A Survey of Gradient Estimators for Binary Neural Networks for Image Classification

Haley So (haleyso@stanford.edu)
Stanford University

Motivation

- Convolutional Neural Networks (CNNs) are widely used for imaging and computer vision tasks. However, the state-of-the-art networks require ample memory and compute power.
- Recently, the emergence of new sensors with increased circuitry per pixel allows for on focal plane computation. But, with limited memory capacity, such large CNNs cannot run on the sensor.
- We want to understand how to train lightweight, binarized networks for on sensor computation and understand the tradeoff between memory and precision. However, gradient estimation is a key problem.

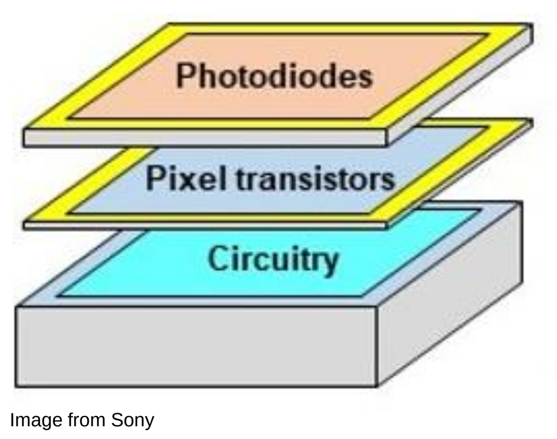
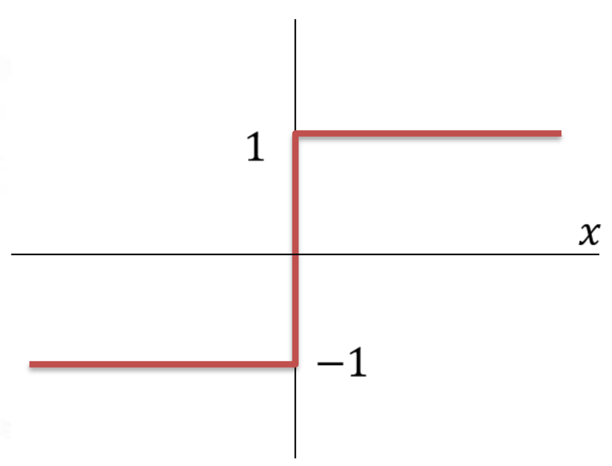
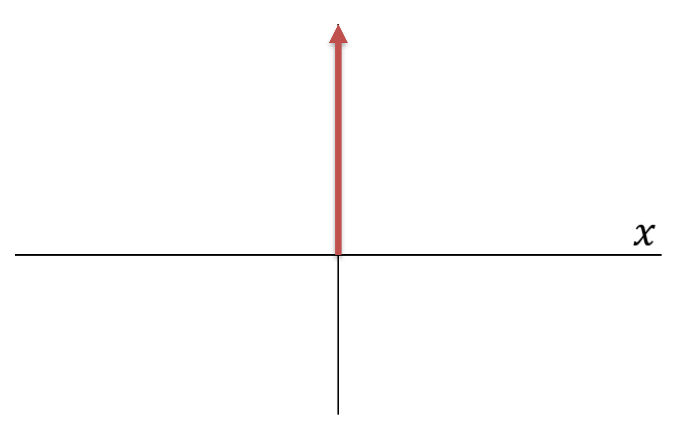


