BLISS: A Billion scale Index using Iterative Re-partitioning [KDD'22]

Authors:

Gaurav Gupta, Tharun Medini, Anshumali Shrivastava, and Alexander J. Smola

Presented by:

Kyung Jae Lee

March 05, 2025

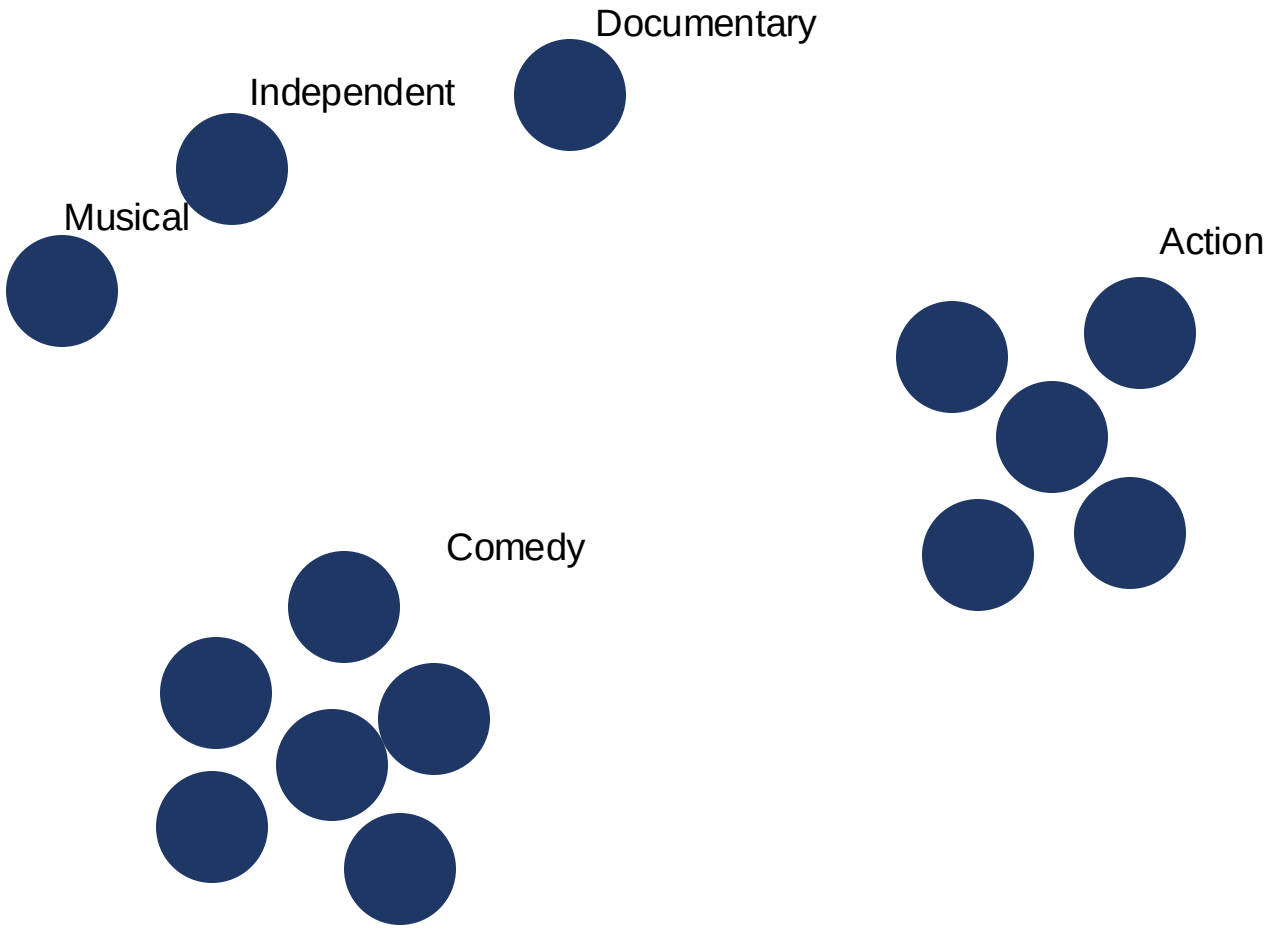
UNIVERSITY OF TORONTO

DEFY GRAVITY

