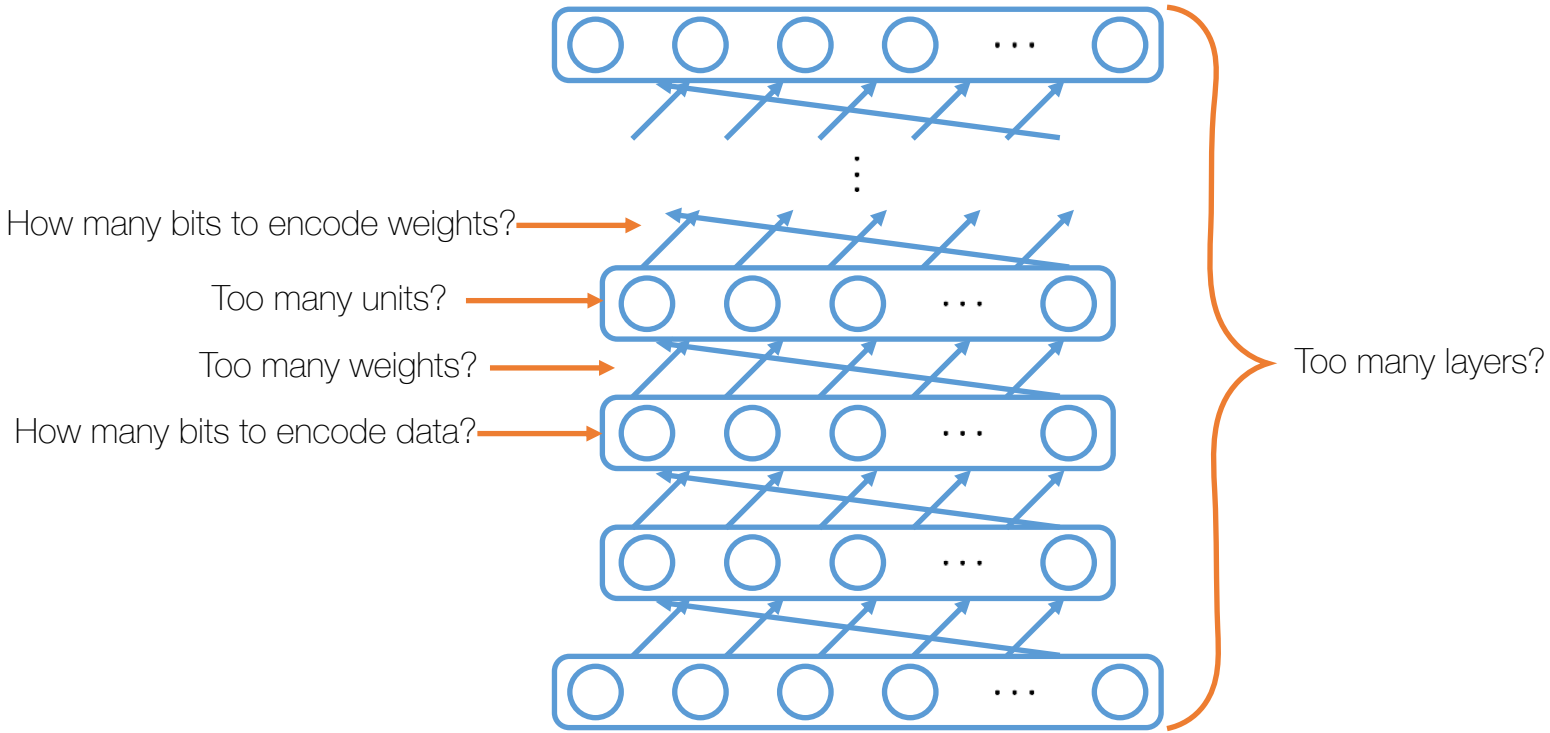
ENGR-E 533 “Deep Learning Systems”
Lecture 09: Network Compression

Minje Kim
Department of Intelligent Systems Engineering
Email: minje@indiana.edu
Website: http://minjekim.com
Research Group: http://saige.sice.indiana.edu
Meeting Request: http://doodle.com/minje

