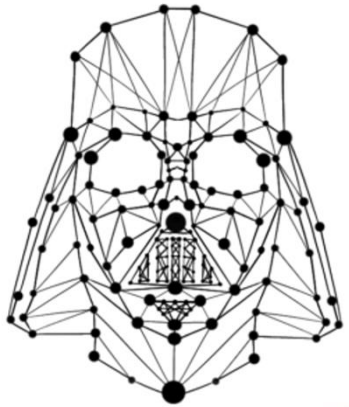


DARTH: Declarative Recall through Early Termination for Approximate Nearest Neighbor Search

Manos Chatzakis
Université Paris Cité, France
manos.chatzaki@gmail.com

Yannis Papakonstantinou
Google, USA

Themis Palpanas
Université Paris Cité, France

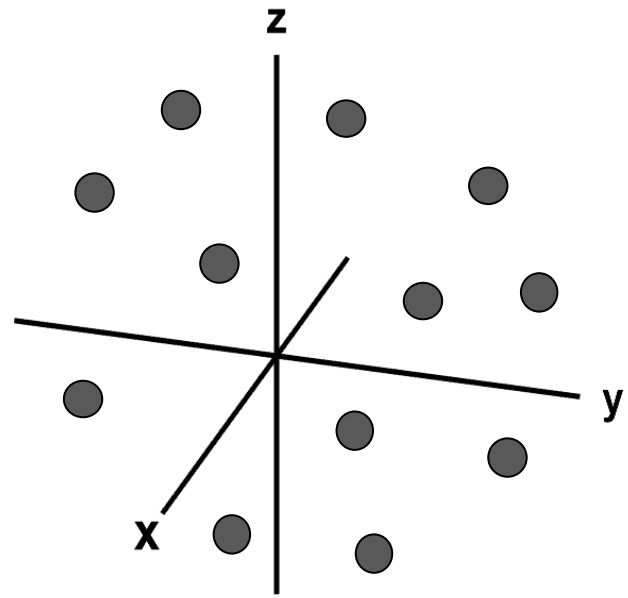
LIPADE
Laboratoire d'Informatique PAris DEscartes

Université Paris Cité

Google

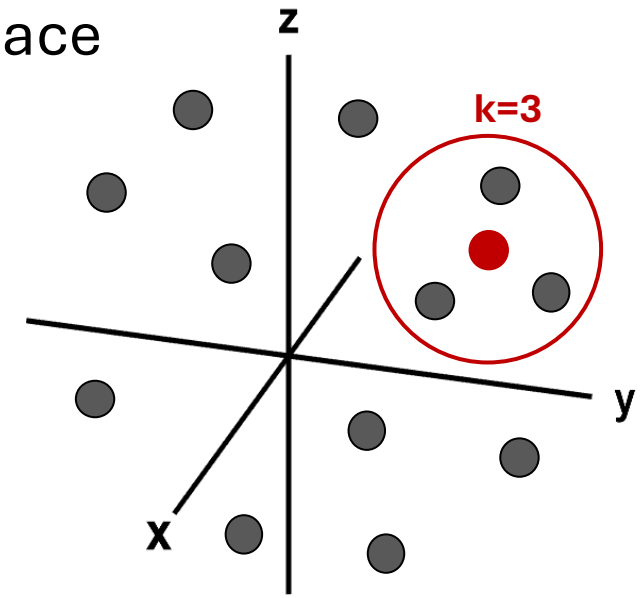
Nearest Neighbor Search

- Nearest Neighbor vector Search



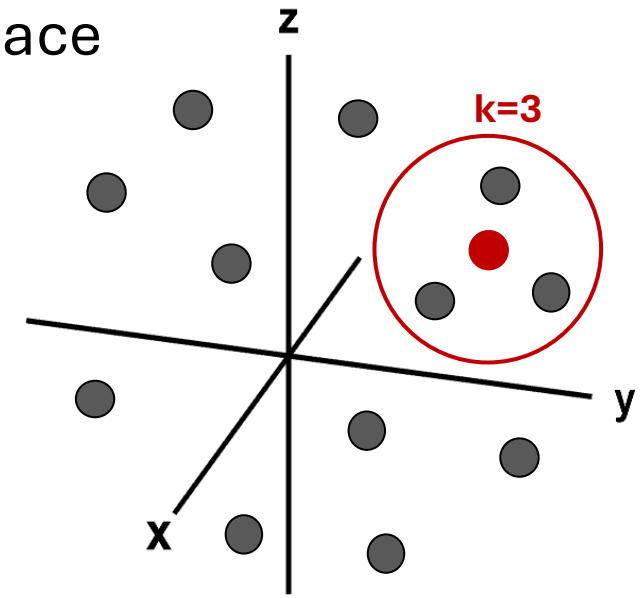
Nearest Neighbor Search

- Nearest Neighbor vector Search
  - Find top-k nearest vectors of a **query** in a high-dimensional space



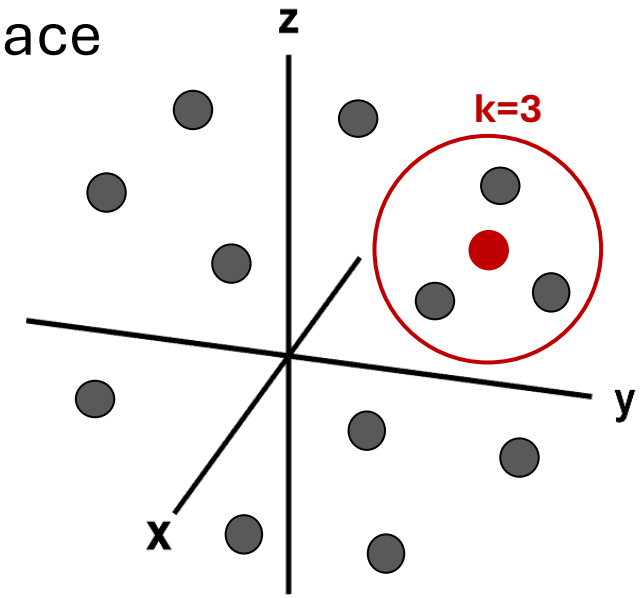
Nearest Neighbor Search

- Nearest Neighbor vector Search
  - Find top-k nearest vectors of a **query** in a high-dimensional space
  - **Approximate NNS (ANNS):** Not exact but much faster!



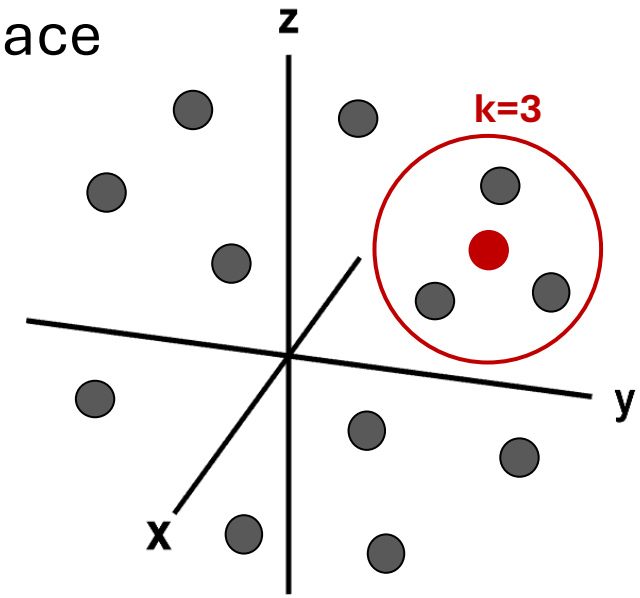
Nearest Neighbor Search

- Nearest Neighbor vector Search
  - Find top-k nearest vectors of a **query** in a high-dimensional space
  - **Approximate NNS (ANNS):** Not exact but much faster!
- Index the vector collection and then search
  - **ANNS(q, k, params)** → algorithm-dependent params



Nearest Neighbor Search

- Nearest Neighbor vector Search
  - Find top-k nearest vectors of a **query** in a high-dimensional space
  - **Approximate NNS (ANNS):** Not exact but much faster!
- Index the vector collection and then search
  - **ANNS(q, k, params)** → algorithm-dependent params
- Fundamental **tradeoff** of **latency** and **recall**
  - Users interested in specific **recall targets**
  - ANNS algorithms have different hyperparameters → **extensive tuning!**



ANNS comes with cumbersome tuning!

