FAISS vs. ScaNN: Similarity Search for Gene Embeddings

CIMI PATANI, GAVIN HEARNE, HARRISON MULLER, KIERAN LYNCH, SALEH REFAHI

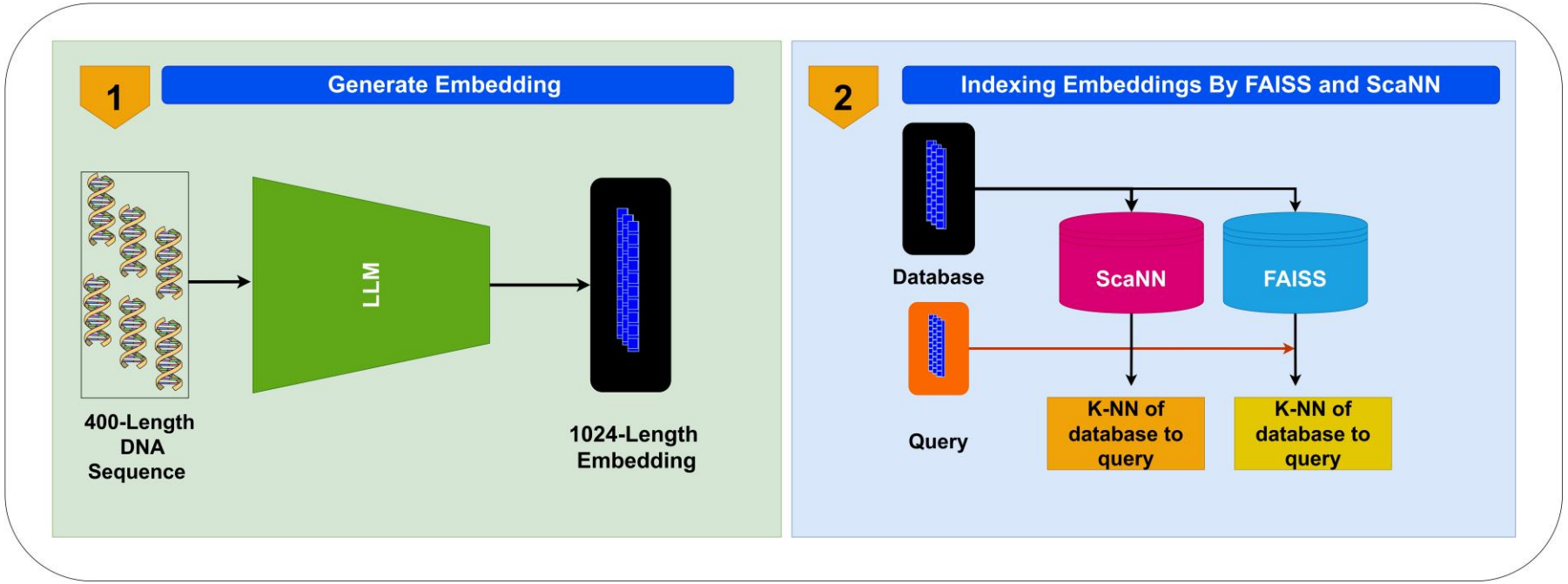
Introduction

- Alignment methods for finding similar sequences can be computationally intensive, posing challenges when applied to large datasets or when comparing sequences against extensive databases.
- In metagenomics, alignment methods encounter difficulties in detecting unknown (novel) organisms that are absent in reference databases.
- Representation learning: Researchers harness Natural Language Processing (NLP) to train machine-learning models, generating vector representations from word sequences.
- In biological sequences, each sequence is analogous to a sentence, and DNA sequences are composed of individual nucleotides.
- One of Effective extraction of meaningful sub-words from biological sequences using k-mer tokenization.
- Language modeling is a specific task in NLP where the model tries to predict a token based on surrounding tokens.

Masked Language Modeling 6-mer Representation

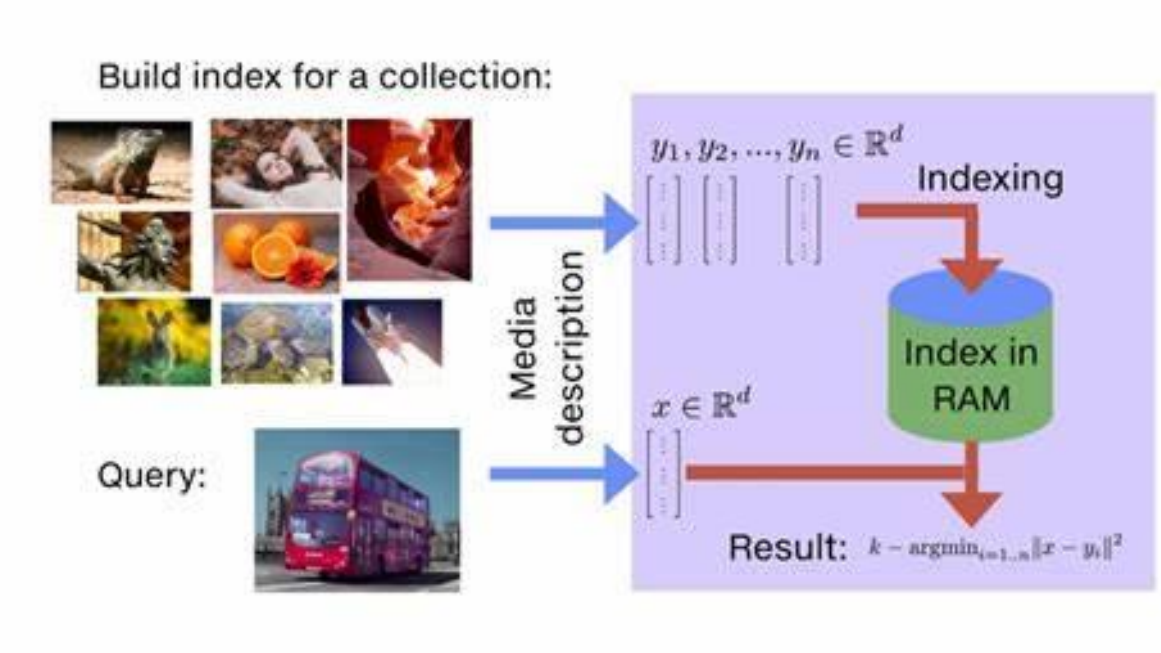
Our Approach

- RoBERTa generates 1024-length embedding vectors for 165k sequences in a Database set and 14k sequences in a Query set (7k in-domain, 7k out-of-domain).
- ScANN and FAISS are employed for efficient similarity searches on the training set's embeddings.
- Distances between test set embeddings and indexed database set are calculated using ScANN and FAISS search functions.
- In-domain and out-of-domain subsets of the test set are created, and nearest neighbors in the database set are identified based on calculated distances.
- The objective is to evaluate ScANN and FAISS efficiency in finding nearest neighbors and distances using RoBERTa-generated embeddings.



