Poster ID: TUE-PM-360

Paper tag: 360

# Constructing Deep Spiking Neural Networks from Artificial Neural Networks with Knowledge Distillation

Qi Xu[1], Yaxin Li[1], Jiangrong Shen[2*], Jian K. Liu[3], Huajin Tang[2], Gang Pan[2*]

[1]Dalian University of Technology, [2]Zhejiang University, [3]University of Leeds

# Introduction

Motivation: improve the performance of SNN through constructing deeper structures

ANNs: easy to train a deep SNN directly through backpropagation method

SNNs: difficult to train a deep SNN directly due to non-differentiable spikes

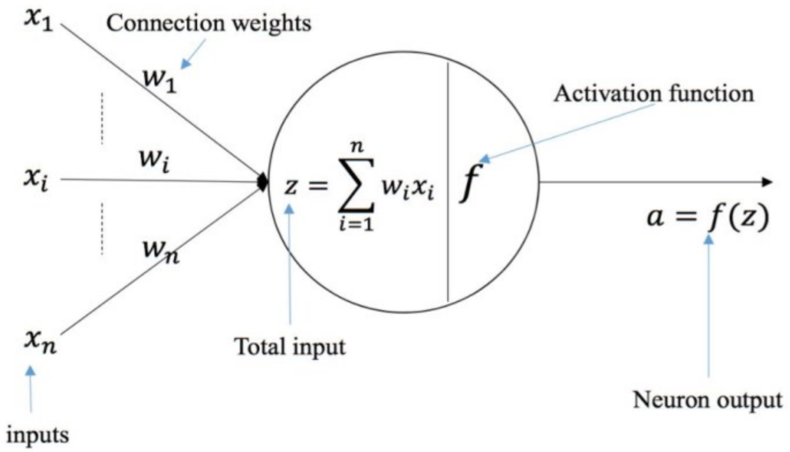


Fig1. The artificial neuron of ANNs.

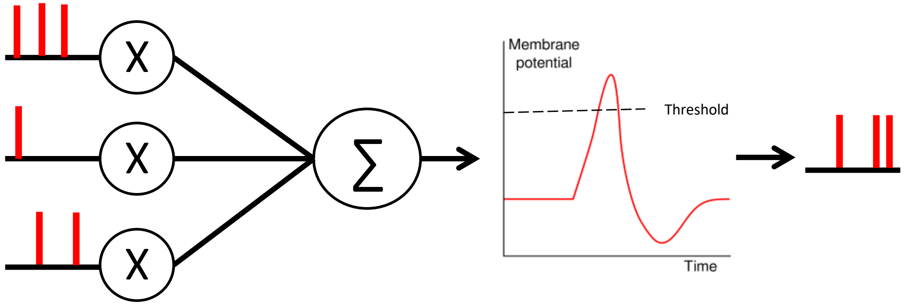


Fig2. The spiking neuron of SNNs.

