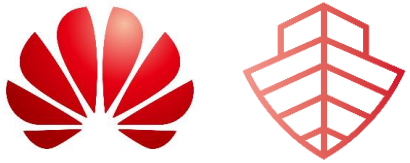
Winograd Algorithm for AdderNet

ICML 2021, Short talk

Wenshuo Li, Hanting Chen, Mingqiang Huang, Xinghao Chen, Chunjing Xu, Yunhe Wang



Huawei Noah's Ark Lab

PKU

SIAT

AdderNet

Forward:

$$Y(m,n,t) = -\sum_{i,j,k} |F(i,j,k,t) - X(m+i,n+j,k)|,$$

Backward:

$$\frac{\partial Y(m,n,t)}{\partial F(i,j,k,t)} = X(m+i,n+j,k) - F(i,j,k,t), \quad (2)$$

$$\frac{\partial Y(m,n,t)}{\partial X(m+i,n+j,k)} = HT(F(i,j,k,t) - X(m+i,n+j,k)), \quad (3)$$

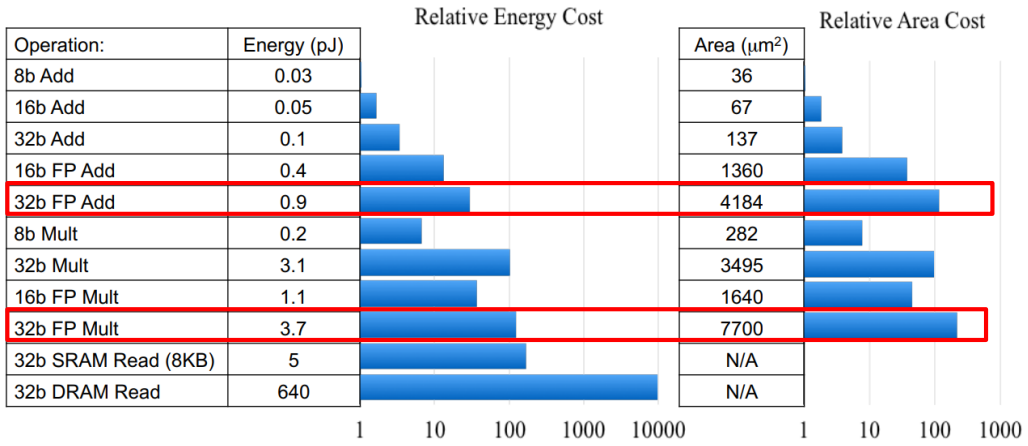
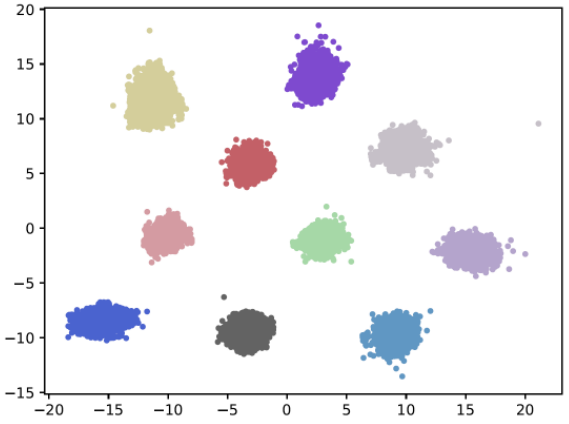
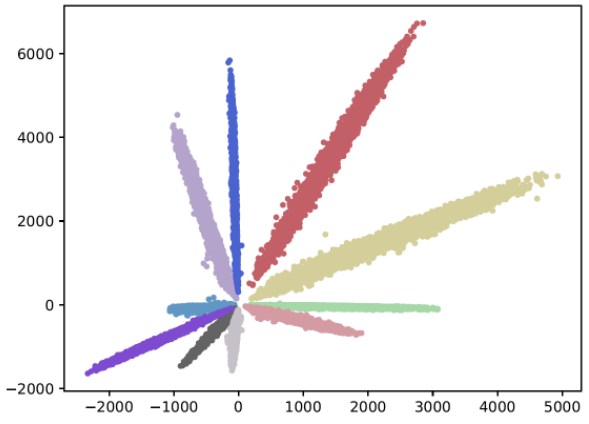


Table 2. Classification results on the CIFAR-10 and CIFAR-100 datasets.

