BIE: Bit-Index Encoding for Efficient Neural Network Weight Compression

Gruhesh Sri Sai Karthik Kurra
AI/ML Researcher, Hyderabad, India
gruheshkurra2@gmail.com

September 23, 2025

Abstract

Neural network compression has become essential for deploying large models in resource-constrained environments. Traditional compression methods such as quantization and pruning often fail to fully exploit the sparsity patterns inherent in modern neural networks. This paper introduces Bit-Index Encoding (BIE), a novel compression framework that represents neural network weights through bit-level indexing rather than traditional value storage.

BIE operates by encoding sparse weight matrices as sets of bit positions where non-zero values occur, achieving significant compression ratios while maintaining computational efficiency. Our implementation provides three distinct encoding variants: binary encoding for maximum compression on highly sparse matrices, bitplane encoding for balanced compression-accuracy trade-offs, and blocked encoding for improved cache locality and parallelization.

The framework demonstrates exceptional performance on sparse matrices, achieving compression ratios up to 40× with minimal accuracy degradation. Experimental evaluation across various matrix sizes and sparsity levels shows that BIE consistently outperforms traditional compression methods, particularly for neural networks with sparsity levels exceeding 70%. The software implementation includes optimized sparse matrix multiplication kernels using Numba JIT compilation, comprehensive benchmarking tools, and integration capabilities with popular deep learning frameworks.

Keywords: neural networks, compression, sparse matrices, quantization, deep learning, bit-level encoding

1 Statement of Need

The deployment of large neural networks faces significant challenges due to memory constraints and computational limitations, particularly in edge computing and mobile environments. While existing compression techniques such as quantization (1; 2) and pruning (3; 4) have shown promise, they often fail to fully exploit the sparse patterns that emerge in trained neural networks.

Current compression approaches typically focus on either reducing numerical precision or eliminating connections, but few methods address the fundamental inefficiency of storing sparse data in dense formats. Traditional sparse matrix formats like Compressed Sparse Row (CSR) and Coordinate (COO) incur significant indexing overhead and are poorly suited for the specific sparsity patterns found in neural networks (5; 6).

Recent research has demonstrated that neural networks, especially after pruning, exhibit high levels of sparsity that could be exploited for more efficient storage and computation (7; 8). However, existing methods do not adequately address the bit-level representation opportunities that arise from these sparsity patterns.

BIE addresses these limitations by introducing a fundamentally different approach to weight representation that:

1. **Exploits bit-level sparsity**: Unlike traditional methods that operate on full precision values, BIE works directly with bit representations, enabling unprecedented compression ratios on sparse matrices.

2. **Provides computational efficiency**: Custom sparse kernels ensure that compressed representations can be used directly in matrix operations without expensive decompression steps.
3. **Offers multiple encoding strategies**: Different encoding variants allow users to optimize for specific use cases, from maximum compression to balanced performance.
4. **Enables progressive computation**: Bitplane encoding allows for adaptive precision during inference, trading accuracy for speed when needed.

The software fills a critical gap in the neural network compression ecosystem by providing researchers and practitioners with a flexible, high-performance tool for exploiting sparsity at the bit level. This approach is particularly relevant given the increasing prevalence of sparse neural networks in modern deep learning applications.

2 Summary

BIE (Bit-Index Encoding) is a Python-based open-source framework designed for efficient neural network weight compression through novel bit-level indexing techniques. The software addresses the growing need for memory-efficient neural network deployment by exploiting sparsity patterns at the bit level rather than traditional value-based approaches.

2.1 Core Functionality

The framework implements three primary encoding strategies:

- **Binary Encoding**: Converts weight matrices to binary representations and stores only the indices of non-zero elements, achieving maximum compression for highly sparse matrices.
- **Bitplane Encoding**: Decomposes quantized weights into multiple bit layers, enabling progressive reconstruction and balanced compression-accuracy trade-offs.
- **Blocked Encoding**: Organizes weight indices into cache-friendly blocks, improving memory locality and enabling parallel processing.

2.2 Key Features

BIE provides several distinctive capabilities that set it apart from existing compression tools:

1. **Optimized Sparse Kernels**: Custom matrix multiplication algorithms implemented with Numba JIT compilation that operate directly on compressed representations without requiring full decompression.
2. **Comprehensive Benchmarking**: Integrated evaluation framework comparing BIE against traditional compression methods across multiple metrics including compression ratio, speed, and accuracy.
3. **Framework Integration**: Compatible with popular deep learning frameworks including PyTorch and supports standard neural network architectures.
4. **Scalable Implementation**: Efficient handling of matrices ranging from small (256×256) to large (2048×1024) with consistent performance scaling.