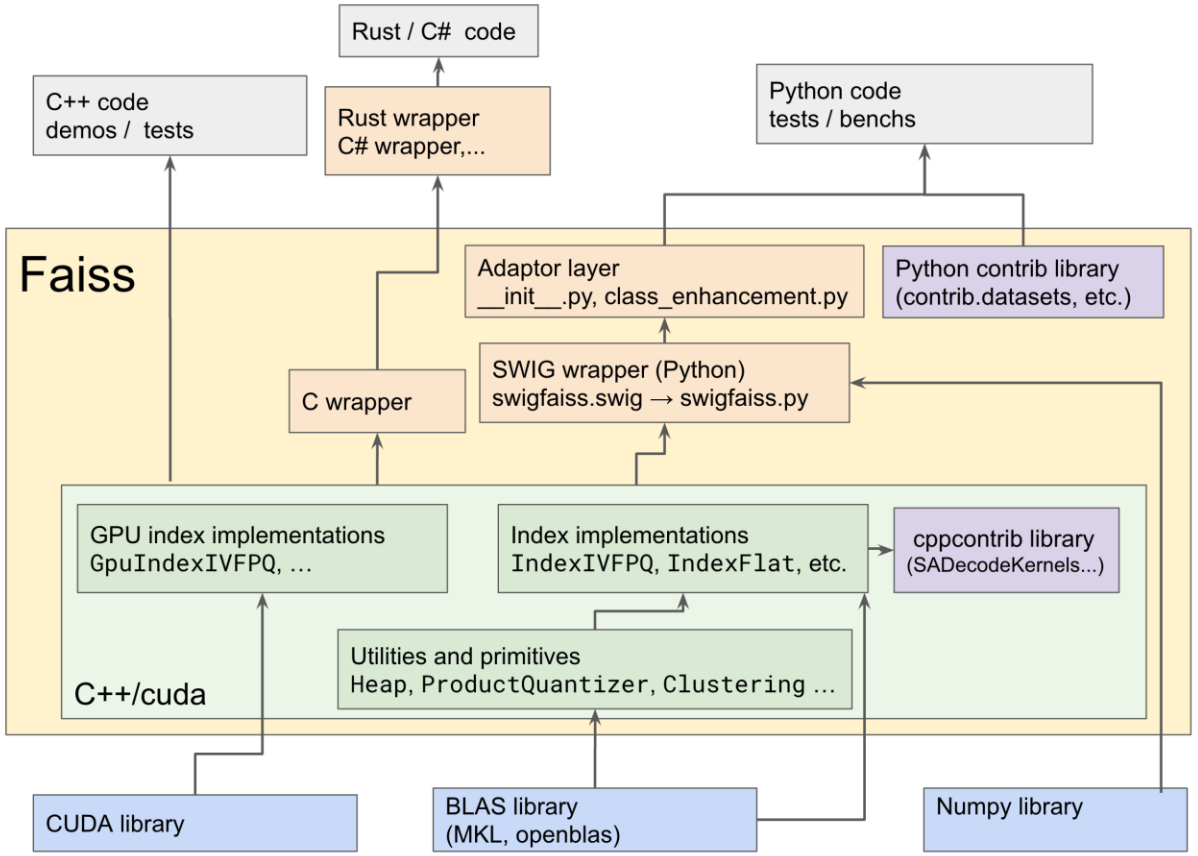
FAISS

- Industrial C++/Python library developed by Facebook AI
  - Performs efficient similarity search on embeddings
  - Generates an index from a set of vectors and queries this index to find similarities using the approximate nearest neighbor search algorithm (ANNS)
  - Use cases in trillion-scale indexing, text retrieval, and data mining



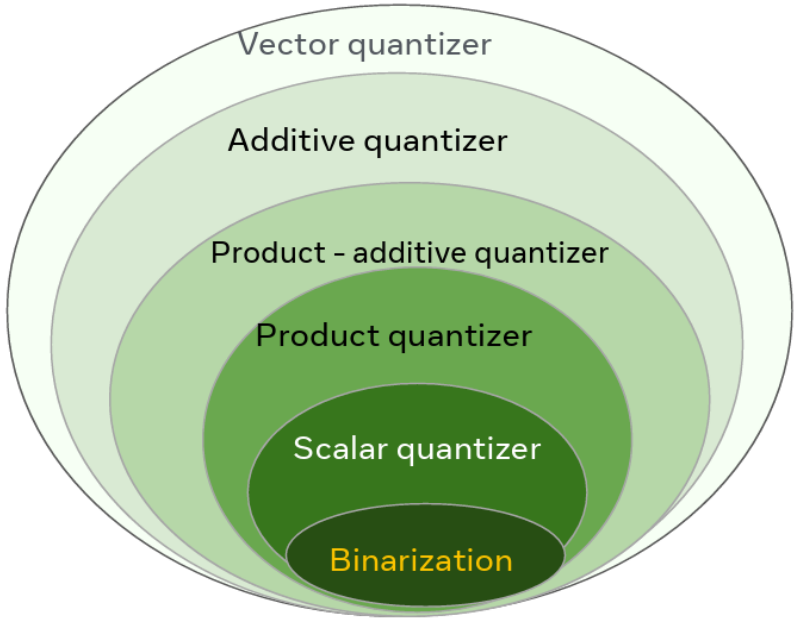
Features

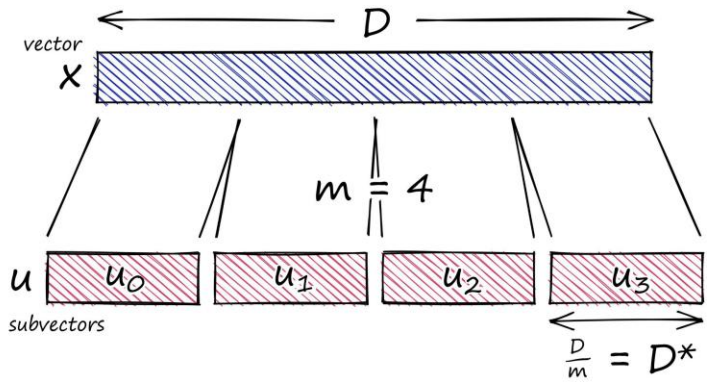
- Unique for the number of features it provides
  - Multiple indexing methods
    - Brute force (Flat), product quantization (PQ), locality-sensitive hashing (LSH), graph based (HNSW), etc.
    - Highly customizable
      - Use chain of components: pre-processing, compression, non-exhaustive search, etc.
      - User can specify which methods they want based on the dataset and desired output
  - GPU and CPU
    - FAISS can be easily transferred from the CPU to all available GPUs
    - Easy to use but not thread-safe



Indexing Methods

- Indexing Methods
  - Brute Force (Flat)
    - Encode vectors of fixed sizes and compare to query vectors at search time
    - Supports encoding options but do not support much customization
  - Cell-probe Methods (IVF)
    - Feature space partitioned into n cells and vectors assigned a cell using a quantizer
    - Search is performed using n probes or inverted lists
  - Quantization-based
    - Vector Quantization, Additive Quantization, etc.
    - Product quantization
      - Separates the feature space into subspaces and queries these separately
        - Compression and GPU friendly
      - Accepts the same parameters as Flat or IVF depending on usage





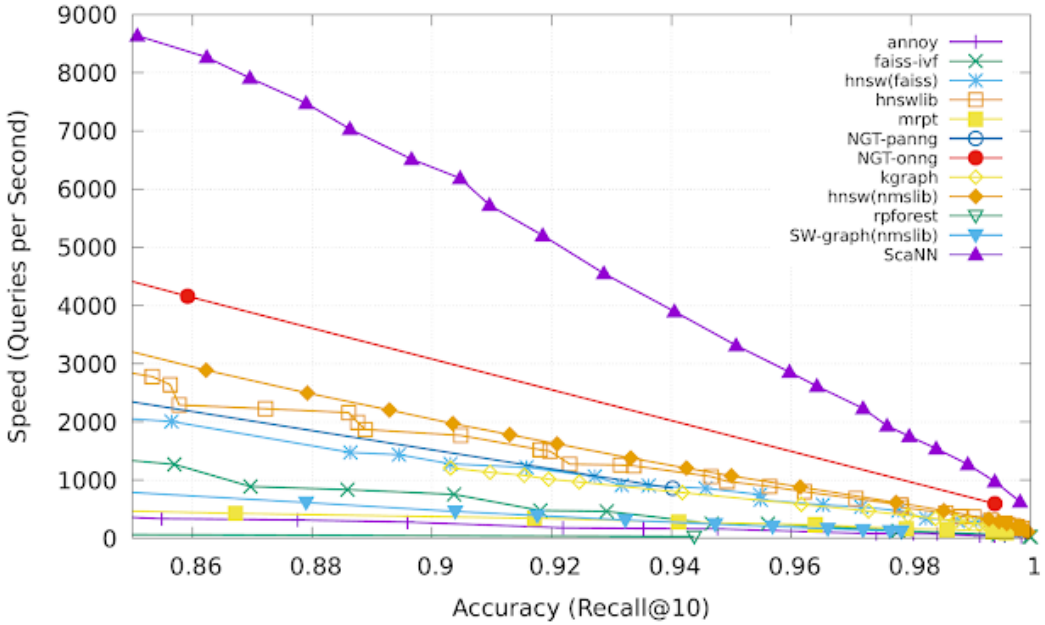
Additional Features

- Additional options
  - Pre-processors
    - PCA
      - Reduces dimensions of the dataset
      - Increases speed for Flat methods
    - OPQ
      - Applies rotation to the input vectors
      - Increases accuracy of PQ methods
  - Fast-scan
    - Improves speed for PQ and IVF

ScaNN

What is it?

- ScaNN is a vector similarity search tool, and is on the outside very similar to FAISS
- It first generates an index under a variety of customizable parameters, which can then be queried using several different techniques



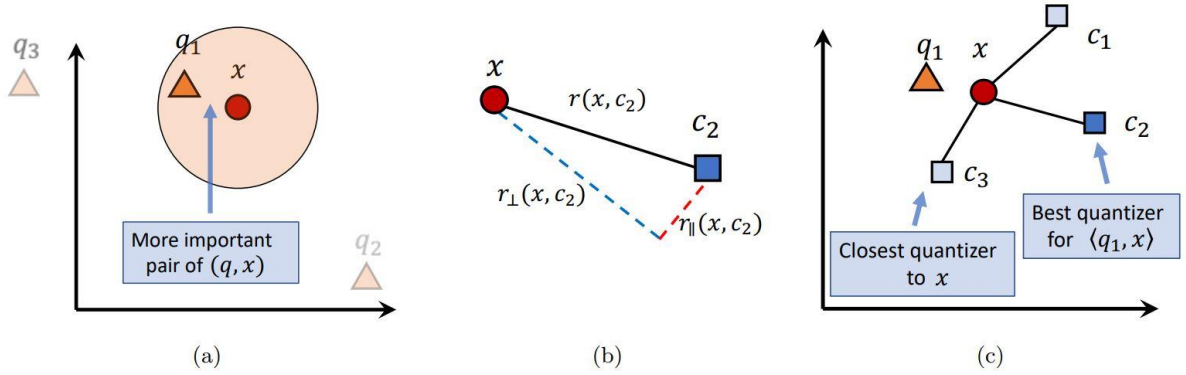
Why use ScaNN?

