Method	Convert AP	Not Convert AP
VGG-16	24.88	49.53
ResNet-34	50.21	52.87

Table 8. Impact of AvgPool2d layers in SNN.

A. Conversion Error Analysis

We use $||\cdot||$ to denote the Frobenius norm. According to Eq. (1) and Eq. (10), the activation of ANN can be computed by

$$\begin{aligned} ||\mathbf{x}^{(n)} - \bar{\mathbf{s}}^{(n)}|| &= ||h(\mathbf{W}^{(n-1)}\mathbf{x}^{(n-1)}) - g(\mathbf{W}^{(n-1)}\bar{\mathbf{s}}^{(n-1)})|| \\ &= ||h(\mathbf{W}^{(n-1)}\mathbf{x}^{(n-1)}) - h(\mathbf{W}^{(n-1)}\bar{\mathbf{s}}^{(n-1)}) + h(\mathbf{W}^{(n-1)}\bar{\mathbf{s}}^{(n-1)}) - g(\mathbf{W}^{(n-1)}\bar{\mathbf{s}}^{(n-1)})|| \quad (20) \\ &\leq ||\mathbf{W}^{(n-1)}(\mathbf{x}^{(n-1)} - \bar{\mathbf{s}}^{(n-1)}) + Err^{(n)}||, \quad (21) \end{aligned}$$

where $h(\cdot)$ is the ReLU function in ANN and $g(\cdot)$ is the clipfloor function in Eq. (10). Eq. (21) is based on the fact that activation $h(\cdot)$ is a piecewise linear function with gradient less than or equal to 1. $Err^{(n)} = h(\mathbf{W}^{(n-1)}\bar{\mathbf{s}}^{(n-1)}) - g(\mathbf{W}^{(n-1)}\bar{\mathbf{s}}^{(n-1)})$. Recursively apply Eq. (21), then we have

$$\begin{aligned} ||\mathbf{x}^{(n)} - \bar{\mathbf{s}}^{(n)}|| &\leq ||\mathbf{W}^{(n-1)}(\mathbf{x}^{(n-1)} - \bar{\mathbf{s}}^{(n-1)}) + Err^{(n)}|| \\ &\leq ||\mathbf{W}^{(n-1)}\mathbf{W}^{(n-2)}(\mathbf{x}^{(n-2)} - \bar{\mathbf{s}}^{(n-2)}) + \mathbf{W}^{(n-1)}Err^{(n-1)} + Err^{(n)}|| \quad (22) \\ &\leq ||(\mathbf{x}^{(1)} - \bar{\mathbf{s}}^{(1)})\prod_{\ell=1}^{n}\mathbf{W}^{(\ell)} + \sum_{\ell=1}^{n}Err^{(\ell)}\prod_{k=\ell}^{n}\mathbf{W}^{(k)}|| \quad (23) \end{aligned}$$

Note that the first term in Eq. (23) is 0 if we use the same image input to ANN and SNN.

B. Converting Average Pooling Layers

Consider a batch of input to AvgPool2d layers with size of $[N, C, H, W]$ where N is the batch size, C is the channel number, H and W are the width and heights of inputs, as well as kernel size of AvgPool2d represented by $[kW, kH]$, we can precisely describe the AvgPool2d forwarding as

$$out(N_i, C_j, h, w) = \frac{1}{kW \times kH}\sum_{m=0}^{kH-1}\sum_{n=0}^{kW-1} input(N_i, C_j, stride[0] \times h + m, stride[1] \times w + n). \quad (24)$$

This forwarding function is a special case of depthwise Conv2d, where the kernel size of Conv2d is equal to $[kW, kH]$. The weights of the Conv2d should be constant for all elements in the kernel $\frac{1}{kW \times kH}$, and the bias of the Conv2d should be zero.

We hereby conduct an ablation study that investigates the impact of AvgPool2d layers conversion. We test VGG-16 and ResNet-34 on ImageNet ($T = 32$, Light Pipeline). VGG-16 has five 2×2 AvgPool2d layers. ResNet-34 has one 2×2 and one 7×7 AvgPool2d layers. On VGG-16, the accuracy is much lower if we convert the AvgPool2d layers. On ResNet-34, the impact of AvgPool2d layers is much lower than that on VGG-16, with only a 2% accuracy drop if we convert AP.

