IVF^2 Index: Fusing Classic and Spatial Inverted Indices for Fast Filtered ANNS

Ben Landrum

ParlayANN

December 16, 2023

Ben Landrum

University of Maryland

Magdalen Dobson Manohar

Carnegie Mellon University

Mazin Karjikar

University of Maryland

Laxman Dhulipala

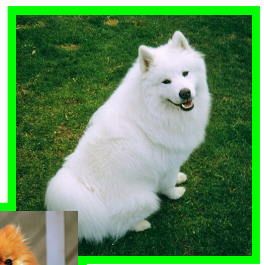
University of Maryland

Outline

Approximate Nearest Neighbors Search












Approximate Nearest Neighbors Search

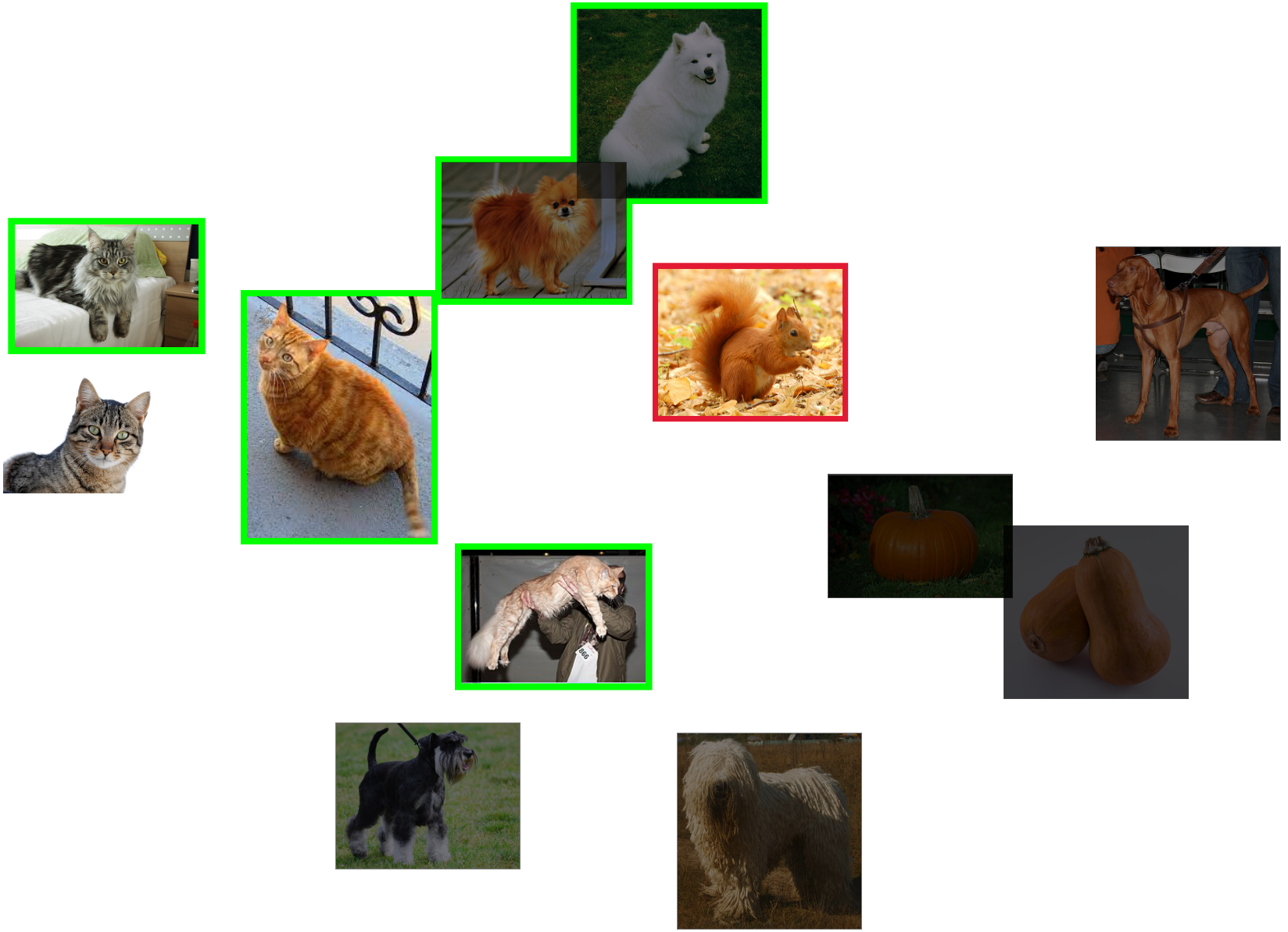
The Classical ANNS Problem

Given:

- a query q
- a set of points P
- some distance function d

find a set of k points $p_i \in P$ that minimizes $\sum d(q, p_i)$.

Filtered ANNS



Filtered ANNS

Filtered ANNS

Given:

- a query q
- a set of points P
- some distance function d
- n_f sets of points $\mathfrak{L} = \{\mathfrak{l}_1, \ldots \mathfrak{l}_{n_f}\}$ where $x \in \mathfrak{l}_i$ iff point x has label i
- a predicate $\mathfrak{F}$ which is a boolean combination of the elements of $\mathfrak{L}$

find a set of k points $p_i \in \mathfrak{F}$ that minimizes $\sum d(q, p_i)$.

Classic Inverted File Index

Documents

1: {. . . far better thing that I do . . . }
2: {Friends, Romans, countrymen . . . }
3: {. . . taste and decency . . . }
4: {. . . Fair is foul, and foul is fair . . . }
5: {. . . It is a truth universally . . . }
6: {. . . and throw the peel away . . . }

