Is Quantized ANN Search Cursed? Case Study of Quantifying Search and Index Quality

Gylfi Þór Guðmundsson(✉) and Björn Þór Jónsson

Reykjavik University, Menntavegur 1, 102 Reykjavik, Iceland
{gylfig,bjorn}@ru.is

Abstract. Traditional evaluation of an approximate high-dimensional index typically consists of running a benchmark with known ground truth, analyzing the performance in terms of traditional result quality and latency measures, and then comparing those measures to competing index structures. Such analysis can give an overall indication of the suitability of the index for the application that the benchmark represents. When the index inevitably fails to return the sought items for some queries, however, this methodology does not help to explain why the index fails in those cases. Furthermore, when considering many different parameter settings, the process of repeatedly indexing the entire collection is prohibitively time-consuming. In this paper, we define three causes for failures in hierarchical quantized search. We show that the two failure cases that relate to the index can be evaluated and quantified using only the index structure and ground-truth data. In our evaluation, we use eCP, a lightweight algorithm that builds the index hierarchy top-down a priori without any costly segmentation of the dataset, and show that significant insight can be gained into the quality of the index structure, or lack thereof.

Keywords: High-dimensionsional indexing · Hierarchical vectorial quantization · Evaluation methodology

1 Introduction

Approximate near-neighbour search techniques (ANN) are used to accomplish efficient similarity search over a large volume of high-dimensional data. To evaluate such a search engine, we need data and queries where we know the "correct answers," also known as the *ground-truth*. The evaluation process can then be best described as "proof by doing" as we first instantiate the search engine by indexing the evaluation data and then use the queries in the ground-truth to run ANN search and calculate metrics such as *precision*, *recall* and *mAP*. This "full-scale" approach is not only time consuming but also inadequate, as when the search fails those metrics neither explain why nor quantify by how much.

Today's state-of-the-art large scale ANN algorithms are based on proximity graph algorithms [7], which are very costly to construct. Compression-based

O. Pedreira and V. Estivill-Castro (Eds.): SISAP 2023, LNCS 14289, pp. 163–170, 2023.
https://doi.org/10.1007/978-3-031-46994-7_14

techniques [6] are also popular, but they struggle with billions of features. In comparison, older tree-like hierarchical vectorial quantizers (HVQ) can be constructed quickly, have a lower memory footprint, and have been used at Web-Scale, but they give lower quality results. It is therefore of interest to study where HVQ approaches fail, to understand whether they can be improved to be more competitive.

The goal of the quantizer is to partition the data collection C into small segments. Let us define r as the representative for each such segment and R as the set of all representatives. At *indexing time*, segmentation is based on finding the most similar $r \in R$ for each vector v in the dataset. Then, at *search time*, one (or a few) most similar $r \in R$ are identified and only those segments make up the search result. Let us define the truly most similar representative $r \in R$ as r^t and a ranked list result as L^R. Finding r^t requires a scan over R, but as R is often large, an ANN search using a tree-like hierarchy is used instead. Let us denote the most similar representative found via the index as r^i and a ranked list result as L^i.

For a query q, let us define N_q as the set of its true neighbours. N_q can be said to form a *neighbourhood* and the density of data in this region, e.g. whether the near-neighbour relation is reciprocated or not, is a good indicator of how difficult it is to match q with its N_q. A "good" neighbourhood is when q and its N_q are dense and well represented, i.e., N_q falls into one (or a few) segment(s) similar to q. A "bad" neighbourhood is either a dense grouping that has no good representation, or a neighbourhood so sparse that some of N_q fall into segments that are far down the list of representatives most similar to q.

We can now identify three possible causes for HVQ search failures:

1. **Representation Failure (RF)** occurs when no representative $r \in R$ is a clear best option for a given neighbourhood, i.e., for both q and its N_q. In such a case, the vectors are likely to be fragmented over many segments, causing the ANN search to fail despite correctly indexing the vectors, i.e., for each $n \in N_q$, $r_q^t \neq r_n^t$.
2. **Index Hierarchy Failure (IHF)** occurs when the index is not assigning vectors correctly. That is, for a given vector v the most similar segment identified by the index, r_v^i, does not match the most similar representative r_v^t found by scanning all segment representatives R. Note that IHF can occur both when indexing the data and when using the index to identify what segment(s) should be used in the ANN search.
3. **Segment Search Failure (SSF)** occurs when the ANN search fails to find q's most similar $v \in C$ despite looking in the right segment. This can happen in systems that compress, aggregate, or otherwise approximate the search process inside segments.