Cluster-Based Indexes

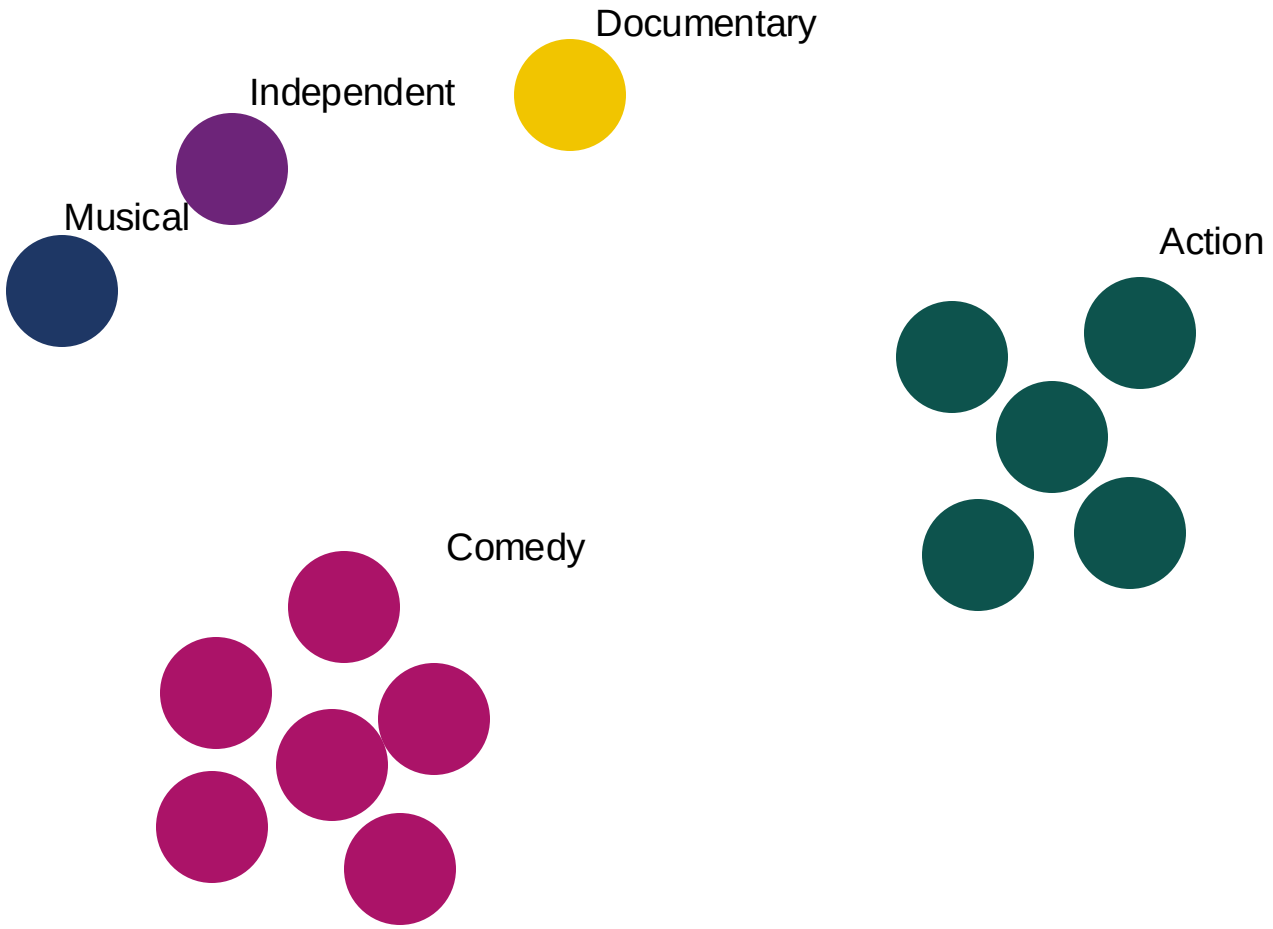
- Assign vectors in dataset to clusters to construct index, and map query vector to small set of clusters likely to contain nearest neighbours
- Examples:
  - Locality Sensitive Hashing
  - K-Means Based (IVF)