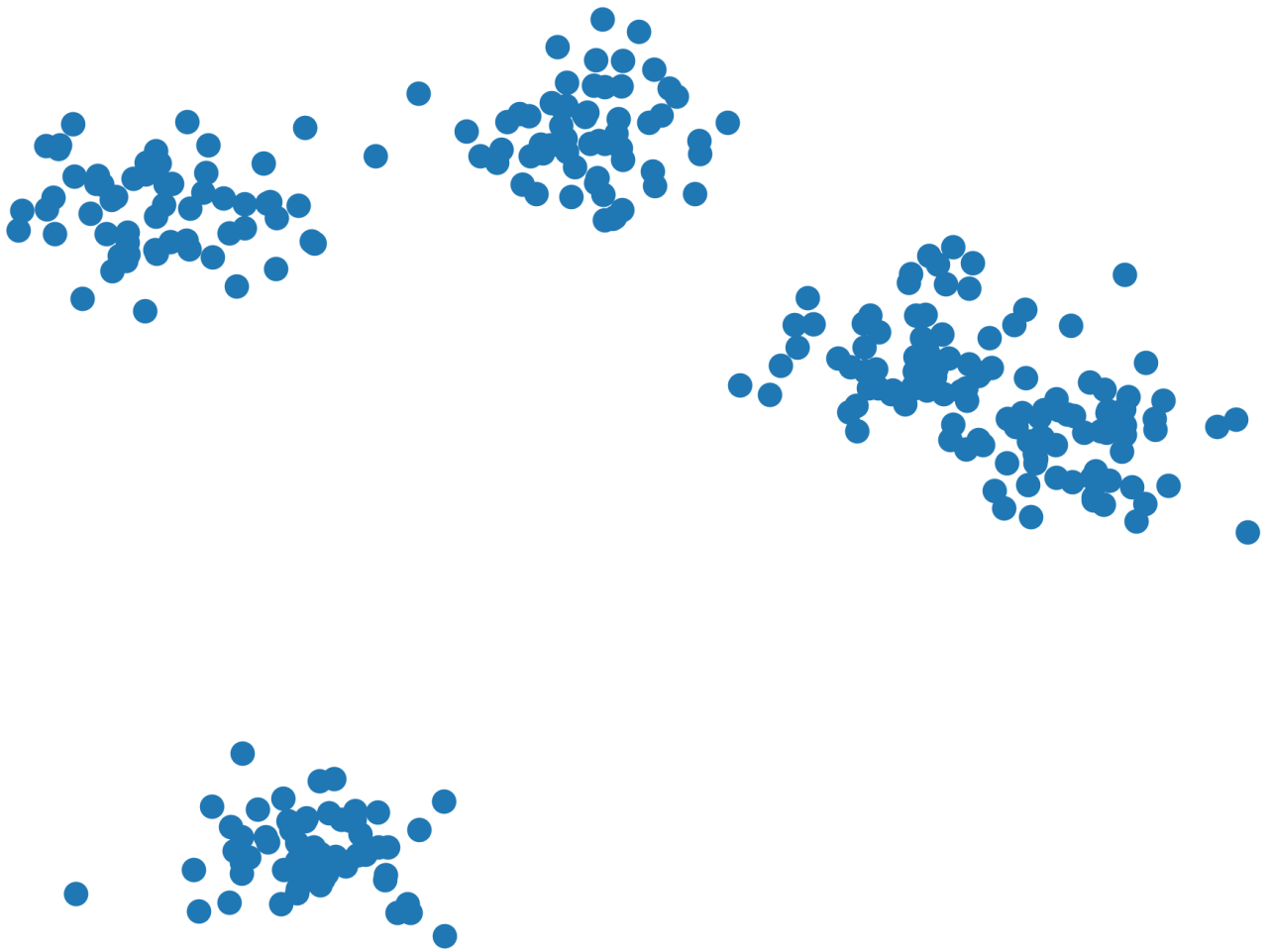
⋮

Inverted File Index

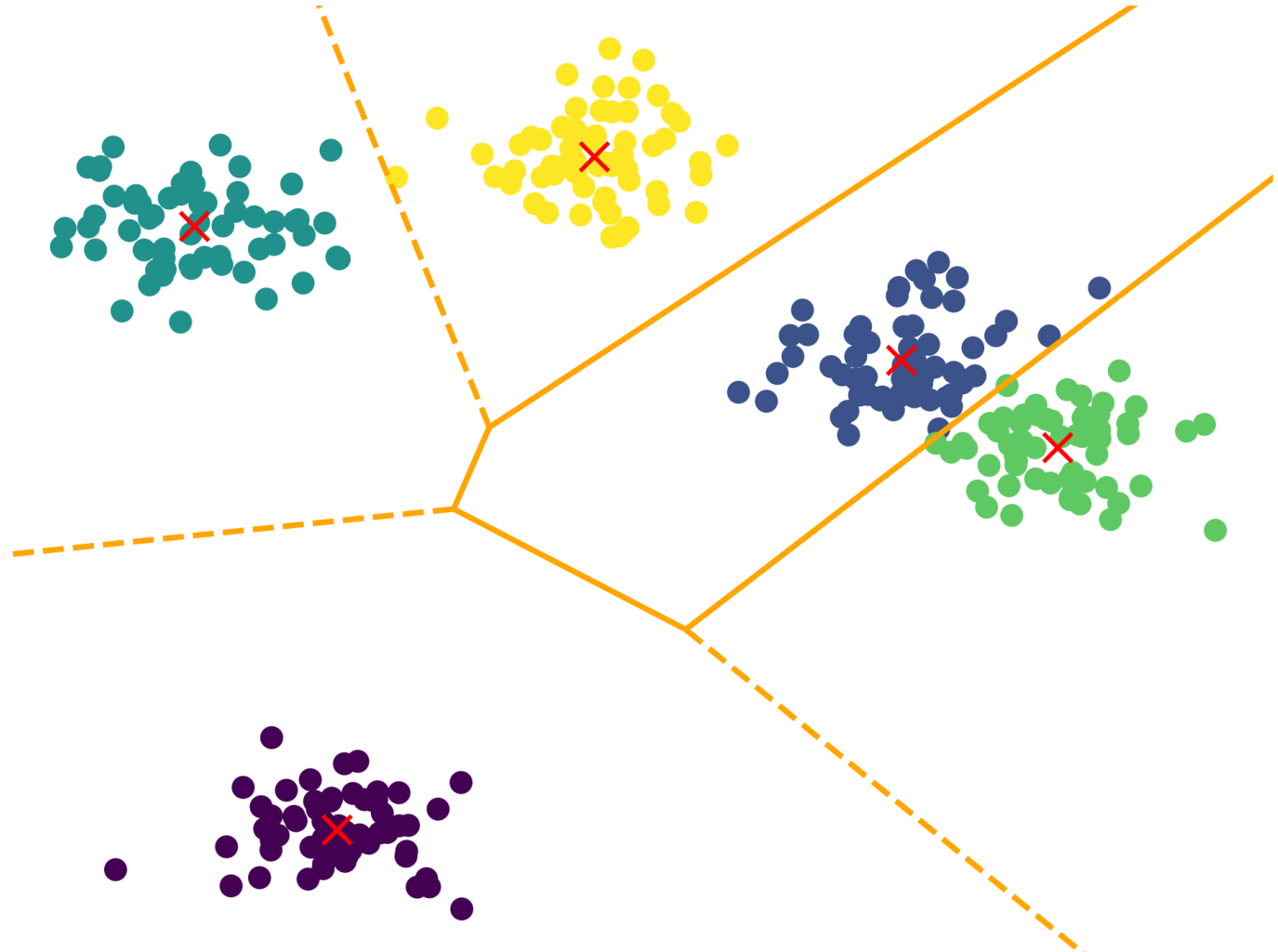
a: {5, . . . }
and: {3, 4, 6, . . . }
away: {6, . . . }
better: {1, . . . }
countrymen: {2, . . . }
decency: {3, . . . }

