CS GY 6513
BIG DATA

Efficient Repository for fine-tuning Self-driving vehicles using Big Data

TEAM MEMBERS

1. Anudeep Tubati
2. Geetika Bandlamudi
3. Ramanarayanan Sankaranarayanan
4. Sakthi Uma Maheswari
5. Yamini L Lakshmi narasimhan

Point of this project...

1. Use Case - Data inflow rate + Storage size (Motivation slide)
2. Python drawbacks and limitations (KNN)
3. Hadoop Algorithm metrics + Limitations
4. ~~Pyspark Algorithm (Template matching + Hog Filter) + Limitations~~
5. ~~Pyspark + Resnet50 + LSH~~
6. Architecture - bird’s eye view
7. Architecture - component view - data flow
8. Optimizations (next slide)
9. Limitations
10. Future Work

Pending Work

1. What optimization can be done in Big Data?
    a. New Images date wise prediction - caching
    b. Freeze the last layer
2. Quantitative Analysis Metrics
3. Graphs/Charts
    a. Time improvement across algorithms
    b. Time improvement across datasets
4. Dataset
5. Storage

Motivation

- One car - One day 4GB Data points - 160,000 cars for now with Beta FSD
- We can only imagine how much data flows into their system on a daily basis.
- Now adding all this to your training data will take more than 2 years to train for the smallest model

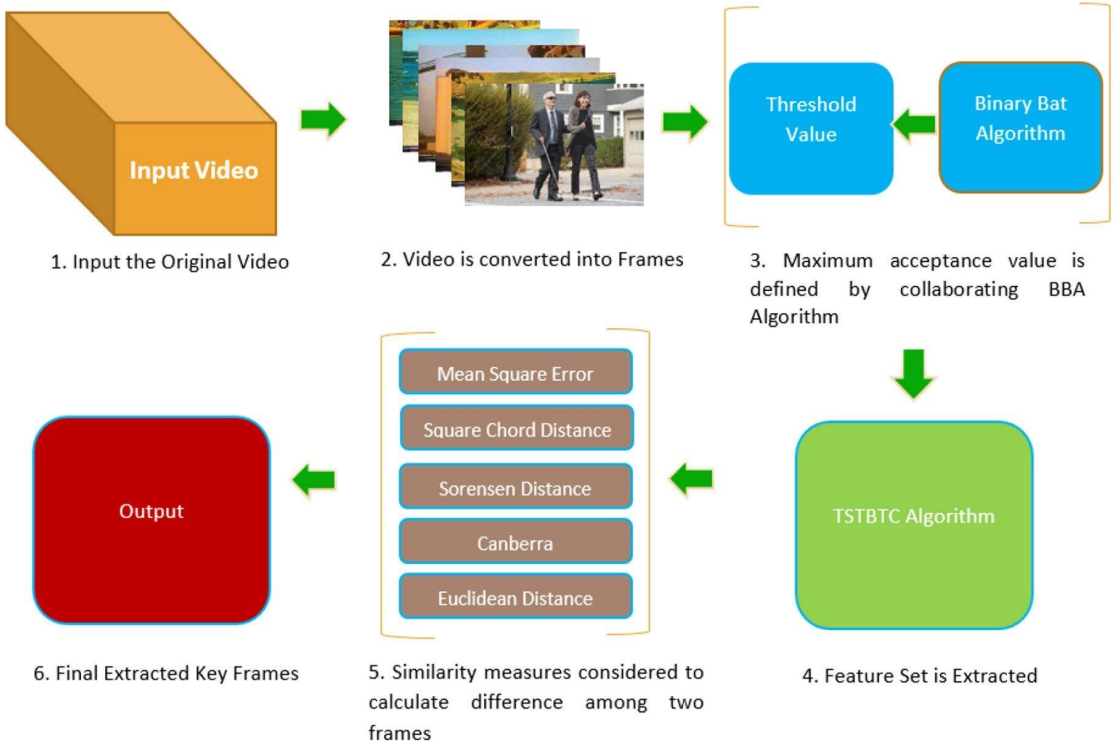
https://www.youtube.com/watch?v=y57wwucbXR8

Dataset and Scale

https://www.kaggle.com/datasets/andrewmvd/road-sign-detection ,
https://public.roboflow.com/object-detection/self-driving-car

