,

ACCELERATED IMAGE-BASED PRODUCT RECOMMENDATION SYSTEM

Vaishnavi Channakeshava , channake@usc.edu
Sanjana Vasudeva, sanjanav@usc.edu

December 10, 2022

1 Introduction

Recommendation System is a software system widely used in E-commerce websites such as Amazon, Walmart, etc., allowing users to find suitable products based on their current choice. Instead of allowing the users to perform search after search to get their desired items, recommending such relevant items are more impressive and provide a better sense of satisfaction. Our project aims to build a basic parallelized image-based product recommendation system that accepts images as input and returns a set of similar products.

To guarantee retrieval accuracy, hundreds or thousands of feature points are extracted to represent an image or a frame [1]. Each feature point will be further described with information around it and is stored in a high-dimensional vector. The comparison between the query image vector and the vectors of the image database happens in real time. Therefore, the algorithms are not only computationally intensive but also data intensive. Image comparison between the query image and a large dataset leads to high processing speed.

A recommendation system with high performance will provide the response to the user query in a short time thus improving the user experience. A faster recommendation system will convert shoppers into customers and boosts the sales and revenue of the company. Such a recommendation system helps in maintaining the brand experience.

To achieve a faster recommendation system our project aims to accelerate the query search time. The following will be implemented in this project, namely:

- Implement serially the comparison of query image with the images in database.
- Implement a parallel version of query image comparison with the images in database to see how parallelization can speed up comparisons.
- Evaluate query search time using different similarity measures in serial and parallel.

2 Algorithm

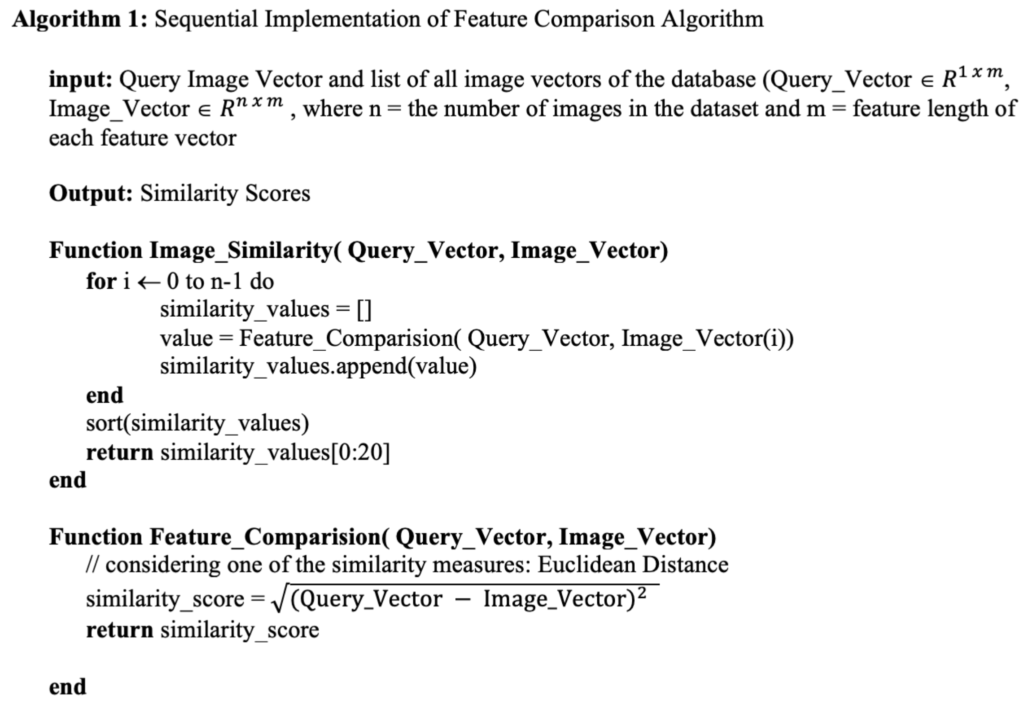
Determining the set of related products for a given query image consists of comparing the query image vector with the image vectors in the database using various similarity measures. The following similarity measures have been experimented with:

- **Euclidean Distance** : It calculates the distance between the two vectors. Euclidean Distance values are non-negative and a value of zero means the images being compared are identical and a higher value implies lesser similarity.
- **Chi-Squared** : It is a statistical method used to measure the similarity between the two images, which is based on weighted euclidean distance. A distance of zero indicates the images are similar, and dissimilarity increases with increasing distance.