⋮

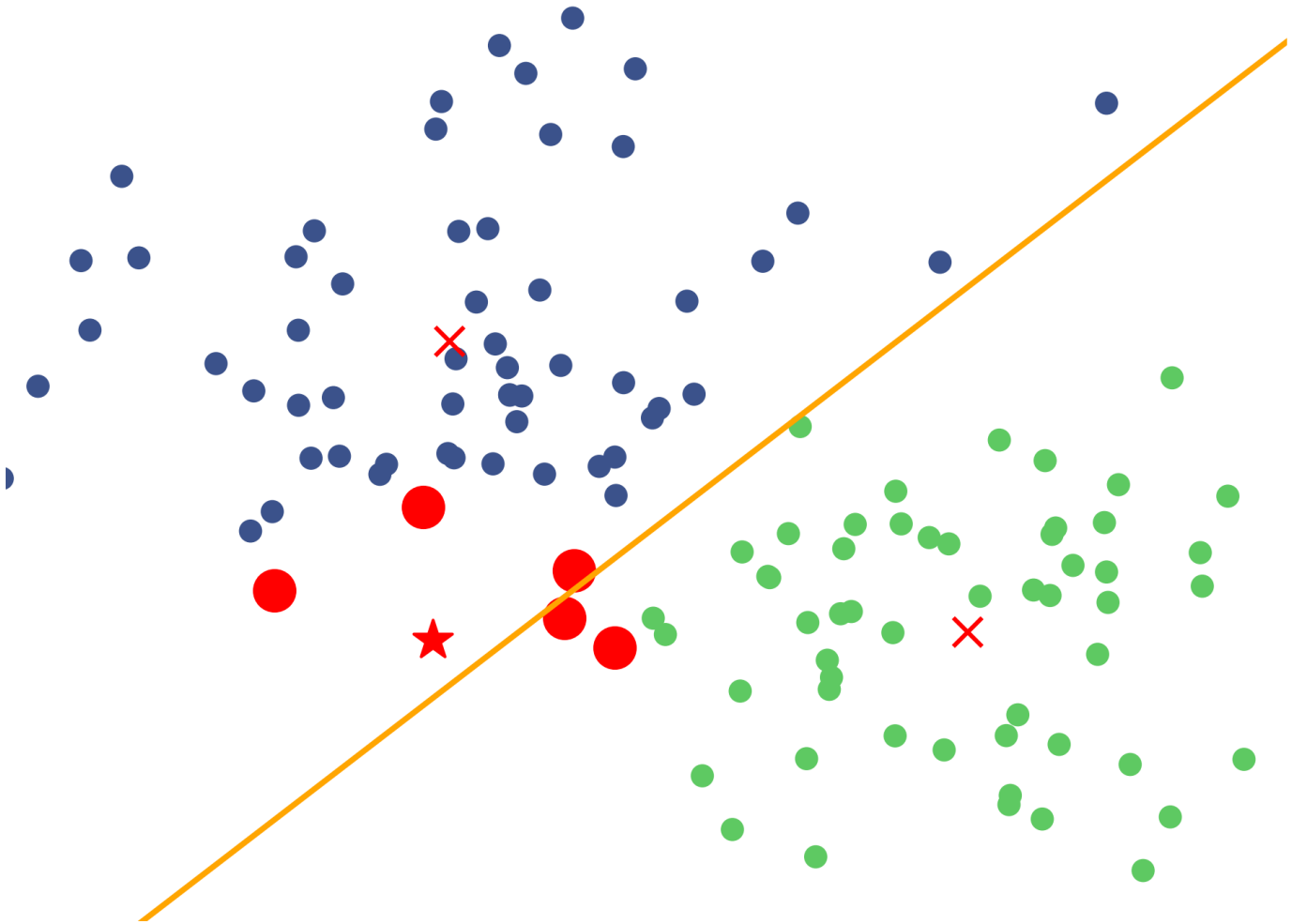
IVF Index



IVF Index



IVF Index



Ethos of the IVF^2 Index

Many existing indices (e.g. IVF, LSH, kd/ball-tree, Annoy, etc.) operate by partitioning the vector space.

The labels represent a useful partition

IVF^2 Overview

We build a classic inverted file index over the labels, indexing the vectors associated with each label independently.

Large labels

For labels with many vectors, we build:

- An IVF index
- A (relatively) lightweight Vamana search graph
- A bitvector of length n allowing fast lookup of whether a given vector is associated with the label

Small labels

For labels with few vectors, we just store the indices of the vectors associated with the label.

Single-Filter Query

We are given a query vector q and a single label ℓ.

If ℓ is a large label

We use the very fast Vamana search graph to find the k nearest neighbors of q among the vectors associated with ℓ in a classical k-NN query.

If ℓ is a small label

We exhaustively check the vectors associated with ℓ.

'AND' Query Approach

We are given a query vector q and two labels $\mathfrak{l}_a$ and $\mathfrak{l}_b$, where $\mathfrak{l}_a$ has fewer points associated with it than $\mathfrak{l}_b$.
We want to restrict our search to as few candidates as possible before doing expensive distance computations.
There are two natural ways to do this:

- Filter $\mathfrak{l}_a$'s vectors by membership in $\mathfrak{l}_b$
- Get many likely candidates from each label, and then join the two sets

Filtering by Membership

If $\mathfrak{l}_a$ is especially small and $\mathfrak{l}_b$ has a filter, we can filter $\mathfrak{l}_a$'s vectors by membership in $\mathfrak{l}_b$.

Advantages

- Each item is a single read from memory
- The results are exact

Joining Two Sets

For each of the two labels, we want to fetch a large set of possible candidates, and then take the intersection of their respective candidate sets.

Large Labels

- Compare q to the representative points of the IVF index
- Collect up to some predetermined number of points from the nearest partitions into a sorted list

Small Labels

Take the existing sorted array of points associated with the labels

A Note on Cache Optimization

If you can order a batch of queries effectively, you can keep relevant parts of the index in cache between queries.