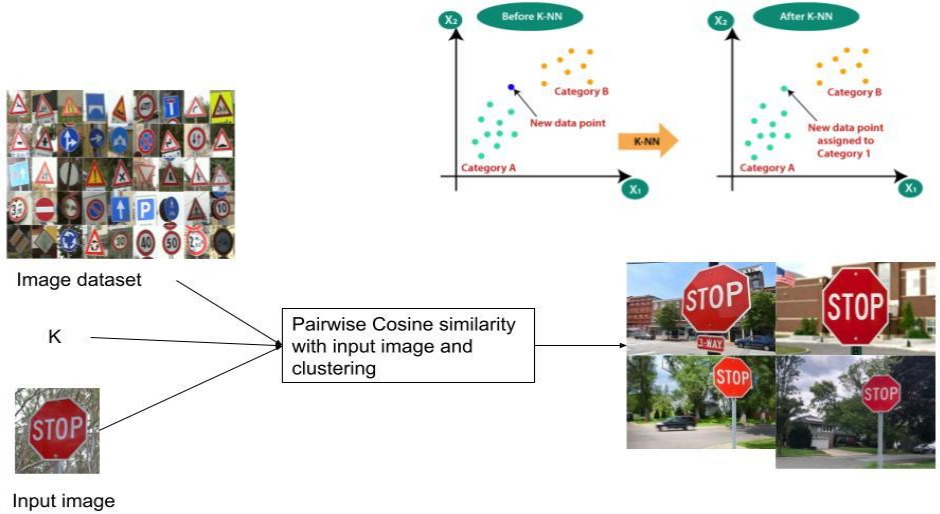
- Data reaches upto 4TB per day from self-driving cars (source)
- Autonomous test vehicles generate 5TB - 20TB of data *per day per vehicle* (source)
- Need pre-processing of video data which is an expensive operation

Video Key-Frame extraction algorithm



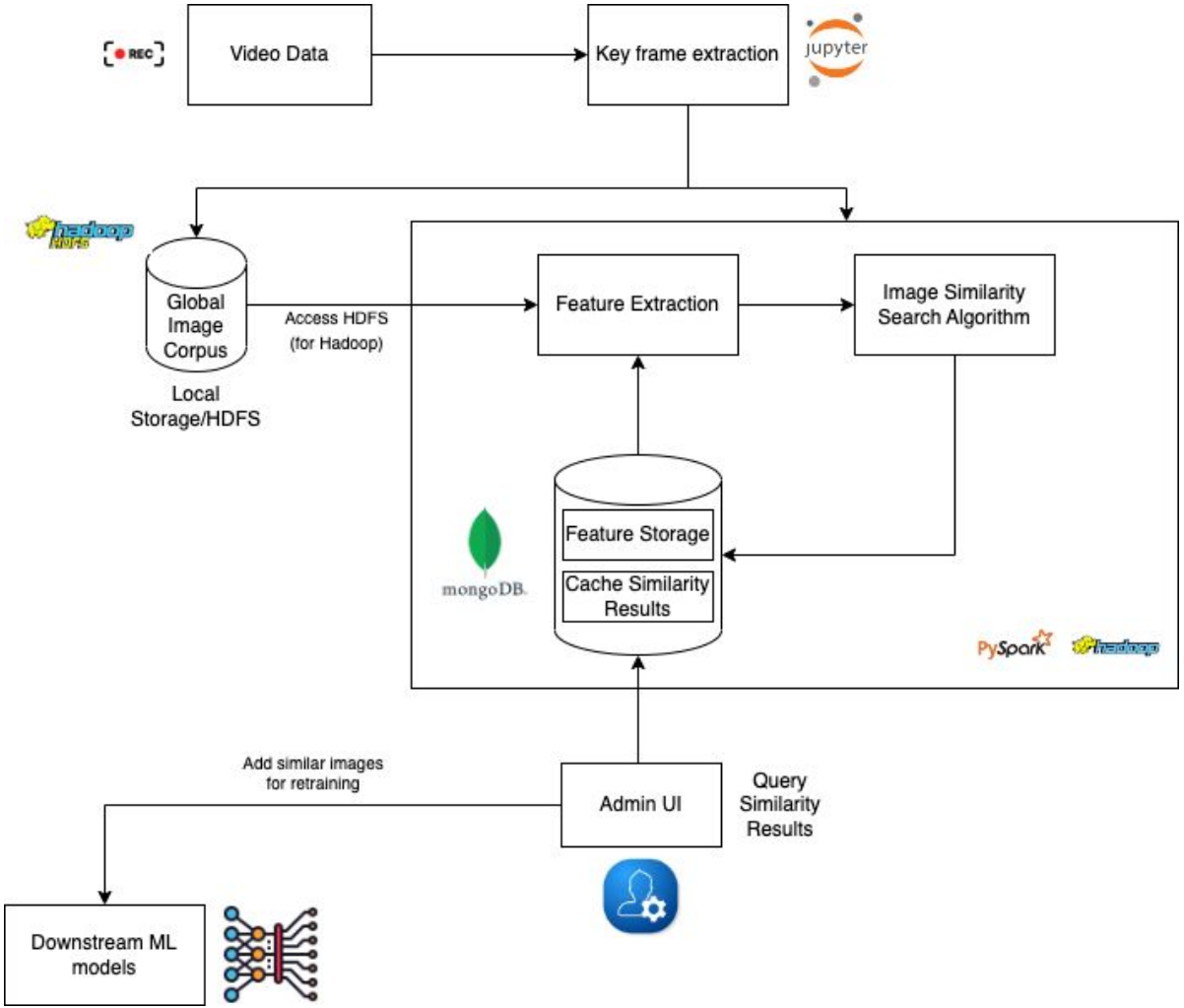
Python based approach - Observations & limitations