The Tuning Process

- Tuning requires enormous time-consuming trial and error
  1. Select an ANNS algorithm
  2. Select a tuning query workload

The Tuning Process

- Tuning requires enormous time-consuming trial and error
  1. Select an ANNS algorithm
  2. Select a tuning query workload
  3. Initialize hyperparameters for desired target recall

The Tuning Process

- Tuning requires enormous time-consuming trial and error
  1. Select an ANNS algorithm
  2. Select a tuning query workload
  3. Initialize hyperparameters for desired target recall
  4. Build index → Wait several hours!

The Tuning Process

- Tuning requires enormous time-consuming trial and error
  1. Select an ANNS algorithm
  2. Select a tuning query workload
  3. Initialize hyperparameters for desired target recall
  4. Build index → Wait several hours!
  5. Check if chosen hyperparameters achieve **the recall target on average**

The Tuning Process

- Tuning requires enormous time-consuming trial and error
  1. Select an ANNS algorithm
  2. Select a tuning query workload
  3. Initialize hyperparameters for desired target recall
  4. Build index → Wait several hours!
  5. Check if chosen hyperparameters achieve **the recall target on average**
     - If not, **delete index**, pick new parameters, **start over**! ☹

Tuning is a resource- and time-intensive task!

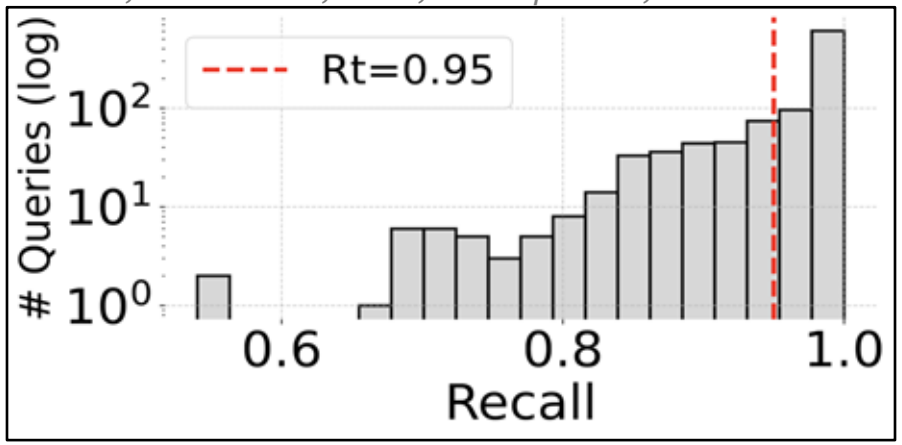
Problems After Tuning

- Performance might still not be optimal: Parameters predefined **average** recall over the workload
  - Cannot **adapt** to individual query hardness
  - Suboptimal results for many queries

Problems After Tuning

- Performance might still not be optimal: Parameters predefined **average** recall over the workload
  - Cannot **adapt** to individual query hardness
  - Suboptimal results for many queries

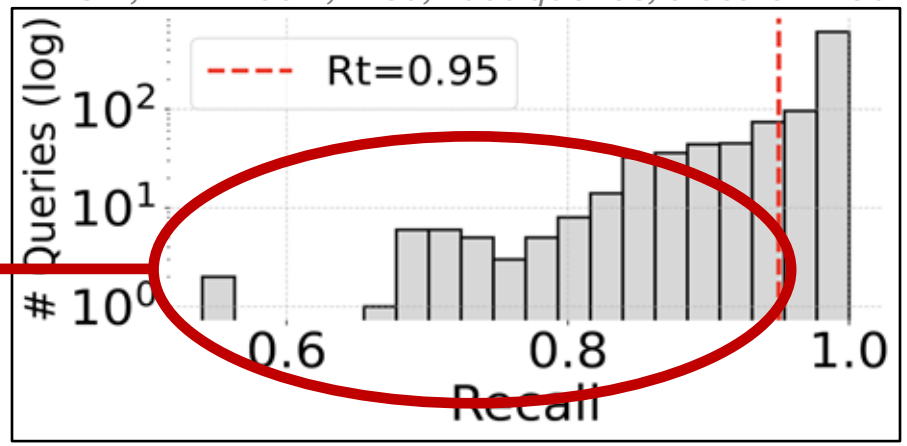
HNSW, DEEP100M, k=50, 1000 queries, efSearch=750



Problems After Tuning

- Performance might still not be optimal: Parameters predefined **average** recall over the workload
  - Cannot **adapt** to individual query hardness
  - Suboptimal results for many queries

HNSW, DEEP100M, k=50, 1000 queries, efSearch=750



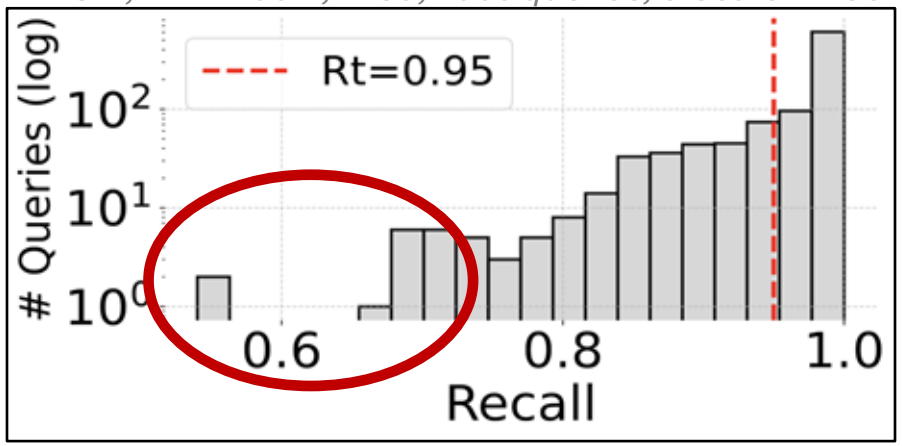
Semantically innacurate results!

Even after tuning, **many queries have suboptimal results!**

Problems After Tuning

- Performance might still not be optimal: Parameters predefined **average** recall over the workload
  - Cannot **adapt** to individual query hardness
  - Suboptimal results for many queries

HNSW, DEEP100M, k=50, 1000 queries, efSearch=750



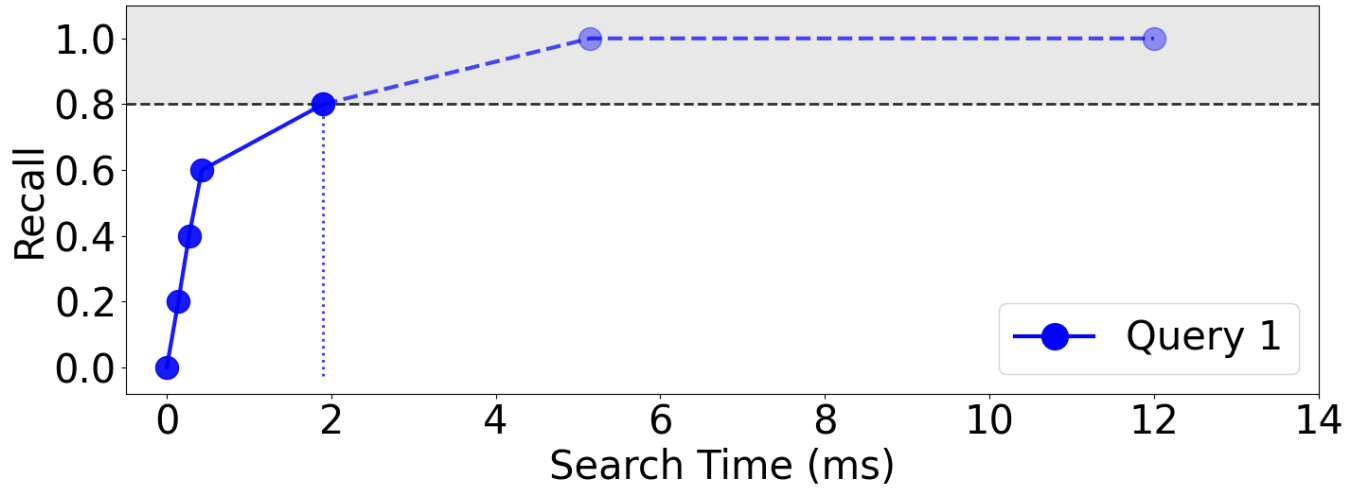
Even after tuning, **many queries have suboptimal results!**
some of them, **really bad results!**