C. ANN Training Implementation

C.1. ImageNet

The ImageNet dataset (Deng et al., 2009) contains 120M training images and 50k validation images. For training preprocessing, we random crop and resize the training images to 224×224. We additionally apply CollorJitter with brightness=0.2, contrast=0.2, saturation=0.2, and hue=0.1. For test images, they are center-cropped to the same size. For all architectures we tested, the Max Pooling layers are replaced to Average Pooling layers and are further converted to depthwise convolutional layers. The ResNet-34 contains a deep-stem layer (i.e., three 3×3 conv. layers to replace the original 7×7 first conv. layer) as described in He et al. (2019). We use Stochastic Gradients Descent with a momentum of 0.9 as the optimizer. The learning rate is set to 0.1 and followed by a cosine decay schedule (Loshchilov & Hutter, 2016). Weight decay is set to 10^{-4}, and the networks are optimized for 120 epochs. We also apply label smooth (Szegedy et al., 2016)(factor=0.1)

and EMA update with 0.999 decay rate to optimize the model. For the MobileNet pre-trained model, we download it from `pytorchcv`[2].

C.2. CIFAR

The CIFAR 10 and CIFAR100 dataset (Krizhevsky et al.) contains 50k training images and 10k validation images. We set padding to 4 and randomly cropped the training images to 32×32. Other data augmentations include (1)random horizontal flip, (2) Cutout (DeVries & Taylor, 2017) and (3) AutoAugment (Cubuk et al., 2019). For ResNet-20, we follow prior works (Han et al., 2020; Han & Roy, 2020) who modify the official network structures proposed in He et al. (2016) to make a fair comparison. The modified ResNet-20 contains 4 stages with an additional deep-stem layer. For VGG-16 without BN layers, we add Dropout with a 0.25 drop rate to regularize the network. For MobileNet-CIFAR, we set the stride of the first conv. Layer to 1 to decreases the stage number to 4. For the model with BN layers, we use Stochastic Gradients Descent with a momentum of 0.9 as the optimizer. The learning rate is set to 0.1 and followed by a cosine decay schedule (Loshchilov & Hutter, 2016). Weight decay is set to 5×10^{-4} and the networks are optimized for 300 epochs. For networks without BN layers, we set weight decay to 10^{-4} and learning rate to 0.005.