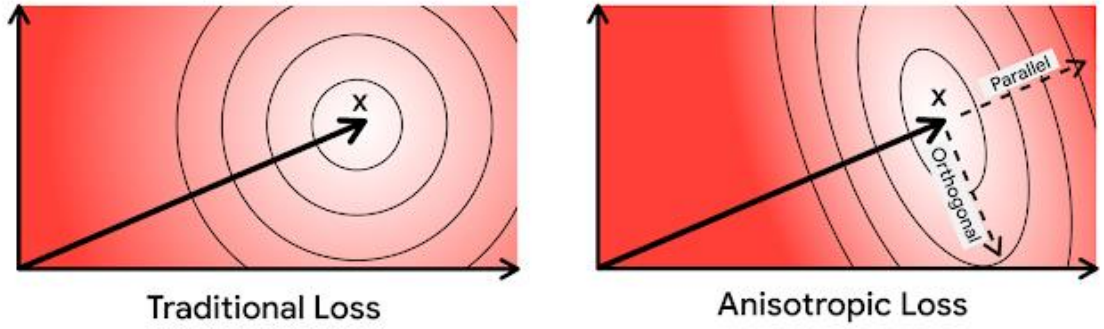
- ScaNN utilizes a novel loss function combined with several speed improvements to advance on other approaches, particularly in the field of queries/second.
- While ScaNN has fewer features than FAISS, this simplicity allows it to be quickly and easily implemented

Vector Quantization

ScaNN employs two primary novelities to improve its quantization method:

1. Score aware loss
    - Not all pairs of vectors are equally important, so we can weigh the quantization based on the inner product between the query and the datapoints
2. Anisotropic loss
    - Penalizing quantization error that is parallel to the original vector is penalized more heavily
    - This trades increased error of lower inner products for better accuracy of high inner products





Speed Gains

Partitioning

- Vector quantization based-trees reduce the number of scores that have to be calculated
- While this step can increase the indexing time, if implemented properly it can dramatically reduce the number of computations

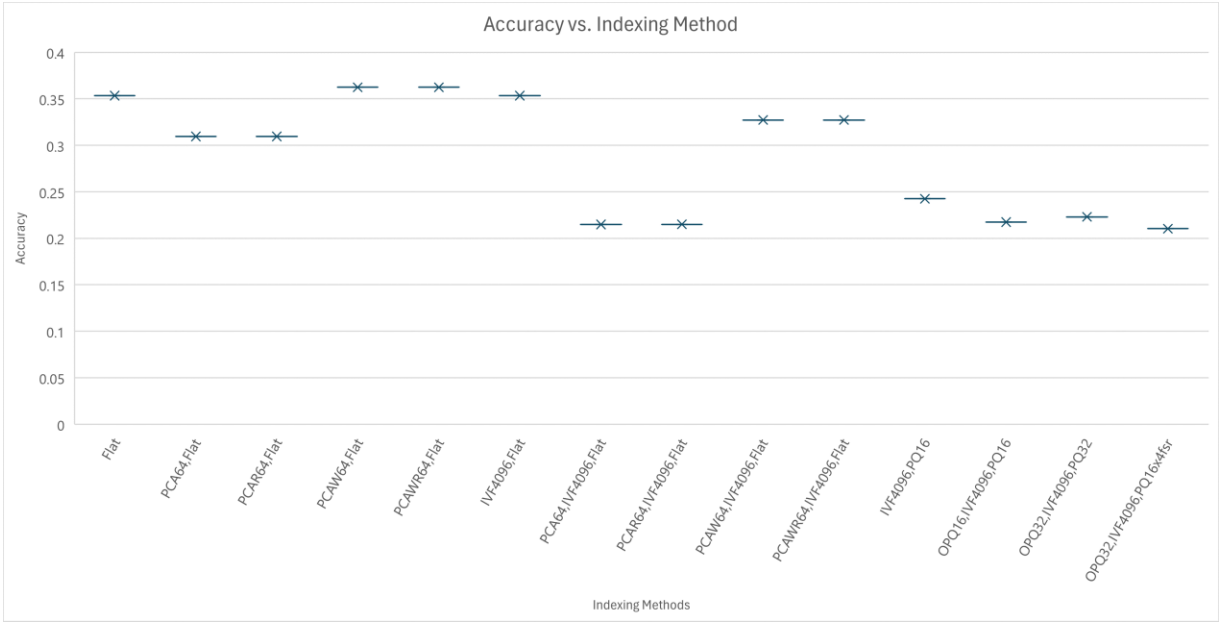
- Single Instruction Multiple Data (SIMD) in-register lookup tables
  - By reducing the size of distance representations, ScaNN can run SIMD registers
  - Register access is much faster than main memory, and so running on this increases speed significantly

FAISS

- Parameter Tuning
  - Used index_factory() to change indexing methods
    - Provide a string to build the index automatically
    - e.g. "IVF4096, Flat" creates an IVF-Flat index with Flat quantizer and 4096 cells
    - Varied indexing method, pre-processor, quantizer, etc.
  - Used varying values for the number of probes for IVF and PQ methods
    - Main tradeoff variable for speed and accuracy during the search process according to FAISS documentation
    - Higher value theoretically means greater accuracy and lower speed

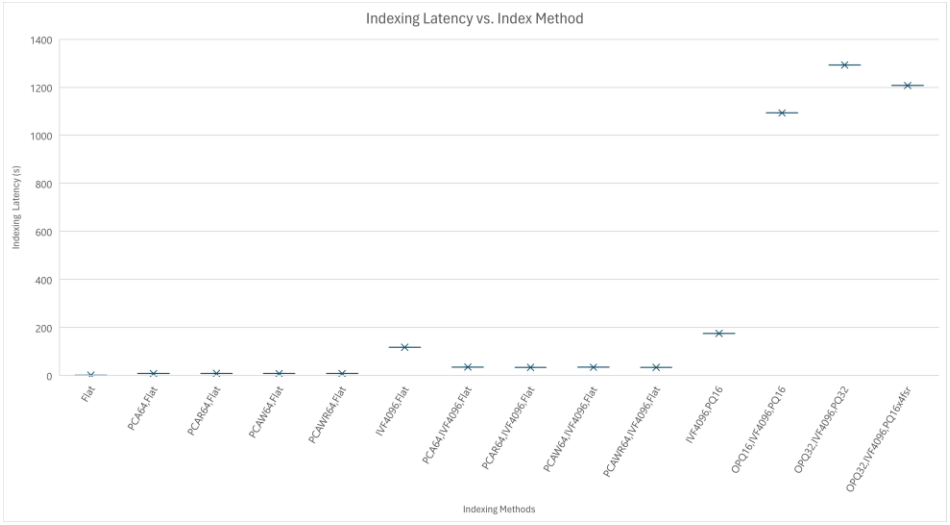
FAISS Accuracy vs. Indexing Method

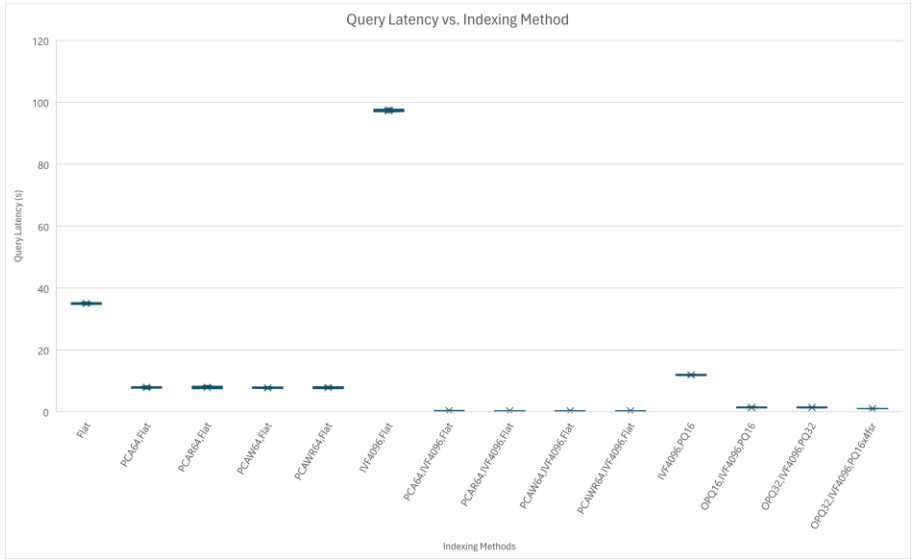
- Accuracy of the Flat indexing methods appears much greater
- PQ methods generally not very accurate due to compression
- Flat with a PCA pre-processor appears to be the most accurate