# Introduction

## Related Work: it is difficult to train deep SNNs with loss function directly

### Surrogate Gradient Training Methods

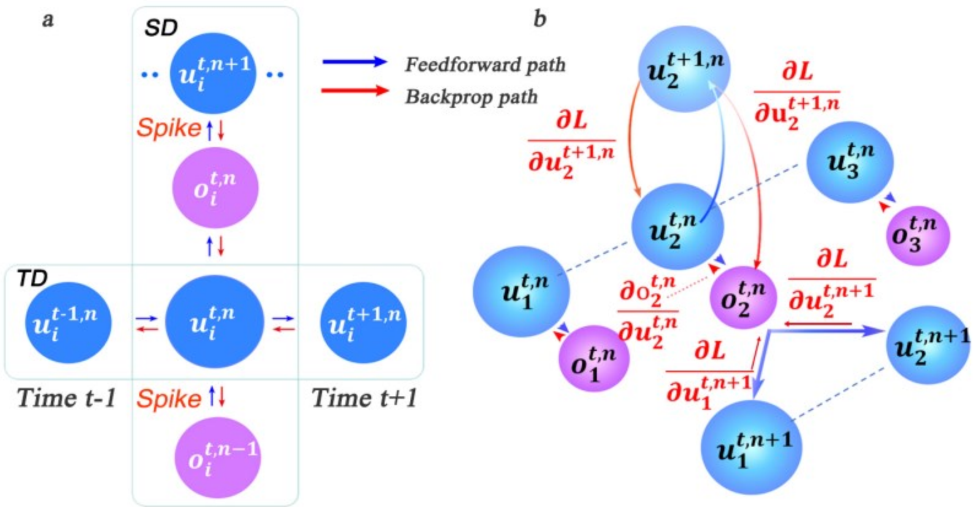


Fig3. Using surrogate gradients to implement error backpropagation for training SNNs.[1]

### ANN-to-SNN Conversion Methods

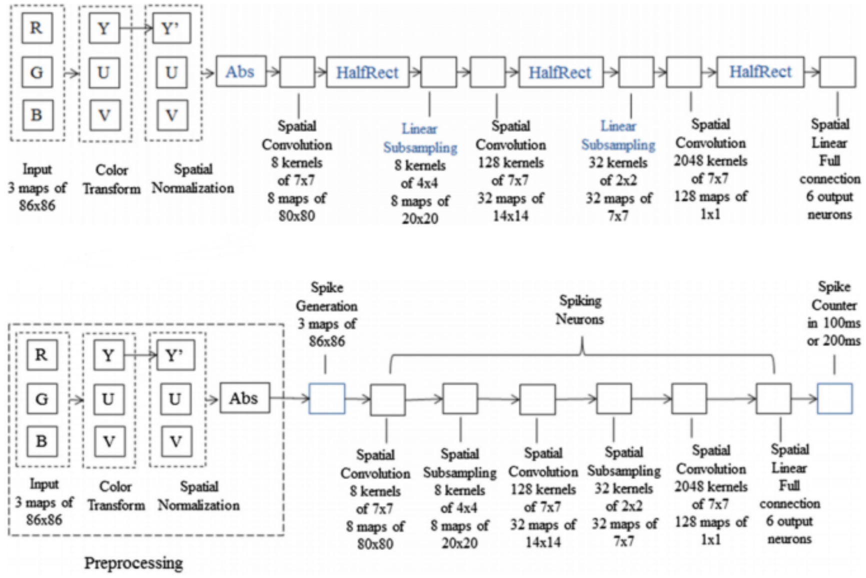


Fig4. Transferring the trained weights from an ANN to an SNN to indirectly train the SNN.[2]

[1] Wu, Y., Deng, L., Li, G., Zhu, J. and Shi, L., 2018. Spatio-temporal backpropagation for training high-performance spiking neural networks. Frontiers in neuroscience, 12, p.331.
[2] Cao, Y., Chen, Y. and Khosla, D., 2015. Spiking deep convolutional neural networks for energy-efficient object recognition. International Journal of Computer Vision, 113, pp.54-66.

# Methodology--overall training algorithm



Method: The joint ANN-to-SNN knowledge distillation method transfers **the hidden knowledge** in a pre-trained teacher ANN model to the student SNN model to guide the training of SNN.

**Spiking neuron model: IF model**

$$H[t] = f(V(t-1), X(t)) = V(t-1) + X(t)$$

$$S[t] = \Theta(\mathrm{H}[\mathrm{t}] - V_{\mathrm{th}}) = \begin{cases} 1, & \mathrm{H}[\mathrm{t}] \geq V_{\mathrm{th}} \\ 0, & \mathrm{H}[\mathrm{t}] < V_{\mathrm{th}} \end{cases}$$

**Algorithm 1** Training student SNN model with knowledge distillation.

**Require:** pre-trained teacher ANN model T, initialized student SNN model S, input dataset samples X, true labels $y_{\text{true}}$.
**Ensure:** SNN model with KD