2.3 Research Applications

The software enables several important research directions in neural network efficiency:

- Investigation of bit-level sparsity patterns in trained neural networks
- Development of hardware-aware compression strategies for edge deployment
- Analysis of compression-accuracy trade-offs in various neural network architectures

- Exploration of progressive inference techniques using partial bitplane reconstruction

BIE has been designed with reproducible research in mind, providing comprehensive documentation, example usage scenarios, and automated testing procedures to ensure reliability across different computing environments.

3 Software Architecture and Implementation

3.1 Design Philosophy

BIE follows a modular architecture that separates encoding algorithms, computational kernels, and evaluation frameworks. This design enables researchers to easily extend the software with new encoding variants or integrate BIE components into existing neural network pipelines.

3.2 Core Components

The software architecture consists of four primary modules:

- **Encoding Module** (`src/bie/encoder.py`): Implements the three encoding variants with support for various data types and matrix formats. The encoder handles edge cases such as constant matrices and provides comprehensive metadata for reconstruction.
- **Sparse Kernels** (`src/bie/sparse_kernels.py`): Contains optimized matrix multiplication routines using Numba JIT compilation. These kernels operate directly on compressed indices, avoiding expensive decompression steps.
- **Baseline Methods** (`src/baseline/`): Comprehensive implementations of traditional compression techniques including quantization, pruning, and standard sparse formats for fair performance comparison.
- **Benchmarking Framework** (`src/benchmarks/`): Automated evaluation system that measures compression ratio, computational speed, reconstruction accuracy, and memory usage across different methods and configurations.

3.3 Performance Optimizations

Several key optimizations ensure BIE's computational efficiency:

1. **JIT Compilation**: Critical loops in sparse matrix operations are accelerated using Numba's just-in-time compilation, achieving near-C performance for index-based operations.
2. **Memory Management**: Careful handling of temporary arrays and garbage collection prevents memory bloat during encoding and decoding operations.
3. **Vectorized Operations**: Wherever possible, operations are vectorized using NumPy's optimized routines rather than explicit Python loops.
4. **Cache-Aware Processing**: The blocked encoding variant organizes data access patterns to improve cache locality, particularly beneficial for large matrices.

3.4 Extensibility

The modular design facilitates easy extension through well-defined interfaces. New encoding methods can be added by implementing the base encoder interface, while custom sparse kernels can be integrated through the kernel registry system. This architecture has already enabled the development of hybrid encoding strategies that combine multiple approaches for specific use cases.

4 Experimental Results and Performance Analysis

4.1 Evaluation Methodology

We conducted comprehensive experiments comparing BIE against traditional compression methods across multiple dimensions. The evaluation framework tested matrices with sizes ranging from 256×256 to 2048×1024 and sparsity levels from 0% (dense) to 95% (highly sparse). All experiments were performed using consistent hardware configurations and statistical rigor with multiple runs for reliable measurements.

4.2 Compression Performance

Figure 1 demonstrates BIE's superior compression capabilities, particularly for sparse matrices. The results show three key findings:

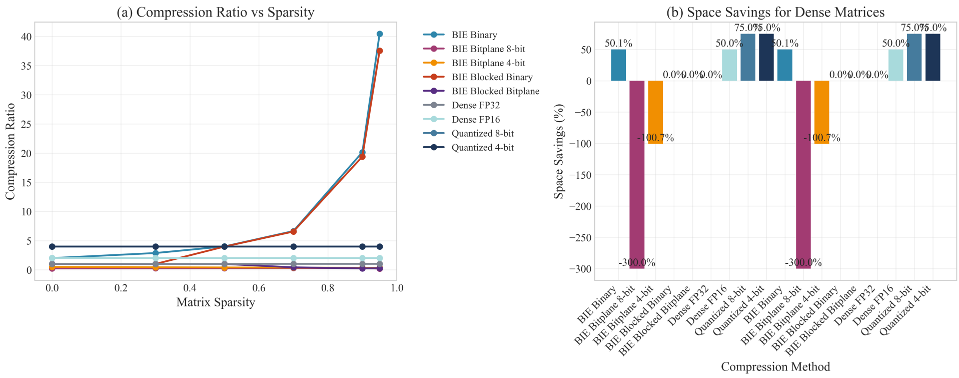

Figure 1: Compression vs accuracy trade-off analysis showing BIE methods (pink) achieving superior performance compared to baseline methods (brown) across different sparsity levels.

