Self-Supervised Contrastive Learning In Spiking Neural Networks

Yeganeh Bahari and Saeed Reza Kheradpisheh*
Departement of Computer and Data Sciences, Faculty of Mathematical Sciences, Shahid Beheshti University, Tehran, Iran
Emails: bahary.yeganeh@gmail.com, s_kheradpisheh@sbu.ac.ir
* Corresponding author

***Abstract*—Spiking neural networks (SNNs), inspired by the biological neural processing of the brain, are vastly growing due to their higher potential to handle spatiotemporal patterns with lower energy consumption, especially, if implemented on neuromorphic devices. In this study, we propose self-supervised contrastive learning (SSL) for SNNs to learn informative latent representations from a large set of unlabeled data. The proposed SSL pre-trained SNN is then fine-tuned on a small set of labeled samples of a downstream supervised task. To evaluate the proposed method, we trained convolutional SNNs using SSL on MNIST and CIFAR10 datasets with 80% of images as unlabeled samples, then fine-tuned the networks on the remaining 20% images. The proposed SSL-based SNNs could reach 94.23% and 62.24% recognition accuracies on testing sets of MNIST and CIFAR10, respectively.**

***Index Terms*—Spiking Neural Networks, Self-Supervised Learning, SimCLR, Data Augmentation.**

I. Introduction

Spiking Neural Networks (SNNs) hold promise for efficient neural computation, particularly on neuromorphic processors, owing to their reduced energy costs compared to traditional neural networks [1]. The energy efficiency of SNNs arises from the discrete nature of spiking activations sparsely distributed in time [2]. However, supervised learning in SNNs is faced with difficulties, including the non-differentiability of spike functions due to their discrete property, precluding the use of backpropagation in the learning process [3].

Different solutions are proposed to overcome this challenge, including ANN-to-SNN conversion [4], using ANNs as a proxy [5], and the use of surrogate gradients [6]. In our research, we address these challenges by employing surrogate gradients, a technique commonly applied in supervised SNN learning. In this method, the derivation of discrete spike activation with the derivation of an approximating continuous activation function [7]. We extend this methodology to a self-supervised approach, introducing a novel paradigm in SNN learning. [8].

Self-supervised learning (SSL) is a representation learning approach that is solely based on unlabeled data. Technically, the aim of SSL is to learn an informative latent representation from a large set of unlabeled data in a self-defined supervised task, which then, can be used as a pre-training in a downstream real supervised task with few labeled data. Contrastive learning is one of the main approaches in SSL, where, the learner is encouraged to learn similar (dissimilar) representations for two augmentations of the same (different) inputs.

Here, we proposed a self-supervised contrastive learning for spiking networks with surrogate gradients in the backward pass. The proposed method reaches comparable results on two SNN benchmarks including MNIST and CIFAR10 datasets. Also, the latent representations learned during the SSL training phase are assessed by a KNN-based method [8].

II. Self-Supervised Learning

In recent studies [9], Self-Supervised Learning (SSL) for image processing has shown remarkable progress, offering models capable of learning representations without relying on labeled data. These models typically use two-branch networks for processing augmented images, followed by applying a loss function to refine the learned representations [10]. One standout SSL model is SimCLR (Simple Contrastive Learning) [11], the main architecture of SimCLR is shown in Fig 1. Self-supervised learning (SSL) for image processing, particularly in models like SimCLR, involves a multi-step process. It begins with data augmentation, where the dataset is expanded through various transformations of the images. The augmented data then enters the model's encoder, responsible for extracting meaningful representations from the images.

Following the encoder, the learned representations pass through a projection head, typically consisting of Multi-Layer Perceptron (MLP) layers. The projection head further refines the representations, enhancing their quality for downstream tasks [12]. Finally, the refined representations are fed into a contrastive loss function. This loss function maximizes the agreement between augmented views of the same image while minimizing the agreement with views from other images, fostering effective feature learning without the need for explicit labels.