D. Results on CIFAR

[2] https://pypi.org/project/pytorchcv/

Method	Use BN	Convert AP	ANN Acc.	$T = 32$	$T = 64$	$T = 128$	$T = 256$	$T \geq 2048$
ResNet-20 (He et al., 2016) CIFAR100								
Spike-Norm (Sengupta et al., 2018)	✗	✗	69.72	-	-	-	-	64.09
RMP (Han et al., 2020)	✗	✗	68.72	27.64	46.91	57.69	64.06	67.82
TSC (Han & Roy, 2020)	✗	✗	68.72	-	-	58.42	65.27	68.18
Opt. (Deng & Gu, 2021)*	✗	✓	68.40	63.39	67.51	68.37	68.53	68.37
Ours (Light Pipeline)	✗	✓	68.40	**65.14**	**67.63**	**68.28**	**68.42**	68.37
Opt. (Deng & Gu, 2021)*	✓	✓	77.16	51.27	70.12	75.81	77.22	77.19
Ours (Light Pipeline)	✓	✓	77.16	**75.53**	**77.08**	**77.50**	**77.59**	77.25
Ours (Advanced Pipeline)	✓	✓	77.16	**76.32**	**77.29**	**77.73**	**77.63**	77.25
VGG-16 (Simonyan & Zisserman, 2014) CIFAR100								
Spike-Norm (Sengupta et al., 2018)	✗	✗	71.22	-	-	-	-	70.77
RMP (Han et al., 2020)	✗	✗	71.22	-	-	63.76	68.34	70.93
TSC (Han & Roy, 2020)	✗	✗	71.22	-	-	69.86	70.65	70.97
Opt. (Deng & Gu, 2021)*	✗	✓	70.21	56.16	62.93	67.45	69.36	70.35
Ours (Light Pipeline)	✗	✓	70.21	**64.53**	**67.14**	**68.99**	**69.98**	70.30
Opt. (Deng & Gu, 2021)*	✓	✓	77.89	7.64	21.84	55.04	73.54	77.71
Ours (Light Pipeline)	✓	✓	77.89	**65.73**	**72.38**	**75.82**	**77.12**	77.87
Ours (Advanced Pipeline)	✓	✓	77.89	**73.55**	**76.64**	**77.40**	**77.68**	77.87
MobileNet (Howard et al., 2017) CIFAR100								
Opt. (Deng & Gu, 2021)*	✓	✓	73.23	1.28	4.88	39.39	65.79	73.01
Ours (Light Pipeline)	✓	✓	73.23	**40.06**	**62.81**	**69.41**	**71.98**	73.19
Ours (Advanced Pipeline)	✓	✓	73.23	**42.64**	**63.24**	**71.02**	**72.54**	73.18
ResNet-20 (He et al., 2016) CIFAR10								
Spike-Norm (Sengupta et al., 2018)	✗	✗	89.10	-	-	-	-	87.46
Hybrid Train (Rathi et al., 2019)	✗	✗	93.15	-	-	-	92.22	92.94
RMP (Han et al., 2020)	✗	✗	91.47	-	-	87.60	89.37	91.36
TSC (Han & Roy, 2020)	✗	✗	91.47	-	69.38	88.57	90.10	91.42
Opt. (Deng & Gu, 2021)*	✗	✓	93.94	86.67	91.96	93.48	93.76	93.94
Ours (Light Pipeline)	✗	✓	93.94	**93.00**	**93.61**	**93.85**	**93.89**	93.94
Opt. (Deng & Gu, 2021)*	✓	✓	95.46	84.06	92.48	94.68	95.30	94.42
Ours (Light Pipeline)	✓	✓	95.46	**94.44**	**95.20**	**95.29**	**95.36**	95.47
Ours (Advanced Pipeline)	✓	✓	95.46	**94.78**	**95.30**	**95.42**	**95.41**	95.45
VGG-16 (Simonyan & Zisserman, 2014) CIFAR10								
Spike-Norm (Sengupta et al., 2018)	✗	✗	91.70	-	-	-	-	91.55
Hybrid Train (Rathi et al., 2019)	✗	✗	92.81	-	-	91.13	-	92.48
RMP (Han et al., 2020)	✗	✗	93.63	60.30	90.35	92.41	93.04	93.63
TSC (Han & Roy, 2020)	✗	✗	93.63	-	92.79	93.27	93.45	93.63
Opt. (Deng & Gu, 2021)*	✗	✓	93.51	88.79	91.12	92.74	93.29	93.55
Ours (Light Pipeline)	✗	✓	93.51	**91.82**	**92.57**	**93.21**	**93.35**	93.54
Opt. (Deng & Gu, 2021)*	✓	✓	95.72	76.24	90.64	94.11	95.33	95.73
Ours (Light Pipeline)	✓	✓	95.72	**94.02**	**95.20**	**95.61**	**95.77**	95.73
Ours (Advanced Pipeline)	✓	✓	95.72	**93.71**	**95.14**	**95.65**	**95.79**	95.79
MobileNet (Howard et al., 2017) CIFAR10								
Opt. (Deng & Gu, 2021)*	✓	✓	92.48	9.98	23.99	79.35	90.49	92.35
Ours (Light Pipeline)	✓	✓	92.48	**81.94**	**89.47**	**91.61**	**92.20**	92.44
Ours (Advanced Pipeline)	✓	✓	92.48	**82.37**	**90.40**	**91.70**	**92.29**	92.47

Table 9. Comparison of our algorithm with other existing SNN conversion works on CIFAR10 and CIFAR100. *Use BN* means use BN layers to optimize ANN, *Convert AP* means use Conv layers to replace Average Pooling layers. * denotes self-implementation results.