RSA and has been successfully applied to several studies. For example, Nguyen et al. (2020) analyzes the representation pattern in extremely deep and wide neural networks. Raghu et al. (2021) studies the representation difference between convolutional neural networks and vision transformers with CKA. In this work, we leverage this tool for comparing ANN and SNN.

# 3 PRELIMINARY

## 3.1 ARTIFICIAL NEURONS AND SPIKING NEURONS

In this paper, vectors/matrices are denoted with bold italic/capital letters (*e.g.* $\boldsymbol{x}$ and $\mathbf{W}$). Constants are denoted by small upright letters. For non-linear activation function in artificial neurons, we use the rectified linear unit (ReLU), given by $\boldsymbol{y} = \max(0, \mathbf{W}\boldsymbol{x})$. As for the non-linear activation function in spiking neurons, we adopt the well-known Leaky Integrate-and-Fire (LIF) model. Formally, given a membrane potential $\boldsymbol{u}^{(t)}$ and a pre-synaptic input $\mathbf{W}\boldsymbol{x}^{(t+1)}$ at time step $t$, the pre-synaptic potential ($\boldsymbol{u}^{(t+1),\text{pre}}$) in LIF neuron is updated as

$$\boldsymbol{u}^{(t+1),\text{pre}} = \tau \boldsymbol{u}^{(t)} + \mathbf{W}\boldsymbol{x}^{(t+1)} \tag{1}$$

Here, $\tau$ is a constant leak factor within $(0, 1)$. Let $v_{th}$ be the firing threshold, the membrane potential will fire a spike when it exceeds the threshold, and then, is hard-reset to 0, given by

$$\boldsymbol{y}^{(t+1)} = \begin{cases} 1 & \text{if } \boldsymbol{u}^{(t+1),\text{pre}} > v_{th} \\ 0 & \text{otherwise} \end{cases}, \quad \boldsymbol{u}^{(t+1)} = \boldsymbol{u}^{(t+1),\text{pre}} \cdot (1 - \boldsymbol{y}^{(t+1)}) \tag{2}$$

After firing, the spike output $\boldsymbol{y}^{(t+1)}$ will propagate to the next layer and become the input $\boldsymbol{x}^{(t+1)}$ of the next layer. Note that here the layer index is omitted for simplicity.

## 3.2 CENTERED KERNEL ALIGNMENT

Let $\mathbf{X}_s \in \mathbb{R}^{m \times Tp_1}$ and $\mathbf{X}_a \in \mathbb{R}^{m \times p_2}$ contain the representation in an arbitrary layer of SNN with $p_1$ hidden neurons across $T$ time steps and the representation in an arbitrary layer of ANN with $p_2$ hidden neurons, respectively. Here $m$ is the batch size and we concatenate features from all time steps in the SNN altogether. We intend to use a similarity index $s(\mathbf{X}_s, \mathbf{X}_a)$ to describe how similar they are. We use the Centered Kernel Alignment (CKA) (Kornblith et al., 2019) to measure this:

$$\text{CKA}(\mathbf{K}, \mathbf{L}) = \frac{\text{HSIC}(\mathbf{K}, \mathbf{L})}{\sqrt{\text{HSIC}(\mathbf{K}, \mathbf{K})\text{HSIC}(\mathbf{L}, \mathbf{L})}}, \quad \text{HSIC}(\mathbf{K}, \mathbf{L}) = \frac{1}{(m-1)^2} tr(\mathbf{KHLH}). \tag{3}$$

Here, $\mathbf{K} = \mathbf{X}_s\mathbf{X}_s^\top, \mathbf{L} = \mathbf{X}_a\mathbf{X}_a^\top$ are the Gram matrices as shown in Fig. 1. Each Gram matrix has the shape of $m \times m$, reflecting the similarities between a pair of examples. For example, $\mathbf{K}_{i,j}$ indicates the similarity between the $i^{th}$ and $j^{th}$ example in the SNN feature $\mathbf{X}_s$. Further measuring the similarity between $\mathbf{K}$ and $\mathbf{L}$, one can measure whether SNN has a similar inter-example similarity matrix with ANN. Let $\mathbf{H} = \mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^\top$ be the centering matrix, the Hilbert-Schmidt Independence Criterion (HSIC) proposed by Gretton et al. (2005) can conduct a test statistic for determining whether two sets of variables are independent. HSIC $= 0$ implies independence. The CKA further normalizes HSIC to produce a similarity index between 0 and 1 (the higher the CKA, the more similar the input pair) which is invariant to isotropic scaling. In our implementation, we use the unbiased estimator of HSIC (Song et al., 2012; Nguyen et al., 2020) to calculate it across mini-batches.

# 4 DO SNNS LEARN SIMILAR REPRESENTATION WITH ANNS?

In this section, we comprehensively compare the representation learned in SNNs and ANNs. Our primary network architecture is ResNet with identity mapping block (He et al., 2016b), which is the standard architecture in modern deep learning for image recognition. We also provide RSA on VGG-series networks in Appendix A.1. There are two differences between our SNNs and ANNs. First, ANNs adopt Batch Normalization layer (Ioffe & Szegedy, 2015) and SNNs use the time-dependent Batch Normalization layer (Zheng et al., 2020) which normalizes the feature across all

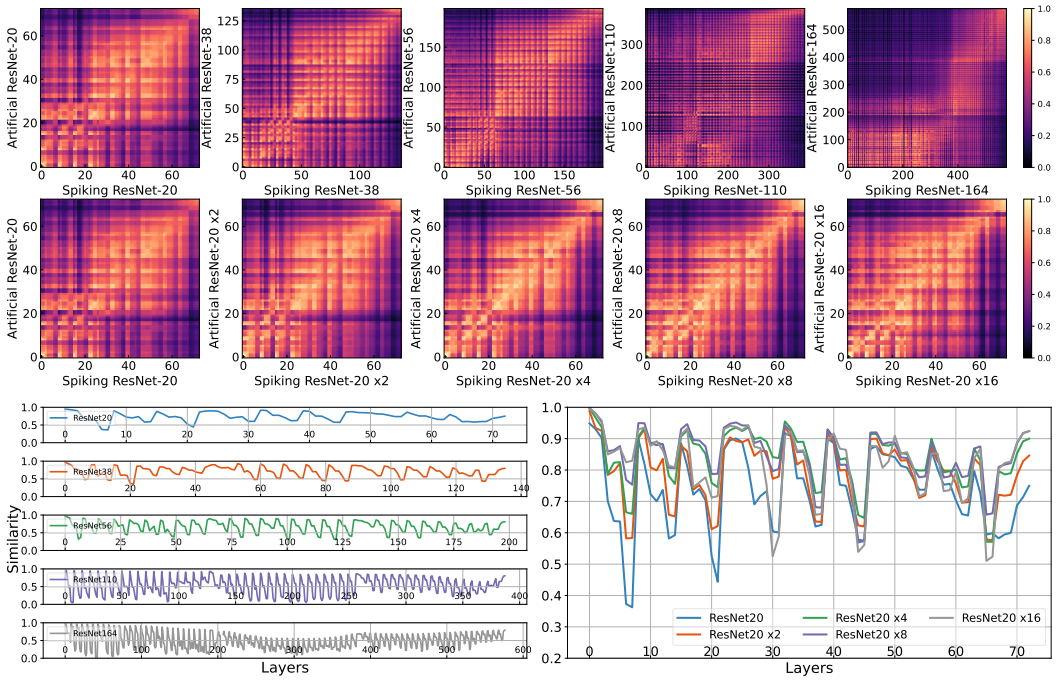


Figure 2: **CKA heatmap between SNNs and ANNs with different depth and width on the CIFAR-10 dataset. Top**: the CKA cross-layer heatmap across different depths from 20 layers to 164 layers. **Middle**: the CKA cross-layer heatmap across different widths from the original channel number to 16 times. **Bottom**: visualizing only the corresponding layer, which is the diagonal of the CKA heatmap.

time steps (*i.e.* $\mathbf{X}_s$). Second, the ANNs use ReLU activation, and SNNs leverage the LIF spiking neurons. For default SNN training, we use direct encoding, $\tau = 0.5$ for leaky factor, $v_{th} = 1.0$ for firing threshold, and time step $T = 4$, which are tuned for the best training performance on SNNs. We train the network on CIFAR10/100 (Krizhevsky et al.), and several event-based datasets (Li et al., 2017; Orchard et al., 2015). SNNs are trained with surrogate gradients. Detailed training setup and codes can be found in the supplemental material.

### 4.1 Scaling up Width or Depth

We begin our study by studying how the spatial dimension of a model architecture affects internal representation structure in ANNs and SNNs. Specifically, we gradually scale up the depth or the width of the network and use the CKA representation similarity metric. Starting from a ResNet-20, we either increase its number of convolutional layers up to 164 or increase its channel number up to 16 times as original (see detailed network configuration in Table C.1). For each network, we compute CKA between all possible pairs of layers, including convolutional layers, normalization layers, ReLU/LIF layers, and residual block output layers. Therefore, the total number of layers is much greater than the stated depth of the ResNet, as the latter only accounts for the convolutional layers in the network. Then, we visualize the result as a heatmap, with the $x$ and $y$ axes representing the layers of the network, going from the input layer to the output layer. Following Nguyen et al. (2020), our CKA heatmap is computed on 4096 images from the test dataset. We also lay out the top-1 accuracy of each model in Table B.1.

