Faster reverse image search using LSH

Dinesh Avula - CS20BTECH11005
Ayush Jha - CS20BTECH11006

April 28, 2023

Table of Contents

Goals of implementation

Main goal

- The main goal of our implementation is to speed up the query of Top-K similar images to given query without losing too much in accuracy

Table of Contents

Motivation for using LSH

Baseline algorithm

There could be multiple simple approaches to the problem firstly,

- We can iterate overall images for the dataset in order to find the similar images
- We can clearly see the problem associated with this approach namely, $\mathcal{O}(n)$
- Usually we have very large datasets so approach not scalable

Solution to the problem

- As we have a large dataset we need to speedup our query for that we use LSH in a specific manner to restrict our search space to a smaller dataset

Motivationn for using CNNs

How to give input to LSH?

There could be multiple simple approaches to the problem firstly,

- Can we vectorize the input images and convert them into vectors?
- Clearly the problem will be the large size of input vectors and no semantic meaning related to the images,

Figure: Penguin

Figure: Also penguin

What is CNN?

- Images can vary greatly in complexity, containing different objects, textures, colors, and shapes.
- To capture the semantic and contextual details from an image, a model is needed that can learn the relevant features.
- Convolutional Neural Networks (CNNs) are a type of deep learning model that can learn these features.

Conversion from Image to Vectors

- CNNs can just be thought of Image2vec algorithm
- We use a pretrained Resnet50 model, pass the images and get out the $d = 1000$ dimensional vectors, which we use as inputs for the LSH.

What is LSH

- LSH or Locality Sensitive Hashing is an algorithm that is used to hash entries into bins wil some guarentees.
- The guarentees of LSH for a given tuple (r, Cr, p_1, p_2) and inputs x,y are
  - If $distance(x, y) < r$ then $Pr_{h \in H}[h(x) = h(y)] \geq p_1$ and
  - If $distance(x, y) > Cr$ then $Pr_{h \in H}[h(x) = h(y)] \leq p_2$

  where H is the Universal hash family and C is some constant that is greater than 1

Various LSH

There are multiple types of LSH that can be used some of them are

- LSH using min-hash
- LSH using sim-hash
- LSH using cosine similarity

LSH Algorithm intuition

Key intuition

- Our algorithm takes a input in the form of 1000-dimensional vectors
- We run a cosine similarity based LSH and two vectors fall into the same bin if their cosine similarity is high