FAISS Timing vs. Indexing Method

- # of probes varied for search, with higher values resulting in higher time value
- Flat indexes generally faster compared to PQ
- Flat indexes spend more time in searching compared to PQ

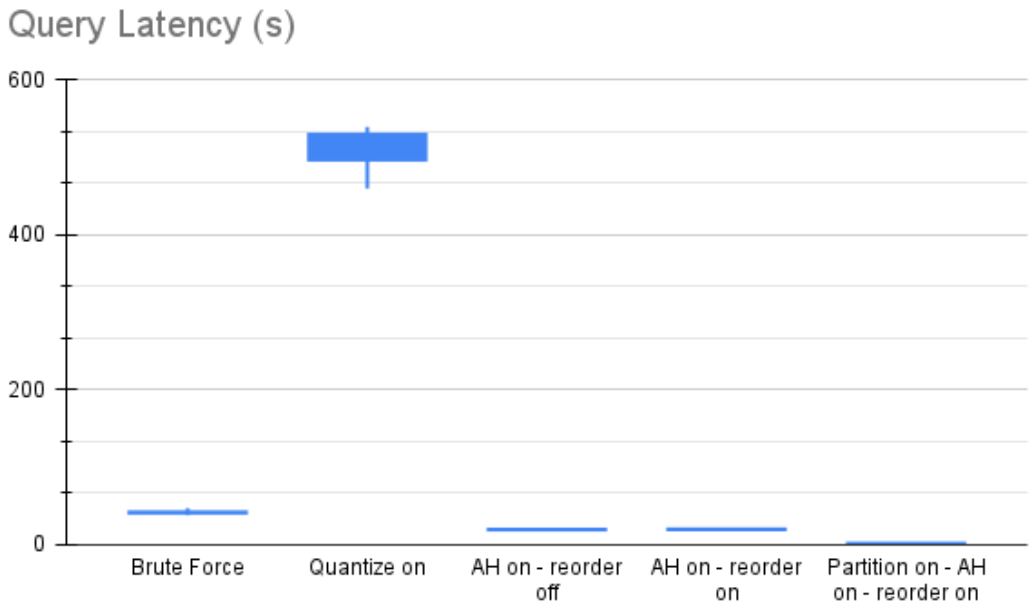


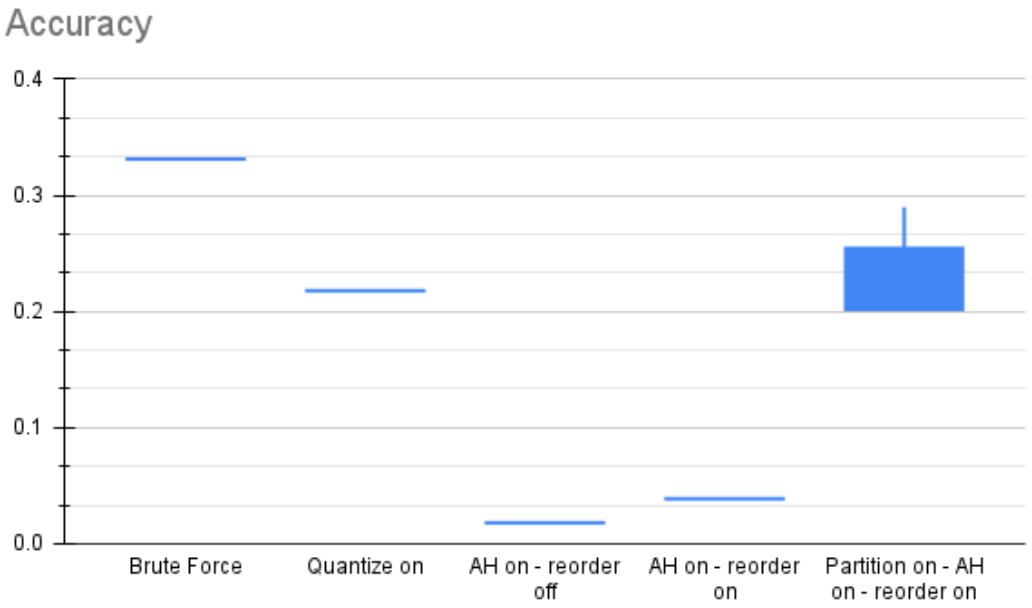


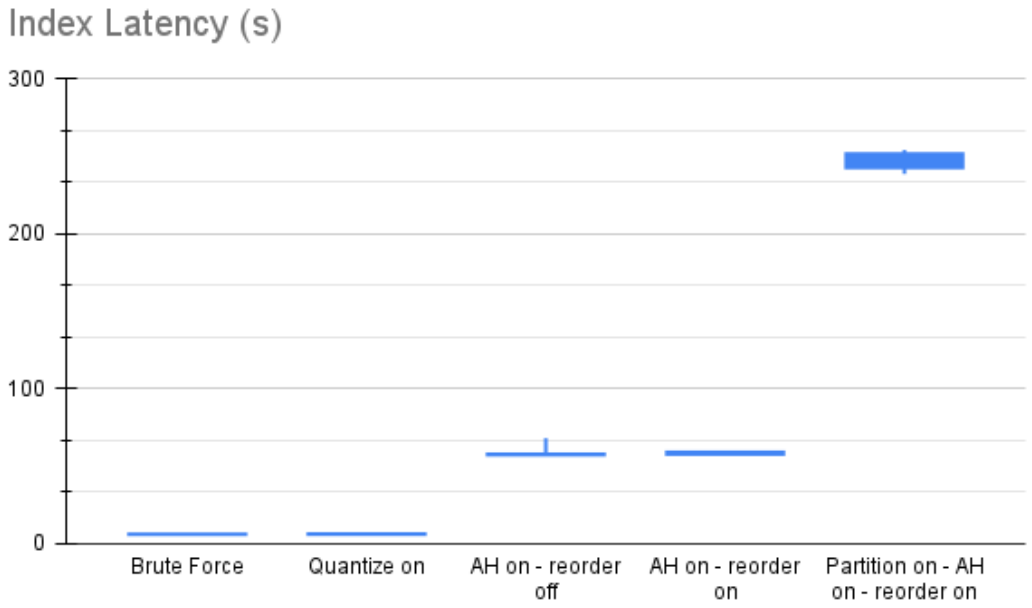
ScaNN

- While ScaNN does not have a equivalent method to FAISS index_factory(), it has far fewer algorithms that can be implemented and tuned.
- Indexing
  - Partitioning (optional) - while training the index, it is possible to build a tree to reduce the number of comparisons in the querying phase. This results in a tradeoff: building a tree increases training time and reduces accuracy, but can significantly speed up query time.
- Querying
  - Scoring – This can be performed through brute force or asymmetric hashing (AH).
  - Rescoring (optional) - This takes the best k' distances from the initial scored and recomputes them more accurately to better select the top k vectors
- Google recommends the following parameters:
  - Small dataset: brute force
  - <100k points: AH -> rescore
  - >100k points: Partition -> AH -> rescore