Model	Method	#Mul.	#Add.	XNOR	CIFAR-10	CIFAR-100
VGG-small	BNN	0	0.65G	0.65G	89.80%	65.41%
	AddNN	0	1.30G	0	93.72%	72.64%
	CNN	0.65G	0.65G	0	93.80%	72.73%
ResNet-20	BNN	0	41.17M	41.17M	84.87%	54.14%
	AddNN	0	82.34M	0	91.84%	67.60%
	CNN	41.17M	41.17M	0	92.25%	68.14%
ResNet-32	BNN	0	69.12M	69.12M	86.74%	56.21%
	AddNN	0	138.24M	0	93.01%	69.02%
	CNN	69.12M	69.12M	0	93.29%	69.74%



(a) Visualization of features in AdderNets



(b) Visualization of features in CNNs

[1] Chen H, Wang Y, Xu C, et al. AdderNet: Do we really need multiplications in deep learning?[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 1468-1477.

[2] http://media.nips.cc/Conferences/2015/tutorialslides/Dally-NIPS-Tutorial-2015.pdf

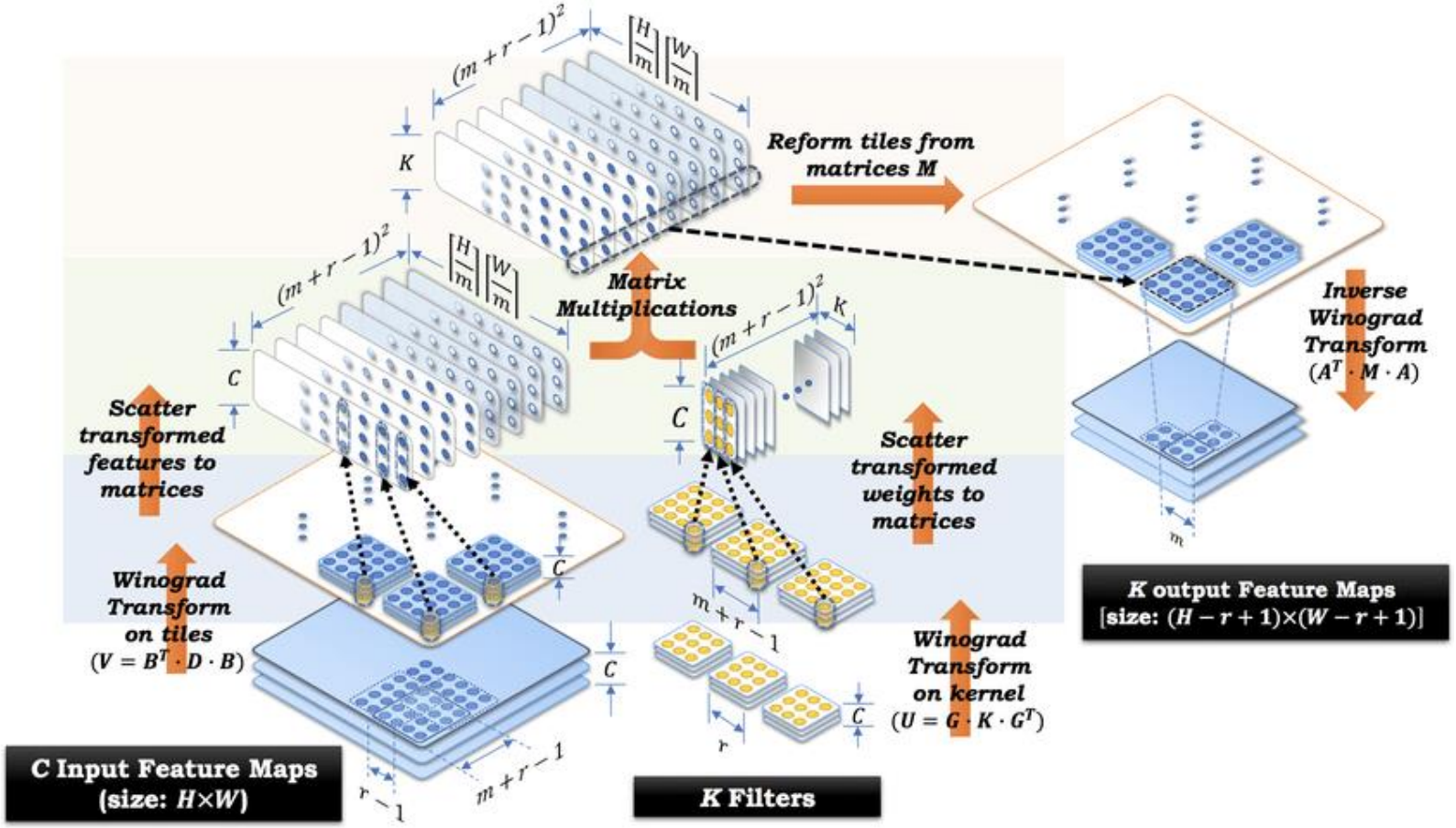
Winograd Algorithm

$$Y = A^T \left[[GgG^T] \odot [B^T dB] \right] A$$

$$B^T = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$A^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & -1 \end{bmatrix}$$



fp16

N	cuDNN msec	cuDNN TFLOPS	F(2x2,3x3) msec	F(2x2,3x3) TFLOPS	Speedup
1	12.52	3.12	5.55	7.03	2.26X
2	20.36	3.83	9.89	7.89	2.06X
4	104.70	1.49	17.72	8.81	5.91X
8	241.21	1.29	33.11	9.43	7.28X
16	203.09	3.07	65.79	9.49	3.09X
32	237.05	5.27	132.36	9.43	1.79X
64	394.05	6.34	266.48	9.37	1.48X

fp32

N	cuDNN msec	cuDNN TFLOPS	F(2x2,3x3) msec	F(2x2,3x3) TFLOPS	Speedup
1	14.58	2.68	5.53	7.06	2.64X
2	20.94	3.73	9.83	7.94	2.13X
4	104.19	1.50	17.50	8.92	5.95X