LSH algorithm intuition

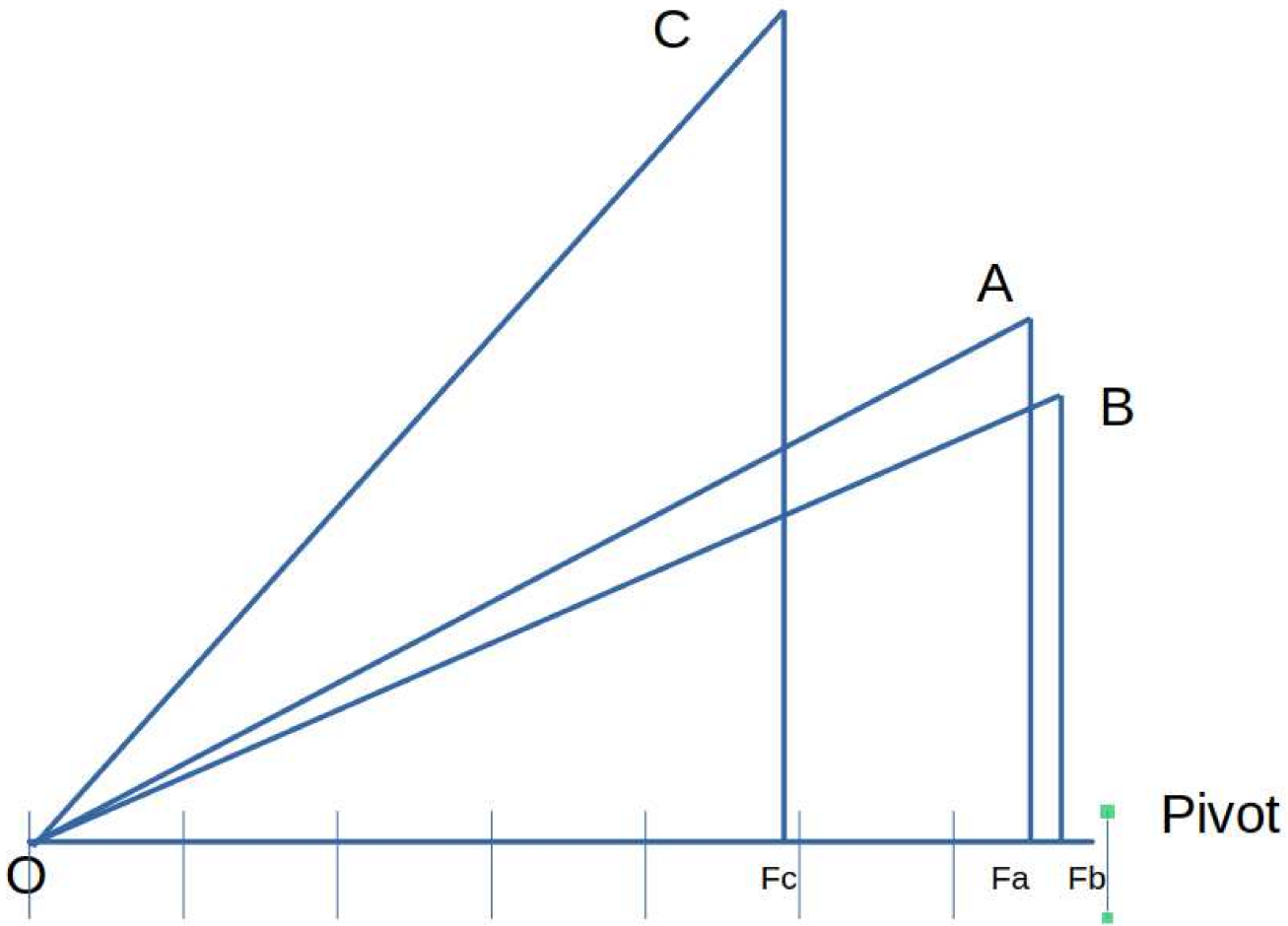

Figure: A small representation of our LSH intuition

LSH algorithm intuition

Intuition of our algo

- Running LSH on the whole dataset before hand will help us shrinking the search space for the Top-K similar images search algorithm
- We take an union of all the bins that our LSH hases to for a given query over L iterations
- As LSH hashes similar items into same bins we expect all the items similar to the query to be in the hashed bins of query item

LSH algorithm

Input: unit d-dimensional output vectors of CNN ,hash size k, number of iterations L
Output: Sketch containing similar inages in the same bins
$i \leftarrow 0$;
$N \leftarrow$ size of the input dataset ;
sketch $\leftarrow$ initialise matrix of $L \times N$;
while $i < L$ **do**
    generate unit pivot P vector of d dimensions;
    store the pivot corresponding to the iteration L;
    $j \leftarrow 0$;
    while $j < N$ **do**
        sketch[i][j] $\leftarrow$ floor(P . data[j])
    end
end
return: sketch

Algorithm 1: LSH for query

Input: unit d-dimensional vector dataset,unit d-dimensional query vector,hash sketch,
Output: vectors of corresponding images that are in the same bins as the query image queryBucket ← dynamic array of d dimensions
while $i < L$ **do**
    P ← pivot corresponding to ith iteration ;
    while $j < N$ **do**
        if *query.P==data[j].P* **then**
            similarVectors = similarVectors ∪ data[j]
        end
    end
end
return: queryBucket;