As shown in Fig. 2, the CKA heatmap emerges as a checkerboard-like grid structure, especially for the deeper neural network. In ResNet-20, we observe a bright block in the middle and deep layers, indicating that ANNs and SNNs learn overlapped representation. As the network goes deep, we find the CKA heatmap becomes darker, meaning that representations in ANNs and those in SNNs are diverging. Notably, we find the last $^2/_3$ layers in artificial ResNet-164 exhibit significantly different representations than spiking ResNet-164, which demonstrates that deeper layers tend to learn disparate features.

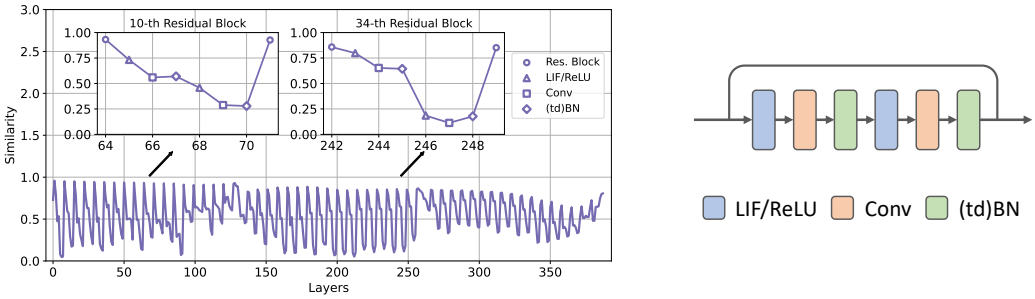


Figure 3: **Emergence of periodic jagged CKA curve. Left**: CKA curve of ResNet-110. We subplot the 10-$th$ and the 34-$th$ residual blocks in ResNet-110, which forms a periodic jagged curve. **Right**: The architecture specification of the residual block we used.

In Fig. 2 middle part, we progressively enlarge the channel number of ResNet-20. In contrast to depth, the heatmap of wider neural networks becomes brighter, which indicates the representations in SNNs and ANNs are converging. Interestingly, although the majority of layers are learning more similar representations between ANN and SNN in wide networks, the last several layers still learn different representations.

We further select only the diagonal elements (*i.e.* the CKA between layers at the same positions) in the heatmap and plot them in Fig. 2 bottom part. Interestingly, we observe that deeper networks tend to derive a curve with a jagged shape. This means some layers in SNN indeed learn different representations when compared to ANN, however, *the difference is intermittently mitigated*. In later sections, we will show that the mitigation of dissimilarity is performed by residual connection. As for width, we generally notice that CKA curves mostly become higher for wider networks, especially when comparing ResNet-20 and ResNet-20 $\times 8$. We additionally provide the CKA results on the CIFAR-100 dataset in Appendix A.3.

### 4.2 The Effect of Residual Connections

In Fig. 2, the CKA curves appear with a periodic jagged shape. To investigate what causes this similarity oscillation, we investigate each layer in a residual block. In Fig. 3 left, we plot the CKA curve of the ResNet-110 and additionally sample two residual blocks, the 10-$th$ and the 34-$th$ block, whose architecture details are depicted in Fig. 3 right. Surprisingly, we find that every time when the residual connection meets the main branch, the CKA similarity restores nearly to 1. Moreover, every time when the activation passes a convolutional layer or an LIF layer, the similarity decreases. The BN layers, in contrast, do not affect the similarity since it is a linear transformation. These results substantiate that the convolutional layers and LIF layers in SNNs are able to learn different representations than ANNs. However, the representation in the residual branch still dominates the representation in post-residual layers and leads to the junction of ANN's and SNN's representation.

To further explore why residual connections can restore the representations in SNNs to ANN-like, we conduct an ablation study. We select one of the three stages in the spiking ResNet-56 where the residual connections are disabled selectively. In Fig. 4, we visualize the CKA heatmaps of *SNN itself*, which means both $x$ and $y$ axes are the same layers in SNN. The first heatmap demonstrates the full residual network, while the remaining three heatmaps show the partial residual networks, with the 1st, 2nd, and 3rd stages disabled, respectively. Our observations can be summarized as:

(1) In terms of inter-stage similarity, residual connections can preserve the input information from the previous stage. In the 1st and 2nd heatmaps in Fig. 4, we find residual blocks can have high similarity with their former stage. The non-residual block, however, does not have this property. In the 3rd and 4th heatmaps, we can see that blocks without residual connections exhibit significantly different representations when compared to their former stage. Therefore, as long as ANN and SNN learn similar representations in the first layer, the similarity can propagate to very deep layers due to residual connections.

(2) In terms of intra-stage similarity, the non-residual stage's heatmap appears with a uniform representation across all layers, meaning that layers in this stage are similar. In contrast, residual stages share a grid structure. In Fig. 4 bottom side, we train a linear probe (Alain & Bengio, 2016)

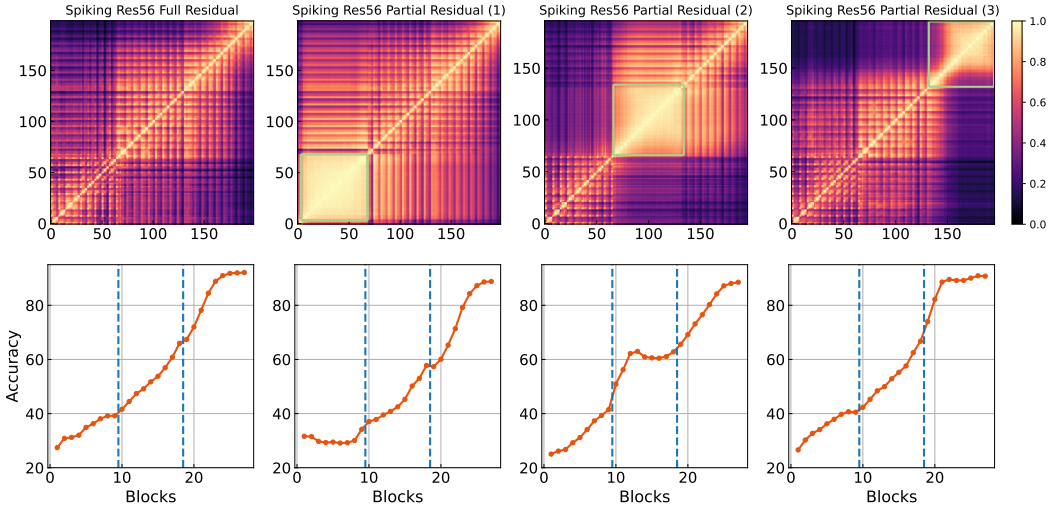


Figure 4: **The effect of residual connections in SNNs.** We selectively disable residual connections in one of three stages in the ResNet-56. **Top**: the CKA heatmap of *SNN itself* with different types of non-residual blocks. **Bottom**: The linear probing accuracy of each block. The non-residual stage is annotated with green square □.

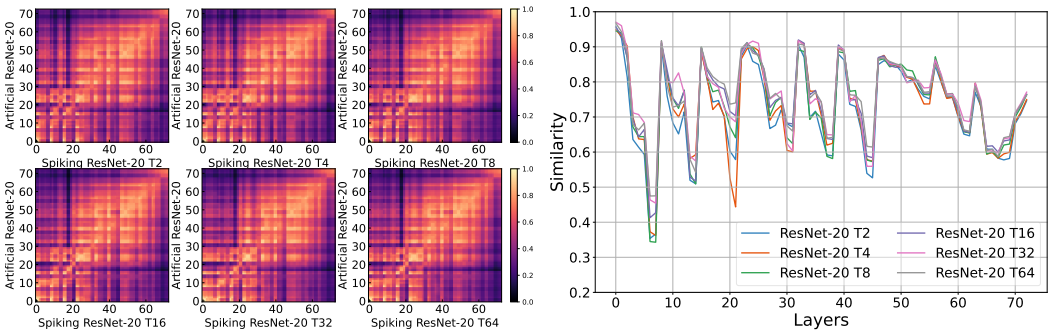


Figure 5: **The effect of time steps in SNNs. Left**: CKA heatmaps between ANNs and SNNs with the different number of time steps. **Right**: The CKA curve of corresponding layers (diagonal values as in left).

layer after each block to investigate how these preserved representations impact task performance throughout the network. We see a monotonic increase in accuracy throughout the blocks that have residual connections, but not in blocks that do not have residual connections. This proves every residual block will contribute to the final performance.

### 4.3 Scaling up Time Steps

The results of the previous sections help characterize the effects of spatial structure on internal representation differences between SNNs and ANNs. Next, we ask whether the time dimension helps SNN learn some unique information. To verify this, we train several spiking ResNet-20 with 2/4/8/16/32/64 time steps and calculate the ANN-SNN CKA similarity. In Fig. 5, we envision the CKA heatmaps and curves respectively between artificial ResNet-20 and spiking ResNet-20 with various time steps. Notably, we cannot find significant differences among these heatmaps. Looking at the CKA curves, we also discover that many layers are overlapped, especially when we focus on the residual block output (the local maximums). We find the similarities between different time steps reaching the same point, meaning that the time step variable does not show dynamics.