ScaNN: Indexing and scoring methods compared

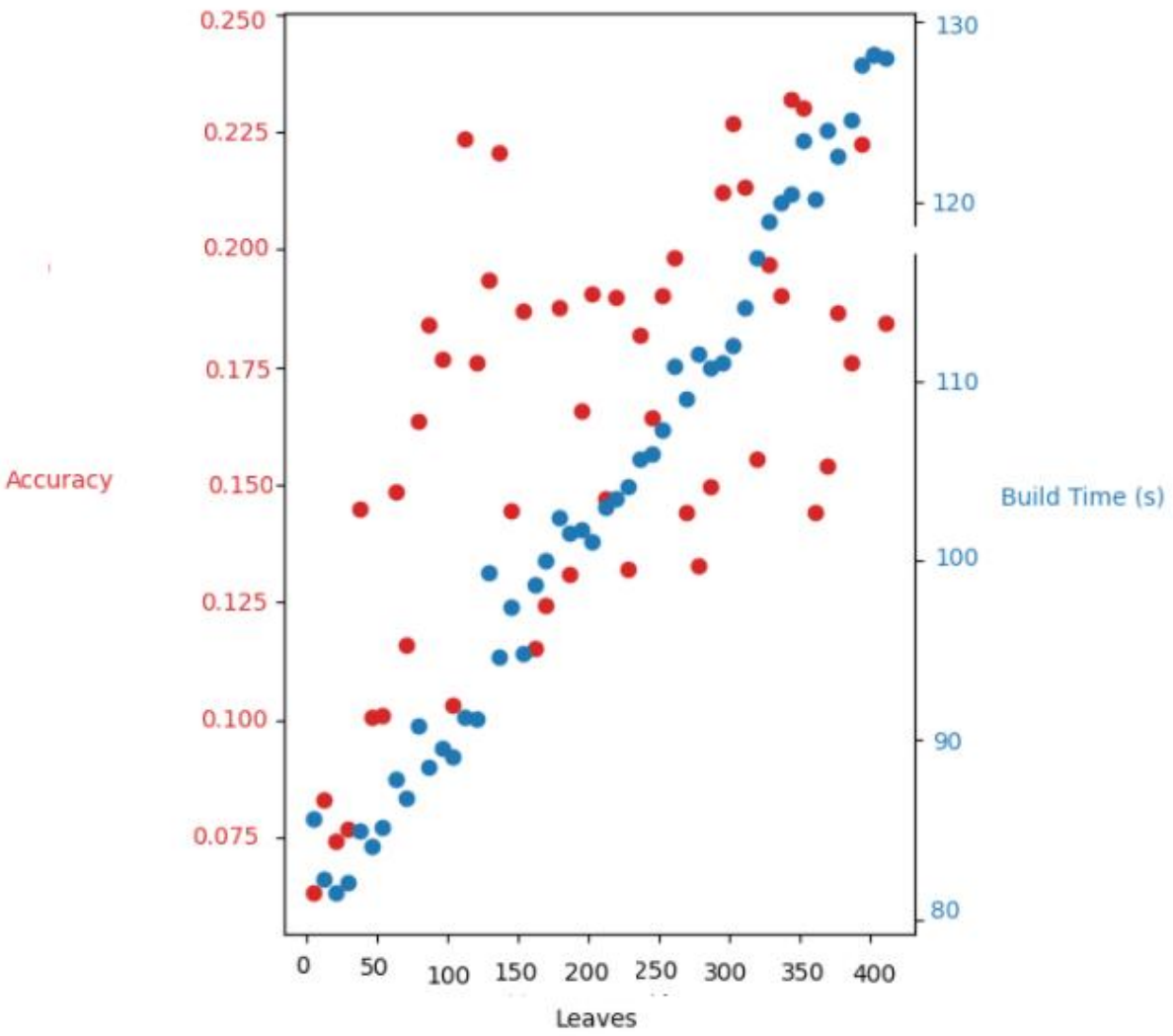




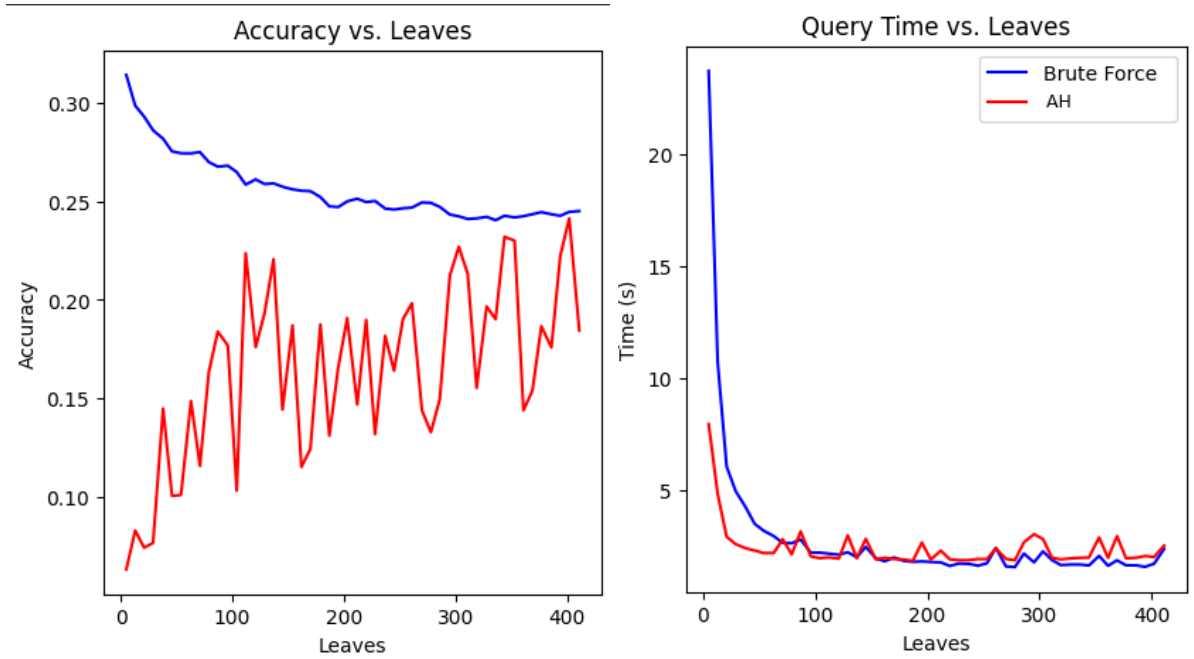


ScaNN Accuracy vs. Build Time

- Leaves represent the number of patitions to find the best matches
- The larger the number of leaves, the more accurate the results
- Run-time is impacted by increasing leaves

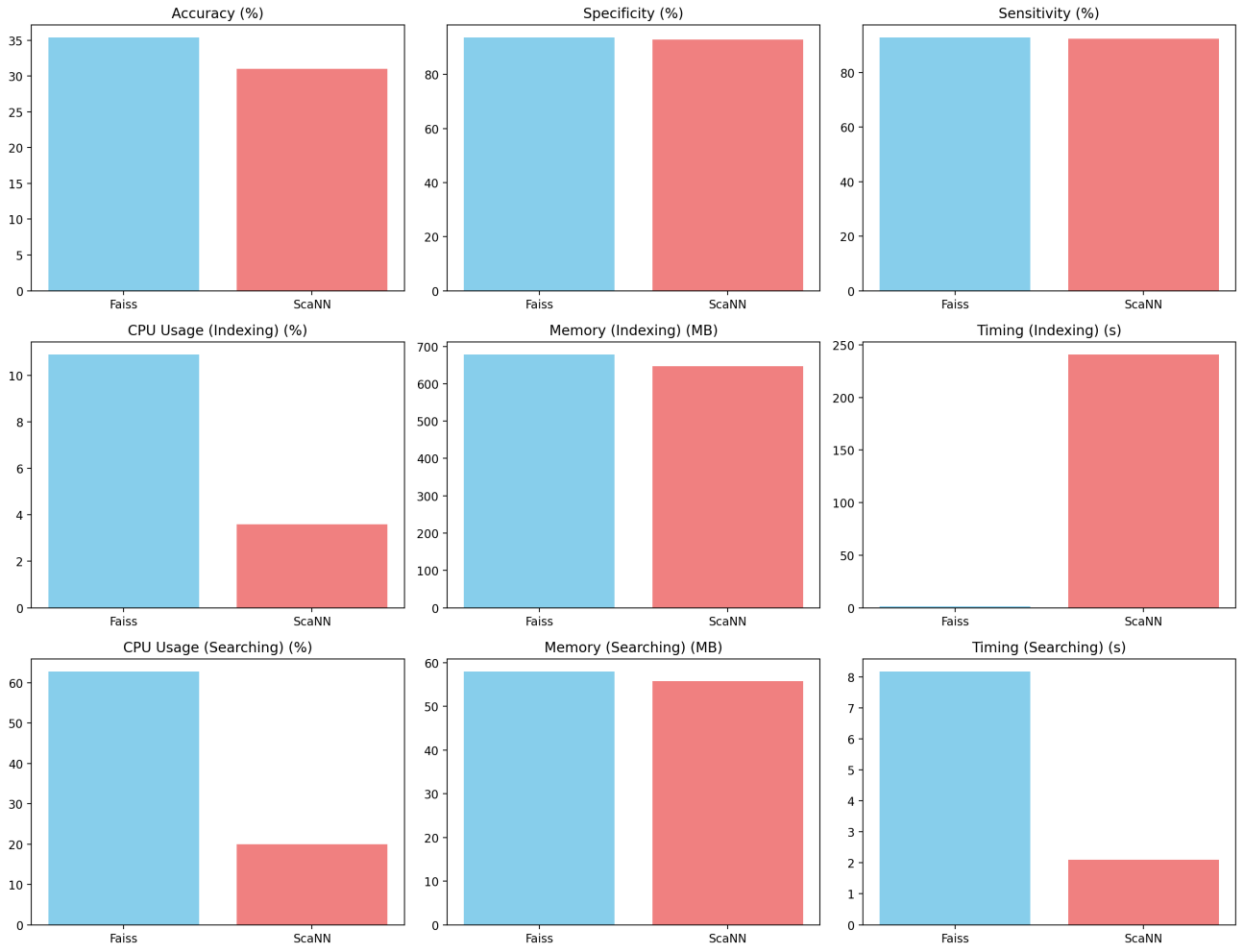


ScaNN Brute Force vs. Asymmetric Hashing



- Brute Force is a naiive, computationally intensive method of finding the best candidate
  - Computer distance between query vector and every trained vector
- Asymmetric Hashing utilizes binning so every distance doesn't have to be computed
  - AH allows for distances to be only computed between bin label and inside of select few