Algorithm 2: LSH for query

LSH algorithm

time complexity

Time complexity comparisons

- The baseline time complexity, linear search is $\mathcal{O}(n)$
- The asymptotic time complexity for LSH is $\mathcal{O}(\frac{L}{K} \times n)$

where n is the total number of points in the dataset, i.e. 10^5

Table of Contents

Experimental Results





Experimental Results





Experimental Results

Experimental Results

Comparison

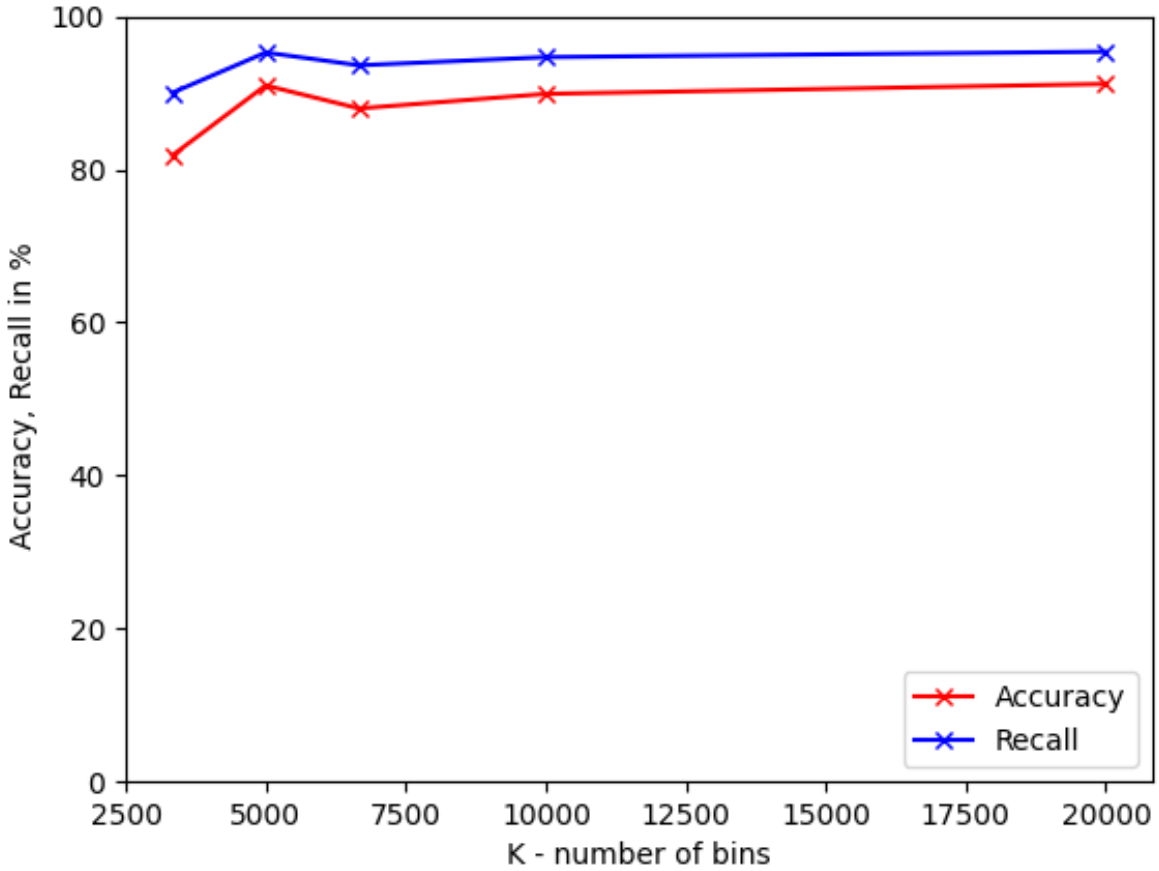

Figure: Accuracy vs K, L = 10

Experimental Results

Comparison

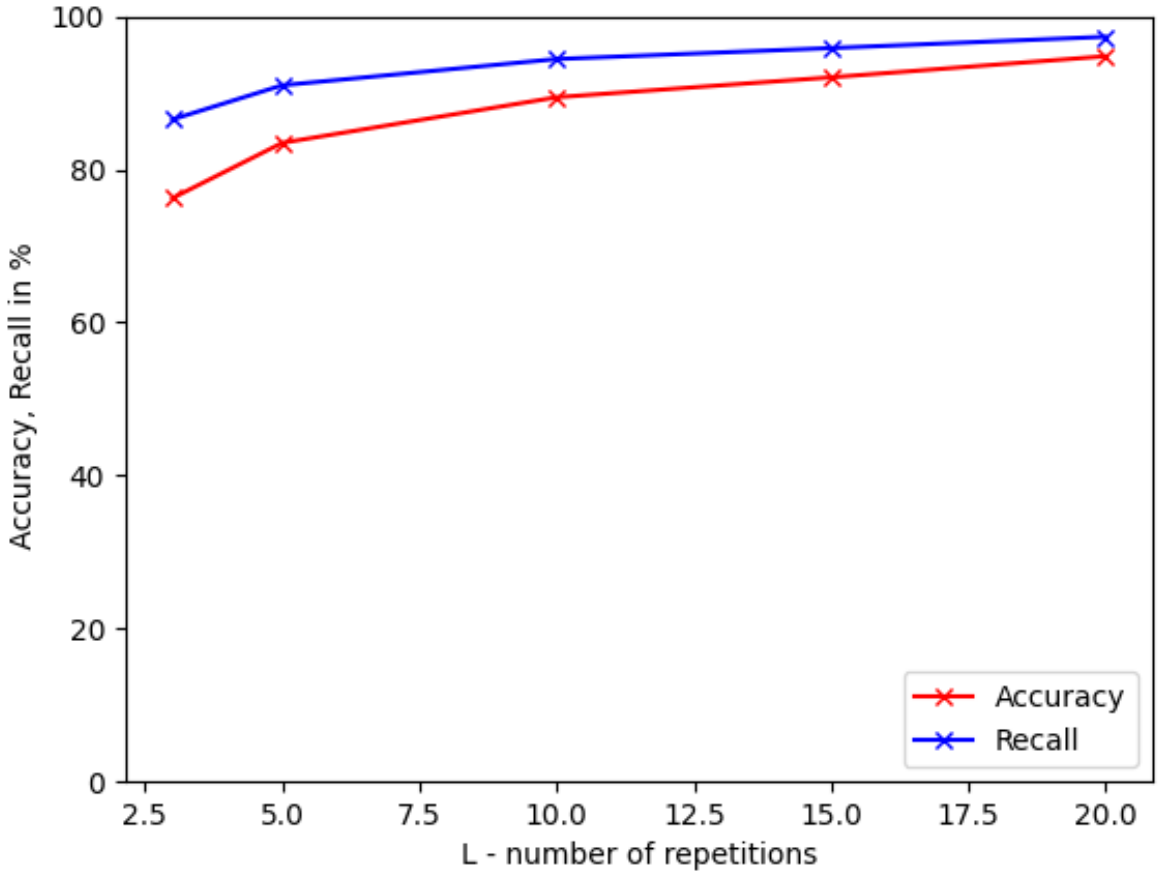

Figure: Accuracy vs L, K = 5000

Experimental Results

Comparison