The Curse of Clustering



- Real world data often follow a power-law relationship
- Hotspots and sparse regions in the embedding space

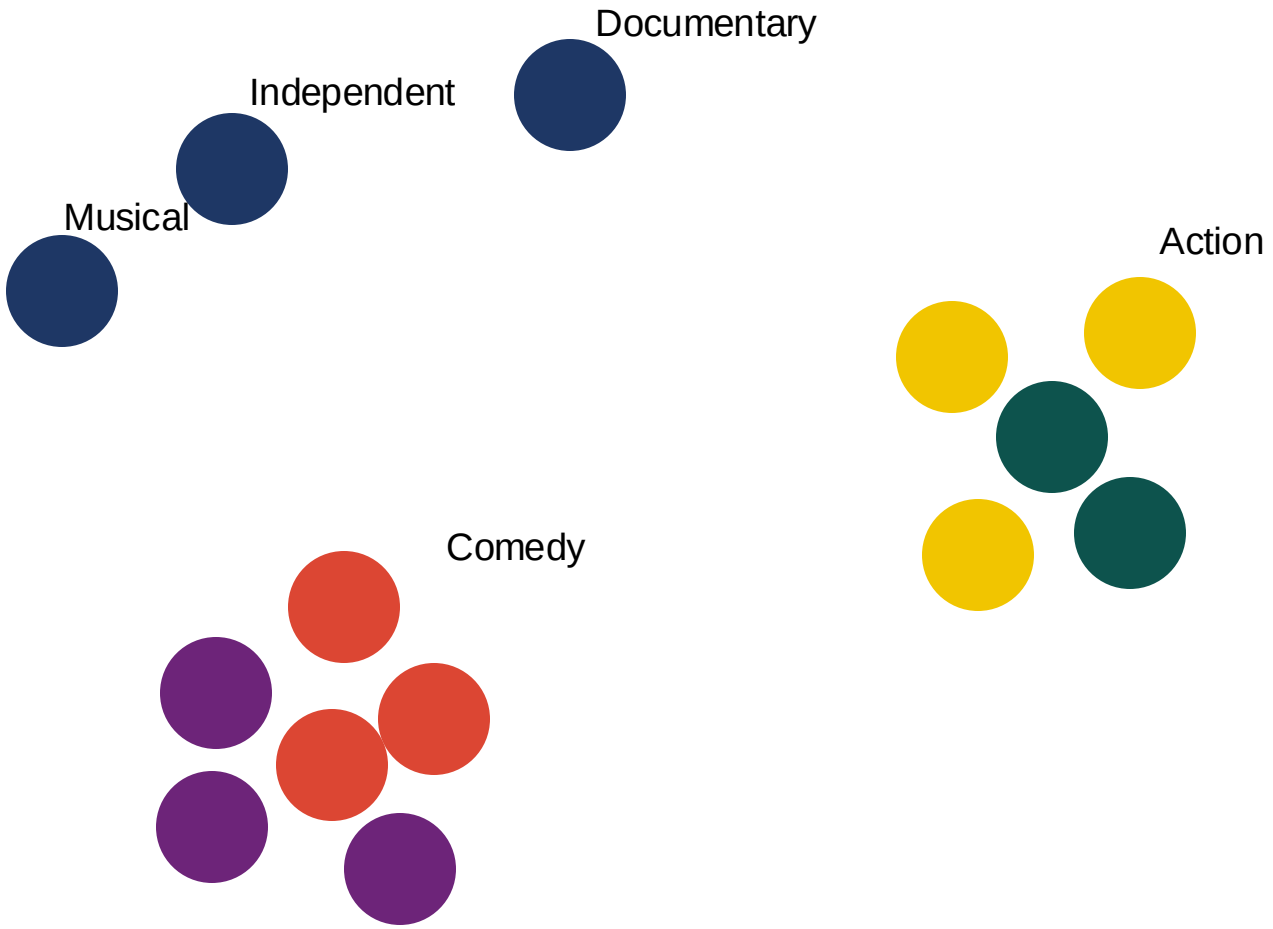
Unbalanced Clustering



- Overutilized clusters
- Higher latency

UNIVERSITY OF TORONTO

Balanced Clustering



- The “curse”:
  - Similar vectors forced to split
  - Irrelevant vectors forced together

