

Machine learning and high dimensional vector search

Matthijs Douze, Meta FAIR

Abstract

Machine learning and vector search are two research topics that developed in parallel in nearby communities. However, unlike many other fields related to big data, machine learning has not significantly impacted vector search. In this opinion paper we attempt to explain this oddity. Along the way, we wander over the numerous bridges between the two fields.

1 Introduction

Most high-dimensional vector search methods are based on statistical tools, signal processing approaches or graph traversal algorithms. Statistical tools include random projections [15], dimensionality reduction (PCA and the SVD). Signal processing is employed primarily to compress vectors with quantization [4, 22, 30] Most recent indexing methods are rely on graphs [3, 11, 34, 49] that are built with graph traversal heuristics.

Vector search (VS) is used in machine learning (ML) for training data deduplication [39] and searching ML embeddings [5, 28]. Therefore, there are many research teams around the world that are competent in both fields.

In their seminal work *The case for learned indexing structures* [32], Kraska et al. introduced a series of ML based algorithms to speed up classical indexing structures like hash tables and B-trees. These structures are classical building blocks of databases, e.g. a B-tree can be seen as a one dimensional VS index. They hoped to open “an entirely new research direction for a decades old field”.

However, 7 years later, it is striking that ML has had very little impact on VS. No ML based method appears in the current VS benchmarks at any scale [2, 46, 47]. The most advanced ML tool that is widely used for VS is the k-means clustering algorithm. There is no deep learning, no use of high-capacity networks.

In the following, we attempt to explain why this is the case. We work out a typical use case of VS (Section 2), then expose the fundamental limits of ML for VS (Section 3); in Section 4, we review interesting applications in the other direction: VS for ML.

2 Information retrieval with machine learning

Let’s start from a typical application of vector search: image retrieval. We manage a collection of N images $C = \{x_1, \dots, x_N\}$, where N is large and possibly growing. From a given query image q , the objective is to find images in C that are relevant to it, for example because they represent the same object.

N -way classifier. The most straightforward ML approach for this is to train a N -way classifier $f_1(q) \in [0, 1]^N$ that outputs a scalar score close to 1 for the images that are relevant and close to 0 for

the others. The best matching item from the collection can be found without even accessing the images after the training phase:

$$I_1 = \operatorname{argmax} f_1(q). \quad (18)$$

The model could be any standard image classification model, convolutional or transformer-based [19, 25]. Note that processing images is inherently slow, since the raw pixels must be converted into semantically relevant information, which requires several neural net layers. More importantly, the model’s size needs to be proportional to the collection size N , and it is impossible to add or remove images from the collection after the model was trained.

Pairwise comparison. Another approach is to design a function f_2 that, given (x, x') returns a score that is high if x' relevant to x and low otherwise. At search time, f_2 is used to compare the query q with all the images from the collection (which must be accessible):

$$I_2 = \operatorname{argmax}_{i=1,\dots,N} f_2(q, x_i). \quad (19)$$

The usual learning machinery can be deployed to train this function: given a training set of matching and non-matching images, the model is optimized to return 0 or 1 with, for example, a binary cross-entropy loss.

This resolves the problem of the model size, that remains fixed, and adding/removing from the collection is painless. The issue with this approach is that the model f_2 is still expensive to evaluate, and querying one image requires N evaluations (forward passes) of f_2 . Therefore it is not tractable beyond a few hundred images.

Embeddings. A way to avoid this is to force f_2 to decompose as:

$$f_3(x, x') = S(E(x), E(x')), \quad (20)$$

where E , the embedder (or feature extractor) is a function that computes a vector in d dimensions from an image, and $S : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is a similarity function between embeddings. This is akin to kernel methods [6, chapter 7], and sometimes called a “two-tower” network.

The feature extraction function E can be costly to evaluate, but it is performed beforehand, when the image is added to the collection: $(e_1, \dots, e_N) = (E(x_1), \dots, E(x_N))$ is precomputed. At search time, we perform only simple vector comparisons with all N :

$$I_3 = \operatorname{argmax}_{i=1,\dots,N} S(E(q), e_i) \quad (21)$$

From a ML point of view, since we impose a constraint on the function f_3 , it is bound to be *less accurate* than the function f_2 , that performs a direct comparison between items: there is an information bottleneck.