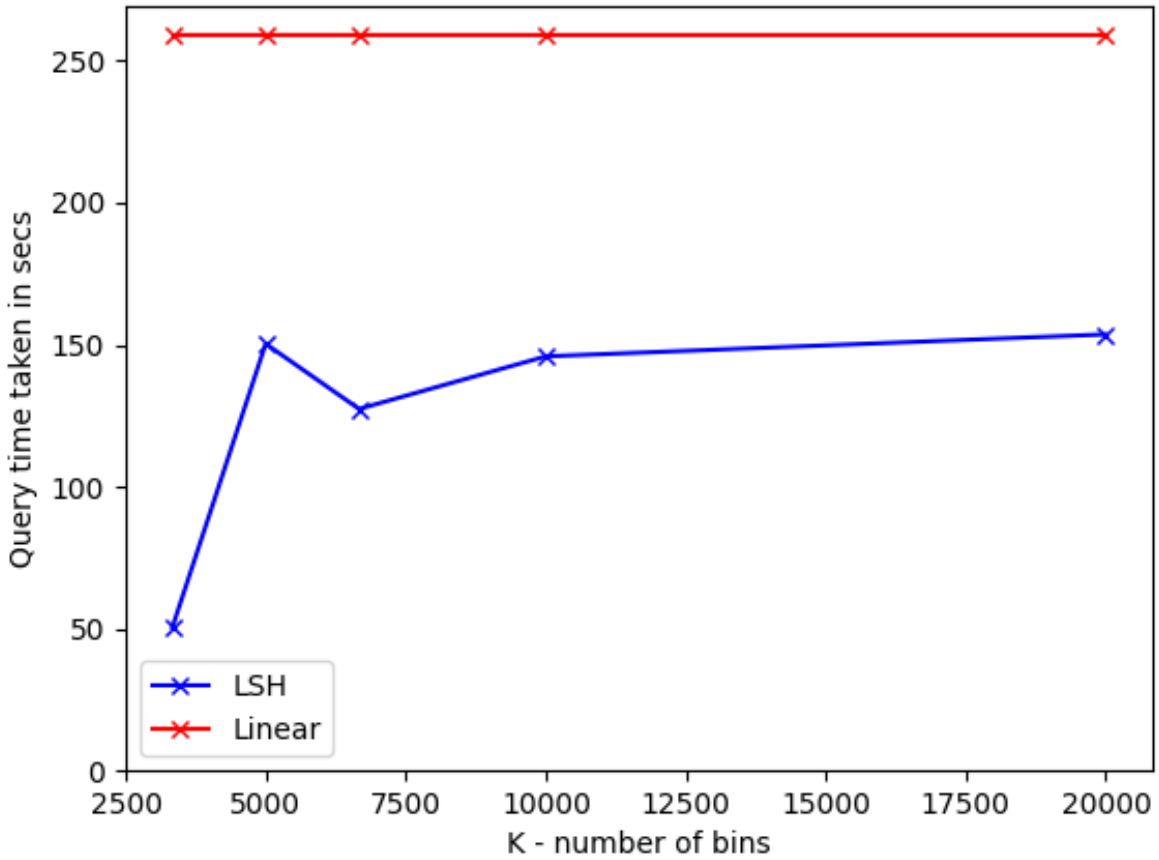

Figure: Query time vs K, L = 10

Experimental Results

Comparison

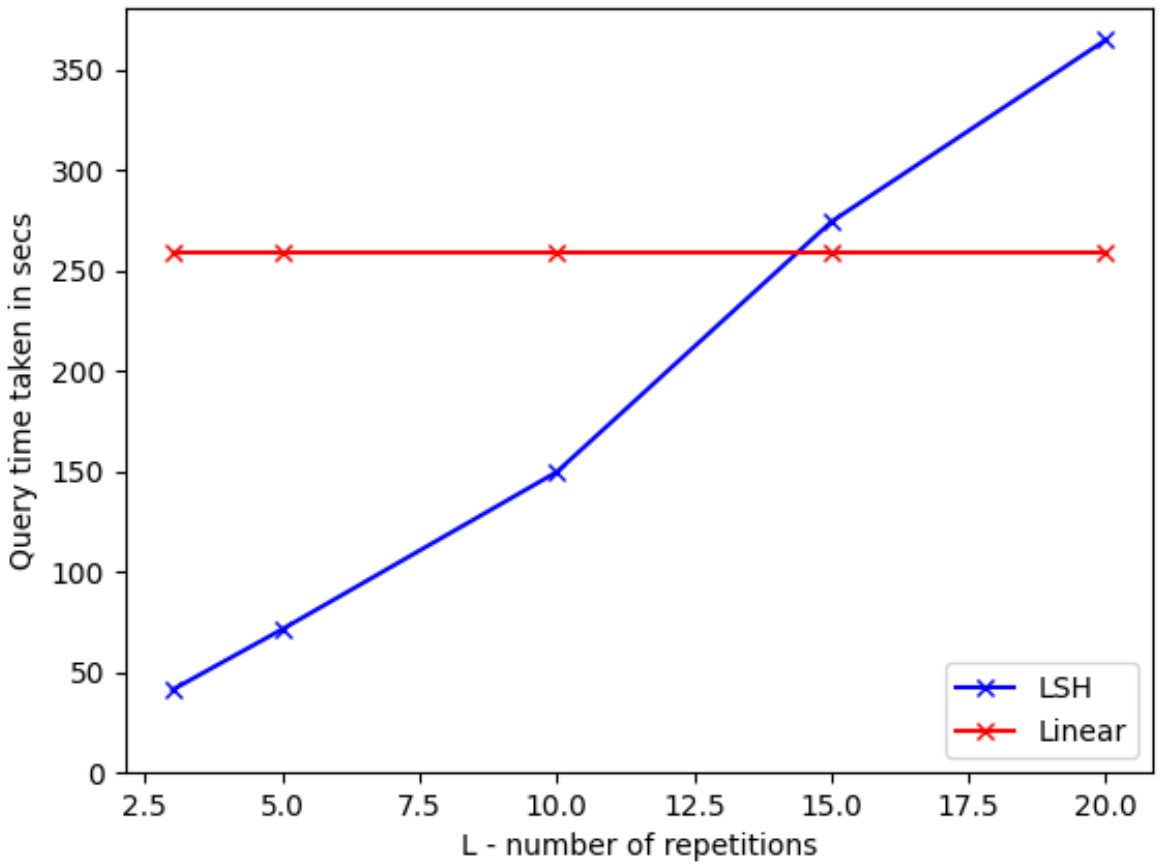

Figure: Query time vs L, K = 5000

Experimental Results