Evaluating RF is all about looking at the neighbourhood. Using the ground truth for query q we obtain the ranked list of most similar segment representatives for q, denoted by L_q^R. For each neighbour n in N_q, from the ground truth, we then get the most similar segment r_n^t. We can now quantify RF by examining how far down the list L_q^R we find r_n^t. Note that getting the truly most similar

segments is costly as it requires a full scan of all index segments $r \in R$. To evaluate IHF, we need to know a) the index-assigned segment r_v^i and b) the ranked list of most similar segments L_v^R for the vector v. Quantifying IHF is then based on how far down the list L_v^R we find r_v^i. If neither RF nor IHF occurred, but the ground truth indicates that the result is not correct, then SSF has occurred.

In this paper, we report on a case study of the eCP algorithm, a cluster-based HVQ. Since eCP reads whole clusters, SSF does not occur. Our results indicate that RF is much more prevalent with eCP than IHF. Furthermore, we demonstrate that while enhancing the representation with k-Means clustering can be done 50x faster by using only the index, the impact on RF is minimal.

The remainder of this paper is structured as follows. In Sect. 2 we explain our evaluation methods and metrics. Section 3 then present the results of our experiments, and finally we conclude the paper in Sect. 4.

2 Evaluating Indexed ANN Search Failures

Our aim is to understand and quantify how and why the ANN search, using HVQ, fails. The algorithm we choose to evaluate in this initial case study is called Extended Cluster Pruning (eCP) [2]. eCP is a simple yet versatile HVQ, which was initially developed for content-based image retrieval using SIFT features [4] and has been extensively evaluated at large scale [3]. The index construction is very efficient as the index is built a priori in a single top-down pass using randomly sampled vectors from the dataset. This is a great advantage as that means we can evaluate the performance of eCP using just the index and a ground-truth benchmark, without doing any of the costly segmenting. To be clear, eCP makes no effort to improve on the randomly chosen segment representatives. To compensate for potentially poor representation, however, it does support *search-expansion* both inside the index hierarchy as well as for the number of segments to retrieve and scan. In this case study, we consider search expansion at query time, processing SE clusters to find near neighbours. We now explain how we evaluate and quantify the possible causes for HVQ search failures. Since eCP is not susceptible to SSF, we can also estimate recall using only the index structure.

2.1 Evaluating Representation Failure (RF)

To evaluate the RF we need a neighbourhood (q and N_q, both obtained from the ground truth) and eCP's set of representatives R. By scanning R we derive L_q^R, the ranked list of q's most similar representatives, as well as the optimal assignment r_n^t for each $n \in N_q$. If r_n^t is at the top of L_q^R, we know that q and n are in the same segment. How far down the list we find each r_n^t tells us how far the ANN search will need to expand such that it finds that neighbour. Remember that q has many neighbours, so we choose to average this rank, creating a metric we call *Average Rank* (AR). Furthermore, we also add a second metric that is based on counting how many neighbours have a rank lower than some value X. We call his metric *Optimal Recall* (OR) as if we set X equal to the SE parameter used, it tells us exactly what portion of N_q we can hope to find in an ANN search.

2.2 Evaluating Index Hierarchy Failure (IHF)

To evaluate and quantify IHF, we consider a set of vectors v and, as before, we scan R to derive L_v^R. Then, using the index structure, we derive the assigned representative r_v^i, for a given SE setting. The metrics we use to quantify IHF is based on looking at where in the ranked list L_v^R we find the index-assigned r_v^i. If $r_v^i = r_v^t$, where r_v^t is the segment at the top of L_v^R, then the index assignment is optimal. Otherwise, we can use the rank of r_v^i in L_v^R to measure how far off the index assignment is. In our experiments, we report a) how many optimal assignments we have and b) how many are within SE of the optimal. This gives a clear indication of how well the index is doing.

