Accelerating LLM Inference via Vector Index Based Output Embeddings

Martin Loretz [1] **Sepp Hochreiter** [1 2]

Abstract

Large output embedding matrices create a significant memory bandwidth bottleneck during autoregressive decoding, especially for compact LLMs with large multilingual vocabularies. We reformulate the output projection followed by top-k token selection as a maximum inner product search over token embeddings and replace the dense vocabulary projection with an HNSW-based vector index. The resulting output head retrieves only a small candidate set of high-scoring tokens and can be integrated into existing decoding pipelines by scattering retrieved logits into a sparse full-vocabulary tensor. On CPU inference with Gemma 3, Llama 3.2, and Qwen 3 models, our method substantially accelerates the output projection and improves end-to-end batch-size-one decoding throughput by up to 82% for Gemma 3 270M, while preserving generation quality under AlpacaEval evaluation. These results suggest approximate retrieval is a practical alternative to dense output projections in latency-sensitive small-batch decoding.

1. Introduction

Compact LLMs are increasingly deployed in latency-sensitive edge settings, where decoding is often performed with batch size one. In this regime, the final vocabulary projection can become a dominant bottleneck. This is especially pronounced for recent small models distilled from larger multilingual systems: although their transformer backbones are compact, they often retain vocabularies with 100k–250k tokens. As a result, each decoding step requires streaming a large output embedding matrix from memory merely to select a small set of plausible next tokens.

To address this fundamental inefficiency and accelerate inference, we reformulate the output projection and subsequent top-k sampling step as a single Maximum Inner Product Search (MIPS) problem that can be efficiently solved using approximate retrieval algorithms typically used in vector databases. This approach allows us to directly retrieve the top-k tokens from the final hidden state without the memory bandwidth overhead of loading the entire dense matrix. Unlike structural approximations such as Adaptive Softmax (Joulin et al., 2017), which require fundamental architectural changes during pre-training, our approach can be seamlessly integrated into existing models post-training. It targets the latency-sensitive, small-batch regime typical of interactive local inference, rather than high-throughput server inference where batched GEMM is highly efficient.

In summary, we make the following contributions:

- We reformulate truncated token sampling as Maximum Inner Product Search (MIPS) over the output embedding matrix.
- We introduce a drop-in Hierarchical Navigable Small World (HNSW)-based output head that retrieves candidate logits without evaluating the full vocabulary projection.
- We evaluate the method across Gemma 3, Llama 3.2, and Qwen 3 models, showing up to 82% end-to-end throughput improvement at batch size one with minimal quality degradation.

The source code of a reference implementation is available at `https://github.com/martinloretzzz/vector-index-embedding`.

2. Background

Token Sampling in Large Language Models During autoregressive generation, language models (Vaswani et al., 2017) predict a probability distribution over the vocabulary V at each decoding step. While stochastic sampling ensures diverse and natural text generation, drawing from the full distribution often introduces the "long tail" problem, where low-probability tokens lead to incoherent outputs. To mitigate this, various truncation methods are utilized, differing primarily in how they determine the distribution cutoff. A prominent approach is top-k sampling (Fan et al., 2018), which restricts the sampling pool to the k most probable tokens, ensuring only semantically plausible candidates are considered while discarding noisy, low-probability tokens.

[1]NXAI, Austria [2] Johannes Kepler University Linz, Austria. Correspondence to: Martin Loretz <martin.loretz@nx-ai.com>.

Proceedings of the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026.

Maximum Inner Product Search (MIPS) The task of identifying the k tokens that maximize the inner product with the final hidden state $h \in \mathbb{R}^d$ is fundamentally a MIPS problem (Shrivastava & Li, 2014). Given a vocabulary size $|V|$, an exhaustive search requires computing the inner product for every row in the output embedding matrix W_{out}. While this brute-force approach guarantees exact results, its $O(|V| \cdot d)$ complexity quickly becomes a computational bottleneck. To achieve sub-linear query times, approximate data structures are utilized to trade marginal losses in retrieval recall for substantial reductions in latency (Indyk & Motwani, 1998).