Image from Sony

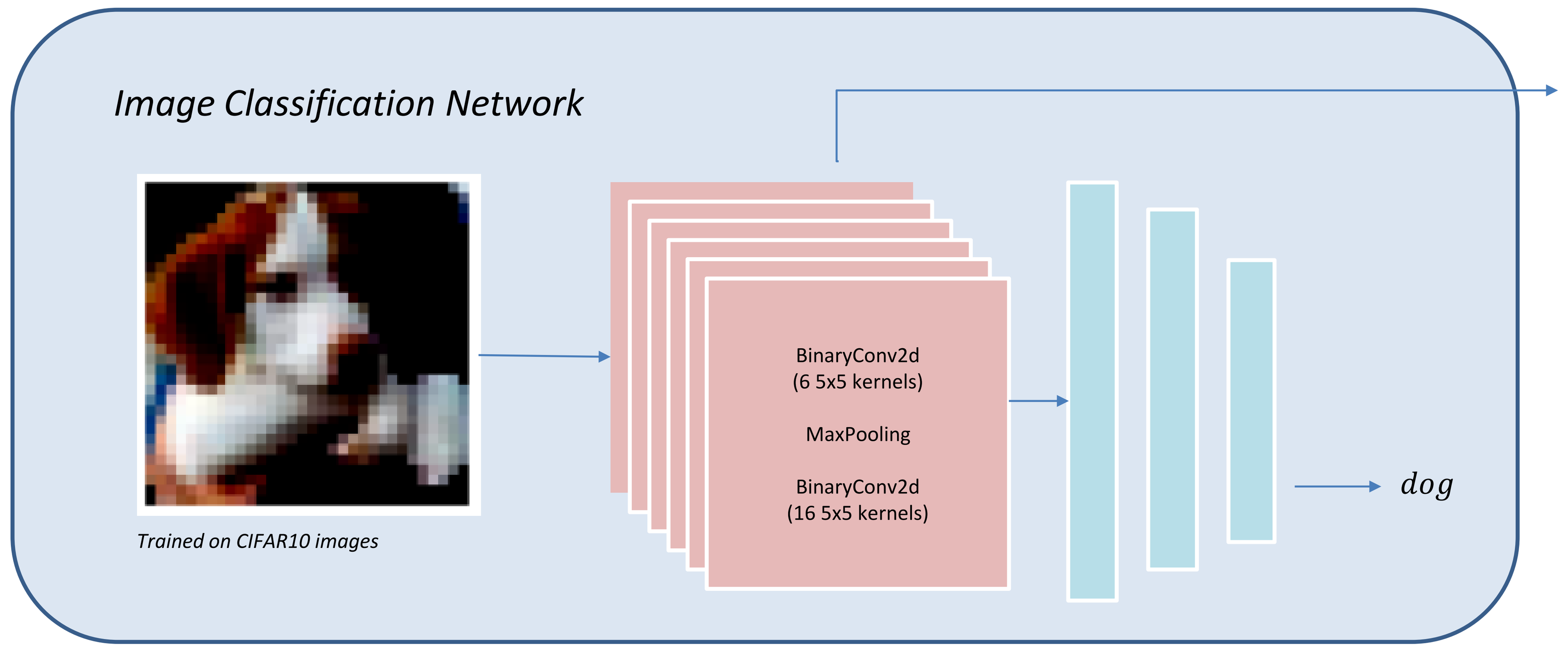


$sign(x)$



$\frac{\delta}{\delta x} sign(x)$

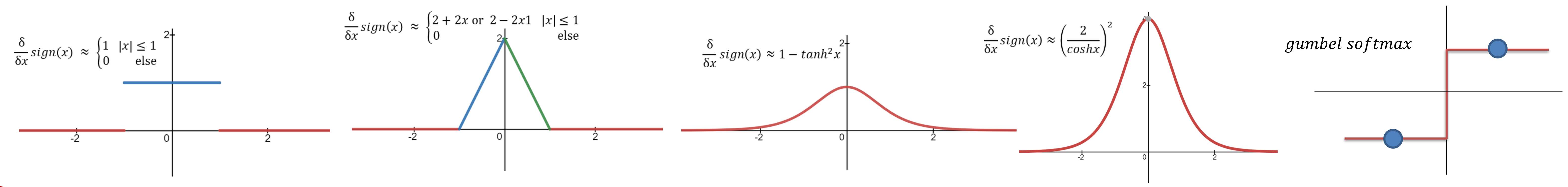
Gradient Estimation




```python
class BinarizeFunction(Function):
    @staticmethod
    def forward(ctx, input):
        ctx.save_for_backward(input)
        out = torch.sign(input)
        return out

    @staticmethod
    def backward(ctx, grad_output):
        input, = ctx.saved_tensors
        grad_input = gradientEstimation(input)
        return grad_output * grad_input
```

