PRODUCT

BinaryConnect: Training Deep Neural Networks with binary weights during propagations.

Produced By Learners
Date - 27th August 2025

Contents

The goal of this report is to provide a unified list of the to opportunities to help inform the upcoming growth process.

01 The Core Problem

02 Quantization Basics

03 BinaryConnect Explained

04 Benefits - Binary Connect

05 Forward Pass

06 Backpropagation

07 Experimental Results

08 Roleplay - Academic Researcher

09 Binary Neural Networks - Results

10 Roleplay - Industry Expert

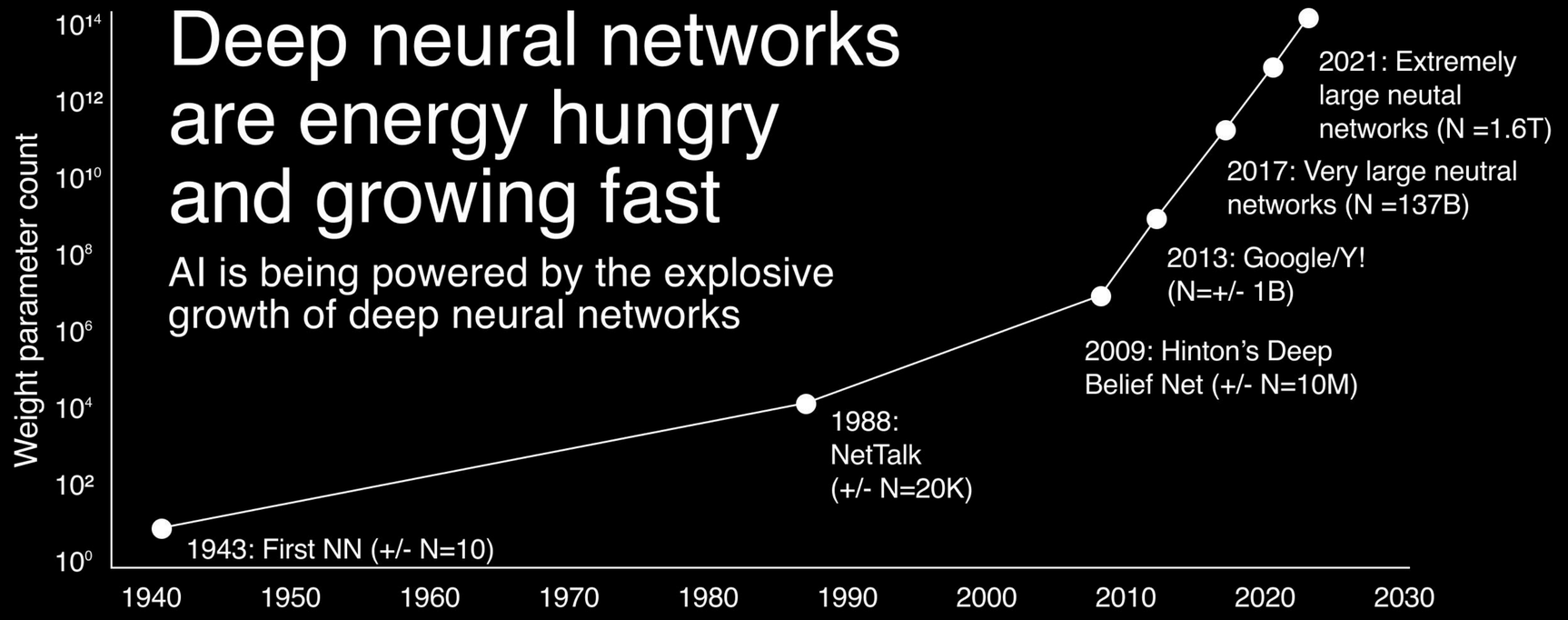
Report and Implementation

Github

Report

Deep neural networks are energy hungry and growing fast

AI is being powered by the explosive growth of deep neural networks



2025 | Increasingly large and complex neural networks for Natural language Processing, Image and Video Processing