A. Data Augmentation

Data augmentation significantly boosts the effectiveness of self-supervised learning (SSL) models. By applying diverse transformations like rotations and flips to input data, SSL models can learn more robust and generalized representations. This approach generates varied views of the same data, helping the model capture essential features of data. In SSL, two versions of augmented data are fed into the encoder simultaneously,

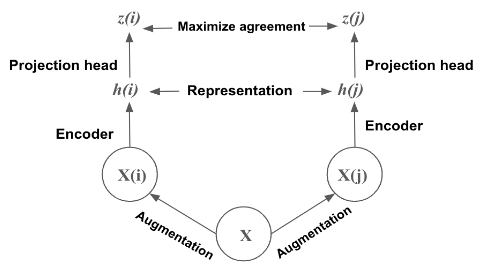

Fig. 1. SimCLR Architecture. Two augmented images are independently fed to the encoder and the goal is to maximize (minimize) the representational similarity between same (different) images.

enhancing the model's ability to learn and extract meaningful representations [13].

B. Loss function

The Normalized Temperature-scaled Cross-Entropy (NT-Xent) loss is a commonly used loss function in self-supervised learning. It is designed to measure the similarity between positive pairs (augmented views of the same instance) and push them closer in the feature space, while simultaneously pushing negative pairs (views from different instances) apart. The temperature parameter in NT-Xent loss controls the smoothness of the probability distribution and influences the scale of the similarity scores [12]. The equation of the NT-Xent loss function is in the following form :

$$\mathcal{L}_{\text{NTXent}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(\text{sim}(z_i, z_{\text{pos}})/\tau)}{\sum_{j=1}^{2N}\exp(\text{sim}(z_i, z_j)/\tau)} \quad (1)$$

where N is the number of samples in the batch, z_i is the augmented representation of X_i, $z_{i_{\text{pos}}}$ is another augmented representation of X_i (where $j \neq i$), sim is the similarity function between representations, and τ is the temperature parameter for scaling.

C. Transfer Learning

Transfer learning is a paradigm that leverages knowledge gained from pre-training on one task to improve performance on a different but related task. In the context of self-supervised learning, models pre-trained on large datasets using SSL techniques can be fine-tuned on downstream tasks with limited labeled data. This transfer of learned representations often leads to improved performance, efficiency, and faster convergence in specific applications, making it a valuable aspect of the self-supervised learning framework [14].

III. Spiking Neural Network

Spiking Neural Networks draw inspiration from the biological structure and functioning of neurons in the brain [15]. The fundamental building block of SNNs is the *Leaky Integrate-and-Fire* (LIF) neuron model, which closely emulates the behavior of biological neurons [16].

In the context of SNNs, the LIF neuron model reflects the concept of information processing through the generation of spikes or action potentials. Similar to biological neurons [17], the LIF neuron accumulates input signals over time, simulating a form of integration. However, unlike a perfect integrator, the LIF neuron has a leaky behavior, allowing a gradual leakage of the accumulated charge over time [18].

The important moment in the LIF neuron's operation is the firing threshold. When the accumulated charge surpasses this threshold, the neuron *fires*, generating a spike that propagates through the network. This spiking behavior introduces a temporal dimension to information processing, contributing to the efficiency and energy-saving features of SNNs [19].

A. Leaky Integrate-and-Fire (LIF) Neuron Model

The Leaky Integrate-and-Fire (LIF) neuron model is described by the following differential equations:

$$\tau_m \frac{dU(t)}{dt} = -(U(t) - U_{rest}) + R \cdot I_{in}(t) \quad (2)$$

In this equation, τ_m represents the membrane time constant, $U(t)$ is the membrane potential, U_{rest} is the membrane potential at rest, R is the membrane resistance, and $I_{\text{in}}(t)$ is the input current. The equation specifies how the membrane potential changes over time in response to the input current, considering the leakiness of the membrane, the rest potential, and the membrane resistance.

This iterative process can be used to simulate the membrane potential dynamics over time, capturing the spiking behavior of the LIF neuron. If the time interval is considered a very small discrete value, the equation can be solved by Euler's method in the following form :

$$u(t+\Delta t) = u(t) + \Delta t\left(-\frac{1}{\tau}\cdot(u(t) - u_{\text{rest}}) + I_{\text{in}}(t)\cdot R\right) \quad (3)$$

Now we will transform the equation into a discrete recursive form. Let's define $\beta = 1 - (1/\tau)$ as the decrease rate for membrane potential and set $\Delta t = 1$:

$$U(t+1) = \beta \cdot U(t) + (1-\beta)I_{in}(t+1) \quad (4)$$

Additionally, as shown in Fig. 2, a hyperparameter known as the firing threshold, θ, is defined. If $U(t) > \theta$, the neuron fires and emits a spike, and returns to the resting potential (U_{rest}).
For the input current (I) in the first layer, we express it as $I(t) = wx$, where w represents the weights associated with the input image (X).