#forward propagation<br>
**for** t=1 to L-1 **do**<br>
  $o[0] = encode(X)$;<br>
  **for** t=1 to T **do**<br>
    #calculate the membrane potential<br>
    $V(t) = V(t-1) + W_l O_{l-1}(t)$;<br>
    **if** $V(t) > V_{th}$ **then**<br>
      #fire a spike<br>
      $O_l(t) = 1$;<br>
      # reset the membrane potential<br>
      $V(t) = V_{reset}$;<br>
    **end if**<br>
  **end for**<br>
**end for**<br>
#calculate the spike rate<br>
$O_L(t) = counter(L)/T$;<br>
calculate the total loss $L_{KD}$;<br>
#backward propagation<br>
**for** t=L-1 to 1 **do**<br>
  **for** t=1 to T **do**<br>
    calculate the gradients $\frac{\partial L_{KD}}{V_l(t)}$;<br>
    update $W_l$;<br>
  **end for**<br>
**end for**

# Methodology--response-based knowledge distillation

Method: The **response-based knowledge distillation** transfers the knowledge from the output layer (teacher ANN model) to the student SNN model to guide the training of the SNN.

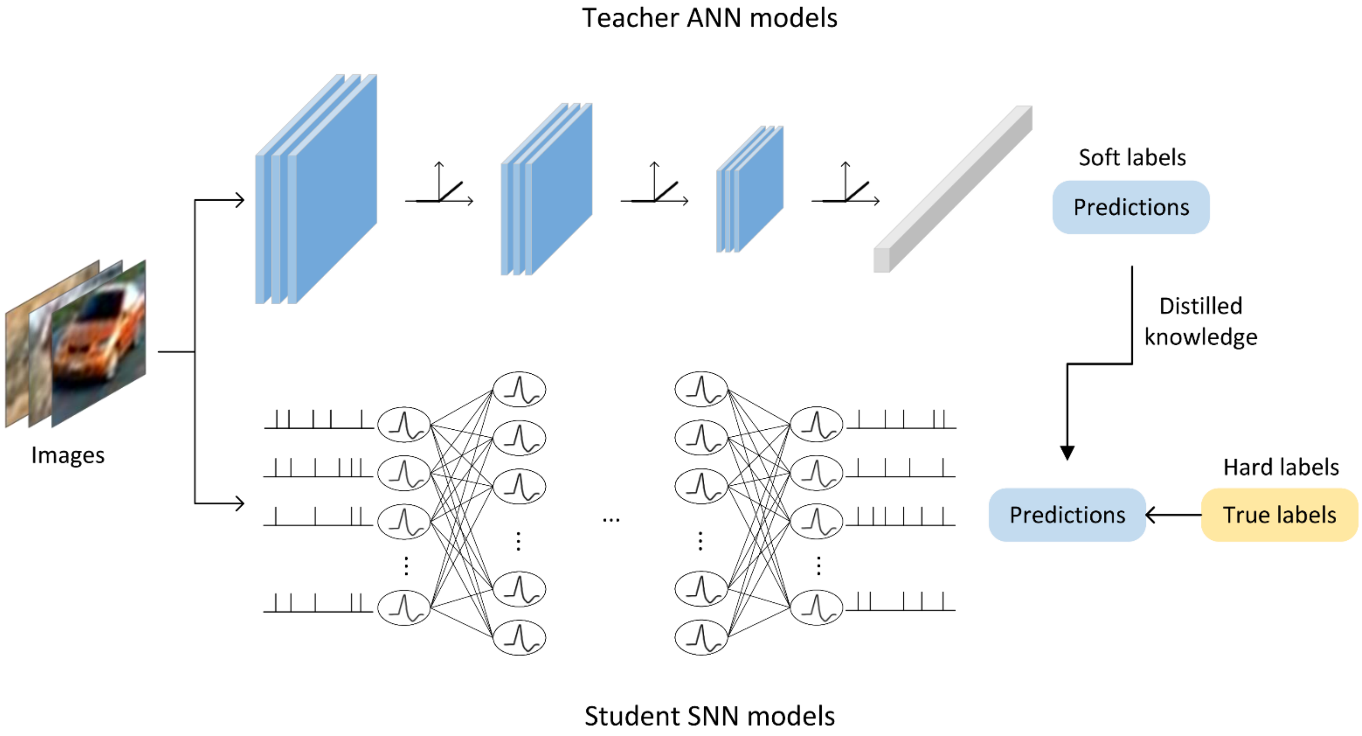


Fig5. KDSNN with response-based knowledge distillation.

## Loss function

$$L_{KD} = \alpha T^2 * CrossEntropy(Q_S^\tau, Q_T^\tau) + (1-\alpha) * CrossEntropy(Q_S, y_{true})$$

$T$: temperature

$Q_S$ : the output of student model

$Q_S^\tau, Q_T^\tau$ : the output of student and teacher model after softening

$y_{true}$ : the true labels

# Methodology--feature-based knowledge distillation

Method: The **feature-based knowledge distillation** utilizes the hidden knowledge in some intermediate layers of ANN to guide the training of SNN.