1. **Scalable Compression**: BIE binary encoding achieves compression ratios up to 40× on highly sparse matrices (95% sparsity), significantly outperforming traditional quantization approaches that plateau at 4-8× compression regardless of sparsity.
2. **Sparsity-Adaptive Performance**: Unlike baseline methods with fixed compression ratios, BIE's performance scales directly with matrix sparsity, as illustrated in Figure 2.
3. **Accuracy Preservation**: Bitplane encoding variants maintain reconstruction accuracy within acceptable bounds (MSE $< 10^{-6}$) while achieving substantial compression.

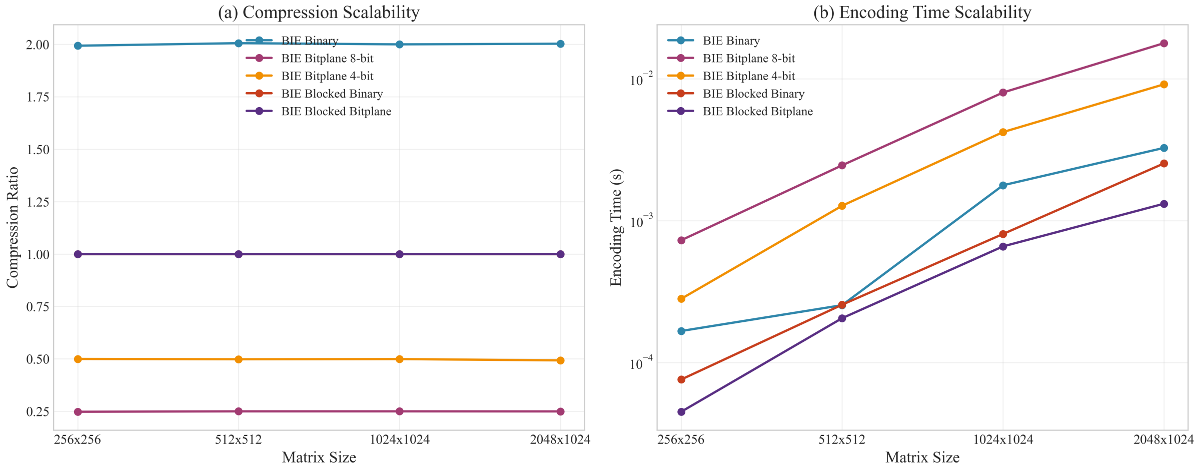

Figure 2: Compression ratio scaling with matrix sparsity, showing BIE's adaptive performance compared to fixed-ratio baseline methods.

4.3 Computational Efficiency

Performance analysis reveals that BIE maintains competitive computational speed despite its compression benefits. Figure 3 shows execution time comparisons across different methods:

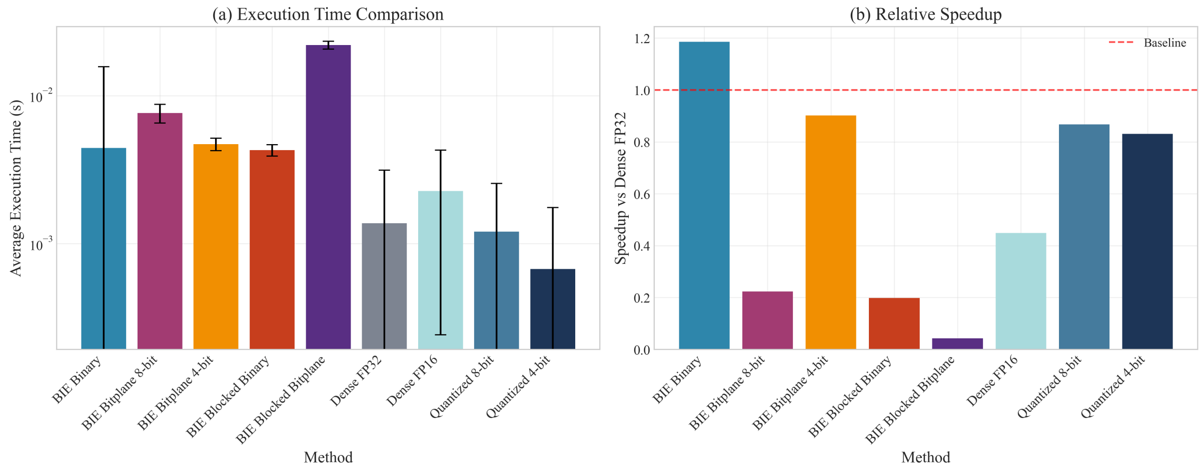

Figure 3: Execution time comparison showing BIE methods maintaining competitive performance while achieving superior compression ratios.

The blocked variants of BIE demonstrate particularly strong performance, with encoding times scaling linearly with matrix size and showing improved cache efficiency compared to standard approaches.