8	241.87	1.29	32.61	9.57	7.42X
16	204.01	3.06	62.93	9.92	3.24X
32	236.13	5.29	123.12	10.14	1.92X
64	395.93	6.31	242.98	10.28	1.63X

[1] Lavin A, Gray S. Fast algorithms for convolutional neural networks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016: 4013-4021.
[2] Shi F, Li H, Gao Y, et al. Sparse Winograd Convolutional neural networks on small-scale systolic arrays[J]. arXiv preprint arXiv:1810.01973, 2018.

Winograd Algorithm for AdderNet

Difficulties: the distributive law in multiplication is not valid for the l1-norm

In Multiplication

$$F(2,3) = \begin{bmatrix} d_0 & d_1 & d_2 \\ d_1 & d_2 & d_3 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} m_1 + m_2 + m_3 \\ m_2 - m_3 - m_4 \end{bmatrix}$$

$$m_1 = (d_0 - d_2)g_0 \qquad m_2 = (d_1 + d_2)\frac{g_0 + g_1 + g_2}{2}$$
$$m_4 = (d_1 - d_3)g_2 \qquad m_3 = (d_2 - d_1)\frac{g_0 - g_1 + g_2}{2}$$

In Adder Op

$$m_1 = -|(d_0 - d_2) - g_0| \neq -|d_0 - g_0| + |d_2 - g_0|$$
$$m_1 + m_2 + m_3 \neq -|d_0 - g_0| - |d_1 - g_1| - |d_2 - g_2|$$

$$Y = A^T[\hat{g} \odot [B^T dB]]A.$$

$$\Downarrow$$

$$Y = A^T[-|\hat{g} \ominus [B^T dB]|]A.$$

$$\hat{g} = GgG^T$$

Can we directly optimize $\hat{g}$?

Winograd Algorithm for AdderNet

Optimal Transform Matrix: solve the feature unbalanced problem

$$B^T = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$A^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & -1 \end{bmatrix}$$

⟹ ?