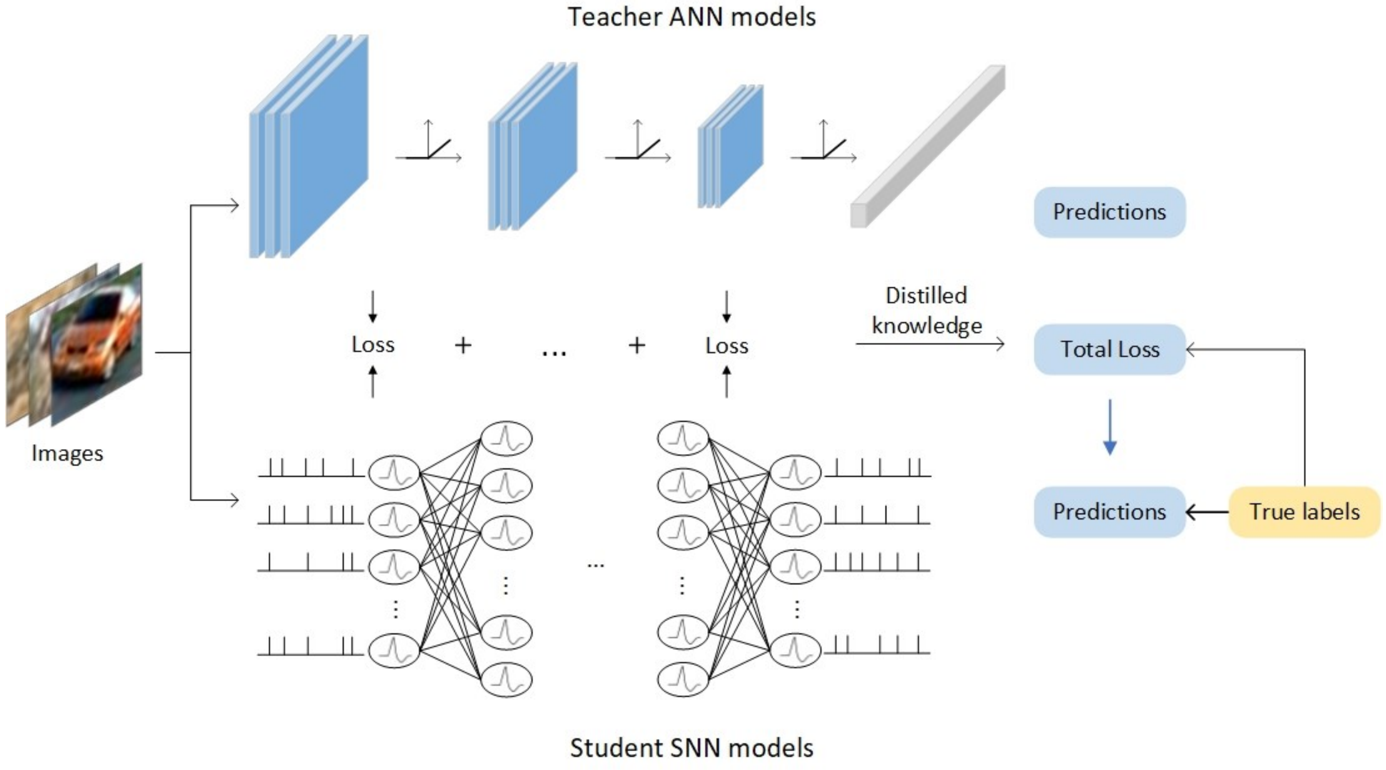


Fig6. KDSNN with feature-based knowledge distillation.

## Loss function

$$L_{distill} = \sum_{i}^{WHC} \begin{cases} 0 & \text{if } S_i \leq T_i \leq 0 \\ (T_i - S_i)^2 & \text{otherwise} \end{cases}$$

$$L_{KD} = L_{task} + \alpha * L_{distill}$$

$T_i$ : the features of the teacher ANN model
$S_i$ : the spiking based features of the student SNN model
$L_{task}$ : the loss between true labels and the real output of the student SNN modey $_{true}$ : the true
$L_{distill}$ : the loss of the intermediate layers

# Experiments

## Results: Evaluation under **different knowledge levels and architectures**

| Method | SNN Model | ANN Model | ANN Acc.(%) | SNN Acc.(%) | KDSNN ACC.(%) | Improvement(%) |
|---|---|---|---|---|---|---|
| Response-based | VGG11 | ResNet18 | 93.20 | 88.44 | 89.12 | 0.68 |
| | | WRN28-4 | 93.10 | 88.44 | 89.43 | 0.99 |
| | | Pyramidnet18 | 95.10 | 88.44 | 89.51 | 1.07 |
| | WRN16-2 | ResNet18 | 93.20 | 90.34 | 90.98 | 0.64 |