To figure out why the time dimension has a limited impact on the ANN-SNN similarity, we compare the CKA among various time steps. Concretely, for any layer inside an SNN, we reshape the feature $\mathbf{X}_s$ to $[\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \dots, \mathbf{X}^{(T)}]$ where $\mathbf{X}^{(i)}$ is the $i\text{-}th$ time step's output. By computing the CKA similarity between arbitrary two time steps, *i.e.,* CKA$(\mathbf{X}^{(i)}, \mathbf{X}^{(j)})$, we are able to construct a CKA heatmap with $x, y$ axes being the time dimension, which demonstrates whether the features are

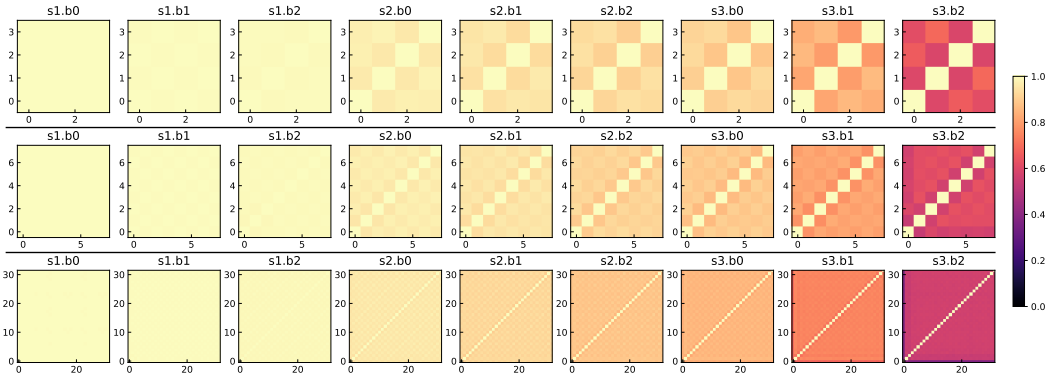


Figure 6: **The similarity across times in SNN.** Each heatmap shows the CKA among different time steps in the output of the residual block. "s" means stage, and "b" means block. The top/middle/bottom rows stand for spiking ResNet-20 with 4/8/32 time steps.

similar across different time steps. Fig. 6 illustrates such CKA heatmaps of outputs from all residual blocks in the spiking ResNet-20, with time steps varying from 4 to 32. In general, deeper residual block output exhibits darker CKA heatmaps and the shallower layers tend to become yellowish. In particular, all the residual blocks from the first stage have an all-yellow CKA heatmap, indicating extremely high similarity in these blocks. The second stage starts to produce differences across time steps, but they still share >0.8 similarities between any pair of time steps. The last stage, especially the last block, demonstrates around 0.5 similarities between different time steps. To summarize, the impact of time in SNN is gradually increased as the feature propagates through the network. In Appendix A.4, we provide the heatmap of convolutional/LIF layers and find a similar trend.

(a) Temporal Pruning.

| $T$ | $K$ | **Latency** | **Acc. (before pruning )** | **Acc. (after pruning )** |
|---|---|---|---|---|
| 4 | 1 | $\downarrow 26.8\%$ | 89.63 | 89.20 (-0.43) |
| 4 | 2 | $\downarrow 17.8\%$ | 89.63 | **89.72 (+0.09)** |
| 8 | 2 | $\downarrow 26.8\%$ | 90.81 | 90.45 (-0.36) |
| 8 | 4 | $\downarrow 17.8\%$ | 90.81 | **90.79 (-0.02)** |

(b) CKA Regularization.

| $T$ | $\lambda$ | **Acc. before reg.** | **Acc. after reg.** |
|---|---|---|---|
| 4 | $1e-4$ | 89.63 | **90.45** |
| 4 | $5e-4$ | 89.63 | 89.22 |
| 8 | $1e-4$ | 90.81 | **91.35** |
| 16 | $1e-4$ | 91.05 | **91.50** |

### 4.4 Improving SNN with CKA

Our analysis in time dimension envisions the temporal dynamics of SNNs. Here, we show that we can leverage these novel insights to improve SNN.

**Temporal Pruning.** Pruning in the time dimension has been proposed in Chowdhury et al. (2021) where the SNN is first trained with higher time steps and then fine-tuned with lower time steps. Here, we argue that we can prune a portion of the SNN to the low time step. In Fig. 6, we find that time has a limited effect on the first stage of the network. Here, we propose to infer the SNN using fewer time steps in the first stage while still preserving accuracy. Specifically, we retain only the activation of the first $K$ time steps, stacking several duplicates of them (assume $T$ is the multiplier of $K$) to $T$ time steps starting from the second stage, and then fine-tune the model for several epochs. Therefore, the overall computation is roughly reduced by $\frac{T-K}{3T}$ of the original computation. In Table 1a, we give several examples of our structured temporal pruning. It can be observed that an SNN can be pruned with 17.8% less latency without losing accuracy. If the time steps in the first stage are reduced to its $\frac{1}{4}$, the SNN will drop 0.4% accuracy with 26.8% less latency.

**CKA Regularization.** Besides improving the efficiency of SNN, we seek to improve the task performance of SNN. Here, we focus on the temporal dynamics by imposing the regularization built from CKA. Specifically, we compute the average value of the CKA matrix, i.e., the mean values of heatmaps shown in Fig. 6. Then, we force the network to learn more dynamic features across different time steps by punishing the high average CKA value, given by

$$\min_{\mathbf{W}} L_{\text{crossentropy}}(\boldsymbol{x}, \mathbf{W}, \boldsymbol{y}) + \lambda \overline{\text{CKA}}(\boldsymbol{x}, \mathbf{W}). \tag{4}$$

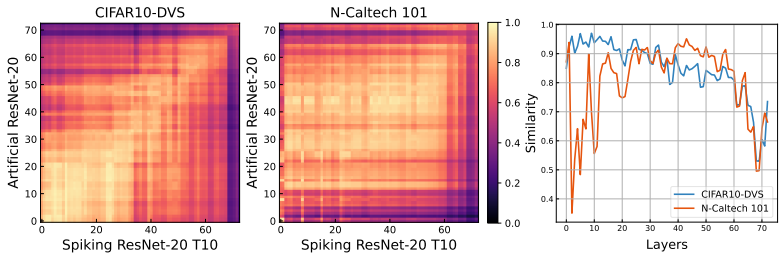


Figure 7: **CKA Similarity on Event Dataset.** We train spiking and artificial ResNet-20 on CIFAR10-DVS and N-Caltech 101, respectively. **Left**: the CKA heatmaps. **Right**: The CKA curves of corresponding layers between ANN and SNN.

We use a coefficient $\lambda$ to adjust the regularization strength. The results are summarized in Table 1b. SNNs trained with CKA regularization and an appropriate $\lambda$ can improve 0.5$\sim$0.8% accuracy, indicating that appropriate temporal dynamics benefit the task performance of SNN.

### 4.5 CKA ON EVENT-BASED DATASET

In this section, we evaluate the CKA similarity on the event-based dataset. We choose CIFAR10-DVS (Li et al., 2017), N-Caltech 101 (Orchard et al., 2015) and train spiking/artificial ResNet-20. Since the ANN cannot process 4-dimensional spatial-temporal event data easily, we integrate all events into one frame for ANN training and 10 frames for SNN training. Fig. 7 provides the CKA heatmaps/curves on the event dataset, showing a different similarity distribution than the previous CIFAR-10 dataset. The heatmaps have a different pattern and the curves also do not appear with a periodic jagged shape. In addition, the similarity distribution differs at the dataset level, i.e., the CIFAR10-DVS and N-Caltech 101 share different CKA curves and heatmaps. On N-Caltech 101, the SNN learns different feature representations compared with ANN in shallow and deep layers, but similar representation in intermediate layers. For CIFAR10-DVS, the similarity continues to decrease from 0.9 to 0.5 as the layers deepen. In summary, with the event-based dataset, SNNs and ANNs share less similarity in comparison to the natural image dataset, which implies that SNNs may have further optimization space in this type of dataset. We put more CKA results on various models for the CIFAR10-DVS dataset in Appendix A.2.

### 4.6 UNDERSTANDING ROBUSTNESS

We next study the robustness of SNN and ANN using CKA. Previous works have made explorations towards understanding the inherent robustness of SNNs (Sharmin et al., 2020; Kundu et al., 2021; Liang et al., 2021). However, they either evaluate on converted SNN or using rate encoded images. Here, we test PGD attack (Madry et al., 2017) on the directly trained SNN and ANN using direct encoding. Formally, we generate the adversarial images by restricting the $L$-infinity norm of the perturbation, given by

$$\boldsymbol{x}_{adv}^{k+1} = \Pi_{P_\epsilon(\boldsymbol{x})}(\boldsymbol{x}_{adv}^{k} + \alpha \text{sign}(\nabla_{\boldsymbol{x}_{adv}^{k}} L(\boldsymbol{x}_{adv}^{k}, \boldsymbol{w}, \boldsymbol{y}))), \tag{5}$$

where $\boldsymbol{x}_{adv}^{k}$ is the generated adversarial sample at the $k$-$th$ iteration. $\Pi_{P_\epsilon(\boldsymbol{x})}(\cdot)$ projects the generated sample onto the projection space, the $\epsilon - L_\infty$ neighborhood of the clean sample. $\alpha$ is the attack optimization step size. With higher $\epsilon$, the adversarial image is allowed to be perturbed in a larger space, thus degrading task performance.

