Intrinsic Detective System

Phase IV – Report

CS 5543 Real-Time Big Data Analytics

by

Group 6

Sri Harsha Chennavajjala (4)

Priyadarsini Nidadavolu (16)

Tej Kumar Yentrapragada (27)

Chaitanya Sai Manne (14)

Table of Contents

1. Project Objectives 4
   1.1 Significance 4
   1.2 Features 4
   1.3 Project Uniqueness 4
2. Approach 5
   2.1 Data Sources 5
   2.2 Analytic Tools 5
   2.3 Analytical Tasks 5
   2.4 Expected Inputs/Outputs 5
   2.5 Algorithms 6
3. Related Work 6
   3.1 Open Source Projects 6
4. Application Specification 6
   4.1 System Specification 6
      4.1.1 Software Architecture 6
   4.2 Design of Big Data Analytics Server 8
      4.2.1 Parallelism/Distribution: Task/Data 8
      4.2.2 Features, workflow, technologies 9
   4.3 Design of Mobile Client (smartphone/web) 11
      4.3.1 Features, GUI, technologies 11
   4.4 Existing Applications or Services Used 12
5. Implementation 12
   5.1 Implementation of server using Spark and Storm and Kafka 12
   5.2 Implementation of any machine learning library or multimedia library. 13
   5.3 Implementation of mobile clients 13
6. Project Management 14
   6.1.1 Plan & Project Timelines 14
   6.1.2 Project Members 14
   6.1.3 Project GitHub Repository Link 14
   6.2 Implementation Status Report 14
      6.2.1 Phase 1: 14
      6.2.2 Phase 2: 21

6.2.3 Phase 3: 24
6.2.4 Phase 4: 27
6.3 Work completed 31
6.3.1 Phase 1: 31
6.3.2 Phase 2: 32
6.3.3 Phase 3: 33
6.3.4 Phase 4: 33
6.4 Work to be completed 34
6.5 Git Issues/Concerns 34
7. Future Scope 35
8. Bibliography 35

1. Project Objectives

1.1 Significance

In recent times, the use of surveillance cameras by the organizations as well as by the individuals has been increased in order to increase the security to their properties. But most of these surveillance systems do not include the automatic decision making capabilities (machine learning) such as alerting the owners about the possible theft, suggesting items that can be bought by the customers etc. Our project minimizes the long tedious manual work of identifying the suspects in the surveillance video by recognizing the persons in the video. If the system finds a possible threat, it will immediately alert the security authorities. There by we can reduce the crimes in the organization.

1.2 Features

- Database of faces: To develop an application that automates the process of extraction of human faces from a stream of video and stores it in a database.
- Face Recognition: To develop the functionality that compares the human faces from a video stream and recognizes the persons.
- Notification: To develop the functionality where the application alerts the security personnel if it finds a suspect in the vicinity.
- Reports: The application logs each and every instance of security threat and generates reports like the time of the day at which the possibility of robbery is high, the suspects history of robberies etc.

1.3 Project Uniqueness

The Unique Selling Point of the project is that the project not only records the activities, it can also identify the persons in the video by comparing them with the local database and takes appropriate decisions. This process reduces the man power required and also makes it easy for the security personnel to take actions immediately before the situation goes out of hands.

2. Approach

2.1 Data Sources

Data can be streamed from any video capturing device like webcams, handy cams, mobiles, social networking video sources etc.

2.2 Analytic Tools

- Spark
- Storm
- Kafka

2.3 Analytical Tasks

- Spark will be used to generate the training dataset. The machine will be trained to identify about the possible image or the scene using Spark's MLLib.
- On the other side, Storm will have the testing data set with which the analysis must be done.
- Kafka acts as a producer/consumer, which collects the streaming data from the video source and sends the features that were extracted out of the recorded video to the Spark and Storm.
- MongoDB is a NoSQL database, which helps to store the data in JSON format. Hence the model that is generated by the Spark will be conveyed to Storm server through MongoDB. Also, the results from the storm will be sent back to DB, so as to alert the mobile device.

2.4 Expected Inputs/Outputs

- **Input:** A continuous stream of video possibly with the human faces is expected as an input to our system.
- **Output:** Person recognition and an alarm notification to the mobile device would be the possible output.

2.5 Algorithms

Machine Learning algorithms for Image Classification:

- Decision Tree
- Random Forest

We have tested with both the algorithms and found that Decision Tree gave us the accuracy of about 74%. Hence we are moving forward with Decision Tree Classification.

3. Related Work

3.1 Open Source Projects

- **Face recognition using eigenfaces:** This application uses the eigenvectors to map a face to the 'face space'. Later it compares the obtained faces with these eigenfaces.
- **Face recognition using Laplacianfaces:** This application works based on appearance-based face recognition method called the Laplacianface approach. This can be achieved by using locality preserving projections.

4. Application Specification

4.1 System Specification

The system that we develop will have the capability of handling large set of data and also it can produce near real-time analytical results. This can be achieved by using distributed and parallel data processing tools such as Apache Spark, Apache Storm and Apache Kafka.

4.1.1 Software Architecture

The below pictorial representation of architecture shows that there will be an Input video sourcing device which could send the stream of features (Video features) to Kafka Message Broker. There can be two kinds of video streaming features, that will be given to the Kafka. The training data set which is to train the system about the person we would like to recognize and the testing data set in which we need to identify the person. The training data set that is sent to Kafka will be forwarded to Spark, where it uses its MLLib (Machine

Learning Library) to train the model about the person which it needs to identify. On the other side, the features of the live video from the public areas will be streamed to Kafka, which is then sent to Storm. Storm communicates with the Spark Server through MongoDB and get an idea about the training data set with which it identifies the person in the video frame. When a match is found, it will store the results to database(MongoDB) and send an alert to the mobile device.

Spark receives the input training video features from the input source via Kafka. Upon receiving the video frames, performs classification based on the received data and builds a Decision Tree Model. This model will be stored in the MongoDB. On the other side Storm upon getting the input data, it tries to get the model built by Spark. This happens through MongoDB REST service calls. Using this model, Storm identifies the person in the stream of video and store the final results back to MongoDB. Using the REST Service calls to the mobile application, a notification will be sent to the user when there is a new trigger to MongoDB from Storm. Apart from these feature, we would also like to find out the scene of the video, where it is happening based on the video that's been processed and also to identify the expression of the person whom we identified so as to make a decision on his upcoming activities. For ex: if a person is aggressive, we can predict that he is up to something, else if his expression is showing as tensed, he might have got into some trouble which might involve other culprits.

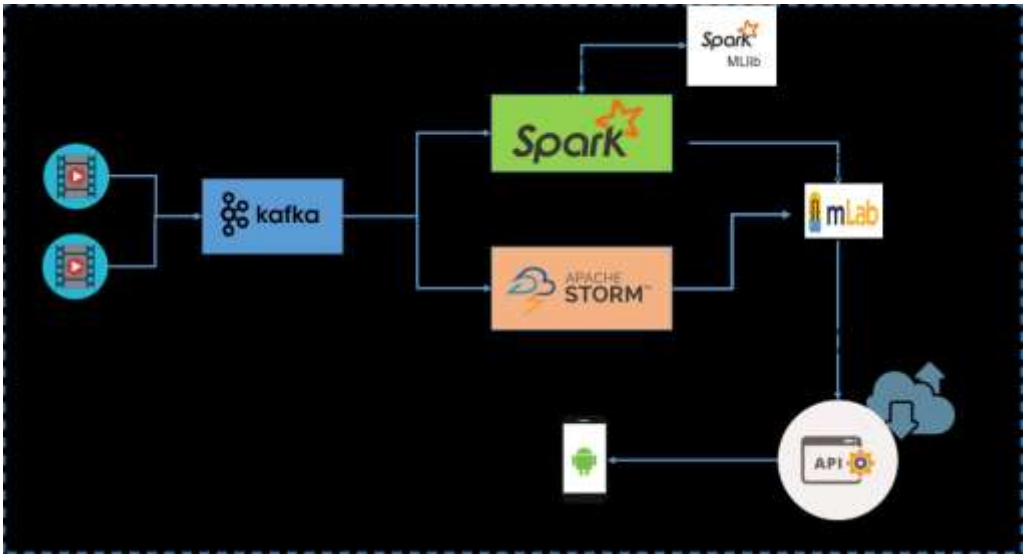


Fig. Software Architecture Diagram

4.2 Design of Big Data Analytics Server

Basically, our analytics server has been designed to detect the person, the situation of the scene and his expressions. This design is achieved by making use of the existing servers namely Spark, Storm and Kafka. The streaming video features will be extracted at the client site and will be sent to Kafka, which acts as a producer and consumer in our system. Kafka processes the features and sends it to the Spark. Spark will build a model by using MLLib. This model is trained to know the face of a person whom we want to detect. The model generated by the Spark will be stored in the MongoDB.

On the other side, Storm receives a features of the video stream from Kafka. Storms uses this data for its testing purpose. Storm communicates with the MongoDB to get the trained model that's been generated by Spark and uses this in person detection. So, Storm perform parallel processing, because of which our model will be faster in detection. Lastly, upon detecting the person the output data will be stored in the MongoDB. Using a REST service call, the final results will be triggered to the Mobile Application in the form of notification.

4.2.1 Parallelism/Distribution: Task/Data

Models are used to classify the process to extract the human faces from a stream of video and put it in database. Machine learning algorithms are used here and can be processed on multiple nodes and send to the master node to process this.

Storm: A batch processing engine which can undergo micro-batch processing and works on task parallel computations using Spouts and Bolts topological architecture. It has a distributed architecture where each bolt can be parallelized. Parallelism is done on each node and the person identification is executed on these nodes.

Spark: Apache Spark uses the map-reduce paradigm and perform computations in a distributed manner. The inbuilt machine learning library name MLLib, has be used to train the model. Decision Tree, which is a supervised classification algorithm is helpful in building a tree based structure which is helpful in determining the object. The correctness

and accuracy of this algorithm can be tested using many metrics like Confusion Matrix, error rate etc.

4.2.2 Features, workflow, technologies

Spark is contributed as a UI and will be trained to identify the image from spark lib. It's been used to develop a model using its machine learning library called MLLib. The Storm is used for person and object detection. All the analysis from the videos are done in Storm. Storm converts the raw streaming data into complete data products. Kafka is used as a Pub/Sub real-time messaging system which provides durability and fault-tolerance. Kafka acts as a message broker between the Client - Spark and Client - Storm.

Storm: Apache Storm has the ability to process very large amount of real-time data. There will be many nodes in a cluster and each node is very fast and can process millions of data. There are three abstractions termed as spout, bolt and topology. Spout acts like an input i.e. the source of data which receives data from Twitter API or Kafka or any source which has the information. Bolt on the other hand process this incoming data from spout and produces many output streams. Finally, topology is the network of spouts and bolts, where the edges are connected to bolts. It is language independent and can support all languages.

