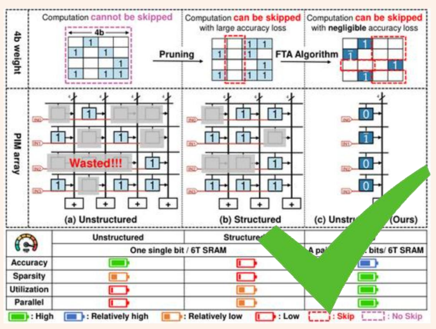
Interleaved summarization from **M-DocSum-7B:**

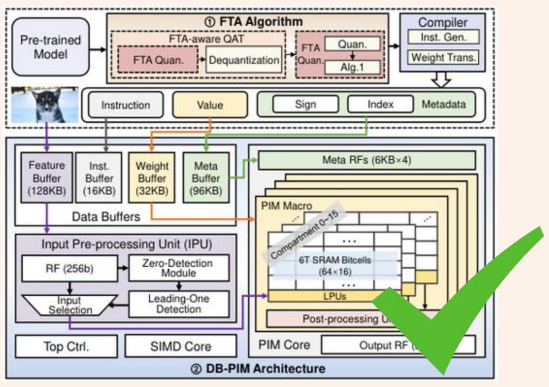
Research Background: Deep Neural Networks (DNNs) are widely used in applications such as image recognition, speech recognition, and object detection, but they require substantial memory and computing resources, which can be a challenge for resource-constrained devices. Processing-in-memory (PIM) offers a potential solution by executing multiply-accumulate (MAC) operations in memory, thereby eliminating the Memory Wall caused by frequent data movement. Current SRAM-PIM research focuses on sparsity support to further improve computational efficiency, with most studies leveraging value-level sparsity. However, bit-level sparsity, which eliminates redundant computations associated with zero bits in values, has been explored less. While bit-level sparsity has been applied in traditional digital accelerators, its efficient utilization in SRAM-PIM remains a significant challenge due to the rigid crossbar structure of PIM-based accelerators.

Com: 0.667 Acc: 1.0



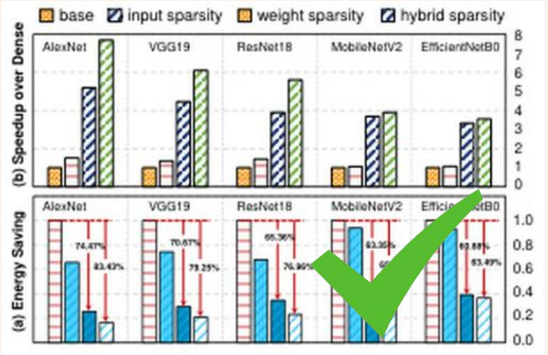
Proposed Method: The Dyadic Block PIM (DB-PIM) framework is introduced as a groundbreaking algorithm-architecture co-design solution to effectively harness unstructured bit-level sparsity in SRAM-PIM. At the algorithm level, the Fixed Threshold Approximation (FTA) algorithm is proposed alongside a unique bit-level sparsity pattern called the dyadic block (DB), which employs Canonical Signed Digit (CSD) encoding. This approach maintains the random distribution of non-zero bits for accuracy while restricting the number of non-zero bits in each weight to improve regularity. At the architecture level, a customized PIM macro is designed to include dyadic block multiply units (DBMUs) and CSD-based adder trees, specifically tailored for Multiply-Accumulate (MAC) operations. Additionally, an input pre-processing unit (IPU) is developed to refine performance and efficiency by capitalizing on block-wise input sparsity.

Com: 0.357 Acc: 1.0



Experimental Results: DB-PIM was evaluated on a 28 nm technology platform with a 128 KB feature buffer, a 16 KB instruction buffer, and other hardware configurations. The FTA algorithm demonstrated minimal accuracy loss, with accuracy degradation remaining below 1% even when applied to all layers of compact NN models. The framework achieved significant speedup across various NN models, with a speedup of about 5.20× for AlexNet and 4.46× for VGG19 by utilizing weight sparsity. Energy savings were also substantial, with energy efficiency improving by 63.49% to 83.43% across five classical NN models. The area overhead of DB-PIM was relatively minor, with an overhead of approximately 0.48% compared to the digital PIM baseline. The framework outperformed existing state-of-the-art PIM-based accelerators in terms of utilization, peak throughput per macro, energy efficiency, and energy efficiency per unit area.