Key Observation

- Query reaching very high recall satisfies all lower recalls as well
  - Early termination opportunities
  - Creating high-scoring index is easy

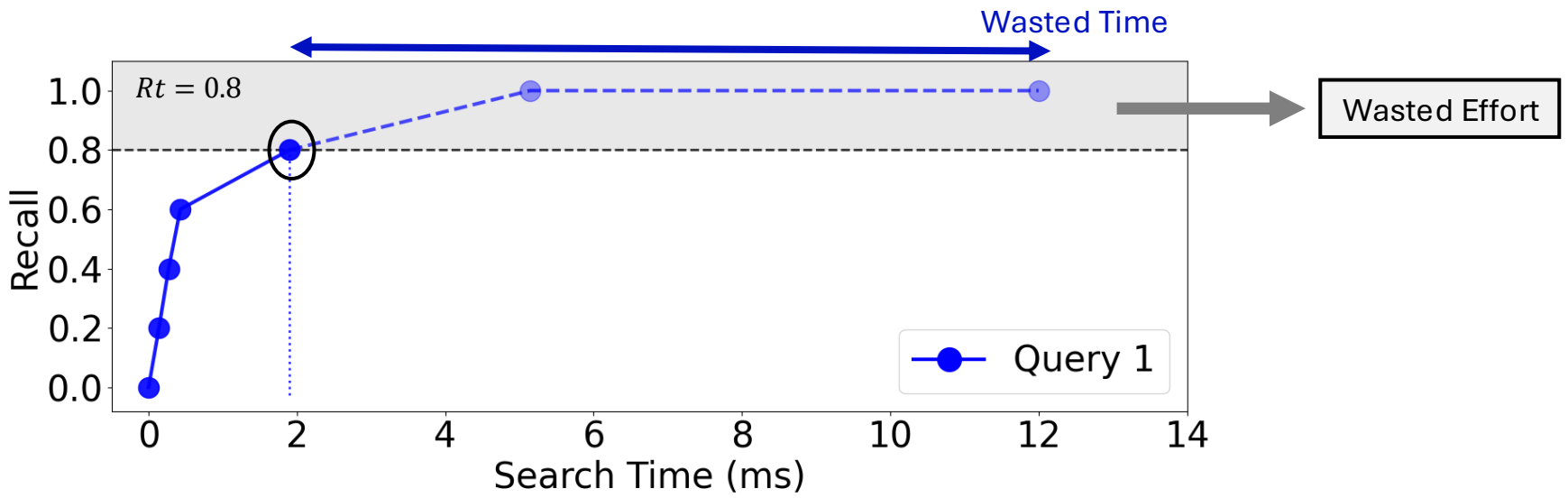
Key Observation

- Query reaching very high recall satisfies all lower recalls as well
  - Early termination opportunities
  - Creating high-scoring index is easy



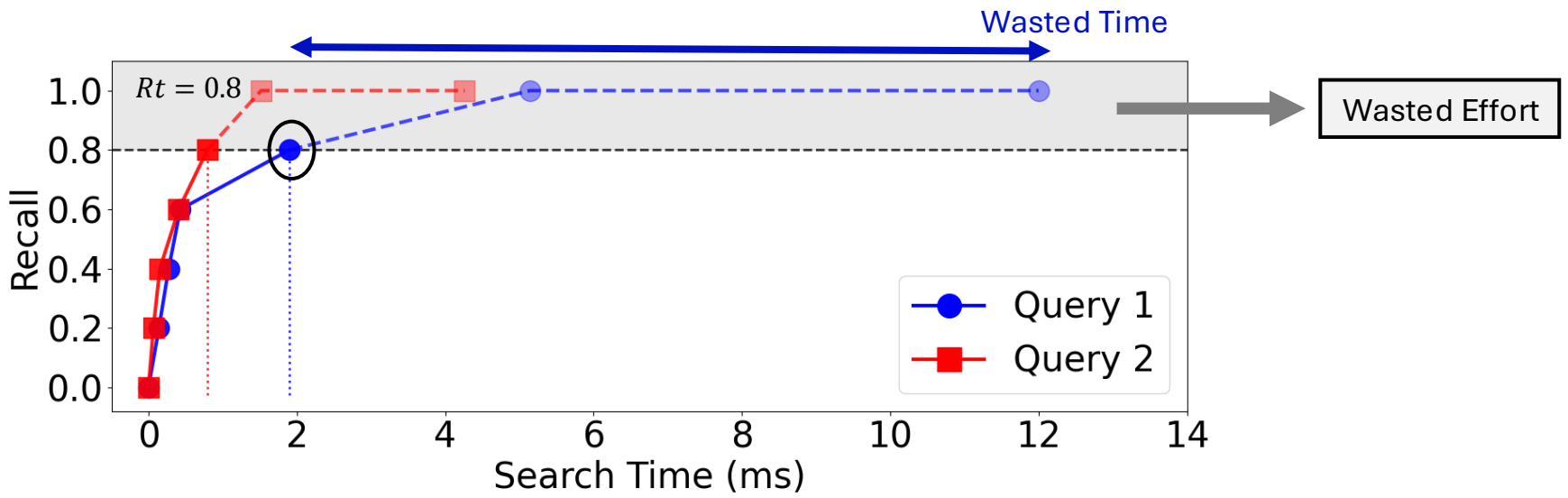
Key Observation

- Query reaching very high recall satisfies all lower recalls as well
  - Early termination opportunities
  - Creating high-scoring index is easy