Time complexity :
O(Number of images * Number of features)



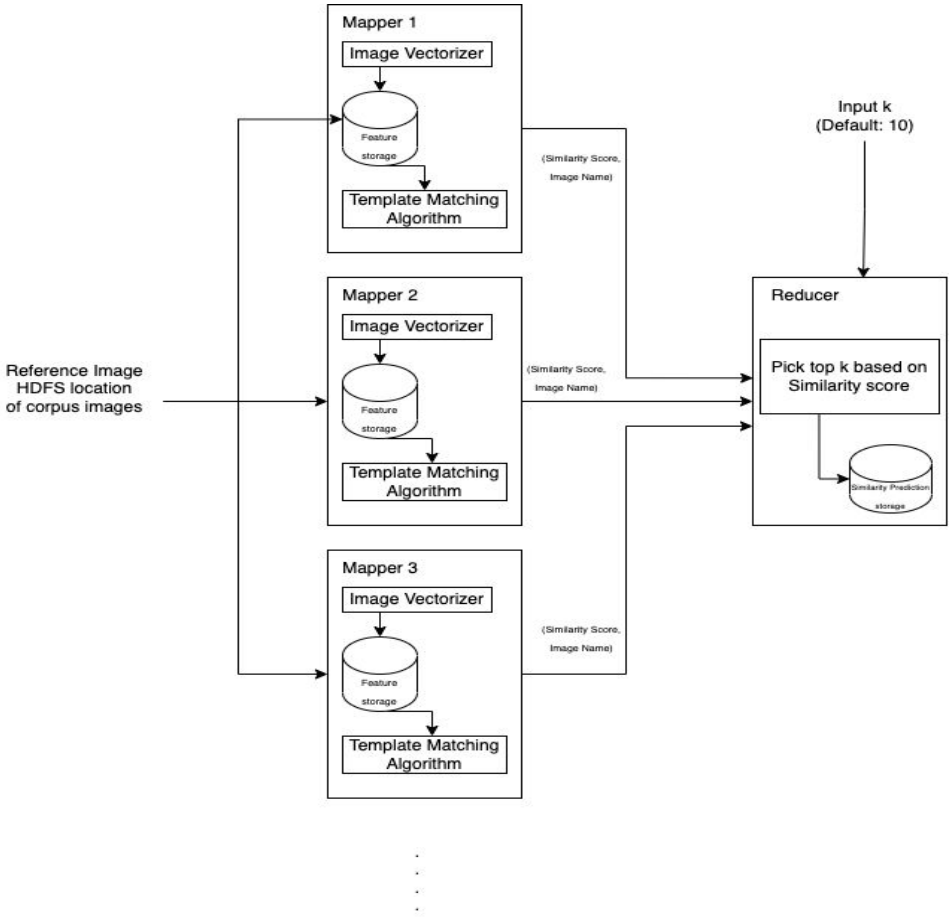
- Time taken for 1.1GB = approx. 2 hours

Bird's Eye View



Mapper and Reducer in Hadoop

- Parallelize Feature Extraction in Mapper
- Store image features in MongoDB
- Performance enhancement due to parallel processing
- Limitations to employ advanced ML algorithms