We evaluate the spiking and artificial ResNet-20 on the CIFAR-10 dataset with $\epsilon$ from $\{0.001, 0.005, 0.01, 0.02, 0.05\}$. After generating the adversarial images, we measure the CKA value between the clean image's features and the adversarial corrupted image's features. We first test the accuracy after the adversarial attack and summarize the result in Fig. 8 left. The clean accuracy of ANN is higher than that of SNN, but SNN has higher robustness against adversarial attacks. As an example, the PGD attack with 0.01 $L$-infinity norm perturbation reduces 43% accuracy of ANN, but only reduces 22% accuracy of SNN. We also investigate the CKA similarity as shown in the 2nd and 3rd subplots of Fig. 8. Intuitively, the higher the robustness against adversarial attacks, the higher the similarity between clean and corrupted images will be. The CKA curves indeed testify to this intuition, the SNN has a higher similarity than ANN does. We also observe several interesting phenomena. For instance, the ANN suffers from a large decrease in the similarity of the first block,

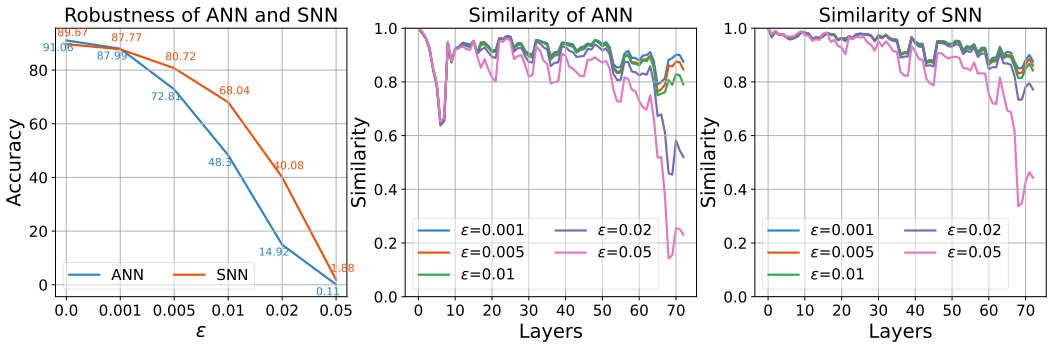


Figure 8: **The robustness against adversarial attack. Left**: The accuracy of SNN and ANN after attack under different $\epsilon$. **Right**: The CKA curve between clean images and adversarial images of ANN and SNN, respectively.

i.e., the first block has below 0.7 similarities even with the $\epsilon = 0.001$ attack. Also, if we focus on the purple line ($\epsilon = 0.02$), we notice ANN and SNN have similar perseverance in earlier layers, but ANN drops much more similarity than SNN in the last block. These results provide interpretation to model robustness and suggest SNNs have better robustness than ANN, especially in their shallow and deep layers.

## 5 Discussion and Conclusion

Given the fact that SNNs are drawing increasing research attention due to their bio-plausibility, it is necessary to verify if SNNs can or have the potential to truly develop desired features different from ANNs. In this work, we conduct a pilot study to examine the internal representation of SNNs and compare it with ANNs. At first, an analysis of scaling up the spatial dimension of both ANNs and SNNs finds depth discourages similarity, while width encourages similarity. Next, we pinpoint the role of the residual connection in SNNs—carrying the information from input and somehow making SNNs' representation like ANNs. We also investigate the unique time dimension of SNNs using both inter-network CKA and intra-network CKA.

Unfortunately, our results may not fully support the opinion that SNNs learn highly effective and distinct temporal representation compared to ANNs. Current SNN learning relies on the residual connection to obtain decent task performance. However, our study suggests that this task performance is highly credited to the similar representation with ANN brought by residual connections. Furthermore, the time dimension brings limited effect to the SNN representation on static datasets like CIFAR10 and CIFAR100. In particular, the first stage of ResNets results in quite similar across time steps. Nonetheless, we should not be pessimistic about SNN. In Sec. 4.4, we discuss potential optimizations of current SNN using temporal pruning and CKA regularization to improve efficiency and accuracy. In Sec. 4.6, we notice better robustness in SNN against adversarial attacks. These results suggest the probabilities to further optimize SNN and develop more effective representations or features. Here, we provide several directions worth studying in the future: a) Optimization of deeper SNNs with better learning algorithms to avoid the usage of residual connections: Deep SNNs evolve more distinct features but also becomes harder to be trained. Thus, a learning rule beyond the gradient-based method could be promising. b) Exploring other neuron models in SNN: Current LIF neurons seem to bring limited effect in the time dimension. A neuron model that can bring effective temporal dynamics may significantly improve SNN performance. c) Understanding the robustness of SNN: Adversarial attack is inconsequential for human visual systems, which may be reflected in SNN as well. We believe the SNN robustness can be significantly improved and our study on representation of SNN provides a good example.

Our analysis also has limitations. The CKA we used summarizes measurements into a single scalar. To provide quantitative insights on representation similarity, more fine-grained methods may reveal additional insights and variations in the representations.

## REFERENCES


Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE transactions on computer-aided design of integrated circuits and systems*, 34(10):1537–1557, 2015.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Anthony N Burkitt. A review of the integrate-and-fire neuron model: I. homogeneous synaptic input. *Biological cybernetics*, 95(1):1–19, 2006.

Sayeed Shafayet Chowdhury, Isha Garg, and Kaushik Roy. Spatio-temporal pruning and quantization for low-latency spiking neural networks. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9. IEEE, 2021.

Dennis Valbjørn Christensen, Regina Dittmann, Bernabé Linares-Barranco, Abu Sebastian, Manuel Le Gallo, Andrea Redaelli, Stefan Slesazeck, Thomas Mikolajick, Sabina Spiga, Stephan Menzel, et al. 2022 roadmap on neuromorphic computing and engineering. *Neuromorphic Computing and Engineering*, 2022.

Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

Lei Deng, Yujie Wu, Xing Hu, Ling Liang, Yufei Ding, Guoqi Li, Guangshe Zhao, Peng Li, and Yuan Xie. Rethinking the performance comparison between snns and anns. *Neural Networks*, 121:294 – 307, 2020.

Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. *arXiv preprint arXiv:2103.00476*, 2021.

Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2661–2671, 2021.

Samanwoy Ghosh-Dastidar and Hojjat Adeli. Spiking neural networks. *International journal of neural systems*, 19(04):295–308, 2009.

Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pp. 63–77. Springer, 2005.

Bing Han and Kaushik Roy. Deep spiking neural network: Energy efficiency through time based coding. In *Proc. IEEE Eur. Conf. Comput. Vis.(ECCV)*, pp. 388–404, 2020.

Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13558–13567, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016b.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Seyed-Mahdi Khaligh-Razavi and Nikolaus Kriegeskorte. Deep supervised, but not unsupervised, models may explain it cortical representation. *PLoS computational biology*, 10(11):e1003915, 2014.

Youngeun Kim and Priyadarshini Panda. Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. *Frontiers in neuroscience*, pp. 1638.

Youngeun Kim, Yuhang Li, Hyoungseob Park, Yeshwanth Venkatesha, and Priyadarshini Panda. Neural architecture search for spiking neural networks. *arXiv preprint arXiv:2201.10355*, 2022.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pp. 3519–3529. PMLR, 2019.

Nikolaus Kriegeskorte, Marieke Mur, and Peter A Bandettini. Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2:4, 2008.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Souvik Kundu, Massoud Pedram, and Peter A Beerel. Hire-snn: Harnessing the inherent robustness of energy-efficient deep spiking neural networks by training with crafted input noise. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5209–5218, 2021.

Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11:309, 2017.

Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In *International Conference on Machine Learning*, pp. 6316–6325. PMLR, 2021.

Ling Liang, Xing Hu, Lei Deng, Yujie Wu, Guoqi Li, Yufei Ding, Peng Li, and Yuan Xie. Exploring adversarial attack in spiking neural networks with spike-compatible gradient. *IEEE transactions on neural networks and learning systems*, 2021.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. *Advances in Neural Information Processing Systems*, 31, 2018.

Byunggook Na, Jisoo Mok, Seongsik Park, Dongjin Lee, Hyeokjun Choe, and Sungroh Yoon. Autosnn: Towards energy-efficient spiking neural networks. *arXiv preprint arXiv:2201.12738*, 2022.

Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. *arXiv preprint arXiv:2010.15327*, 2020.

Garrick Orchard, Ajinkya Jayawant, Gregory K Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience*, 9:437, 2015.

Priyadarshini Panda, Sai Aparna Aketi, and Kaushik Roy. Toward scalable, efficient, and accurate deep spiking neural networks with backward residual connections, stochastic softmax, and hybridization. *Frontiers in Neuroscience*, 14:653, 2020.

Jing Pei, Lei Deng, Sen Song, Mingguo Zhao, Youhui Zhang, Shuang Wu, Guanrui Wang, Zhe Zou, Zhenzhi Wu, Wei He, et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767):106–111, 2019.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Advances in neural information processing systems*, 30, 2017.

Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34, 2021.