The functions S and E are bound by an *embedding contract* [20], that states that E should generate embeddings such that the comparison with S is semantically meaningful.

Embedding extraction. There is a vast literature on embedding extraction for images. It started in the pre-deep era [31, 40]. Later, image embedding models were first sampled from some layer of a classification model [5], then trained specifically using a clustering supervision [9], or variants of contrastive learning [10, 12, 16, 29]. The embeddings can be specialized for different granularities of image matching: from class-level via instance-level to copy-level [7, 41].

Embeddings can be extracted from other modalities as well, with slight variations. For text embeddings (like BERT [17, 27]) the size of the embedding vector is usually *larger* than the original text – embeddings are not guaranteed to compress the items they are computed from. These text embeddings are the basis for the retrieval augmented generation for LLMs [28]. Text and image embeddings can be matched together as with the CLIP embeddings [43]. Recommendation systems are often based on two-tower networks, with different embedding functions for the users and the items to recommend [38].

3 Vector search with machine learning

Performing the embedding search of Equation 21 is an operation whose complexity is $\mathcal{O}(N \times d)$ for most similarity functions. The research on vector search aims at keeping this complexity under control when N becomes large. In the process, some accuracy can be sacrificed, and the brute force search of Equation 21 becomes an Approximate Nearest Neighbor Search (ANNS) task.

Vector search is a linear classifier. Let’s assume that the similarity function, in Equation (21) is a dot product¹. In that case, the operation to be performed is a matrix-vector multiplication between $E(q)$ and the matrix that stacks embeddings $[e_1, \dots, e_N]$. This is equivalent to a linear layer of size $N \times d$. This means that vector search can be solved with the simplest of machine learning tools: a linear classifier of size N .

This equivalence between vector search and classification is used for k-NN classifiers [10, 36, 42] (but k-NN classifiers are less accurate than training linear classifiers end-to-end).

A fundamental divergence. ANNS and ML both strive to optimize an accuracy objective. However, vector search starts where machine learning stops. The problems to handle when training a machine learning model are the train-test discrepancy, overfitting, regularization, etc. Resource utilization is a second thought. Vector search does not have any of these problems, because computing the perfect result is straightforward (Equation 21). The problem of vector search is how to compute the result accurately with limited resources.

This explains why directly applying ML does not solve VS: as soon as ML starts to apply operations at scale N , it is already slower than performing brute-force VS. This observation was already done in [32]. Their approach is to break down the indexing structures into a distribution modeling part, that can be solved by ML and the indexing structure itself, that scales to size N .

ML for vector distribution modeling. A direct application of this approach for VS is to train a transformation that maps vectors to a space that is easier to index [45]. In this approach the input vectors are transformed into a more uniform space while maintaining neighborhood relations; a lattice quantizer is then used to encode the uniform vectors.

Modeling the vector distribution can be used to apply lossy compression to vectors, which is equivalent to quantization. The k-means algorithm is a strong baseline for quantization because it obeys Lloyd’s conditions and is efficient. In fact, the way VQ-VAEs are trained, with an exponential moving average, is similar to online k-means [44].

However, to scale it to more accurate compression, k-means must be applied several times (multi-codebook quantization), which yields less optimal representations like product quantization [30] or additive quantization [4].

¹This is without loss of generality: most standard distances can be reduced to computing dot products in a transformed space [20].

Deep learning models can be applied to improve multi-codebook quantization. This is done in the UNQ quantizer [37], where a learned transformation maps the vectors to a space where it is easier to apply quantization. In the QINCo series of works [26, 50] the codebooks of a residual quantizer are adapted using a neural net to better approximate the vector distribution. One interesting work shows that, given a quantizer, it is possible to train a decoder that improves its accuracy [1].

One important family of VS methods is to partition the vector space. At search time, only a subset of partitions are visited. Usually this partitioning is based on k-means [30]. However, the partitions can also be predicted from a vector with a classifier [18, 35].

Discussion. These approaches show that there are operating points where ML can help improving VS. However, in many cases, the practitioner hits hard limits. The first limit is that the complexity of the quantization must remain below that of a brute-force vector search. The second limit is that often, increasing the capacity of the model does not improve the fit of the vector distribution: a shallow model (or k-means itself) is hard to outperform with deeper models. Why this happens is an open research question.

4 Machine learning with vector search

We review methods where VS is included within deep learning models.