01 The Core Problem

02 Quantization Basics

03 Binary Connect

04 Benefits

05 Forward Pass

06 Backpropagation

07 Experimental Results

08 Academic Researcher

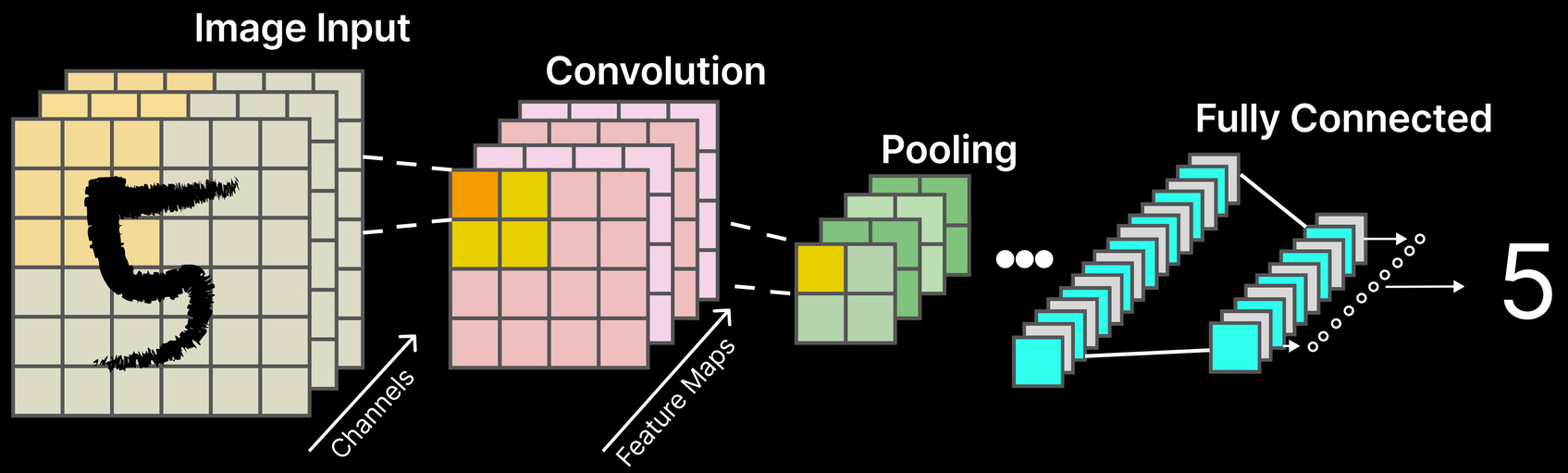
09 BNN - Results

10 Industry Expert

Motivation : NNs do a lot of number crunching

For example, Convolutional Neural Networks:

- **Many layers** (100+) for large pipelines with self-learning feature extractors
- Running time dominated by **multiply-accumulate** operations: 75-85%
- Significant **traffic** between layers: up to 200MB per layer



01 The Core Problem

02 Quantization Basics

03 Binary Connect

04 Benefits

05 Forward Pass

06 Backpropagation

07 Experimental Results

08 Academic Researcher

09 BNN - Results

10 Industry Expert

What can we do to make things better?

Several options have been researched and developed:

- Parallelization
- GPUs
- Quantization (Reduced Precision)
- Compression (pruning, trained quantization)
- Better algorithms (FFT/Winograd for convolutions)
- Special purpose hardware (ASICs, FPGAs, and others)