Spark: Apache Spark is an extended map-reduce paradigm which supports more types of computations and has been designed to be fast. It supports higher-level tools and APIs in SparkSQL, Java, Scala, R etc. SparkSQL is used for structured data processing which uses SQL and Hive. It also provides API for graph operations. Spark works on batch processing which can process millions of data on thousands of nodes very quickly.

Kafka: Apache Kafka is distributed messaging system which commits log services. It is implemented in Java and Scala which has a high throughput to supports huge volume of data. It also supports real-time processing and handles system or machine failures. Kafka has three elements producer, consumer and Broker. Producers write the data to the brokers. Consumers read the data from brokers to process further. All the data is stored in the form of Topics which splits into partitions and each partition is replicates. It has high writes and

reads. For live streaming recently Kafka introduced Zookeeper where streaming is done through Spark package.

Workflow Diagrams:

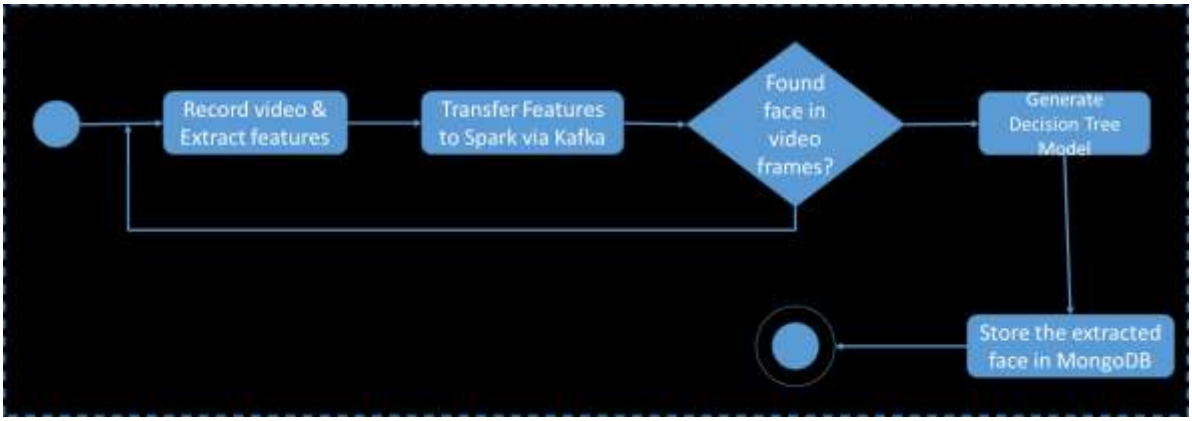


Fig. Activity diagram for collection of training dataset

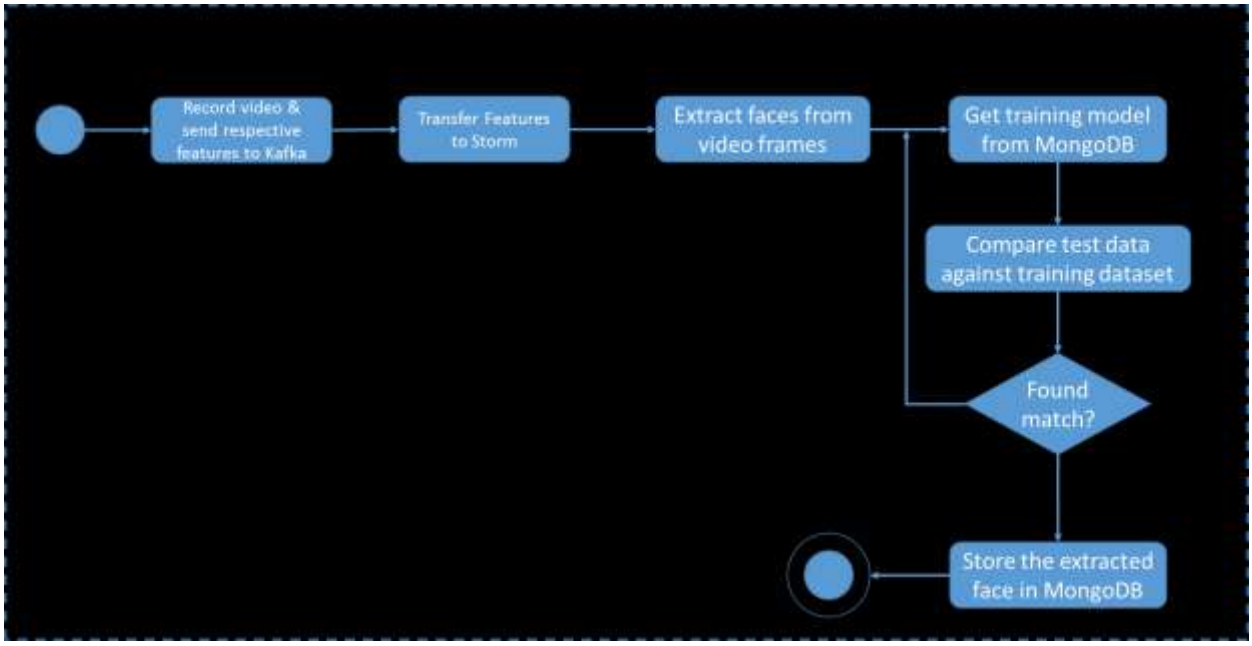


Fig. Activity diagram for person recognition

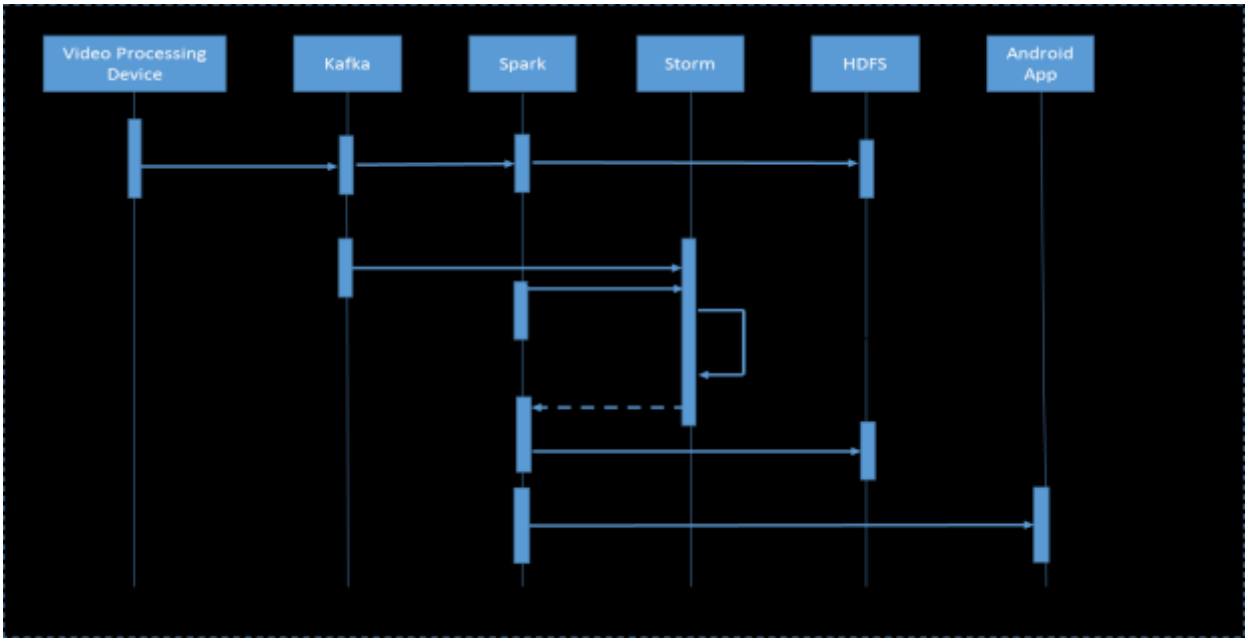


Fig. Sequence diagram for the system workflow

4.3 Design of Mobile Client (smartphone/web)

A smart phone application which is Android based will be designed in order to get an alert message. This application basically is helpful in identifying the possible threat activities.

4.3.1 Features, GUI, technologies

A push notification will be shown, when there is any trigger from Storm.

Android Java will be used to design the Android Application.

Activity Diagram (workflow, data, task)

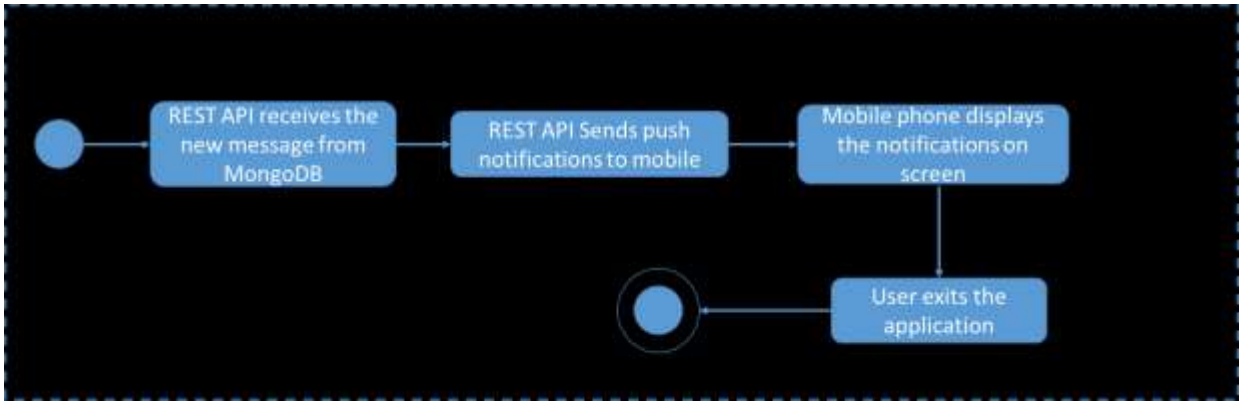


Fig. Activity diagram for alert message display on mobile

Sequence Diagram (interaction/collaboration)

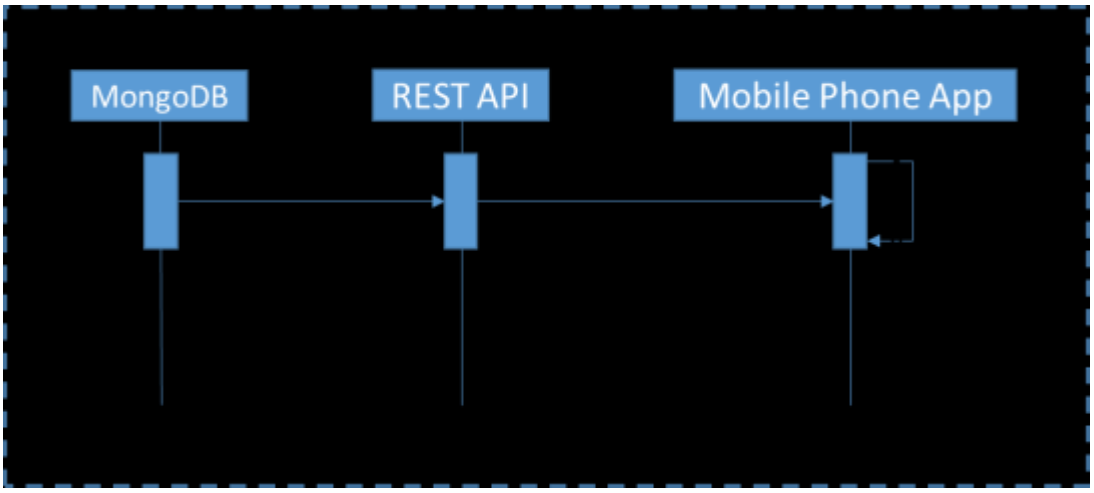


Fig. Sequence diagram for the system workflow

4.4 Existing Applications or Services Used

4.4.1 Mongo DB

We have used MongoDB which is an un structured database system, to store irrelational data in JSON format. MLab services are used in variety of ways in order to store the decision tree model generated by Spark and to store the predicted results by storm. We have used REST services in order to connect MLab services.