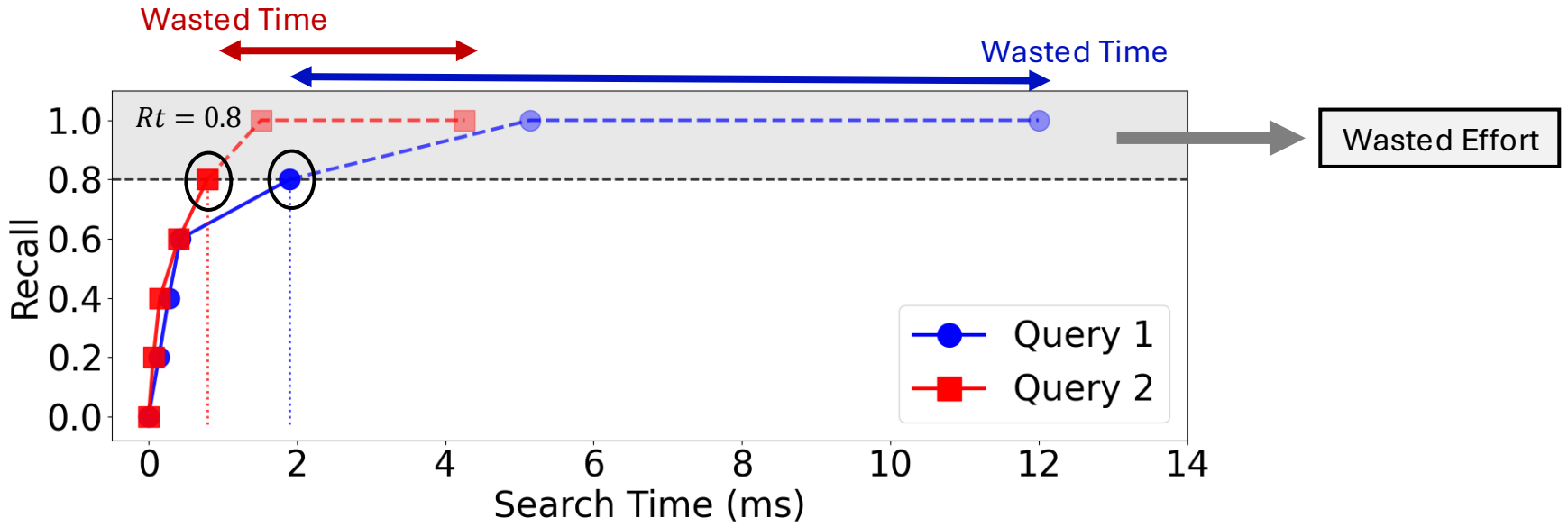
Key Observation

- Query reaching very high recall satisfies all lower recalls as well
  - Early termination opportunities
  - Creating high-scoring index is easy



Key Observation

- Query reaching very high recall satisfies all lower recalls as well
  - Early termination opportunities
  - Creating high-scoring index is easy

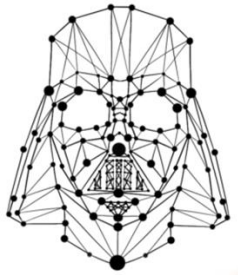


Different queries reach different recalls at different times!

Our Contributions

- Solution through **declarative recall**:
  - From ANNS(q, k, params) to **ANNS(q, k, Rt)!**

Our Contributions



- Solution through **declarative recall**:
  - From ANNS(q, k, params) to **ANNS(q, k, Rt)!**

- DARTH **predicts the query recall** at any point of the search → Early termination
  - Eliminate the need for parameter tuning
  - Support different indexes and distance measures
  - Support any Rt at query time and achieve it for each **individual** query →adapts to **query hardness**
  - Achieves **state-of-the-art results** when compared to baselines!

DARTH is SotA approach for declarative recall!

How to predict recall?

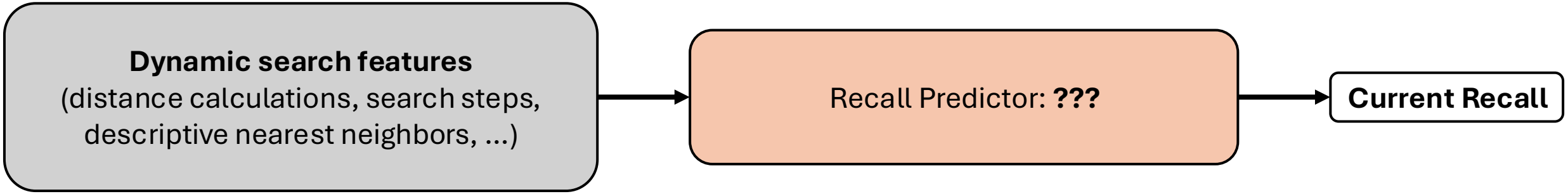
10K queries: 100M samples (many samples per query)

Dynamic search features
(distance calculations, search steps, descriptive nearest neighbors, ...)

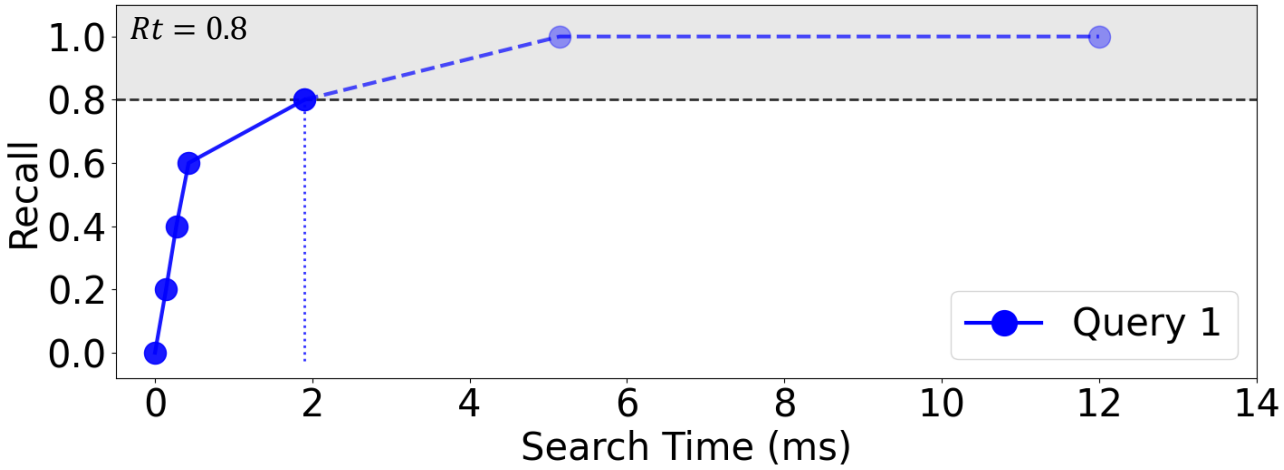
Current Recall

How to predict recall?

10K queries: 100M samples (many samples per query)



How to predict recall?



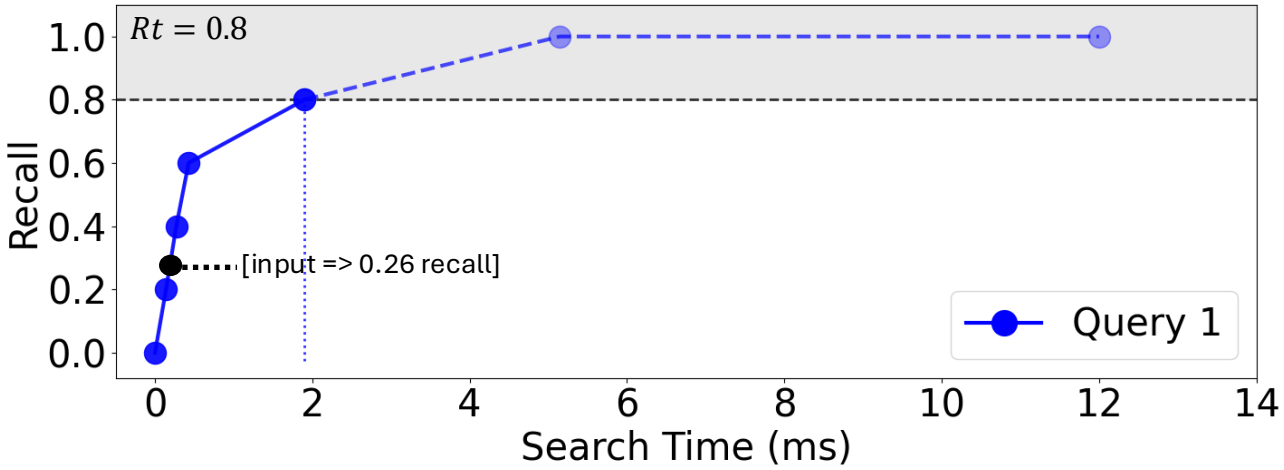
10K queries: 100M samples (many samples per query)

Dynamic search features
(distance calculations, search steps, descriptive nearest neighbors, ...)

→ Recall Predictor: **???** →

Current Recall

How to predict recall?