01 The Core Problem

02 Quantization Basics

03 Binary Connect

04 Benefits

05 Forward Pass

06 Backpropagation

07 Experimental Results

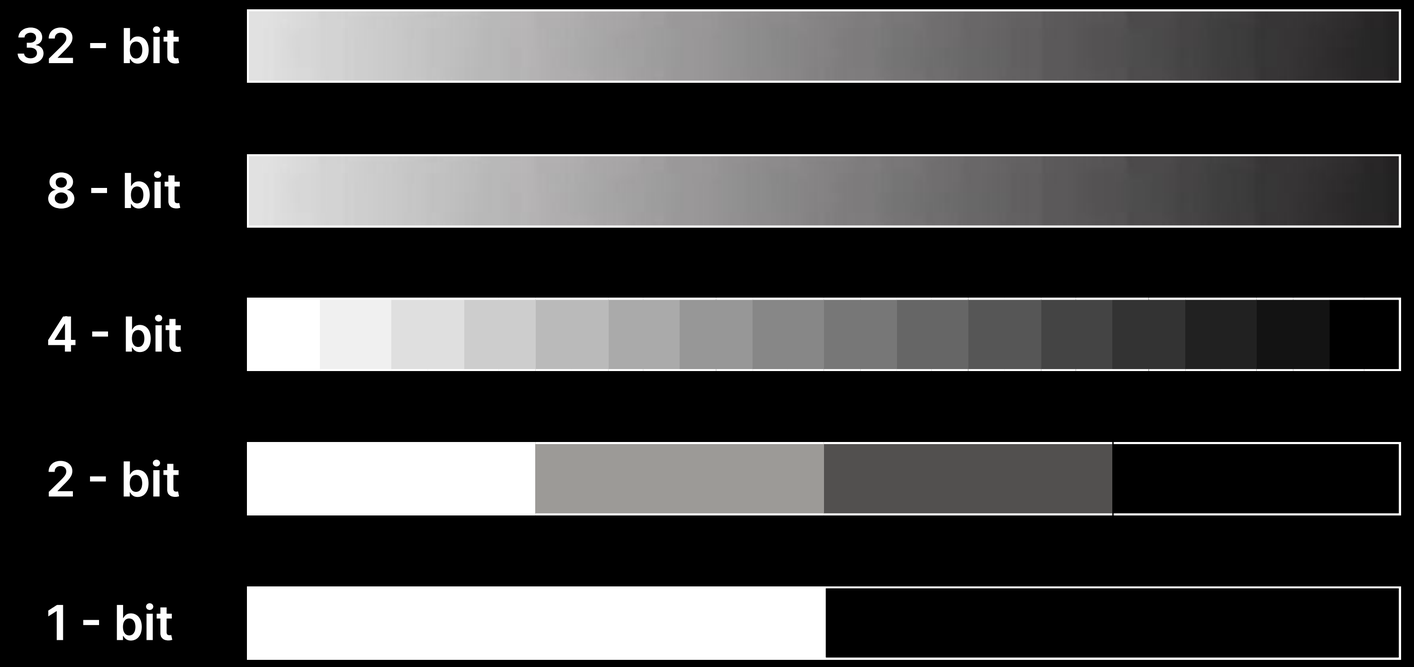
08 Academic Researcher

09 BNN - Results

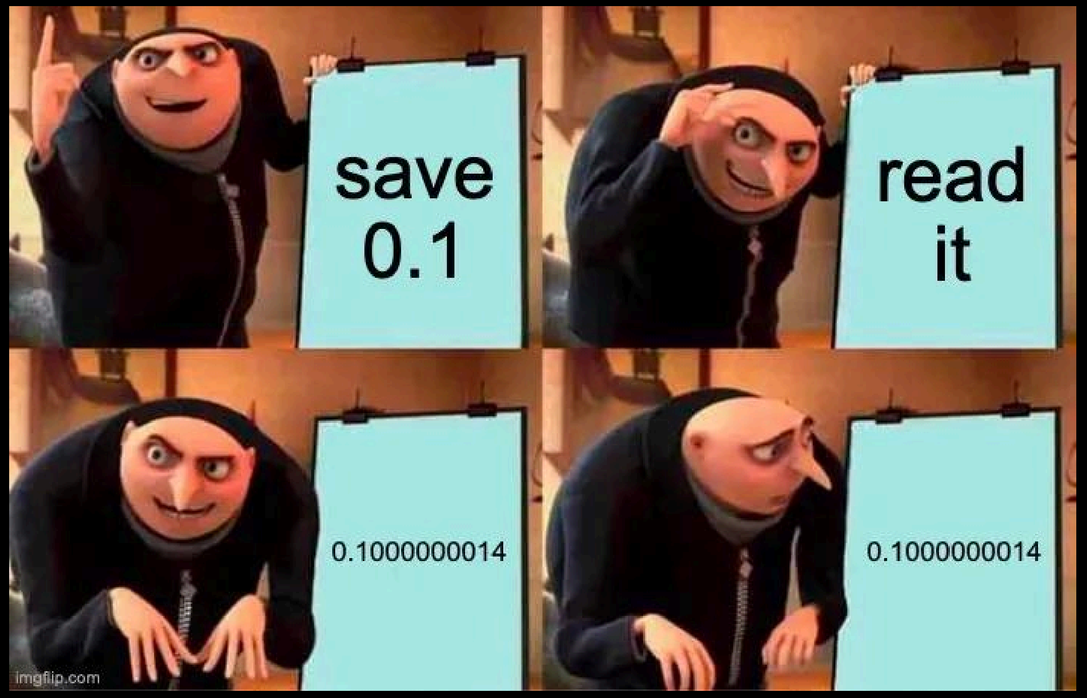
10 Industry Expert

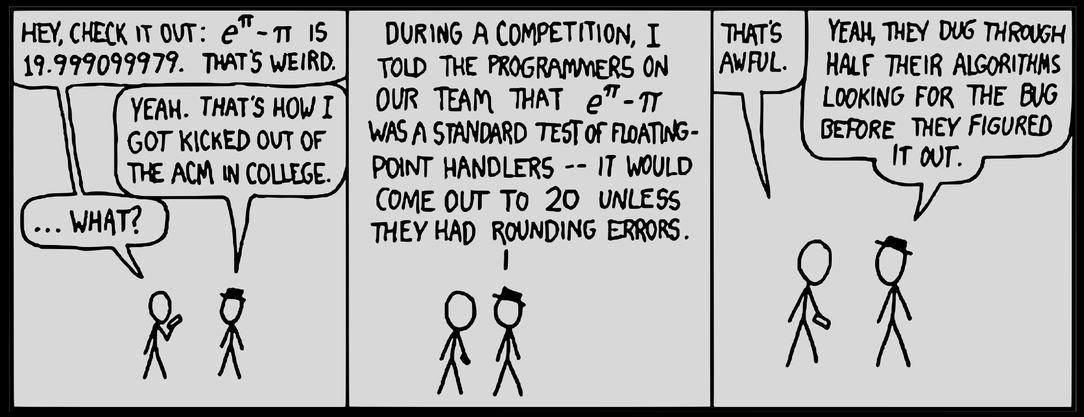
Reducing the number of bits

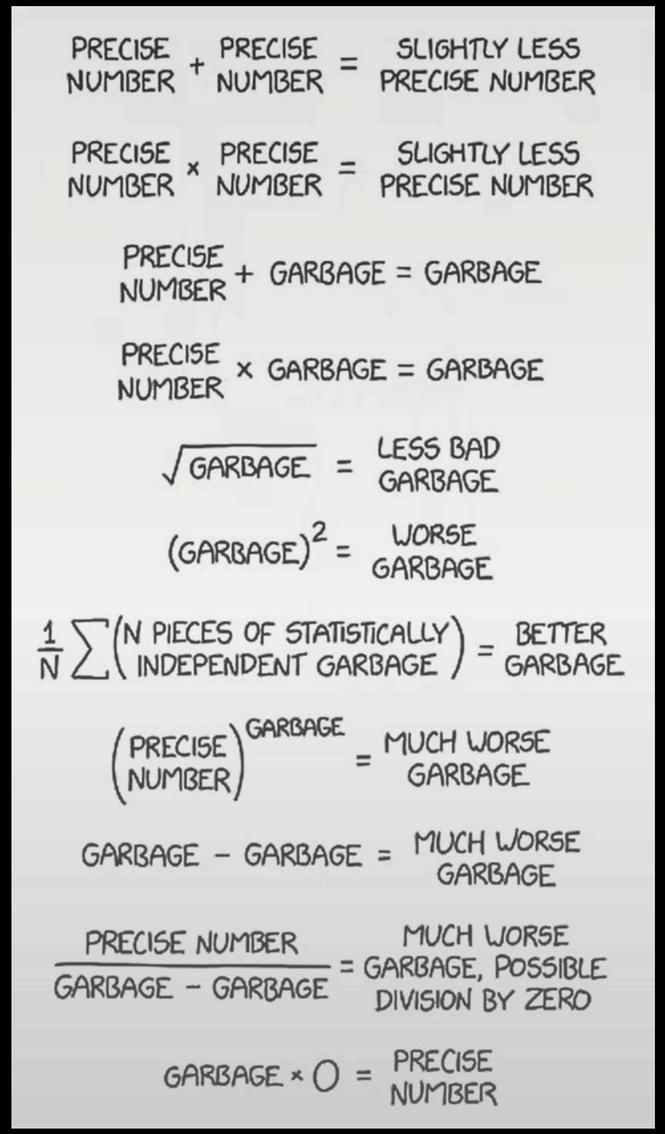
In context of DL, quantization refers to the process of reducing the number of bits that are used to represent a value:



01 The Core Problem
02 Quantization Basics
03 Binary Connect
04 Benefits
05 Forward Pass
06 Backpropagation
07 Experimental Results
08 Academic Researcher
09 BNN - Results
10 Industry Expert







01 The Core Problem
02 Quantization Basics
03 Binary Connect
04 Benefits
05 Forward Pass
06 Backpropagation
07 Experimental Results
08 Academic Researcher
09 BNN - Results
10 Industry Expert

Binary Connect

BinaryConnect is a way to train deep neural networks using binary weights — just +1 or −1 — during forward and backward passes.
Real weights kept for updates → **stable learning**

Advantages:

- Fewer multiplications → efficient hardware
- Built-in regularization → better generalization