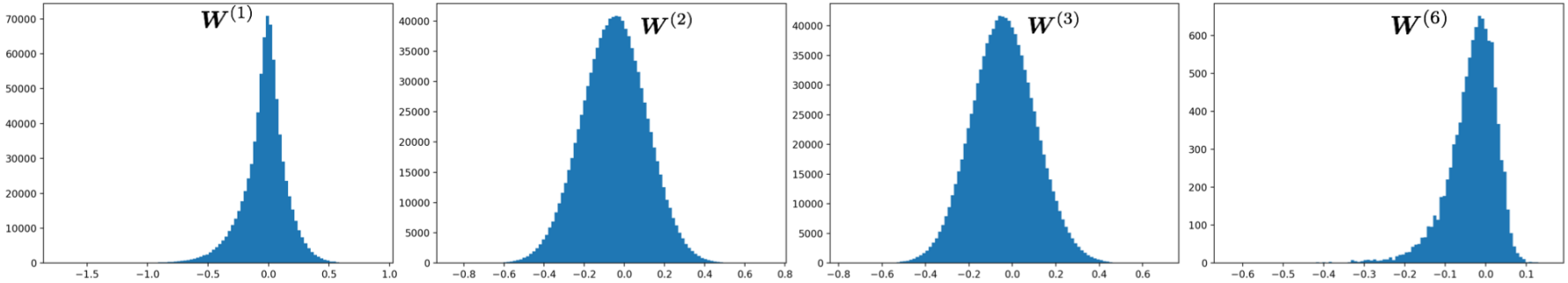
INDIANA UNIVERSITY
SCHOOL OF INFORMATICS,
COMPUTING, AND ENGINEERING

Different Ways to Compress a Network



DNNs Are Redundant

- There are many small weights
- 1024X5 FC for MNIST

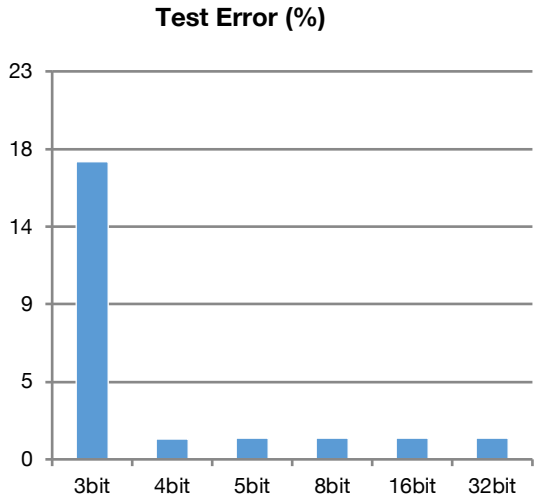


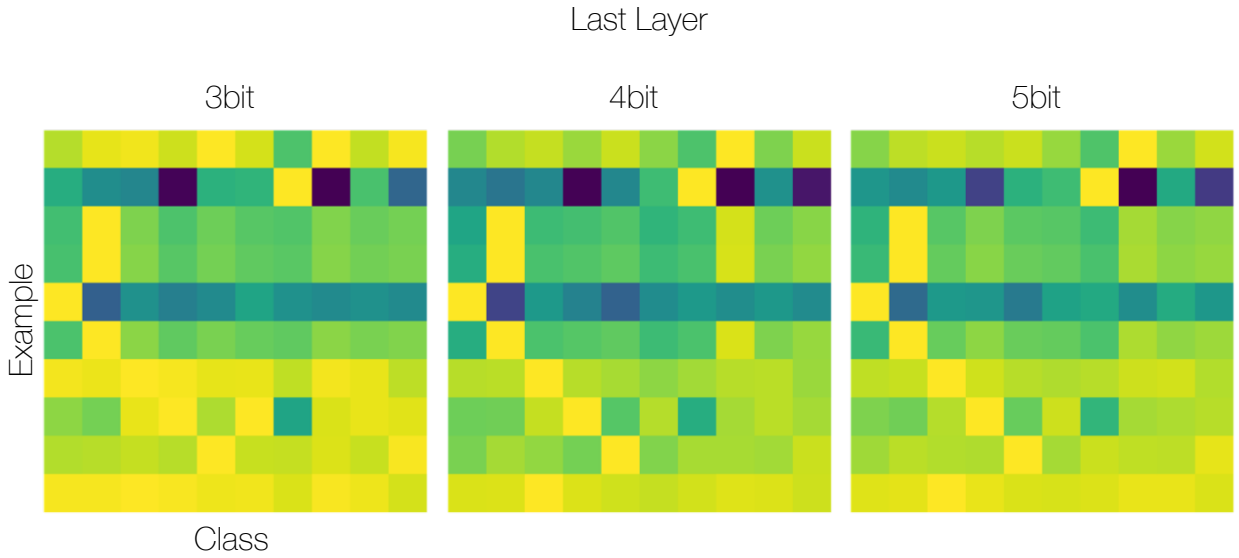
- Their contribution is less than the larger weights
  - 1024X5 FC net for MNIST
    - Test error: 1.25%
  - Kill all weights larger than -0.1 and smaller than 0.1 (from 1st to 5th layer)
    - Test error: 1.64%
    - How much do we save?: 2.5M versus 5M
- Is there any better way?