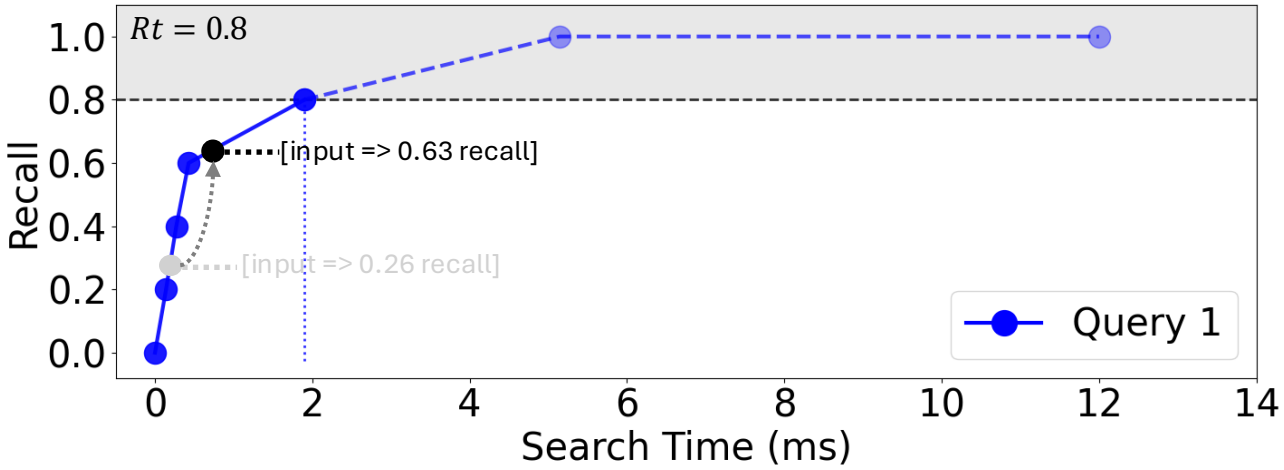
10K queries: 100M samples (many samples per query)

Dynamic search features
(distance calculations, search steps, descriptive nearest neighbors, ...)

→ Recall Predictor: **???** →

Current Recall

How to predict recall?



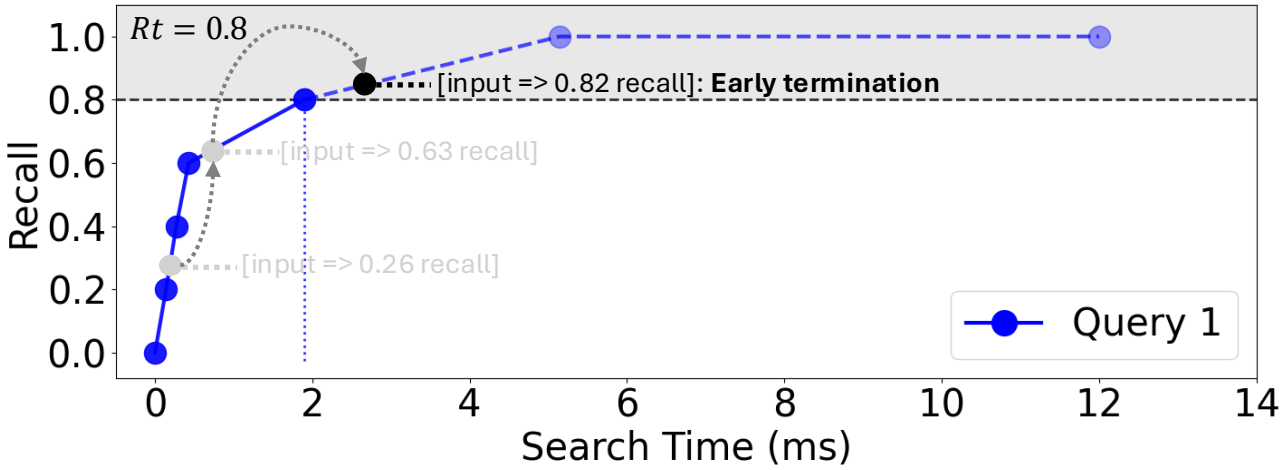
10K queries: 100M samples (many samples per query)

Dynamic search features
(distance calculations, search steps, descriptive nearest neighbors, ...)

→ Recall Predictor: **???** →

Current Recall

How to predict recall?



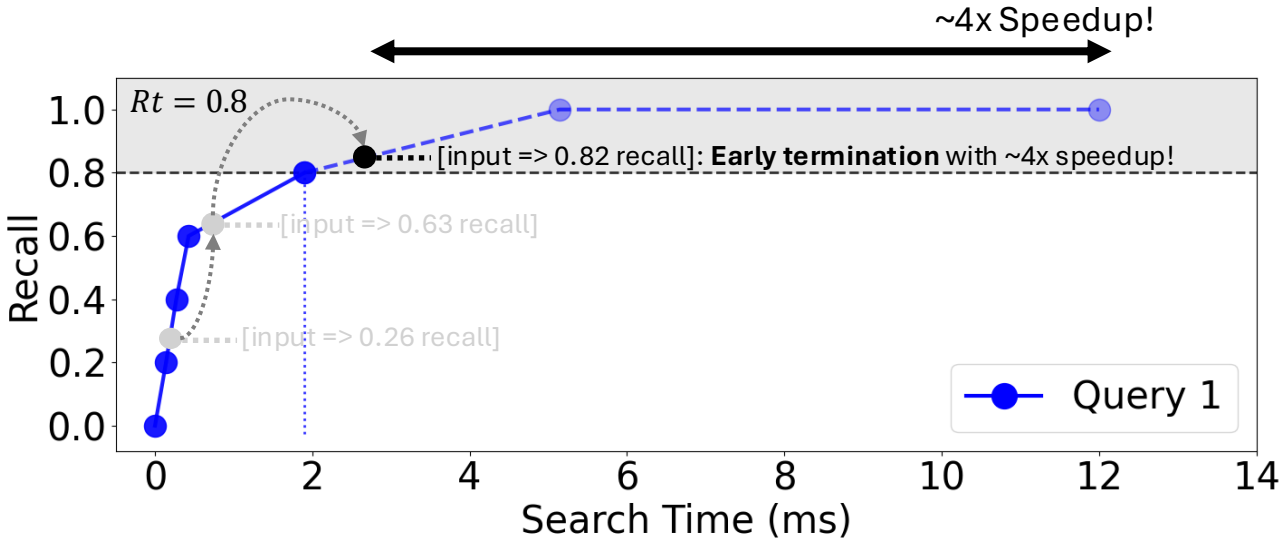
10K queries: 100M samples (many samples per query)

Dynamic search features
(distance calculations, search steps, descriptive nearest neighbors, ...)

→ Recall Predictor: **???** →

Current Recall

How to predict recall?



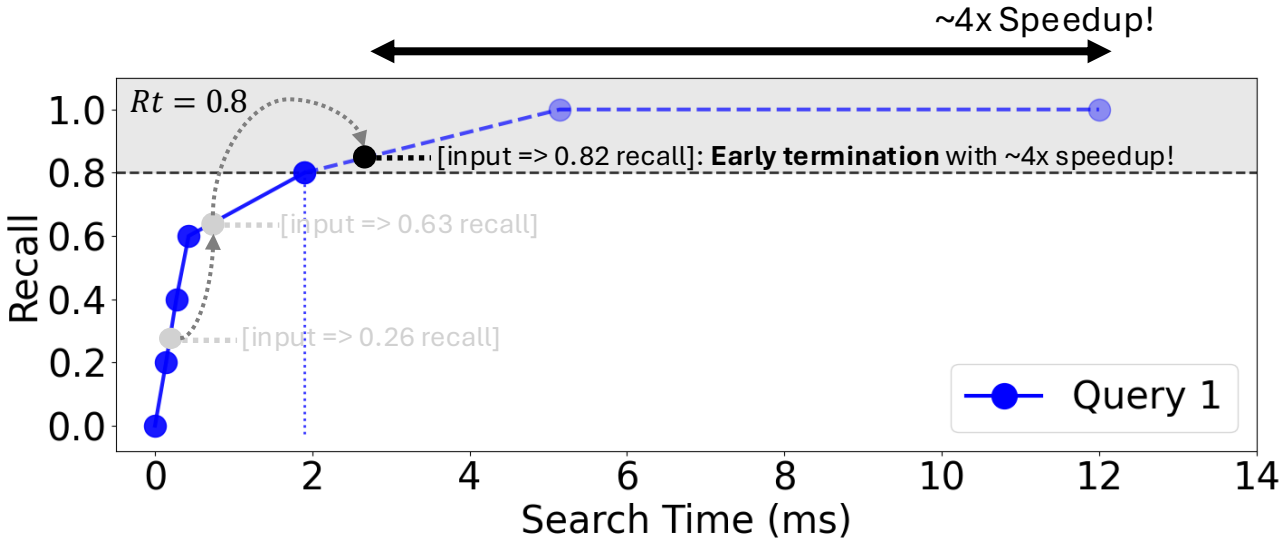
10K queries: 100M samples (many samples per query)

Dynamic search features
(distance calculations, search steps, descriptive nearest neighbors, ...)