Comparison

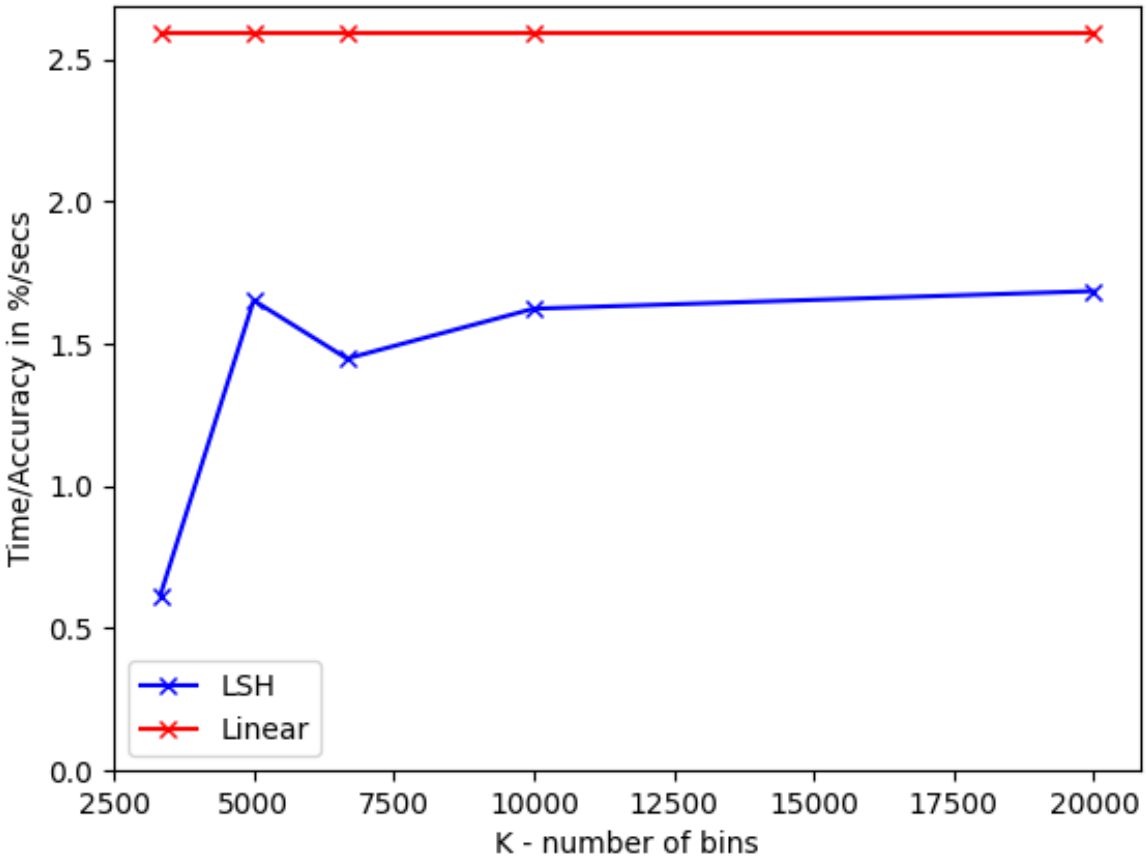

Figure: Accuracy/time taken vs K, L = 10

Experimental Results

Comparison

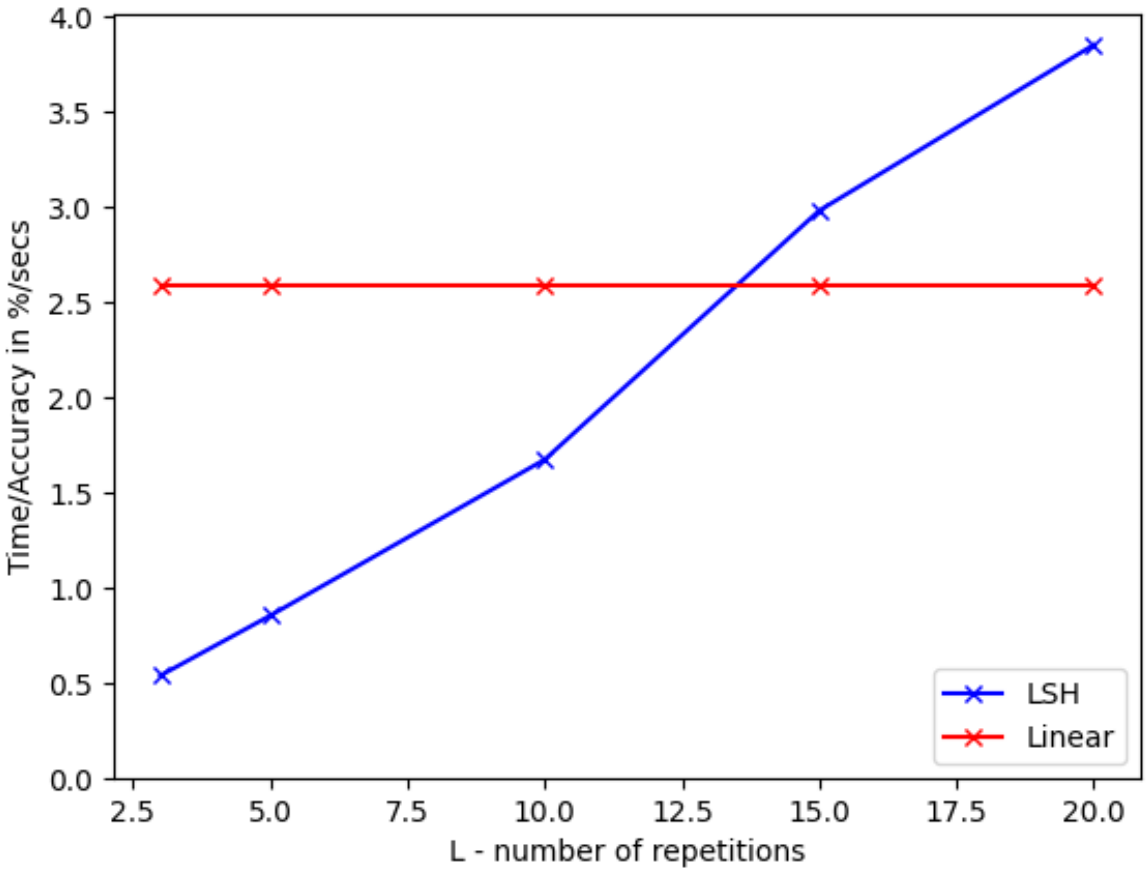

Figure: Accuracy/time taken vs L, K = 5000