For subsequent layers, the input current, I, is given by $I(t) = \sum_i w_i s_i(t)$, where , w_i are the weights, and $s_i(t)$ is the spike train generated by neurons in the previous layer (see Fig. 2).

Let's define the Spike function as:

$$\text{S}_{\text{out}}(t) = \begin{cases} 1, & \text{if } U(t) \geq \theta \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

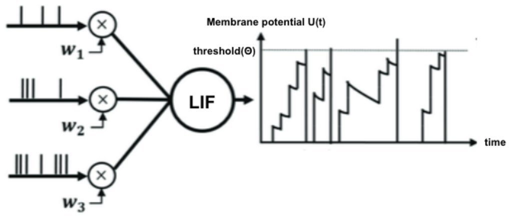

Fig. 2. Leaky integerate and fire neuron.

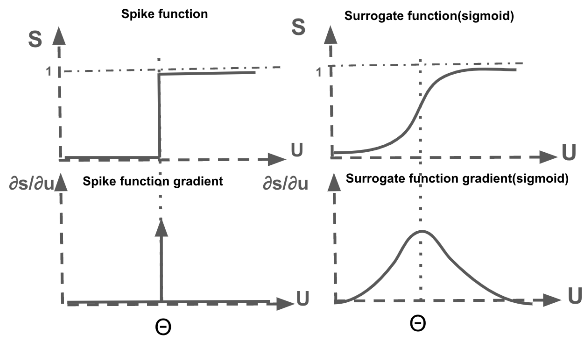

Fig. 3. Spike Function and Surrogate Gradient Function.

By considering the above definitions of spike function and input current, we can update the LIF membrane potential equation as:

$$U(t) = \beta U(t-1) + I(t) - S_{out}(t-1) \cdot \theta \tag{6}$$

where $S_{out}(t-1) \cdot \theta$ is a soft reset.

B. Surrogate Gradient

To adapt backpropagation to SNNs, it is crucial to modify a surrogate function to have a non-zero derivative. The absence of a gradient in the function implies that the network does not learn effectively. This technique involves replacing the derivative of the function in the chain rule with a related surrogate. In the domain of SNNs, this idea has been widely employed, where the gradient of the function is substituted with the gradient of a similar non-zero derivative activation function [8]. As shown in Fig. 3, here, we use Sigmoid as the surrogate function.

IV. Self-Supervised SNN

The proposed SNN architecture is comprised of a cascade of interlaying convolutional and max-pooling layers, followed by fully-connected layers. As explained in Section III-A, the input current to each LIF neuron in convolutional and fully-connected layers is the weighted summation of spikes it receives from its predecessors at each time step. While, the entry to each neuron in the input layer, is a constant current proportional to the pixel intensities. If the membrane potential of any of these neurons surpasses a specific threshold, it emits a spike immediately. In the max-pooling layers, neurons emit a spike right after receiving a spike from any of their afferents.

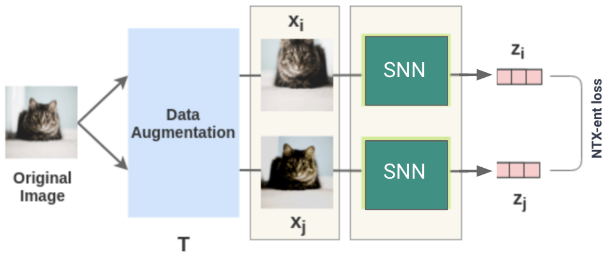

Fig. 4. The self-supervised contrastive learning in proposed SNN for a positive pair of augmentations.