Com: 0.5 Acc: 0.5



Conclusion: The DB-PIM framework effectively utilizes unstructured bit-level sparsity in digital SRAM-PIM, overcoming the limitations of conventional designs. The results show that DB-PIM achieves a remarkable 5.20× speedup and a 74.47% improvement in energy saving by utilizing unstructured weight bit sparsity. When combined with input bit sparsity, the framework attains even more remarkable results, with a 7.69× speedup and an 83.43% increase in energy efficiency. These results validate the co-design approach and highlight its potential to utilize bit-level sparsity. Future work aims to combine the approach with value-level sparsity to maximize the exploitation of sparsity in NN models.

Com: 0.5 Acc: 1.0

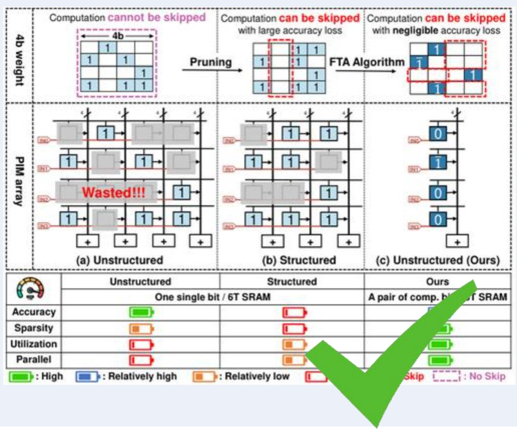
Image Idx: None

✓

Interleaved summarization from **Qwen2-VL-7B**

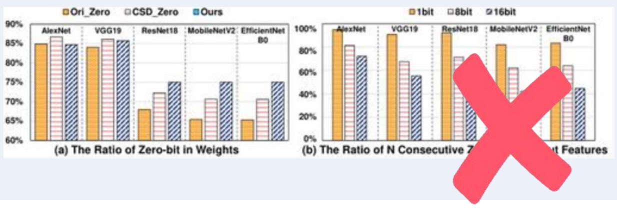
This paper addresses the computational efficiency problem in deep neural networks (DNNs) and proposes a novel algorithm-architecture co-design framework, DB-PIM, to effectively exploit unstructured bit-level sparsity in digital SRAM-PIM architectures. The framework includes an FTA algorithm that maintains model accuracy while preserving the randomness of non-zero bits and a DB-PIM architecture that accelerates these patterns. The results show that DB-PIM achieves a remarkable speedup of up to 7.69x and energy savings of 83.43%.

Com: 0.083 Acc: 0.5



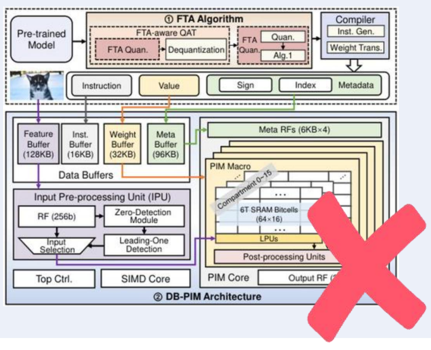
The proposed DB-PIM framework includes an FTA algorithm and a DB-PIM architecture. The FTA algorithm employs a fixed threshold approximation (FTA) algorithm and a unique bit-level sparsity pattern, termed dyadic block (DB), which employs the Canonical Signed Digit (CSD) encoding method. This approach maintains the randomness of non-zero bits while limiting the number of non-zero bits in each weight to improve regularity. The DB-PIM architecture is designed to accelerate these patterns, specifically tailored for Multiply-Accumulate (MAC) operations. It includes dyadic block multiply units (DBMUs) and CSD-based adder trees, which are specifically designed for efficient MAC operations on randomly distributed non-zero bits. Additionally, the architecture dynamically detects sparse blocks from input features, bypassing all-zero-bit blocks to enhance computational efficiency.