Network compression. In the previous section, we showed the equivalence between VS and a linear layer. Therefore, it is natural to apply vector search techniques to these layers, especially when the weight matrix is skinny ($N \gg d$).

Techniques originally developed for VS have been applied to compressing linear layers. Product quantization is used to compress convolutional neural nets [23] and language models [48], with or without retraining the model (quantization-aware training or post-training quantization). This is especially useful for large recommendation models where most of the network parameters are in embedding tables [38], i.e. $N \gg d$. In some settings, a compressed linear operator can be applied to its input without decompression [24].

Attention operators. Large language models are built around an attention operator that functions as an associative memory: a query vector is matched with embeddings of all the previous tokens of the prompt. This is basically a vector search task.

When the prompt size increases (the “long context” setting, $N \gg d$), it becomes beneficial to use ANNS. This includes compression [21], and also non-exhaustive VS with partitions [8], random projections [13] or graphs [33]. One difficulty of attention operators is that the queries and database vectors have different distributions. For partition-based VS, this can be addressed by training different partition classifiers for queries and database vectors [35].

Discussion. VS has a potential to be an accelerator for ML models, whenever the input needs to be compared to a large number of vectors. The limit is that ML is typically run on hardware that is so efficient in doing exact VS (matrix multiplication) [14] that the size above which it makes sense to use ANNS is large (currently around $N = 10^5$) and growing.

5 Conclusion

We reviewed several use cases where VS and ML are interlinked. It is clear that ML cannot replace the VS data structures, but it can help modeling the data distributions. One direction that deserves more exploration is leverage ML to better build graph-based indexes. In the other direction, there are tasks in ML where VS can be very useful, in particular to solve large classification problems and to perform attention on long contexts.

The case for learned indexing structures [32] speculated that the increase of compute capacity would make ML models more amenable to indexing methods. What happened is rather that for many use cases of VS, compute has become so cheap that they can just be solved in brute force.

6 Bio

Matthijs Douze is a research scientist at Meta Fundamental Artificial Intelligence Research (FAIR) in Paris, France. His current research interests include computer vision, vector search and watermarking. Prior to that, he has been with INRIA for 10 years, working on image and video indexing and classification, real-time 3D reconstruction and video alignment. He obtained a PhD from the university of Toulouse.

References

- [1] Kenza Amara, Matthijs Douze, Alexandre Sablayrolles, and Hervé Jégou. Nearest neighbor search with compact codes: A decoder perspective. In *ICMR*, 2022.
- [2] Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems*, 87:101374, 2020.
- [3] Ilias Azizi, Karima Echihabi, and Themis Palpanas. Graph-based vector search: An experimental evaluation of the state-of-the-art. *Proc. ACM Manag. Data* 3(1), 2025.
- [4] Artem Babenko and Victor Lempitsky. Additive quantization for extreme vector compression. In *CVPR*, 2014.
- [5] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part I 13*, pages 584–599. Springer, 2014.
- [6] Francis Bach. *Learning theory from first principles*. MIT press, 2024.
- [7] Maxim Berman, Hervé Jégou, Andrea Vedaldi, Iasonas Kokkinos, and Matthijs Douze. Multigrain: a unified image embedding for classes and instances. *arXiv preprint arXiv:1902.05509*, 2019.
- [8] Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew R Gormley. Unlimiformer: Long-range transformers with unlimited length input. *arXiv preprint arXiv:2305.01625*, 2023.
- [9] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149, 2018.
- [10] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

- [11] Meng Chen, Kai Zhang, Zhenying He, Yinan Jing, and X Sean Wang. Roargraph: A projected bipartite graph for efficient cross-modal approximate nearest neighbor search. *arXiv preprint arXiv:2408.08933*, 2024.
- [12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.
- [13] Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Jianyu Zhang, Niklas Nolte, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. Magicpig: Lsh sampling for efficient llm generation. In *ArXiv*, 2024.
- [14] Felix Chern, Blake Hechtman, Andy Davis, Ruiqi Guo, David Majnemer, and Sanjiv Kumar. Tpu-knn: K nearest neighbor search at peak flop/s. *Advances in Neural Information Processing Systems*, 35:15489–15501, 2022.
- [15] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, 2004.
- [16] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4690–4699, 2019.
- [17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [18] Yihe Dong, Piotr Indyk, Ilya Razenshteyn, and Tal Wagner. Learning space partitions for nearest neighbor search. *arXiv preprint arXiv:1901.08544*, 2019.
- [19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.
- [20] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *arXiv preprint arXiv:2401.08281*, 2024.
- [21] Vage Egiazarian, Andrei Panferov, Denis Kuznedeleev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. *arXiv preprint arXiv:2401.06118*, 2024.
- [22] Jianyang Gao and Cheng Long. Rabbitq: Quantizing high-dimensional vectors with a theoretical error bound for approximate nearest neighbor search. *Proceedings of the ACM on Management of Data*, 2(3):1–27, 2024.
- [23] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.
- [24] Zaid Harchaoui, Matthijs Douze, Mattis Paulin, Miroslav Dudik, and Jérôme Malick. Large-scale image classification with trace-norm regularization. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3386–3393. IEEE, 2012.