Hierarchical Navigable Small Worlds (HNSW) A state-of-the-art algorithm for solving the MIPS problem in high-dimensional spaces is HNSW (Malkov & Yashunin, 2018). This graph-based approach constructs a proximity graph where nodes represent embedding vectors, and edges connect each node to up to M nearby vectors under the chosen similarity metric. Retrieval is performed by traversing the graph from a predefined entry node toward the query vector h using a greedy search. HNSW utilizes a multi-layered structure where the top layers contain sparse graphs with long-range edges for rapid coarse-grained navigation. As the search descends through the hierarchy, the graphs become increasingly dense, allowing the algorithm to refine the search in the bottom layer. This hierarchical approach enables the search to achieve logarithmic query complexity, $O(\log(|V|) \cdot d)$.

3. Vector Index Embedding

In language models, the final hidden state vector $h \in \mathbb{R}^d$ is projected onto the vocabulary space to produce a logit vector $z \in \mathbb{R}^{|V|}$, where $|V|$ is the vocabulary size and d is the hidden dimension. This projection is defined as:

$$z = W_{out}h$$

where $W_{out} \in \mathbb{R}^{|V| \times d}$ represents the output embedding matrix. The logit for an individual token i is the inner product of the hidden state and the token's corresponding embedding vector w_i:

$$z_i = \langle w_i, h \rangle$$

Since truncated decoding strategies, such as top-k sampling, only require the largest logits, computing the full vector z introduces an avoidable cost. Treating the rows of W_{out} as a static database of token vectors, selecting the largest logits is exactly a MIPS problem with query h.

To solve this search problem efficiently, we evaluated several approximate nearest neighbor algorithms using the FAISS (Douze et al., 2024) library. Our empirical results demonstrate that HNSW (Malkov & Yashunin, 2018) performs best within this high-dimensional embedding space, achieving an optimal trade-off between retrieval quality and search latency. While alternative algorithms based on clustering (Jegou et al., 2010) or the GPU-accelerated CAGRA (Ootomo et al., 2024) were considered, HNSW was the only candidate that provided sufficient retrieval accuracy to preserve the generative quality of the model.

The index is constructed from the rows of the output embedding matrix using the inner product as the similarity measure. Since these embeddings are static post-training, the index is precomputed and distributed alongside the model checkpoints. This allows us to prioritize index quality over construction time. During inference, the trade-off between throughput and approximation accuracy is controlled through the ef parameter, which dictates the size of the dynamic candidate list maintained during the traversal of the graph. While increasing ef improves recall, smaller values maximize throughput.

Due to the exponentiation in the softmax operation, the final sampling distribution is overwhelmingly dominated by the largest logits. Consequently, the primary risk of an approximate search is failing to retrieve the maximum logit when the model is highly confident, as such an omission causes a large divergence from the ground-truth distribution. We therefore explicitly compute the logits for some special tokens, common punctuation, and structural words to guarantee their inclusion.

While sampling can be performed directly on the retrieved top-k elements, a seamless integration with existing deep learning frameworks requires reconstructing the full logit tensor $z \in \mathbb{R}^{|V|}$ by mapping the retrieved inner products to their vocabulary indices and setting all unretrieved entries to $-\infty$. This approach allows for seamless integration with existing decoding pipelines, enabling the use of standard logit processors, such as top-p (Holtzman et al., 2019) or min-p (Nguyen et al., 2024) sampling, as a second stage.

While our approach drastically reduces memory bandwidth, storing the proximity graph alongside the vector index slightly increases the static memory footprint of the output layer. Assuming 32-bit representations, the memory requirement grows from $|V| \cdot 4d$ bytes for the dense matrix to $|V| \cdot (4d + 8M)$ bytes, representing a 5–10% overhead.

Overall, this architectural modification reduces the asymptotic complexity of the final projection from $O(d \cdot |V|)$ to approximately $O(d \cdot \log |V|)$. Although the random memory accesses inherent to graph traversal incur a substantially higher constant overhead than contiguous matrix multiplication, these algorithmic efficiency gains ultimately dominate at small batch sizes.