5. Implementation

5.1 Implementation of server using Spark and Storm and Kafka

Storm: Apache Storm has the ability to process very large amount of real-time data. There will be many nodes in a cluster and each node is very fast and can process millions of data. There are three abstractions termed as spout, bolt and topology. Spout acts like an input i.e. the source of data which receives data from Twitter API or Kafka or any source which has the information. Bolt on the other hand process this incoming data from spout and produces many output streams. Finally,

topology is the network of spouts and bolts, where the edges are connected to bolts. It is language independent and can support all languages.

Spark: Apache Spark is an extended map-reduce paradigm which supports more types of computations and has been designed to be fast. It supports higher-level tools and APIs in SparkSQL, Java, Scala, R etc. SparkSQL is used for structured data processing which uses SQL and Hive. It also provides API for graph operations. Spark works on batch processing which can process millions of data on thousands of nodes very quickly.

Kafka: Apache Kafka is distributed messaging system which commits log services. It is implemented in Java and Scala which has a high throughput to supports huge volume of data. It also supports real-time processing and handles system or machine failures. Kafka has three elements producer, consumer and Broker. Producers write the data to the brokers. Consumers read the data from brokers to process further. All the data is stored in the form of Topics which splits into partitions and each partition is replicates. It has high writes and reads. For live streaming recently Kafka introduced Zookeeper where streaming is done through Spark package.

5.2 Implementation of any machine learning library or multimedia library.

Sparks Machine learning library named Spark MLLib, provides us with many algorithms and utilities for classification, regression etc. It is a well-developed library and proven with the best results. We have used Decision Tree algorithm out of it to generate our model which is used for person identification.

5.3 Implementation of mobile clients

Implemented an android based mobile application, which is developed using Android Studio. The application basically aims to notify the user about any suspicious activity. It gets its notification information from the MongoDB.

6. Project Management

6.1.1 Plan & Project Timelines

Increment	Deadline
Increment 1	23 September 2016
Increment 2	14 October 2016
Increment 3	18 November 2016
Increment 4	5 December 2016
Final Submission	9 December 2016

6.1.2 Project Members

- Sri Harsha Chennavajjala (4)
- Priyadarsini Nidadavolu (16)
- Tej Kumar Yentrapragada (27)
- Chaitanya Sai Manne (14)

6.1.3 Project GitHub Repository Link

https://github.com/npdarsiniOrg/RTB-Project/tree/master/Source/Phase-1

https://github.com/npdarsiniOrg/RTB-Project/tree/master/Source/Phase-2

https://github.com/npdarsiniOrg/RTB-Project/tree/master/Source/Phase-3

https://github.com/npdarsiniOrg/RTB-Project/tree/master/Source/Phase-4

6.2 Implementation Status Report

6.2.1 Phase 1:

Key frame detection and tags generation: We used SIFT techniques to find the key frames in the video. Later we passed the key frames to Clarifai API to generate the tags for the objects in

the video frames. These tags were added to each video frame and we created a short video. Output of this short video is as shown in the below figures.