DNNs Are Redundant

- Are they real?

- Single precision could be an overkill



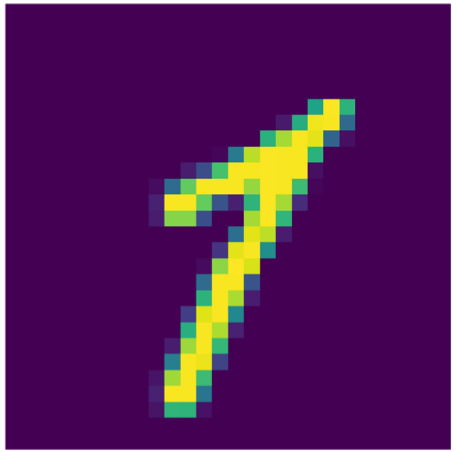


- Can we do this better?

Distilling Knowledge from Neural Networks

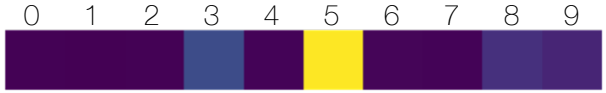
- Dark Knowledge

- A confusing “7”
  - Looks like a “1” to me, too



- A confusing “5”
  - Looks like a “3” to me, too

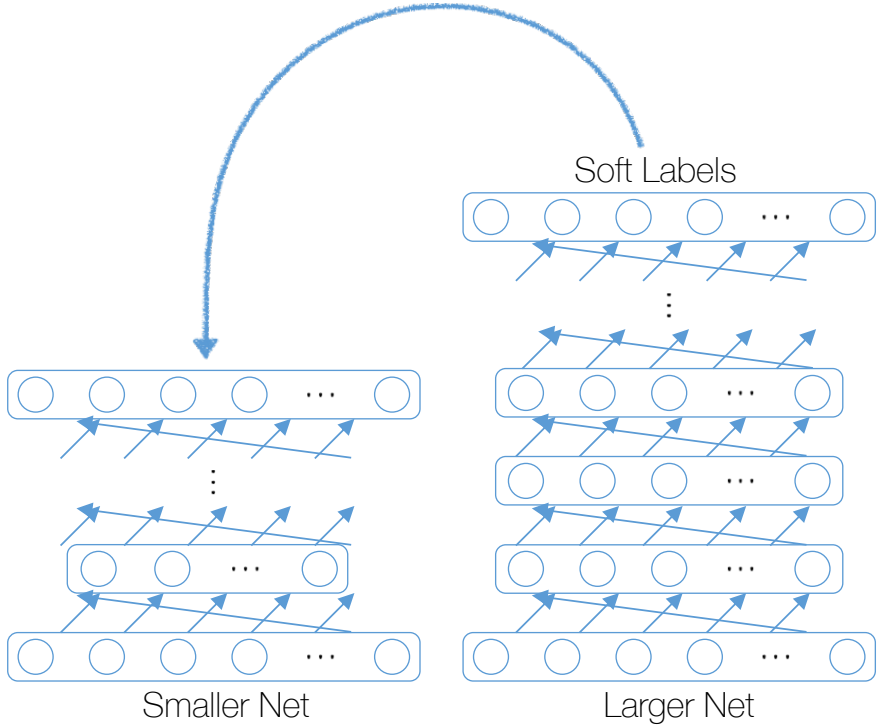
- Neural network predictions encode this information



Distilling Knowledge from Neural Networks

- Dark Knowledge

- Even a properly trained network can create some confusion between classes
  - So, the softmax output can be seen as soft labels
- Hard class labels (one-hot representation) do not contain this information
- Geoffrey Hinton thought that this soft labels mean something
  - This is so called “dark knowledge”
- To train a small network we have two choices
  - Train it using the hard labels as usual
    - The small net will eventually produce some soft labels, perhaps not good enough
  - Train it using the soft labels got from another network
- Another network?
  - A larger and deeper network—which can effectively encode the “dark knowledge”
  - An ensemble of large networks (can be very complex)



Distilling Knowledge from Neural Networks

- Dark Knowledge

- MNIST
  - The large net
    - 1200X2, ReLU
    - Dropout, data augmentation (jittering)
    - Test error: 0.67%
  - The small net
    - 800X2 ReLU
    - No regularization
    - Test error: 1.46%
  - The small distilled net (full dataset)
    - With soft labels from the large net
    - Test error: 0.74%
  - The small distilled net (no “3”)
    - With soft labels from the large net
    - Test error: 1.09% (0.14% of them are from “3”)
- Ensemble for automatic speech recognition
  - Baseline
    - 2560X8, ReLU
    - Test error: 58.9%
  - Ensemble (10 instances)
    - Test error: 61.1%
  - Distilled single model
    - Test error: 60.8%