4. Experiments

We evaluate our approach on several open-source models from the Gemma 3 (Google DeepMind, 2025), Llama 3.2 (Grattafiori et al., 2024), and Qwen 3 (Yang et al., 2025) families by replacing their standard output projections with our vector index embedding. Due to the lack of GPU-accelerated HNSW implementations, all evaluations are done on the CPU in single-precision (`float32`) format. Although weight quantization (Frantar et al., 2022) and lower-precision data types are standard techniques for reducing memory bandwidth requirements, we focus on algorithmic acceleration, as our approach is orthogonal to these existing methods. We use our own fork of the `hnswlib` library that optimizes memory access patterns and multi-threading for high-dimensional vectors. Benchmarking was performed on a consumer-grade system equipped with a 6-core Intel Core i7-10750H CPU and 32 GB of DDR4 SDRAM, with the CPU frequency locked to 2.60 GHz. All HNSW vector indices were constructed using $M = 32$ and $ef_{\text{construction}} = 5000$. During text generation, we employed top-k sampling with $k = 50$.

We set $ef = 200$ for all primary experiments, as no subjective quality degradation is observed at this threshold. Halving ef to 100 effectively doubles search throughput, offering a viable trade-off for latency-critical applications where minor quality reductions are acceptable (Section 4.3).

4.1. Output Embedding Benchmarks

To isolate the performance gains attributable to our approach, we conduct micro-benchmarks comparing the median inference throughput of the HNSW-based output embedding against the standard dense output projection. These evaluations utilize real hidden states extracted from a model processing a Wikipedia article.

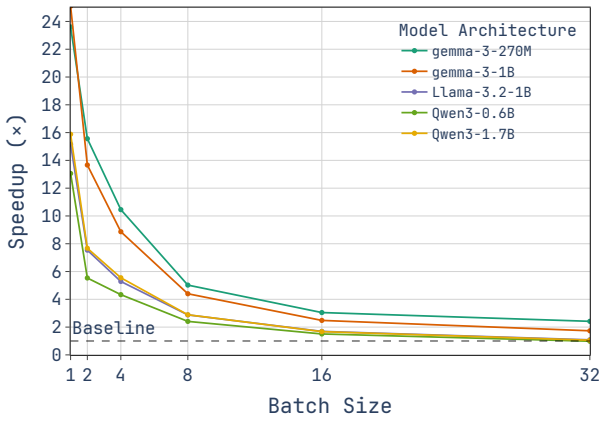


Figure 1. Relative inference speedup for the output projection at $ef = 200$, measured as the ratio of the median throughput of the vector index embedding to the exact matrix multiplication.

As demonstrated in Figure 1, our approach significantly accelerates the output projection relative to exact matrix multiplication, most notably at lower batch sizes. Under these conditions, inference throughput is primarily bottlenecked by memory bandwidth. By loading only a small subset of the embedding matrix, our HNSW formulation effectively bypasses the transfer overhead inherent to evaluating the full projection.

This advantage is most apparent at a batch size of 1, where we observe a speedup exceeding 12× for the Gemma models. As expected, performance gains further scale with vocabulary size, as the 256k-token Gemma models exhibit greater speedups than the smaller-vocabulary Llama (128k) and Qwen (152k) families.

4.2. Full Model Benchmarks

Although the vector index is substantially faster than the exact output projection, the aggregate end-to-end performance gain remains fundamentally constrained by the invariant computational overhead of the preceding layers. To quantify this impact, we evaluate the decoding throughput by generating 128-token responses to different short questions across varying batch sizes.

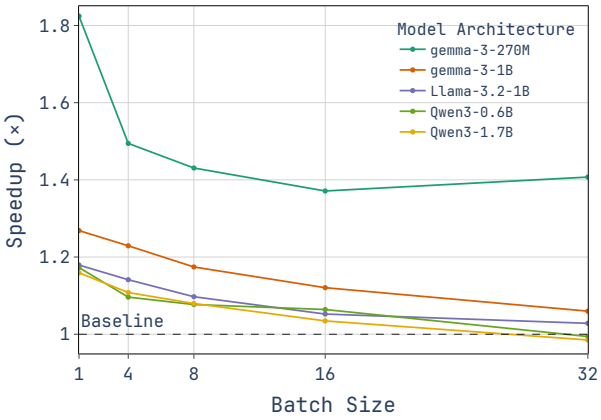


Figure 2. Relative inference speedup for the full model at $ef = 200$, measured as the ratio of the median throughput of the vector index based model to that of the standard baseline.