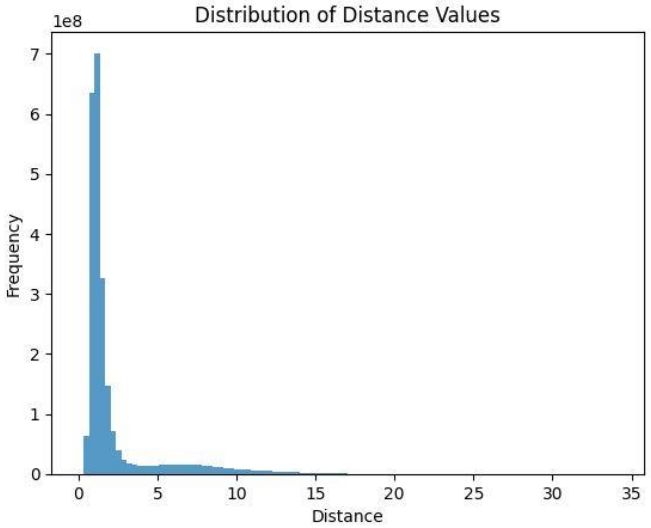
Performance Metrics Comparison: FAISS & SCANN



Novelty Detection

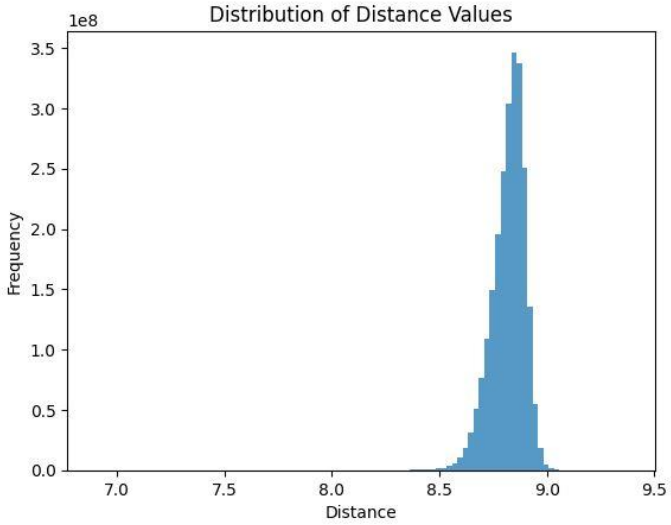
- Exploring Distinction between the distribution of the distances of Faiss & ScaNN embeddings
- High-dimensional distance matrix structure with a shape of (14256, 165615)
- Aim to identify organisms at a new taxonomic level that exhibit unique genetic signatures distinct from those in known phyla to the training set

Distribution of Distances



Faiss

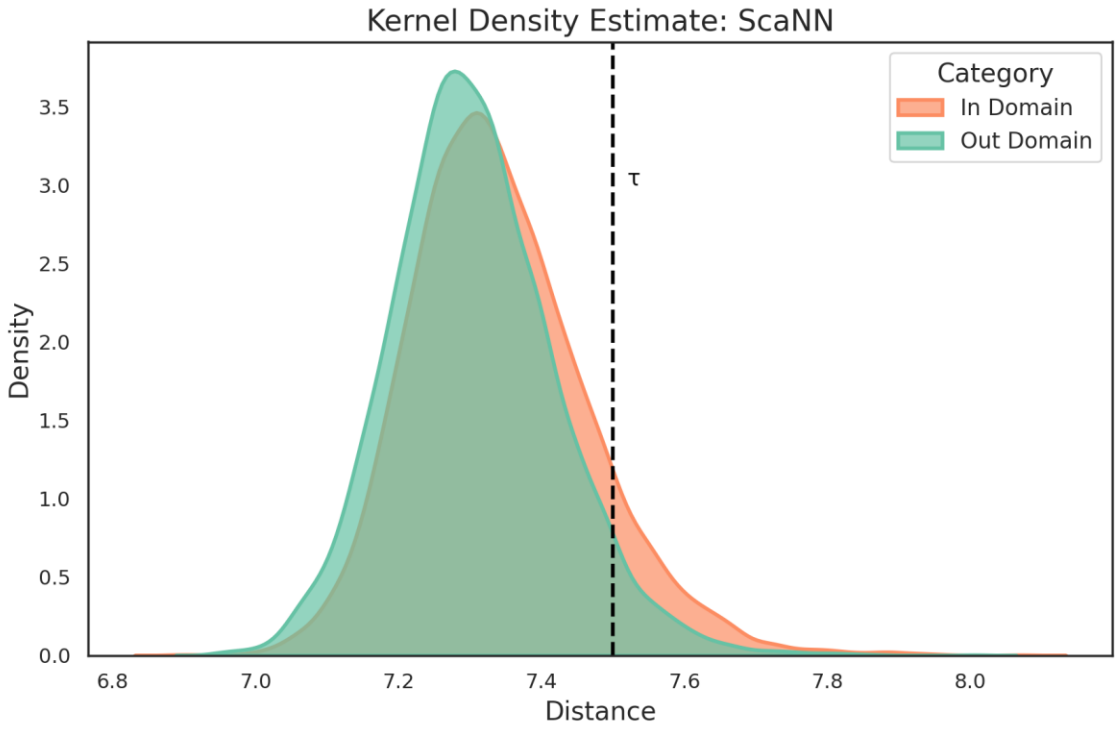
Min distance: 0.0
Max distance: 34.0929
Mean distance: 2.196
Median distance: 1.231
Standard deviation: 2.593

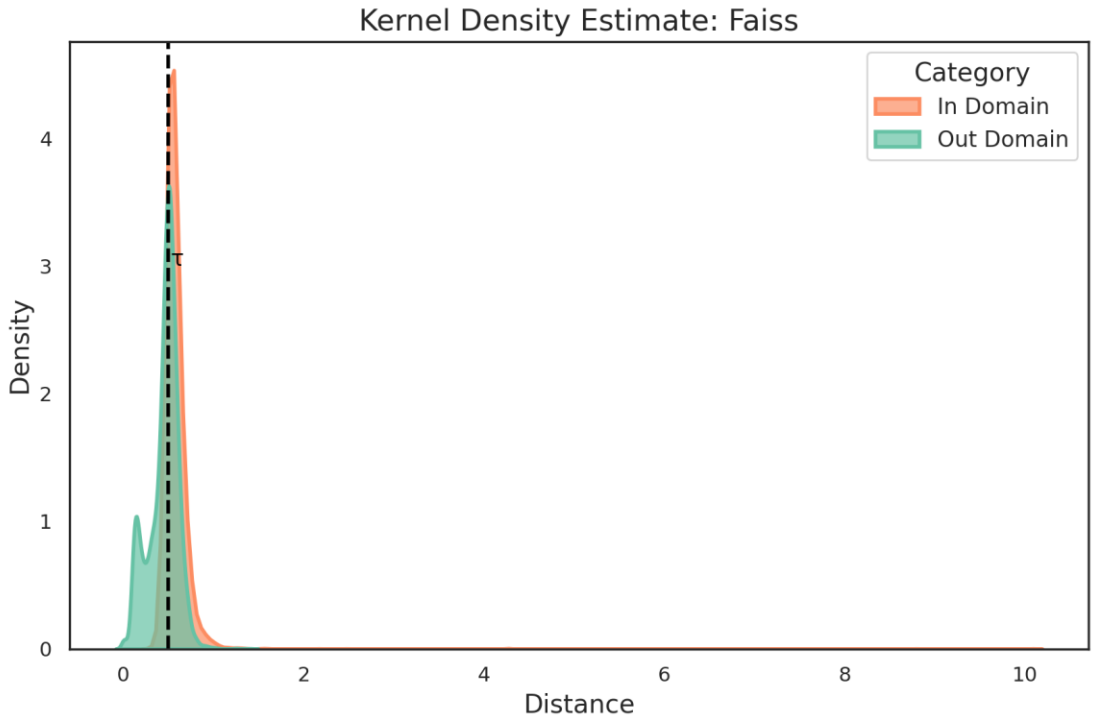


ScaNN

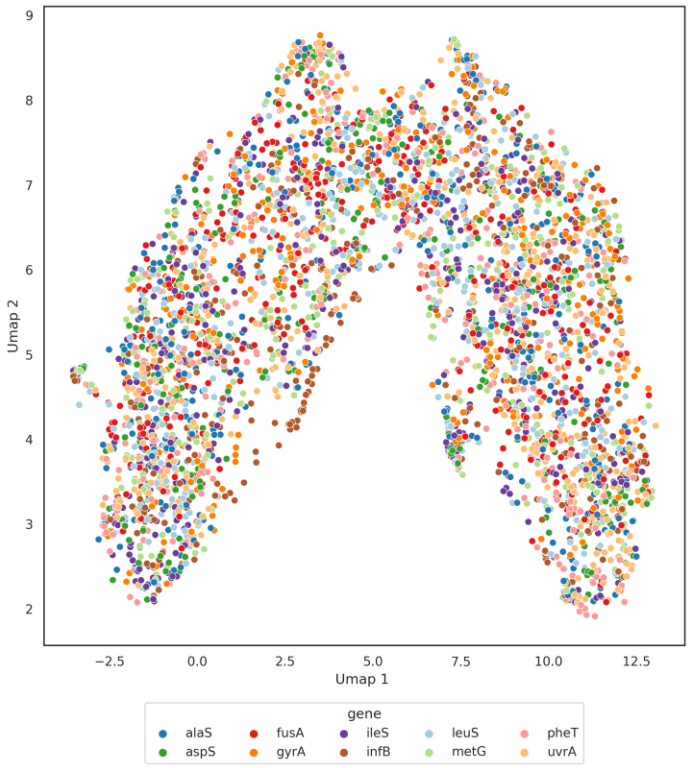
Min distance: 6.8944
Max distance: 9.380
Mean distance: 8.820
Median distance: 8.832
Standard deviation: 0.0763