Gradient Estimation Methods:



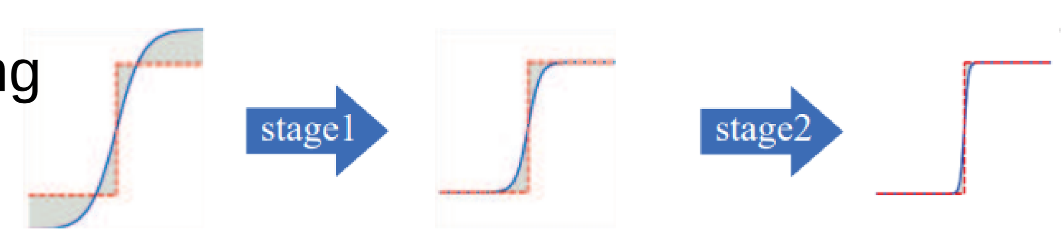
Related Work

XNOR-net: ImageNet Classification Using Binary Convolutional Neural Networks

- Representation of bits as -1,1
- Convolutions become XNOR and bit-count operations

Forward and Backward Information Retention for Accurate Binary Neural Networks

- Slowly quantize with training



References

[1] Rastegari, Ordonez, Redmon, and Farhadi, XNOR-net: ImageNet Classification Using Binary Convolutional Neural Networks, ECCV, 2016
[2] Liu, Wu, Luo,Yang, Liu, and Cheng, Bi-Real Net: Enhancing the Performance of 1-bit CNNs With Improved Representational Capability and Advanced Training Algorithm, ECCV, 2018
[3] Qin, Gong, Liu, Wei, Yu, and Song, Forward and Backward Information Retention for Highly Accurate Binary Neural Networks, CVPR, 2020

Experimental Results

Gradient Estimator/Model	Weights Precision	Convolutional layers parameter size	Precision top-1 (without bias)
None (Full Precision) / Base Model (BM)	32 bit	91,200 bits	75.22
Naïve: Quantized Weights at the End / BM	1 bit	2850 bits	11.86
Straight Through Estimator (STE) / BM	1 bit	2850 bits	59.98
Second Order Approximation / BM	1 bit	2850 bits	9.996
Tanh estimator / BM	1 bit	2850 bits	59.44
2/coshx estimator / BM	1 bit	2850 bits	58.10
Gumbel Softmax /BM	1 bit	2850 bits	56.64

(Reported percentages are the best trained with parameter tuning. IR-Net, BiRealNet ResNet50 were also trained with 87.6% and 83.9% accuracy but they had very specific architectures and training procedures. This table only has comparable architectures to show the effect of the gradient estimator.)

Memory Footprint Comparison:
- 2 full precision conv2d layers = 91,200 bits
- To the base model shown above, we could add 98 more binaryConv2d(in=6, out=6, kernel=5) layers (each is 900bits) for the same memory footprint

Model (changes)	Conv2d Parameters	Precision top-1
STE (+50 conv2d layers)	47850 bits	10.00
STE (+10 conv2d layers)	11850 bits	9.986
STE (+1 conv2d layer)	3750 bits	32.72
STE (kernel size=3)	1026 bits	54.17

Discussion:
- Many of these gradient approximations are comparable, but there is still a gap between the full precision and binarized network performances
- Binary filters may not be able to adequately capture features
- Without going to better architectures, increasing the depth of the network does not necessarily help. Training gets more challenging as depth increases so hyper-parameter tuning becomes really important.

Future Directions:
- Instead of the extreme (binarization), we can try highly quantized (eg. 8-bits)
- Create better architectures in tandem with gradient estimation methods