Distilling Knowledge from Neural Networks

- Dark Knowledge

- The distilled model: softmax with temperature

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

Large temperature can smear the distribution

Small probabilities are boosted

- The large model: softmax with temperature

$$p_i = \frac{\exp(v_i/T)}{\sum_j \exp(v_j/T)}$$

- The BP error in the last year

$$\frac{\partial \mathcal{E}}{\partial z_i} = \frac{1}{T}(q_i - p_i)$$

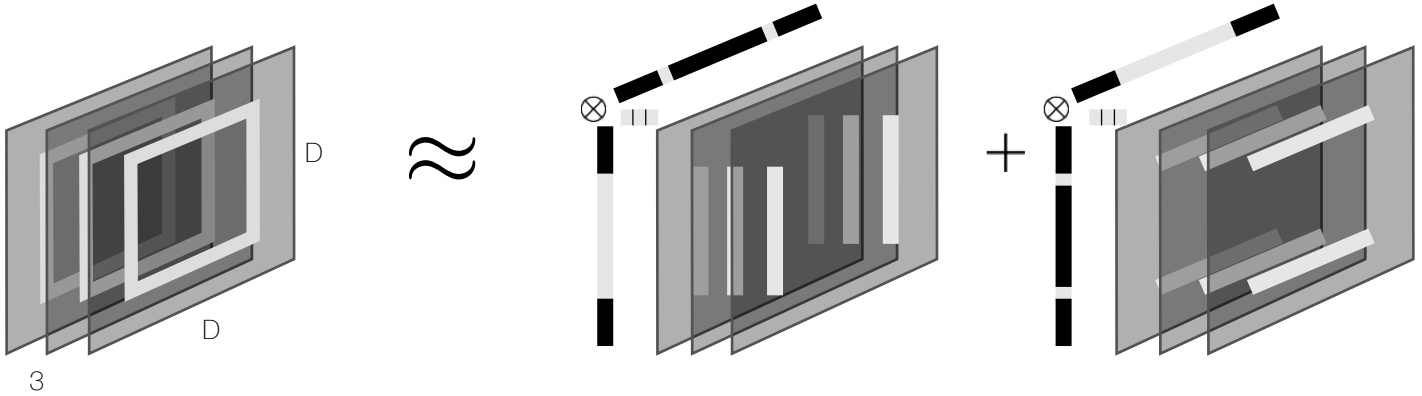
- You can mix up the soft target and the hard label, too

$$\frac{\partial \mathcal{E}}{\partial z_i} = \frac{1}{T}\Big(q_i - \big(\alpha p_i + (1-\alpha)t_i\big)\Big)$$

Low-Rank Approximation of Weights

- Singular Value Decomposition

- The RGB color code is misleading
  - They are just intensities
  - We happen to know which channel stands for which color
  - So the gray scale images are a more accurate representation
- Sometimes I can come up with a better representation than the usual 3D tensor



  - How many parameters?
    - $3D^2$ VS $4D+6$

Denton, E. et al., "Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation," NIPS 2014

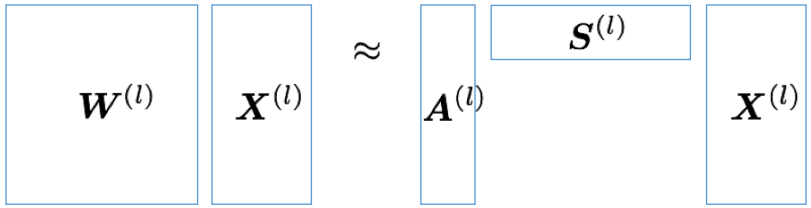
Low-Rank Approximation of Weights

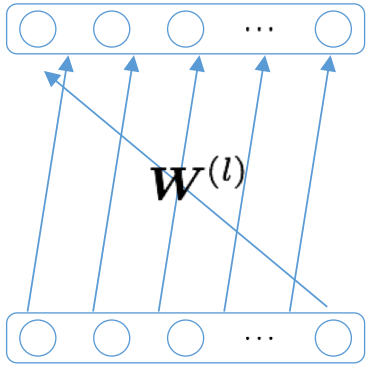
- Singular Value Decomposition

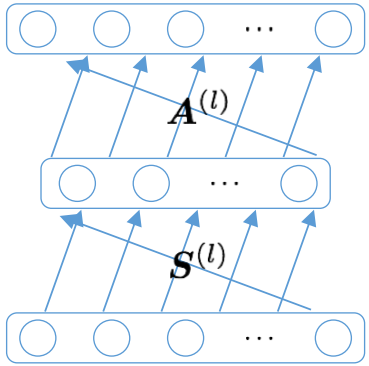
- For 2D cases (FC nets)?
- Same idea, but on a matrix not a tensor

$$W^{(l)} \approx A^{(l)} S^{(l)}$$

$$W^{(l)} X^{(l)} \approx A^{(l)} S^{(l)} X^{(l)}$$







- You can go ahead and learn those factorized weights via BP
- But, SVD is a common choice, too