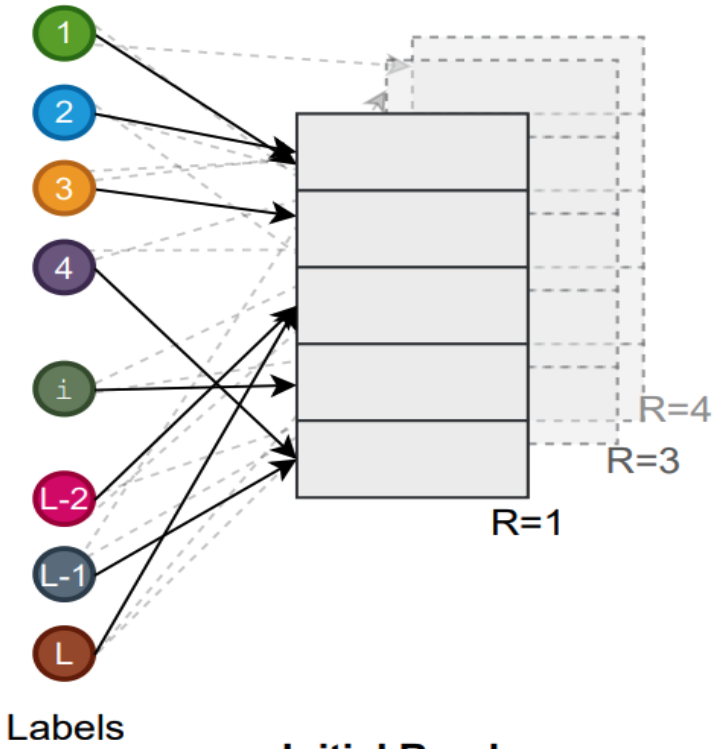
BLISS: BaLanced Index for Scalable Search

High-level learning process:

1. Train model to predict bucket assignment
2. Redistribute vectors according to model
3. Repeat

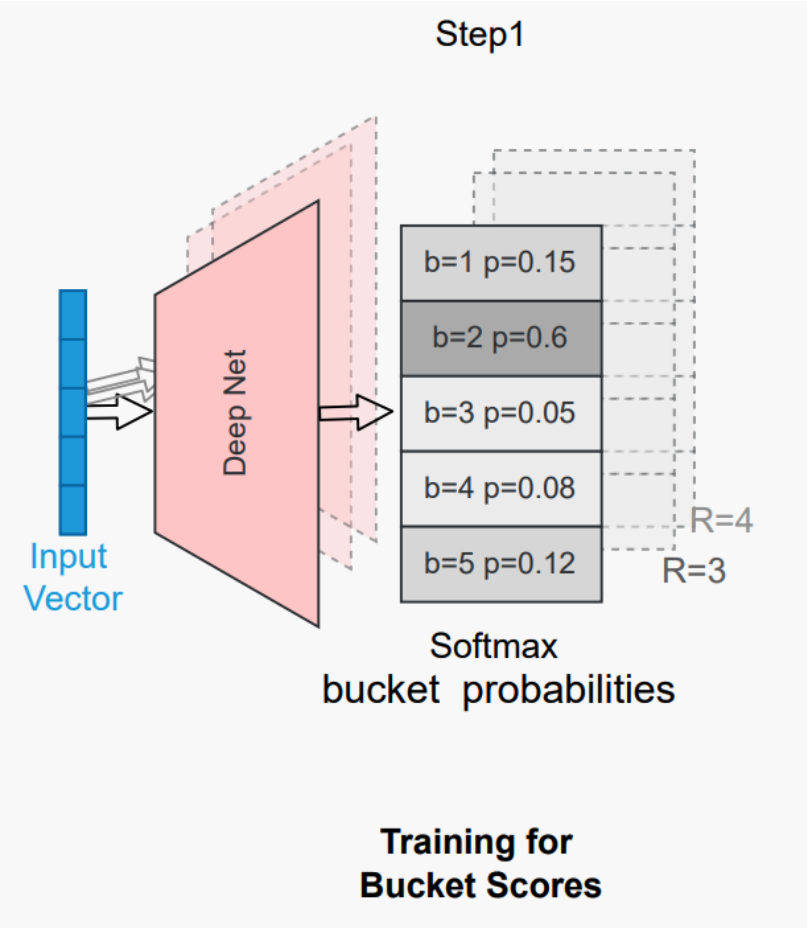
Trained model provides probabilistic assignment of vectors to buckets