Nitin Rathi and Kaushik Roy. Diet-snn: Direct input encoding with leakage and threshold optimization in deep spiking neural networks. *arXiv preprint arXiv:2008.03658*, 2020.

Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, and Michael Pfeiffer. Theory and tools for the conversion of analog to spiking convolutional neural networks. *arXiv preprint arXiv:1612.04052*, 2016.

Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.

Statistics Student Seminar. Convergent learning: Do different neural networks learn the same representations? 2016.

Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience*, 13:95, 2019.

Saima Sharmin, Nitin Rathi, Priyadarshini Panda, and Kaushik Roy. Inherent adversarial robustness of deep spiking neural networks: Effects of discrete input encoding and non-linear activations. In *European Conference on Computer Vision*, pp. 399–414. Springer, 2020.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Le Song, Alex Smola, Arthur Gretton, Justin Bedo, and Karsten Borgwardt. Feature selection via dependence maximization. *Journal of Machine Learning Research*, 13(5), 2012.

Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural networks*, 111:47–63, 2019.

Liwei Wang, Lunjia Hu, Jiayuan Gu, Zhiqiang Hu, Yue Wu, Kun He, and John Hopcroft. Towards understanding learning representations: To what extent do different neural networks learn the same representation. *Advances in neural information processing systems*, 31, 2018.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 1311–1318, 2019.

Daniel LK Yamins, Ha Hong, Charles F Cadieu, Ethan A Solomon, Darren Seibert, and James J DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the national academy of sciences*, 111(23):8619–8624, 2014.

Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. *arXiv preprint arXiv:2011.05280*, 2020.

# A ADDITIONAL CKA RESULTS

## A.1 RESULTS ON VGG NETWORKS

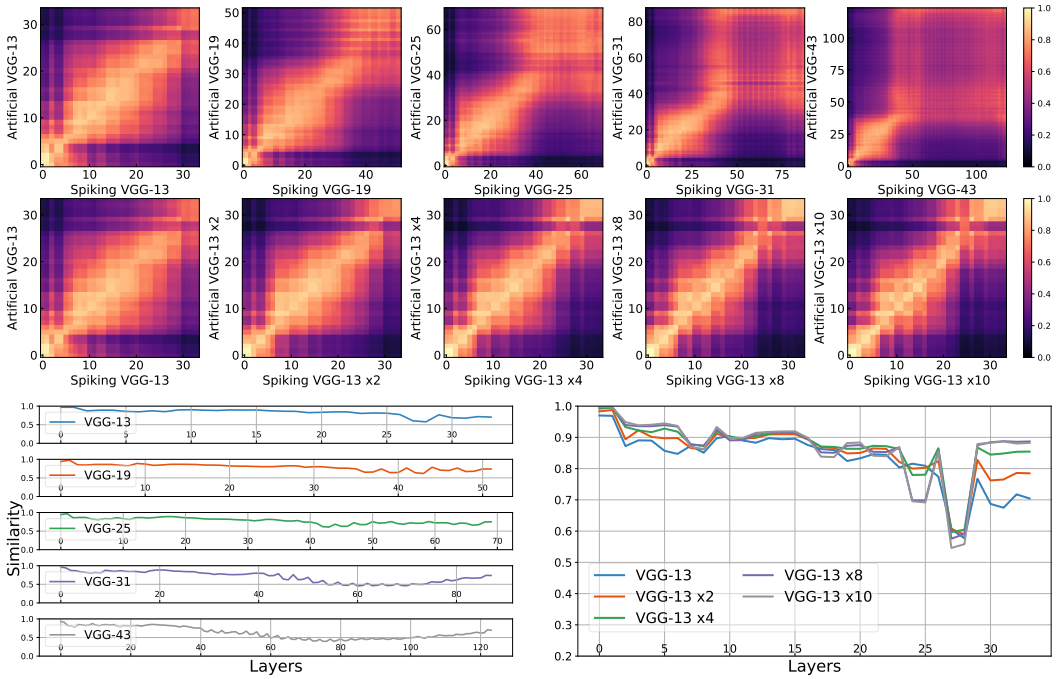


Figure A.1: **CKA heatmap between spiking VGGs and artificial VGGs with different depth and width on CIFAR-10 dataset. Top**: the CKA cross-layer heatmap across different depth from 13 layers to 43 layers. **Middle**: the CKA cross-layer heatmap across different width from original channel number to 8 times. **Bottom**: visualizing only the corresponding layer, which is the diagonal of the CKA heatmap.

### A.1.1 SCALING UP WIDTH OR DEPTH

In this section, we study the representation similarity between ANNs and SNNs based on VGG-series networks Simonyan & Zisserman (2015). Since VGG-Networks do not employ residual connections, they may bring different representation heatmaps when compared to ResNets. We first evaluate if the spatial scaling observation in ResNets can also be found in VGG Networks. Starting from a VGG-13, we either increase its channel size to 10 times as before or its number of layers to 43, (detailed network configuration is provided in Table C.2). Since VGG networks do not contain residual connections, we could scale less depth in VGG networks than ResNets.

The results are illustrated in Fig. A.1. We can find that for deeper networks, the heatmaps tend to exhibit a hierarchical structure, which means the shallow layers and deeper layers have different representations. Increasing the number of layers in VGG networks to 19 or 25, the shallower and deeper layers only have 0.3 CKA similarity (purple). More seriously, when the network depth reaches 31 or 43, the similarity becomes 0.4 even across each other in deeper layers, indicating the representations are diverging. Another important discovery for deep VGG networks is the disappearance of periodical CKA curve, which is likely due to the lack of residual connections.

As for the wider networks, the observations are consistent with ResNet families. The wider networks have both brighter heatmaps and higher CKA curves. These results confirm the similar representation in extremely wide networks.

### A.1.2 SCALING UP TIME STEPS

We next study whether the time dimension in spiking VGG networks has a similar effect to that in spiking ResNets. As can be found in Fig. A.2, the spiking VGG-13s with 4/8/16/32 time steps do

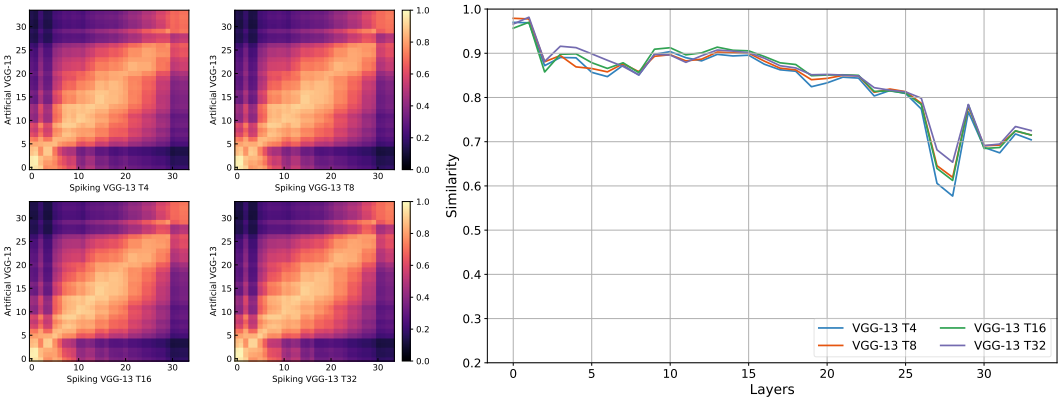


Figure A.2: **The effect of time steps in spiking VGG networks. Left**: CKA heatmaps between ANNs and SNNs with the different number of time steps (from 4 to 32). **Right**: The CKA curve of corresponding layers (diagonal values as in left).

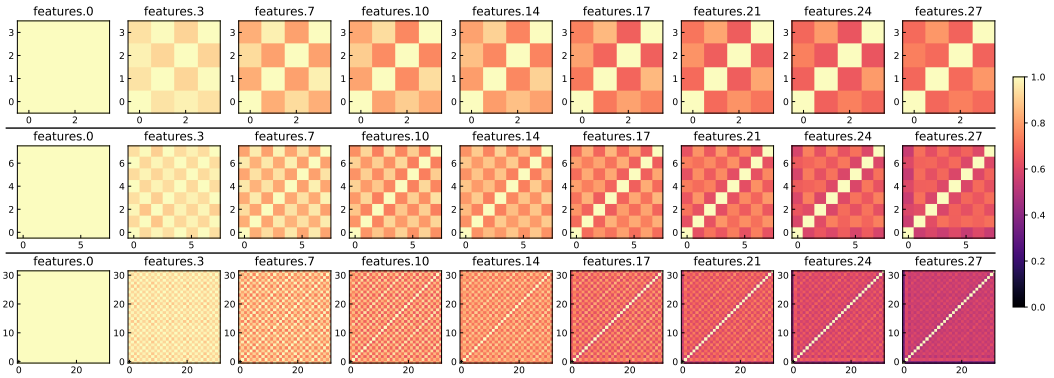


Figure A.3: **The similarity across times in Spiking VGG-13.** Each heatmap shows the CKA among different time steps in the output of convolutional layers. The top/middle/bottom rows stand for spiking ResNet-20 with 4/8/32 time steps.

not show a significant difference in CKA heatmaps as well as CKA curves, which is the same with Fig. 5. Fig. A.3 shows the CKA across different time steps in the spiking VGGs. In the first layer, the convolution does not have dynamic representation through time. As the layer goes deeper, the dissimilarity across time steps continues to increase, similar to Fig. 6.