Pruning

- Not all weights are important

- Imagine a CNN-based image processing filter
  - Converting gray-scale images to color images
  - Segmenting the foreground object out of backgrounds
  - This kind of functions are heavy even in large desktops
- Storage
  - AlexNet
    - >200MB
  - VGG-16
    - >500MB
- Memory access consumes power
  - 32bit floating point addition: 0.9pJ
  - 32bit SRAM cache access: 5pJ
  - 32bit DRAM access: 640pJ

S. Han et al., “Learning both Weights and Connections for Efficient Neural Networks,” NIPS 2015.

Pruning

- Not all weights are important

1. Single-step aggressive pruning
   a. First, train a regular CNN+FC net
   b. Then, kill weights that are below a threshold
   c. Do some more BP on the pruned net

- Iterative pruning
  - 1.a., then repeat 1.b. and 1.c.
- Pruning neurons
  - If a neuron doesn't have any incoming or outgoing weights, kill it
- Pruning can keep the same accuracy while with much smaller weights

	Weights	FLOP	WEIGHTS (%)	FLOP (%)
LeNet-300-100	266K	532K	8%	8%
LeNet-5	431K	4586K	8%	16%
AlexNet	61M	1.5B	11%	30%
VGG-16	138M	30.9B	7.5%	21%

AlexNet	Weights (%)	FLOP (%)
conv1	84%	84%
conv2	38%	33%
conv3	35%	18%
conv4	37%	14%
conv5	37%	14%
fc1	9%	3%
fc2	9%	3%
fc3	25%	10%
total	**11%**	**30%**

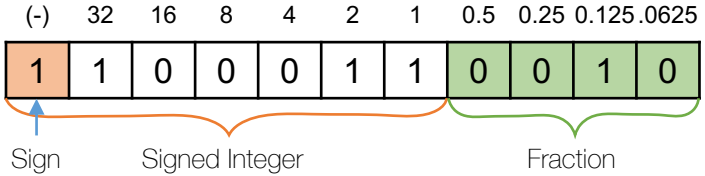
- But, much more effective in compressing the FC layers, not the convolutional layers
- Need to come up with a sparse representation that makes sense in HW

S. Han et al., "Learning both Weights and Connections for Efficient Neural Networks," NIPS 2015.

Quantization

- Control over the bit allocation

- A little bit about fixed points
  - What does this mean?
    - -35.1250
  - Why?
    - A fixed-point bit string consists of three parts: sign, integer, and fraction
  - It's called "fixed point" representation because once set the resolution doesn't change



- How to choose the right number of bits for my DNN?
  - You can take a look at the histogram of the variable of your interest and gauge the dynamic range of the values
  - For example
    - Weights: don't usually have a large value especially because of the weight decaying (to prevent overfitting)
    - Input signals: depends on the problem
    - Output: regression can be tricky (basically same with the input signals case), softmax is easy
    - Activation of a hidden unit: can be large depending on the choice of the activation function, but you don't like a too strong activation anyway
    - Input to a hidden unit: can be potentially with a large dynamic range
- From now on, I'll be naïve and assume that the integer part is with enough bits
  - I'll test on the fraction part by changing its bit length

Quantization

- Control over the bit allocation

- 3bit fractions for the weights: 17.28% test error (see slide 4)
- 3bit fractions for all values: 27.32%
- 4bit fractions for all values: 1.18%
- So, we need at least 4bit fraction?
  - Are there any better ways?
- Quantization noise injection
  - Keep the “ghost” floating-point variables
  - Feedforward and BP using quantized values
  - Update the ghost variable, not the quantized ones

Quantization

- Control over the bit allocation

- Ordinary feedforward (for a sample)

$$z^{(l)} \leftarrow W^{(l)} x^{(l)} + b^{(l)} \qquad x^{(l+1)} \leftarrow g(z^{(l)})$$

- Noisy feedforward

$$\bar{z}^{(l)} \leftarrow \bar{W}^{(l)} \bar{x}^{(l)} + \bar{b}^{(l)} \qquad \bar{x}^{(l+1)} \leftarrow g(\bar{z}^{(l)})$$