As shown in Figure 2, the end-to-end inference speedup reaches 82% during batch-size-one inference for the Gemma 3 270M model, whose embedding matrix alone accounts for approximately 62% of the total parameter count. Such performance gains underscore the disproportionate computational bottleneck created by large vocabularies in smaller architectures. Notably, this advantage persists at moderate batch sizes, with all evaluated models consistently outperforming the baseline. However, as the batch size exceeds 32, the search overhead offsets these efficiency gains, causing our method to fall behind for some models. Here,

the total number of unique tokens retrieved across the batch necessitates loading a substantial fraction of the embedding matrix into memory, thereby negating the memory bandwidth advantages of the vector index. Beyond this crossover point, exact matrix multiplication regains dominance due to the superior cache locality and contiguous memory access patterns of highly optimized GEMM kernels. This reduction in latency is visually illustrated in the profiler traces provided in Figure 3 in the Appendix.

4.3. Index Accuracy

To evaluate retrieval performance, we measure Recall@2 across varying search depths (ef) on 20k tokens from Wikipedia. Here, recall is the percentage of the true top 2 nearest neighbors retrieved by the vector index compared to the exact linear layer. We prioritize Recall@2 as the top two tokens typically capture roughly 70% of the total softmax probability mass, making them the most critical for preserving generation quality.

Table 1. Retrieval Recall@2 of the HNSW vector index across varying search depths (ef).

	RECALL@2 (%)		
MODEL	$ef = 100$	$ef = 200$	$ef = 400$
GEMMA 3 270M	97.3	99.1	99.7
GEMMA 3 1B	98.0	99.5	99.9
LLAMA 3.2 1B	97.9	99.5	99.9
QWEN3 0.6B	97.5	99.4	99.9
QWEN3 1.7B	96.3	99.0	99.8

As shown in Table 1, recall improves monotonically with ef. While the index achieves over 96% recall at $ef = 100$, it introduces a slight degradation in generation quality. Raising the search depth to $ef = 200$ pushes recall above 99%, virtually eliminating divergence from the exact baseline. While $ef = 400$ reaches up to 99.9% recall, the added search overhead justifies our choice of $ef = 200$ as the optimal balance between accuracy and speed.

4.4. Generation Quality

To evaluate the generative quality of our approach, we employ the LLM-as-a-judge paradigm (Zheng et al., 2023) using the AlpacaEval (Li et al., 2023) framework. Specifically, we use GPT-5 Nano (Singh et al., 2025) as the judge to perform a pairwise comparison between the text generated by our vector index model and the standard baseline at $ef = 200$. Traditional benchmarks such as MMLU (Hendrycks et al., 2020) are ineffective in this context, as our method preserves the model's internal representations, resulting in nearly identical scores.

Table 2. Pairwise length-controlled win rates on AlpacaEval, comparing our vector index approach ($ef = 200$) to the exact baseline across various model families.

MODEL	LENGTH CONTROLLED WIN RATE
GEMMA 3 270M	48.1 ± 0.10
GEMMA 3 1B	45.9 ± 0.23
LLAMA 3.2 1B	49.1 ± 0.25
QWEN3 0.6B	47.5 ± 0.26
QWEN3 1.7B	46.9 ± 0.22

As shown in Table 2, the length-controlled win rates for our vector index approach range from 45.9% to 49.1% when evaluated against the exact baseline. This self-benchmarking setting is notably demanding because the evaluation is strictly relative, where even minor variations can prompt the judge to favor the baseline. Consequently, win rates approaching 50% imply the LLM judge could rarely distinguish the optimized model from the exact baseline. While these scores reflect a slight, expected degradation inherent to approximation, the impact on generation quality is negligible. Evaluated alongside the substantial performance gains, this reduction represents a highly favorable trade-off.

5. Conclusion

In this work, we introduced a novel framework to accelerate LLM inference by replacing the standard dense output projection with a vector index-based mechanism. This drastically reduces the compute and memory bandwidth requirements for this operation by eliminating the need to evaluate the full projection, while effectively preserving the core generative capabilities of the model. Our results show that for small models with large vocabularies, replacing the dense output layer with a vector index can increase end-to-end inference throughput by as much as 82% during single-batch inference. These results are particularly relevant for efficient on-device deployment across edge environments such as mobile, automotive, and industrial systems.