2.3 Evaluating Recall in Absence of Segment Search Failure (SSF)

eCP is not susceptible to SSF, as the ANN search identifies and then scans whole segment(s) to create the final k-NN result. This allows us to calculate what the recall of eCP's ANN search would be using only the index and ground-truth data. For each query q, we obtain L_q^i, the ranked list of the SE most similar representatives for segments that should be scanned. By looking up each $n \in N_q$ from the ground truth to get r_n^i, we can compute recall by counting how many of the r_n^i assignments are anywhere in L_q^i.

3 Evaluation

3.1 Setup

Dataset and Ground-Truth: We use BIGANN [1] in our evaluation. The full set has 1B 128-dimensional SIFT features, but we use the 100M subset as this is sufficient to build a large eCP index. The original ground-truth consists of 10k queries with 1k NNs for each, but to make the computational load manageable we use a subset that consists of the first 50 queries along with all of their 1k near neighbours, for 50k vectors in total. We should note that BIGANN is a difficult dataset. The baseline recall given for track 1 of the *BigANN benchmark* using the BIGANN data is 63.5% recall and the best competitors got 71.4% [7].

Indices and Search Settings: Guidelines exist regarding picking the "right" number of clusters to build an eCP index. As we do not intend to build the full search engine, however, we build four different 3-level deep indices using R= 40k, 80k, 160k and 320k vectors to study the impact of index size. The R vectors are randomly sampled from the 100M SIFT subset. In all experiments we perform search expansion, with maximum $SE = 20$, but in the analysis we consider the impact of varying the SE parameter from 1 to 20..

Hardware and Software: Experiments are all run on a single machine with an Intel i9-7900X CPU, 64 GiB of RAM, and a 1 TB Samsung 960 Pro SSD disk. The OS is Ubuntu 18_04 and we use Spark 2.4.5 with Java–openJDK version 11.0.17 and Scala version 2.11.12. Note that the original SIFT features are 128 dimensions of unsigned 8 bits (0–255), but since Java does not support unsigned data, we scale the values to be *Byte* (-128 to 127).

Table 1. Evaluation of Representation Failure (RF). Average Rank (AR) is the average location of r_n^t in L_q^R across all 50k queries. Optimal Recall (OR) counts how many of the 1k neighbours have a rank lower than $SE = 20$, averaged across 50 queries

Index	AR	OR
40k	55.76	602.54
80k	88.78	547.42
160k	132.12	489.96
320k	222.35	408.82

3.2 Experiment 1: Representation Failure (RF)

In this experiment we use the 50 queries, each having 1k NNs, and evaluate on all 4 index sizes. In Sect. 2.1 we defined the two metrics, AR and OR, that we report. The results are presented in Table 1.

Here, AR is the average rank of the N_q=1k vectors, averaged over all 50 queries. The 40k index has AR of ∼56 while 320k has AR of ∼222. This implies that adding more clusters seems to give the "bad" neighbourhoods more options, spreading the NNs even further, making it even harder to match q with its N_q.

The OR metric is even more interesting as it essentially indicates the best-case recall for a given search expansion. Here, we average the OR metric over the 50 queries, using $SE = 20$. The 40k index scores ∼603 out of 1k queries, while the 320k scores ∼409, which can be read as "optimal recall" of 60.3% and 40.9% respectively.

While interesting in itself, the true value of the OR is that it allows us to put the ANN search results of later experiments into context. The main conclusion we can draw from this experiment, however, is that the RF is significant.

3.3 Experiment 2: Index Hierarchy Failure (IHF)

Here we focus on evaluating whether the index is able to assign vectors correctly. From the index, we retrieve for each v of the 50k NN vectors r_v^i and we also scan R to get L_v^R, each vector's ranked list of most similar representatives. The results are presented in Fig. 1.