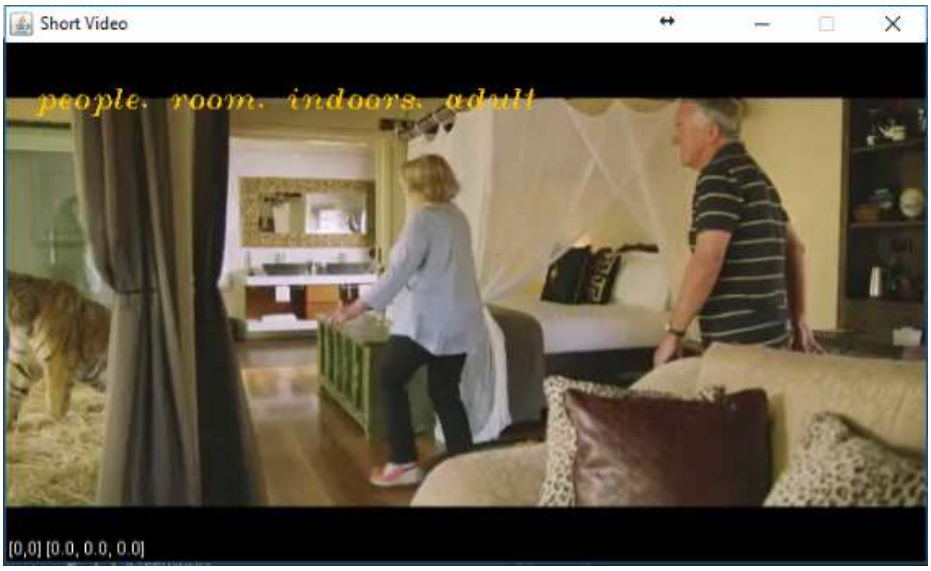


Fig. Key frame with relevant tags

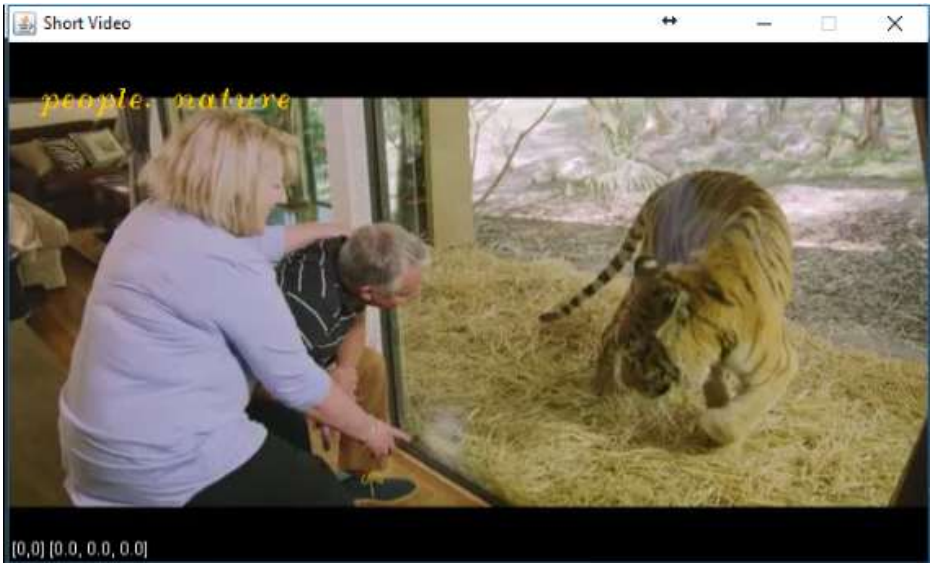


Fig. Key frame with relevant tags

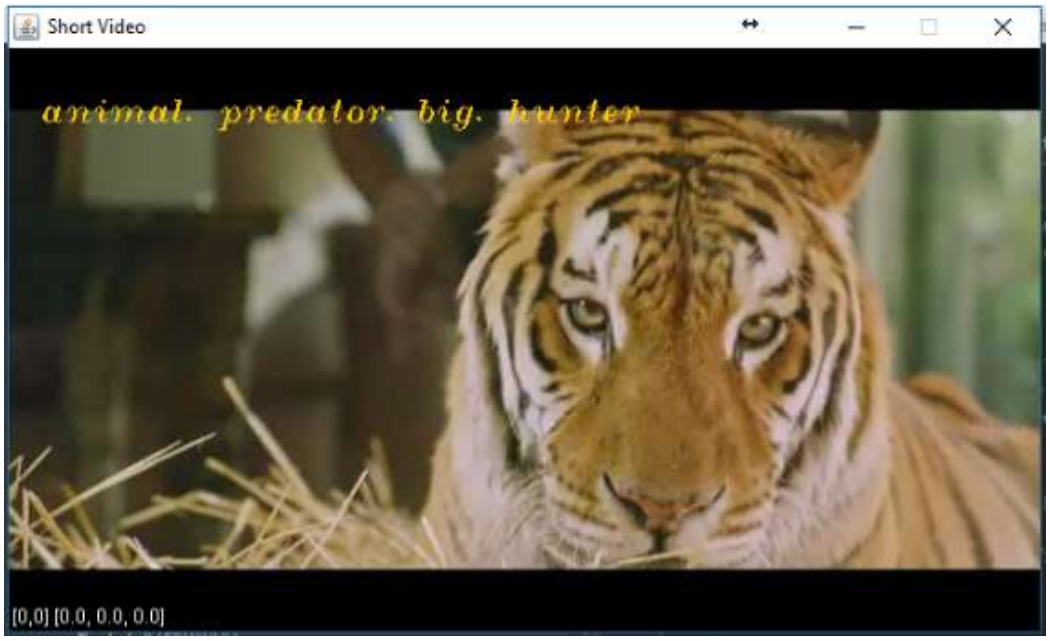


Fig. Key frame with relevant tags

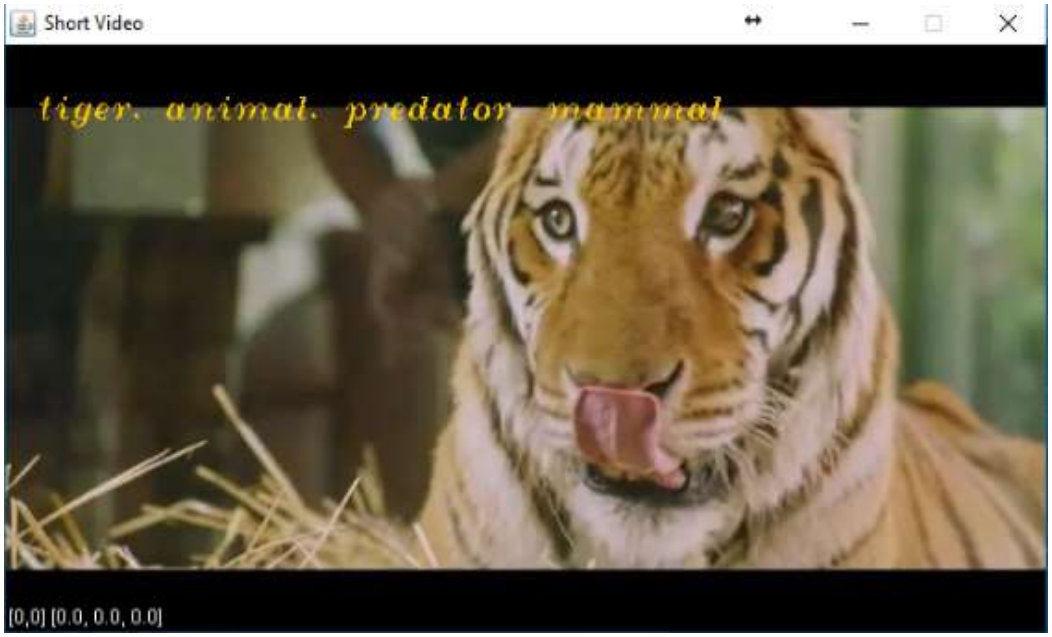


Fig. Key frame with relevant tags

Fig. Key frame with relevant tags

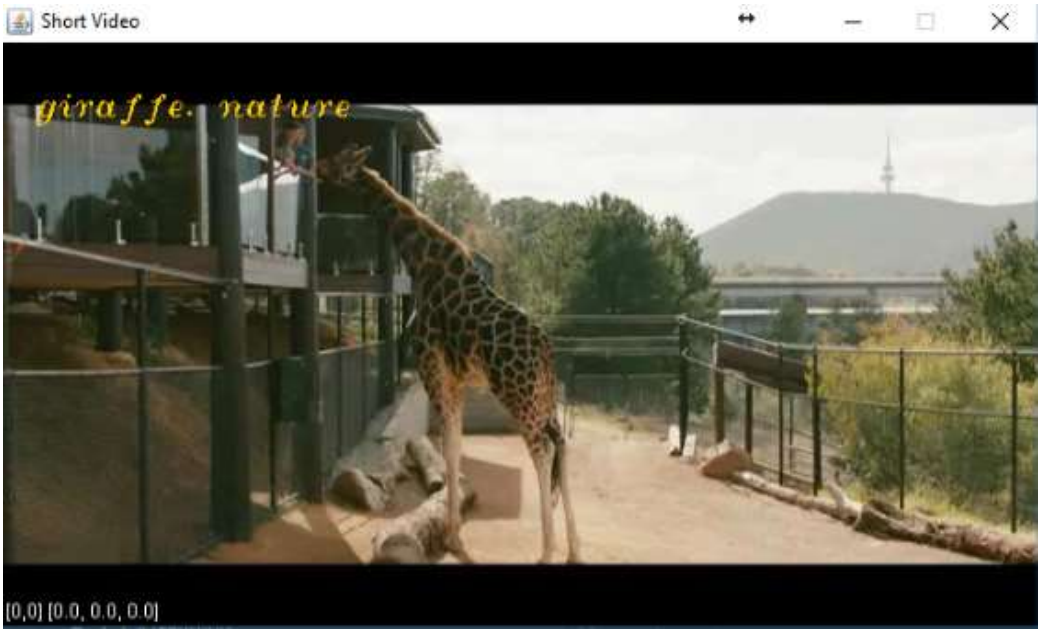


Fig. Key frame with relevant tags

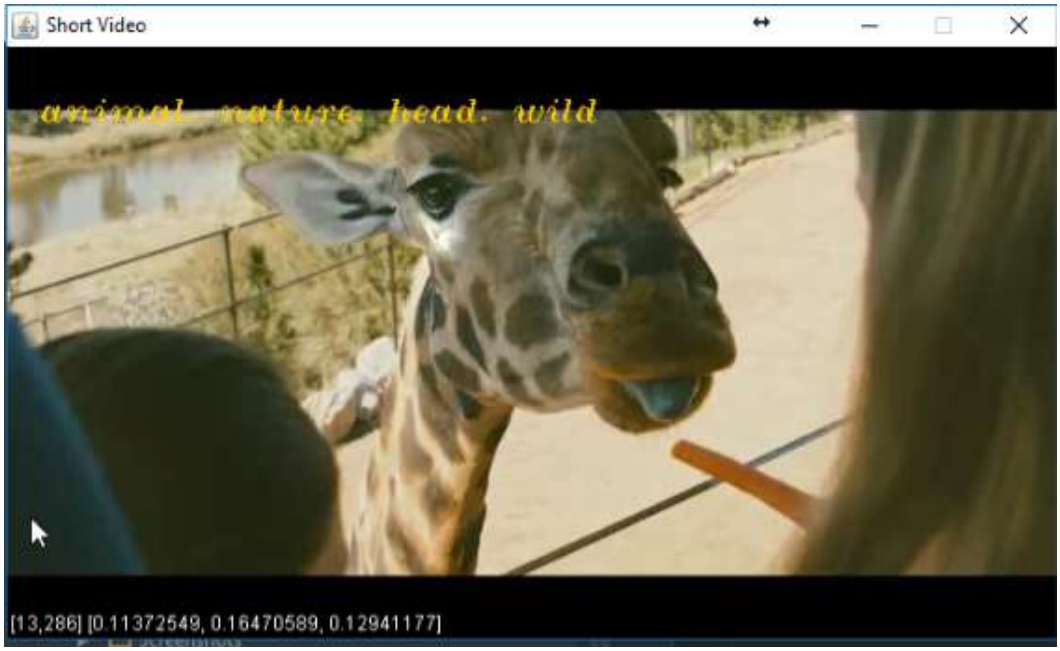


Fig. Key frame with relevant tags

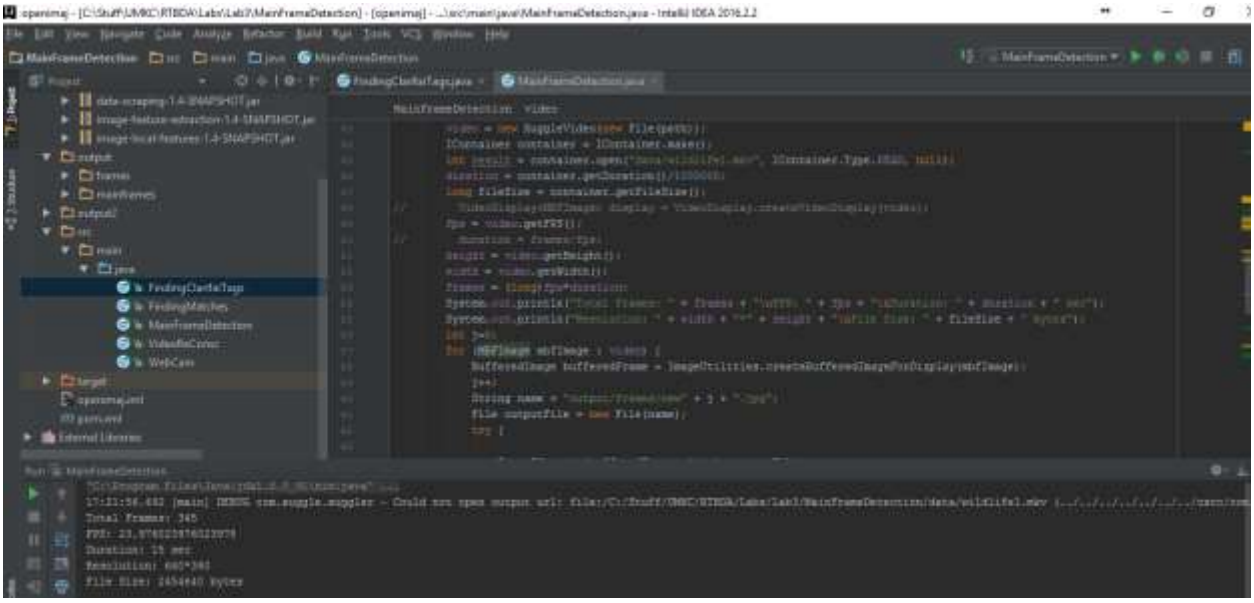


Fig. Implementation of logic in IntelliJ IDEA

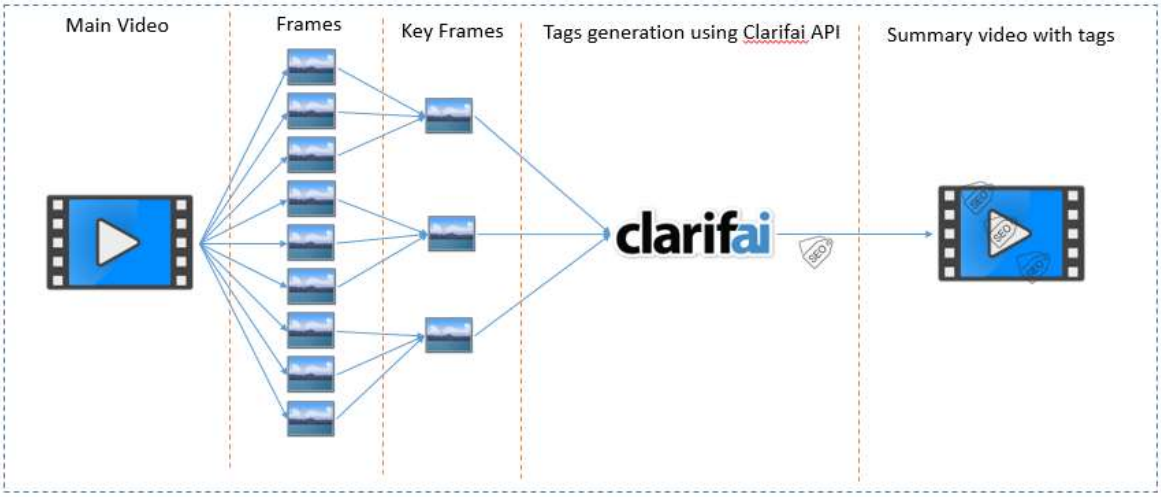


Fig. Workflow of key frame generation and adding tags to key frames using Clarifai API

Person face and full body detection in a video stream:

In this later stage, we first tried to detect the human faces in the input video stream. Later we extended this feature to detect different parts of human body (eyes, nose etc.) and in the final the full body of the person. The detected human body will be highlighted with a rectangular box around it as shown in the below screenshots.