Com: 0.5 Acc: 1.0



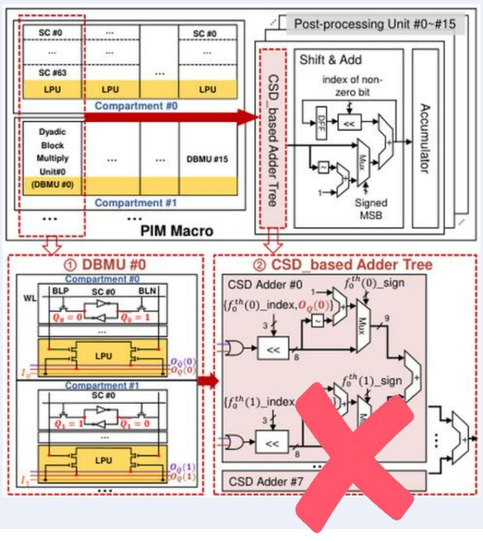
The experimental results demonstrate that the proposed DB-PIM framework achieves a remarkable speedup of up to 7.69x and energy savings of 83.43% over the dense PIM baseline. The speedup is achieved by utilizing unstructured weight bit sparsity, and when combined with input bit sparsity, the speedup increases to 7.69x. The energy efficiency is also improved by up to 83.43% compared to the dense PIM baseline. The area breakdown analysis shows that the additional logic required for the DB-PIM framework is relatively minor, approximately 0.48%. The comparison with prior works shows that DB-PIM achieves up to 3x improvement in actual utilization, 3.14x improvement in peak throughput per macro, and higher energy efficiency and energy efficiency per unit area.

Com: 0.286 Acc: 0.0



This paper proposes a novel algorithm-architecture co-design framework, DB-PIM, to effectively exploit unstructured bit-level sparsity in digital SRAM-PIM architectures. The framework includes an FTA algorithm that maintains model accuracy while preserving the randomness of non-zero bits and a DB-PIM architecture that accelerates these patterns. The results show that DB-PIM achieves a remarkable speedup of up to 7.69x and energy savings of 83.43%.

Com: 0.3 Acc: 0.5



Evaluation Com and Acc of **Conclusion:**

Partially Included +0.5

Completely Included +1

From M-DocSum:
Com = 2.5 / 5 = 0.5

From Qwen2-VL:
Com = 1.5 / 5 = 0.3

Key points from paragraph 4:

1. This paper proposes the Dyadic Block PIM (DB-PIM) framework to exploit unstructured bit-level sparsity in digital SRAM-PIM.
2. The framework integrates the FTA algorithm, CSD encoding, and a customized architecture to address computational dependency and low utilization issues.
3. Experimental results demonstrate that DB-PIM achieves up to 7.69× speedup and 83.43% energy savings, with minimal accuracy loss across various neural network models.
4. The proposed approach significantly improves utilization, throughput, and energy efficiency compared to existing PIM-based accelerators.
5. Future work aims to combine the DB-PIM framework with value-level sparsity techniques to maximize the exploitation of sparsity in neural network models.

From M-DocSum:

"accuracy": "Completely Accurate",
"fluency": "Fluent",
"repetitions": 0,
"hallucinations": 0,
"distortions": 1

→ Acc = (4 - 0 - 0) / 4 = 1.0

From Qwen2-VL:

"accuracy": "Mostly Accurate",
"fluency": "Fluent",
"repetitions": 1,
"hallucinations": 0,
"distortions": 0

→ Acc = (3 - 0 - 1) / 4 = 0.5

accuracy: "Completely Accurate" 4, "Mostly Accurate" 3, "Partially Accurate" 2, "Inaccurate" 1
fluency: "Fluent" 0, "Mostly Fluent" 1, "Not Fluent" 2
penalty_times = repetitions + hallucinations + distortions
Acc = (accuracy - fluency - penalty_times) / 4

Evaluation Criteria

Evaluation of **Reference Images:**

Reference images from M-DocSum: [1, 3, 6, 'None']

Ground truth reference images: [1, 3, 6, 'None']

Reference images from Qwen2-VL: [1 , 2, 3, 4]

ImgAcc = 3 / 3 = 1 NonAcc = 1 / 1 = 1 OMatch = 4 / 4 = 1
JacSim = Jaccard Similarity(set(1,3,6,'None'), set(1,3,6,'None')) = 4 / 4 = 1

ImgAcc = 1 / 3 = 0.33 NonAcc = 0 / 1 = 0 OMatch = 1 / 4 = 0.25
JacSim = Jaccard Similarity(set(1,3,6,'None'), set(1,2,3,4)) = 2 / 4 = 0.5