As shown in Fig. 4, the SNN model independently processes the two image augmentations (positive or negative pairs) and the latent representations, z_i and z_j, are obtained by counting the number of spikes for each output neuron. These representations then goes into the NTX-ent loss function in order to maximize (minimize) similarity between positive (negative) pairs.

After completing the self-supervised learning proccess over unlabeled samples, the network will be fine-tuned over a small set of labeled samples of the downstream task. To do so, a fully-connected classifier layer is added on top of the SSL pre-trained SNN model to do the final classification. Labled samples are fed to the SNN model and a cross-entropy loss function is used to train the classification layer and fine-tune the pre-trained hidden layers. Note that, hidden layers are fine-tuned with a lower learning rate.

V. Evaluation scheme

Given that the self-supervised learning approach enables the generation of rich latent representations from unlabeled data, it is crucial to assess the reliability of these representations. Here, two evaluation protocols are implemented on the latent representations. First is a nearest neighbor analysis to visualize samples with similar representations. The second metric is the evaluation of SSL pre-trained SNN in a supervised classification task with a small set of labeled samples. To this end, we compare the fine-tuned SNN with SSL pre-training to an SNN with the same architecture trained on the labeled samples only.

A. Latent space evaluation

Typically, the nearest neighbor algorithm assigns the original data to the majority group of its neighbors, which have the least distance in the sample space. However, in this project, the nearest neighbor algorithm is employed for assessing the representations generated by the self-supervised learning algorithm. Here, we visualize images with the most similar latent representations to a target image. We expect the SSL pre-trained SNN to provide similar representations for images of the same category.

B. *Fine-tuning on MNIST and CIFAR10*

In addition to the aforementioned training approach, transfer learning is incorporated to enhance the network's performance. After the initial self-supervised learning phase, where representations are pre-trained from unlabeled data, a classifier layer is introduced. This classifier, typically implemented as a fully-connected layer, is trained using a limited amount of labeled data through supervised training. Notably, the weights of the pre-trained self-supervised learning phase are fine-tuned by a lower learning rate. The classifier's adaptation to the specific task is facilitated by utilizing the knowledge gained from the pre-trained self-supervised representations.

The transfer learning process aims to capitalize on the robust features learned in the pre-trained self-supervised phase and adapt them to the supervised classification task. By using the pre-trained self-supervised weights, the network leverages its ability to extract meaningful representations from unlabeled data. The introduction of the classifier layer in the fine-tuning phase allows the network to specialize in the specific classification task by adjusting the weights of the classifier based on the available labeled data. This two-step approach, combining pre-trained self-supervised learning and subsequent transfer learning for classification, is designed to optimize the network's accuracy and generalization ability in the final classification task with limited labeled data. The effectiveness of this strategy is assessed through a comparison of results between the two training approaches (with and without SSL pre-training), providing insights into the impact of pre-trained self-supervised learning on subsequent classification performance.

VI. Experiments and Results

For our implementation, we applied three widely used data augmentation methods which are rotation, crop, and Gaussian blur on CIFAR-10 and MNIST datasets to explore their impact on the performance of the SNN model in the self-supervised learning method. In each batch of data, different augmentions of the same image are fed to the SNN as positive pairs, and augmentations of different images are considered as negative pairs. To develop SNNs and model leaky integrate-and-fire neurons, the snntorch package [6] was employed, and connections between neurons were implemented using PyTorch [20].

Fig.5 shows images with the nearest latent representations learned by the SLL pre-trained SNN to some randomly selected sample images from CIFAR10 dataset. This gives insights into the efficacy of the self-supervised learning paradigm in capturing meaningful features for representing similar images closer in the latent space and providing distant representations for dissimilar images.

To incorporate a classifier in the transfer learning experiment, we add a fully-connected layer with the number of classes at the end of the SSL pre-trained SNN. By counting the number of spikes emitted by each output neuron, the neuron with the maximum spike count determines the category of the input image. One of the primary goals of self-supervised learning is to reduce the reliance on labeled data, only 20% of the original labeled data is used in the fine-tuning stage of the training process. The remaining 80% of the data has been utilized for self-supervised learning.