Architecture Diagram

Hadoop + SIFT - Results

Input

Output

Pyspark + Different Algorithms

- Architecture
- Limitations
- Performance Discussion

Pyspark + HOG Filter

New Streaming Data

amazon S3

Corpus

HOG Filter Descriptors

jupyter

mongoDB

Input

HOG Filter Descriptors

Cosine Similarity

Output

Spark ML

Time taken for 1.1GB = approx. 25 mins
Features are too sparse (200,000) and hence slow computation

*Saving in MongoDB helps reduce time taken for model to run on the entire corpus again and again

Pyspark + HOG Filter - Results

Input

-

Pyspark + Resnet50 + LSH

New Streaming Data

Input

Resnet 50 prediction - Prediction Vectors

LSH

Output

amazon S3

Corpus

Resnet 50 prediction - Prediction Vectors

PyTorch

mongoDB

Spark ML

- Resnet50 - 1000 classes prediction vectors
- Time taken for 1.1GB (https://public.roboflow.com/object-detection/self-driving-car/3) - (approx 20 min) - approximate template matching

*Saving in MongoDB helps reduce time taken for model to run on the entire corpus again and again

Pyspark + Resnet50 + LSH

- Resnet50 - 1000 classes prediction vectors
- Time taken for 1.1GB (https://public.roboflow.com/object-detection/self-driving-car/3) - (approx 20 min) - approximate template matching

Corpus

New Streaming Data

Resnet 50 prediction - Prediction Vectors

PyTorch

Input - Test Data

jupyter

mongoDB

LSH

Spark ML

Output

*Saving in MongoDB helps reduce time taken for model to run on the entire corpus again and again

Pyspark + Resnet50 + LSH - Results

Input

Output

Why MongoDB?

- Document Oriented to store feature vectors of images

- For storing similar images for each input to avoid recomputation

- Why not SQL?

Documents
image_features_...

image_features_db.features

890 DOCUMENTS 1 INDEXES

Documents | Aggregations | Schema | Explain Plan | Indexes | Validation

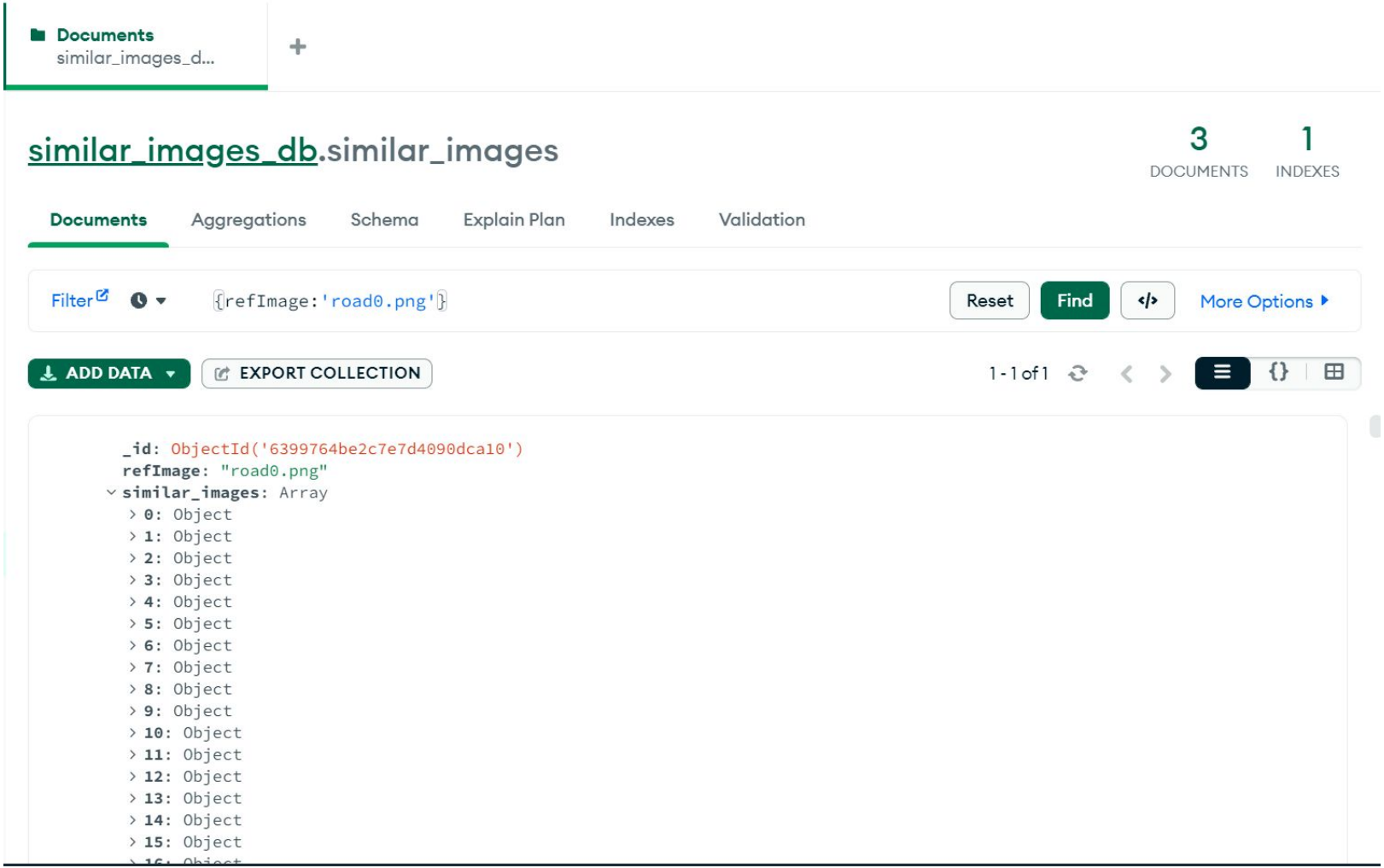
Filter | Type a query: { field: 'value' } | Reset | Find | </> | More Options