### A.2 Results on CIFAR10-DVS

In this section, we conduct a representation similarity analysis on an event-based dataset—CIFAR10-DVS Li et al. (2017). As aforementioned in Fig. 7, the event dataset may generate different CKA heatmaps and curves when compared to the static dataset. Here, we first scale up the spatial dimensions in SNNs and ANNs for the CIFAR10-DVS dataset. As shown in Fig. A.4, we gradually increase either the depth to 110 layers or the width to 8 times as before.

Increasing the depth of ResNets on CIFAR10-DVS demonstrates a similar effect when compared to static CIFAR-10. Interestingly, the deep ResNet, for example, ResNet-110, emerges a similar periodical pattern to that on static CIFAR-10. However, we can find the peak CKA value is only around 0.75, (recall that the peak CKA value on the CIFAR10 dataset is nearly 0.95, cf. Fig. 2), indicating the CIFAR10-DVS creates higher ANN-SNN difference in representations.

For the extremely wide neural network, the trend holds for ResNet-20, ResNet-20 ($\times 2$), and ResNet-20 ($\times 4$). However, we find the ResNet-20 ($\times 8$) exhibits a significantly low CKA curve than other models. The difference may come from the artificial ResNet-20 ($\times 8$). According to the CKA heatmap, the 35th$-$60th layers in artificial ResNet-20 ($\times 8$) become much darker than other

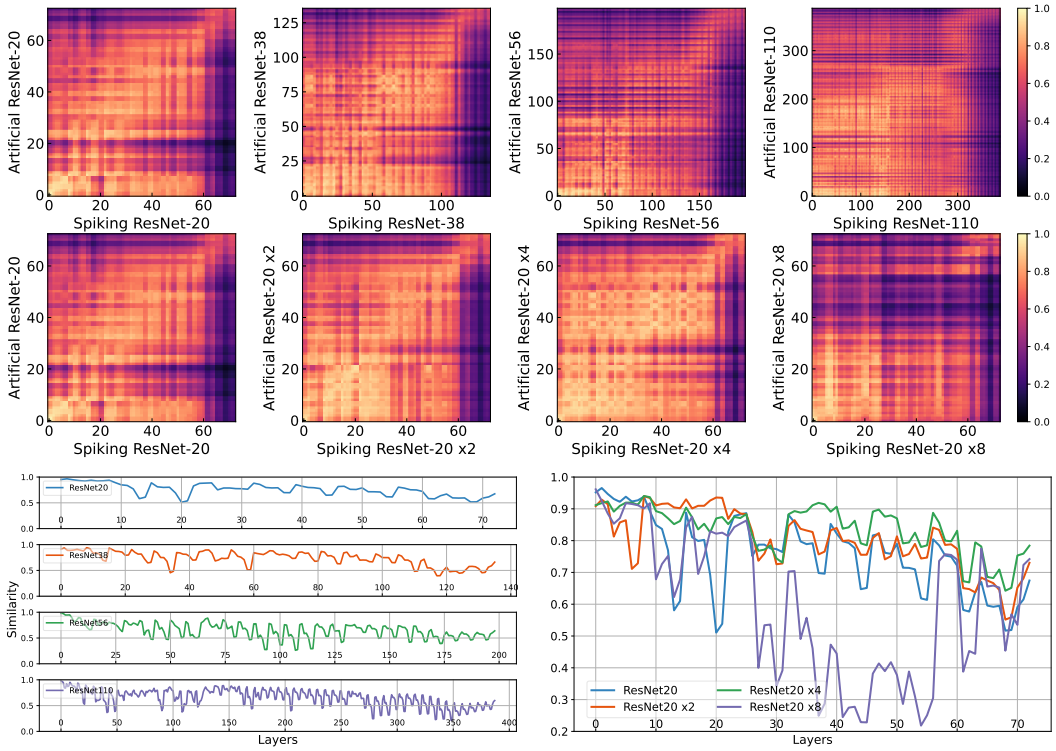


Figure A.4: **CKA heatmap between spiking and artificial ResNets with different depth and width on CIFAR10-DVS dataset. Top**: the CKA cross-layer heatmap across different depths from 20 layers to 110 layers. **Middle**: the CKA cross-layer heatmap across different widths from the original channel number to 8 times. **Bottom**: visualizing only the corresponding layer, which is the diagonal of the CKA heatmap.

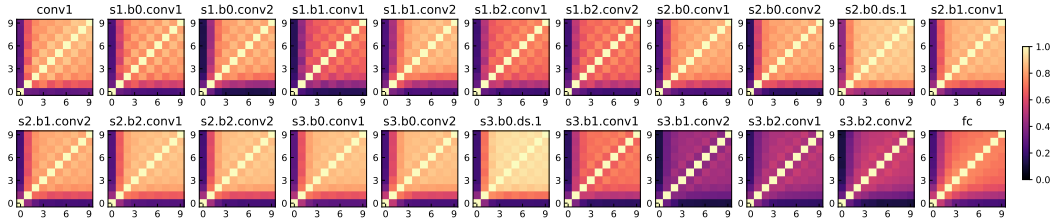


Figure A.5: **The similarity across times in SNN on CIFAR10-DVS.** Each heatmap shows the CKA among different time steps in the output of residual block. "s" means stage, and "b" means block. The model is trained with 10 time steps, i.e., 10 frames integrated for each CIFAR10-DVS data input.

heatmaps. We hypothesize that, with a larger capacity of the neural network on the layer dynamics in the input data, the representation may change significantly.

We next study the internal CKA of each convolution in ResNet-20, which reveals the temporal dynamics of each convolution. As illustrated in Fig. A.5, the first convolution demonstrates dynamic features across 10 time steps, which shows different characteristics with static dataset (cf. Fig. A.12). Another notable difference is that the first and the second time step show very low similarity with other time steps. In summary, the rich temporal information dataset may increase the dynamics in SNNs across time steps and bring more differences when compared to ANNs.

### A.3 RESULTS ON CIFAR-100

The results we reported in the main context are majorly based on the CIFAR-10 dataset. Here, we provide the visualizations on the CIFAR-100 dataset to further strengthen our findings in the main context.

We first report the spatial dimension results, i.e., scaling up the width and depth of the network. Starting from the ResNet-20, we either increase its width to 164 layers or increase its width to 8 times as before. The visualizations are shown below. We find extremely deep networks, e.g., ResNet-164, has a very dark heatmap compared to the ResNet-20 heatmap. The wider network shows somewhat irregular results. The extremely wide network — ResNet-20×8 — yet has even lowest similarity in the last several blocks. However, it has the highest similarity in the output layer.

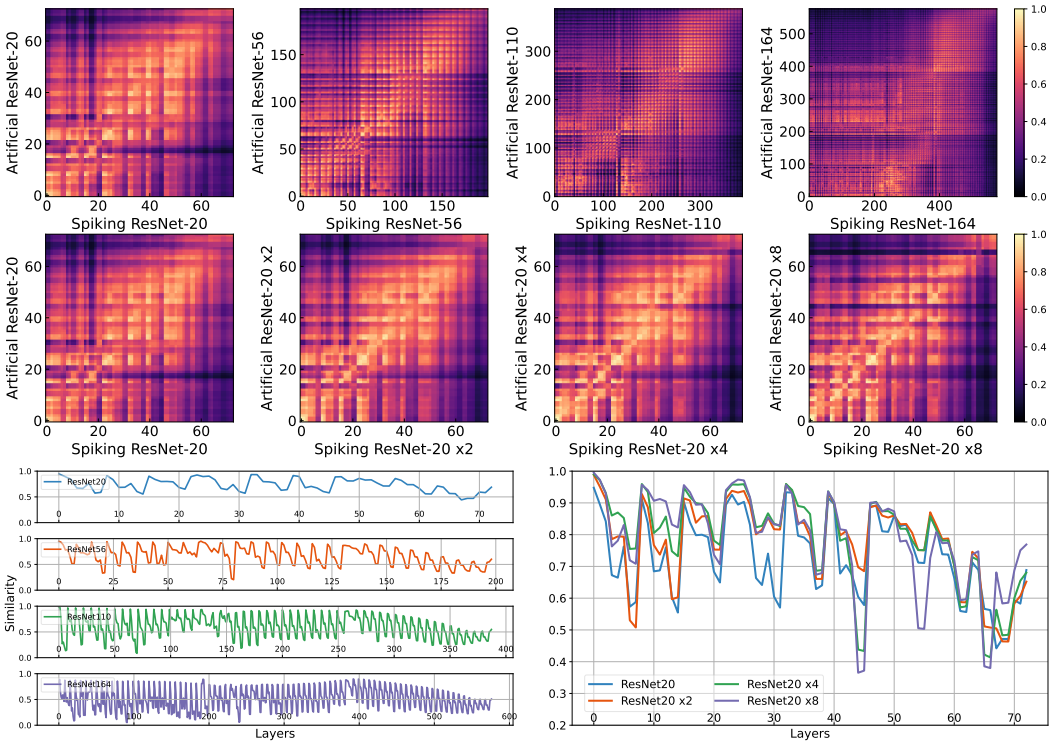


Figure A.6: **CKA heatmap between SNNs and ANNs with different depth and width on CIFAR-100. Top**: the CKA cross-layer heatmap across different depth from 20 layers to 164 layers. **Middle**: the CKA cross-layer heatmap across different width from original channel number to 16 times. **Bottom**: visualizing only the corresponding layer, which is the diagonal of the CKA heatmap.