| | | WRN28-4 | 93.10 | 90.34 | 91.14 | 0.80 |
| | | Pyramidnet18 | 95.10 | 90.34 | 91.11 | 0.77 |
| | ResNet18 | Pyramidnet18 | 95.10 | 92.68 | 93.41 | 0.73 |
| Feature-based | WRN16-2 | WRN28-4 | 93.10 | 90.34 | 91.03 | 0.69 |
| | | Pyramidnet18 | 95.10 | 90.34 | 92.10 | 1.76 |
| | | PreResNet20 | 92.36 | 90.34 | 91.57 | 1.23 |
| | ResNet14 | WRN28-4 | 93.10 | 87.46 | 87.84 | 0.38 |
| | | Pyramidnet18 | 95.10 | 87.46 | 88.20 | 0.74 |
| | | PreResNet20 | 92.36 | 87.46 | 87.90 | 0.44 |

Tab1. Test accuracies of KDSNN with different teacher ANNs and Student SNNs on **CIFAR10**. To show the effectiveness of the proposed KDSNN training method adequately, we design and implement several KD methods to construct efficient student SNN models under the utilization of feature representations of teacher ANNs.

# Experiments--Noise resistance

Results: KDSNN method can learn rich knowledge from teacher ANNs and behave better than original SNNs **in a noisy environment**