Results: Near state-of-the-art on MNIST, CIFAR-10, SVHN

01 The Core Problem

02 Quantization Basics

03 Binary Connect

04 Benefits

05 Forward Pass

06 Backpropagation

07 Experimental Results

08 Academic Researcher

09 BNN - Results

10 Industry Expert

Deterministic vs Stochastic Binarization

Deterministic Binarization

- Rule:

$$w_b = \begin{cases} +1 & \text{if } \ w \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

- Simple sign function, no randomness
- Averaging over many inputs can compensate for information loss

Stochastic Binarization

- Rule:

$$w_b = \begin{cases} +1 & \text{with probability } p = \sigma(w) \\ -1 & \text{with probability } 1 - p \end{cases}$$

- Uses hard sigmoid:

$$\sigma(x) = \text{clip}\left(\frac{x+1}{2}, 0, 1\right)$$

- Piecewise linear and hardware friendly
- Allows finer averaging, better represents real weight statistics

01 The Core Problem

02 Quantization Basics

03 Binary Connect

04 Benefits

05 Forward Pass

06 Backpropagation

07 Experimental Results

08 Academic Researcher

09 BNN - Results

10 Industry Expert

Why Binary Connect works?

1. Precision where it matters (SGD updates)

- Real weights are updated in high precision → stable learning
- **Only forward/backward use binary weights** → no multiplications
- Prior work: 6–12 bits are enough for SGD updates

2. Noise as a regularizer

- Binarization adds weight noise, similar to Dropout/DropConnect
- **Expected weight value remains accurate**
- Noise improves generalization without hurting accuracy

01 The Core Problem

02 Quantization Basics

03 Binary Connect

04 Benefits

05 Forward Pass

06 Backpropagation

07 Experimental Results

08 Academic Researcher

09 BNN - Results

10 Industry Expert

Forward propagation

- Input → layer activations computed using binarized weights.
- No multiplications: only additions or sign flips.
- The real weights (w) are not discarded; they are just

  binarized temporarily ($w_b = \text{binarize}(w)$).

  wb - Binary weight,
  b - bias
  a - Neuron value

  $w_b \leftarrow \text{binarize}(w_{t-1})$

  For $k = 1$ to L, compute a_k knowing $a_{k-1}, w_b,$ and b_{t-1}

01 The Core Problem

02 Quantization Basics

03 Binary Connect

04 Benefits

05 Forward Pass

06 Backpropagation

07 Experimental Results

08 Academic Researcher

09 BNN - Results

10 Industry Expert

Backward propagation

- Gradients are computed using the same binarized weights (w_b).
- This ensures the backward pass is also lightweight — no expensive float multiplications.

Parameter update

- Real weights (w) are updated using the computed gradients.

$$w_t \leftarrow \text{clip}\left(w_{t-1} - \eta \frac{\partial \mathcal{C}}{\partial w_b}\right)$$

- Why clipping? Clips weights to [–1, 1] to prevent drift.
- Biases are also updated normally:

$$b_t \leftarrow b_{t-1} - \eta \frac{\partial \mathcal{C}}{\partial b_{t-1}}$$

01 The Core Problem

02 Quantization Basics

03 Binary Connect

04 Benefits

05 Forward Pass

06 Backpropagation

07 Experimental Results

08 Academic Researcher

09 BNN - Results

10 Industry Expert

Binary Connect - Results

Method	MNIST	CIFAR-10	SVHN
No regularizer	$1.30 \pm 0.04\%$	10.64%	2.44%
BinaryConnect (det.)	$1.29 \pm 0.08\%$	9.90%	2.30%
BinaryConnect (stoch.)	$1.18 \pm 0.04\%$	**8.27%**	2.15%
50% Dropout	$1.01 \pm 0.04\%$		
Maxout Networks [29]	0.94%	11.68%	2.47%
Deep L2-SVM [30]	**0.87%**		
Network in Network [31]		10.41%	2.35%
DropConnect [21]			1.94%
Deeply-Supervised Nets [32]		9.78%	**1.92%**

Datasets: **MNIST, CIFAR-10, SVHN**

- MNIST (MLP): 3×1024 ReLU + L2-SVM → **Stochastic BC ≈ Dropout regularization**
- CIFAR-10 / SVHN (CNN): VGG-style → **BinaryConnect improves test error**

Main intuition

- Binarization injects noise → prevents overfitting → better test performance.
- Trade-off: **Deterministic = speed**
  Stochastic = accuracy.