Currently, the primary limitation of our approach is its reliance on CPU execution. Although the underlying inner product calculations are parallelizable, the fundamentally sequential nature of graph-based index traversal presents a significant bottleneck for GPU architectures. Consequently, future work will focus on developing GPU-accelerated vector indices and extending this method to support quantized data types. Finally, the relative benefit may decrease for heavily quantized dense baselines or large batch sizes, where optimized GEMM kernels better amortize memory access.

Broader Impact This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., Lomeli, M., Hosseini, L., and Jégou, H. The faiss library. 2024.

Fan, A., Lewis, M., and Dauphin, Y. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 889–898, 2018.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Google DeepMind. Introducing gemma 3 270m: The compact model for hyper-efficient ai. Google for Developers Blog, August 2025. URL https://developers.googleblog.com/en/introducing-gemma-3-270m/.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

Indyk, P. and Motwani, R. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pp. 604–613, 1998.

Jegou, H., Douze, M., and Schmid, C. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

Joulin, A., Cissé, M., Grangier, D., Jégou, H., et al. Efficient softmax approximation for gpus. In *International conference on machine learning*, pp. 1302–1310. PMLR, 2017.

Li, X., Zhang, T., Dubois, Y., Taori, R., Gulrajani, I., Guestrin, C., Liang, P., and Hashimoto, T. B. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 5 2023.

Malkov, Y. A. and Yashunin, D. A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018.

Nguyen, M. N., Baker, A., Neo, C., Roush, A., Kirsch, A., and Shwartz-Ziv, R. Turning up the heat: Min-p sampling for creative and coherent llm outputs. *arXiv preprint arXiv:2407.01082*, 2024.

Ootomo, H., Naruse, A., Nolet, C., Wang, R., Feher, T., and Wang, Y. Cagra: Highly parallel graph construction and approximate nearest neighbor search for gpus. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 4236–4247. IEEE, 2024.

Shrivastava, A. and Li, P. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). *Advances in neural information processing systems*, 27, 2014.

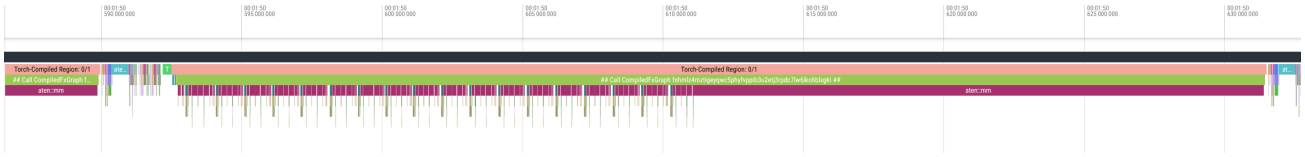
Singh, A., Fry, A., Perelman, A., Tart, A., Ganesh, A., El-Kishky, A., McLaughlin, A., Low, A., Ostrow, A., Ananthram, A., et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

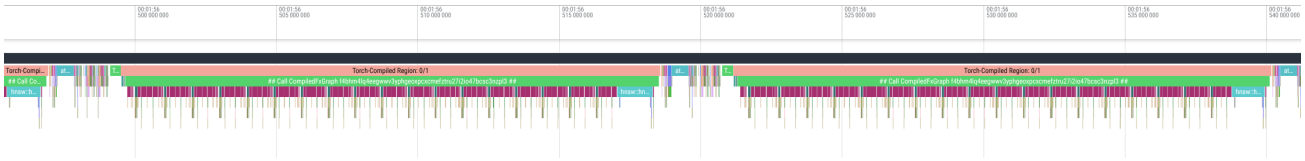
Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36: 46595–46623, 2023.

A. Profiler Traces



(a) Dense Output Projection



(b) Vector Index Embedding

Figure 3. Profiler traces comparing the standard output projection (top) against the vector index embedding (bottom) for the Gemma 3 270M model at a batch size of 1. Both plots share the same timescale and show the final tokens of a 64-token generation. In the baseline trace (a), the per-token latency is heavily dominated by the large matrix multiplications of the output projection. Our proposed vector index embedding (b) eliminates this bottleneck by replacing the dense projection with an HNSW index search. This search takes only a fraction of the time, visible as the narrow blue segment at the end of the decoding step, allowing the model to decode approximately $2\times$ faster than the baseline.