- **Manhattan Distance** : It is a metric in which the distance between two vectors is the sum of the absolute differences of their Cartesian coordinates.

- **Cosine Similarity** : This metric finds the normalized dot product of the two attributes. By determining the cosine similarity, we would effectively try to find the cosine of the angle between the two objects. The cosine of 0° is 1, and it is less than 1 for any other angle.

The pseudocode for representing feature comparison using a similarity measure taking Euclidean Distance as an example is presented in Algorithm 1 shown in Figure 1.



```
Algorithm 1: Sequential Implementation of Feature Comparison Algorithm

    input: Query Image Vector and list of all image vectors of the database (Query_Vector ∈ $R^{1 x m}$,
    Image_Vector ∈ $R^{n x m}$ , where n = the number of images in the dataset and m = feature length of
    each feature vector

    Output: Similarity Scores

    Function Image_Similarity( Query_Vector, Image_Vector)
        for i ← 0 to n-1 do
                similarity_values = []
                value = Feature_Comparision( Query_Vector, Image_Vector(i))
                similarity_values.append(value)
        end
        sort(similarity_values)
        return similarity_values[0:20]
    end

    Function Feature_Comparision( Query_Vector, Image_Vector)
        // considering one of the similarity measures: Euclidean Distance
        similarity_score = $\sqrt{(\text{Query\_Vector} - \text{Image\_Vector})^2}$
        return similarity_score

    end
```

Figure 1: Serial Implementation of the Feature Comparison Algorithm Using Euclidean Distance

3 Context

Image feature vector representation : The feature vectors extracted are stored in a matrix of dimension N x feature length, where N is the number of images in the dataset and feature-length depends on the method of feature extraction namely :

- RGB3D Histogram and Hu Moments (Image Processing Technique)

- MobileNet

The project operates on the assumption that this matrix is already available and the feature comparison task needs to be performed. To obtain the same, the features have been extracted offline using MobileNet and image processing techniques such as - RGB 3D Histogram and HuMoments. The dimension of the features extracted using MobileNet is (1,1000) and the dimension of the features extracted using RGB 3D Histogram and HuMoments is (1,1741). In our project, we perform this feature extraction task in Serial, parallelize using Pthreads as well as Accelerate the baseline serial code using GPU.

4 Parallelization Strategy

As mentioned in the Algorithm section, all the computed vectors of the images in the dataset are compared with the query vector of the image provided. There is no data dependency between the vectors for comparison and this enables us to parallelize the computation.

The strategy is to parallelize feature comparison. The database consists of N images, hence the extracted features vectors are stored in a matrix of size (N x Feature Length). Our parallelizing technique partitions the matrix based on a number of feature vectors (rows). Each partition will have the feature vectors of a subset of the images and these feature vectors will be compared with the query vector. Each partition will be assigned to different threads for parallel calculation. The technique is shown in Figure 2.