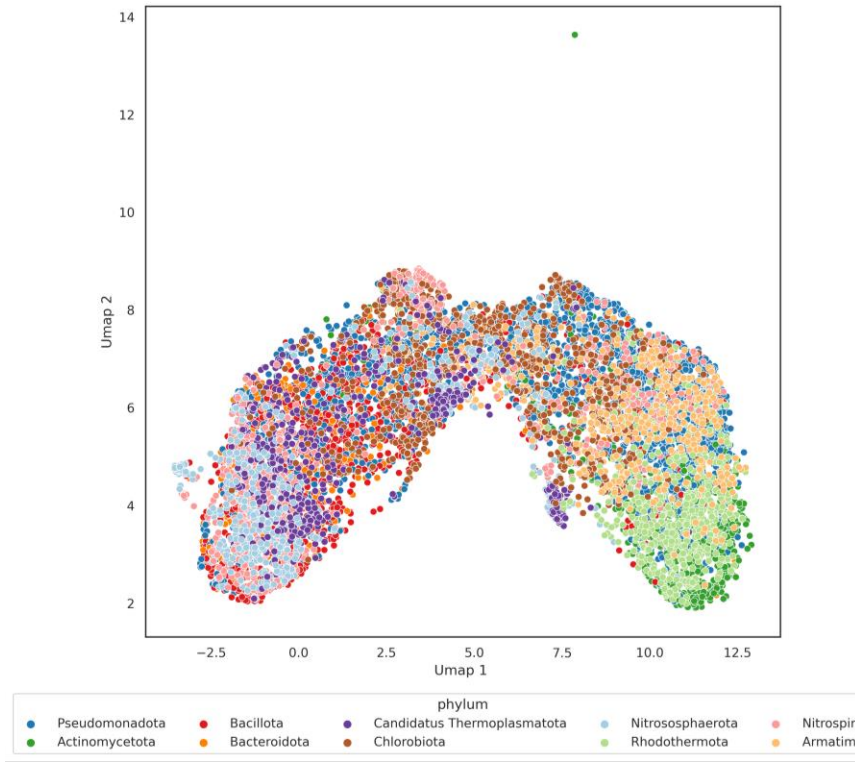
KDE of In Domain and Out Domain sets

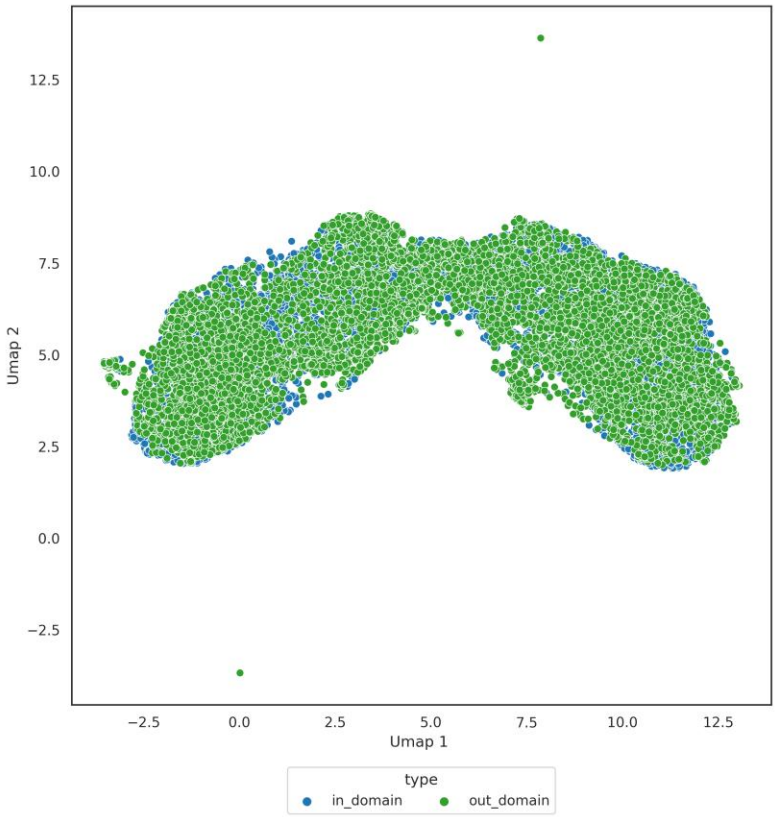




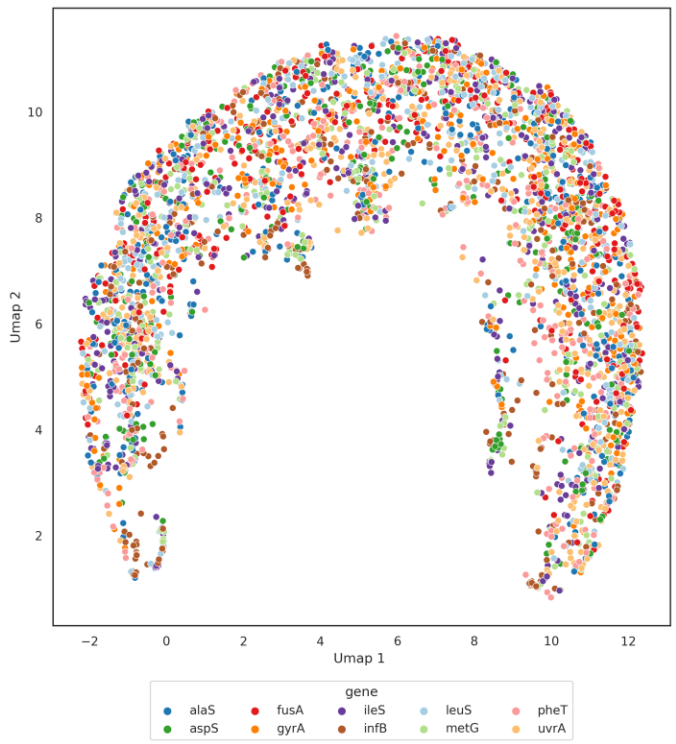
UMAP Projection - ScaNN

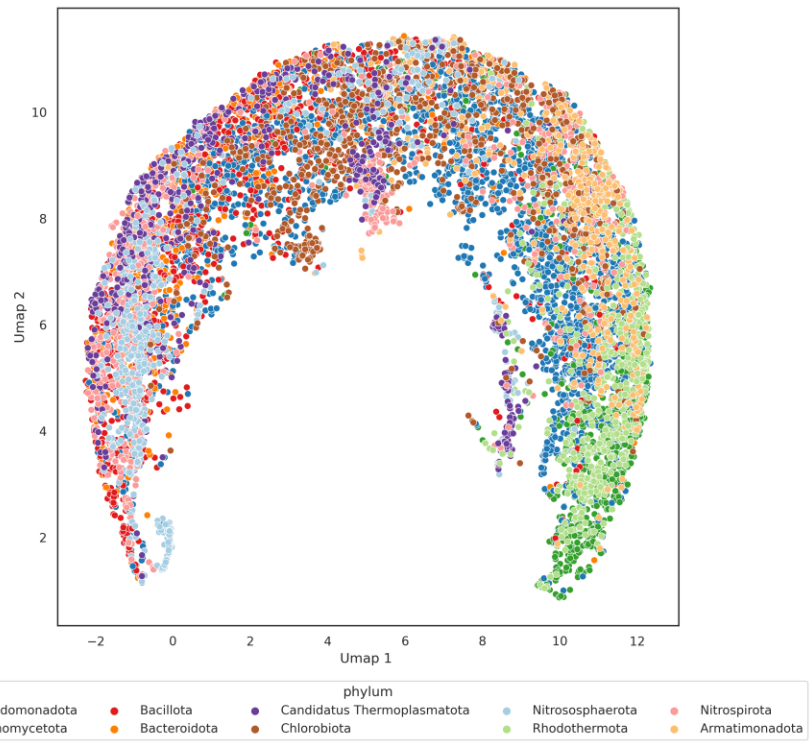


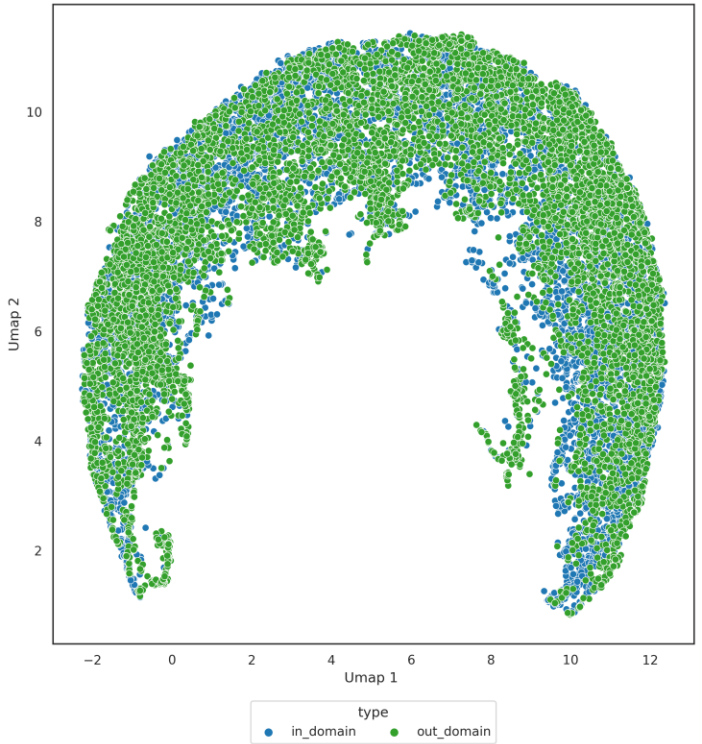




UMAP Projection -FAISS



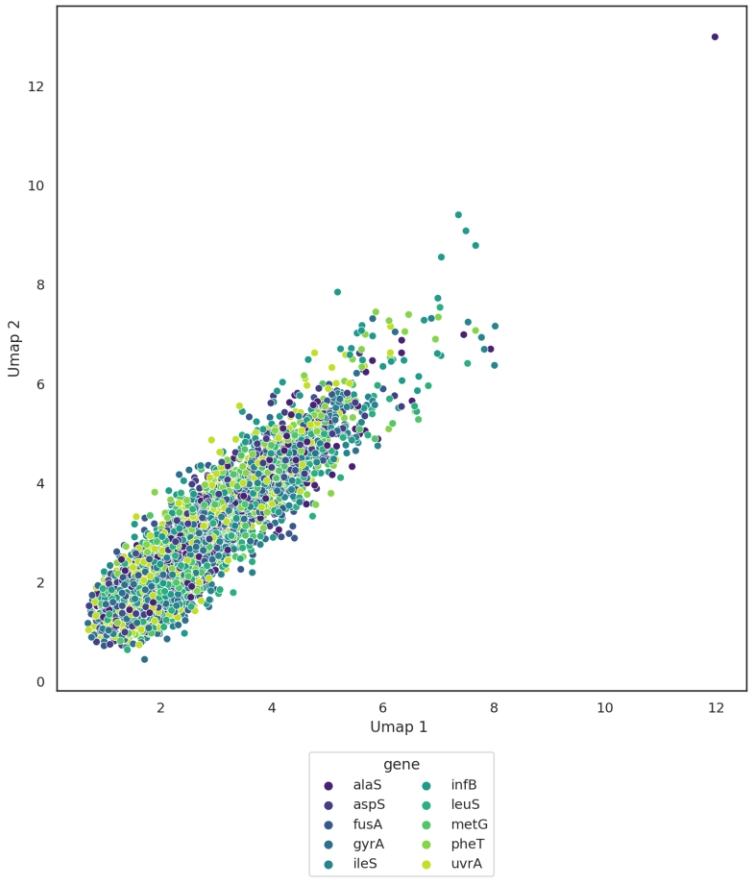


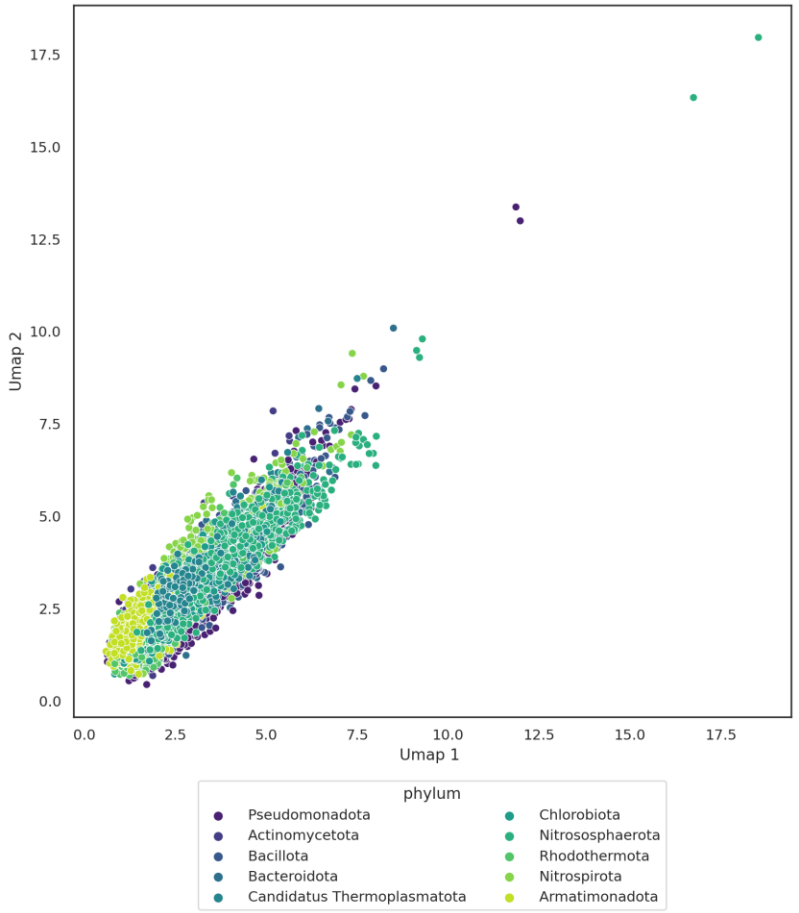


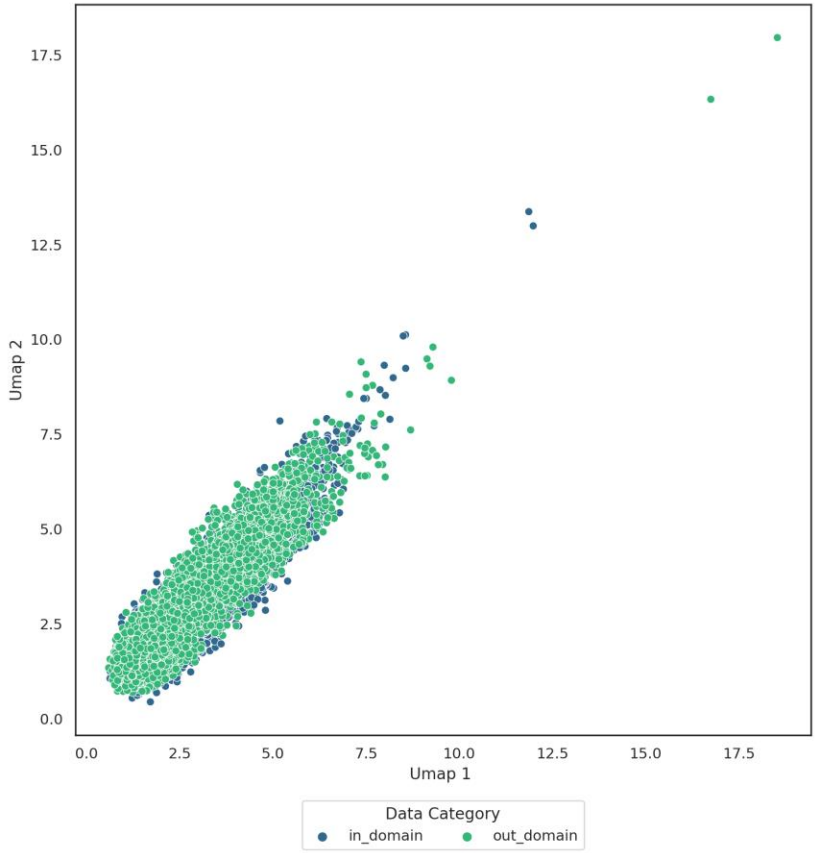
References

- Guo, R., P. Sun, E. Lindgren, Q. Geng, D. Simcha, F. Chern, and S. Kumar (2020). Accelerat-ing large-scale inference with anisotropic vector quantization. In International Conference on Machine Learning, pp. 3887–3896. PMLR.
- Johnson, J., M. Douze, and H. Jégou (2019). Billion-scale similarity search with gpus. IEEE Transactions on Big Data 7 (3), 535–547.
- Jégou, H., M. Douze, and C. Schmid (2011). Product quantization for nearest neighbor search. IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (1), 117–128.
- Olson, D. R., D. Demekas, T. Colligan, and T. Wheeler (2024). Near: Neural embeddings for amino acid relationships. bioRxiv , 2024–01.
- Pearson, W. R. (2013). An Introduction to sequence similarity ("homology") searchings. Current Protocols in Bioinformatics.

UMAP Projection -FAISS







UMAP Projection -ScaNN