01 The Core Problem

02 Quantization Basics

03 Binary Connect

04 Benefits

05 Forward Pass

06 Backpropagation

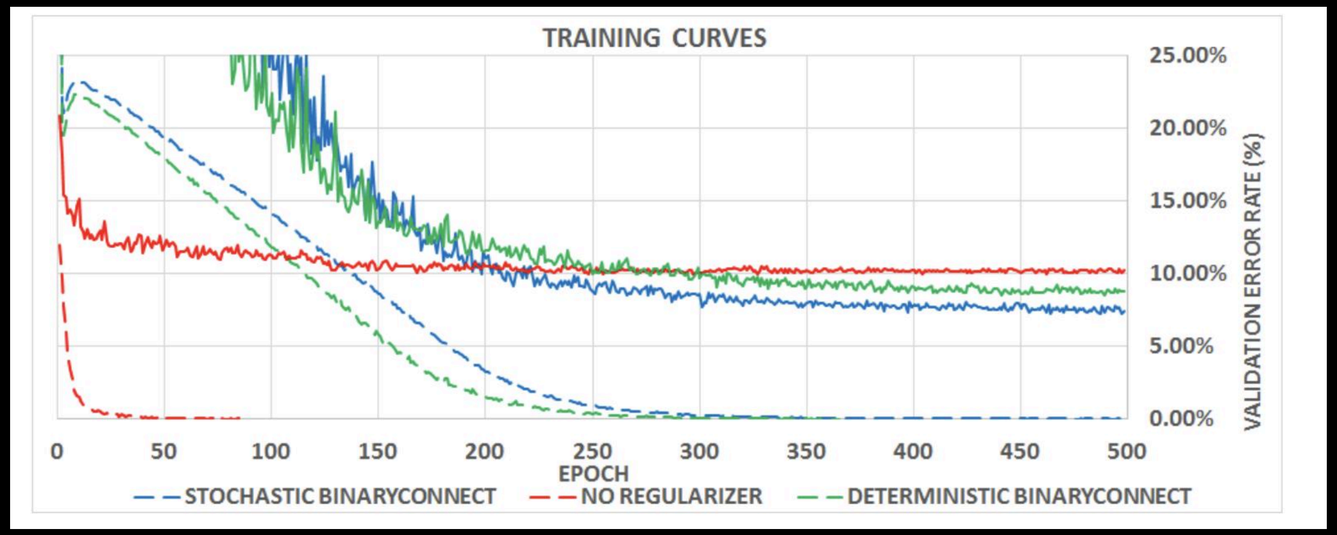
07 Experimental Results

08 Academic Researcher

09 BNN - Results

10 Industry Expert

Binary Connect - Key findings



- BinaryConnect acts as a regularizer, similar to Dropout.
- Accuracy does not degrade despite 1-bit weights, often improves.
- Stochastic BC slightly better generalization, deterministic BC faster inference.
- Weight histograms show weights tend to ±1, confirming binarization behavior.

01 The Core Problem

02 Quantization Basics

03 Binary Connect

04 Benefits

05 Forward Pass

06 Backpropagation

07 Experimental Results

08 Academic Researcher

09 BNN - Results

10 Industry Expert

Binary Neural Network

- To train a Binary Neural Network, we need to:
  - Binarize weights
  - Binarize activations
- Binarization will only really be completed after training.
- There is an early model called BinaryConnect (2015) that only binarizes weights, but it is now standard to require both conditions.
- Let’s see a few basic concepts that we need before focusing on how BNNs are built and trained...

01 The Core Problem

02 Quantization Basics

03 Binary Connect

04 Benefits

05 Forward Pass

06 Backpropagation

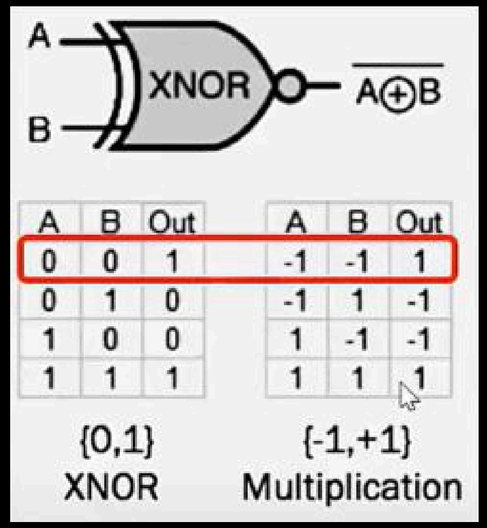
07 Experimental Results

08 Academic Researcher

09 BNN - Results

10 Industry Expert

BNNs: XNOR affords cheap operations



- XNOR (exclusive NOR): XOR + inverter gate
- If 0 is mapped to -1 (and 1 is still a 1), then XNOR corresponds to bitwise multiplication – this is very computationally inexpensive.
- Convolutions are therefore implemented simply by XNOR + POPCOUNT operations.