Next, we visualize the details inside a residual block. In Fig. A.7, we sub-sample the 10-$th$ and the 34-$th$ residual block in a ResNet-110, which shows the same phenomenon. The LIF and convolutional layers cause a decrease in similarity, while the residual addition operation restores the similarity. We also provide the CKA heatmap of the partial residual network. As done in Fig. 4, we train 3 spiking ResNet-56 on the CIFAR-100 dataset with several blocks disabling the residual connections. Moreover, we train a linear probing layer — the fully-connected classifier on top of each block to see if it contributes to the overall performance of the whole network. The visualization is shown in Fig. A.8, where we find similar observations.

We also run experiments to test the time dimension of SNNs on CIFAR-100. In Fig. A.9, we train 4 spiking ResNet-20 with 4/8/16/32 time steps and compute their representations with artificial ResNet-20. Both the CKA heatmap and the CKA curve show little variations by changing the number of time steps. This confirms the results on CIFAR-10. In addition, we visualize the CKA similarities across time steps in each residual block. Fig. A.10 demonstrates that the first stage still produces temporal static features while the last stage has lower similarity across time steps.

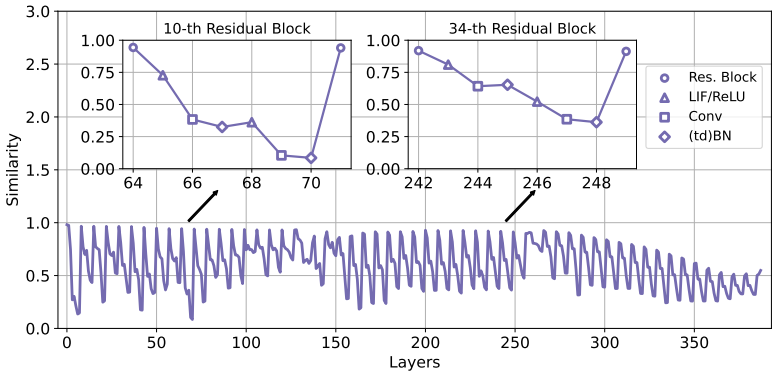


Figure A.7: **Emergence of periodic jagged CKA curve on CIFAR-100.** We subplot the 10-$th$ and the 34-$th$ residual blocks in ResNet-110, which forms a periodic jagged curve.

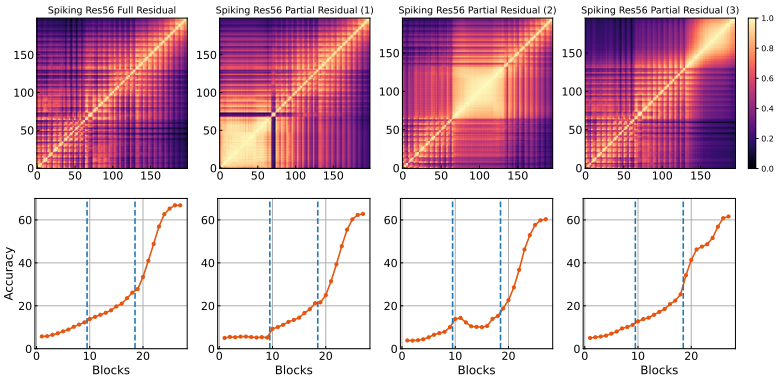


Figure A.8: **The effect of residual connections in the SNN.** We selectively disable residual connections in one of three stages in the ResNet-56. **Top**: the CKA heatmap of *SNN itself*, containing networks with different types of non-residual blocks. **Bottom**: The linear probing accuracy of each block.

Finally, we rerun the adversarial robustness experiments on CIFAR-100. Here, we train two ResNet-20 with 4 times more channels and use PGD attack to measure the robustness against adversarial attack. Also, we plot the CKA curve between the feature of clean images and the feature of corrupted images. The results are shown in Fig. A.11.

### A.4 Similarity Across Time

In addition to the residual block, we also visualize the CKA heatmaps of convolutional layers and ReLU/LIF layers by comparing the similarity among different time steps. As can be seen in Fig. A.12, different from residual blocks, the similarity in convolutional and activation layers is more dynamic. Even in the first stage, the convolutional layers show different outputs across different time steps. This result further confirms our observations in residual blocks, where we found the convolutional and activation layers always decrease the ANN-SNN similarity while the residual block restores the similarity. Therefore, it suggests that a more temporal dynamic CKA heatmap may produce distinct features.

## B Numerical Results

Here, we provide the clean accuracy of our trained models, both on CIFAR-10 and CIFAR-100. All models are trained with 300 epochs of stochastic gradient descent. The learning rate is set to 0.1 followed by a cosine annealing decay. The weight decay is set to 0.0001 for all models. The original ResNet-20 is a 3-stage model, each stage contains 2 residual blocks. The first stage contains 16 channels and the channels are doubled every time when entering the next stage. ResNet-38/56/110/164 contains 6/9/18/27 residual blocks in each stage. The wider networks just simply multiply all the channels by a fixed factor. We provide their top-1 accuracy in **??**.

## C Network Architecture Details

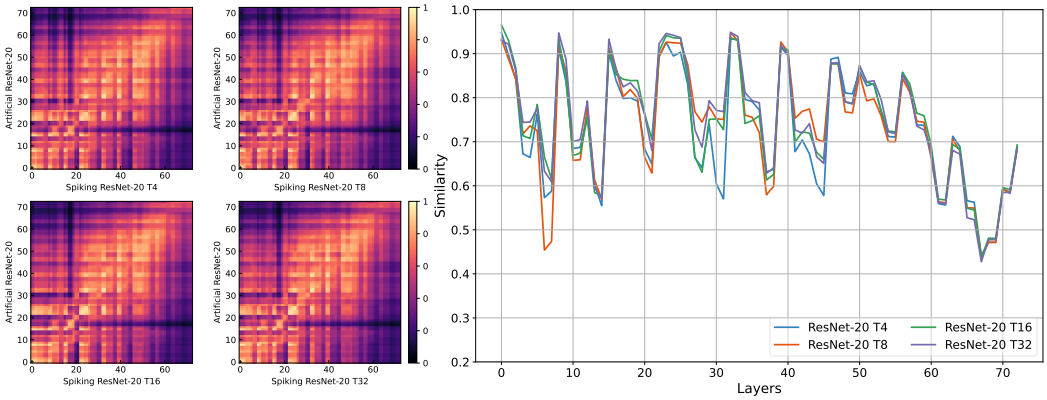


Figure A.9: **The effect of time steps in SNNs. Left**: CKA heatmaps between ANNs and SNNs with the different number of time steps. **Right**: The CKA curve of corresponding layers (diagonal values as in left).

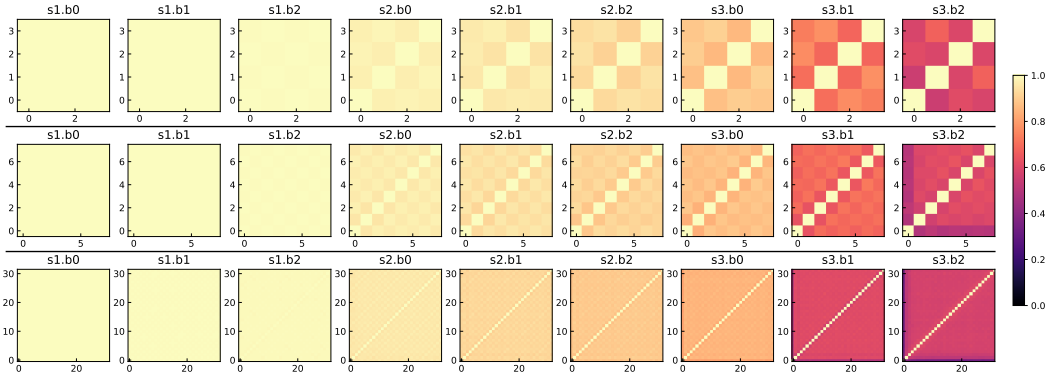


Figure A.10: **The similarity across times in SNN on CIFAR-100.** Each heatmap shows the CKA among different time steps in the output of residual block. "s" means stage, and "b" means block. The top/middle/bottom rows stand for spiking ResNet-20 with 4/8/32 time steps.

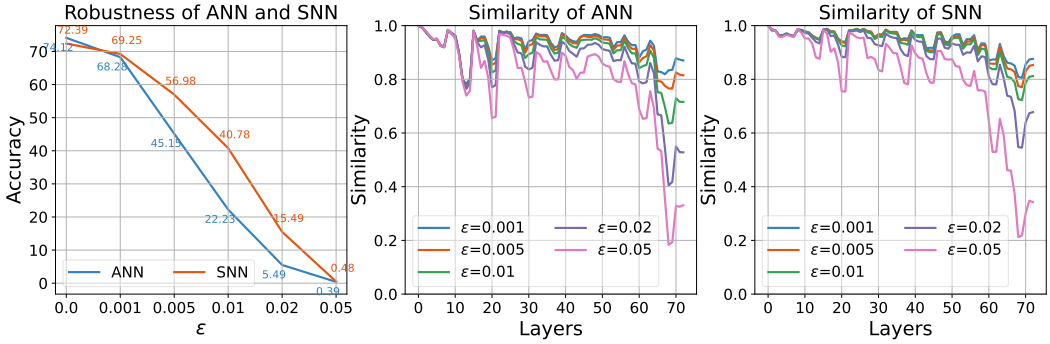


Figure A.11: **The robustness against adversarial attack on CIFAR-100. Left**: The accuracy of SNN and ANN after attack under different $\epsilon$. **Right**: The CKA curve between clean images and adversarial images of ANN and SNN, respectively.