ADD DATA | EXPORT COLLECTION | 1 - 20 of 890

```
_id: ObjectId('63976b4298d355d2d296f8fb')
name: "images_4.jpeg"
> keypoint: Array
description: BinData(128, 'gAJjbnVtcHkuY29yZS5tdWx0aWFycmF5Cl9yZWNvbnN0cnVjdApxAGNudW1weQpuZGFycmF5CnEBSwCFcQJjX2NvZGVjcwp…
```

```
_id: ObjectId('63976b4298d355d2d296f8fc')
name: "images.jpeg"
> keypoint: Array
description: BinData(128, 'gAJjbnVtcHkuY29yZS5tdWx0aWFycmF5Cl9yZWNvbnN0cnVjdApxAGNudW1weQpuZGFycmF5CnEBSwCFcQJjX2NvZGVjcwp…
```

```
_id: ObjectId('63976b4298d355d2d296f8fd')
name: "images_2.jpeg"
> keypoint: Array
description: BinData(128, 'gAJjbnVtcHkuY29yZS5tdWx0aWFycmF5Cl9yZWNvbnN0cnVjdApxAGNudW1weQpuZGFycmF5CnEBSwCFcQJjX2NvZGVjcwp…
```


Documents
similar_images_d...
similar_images_db.similar_images
3
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Explain Plan
Indexes
Validation
Filter
{refImage:'road0.png'}
Reset
Find
More Options
ADD DATA
EXPORT COLLECTION
1 - 1 of 1
_id: ObjectId('6399764be2c7e7d4090dca10')
refImage: "road0.png"
similar_images: Array
0: Object
1: Object
2: Object
3: Object
4: Object
5: Object
6: Object
7: Object
8: Object
9: Object
10: Object
11: Object
12: Object
13: Object
14: Object
15: Object

Future Work

- Process the key frame extraction using spark streaming module and Kafka.

- Object detection (another big data problem by itself) during preprocessing using spark streaming reduces noisy unrelated frames from entering the system. (source)

The journey and lessons learnt along the way...

- Native ***python*** scripts for huge streaming data - compute intensive
- Issues with adding additional packages in ***HDFS*** environment
- ***Hadoop*** interface is not good for visualizing intermediate outputs and debugging algorithmic errors
- Template matching, HOG filter algorithms in ***Pyspark*** achieve faster results than hadoop
- Approximate image search (***LSH***) reduces time complexity
- Eliminating recomputation of feature extraction - ***MongoDB***
- ***Dask*** - A brilliant framework with a very similar usage to Python

Thank you!

We are sincerely grateful to Prof. Rodriguez for taking this course. We are also thankful to the TAs!

References

https://towardsdatascience.com/deep-learning-with-apache-spark-part-1-6d397c16abd - Spark Deep Learning packages

https://365datascience.com/trending/techniques-for-processing-traditional-and-big-data/ - Python limitations

https://docs.dask.org/en/stable/ - Dask