Let $X = -|\hat{g} \ominus [B^T dB]|$

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix}, Y = \begin{bmatrix} y_0 & y_1 \\ y_2 & y_3 \end{bmatrix}$$

We expand the equation $Y = A^T X A$ and get

$$y_0 = x_0 + x_1 + x_2 + x_4 + x_5 + x_6 + x_8 + x_9 + x_{10}, \quad 9 +$$

$$y_1 = x_1 - x_2 - x_3 + x_5 - x_6 - x_7 + x_9 - x_{10} - x_{11}, \quad 3 + 6 -$$

$$y_2 = x_4 + x_5 + x_6 - x_8 - x_9 - x_{10} - x_{12} - x_{13} - x_{14}, \quad 3 + 6 -$$

$$y_3 = x_5 - x_6 - x_7 - x_9 + x_{10} + x_{11} - x_{13} + x_{14} + x_{15}. \quad 5 + 4 -$$

Results of $-|\cdot \ominus \cdot|$ operation are all positive

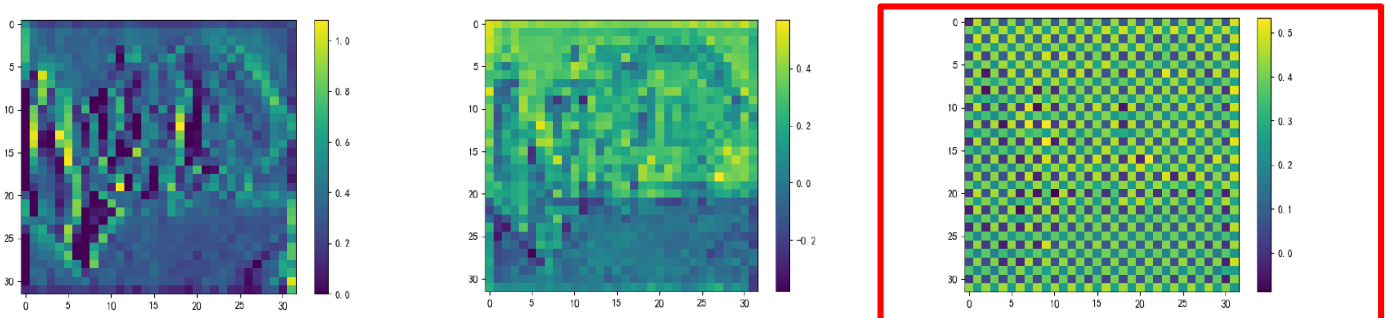
Winograd Algorithm for AdderNet

Optimal Transform Matrix: solve the feature unbalanced problem

$$A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & -1 \\ 0 & -1 \end{bmatrix} \Longrightarrow A_0^T = \begin{bmatrix} -1 & 1 & 1 & 0 \\ 0 & 1 & -1 & 1 \end{bmatrix}$$

$$y_0 = x_0 - x_1 - x_2 - x_4 + x_5 + x_6 - x_8 + x_9 + x_{10}$$
$$y_1 = -x_1 + x_2 - x_3 + x_5 - x_6 + x_7 + x_9 - x_{10} + x_{11}$$
$$y_2 = -x_4 + x_5 + x_6 + x_8 - x_9 - x_{10} - x_{12} + x_{13} + x_{14}$$
$$y_3 = x_5 - x_6 + x_7 - x_9 + x_{10} - x_{11} + x_{13} - x_{14} + x_{15}$$

5 + 4 -



(a) Input features (b) Output features with modified A (c) Output features with original A