- Ordinary BP

$$\Delta^{(l)} \leftarrow \left(W^{(l+1)^\top} \Delta^{(l+1)}\right) \odot g'(Z^{(l)})$$

$$W^{(l)} \leftarrow W^{(l)} - \rho \Delta^{(l)} X^{(l)^\top}$$

- Noisy BP

$$\bar{\Delta}^{(l)} \leftarrow \left((\bar{W}^{(l+1)})^\top \bar{\Delta}^{(l+1)}\right) \odot g'(\bar{Z}^{(l)})$$

$$W^{(l)} \leftarrow W^{(l)} - \rho\left(\bar{\Delta}^{(l)} (\bar{X}^{(l)})^\top\right) \odot Q'(W^{(l)})$$

$$\bar{W}^{(l)} \leftarrow Q(W^{(l)})$$

5.61463	←	0.087312	0.193289	•	3.271394	+	5.283612	Floating-point
-35.51016		-10.90482	-2.382336		0.234808		0.723193	
5.75	←	0.125	0.25	•	3.25	+	5.25	3bit fraction
-35.25		-10.875	-2.375		0.25		0.75	

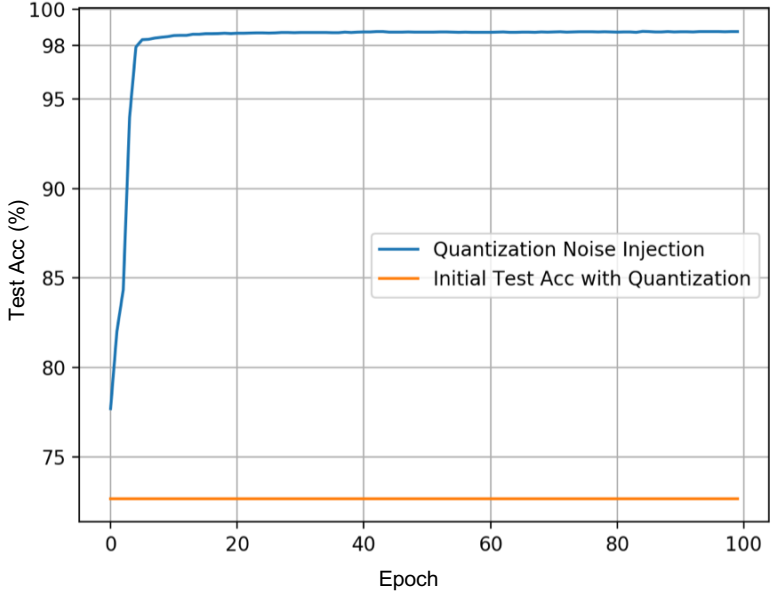
Although quantization is lossy, we'll assume that $Q'(x)$ is an identity function

- Note that updates are done on the real-valued parameters, not on the quantized ones. Why?
  - Quantized weights are not continuous. Smooth updates are possible.