BLISS Initialization



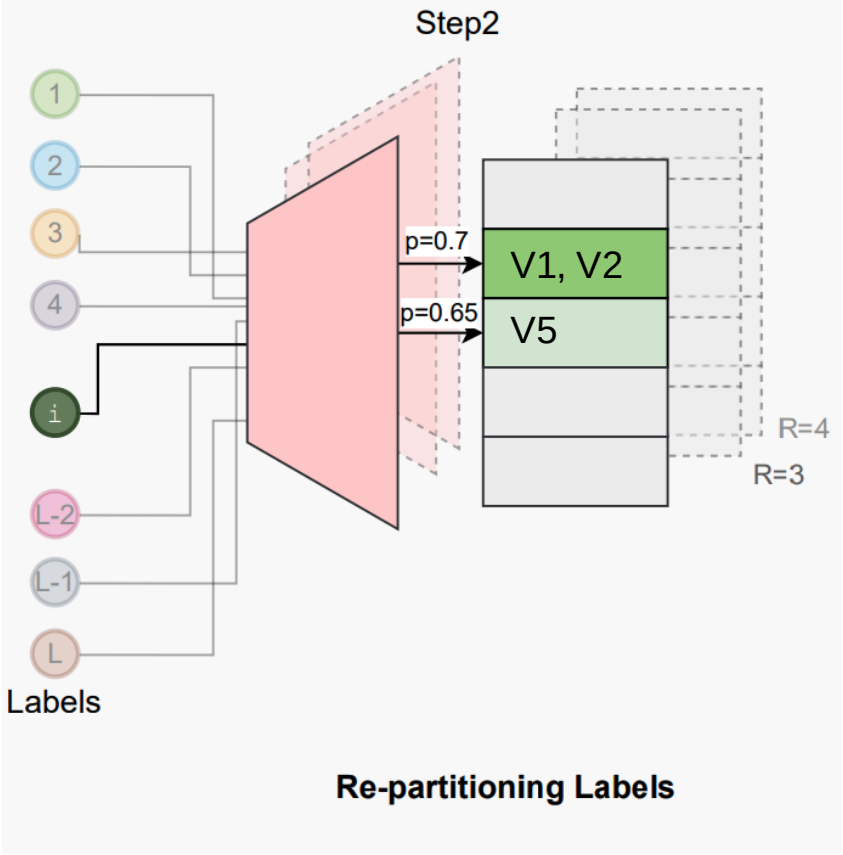
- Goal: uniform bucket sizes
- Random initial assignment into B buckets