Effects of modified A

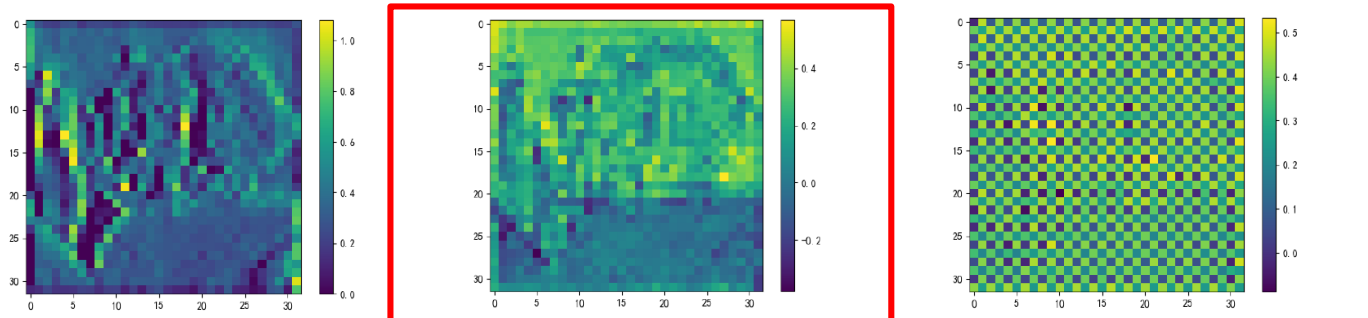
Winograd Algorithm for AdderNet

Optimal Transform Matrix: solve the feature unbalanced problem

$$A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & -1 \\ 0 & -1 \end{bmatrix} \Longrightarrow A_0^T = \begin{bmatrix} -1 & 1 & 1 & 0 \\ 0 & 1 & -1 & 1 \end{bmatrix}$$

$$y_0 = x_0 - x_1 - x_2 - x_4 + x_5 + x_6 - x_8 + x_9 + x_{10}$$
$$y_1 = -x_1 + x_2 - x_3 + x_5 - x_6 + x_7 + x_9 - x_{10} + x_{11}$$
$$y_2 = -x_4 + x_5 + x_6 + x_8 - x_9 - x_{10} - x_{12} + x_{13} + x_{14}$$
$$y_3 = x_5 - x_6 + x_7 - x_9 + x_{10} - x_{11} + x_{13} - x_{14} + x_{15}$$

5 + 4 -



(a) Input features (b) Output features with modified A (c) Output features with original A

Effects of modified A

Winograd Algorithm for AdderNet

L2-to-L1 training: mitigate the gap of Winograd AdderNet and original AdderNet

$$Y = A^T[-([GgG^T] \ominus [B^T dB])^2]A$$
$$= A^T[(2[GgG^T] \odot [B^T dB] \ominus [GgG^T]^2 \ominus [B^T dB]^2)]A$$