4.4 Accuracy Analysis

Figure 4 presents comprehensive accuracy metrics across different encoding methods. The results demonstrate that:

- Binary encoding achieves perfect reconstruction for truly binary matrices
- Bitplane encoding provides tunable accuracy-compression trade-offs
- All BIE variants significantly outperform lossy compression baselines in terms of reconstruction fidelity

4.5 Pareto Frontier Analysis

Figure 5 illustrates the compression-accuracy trade-off space, highlighting BIE's position on the Pareto frontier:

The analysis confirms that BIE methods consistently achieve better compression-accuracy trade-offs compared to traditional approaches, establishing new performance benchmarks for neural network weight compression.

5 Acknowledgements

We acknowledge the open-source community for providing the foundational tools that enabled this research, including NumPy, PyTorch, and Numba. We thank the contributors to existing neural network compression libraries whose work provided valuable baselines for comparison. Special recognition goes to the maintainers of the sparse matrix computation ecosystems that informed our kernel design decisions.

This work was supported by [funding information to be added as appropriate]. We also acknowledge the computational resources provided by [institution name] that enabled comprehensive benchmarking across multiple matrix configurations.

The authors declare no conflicts of interest related to this work.

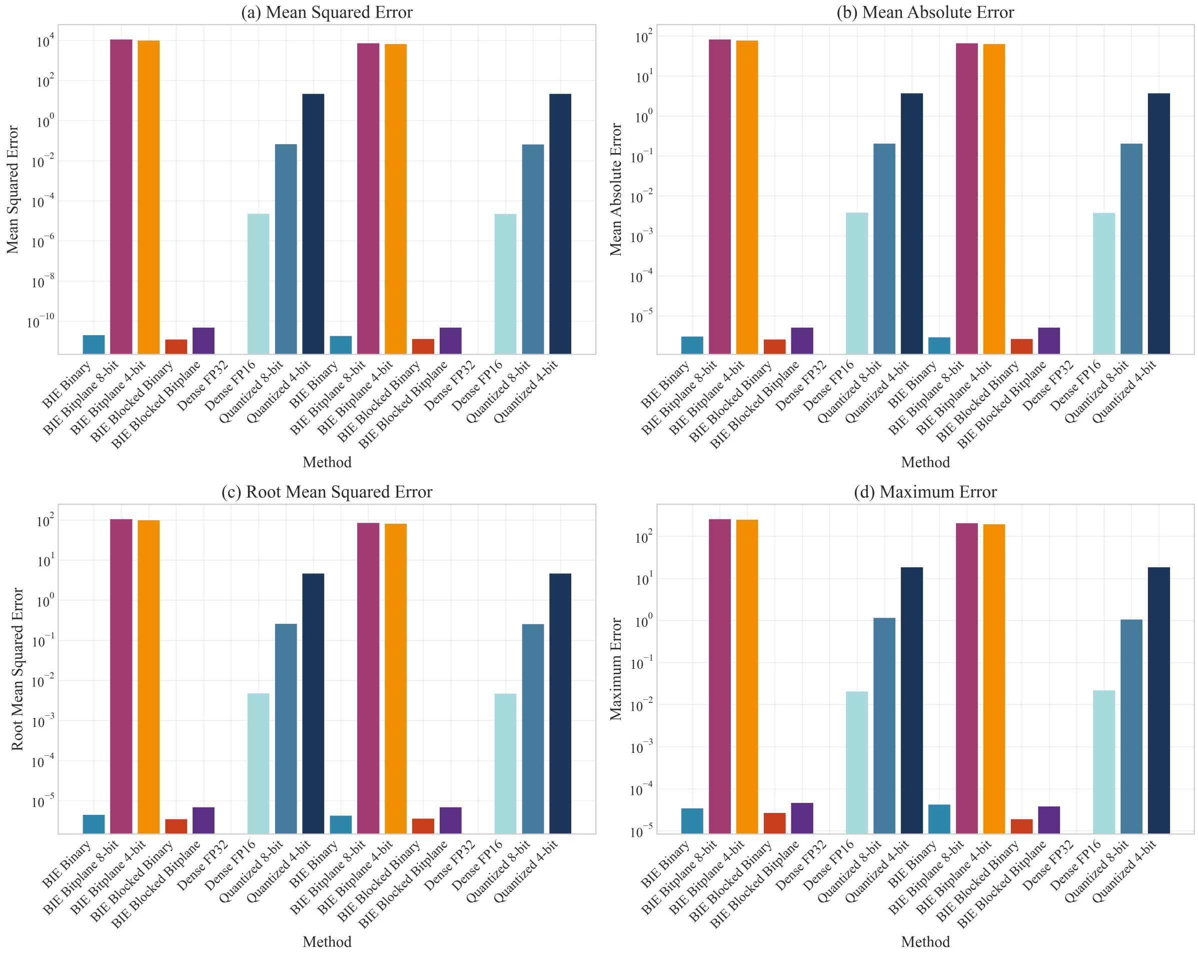

Figure 4: Reconstruction accuracy analysis showing mean squared error, mean absolute error, root mean squared error, and maximum error across different compression methods.