01 The Core Problem

02 Quantization Basics

03 Binary Connect

04 Benefits

05 Forward Pass

06 Backpropagation

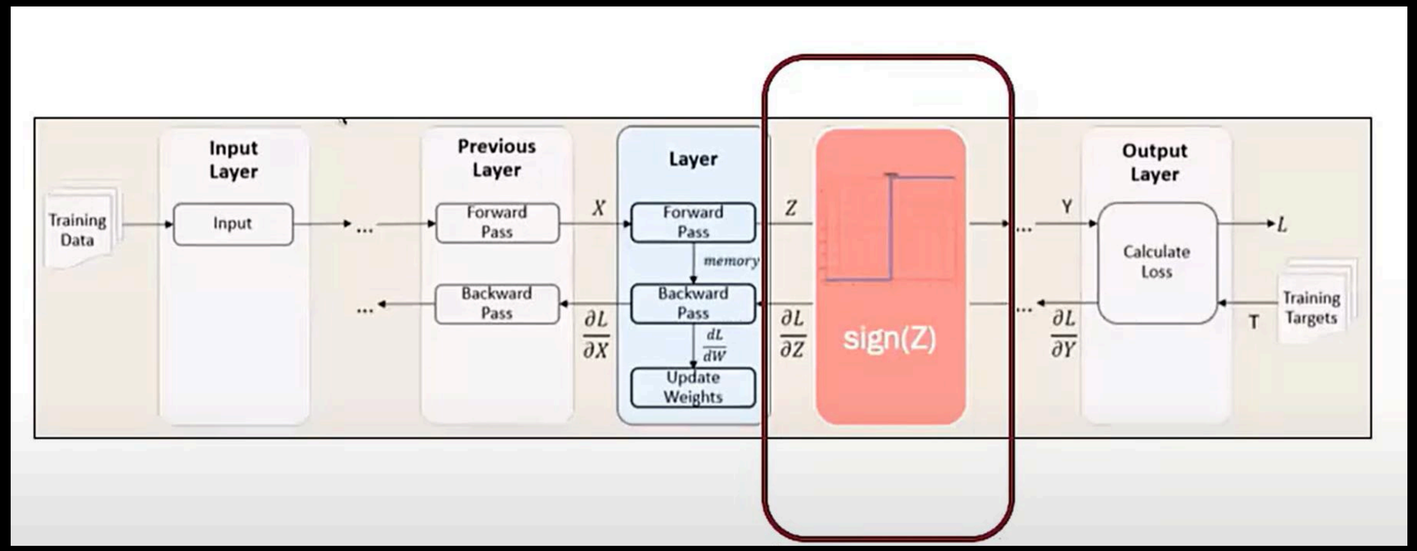
07 Experimental Results

08 Academic Researcher

09 BNN - Results

10 Industry Expert

BNNs: Training



- Implementing this layer is the only change required in order to binarize a neural network.
- There are libraries that take care of the heavy lifting.