The x-axis indicates the search expansions used, while the y-axis shows the average across the 50k queries. As was stated in Sect. 2.2 we report two metrics. The first counts how many of the index assignments are optimal, i.e., $r_v^i = r_v^t$, which is shown with the solid lines. The second metric, reported with the dotted lines, counts how many index assignments are within SE of the optimal

Our first observation is that the results are almost identical despite the largest index (320k) having 8 times more segments than the smallest (40k). Second, we observe that without any search expansion ($SE = 1$), the indices are only correctly assigning about 25–30% of vectors. As we expand the search, this ratio grows and at $SE = 5$, the indices already retrieve over 50%. The reason why the

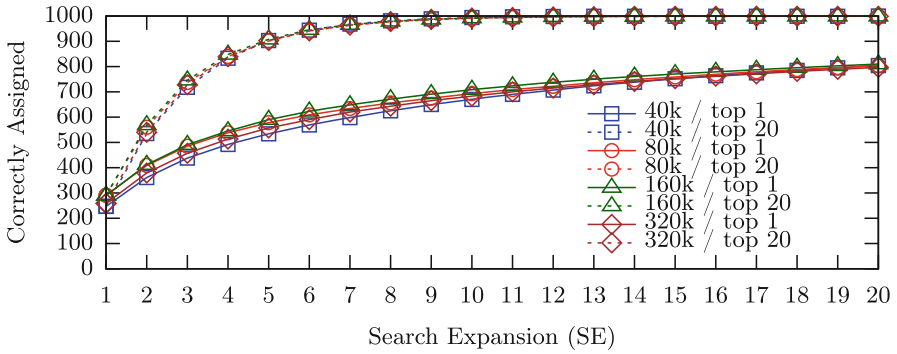

Fig. 1. Evaluation of Index Hierarchy Failures (IHF). The x-axis shows the expansion setting, SE, while the y-axis shows the correctly assigned NNs. Solid lines indicate when $r_v^i = r_v^t$ while the dotted lines indicate that r_v^i is in the top SE clusters of L_q^i

top item of the ranked list improves, is that the search expansion is applied at all levels of the index, reducing the branching errors at the upper levels.

Turning to the dotted lines, which show the neighbours correctly located in the top SE clusters of the ranked list, we observe that they grow even faster. For $SE = 1$, they are identical, but at $SE = 5$ more than 90% of assignments are found. This means that at $SE = 5$, 50% of the neighbours are correctly assigned and another 40% is within 5 of optimal assignment. What remains to be seen is whether this is good enough for the ANN search.

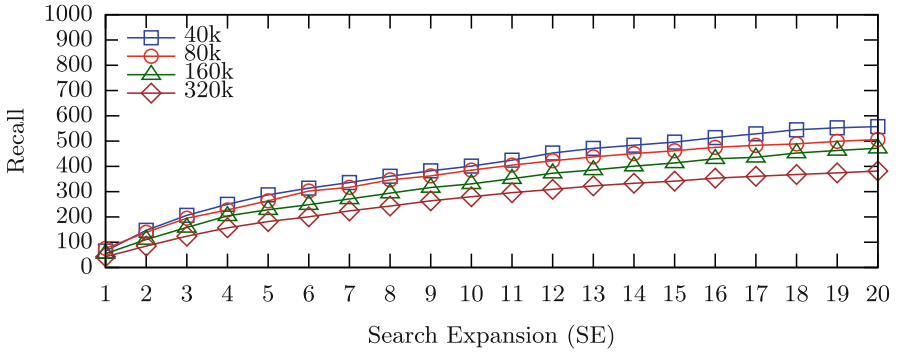

Fig. 2. Evaluation of Recall. The x-axis shows the search expansion, SE, while the y-axis shows the recall, averaged across the 50 queries

3.4 Experiment 3: Estimating Recall

Having investigated RF, producing a baseline for optimal recall, and investigated IHF, we can now check how well the eCP indices actually do in an ANN search.

As said in Sect. 2, this is done by searching for both q and N_q and checking whether the $n \in N_q$ assigned representative r_n^i is in the ranked list of similar representatives for the original query, L_q^i, using search expansion parameter SE. The results are plotted in Fig. 2.

We observe that, as expected, the recall degrades with index size and at first glance the results are not impressive. The 40k index peaks at only 555 matches (∼56% recall) and the 320k at 382 (∼38%). That is well below the 63.5% baseline from [7]. But if consider the OR metric from the previous experiment (at our maximum expansion), we observe that 40k index is in fact scoring ∼56% out of a maximum of ∼60% and the 320k index at ∼38% out of a maximum of ∼40.9%. From this we can assert that despite eCP's indexing hierarchy being very simplistic, it is only responsible for a small fraction of the search failures.