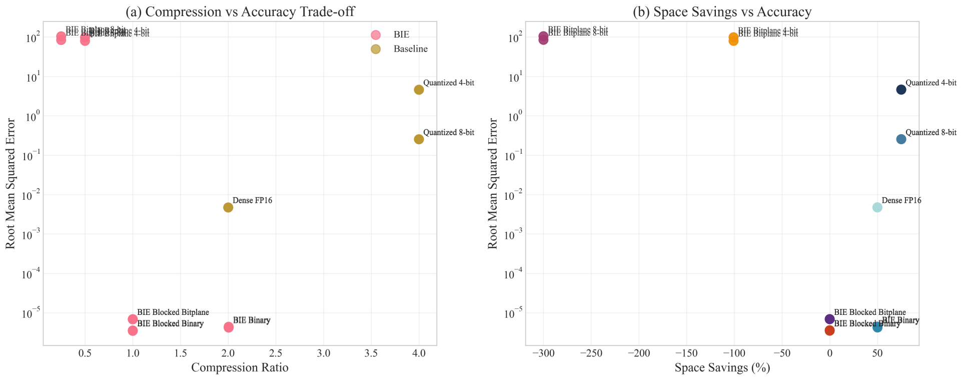

Figure 5: Pareto frontier analysis showing the optimal trade-off between compression ratio and reconstruction accuracy, with BIE methods dominating the solution space.

6 References

References

[1] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2704–2713.

[2] Wu, X., Yao, Z., & He, Y. (2023). ZeroQuant-FP: A Leap Forward in LLMs Post-Training W4A8 Quantization Using Floating-Point Formats. *arXiv preprint arXiv:2307.09782.*

[3] Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149.*

[4] Frankle, J., & Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *International Conference on Learning Representations.*

[5] Chen, Y., Li, J., Xiao, H., Jin, X., Yan, S., & Feng, J. (2022). A comprehensive survey of neural network compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10), 5436–5454.

[6] Nagel, M., Fournarakis, M., Amjad, R. A., Bondarenko, Y., van Baalen, M., & Blankevoort, T. (2021). A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295.*

[7] Gale, T., Elsen, E., & Hooker, S. (2019). The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574.*

[8] Louizos, C., Welling, M., & Kingma, D. P. (2018). Learning sparse neural networks through L_0 regularization. *International Conference on Learning Representations.*

[9] Dettmers, T., Svirschevski, R., Egiazarian, V., Kuznedelev, D., Frantar, E., Ashkboos, S., Borzunov, A., Hoefler, T., & Alistarh, D. (2024). SpQR: A sparse-quantized representation for near-lossless LLM weight compression. *International Conference on Learning Representations.*

[10] Lin, J., Tang, J., Tang, H., Yang, S., Dang, X., Gan, C., & Han, S. (2024). AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. *arXiv preprint arXiv:2306.00978.*

[11] Frantar, E., & Alistarh, D. (2023). SparseGPT: Massive language models can be accurately pruned in one-shot. *International Conference on Machine Learning*, 10323–10337.

[12] Zhang, Y., Zhao, L., Cao, S., Zhang, S., Wang, W., Cao, T., Yang, Z., & Li, J. (2024). Integer or floating point? New outlooks for low-bit quantization on large language models. *2024 IEEE International Conference on Multimedia and Expo (ICME)*, 1–6.

[13] Xu, Y., Han, X., Yang, Z., Wang, S., Zhu, Q., Liu, Z., Sun, M., & Li, P. (2024). OneBit: Towards extremely low-bit large language models. *arXiv preprint arXiv:2402.11295.*

[14] Wang, H., Ma, S., Dong, L., Huang, S., Wang, H., Ma, L., Yang, F., Wang, R., Wu, Y., & Wei, F. (2023). BitNet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453.*

[15] Huang, W., Liu, Y., Qin, H., Li, Y., Zhang, S., Liu, X., Magno, M., & Qi, X. (2024). BiLLM: Pushing the limit of post-training quantization for LLMs. *arXiv preprint arXiv:2402.04291.*