Fig. Person body detection

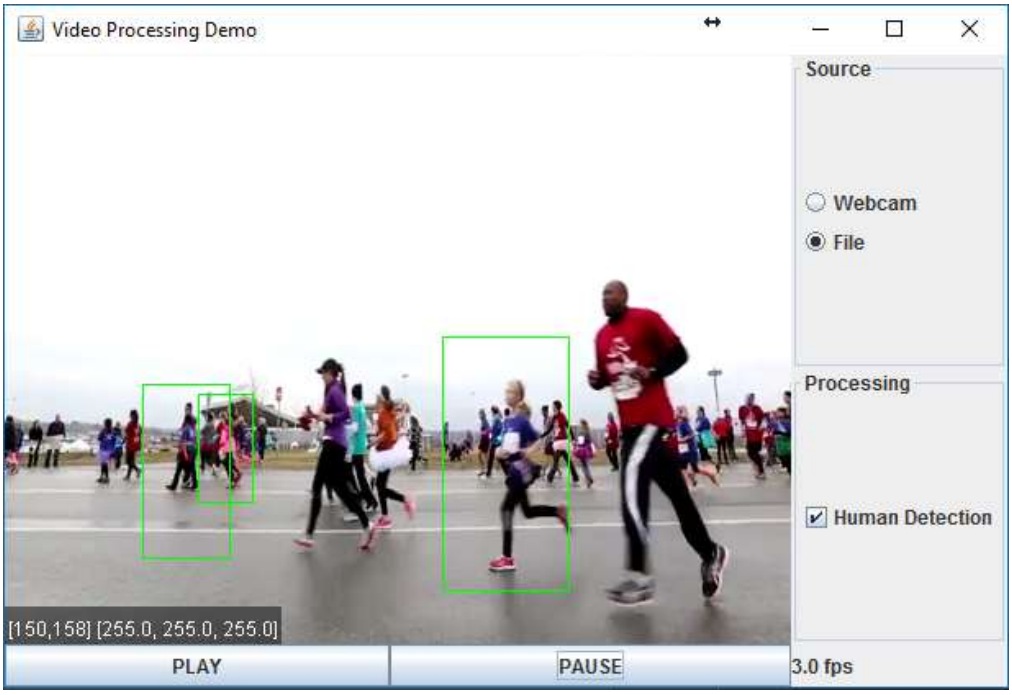


Fig. Person body detection

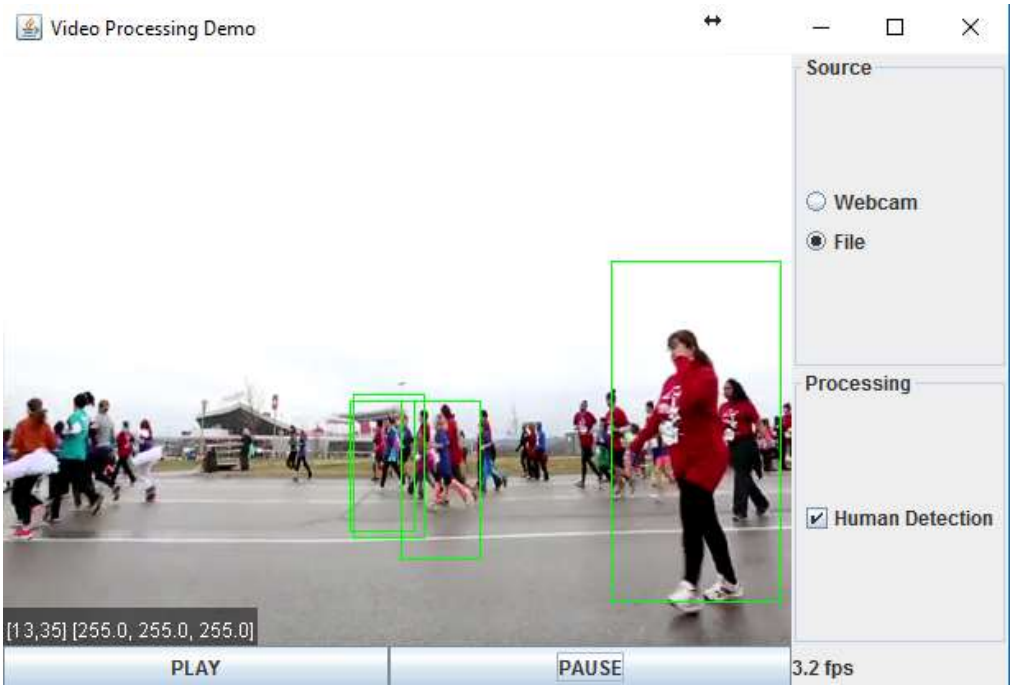
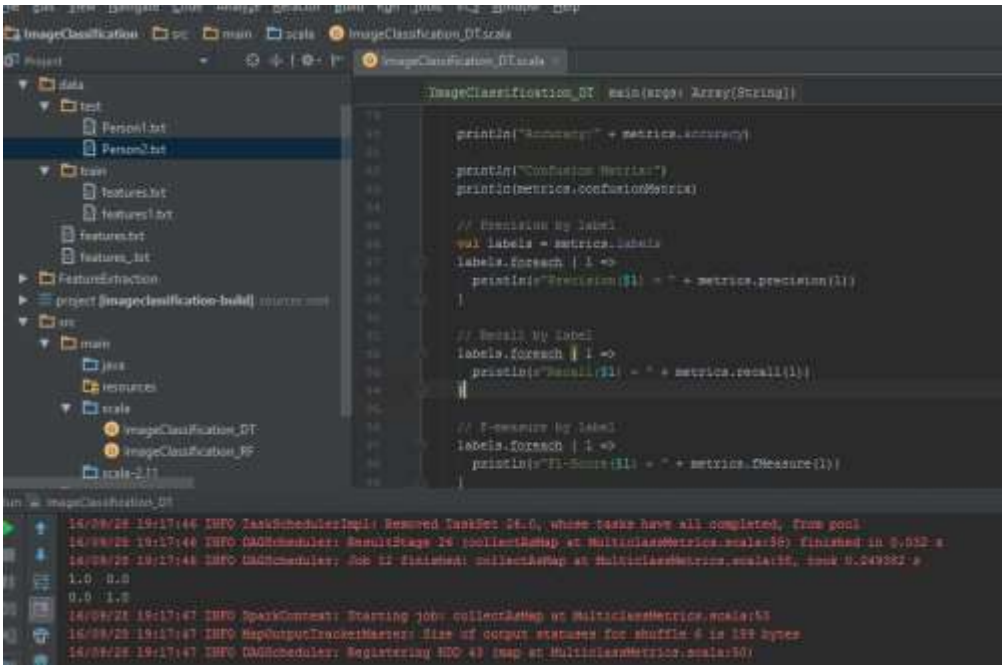


Fig. Person body detection

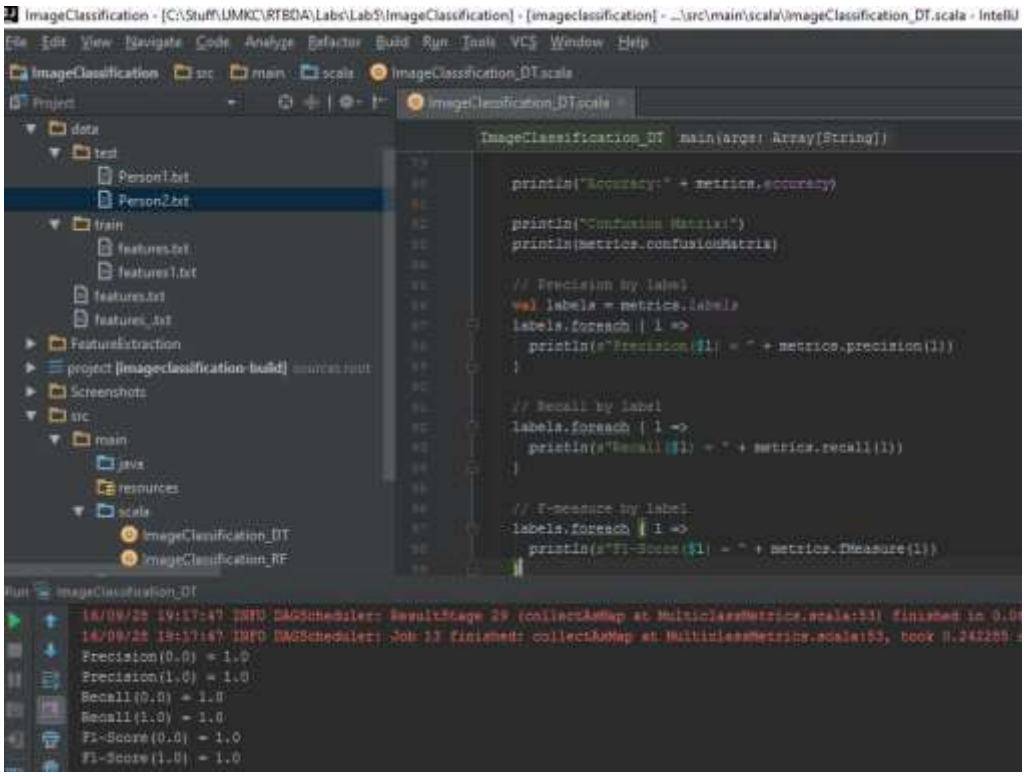
6.2.2 Phase 2:

As part of this phase we are able to perform Image Classification and able to establish communication between Kafka producer and consumer.

- A stream of video is given as an input to Spark. Upon using Random Forest/Decision Tree classification algorithms we are able to detect the person in the streaming video. We had reframed the video and annotated the video with the detected person name. We have tested the accuracy of the classification with both the algorithms and concluded that Decision Tree provides best results. It provided us with an accuracy of 74%. In this we are generating a confusion matrix and validating the effectiveness of our model.
- On the other side, we are able to establish communication between Kafka producer and Kafka Consumer. We are performing Key Frame extraction and detected Main Frames out of the streaming video has been sent to the Kafka consumer from the producer via broker. Able to send multiple image files to the consumer via stream.