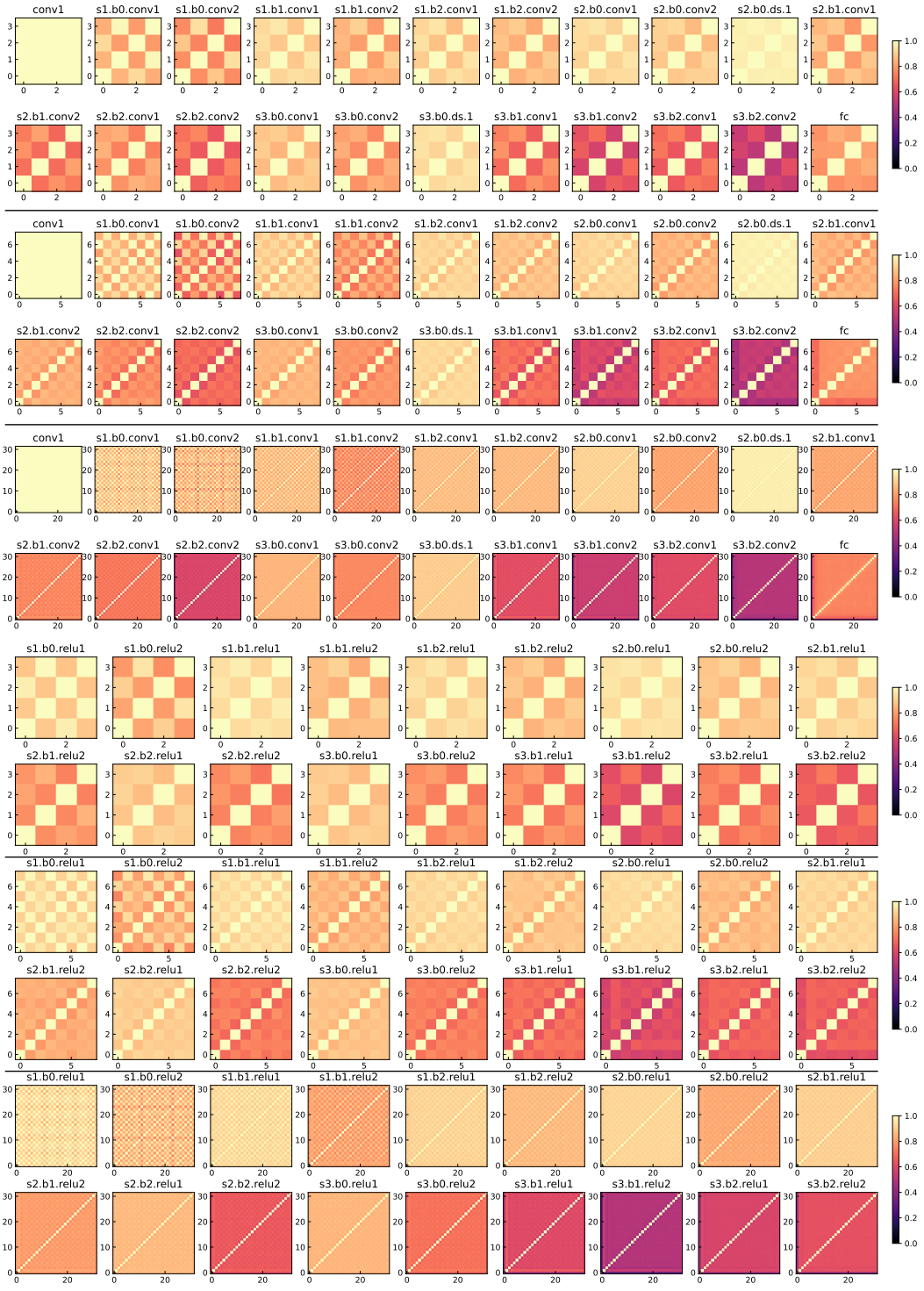


Figure A.12: **The effect of time steps in convolutional and activation layers of SNNs.**

Table B.1: The top-1 accuracy of SNNs and ANNs on CIFAR-10 (aka C10) and CIFAR-100 (aka C100) datasets.

| Layers | Width Factor | Time Steps | ANN (C10) | SNN (C10) | ANN (C100) | SNN (C100) |
|---|---|---|---|---|---|---|
| 20 | 1 | 4 | 91.06 | 89.63 | 64.23 | 61.51 |
| 38 | 1 | 4 | 92.34 | 91.14 | N/A | N/A |
| 56 | 1 | 4 | 92.98 | 91.94 | 68.39 | 66.63 |
| 110 | 1 | 4 | 92.37 | 91.83 | 69.20 | 66.95 |
| 164 | 1 | 4 | 93.00 | 92.05 | 70.27 | 67.09 |
| 20 | 2 | 4 | 93.36 | 92.00 | 70.14 | 68.65 |
| 20 | 4 | 4 | 94.52 | 93.96 | 74.12 | 72.39 |
| 20 | 8 | 4 | 94.78 | 94.48 | 76.57 | 75.81 |
| 20 | 16 | 4 | 94.78 | 94.73 | N/A | N/A |
| 20 | 1 | 8 | 91.06 | 90.64 | 64.23 | 63.03 |
| 20 | 1 | 16 | | 91.05 | | 64.34 |
| 20 | 1 | 32 | | 91.15 | | 64.33 |
| 20 | 1 | 64 | | 91.18 | | N/A |

Table C.1: The architecture details of ResNets.

| | 20 layers | 38 layers | 56 layers | 110 layers | 164 layers |
|---|---|---|---|---|---|
| conv1 | $3\times3, 16, s1$ | $3\times3, 16, s1$ | $3\times3, 16, s1$ | $3\times3, 16, s1$ | $3\times3, 16, s1$ |
| block1 | $\begin{pmatrix}3\times3,16\\3\times3,16\end{pmatrix}\times3$ | $\begin{pmatrix}3\times3,16\\3\times3,16\end{pmatrix}\times6$ | $\begin{pmatrix}3\times3,16\\3\times3,16\end{pmatrix}\times9$ | $\begin{pmatrix}3\times3,16\\3\times3,16\end{pmatrix}\times18$ | $\begin{pmatrix}3\times3,16\\3\times3,16\end{pmatrix}\times27$ |
| block2 | $\begin{pmatrix}3\times3,32\\3\times3,32\end{pmatrix}\times3$ | $\begin{pmatrix}3\times3,32\\3\times3,32\end{pmatrix}\times6$ | $\begin{pmatrix}3\times3,32\\3\times3,32\end{pmatrix}\times9$ | $\begin{pmatrix}3\times3,32\\3\times3,32\end{pmatrix}\times18$ | $\begin{pmatrix}3\times3,32\\3\times3,32\end{pmatrix}\times27$ |
| block3 | $\begin{pmatrix}3\times3,64\\3\times3,64\end{pmatrix}\times3$ | $\begin{pmatrix}3\times3,64\\3\times3,64\end{pmatrix}\times6$ | $\begin{pmatrix}3\times3,64\\3\times3,64\end{pmatrix}\times9$ | $\begin{pmatrix}3\times3,64\\3\times3,64\end{pmatrix}\times18$ | $\begin{pmatrix}3\times3,64\\3\times3,64\end{pmatrix}\times27$ |
| pooling | Global average pooling | | | | |
| classifier | 10-d fully connected layer, softmax | | | | |

Table C.2: The architecture details of VGG networks.

| | 13 layers | 19 layers | 25 layers | 31 layers | 43 layers |
|---|---|---|---|---|---|
| block1 | $(3\times3,64)\times1$ | $(3\times3,64)\times2$ | $(3\times3,64)\times2$ | $(3\times3,64)\times2$ | $(3\times3,64)\times2$ |
| pooling1 | Average pooling, $s2$ | | | | |
| block2 | $(3\times3,128)\times1$ | $(3\times3,128)\times2$ | $(3\times3,128)\times3$ | $(3\times3,128)\times4$ | $(3\times3,128)\times5$ |
| pooling2 | Average pooling, $s2$ | | | | |
| block3 | $(3\times3,256)\times2$ | $(3\times3,256)\times3$ | $(3\times3,256)\times4$ | $(3\times3,256)\times5$ | $(3\times3,256)\times6$ |
| pooling3 | Average pooling, $s2$ | | | | |
| block4 | $(3\times3,512)\times4$ | $(3\times3,512)\times8$ | $(3\times3,512)\times12$ | $(3\times3,512)\times16$ | $(3\times3,512)\times24$ |
| pooling | Global average pooling | | | | |
| classifier | 10-d fully connected layer, softmax | | | | |