- [25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [26] Iris A.M. Huijben, Matthijs Douze, Matthew J. Muckley, Ruud J.G. van Sloun, and Jakob Verbeek. Residual quantization with implicit neural codebooks. In *International Conference on Machine Learning (ICML)*, 2024.
- [27] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*, 2021.
- [28] Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*, 2020.
- [29] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2020.
- [30] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *PAMI*, 2010.
- [31] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3304–3311. IEEE, 2010.
- [32] Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proceedings of the 2018 international conference on management of data*, pages 489–504, 2018.
- [33] Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chenguidong Zhang, Bailu Ding, Kai Zhang, et al. Retrievalattention: Accelerating long-context llm inference via vector retrieval. *arXiv preprint arXiv:2409.10516*, 2024.
- [34] Yu A Malkov and Dmitry A Yashunin. 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.
- [35] Pierre-Emmanuel Mazaré, Gergely Szilvasy, Maria Lomeli, Francisco Massa, Naila Murray, Hervé Jégou, and Matthijs Douze. Inference-time sparse attention with asymmetric indexing, 2025.
- [36] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2624–2637, 2013.
- [37] Stanislav Morozov and Artem Babenko. Unsupervised neural quantization for compressed-domain similarity search. In *ICCV*, 2019.
- [38] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091*, 2019.

- [39] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024.
- [40] Florent Perronnin, Yan Liu, Jorge Sánchez, and Hervé Poirier. Large-scale image retrieval with compressed fisher vectors. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3384–3391. IEEE, 2010.
- [41] Ed Pizzi, Sreya Dutta Roy, Sugosh Nagavara Ravindra, Priya Goyal, and Matthijs Douze. A self-supervised descriptor for image copy detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14532–14542, 2022.
- [42] Hang Qi, Matthew Brown, and David G Lowe. Low-shot learning with imprinted weights. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5822–5830, 2018.
- [43] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- [44] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.
- [45] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. Spreading vectors for similarity search. *ICLR*, 2019.
- [46] Harsha Vardhan Simhadri, Martin Aumüller, Amir Ingber, Matthijs Douze, George Williams, Magdalen Dobson Manohar, Dmitry Baranchuk, Edo Liberty, Frank Liu, Ben Landrum, et al. Results of the big ann: Neurips’23 competition. *arXiv preprint arXiv:2409.17424*, 2024.
- [47] Harsha Vardhan Simhadri, George Williams, Martin Aumüller, Matthijs Douze, Artem Babenko, Dmitry Baranchuk, Qi Chen, Lucas Hosseini, Ravishankar Krishnaswamy, Gopal Srinivasa, et al. Results of the neurips’21 challenge on billion-scale approximate nearest neighbor search. In *NeurIPS 2021 Competitions and Demonstrations Track*, pages 177–189. PMLR, 2022.
- [48] Pierre Stock, Armand Joulin, Rémi Gribonval, Benjamin Graham, and Hervé Jégou. And the bit goes down: Revisiting the quantization of neural networks. *arXiv preprint arXiv:1907.05686*, 2019.
- [49] Suhas Jayaram Subramanya, Rohan Kadekodi, Ravishankar Krishaswamy, and Harsha Vardhan Simhadri. Diskann: Fast accurate billion-point nearest neighbor search on a single node. In *NeurIPS*, 2019.
- [50] Théophane Vallaëys, Matthew Muckley, Jakob Verbeek, and Matthijs Douze. Qinc2: Vector compression and search with improved implicit neural codebooks. In *ICLR*, 2025.