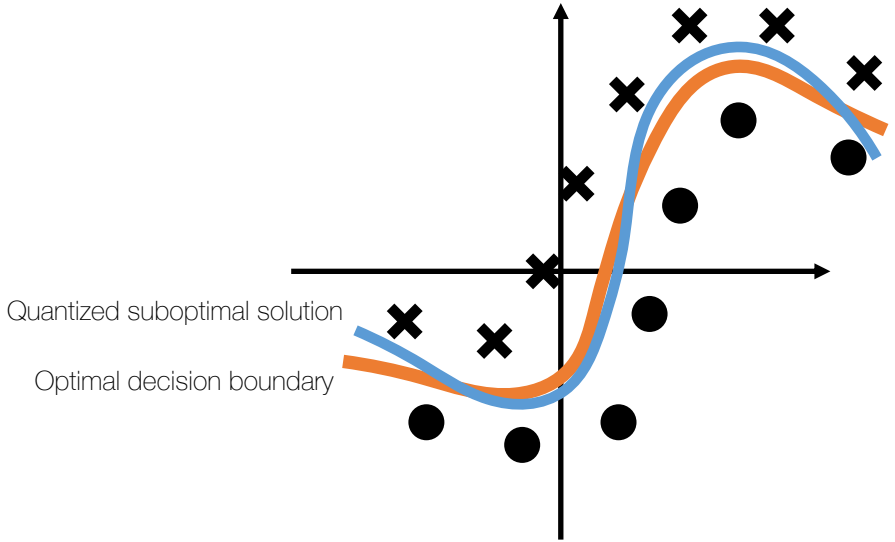
Quantization

- Control over the bit allocation

- 3bit fraction



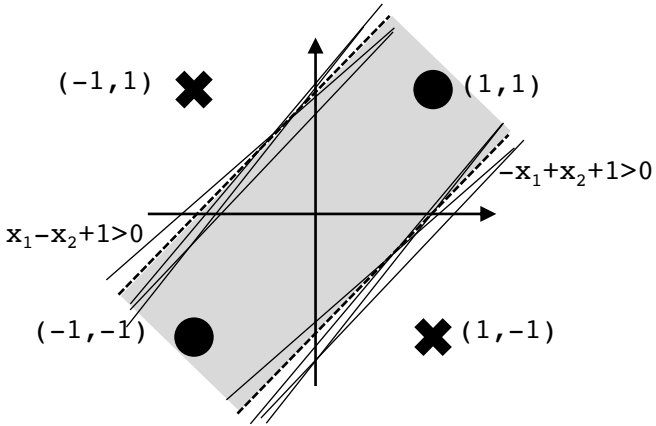
- Quantization noise can introduce more error
  - And, can lead to a suboptimal solution

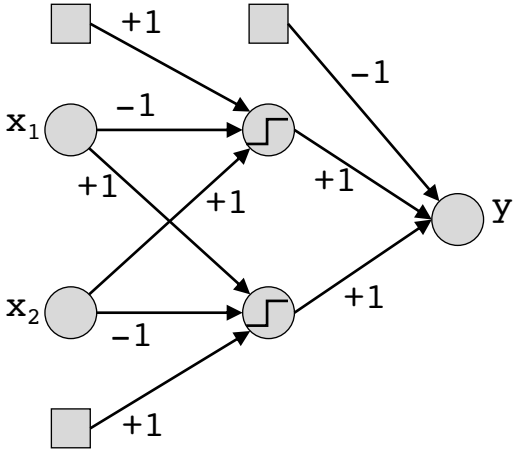


Binarization

- An extreme quantization

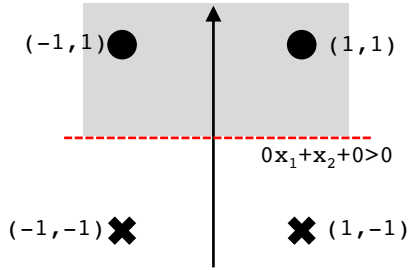
- Bitwise Neural Networks

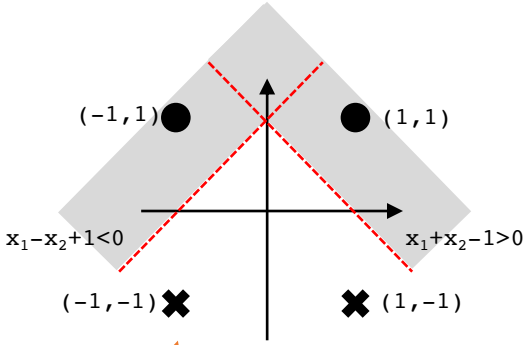


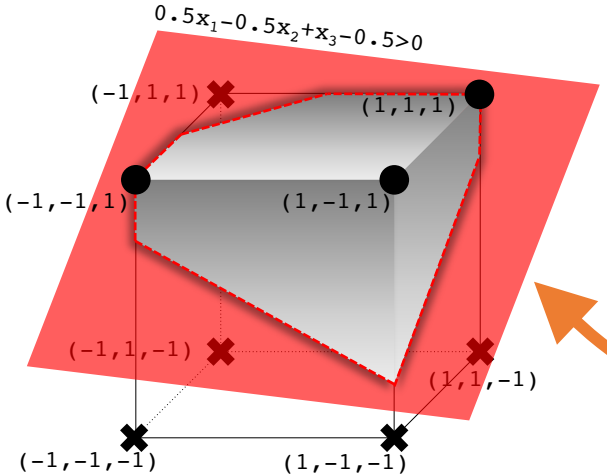


Binarization

- An extreme quantization







BNN with zero-weights solves more problems

A-linearly-separable-but-not-separable-with-BNN case

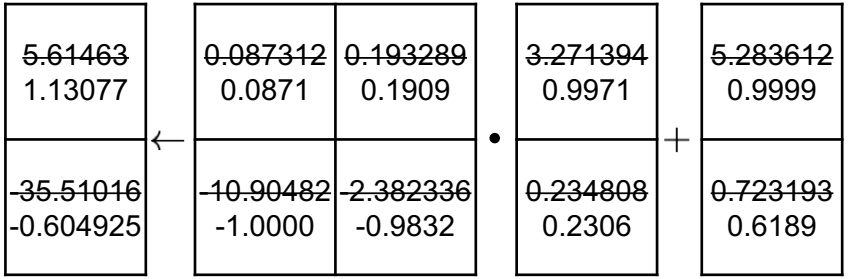
Another-linearly-separable-but-not-separable-with-BNN case