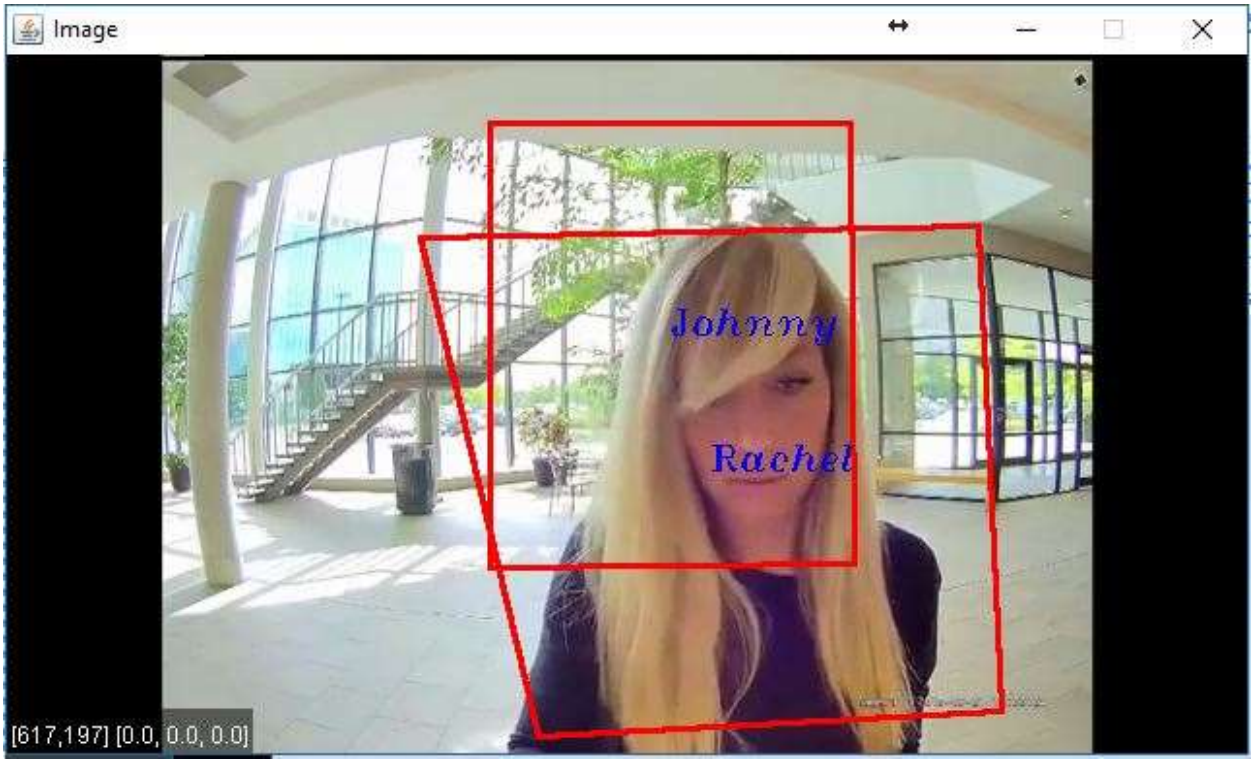


Output of Confusion Matrix which is used to show the model accuracy

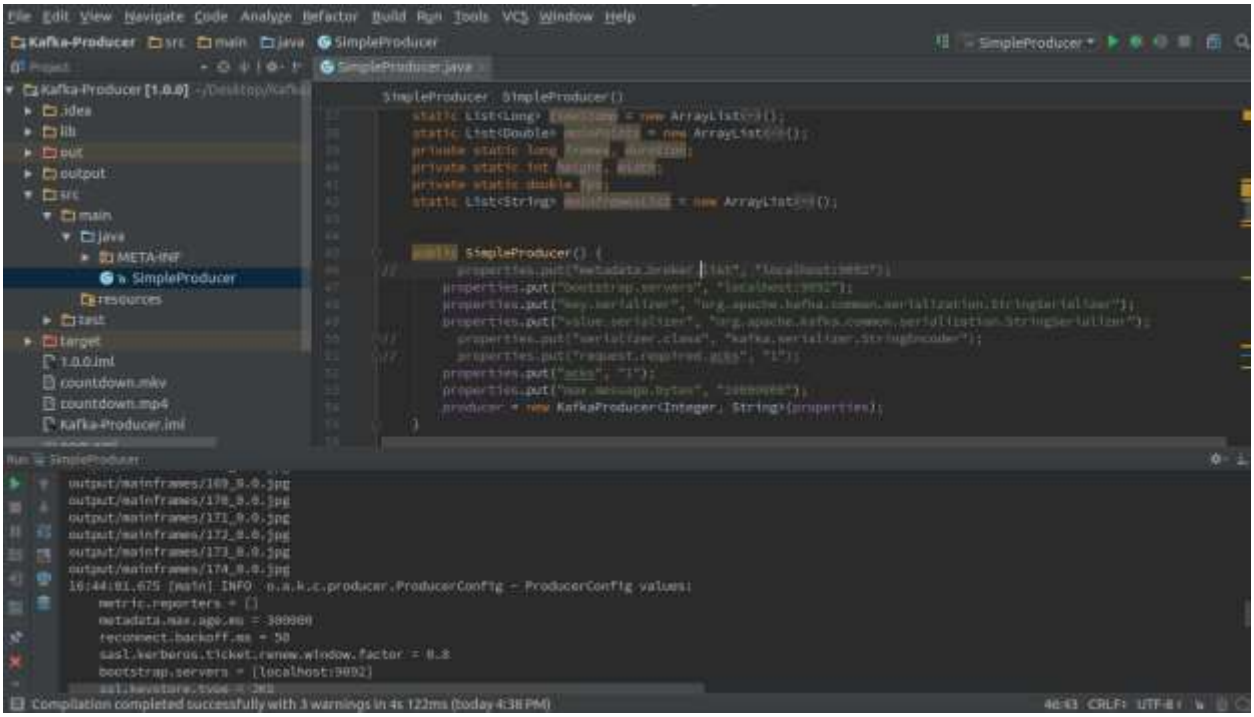


Precision and F Statistics of the model

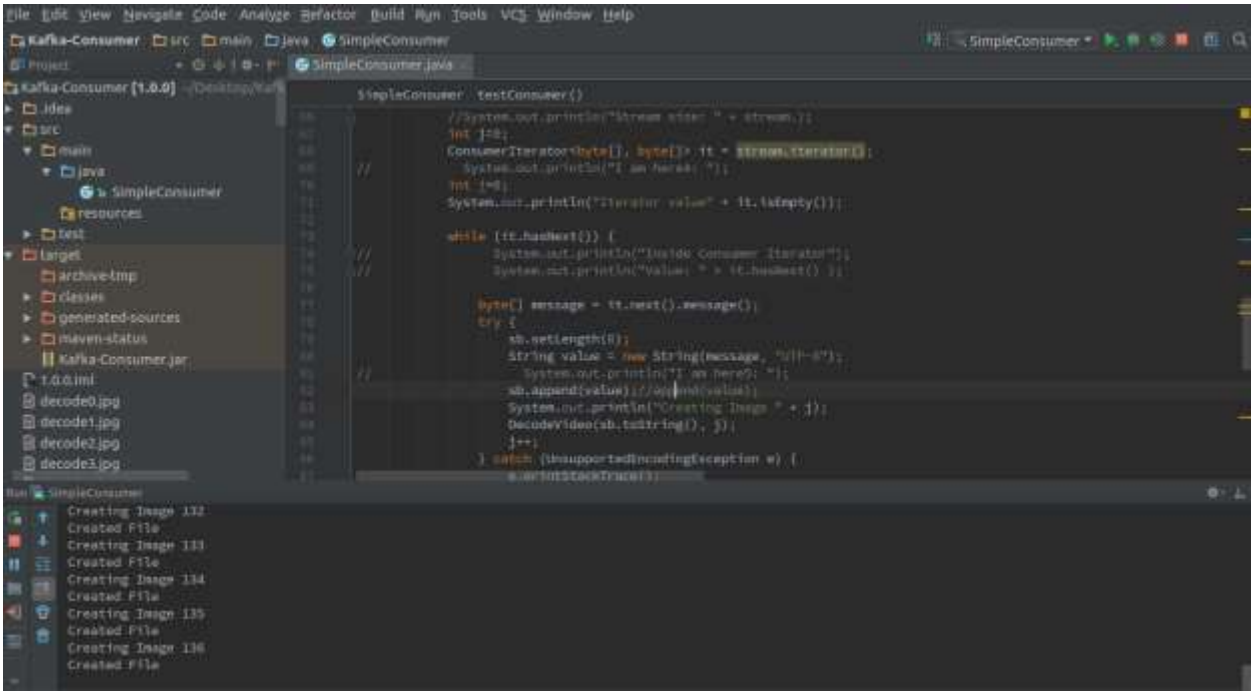
Person Identification and annotation with name - Johnny