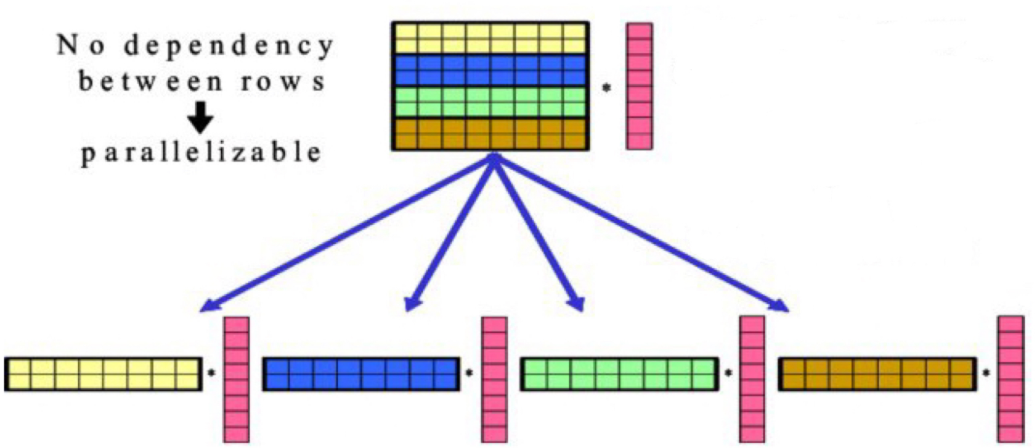


Figure 2: Image Feature Matrix multiplication with Query Feature Vector

5 Hypothesis

Image comparison consists of comparing the query image vector with the image vectors from the database. The image comparison algorithm can be parallelized since there is no data dependency between the feature vectors of the images in the database. Hence, we should be able to speedup the computation by data parallelism instead of task parallelism.

Thus, the hypothesis is:

- Data parallelism is more suitable since the same task of comparison is being performed on different image vectors.
- The query search time depends on the sparsity of the image feature vectors and is not highly affected by the feature vector length.
- The baseline Image Feature comparison algorithm can be further accelerated by GPU.

6 Experimental Setup

6.1 Data preparation

In this project, a custom dataset has been created that consists of 35-40 products like accessories, clothes, electronics, furniture, etc. summing up to 53810 images. A sample of the dataset consisting of the category and its count is tabulated below:

Category	Number of Images
Bag pack	900
Watch	800
Cycle	705
Jacket	500
Sofa	1500

The dataset provides us with input feature matrix consisting of feature vectors extracted feature vectors using the two methods mentioned previously in Context section are stored in two .npy files respectively and the link for the files can be found here. Further, these stored features are used for Feature comparison.

6.2 Serial implementation (baseline)

The feature comparison in serial has been implemented in Python program to obtain the baseline values for further performance evaluation. The Python program runs on an Apple M1 CPU. It has 8 cores with 4 cores running at 3.2 GHz and 4 cores running at 2.064 GHz.

6.3 Pthread acceleration

In this project, Pthread is utilized to examine the speedup of parallelism. As mentioned in Parallelization Strategy section, each thread will be assigned a subset of image feature vectors, and will be responsible to perform feature comparison for those images with the query image. The performance has been evaluated from 2 threads to 64 threads.

6.4 GPU Acceleration

For GPU platform, Tesla T4 provided by Google Colab is utilized. The size of the global memory is 32GB RAM. The number of CUDA cores in this GPU is 2560. The baseline algorithm for different similarity measures has been accelerated in GPU using Cupy. CuPy is an open-source array library for GPU-accelerated computing with Python.

7 Result and Analysis

Similarity Measures	Serial Time (sec)	Parallel Time (sec)					
		2	4	8	16	32	64
Euclidean	17.35	15.52	9.93	12.13	9.43	6.43	6.62
Chi-Squared	23.26	5.76	3.66	3.44	3.52	3.71	4.14
Manhattan	11.52	11.35	7.41	7.35	3.19	3.08	3.97
Cosine Similarity	11.32	7.57	6.76	6.71	5.10	4.90	4.02

Table 1: Feature Vectors from RGB3D Technique

Similarity Measures	Serial Time (sec)	Parallel Time (sec)					
		2	4	8	16	32	64
Euclidean	20.78	17.83	13.71	13.05	9.62	6.86	5.92
Chi-Squared	18.57	13.10	12.87	9.23	7.88	7.85	8.49
Manhattan	22.77	15.99	14.15	8.66	7.44	6.87	6.55
Cosine Similarity	37.14	17.50	14.42	10.15	7.20	5.60	4.27

Table 2: Feature Vectors from MobileNet

Table 1 and Table 2: Table 1 and Table 2 show the serial and parallel time achieved for the feature comparison between the query vector and database image feature vectors obtained using the RGB3D Histogram and Hu Moments method and MobileNet respectively. The baseline serial code feature comparison is parallelized using Pthreads. The parallel time is recorded for different Pthread Values ranging from 2 to 64.