01 The Core Problem

02 Quantization Basics

03 Binary Connect

04 Benefits

05 Forward Pass

06 Backpropagation

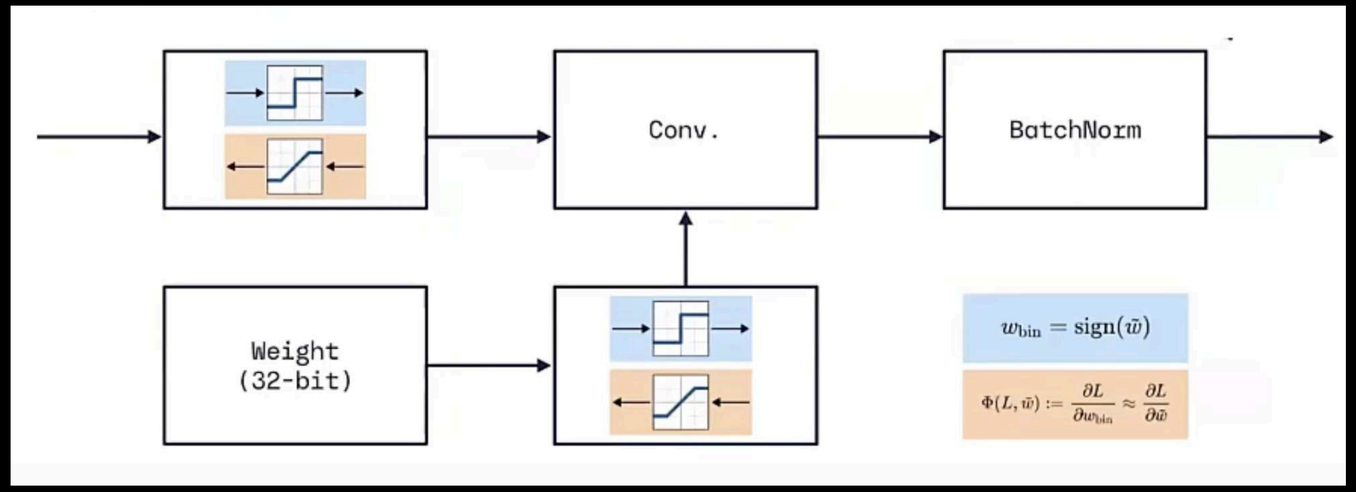
07 Experimental Results

08 Academic Researcher

09 BNN - Results

10 Industry Expert

BNNs: Training



- **Forward pass:** Weights are binarized.
- **Backward pass:** Need to approximate the gradient (Why?)

01 The Core Problem

02 Quantization Basics

03 Binary Connect

04 Benefits

05 Forward Pass

06 Backpropagation

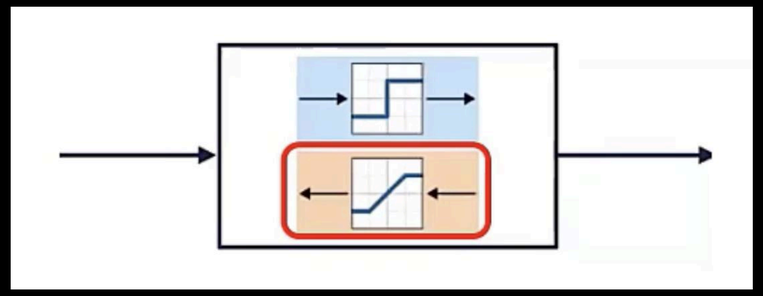
07 Experimental Results

08 Academic Researcher

09 BNN - Results

10 Industry Expert

BNNs: The Straight-through Estimater



- Backpropagation through the sign function yields zero for all derivatives.
- We thus need a way to estimate the gradient since otherwise training will not be possible.
- The Straight-through Estimator (STE):
  - Simply set the incoming gradients equal to the outgoing gradients (i.e., disregard the derivative of the function itself).
  - On average, this is a good estimation that is very easy to implement (via the “hard tanh” function).

01 The Core Problem

02 Quantization Basics

03 Binary Connect

04 Benefits

05 Forward Pass

06 Backpropagation

07 Experimental Results

08 Academic Researcher

09 BNN - Results

10 Industry Expert

Memory and Storage

- Lead to 32 times lower memory storage for weights
- Binarizing both activations and weights in BNNs results in greater memory compression than BinaryConnect

Increased Computational Efficiency and Speed

- Up to 58 times faster than 32-bit CNNs
- 7 times faster inference speed on a custom GPU compared to BinaryConnect on small datasets like MNIST

Lower Energy Consumption and Hardware Area

- The reduction in complex arithmetic operations and memory access leads to lower power consumption

Accuracy (with optimization)

- enabled BNNs to achieve competitive accuracy, often near state-of-the-art results

01 The Core Problem

02 Quantization Basics

03 Binary Connect

04 Benefits

05 Forward Pass

06 Backpropagation

07 Experimental Results

08 Academic Researcher

09 BNN - Results

10 Industry Expert

- **Edge AI and IoT Devices** – for limited compute and memory resources
- **Mobile and Embedded Vision** – for real-time inference with low power
- **Healthcare and Medical Devices** – for portable, energy-efficient diagnostics
- **Cybersecurity and Signal Processing** – for fast signal-level tasks
- **Smart Surveillance** – run on restricted compute/power budgets
- **High-Frequency Trading** – for lightweight, high-speed transaction analysis

01 The Core Problem

02 Quantization Basics

03 Binary Connect

04 Benefits

05 Forward Pass

06 Backpropagation

07 Experimental Results

08 Academic Researcher

09 BNN - Results

10 Industry Expert

The producers of this report

Shreyansh Shrivastava

Subham

Mansi Gupta

Bibhuti B. Panda

Thanks

Questions are welcomed

Snores are not.