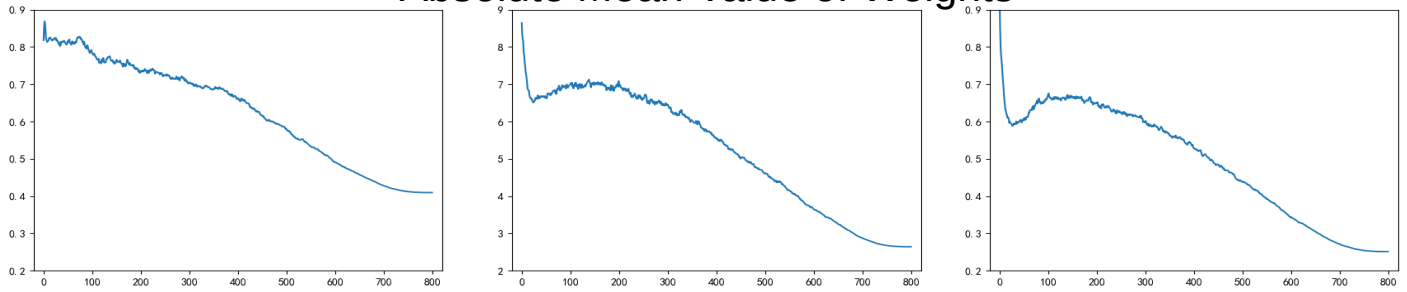
$$t = F(i,j,k,t) - X(m+i, n+j, k).$$

$$Y(m,n,t) = -\sum_{i,j,k}(|t|)^p.$$

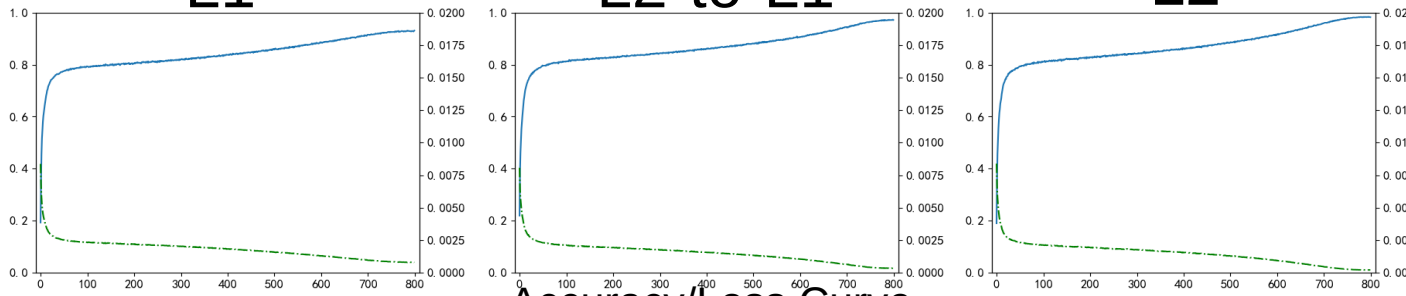
$$\frac{\partial Y(m,n,t)}{\partial X(m+i,n+j,k)} = p \cdot t^{p-1} \cdot sign(t).$$

$$\frac{\partial Y(m,n,t)}{\partial F(i,j,k,t)} = p \cdot (-t)^{p-1} \cdot sign(-t).$$

Absolute Mean Value of Weights



L1 L2-to-L1 L2



Accuracy/Loss Curve

L2-to-L1 training have similar weights distribution to L2 trainir

L2 and L2-to-L1 training have higher accuracy and lower loss

Experiments

Table 1. Results on CIFAR-10 and CIFAR-100 datasets

Model	Method	#Mul	#Add	CIFAR-10 Accuracy	CIFAR-100 Accuracy
ResNet-20	Winograd CNN	19.40M	19.84M	92.25%	68.14%
	AdderNet	-	80.74M	91.84%	67.60%
	Winograd AdderNet	-	39.24M	91.56%	67.96%
ResNet-32	Winograd CNN	31.98M	32.74M	93.29%	69.74%
	AdderNet	-	137.36M	93.01%	69.02%
	Winograd AdderNet	-	64.72M	92.34%	69.87%

Table 2. Results of Winograd AdderNet ResNet-18 with different training epochs on ImageNet dataset

Accuracy	Top-1	Top-5
150 epochs	66.2%	86.8%
250 epochs	66.5%	87.0%

Comparable accuracy on ImageNet
(Top-1 accuracy of AdderNet is 67.0%)

Table 5. Ablation Study on Our Proposed Methods

Mod A	ℓ_2-to-ℓ_1-norm	CIFAR-10	CIFAR-100
		83.87%	54.72%
	√	88.25%	62.00%
√		89.25%	62.83%
√	√	**91.56%**	**67.96%**

Complexity / Energy Consuming

The total additions of original AdderNet are

$$N \times X_h \times X_w \times C_{in} \times C_{out} \times 9 \times 2.$$

Only requires about 4/9 additions of original AdderNet

The total additions of Winograd AdderNet are

$$N \times \frac{X_h}{2} \times \frac{X_w}{2} \times (C_{out} \times C_{in} \times 16 \times 2 + C_{in} \times 3 + C_{out} \times 8) \approx N \times X_h \times X_w \times C_{out} \times C_{in} \times 8.$$

Table 2. FPGA Simulation Results of original AdderNet and Winograd AdderNet

Method	Module	#cycle	Hardware Resource	Total Energy Consuming (Equivalent)[1]
original AdderNet	total	7062	7130	50.4M
Winograd AdderNet	padding	900	31	0.03M
	input transform	3136	433	1.36M
	calculation	3140	6900	21.7M
	output transform	3136	309	0.97M
	total	-	7673	**24.0M**

[1] Since the ratio of hardware resource usage is close to 100%, we use the hardware resource overhead to approximate equivalent power consumption.

Thanks

liwenshuo@huawei.com