→ Recall Predictor: **???** →

Current Recall

How to predict recall?



10K queries: 100M samples (many samples per query)

Dynamic search features
(distance calculations, search steps, descriptive nearest neighbors, ...)

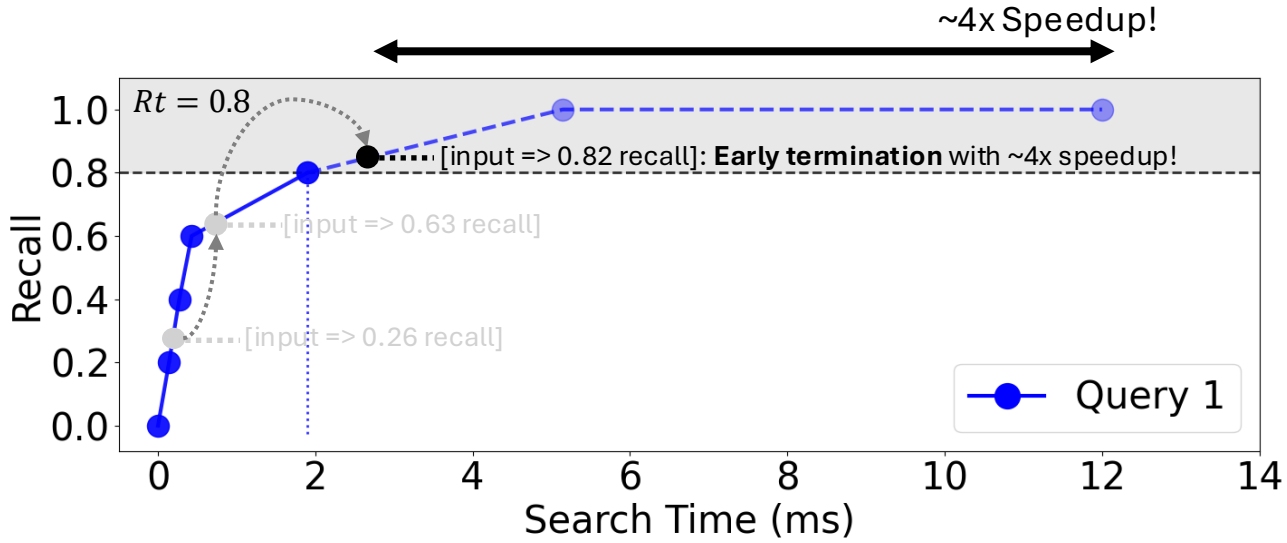
→

Recall Predictor: **GBDT**
-Trained in seconds for 100Ms samples
-Inference in 0.03ms

→

Current Recall

How to predict recall?



10K queries: 100M samples (many samples per query)

Dynamic search features
(distance calculations, search steps, descriptive nearest neighbors, ...)

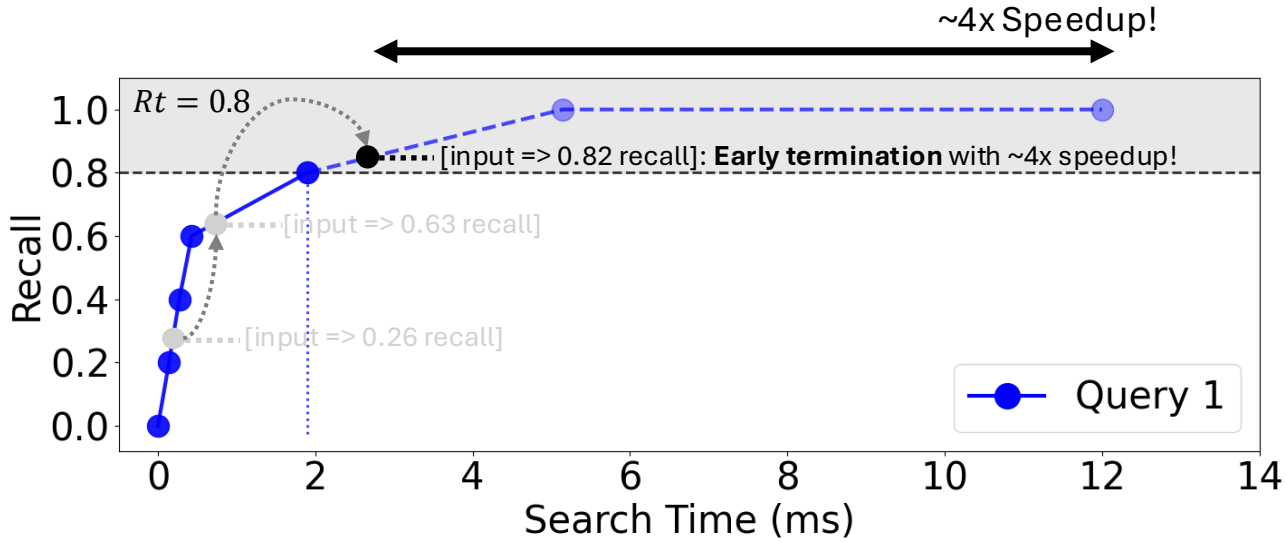
Recall Predictor: **GBDT**
-Trained in seconds for 100Ms samples
-Inference in 0.03ms

Current Recall

Adaptive predictor invocation strategy
Call the predictor
- more often when closer to the recall target!
- less often when far away from it!

Very fast training and inference, optimized invocations!

How to predict recall?



10K queries: 100M samples (many samples per query)

Dynamic search features
(distance calculations, search steps, descriptive nearest neighbors, ...)

→ Recall Predictor: **GBDT**
-Trained in seconds for 100Ms samples
-Inference in 0.03ms

→ **Current Recall**

Adaptive predictor invocation strategy
Call the predictor
- more often when closer to the recall target!
- less often when far away from it!

adopted by Google

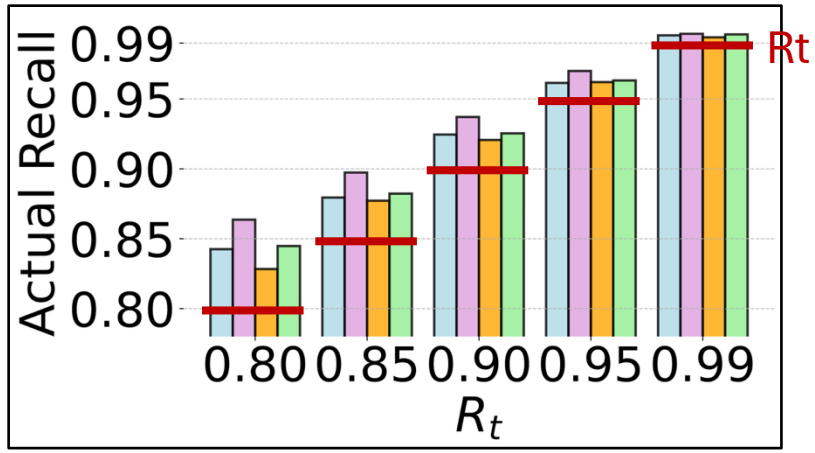
Very fast training and inference, optimized invocations!