Person Identification and annotation with name – Rachel

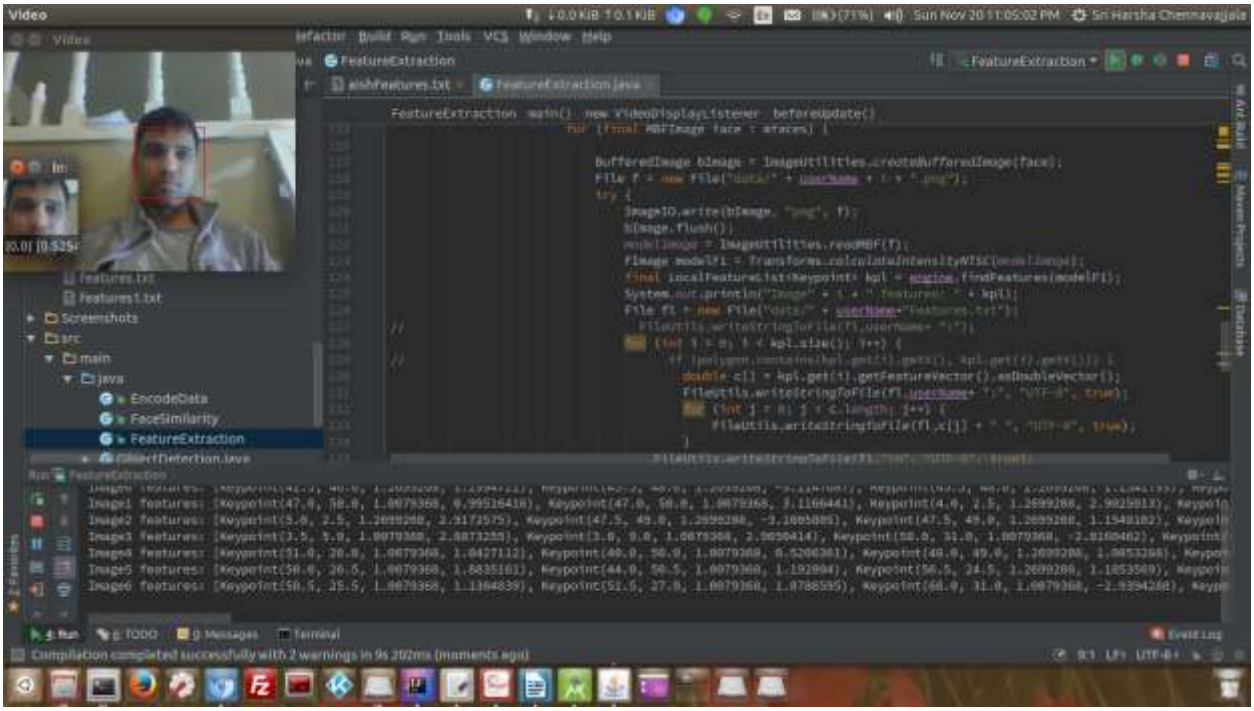


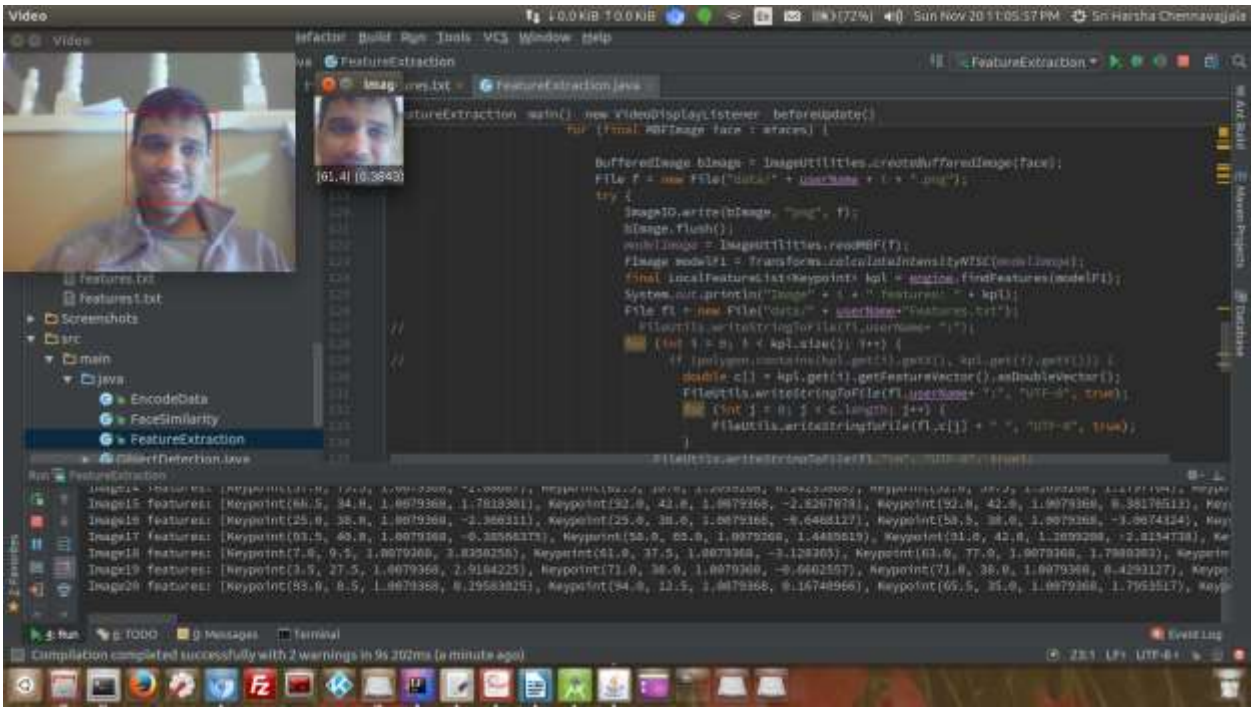
Kafka- Producer sending the key frames



Kafka Consumer- To receive images from Kafka Producer

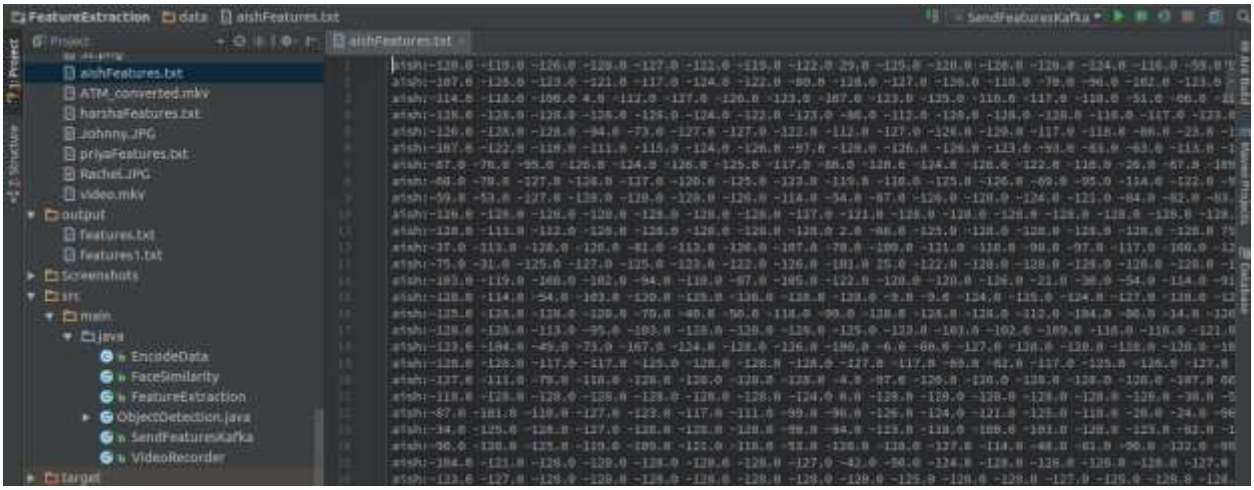
6.2.3 Phase 3:



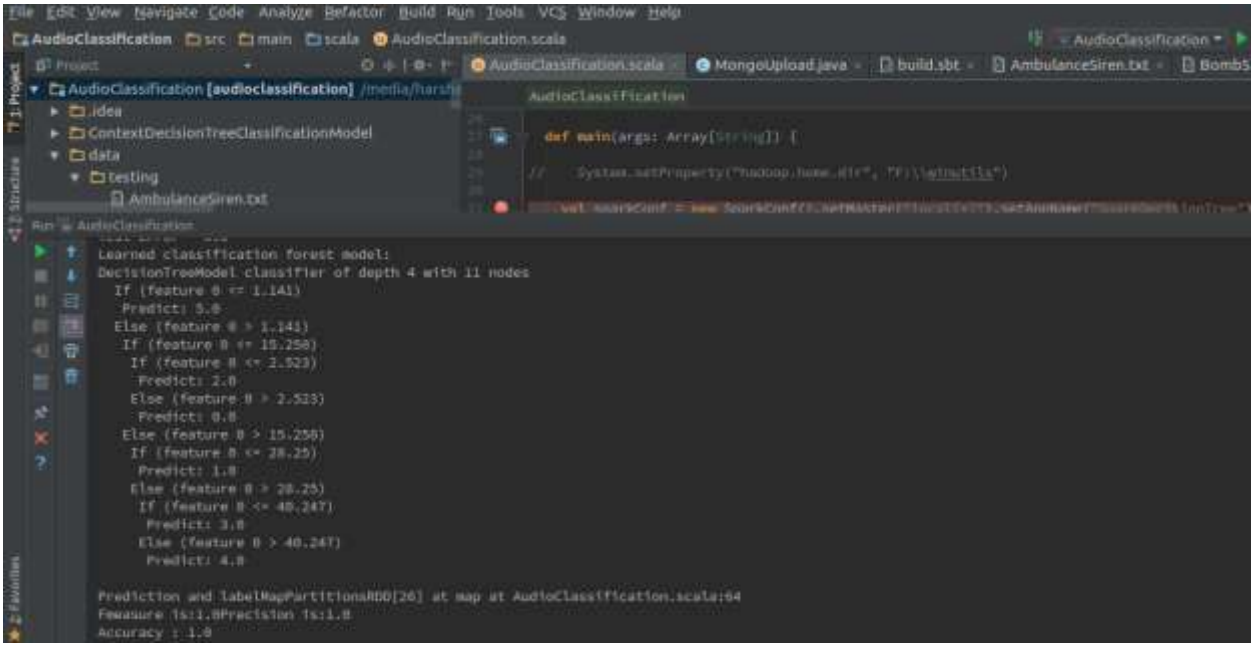


Capturing Input

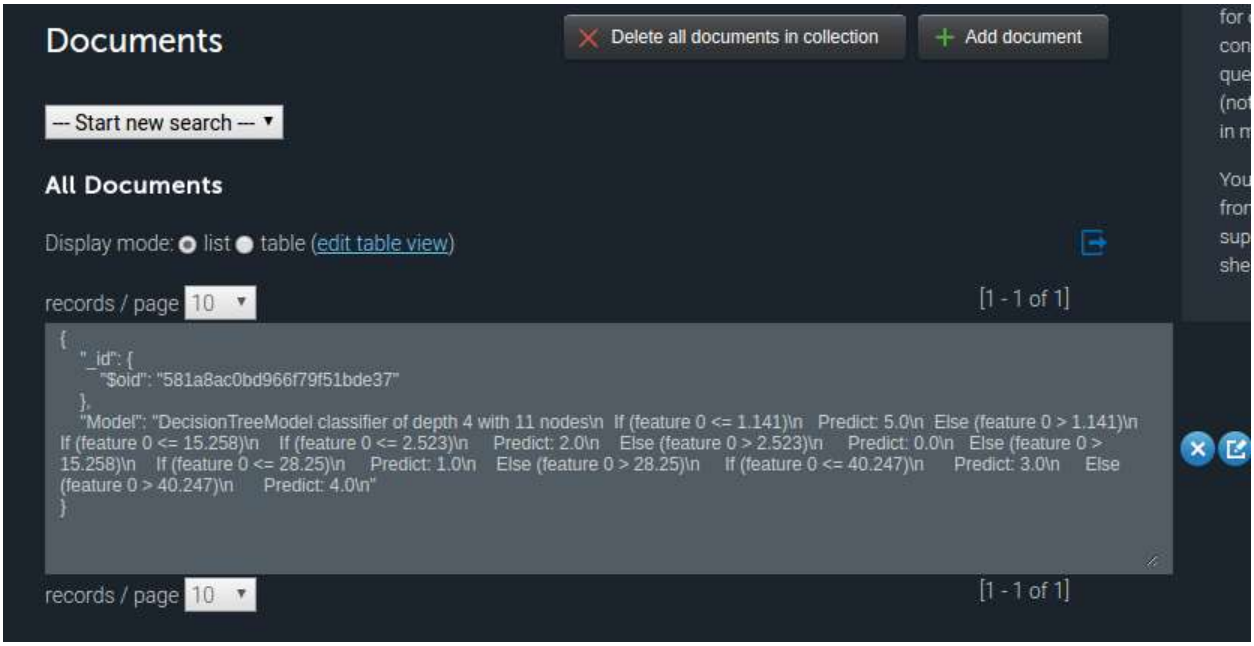
Image Capture using OpenIMAJ.



Features extracted out of the training images



Decision Tree model generated by Spark



Model stored into MongoDB

6.2.4 Phase 4:

In the first stage, we used OpenIMAJ framework to extract features. Our Region Of Interest (ROI) is human face. So, we extracted the face region first. Later we extracted the features for this image. These features were passed to both Storm and Spark with the help of Apache Kafka producer-consumer messaging system.

The spark part will have both the classification (in this case user name) and the extracted features. These features from client application were passed to the Spark application using Kafka. At the Spark application end, we divided the gathered features into testing and training datasets (70,30). We applied decision tree algorithm on these datasets to generate the prediction model. Our datasets have nearly 7500 extracted features. This helped us to get the reasonable results in the prediction phase. A screenshot of facial features extraction is as shown below.