Similarity Measures	RGB3D Technique		MobileNet Features	
	Serial Time (sec)	Parallel Time (sec)	Serial Time (sec)	Parallel Time (sec)
Euclidean	17.35	0.81	20.78	0.45
Chi-Squared	23.26	0.40	18.57	0.35
Manhattan	11.52	0.59	22.77	0.33
Cosine Similarity	11.32	0.32	37.14	1.71

Table 3: GPU Acceleration of the Image Similarity Measures

Table 3: Table 3 shows the parallel time achieved in accelerating the baseline model using GPU for RGB3D Histogram and Hu Moments and MobileNet feature extraction methods.

7.1 Acceleration

The parallel and serial execution time for every similarity measure is compared. Manhattan is the simplest similarity measure involving just subtraction so the execution time is very fast. As the similarity measure calculation becomes complex, the query search time correspondingly increases. As expected, for all the similarity measures, the serial version takes the longest execution time. There is a significant reduction in the query search time by applying the Pthread model and the GPU model as shown in the graphs plotted for RGB3D Features in Figure 3 and MobileNet Features in Figure 4.

From Figure 3 and Figure 4, we can observe that the query search time for MobileNet features is higher than the query search time for RGB3D Histogram and Hu Moments features, despite the length of the features extracted by the latter being almost twice length of the features extracted by the former. This is because the RGB3D Histogram and Hu Moments features vector matrix is a sparse matrix. Hence, performing computations on it takes much lesser time. Therefore, the latency depends on the sparsity of the image feature vector matrix rather than its dimension.