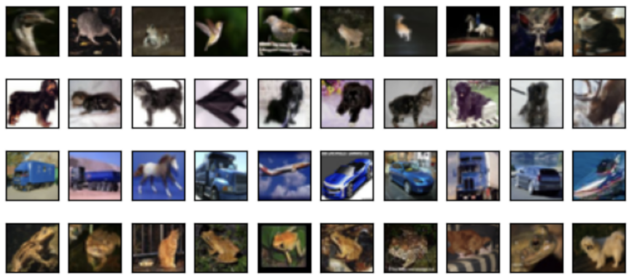
Fig. 5. KNN algorithm on spike representations for some samples of CIFAR-10

In self-supervised learning SNN for generating spike representations, parameters such as batch size, augmentation method, and the number of time steps in the final operation of the network have a significant impact. The experiments conducted to observe the effect of each parameter are presented in the tables I, II, and III. In each table, the best accuracy of the pre-trained SNN is shown for both MNIST and CIFAR-10 datasets. As a result the larger the batch size and the more time step, the better our network performs. For augmentation, gaussian noise and rotation had the best accuracy.

time step	MNIST	CIFAR-10
10	78.21%	51.63%
20	91.54%	57.84%
30	94.23%	62.24%

TABLE I
SELF-SUPERVISED SNN ACCURACY ACCORDING TO DIFFERENT TIME STEPS

batch size	MNIST	CIFAR-10
64	87.45%	52.76%
128	90.87%	57.58%
256	94.23%	62.24%

TABLE II
SELF-SUPERVISED SNN ACCURACY ACCORDING TO DIFFERENT BATCH SIZES

Augmentation	MNIST	CIFAR-10
crop-rotate	92.43%	61.21%
crop-gaussian noise	93.86%	60.12%
rotate-gaussian noise	94.23%	62.24%

TABLE III
SELF-SUPERVISED SNN ACCURACY ACCORDING TO DIFFERENT DATA AUGMENTATIONS

Our model has also a better performance on MNIST dataset with classifier in comparison with [21] (an spiking self-supervised model), and also in comparison with SimCLR ANN in [22] on CIFAR10 dataset. The results are shown in table IV.

Considering the main objective of applying self-supervised learning to minimize the need for labeled data, the expectation

Method	MNIST	CIFAR-10
Ours	94.23%	62.24%
[21]	93.06%	—
[22]	—	60.1%

TABLE IV
COMPARISON BETWEEN OUR SELF-SUPERVISED SNN MODEL ACCURACY AND TWO OTHER MODELS WITH ALMOST THE SAME ARCHITECTURE ON MNSIT AND CIFAR10 DATASET.

is that after this process, the network should exhibit better accuracy in supervised learning on labeled data with a small volume. Consequently, the network with the same architecture is further fine-tuned in a supervised manner on 20% of the total data. The results are then compared in Fig. 6 and Fig. 7, Following this experiment, it can be concluded that fine-tuned SSL pre-trained SNN can outperform the SNN that is only trained on labeled samples. Hence, it can be said that SSL can reduce the need for more labeled samples by using sufficiently large unlabeled samples to learn informative latent representations.

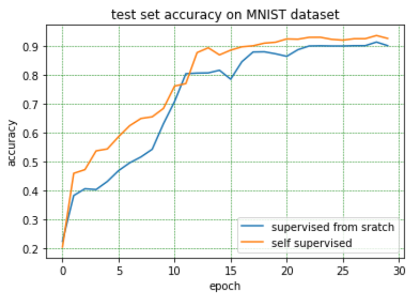

Fig. 6. supervised vs self-supervised method on MNIST dataset

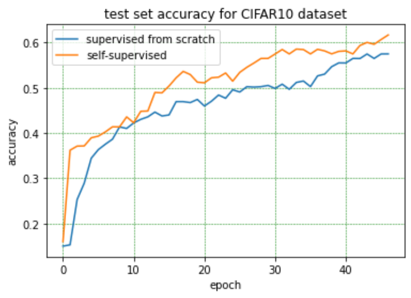

Fig. 7. supervised vs self-supervised method on CIFAR-10 dataset

VII. Conclusion

Exploring self-supervised learning for spiking neural networks offers innovative ways to advance learning techniques in SNNs. Despite challenges such as scarce labeled data and resource-intensive label generation, these approaches show the potential of using unlabeled samples to construct meaningful and informative representations. By harnessing the efficiency of spiking neural networks, self-supervised learning enables robust pattern recognition without extensive labeled data. This has practical applications in various fields, including computer vision. Ongoing improvements aim to boost network accuracy, creating a reliable computational tool for machine learning and data processing applications.