BLISS Model Training



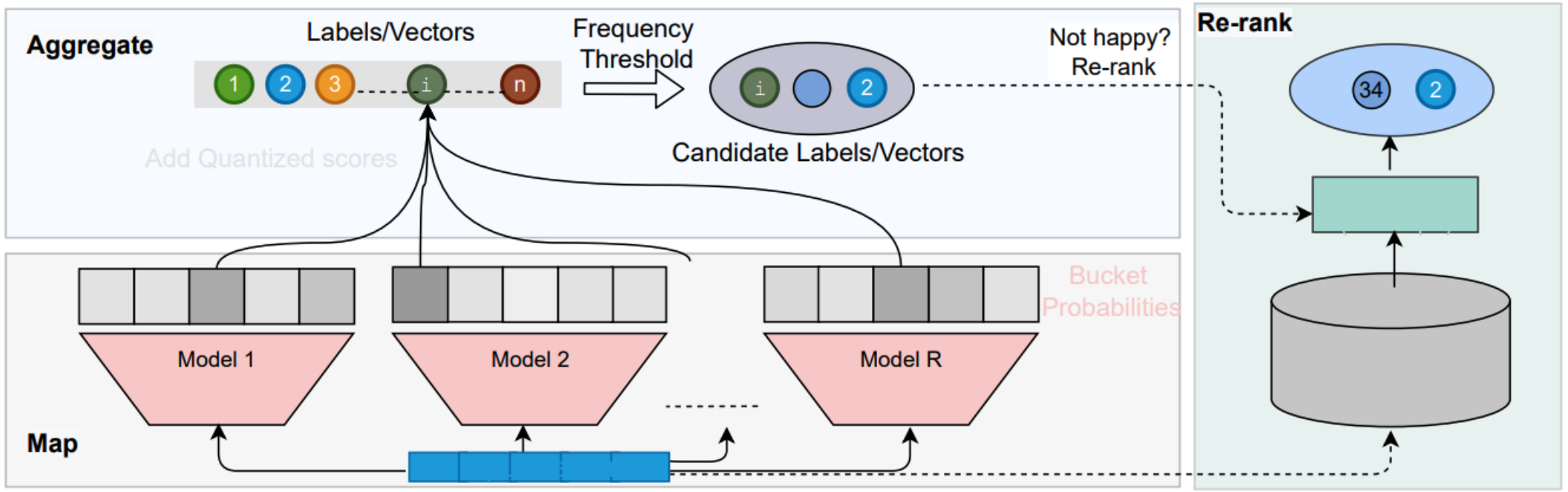
- Goal: train model to “score” bucket given a vector
- MLP with a single hidden layer
- Data: $(x, \bar{y})$ pairs
  - x: vector/query in dataset
  - $\bar{y}$: ground-truth 100 closest vectors to x (pre-computed)
- Cross-entropy loss over all buckets

BLISS Re-partitioning



- Use trained model to get "better" bucket assignments
- Re-assign each vector to least occupied of top K buckets
  - Incentivize load balancing

BLISS Querying



BLISS Configurations

- K: Load balancing parameter
  - 2 ~ 10
- R: Number of (independent) repetitions
  - 1 ~ 4
- B: Number of buckets in system
  - $O(\sqrt{N})$
- m: How many buckets to probe
  - 5 ~ 20
- Training scheme

BLISS+

BLISS+ Reordering

Vector	Bucket	Position
V1	2	0
V2	1	1
V3	3	2
V4	2	3
V5	3	4
V6	1	5

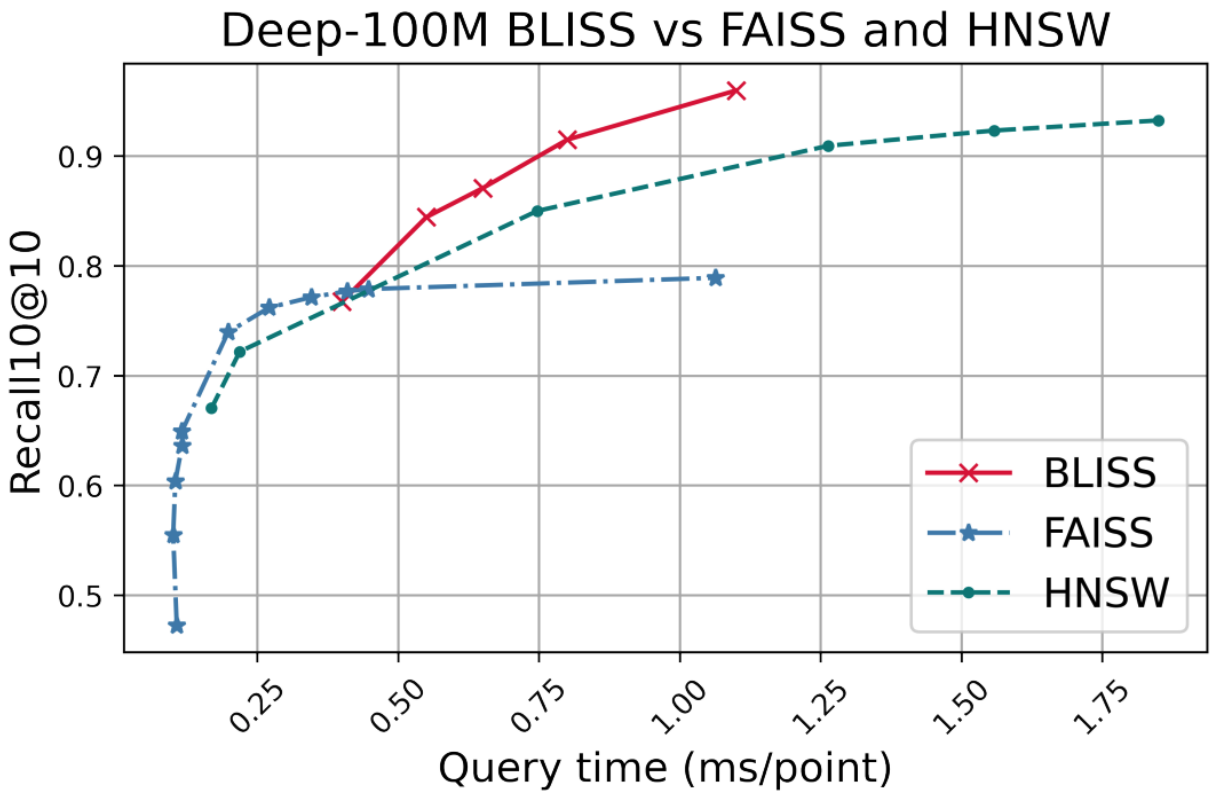
Vector	Bucket	Position
V2	**1**	**0**
V6	1	1
V1	**2**	**2**
V4	2	3
V3	**3**	**4**
V5	3	5