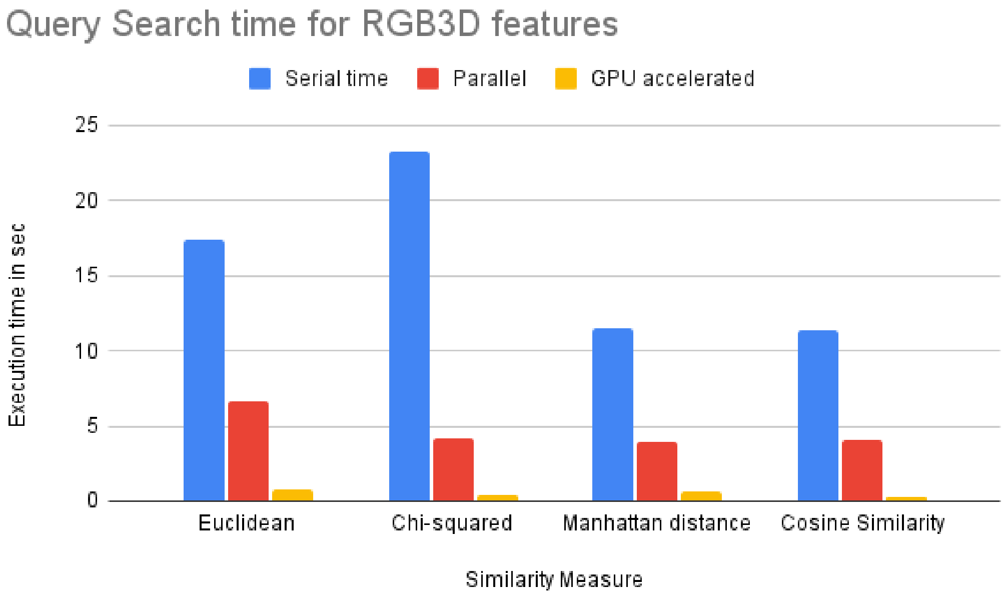


Figure 3: Query Search Time For RGBD3D Features

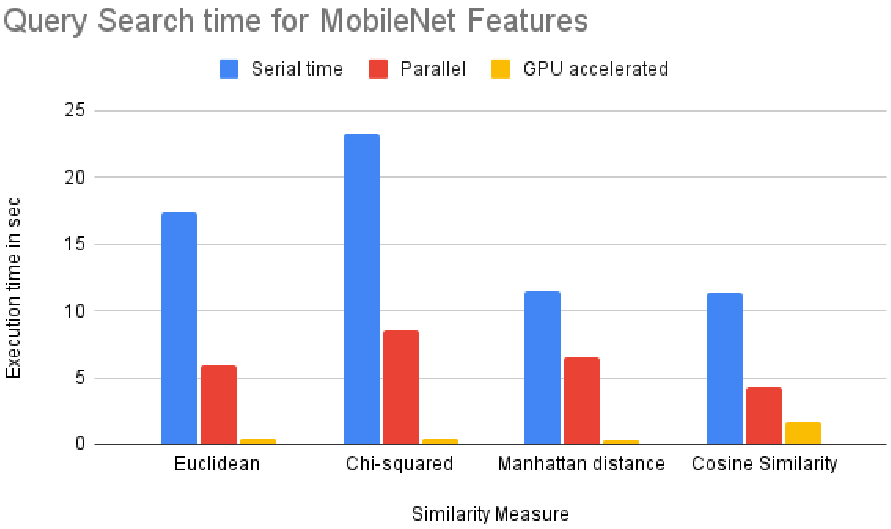


Figure 4: Query Search Time For Mobilenet Features

7.2 Speedup

The serial algorithm as shown in Figure 1 is used as the baseline model. The speedup has been evaluated on all the similarity measures. Figure 5 and Figure 6 show the speedup achieved using the Pthread model with the number of threads equal to 64 and the GPU-accelerated model. For the Pthread model, a speedup of about 2x to 3x has been achieved for all the similarity measures for both feature extraction methods. Similarly for GPU accelerated a speedup of 20x to 60x has been achieved.