Binarization

- An extreme quantization

○ Ordinary feedforward with a tanh-compressed weights

$$\boldsymbol{z}^{(l)} \leftarrow \tanh\left(\boldsymbol{W}^{(l)}\right)\boldsymbol{x}^{(l)} + \tanh\left(\boldsymbol{b}^{(l)}\right) \qquad \boldsymbol{x}^{(l+1)} \leftarrow \tanh(\boldsymbol{z}^{(l)})$$



○ BP

$$\boldsymbol{\Delta}^{(l)} \leftarrow \left(\tanh\left(\boldsymbol{W}^{(l+1)}\right)^{\top}\boldsymbol{\Delta}^{(l+1)}\right) \odot \tanh'(\boldsymbol{Z}^{(l)})$$

$$\boldsymbol{W}^{(l)} \leftarrow \boldsymbol{W}^{(l)} - \rho\left(\boldsymbol{\Delta}^{(l)}\boldsymbol{X}^{(l)\top}\right) \odot \tanh'\left(\boldsymbol{W}^{(l)}\right)$$

Introduced by the chain rule

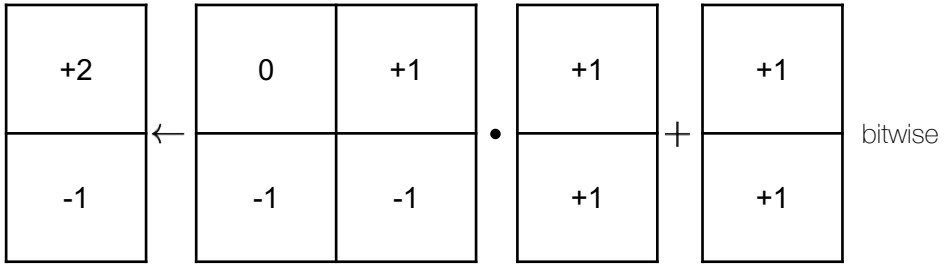
Binarization

- An extreme quantization

○ Bitwise feedforward

$$\bar{\boldsymbol{W}}^{(l)} \leftarrow \boldsymbol{M}^{(l)} \odot \text{sign}(\boldsymbol{W}^{(l)})$$

$$\bar{\boldsymbol{z}}^{(l)} \leftarrow \bar{\boldsymbol{W}}^{(l)} \bar{\boldsymbol{x}}^{(l)} + \bar{\boldsymbol{b}}^{(l)} \qquad \bar{\boldsymbol{x}}^{(l+1)} \leftarrow \text{sign}(\bar{\boldsymbol{z}}^{(l)})$$



○ Noisy BP

$$\bar{\boldsymbol{\Delta}}^{(l)} \leftarrow \left((\bar{\boldsymbol{W}}^{(l+1)})^\top \bar{\boldsymbol{\Delta}}^{(l+1)}\right) \odot \tanh'(\bar{\boldsymbol{Z}}^{(l)})$$

$$\boldsymbol{W}^{(l)} \leftarrow \boldsymbol{W}^{(l)} - \rho\left(\bar{\boldsymbol{\Delta}}^{(l)} (\bar{\boldsymbol{X}}^{(l)})^\top\right) \odot \left(\boldsymbol{M}^{(l)} \odot \tanh'\left(\boldsymbol{W}^{(l)}\right)\right)$$

BNN on Supervised Speech Denoising

- Audio Demo

	Input Noisy Speech	Deep Learning (Binary Input)	Bitwise
Female + Frogs			
Female + Ocean			
Female + Typing			
Male + Eating Chips			
Male + Jungle			

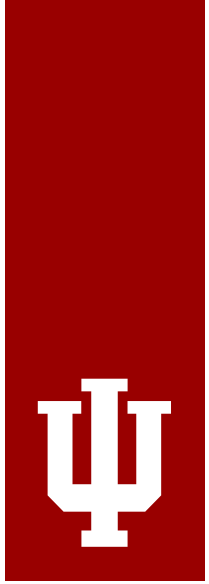
M. Kim and P. Smaragdis, "Bitwise Neural Networks for Efficient Single-Channel Source Separation," ICASSP 2018

Binarization

- An extreme quantization

- XNOR-Net
  - A bitwise version for CNNs
- Binarized Neural Networks
  - With stochastic training algorithm

M. Rastegari et al., "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," arXiv:1603.05279;
Hubara, M. et al., "Binarized neural networks," NIPS 2016



Thank You!