- Optimized for index size
  - Use only a single model
  - Reorder data and only store offsets
- 137M memory for 1B dataset
  - 6000x smaller than HNSW
  - 1500x smaller than FAISS-IVFFlat
  - 100x smaller than BLISS

Experiment Setup

- 1M, 100M, 1B scale datasets
- Use 1% of dataset for training data
- Hardware:
  - Server with 64 cores and 1.5 TB RAM (no GPU)
  - Only use 32 threads to batch queries

Experiments



- BLISS performs well compared to baselines
- No data points for BLISS for low latency regions
  - Presumably overhead in running inference on MLP
- BLISS also has a smaller index size

Experiments

BIGANN TI: 1B image dataset (BLISS doesn't work as well)

	QPS	Recall10@10	Index Size	Construction Time
BLISS	121	0.8443	**15.5GB**	**1hr**
BLISS+	344	0.658	**137MB**	**1.1hr**
HNSW	**909**	**0.8734**	557GB	10hr
FAISS	243	**0.8764**	127GB	>5 days

Future Work

- How well does BLISS generalize to different data / react to updates?
- How long does it take to (re-)train the model?
- More rigorous experiments/analysis for billion-scale datasets
- Is the choice of neural network architecture and training setup optimal?

Thank you!

Questions?

UNIVERSITY OF TORONTO

DEFY GRAVITY

Experiments

Yandex TI: 1B text-to-image dataset (BLISS outperforms due to nature of dataset)

	QPS	Recall10@10	Index Size	Construction Time
BLISS	**110**	**0.568**	**15.5GB**	**1hr**
BLISS+	**384**	0.434	**137MB**	**1.3hr**
HNSW	15	0.566	826GB	16hr
FAISS	4	0.4919	194GB	>5 days

Appendix: How is the “Curse of Clustering” Addressed?

- Main idea: bucket assignments are probabilistic and replicated across R independent trials
  - Functionally similar to assigning to multiple clusters
- Forced to split similar vectors
  - Lower chance of relevant clusters being lost with R repetitions
- Forced to group irrelevant vectors
  - Irrelevant vectors are filtered out by the threshold filter and not considered in the final candidate set

Appendix: Alternative Architectures?

- This ensemble-like approach using R models improves robustness and parallelizability, and helps with curse of clustering
- But the MLP may be too simple to effectively learn this (complex) vector to bucket association

Appendix: Modifications for XML

In XML, the labels are not vectors, so to perform re-partitioning, we need a way to map a label to a bucket

1. For every label, we find all the vectors x which have the label as a ground truth label
2. We compute and sum up $f(x)$ for all such vectors x. This will give you a B dimensional vector that provides scores over buckets
3. Everything else is the same.