$$Speedup = \frac{SerialExecutionTime(Ts)}{ParallelExecutionTime(Tp)}$$

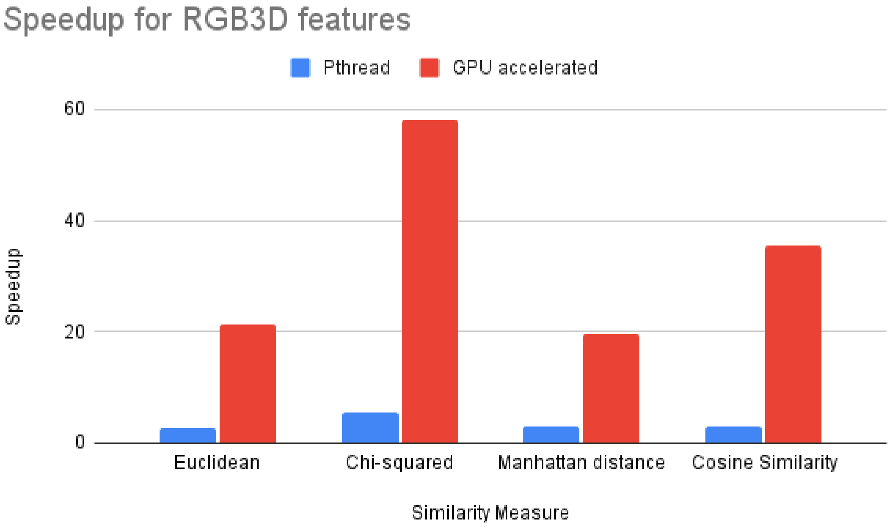


Figure 5: Speedup For RGBD3D Features

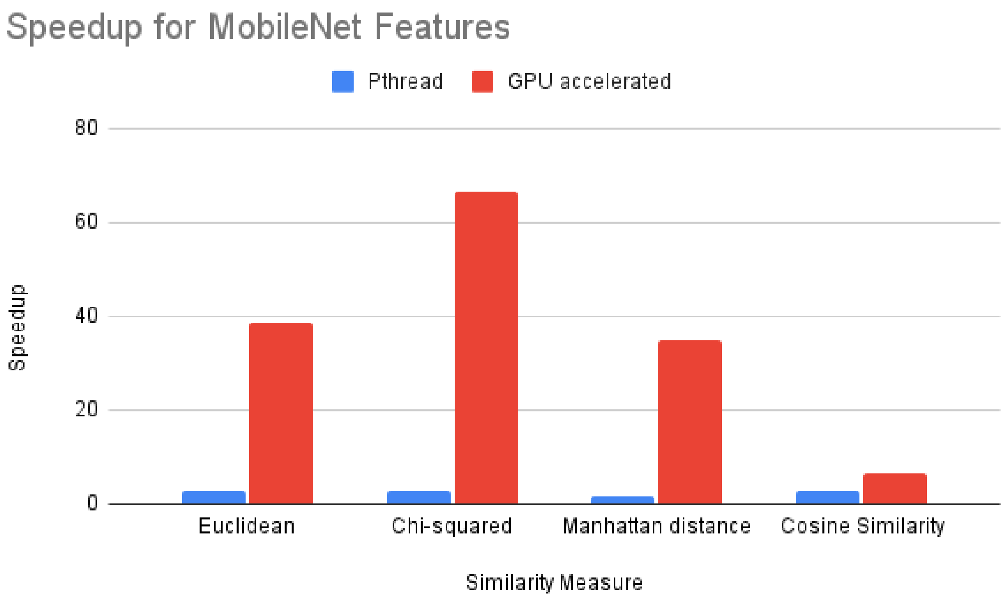


Figure 6: Speedup For Mobilenet Features

7.3 Acceleration using different number of threads

The baseline Serial Algorithm is parallelized using Pthreads for both the RGB3D and MobileNet methods. From Figure 7 and Figure 8, we can infer that as the number of threads increases from 2 to 64, the query search time gradually decreases and saturates for thread no =64.

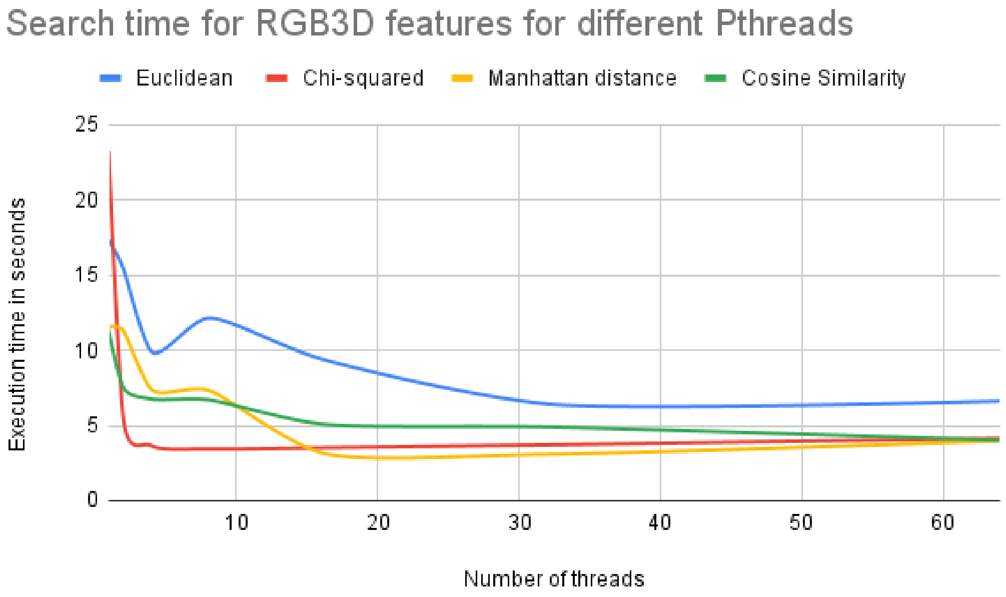


Figure 7: Parallelizing Query Search time for Different Pthread Values (2,4,8,16,32,64) for RGB3D Features