References

[1] A. Taherkhani, A. Belatreche, Y. Li, G. Cosma, L. P. Maguire, and T. M. McGinnity, “A review of learning in biologically plausible spiking neural networks,” *Neural Networks*, vol. 122, pp. 253–272, 2020.
[2] Z. Bing, C. Meschede, F. Röhrbein, K. Huang, and A. C. Knoll, “A survey of robotics control based on learning-inspired spiking neural networks,” *Frontiers in neurorobotics*, vol. 12, p. 35, 2018.
[3] F. Ponulak and A. Kasiński, “Introduction to spiking neural networks: Information processing, learning and applications,” *Acta neurobiologiae experimentalis*, vol. 71, p. 409, 2011.
[4] N. Rathi, G. Srinivasan, P. Panda, and K. Roy, “Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation,” *arXiv preprint arXiv:2005.01807*, 2020.
[5] S. R. Kheradpisheh, M. Mirsadeghi, and T. Masquelier, “Spiking neural networks trained via proxy,” *IEEE Access*, vol. 10, pp. 70769–70778, 2022.
[6] J. K. Eshraghian, M. Ward, E. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu, “Training spiking neural networks using lessons from deep learning,” *arXiv preprint arXiv:2109.12894*, 2021.
[7] T. H. Rafi, “A brief review on spiking neural network-a biological inspiration,” *Preprints*, 2021.
[8] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, “Deep learning in spiking neural networks,” *Neural Networks*, vol. 111, pp. 47–63, 2019.
[9] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, “A survey on contrastive self-supervised learning,” *Technologies*, vol. 9, no. 1, p. 2, 2020.
[10] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer, “S4l: Self-supervised semi-supervised learning,” *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1476–1485, 2019.
[11] P. H. Le-Khac, G. Healy, and A. F. Smeaton, “Contrastive representation learning: A framework and review,” *IEEE Access*, vol. 8, pp. 193907–193934, 2020.
[12] P. Kumar, P. Rawat, and S. Chauhan, “Contrastive self-supervised learning: review, progress, challenges and future research directions,” *International Journal of Multimedia Information Retrieval*, vol. 11, no. 4, pp. 461–488, 2022.
[13] S. Gidaris and N. Komodakis, “Unsupervised representation learning by predicting image rotations,” *International Conference on Learning Representations (ICLR)*, 2018.
[14] H. H. Mao, “A survey on self-supervised pre-training for sequential transfer learning in neural networks,” *arXiv preprint arXiv:2007.00800*, 2020.
[15] R. Moreno-Bote and N. Parga, “Auto-and crosscorrelograms for the spike response of leaky integrate-and-fire neurons with slow synapses,” *Physical Review Letters*, vol. 96, no. 2, p. 028101, 2006.
[16] A. D. Rast, F. Galluppi, X. Jin, and S. B. Furber, “The leaky integrate-and-fire neuron: A platform for synaptic model exploration on the spinnaker chip,” in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, 2010, pp. 1–8.
[17] A. L. Hodgkin and A. F. Huxley, “A quantitative description of membrane current and its application to conduction and excitation in nerve,” *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952.
[18] J. Feng, “Is the integrate-and-fire model good enough?—a review,” *Neural networks*, vol. 14, no. 6-7, pp. 955–975, 2001.
[19] A. N. Burkitt, “A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input,” *Biological cybernetics*, vol. 95, pp. 1–19, 2006.
[20] A. Paszke et al., “PyTorch: An imperative style, high-performance deep learning library,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
[21] B. Chamand and P. Joly, “Self-Supervised Spiking Neural Networks applied to Digit Classification,” *Proceedings of the 19th International Conference on Content-based Multimedia Indexing*, pp. 196–200, 2022.
[22] Y. Bai, Y. Yang, W. Zhang, and T. Mei, “Directional self-supervised learning for heavy image augmentations,” *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16692–16701, 2022.