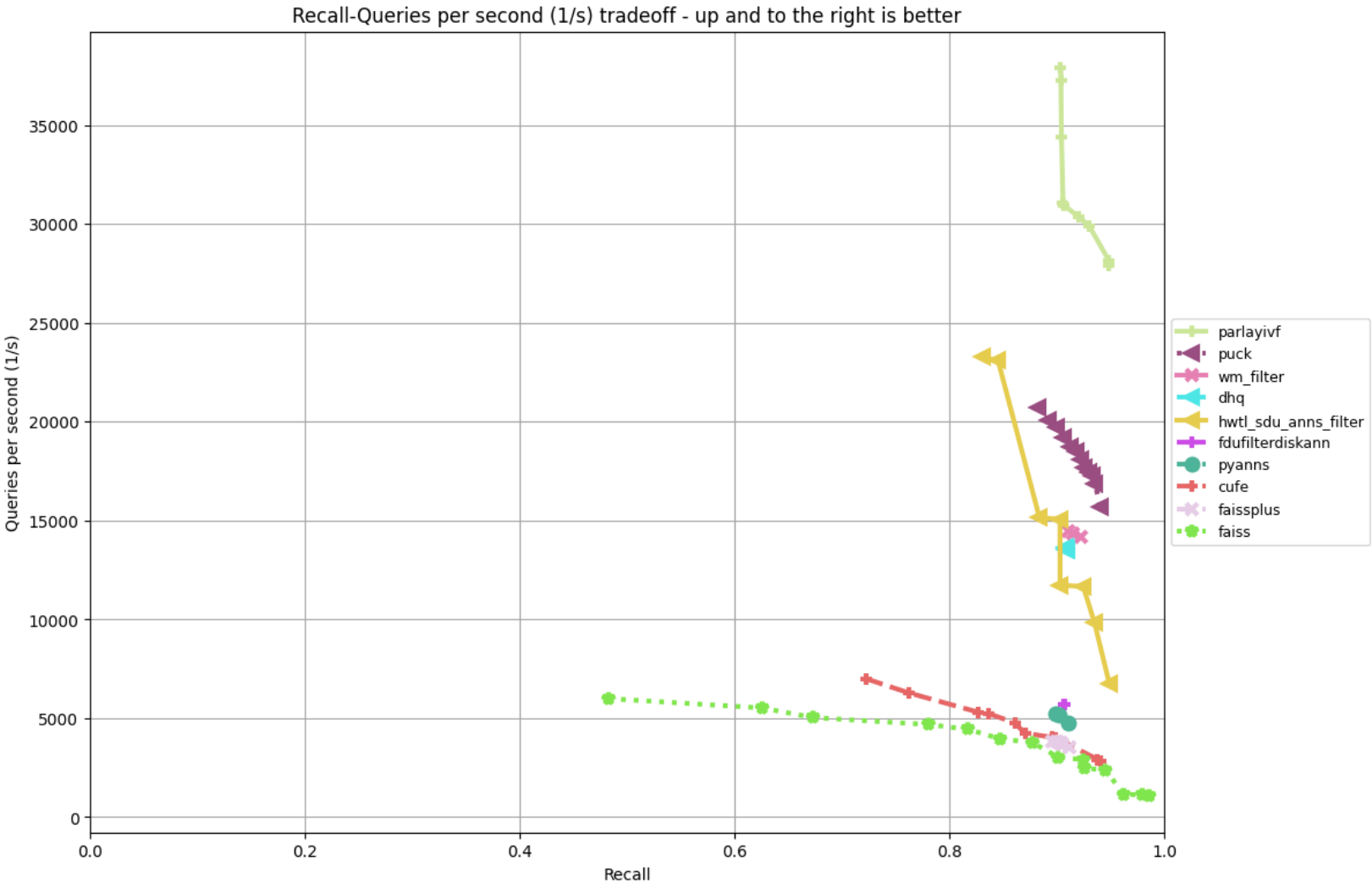
This is difficult in classical ANNS.

Our approach makes this easy, and we see a speedup of $\sim 30\%$ from a principled sort on filters.

BigANN Filter Track

- 10 million vectors
- 200,386 labels
- 100,000 queries

BigANN Filter Track



ParlayANN

- Highly optimized parallel implementations of ANN algorithms
- Built on parlaylib, a framework for fast and easy shared-memory parallelism

Check us out at https://github.com/cmuparlay/ParlayANN