Overview of Results

FAISS & LightGBM (C/C++), k=50, 1000 queries

SIFT100M DEEP100M GLOVE1M GIST1M

DARTH accuracy:
actual recall vs target recall

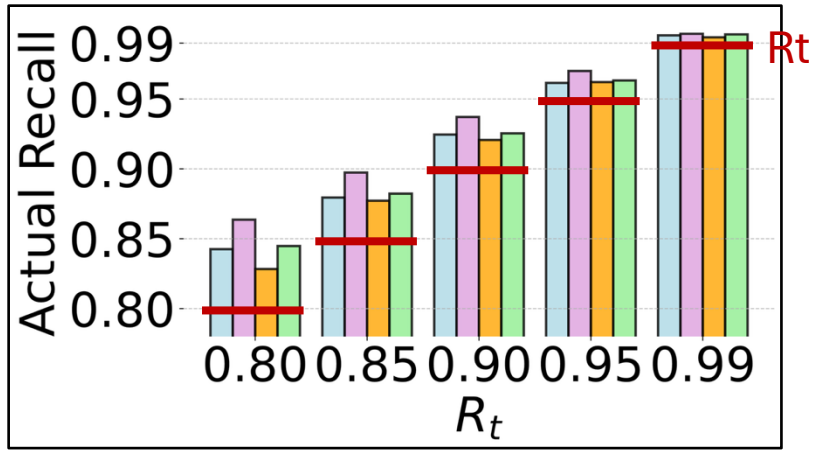


Overview of Results

FAISS & LightGBM (C/C++), k=50, 1000 queries

SIFT100M DEEP100M GLOVE1M GIST1M

DARTH accuracy:
actual recall vs target recall



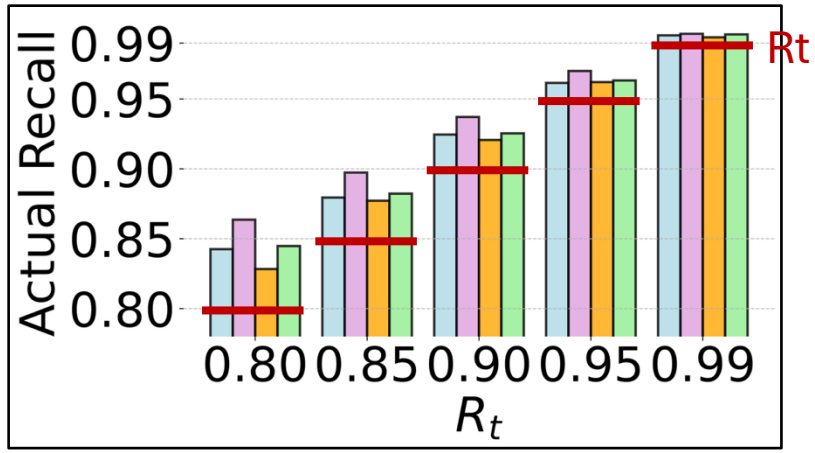
achieves **all recall targets**
across different datasets!

Overview of Results

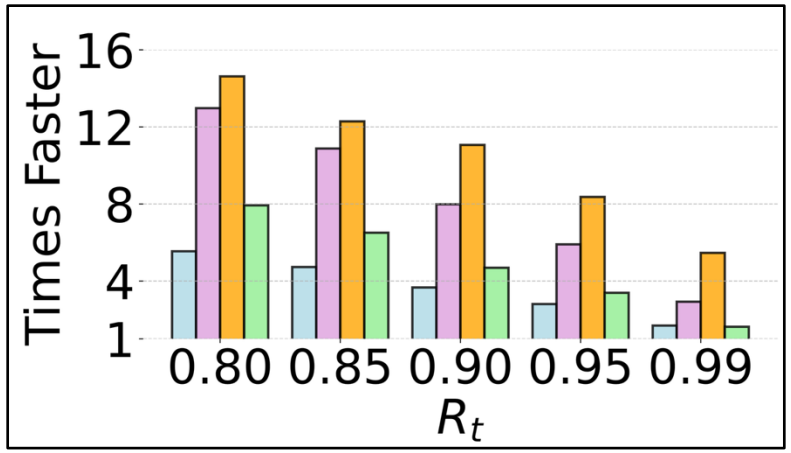
FAISS & LightGBM (C/C++), k=50, 1000 queries

SIFT100M DEEP100M GLOVE1M GIST1M

DARTH accuracy:
actual recall vs target recall



DARTH query answering speed:
speedup (over plain search) vs target recall



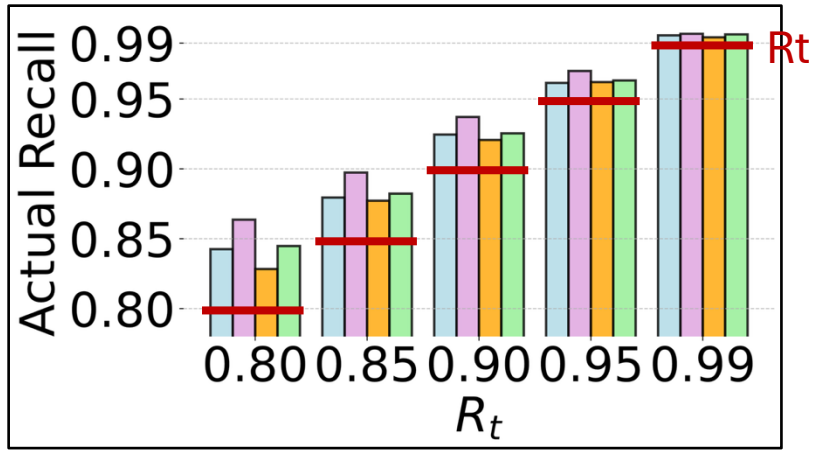
achieves **all recall targets**
across different datasets!

Overview of Results

FAISS & LightGBM (C/C++), k=50, 1000 queries

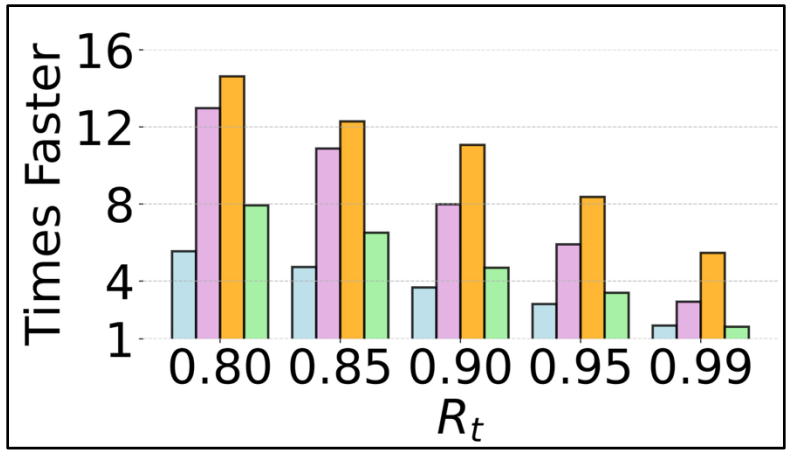
SIFT100M DEEP100M GLOVE1M GIST1M

DARTH accuracy:
actual recall vs target recall



achieves **all recall targets** across different datasets!