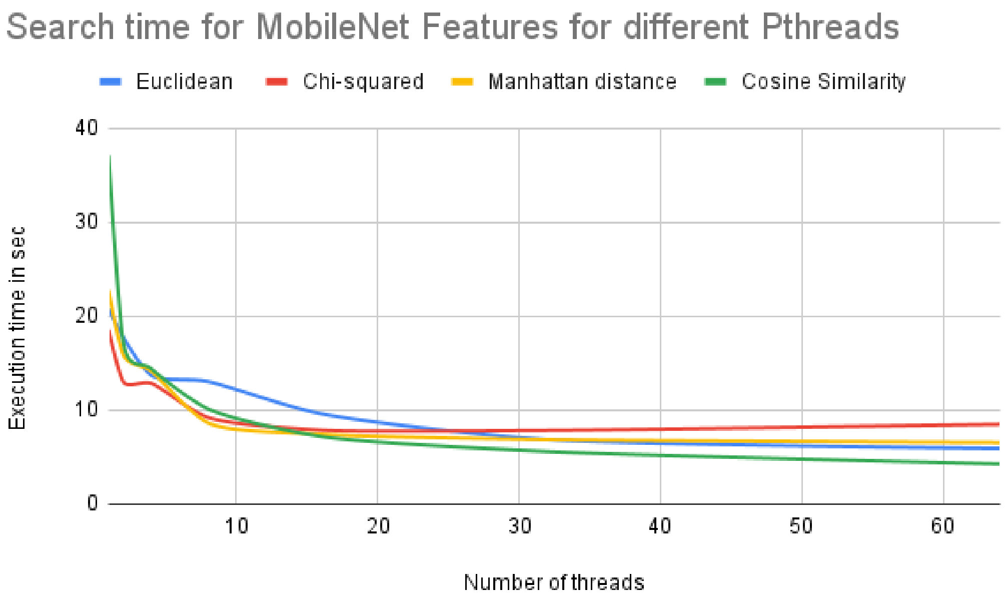


Figure 8: Parallelizing Query Search time for Different Pthread Values (2,4,8,16,32,64) for MobileNet Features

The performance of the Pthread model is limited i.e. a higher speedup cannot be achieved by increasing the number of threads after a certain point. The limitation caused is due to the following two reasons :

- Amdahl's Law: Amdahl's Law observes that if S is the serializable portion of a computation, then the maximum performance improvement is limited to a factor of 1/S.

$$Speedup = \frac{1}{(1-P)+(P/f)}$$

- Memory Bandwidth: Increasing the number of threads additionally for parallelization will lead to an increase in the memory communication cost. This increase in communication cost cannot be supported by a limited memory bandwidth above a certain range of parallelization. As a result, the memory bandwidth will be the limiting factor in our speedup.
- Overhead: With the increase in the number of threads, the overhead to create and gather the results will also increase, resulting in a decrease in efficiency.

7.4 Output of the Image Recommendation System

The image on the top-left corner is the input query image in Figure 9 and Figure 10. In Figure 9, the features have been extracted using RGB3D and Hu Moments and chi-squared distance has been chosen as the similarity measure. In Figure 10, the features have been extracted using MobileNet and chi-squared distance has been chosen as the similarity measure.