Table 2. Evaluation of whether k-Means can improve **RF** and ANN recall. Results shown are AR, OR, time the k-Means took and search results for 40k indices after the given number of steps of k-Means

Steps	**AR**	**OR**	**Running time**	**ANN Recall**
KM0 (original)	55.76	602.54	n/a	555.26
KM1 (full)	50.30	657.76	∼50h	606.82
KM1 (index)	53.04	648.32	48m	602.20
KM2 (index)	51.14	654.26	∼2h	614.34
KM10 (index)	45.91	677.00	∼10h	606.80

3.5 Experiment 4: Can K-Means Fix RF?

A maximum recall of 60% for the 40k index cannot be called a great result. A common proposal to addressing representation is to run k-Means clustering. In this section, we evaluate the impact of this using AR and OR. The results are presented in Table 2.

The first line of the table repeats the data from previous experiments for reference, showing values for the original eCP index. The two following lines, with KM1, represent one iteration of k-Means, assigning the entire 100m dataset using the representatives only or using the index. Comparing to all representatives is more precise but requires more than 4 trillion distance calculations, which took about 50 h. In comparison, using the index to speed up the k-Means assignments is more then 10x faster, taking only about 50 min. We observe that while considering all representatives yields better results, a multi-step clustering process is infeasible with that approach even at this moderate scale.

The final two lines show the results from running 2 and 10 steps of the k-Means process, respectively. We observe that while both AR and OR improve slightly, recall eventually decreases, meaning that the positive impact of k-Means is moderate and RF remains a major issue of the eCP index.

4 Conclusions

In this paper we have investigated ANN search using a hierarchical vectorial quantizer. We have shown that we can evaluate the quality of a) the segment representatives, b) the index hierarchy and c) the ANN search for both successful (recall) and failed queries. This we can do using only the index structure and the evaluation *ground-truth* data. Using algorithms such as eCP, that build their index without any actual segmenting/clustering of the full dataset, means we can perform index evaluation with minimal effort.

When we observe poor ANN search results, such as 58% recall, the inclination is to blame the indexing hierarchy, especially when it is as simple and naive as that of the eCP algorithm. By measuring the representation failure, however, we could put the index assignments into perspective. As it turns out, the eCP index itself is only responsible for a fraction of the failed queries.

We also evaluated whether k-Means clustering could alleviate the representation issues but the results show that clustering at large scale a) is prohibitively expensive, even when using the index to speed it up 10-fold, and b) only of a marginal benefit.

What makes multi-layer similarity graph algorithms, such as HNSW [5], obtain such high-quality results is that they address the representation problem by being highly selective when picking representatives at each layer in their indexing/graph structure and by not limiting the possible search/scan to a fixed region once hierarchy is traversed. However, they pay for this ability with the added complexity, footprint, and high construction time.

References

1. Amsaleg, L., Jégou, H.: BIGANN: abillion-sized evaluation dataset, corpus-texmex.irisa.fr. Accessed 2 June 2023
2. Gudmundsson, G.Þ., Jónsson, B.Þ., Amsaleg, L.: A large-scale performance study of cluster-based high-dimensional indexing. In: Proceedings of the international workshop on Very-Large-Scale Multimedia Corpus, Mining and Retrieval (VLS-MCMR), pp. 31–36 (2010)
3. Gudmundsson, G.Þ, Jónsson, B.Þ, Amsaleg, L., Franklin, M.J.: Prototyping a web-scale multimedia retrieval service using spark. ACM Trans. Multimed. Comput. Commun. Appl. (TOMM) **14**(3s), 1–24 (2018)
4. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision (IJCV) **60**, 91–110 (2004)
5. Malkov, Y., Ponomarenko, A., Logvinov, A., Krylov, V.: Approximate nearest neighbor algorithm based on navigable small world graphs. Inf. Syst. **45**, 61–68 (2014)
6. Matsui, Y., Uchida, Y., Jégou, H., Satoh, S.: A survey of product quantization. ITE Trans. Media Technol. Appl. (MTA) **6**(1), 2–10 (2018)
7. Simhadri, H.V., et al.: Results of the NeurIPS 2021 challenge on billion-scale approximate nearest neighbor search. In: NeurIPS 2021 Competitions and Demonstrations Track, pp. 177–189. PMLR (2022)