DARTH query answering speed:
speedup (over plain search) vs target recall

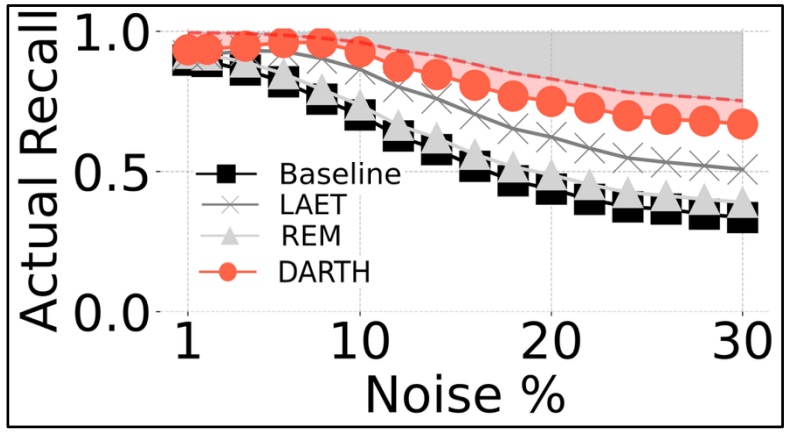


speedups **up to 15x compared to plain search** without early termination!

Overview of Results

FAISS & LightGBM (C/C++), k=50, 1000 queries

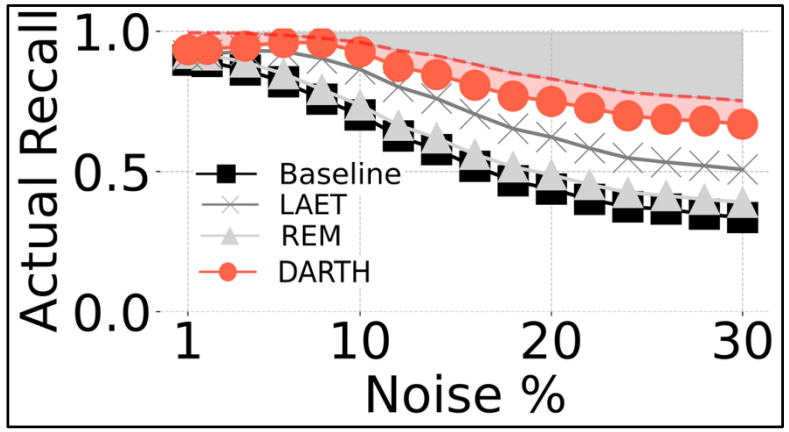
DARTH resilience:
actual recall vs query hardness



Overview of Results

FAISS & LightGBM (C/C++), k=50, 1000 queries

DARTH resilience:
actual recall vs query hardness



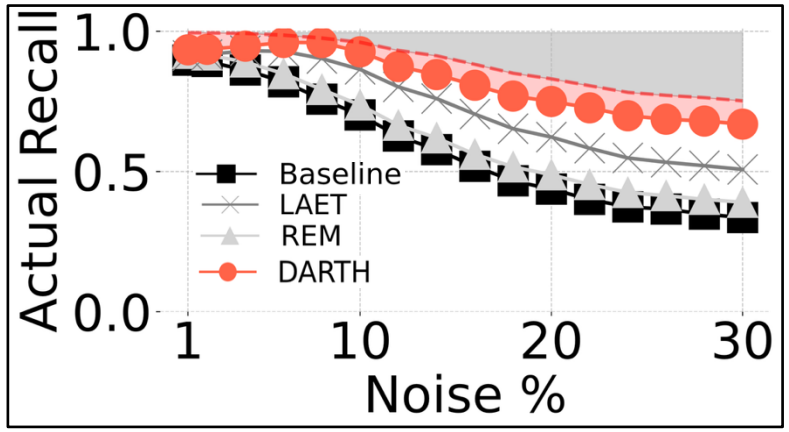
better

Only method that maintains high recall for **progressively harder query workloads**

Overview of Results

FAISS & LightGBM (C/C++), k=50, 1000 queries

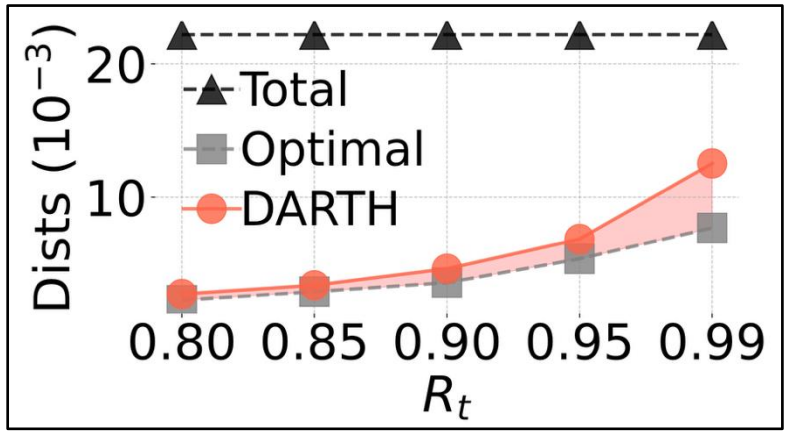
DARTH resilience:
actual recall vs query hardness



better

Only method that maintains high recall for **progressively harder query workloads**

DARTH early termination quality:
Deviation of termination points from the optimal oracle

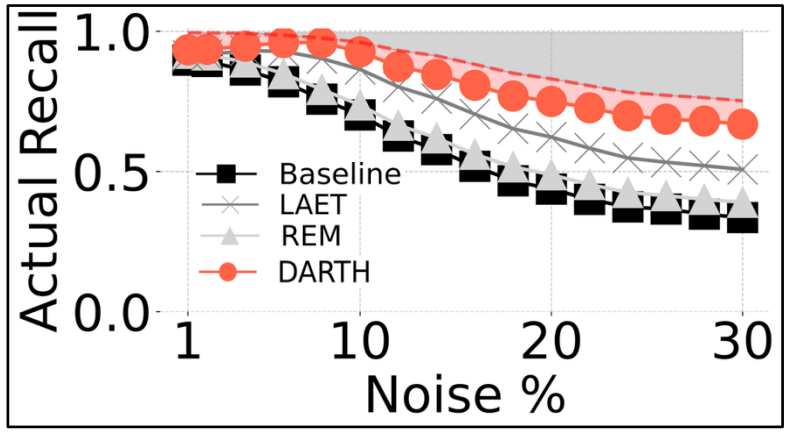


better

Overview of Results

FAISS & LightGBM (C/C++), k=50, 1000 queries

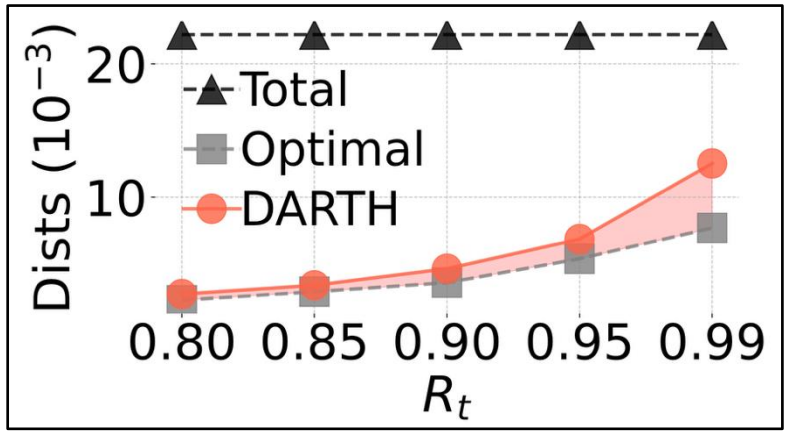
DARTH resilience:
actual recall vs query hardness



better

Only method that maintains high recall for **progressively harder query workloads**

DARTH early termination quality:
Deviation of termination points from the optimal oracle



better

Near-optimal early termination quality!

Conclusions

- Declarative recall can provide a very useful interface for ANNS
- DARTH
  - Accurate and efficient approach for declarative target recall
  - Early termination through recall prediction and adaptive intervals
  - SotA results!
- Future work
  - Probabilistic quality guarantees for predictions
  - Filtered vector search

Artifacts, ongoing works, and more!

Thank you!