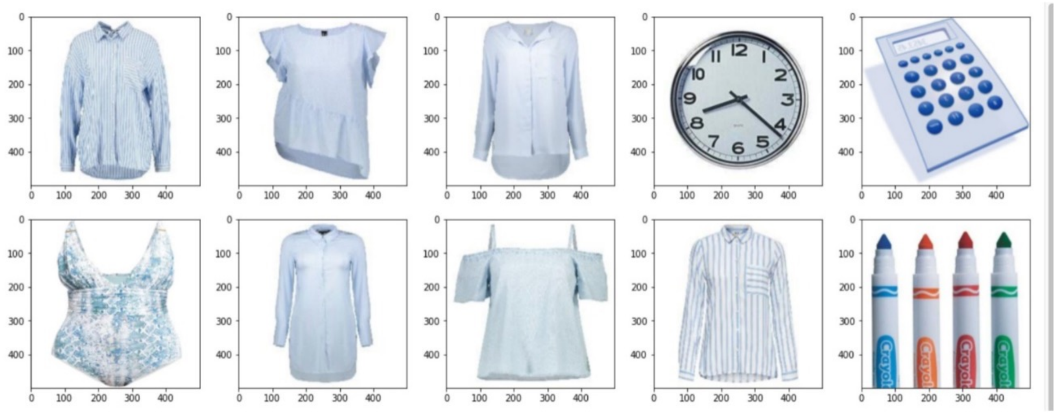


Figure 9: Top 10 products recommended by features extracted using RGB3D and Hu Moments and feature comparison using chi-squared similarity measure

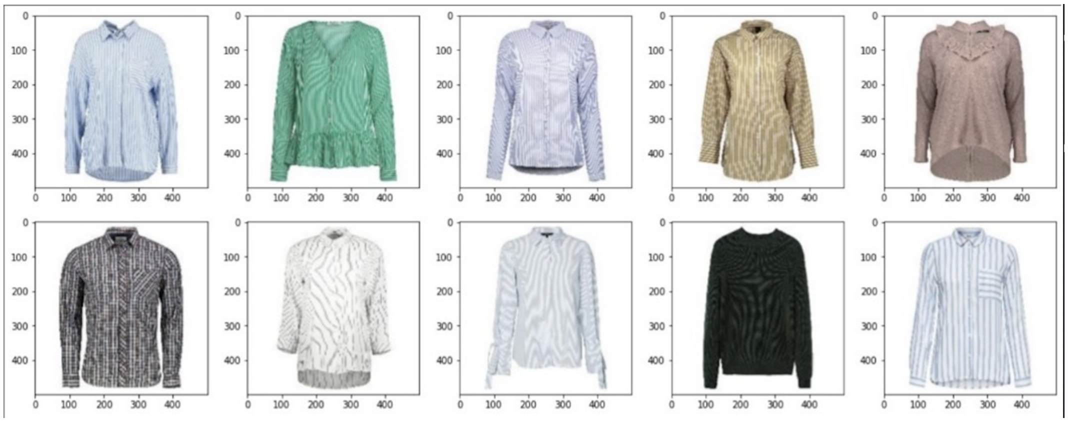


Figure 10: Top 10 products recommended by features extracted using MobileNet and feature comparison using chi-squared similarity measure

8 Conclusion

In this project, an accelerated image-based recommendation system has been implemented to achieve reduced latency and speedup. The serial version of the algorithm for query image vector and image feature vector comparison has been implemented using Python programming which serves as a baseline for this project. For parallel models, the Pthread version of the baseline model is implemented and the speedup is evaluated using the different number of threads. Notice that while the speedup goes higher as more threads are used, the efficiency drops due to memory bandwidth and the serial parts of the code that cannot be parallelized. For GPU execution, we can achieve a 20x to 60x speedup compared to the serial baseline model.

References

[1] Lu, Yunping, et al. ”Parallelizing image feature extraction algorithms on multi-core platforms.” Journal of Parallel and Distributed Computing 92 (2016): 1-14.

[2] Chen, Luyang, Fan Yang, and Heqing Yang. ”Image-based product recommendation system with convolutional neural networks.” (2017).

[3] Ragatha, Divya Venkata, and Divakar Yadav. ”Image querybased search engine using image content retrieval.” 2012 UKSim 14th International Conference on Computer Modeling and Simulation. IEEE, 2012.

[4] Sample dataset: Amazon Berkeley Objects Dataset.

[5] Siamese Network Keras for Image and Text similarity.

[6] Different Distance Metrics.

[7] Using GPU with Python CuPy