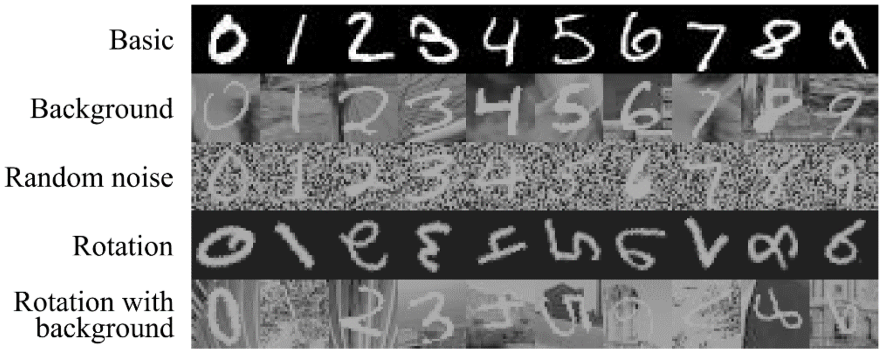


Fig.7 MNIST dataset with different noise

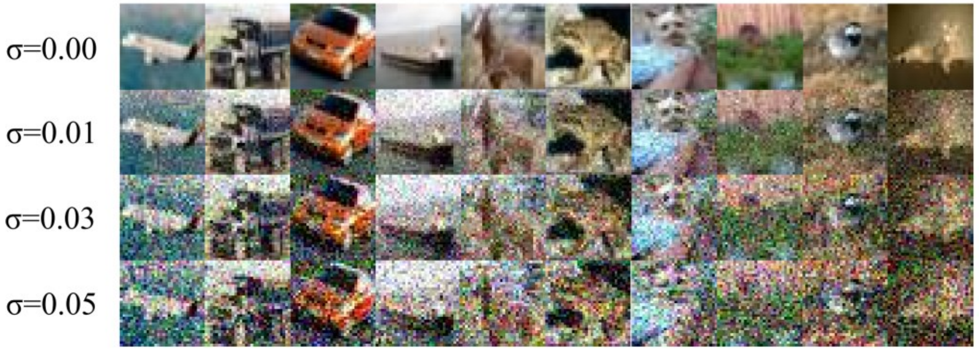


Fig.8 CIFAR10 dataset with gaussian noise

| Dataset | Noise | ANN model | ANN Acc.(%) | SNN model | SNN Acc. (%) | KDSNN Acc. (%) | Improvement(%) |
|---|---|---|---|---|---|---|---|
| CIFAR10 | Gaussian noise($\sigma = 0.01$) | ResNet18 | 83.00 | VGG11 | 81.63 | 82.90 | 1.27 |
| | Gaussian noise($\sigma = 0.03$) | ResNet18 | 80.40 | VGG11 | 76.42 | 77.96 | 1.54 |
| | Gaussian noise($\sigma = 0.05$) | ResNet18 | 77.00 | VGG11 | 73.40 | 74.23 | 0.83 |
| MNIST | Background | ResNet18 | 97.72 | 2conv | 95.04 | 96.35 | 1.31 |
| | Random noise | ResNet18 | 96.95 | 2conv | 95.31 | 95.79 | 0.48 |
| | Rotation | ResNet18 | 96.01 | 2conv | 94.43 | 95.34 | 0.91 |
| | Rotation with background | ResNet18 | 86.59 | 2conv | 80.96 | 81.82 | 0.86 |

Tab2. Classification performance evaluation of KDSNN on CIFAR10 and MNIST with different types of noise.

# Experiments--power efficient

Results: KDSNN method can learn rich knowledge from teacher ANNs and behave better than original SNNs **in a noisy environment**

**Advantage:**
**low power consumption**

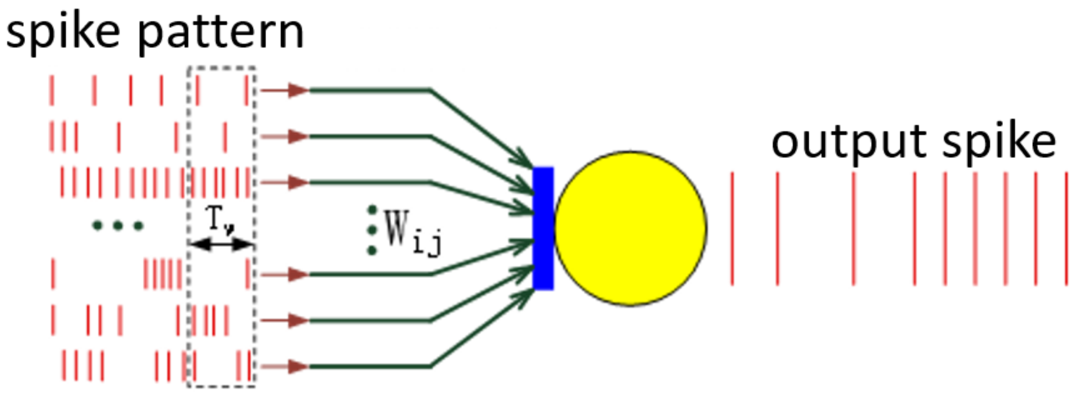


Fig.9 spike sequence

| Dataset | ANN Model | ANN params | ANN FLOPs | SNN Model | SNN params | SNN SynOps |
|---|---|---|---|---|---|---|
| MNIST | ResNet18 | 11.17M | 457.72M | 2conv | 7.39M | 0.05M |
| CIFAR10 | ResNet18<br>WRN28-4<br>Pyramidnet18 | 11.17M<br>5.85M<br>1.56M | 557.88M<br>849.33M<br>368.74M | VGG11<br>WRN16-2<br>ResNet18 | 9.75M<br>0.69M<br>11.17M | 0.09M<br>0.15M<br>0.44M |

Tab3. Comparison of the memory and operations from ANN and the proposed SNN models.

# Experiments

## Results: Performance comparison with other methods

| Dataset | Method | ANN Architecture | SNN Architecture | ANN ACC.(%) | SNN Acc.(%) | timestep |
|---|---|---|---|---|---|---|
| MNIST | SDNN [17] | - | 2conv-2pool | - | 98.40 | 30 |
| | STBP [32] | - | 784-800-10 | - | 98.89 | 30 |
| | ANTLR [18] | - | 784-800-10 | - | 97.60 | 100 |
| | ASF-BP [31] | - | LeNet5 | - | 99.65 | 400 |
| | **Proposed** | ResNet18 | 2conv | 99.59 | 99.37 | 4 |
| CIFAR10 | SPIKE-NORM [27] | VGG16 | VGG16 | 91.70 | 91.55 | 2500 |
| | Hybrid Train [24] | VGG16 | VGG16 | 92.81 | 91,13 | 100 |
| | RMP [13] | VGG16 | VGG16 | 93.63 | 93.63 | 2048 |
| | Opt. [5] | VGG16 | VGG16 | 92.34 | 92.29 | 16 |
| | **Proposed** | Pyramidnet18 | VGG16 | 95.17 | 91.05 | 4 |
| CIFAR10 | SPIKE-NORM [27] | Resnet20 | Resnet20 | 89.10 | 87.46 | 2500 |
| | Hybrid Train [24] | Resnet20 | Resnet20 | 93.15 | 92.22 | 250 |
| | RMP [13] | Resnet20 | Resnet20 | 91.47 | 91.36 | 2048 |
| | Opt. [5] | Resnet20 | Resnet20 | 92.46 | 92.41 | 16 |
| | **Proposed** | Pyramidnet18 | Resnet18 | 95.17 | 93.41 | 4 |

Tab4. Summary comparison of classification accuracies with other spiking based models. To better demonstrate the superior performance of the proposed KDSNN model, we compare the proposed KDSNN training method with other methods. The KDSNN training method could improve the performance of SNNs with high classification accuracy and fewer time steps.

# Conclusion

## Summary and Future Outlook:

- We proposed spiking based surrogate gradient methods and ANN-to-SNN conversion **combination-based training**;
- The proposed method would build SNN models **faster** which means we can use less time to achieve or even exceed the performance of other spiking models;
- In our future work, we will **expand both structures** of ANNs and SNNs to utilize the advantages of the proposed.