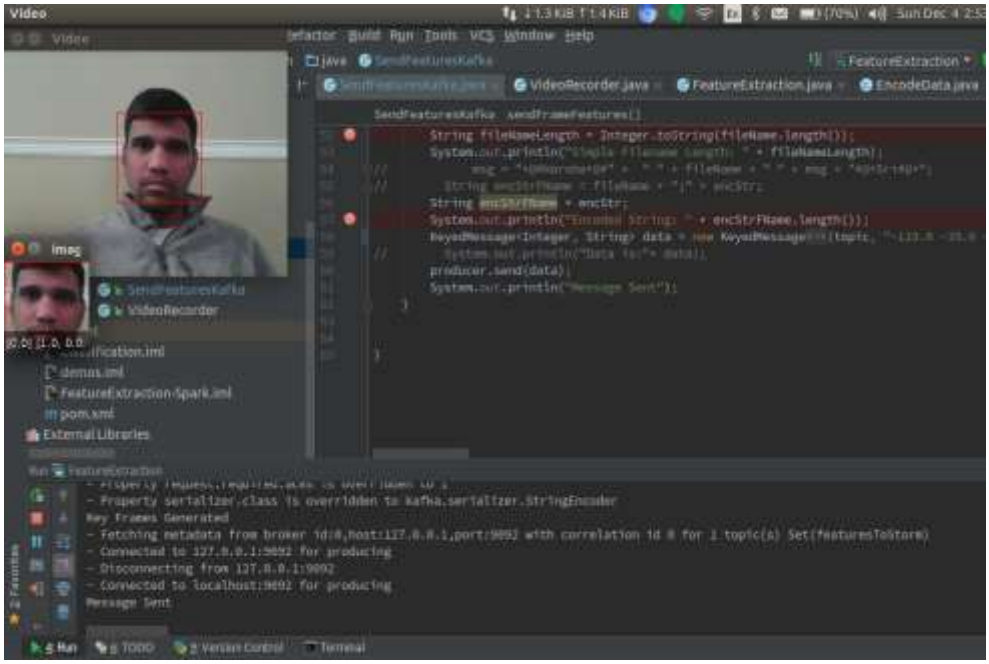


Fig: Facial feature extraction

Later we stored the obtained prediction model to MLab database as shown in below figure. This model can later be used in Storm for person recognition.

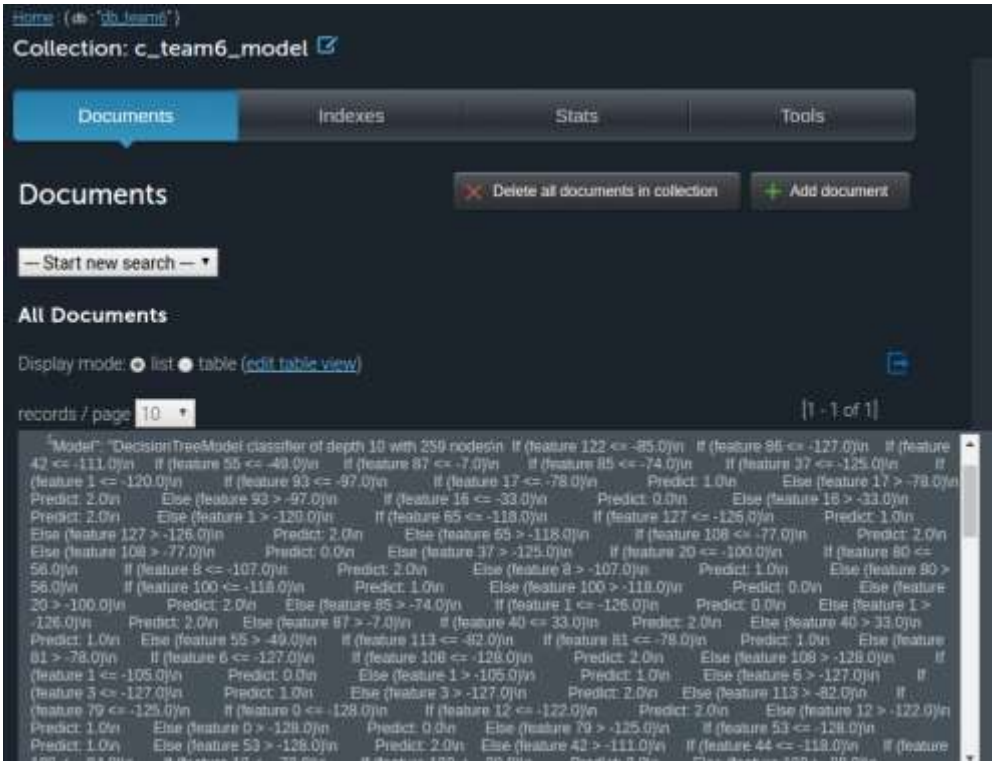


Fig: Storing the DT model in MongoDB

In Storm application, we created a KafkaSpout to get the features stream from the feature extraction application as mentioned above. In our test case, we generated the model to identify three persons harsha, priya and aish. Therefore, for each person we created a bolt. These bolts retrieve the model from the MLab database and parses it according to the user's specification. The received features stream will be passed to these bolts continuously. These bolts will validate the model against the predefined model and returns the results. In our case, if the person is recognized, then we get "true" otherwise we get "false". If we found a person in the video, then the bolt will create a record in MLab database. This record will contain the data like identified user, time at which the user is identified. A screenshot of our topology graph is as shown below.

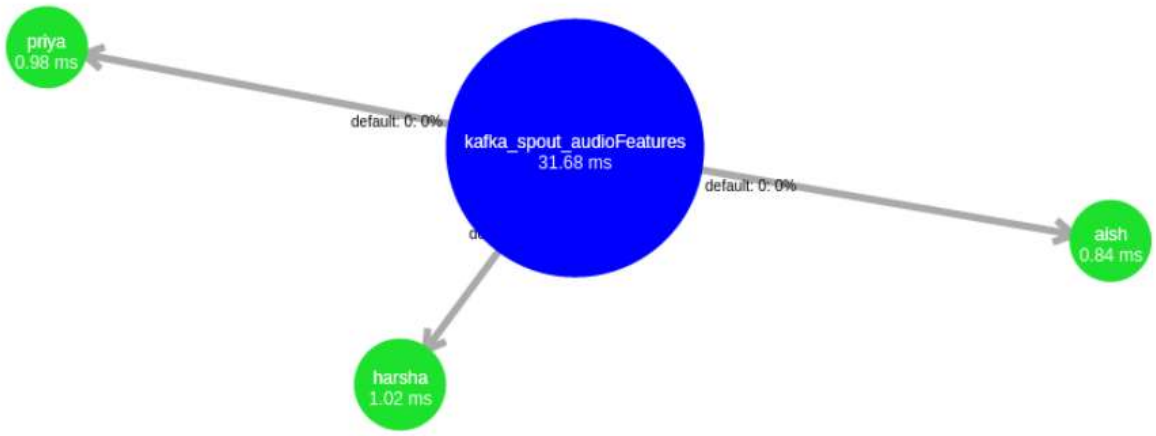


Fig: Storm Topology

Our prediction model identifies the person correctly in 8 out of 10 cases. A screenshot of the stored predictions is as shown below.

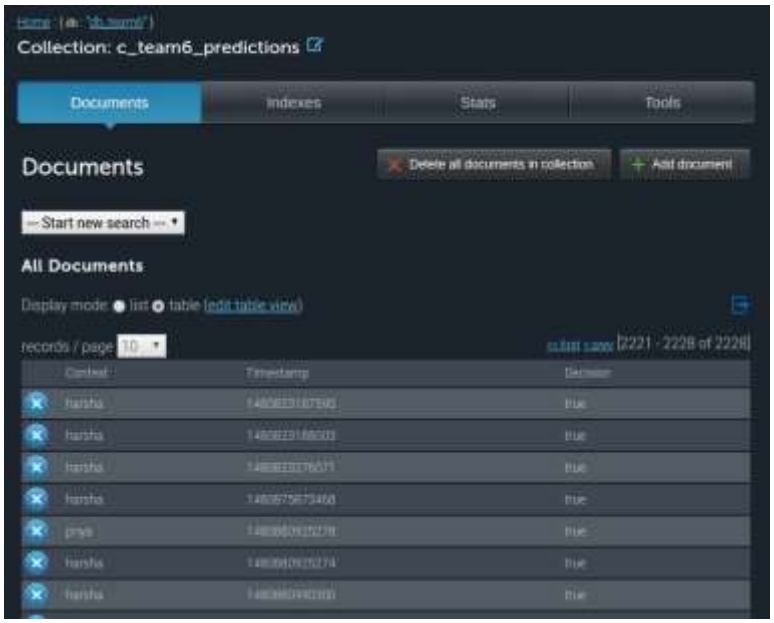


Fig: Prediction results in MongoDB

We also created a REST based mobile application. The REST server and the client application will work with the help of Google Cloud Messaging service (GCM). The node server will keep on running in the background. Once the client registers with the server, the application receives the alert messages. The registered device details will be stored in the MLab database as shown below.

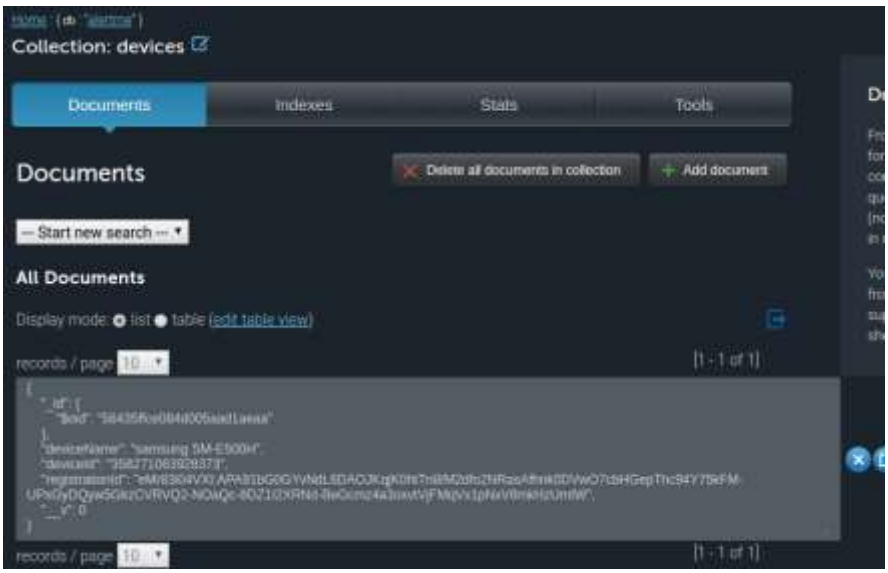


Fig: Storing the GCM registered devices in MongoDB

Whenever there is a person recognized in the Storm application, the node server will send an alert message to the android device as shown in below picture. The alert message can be delivered even when the application is not running.

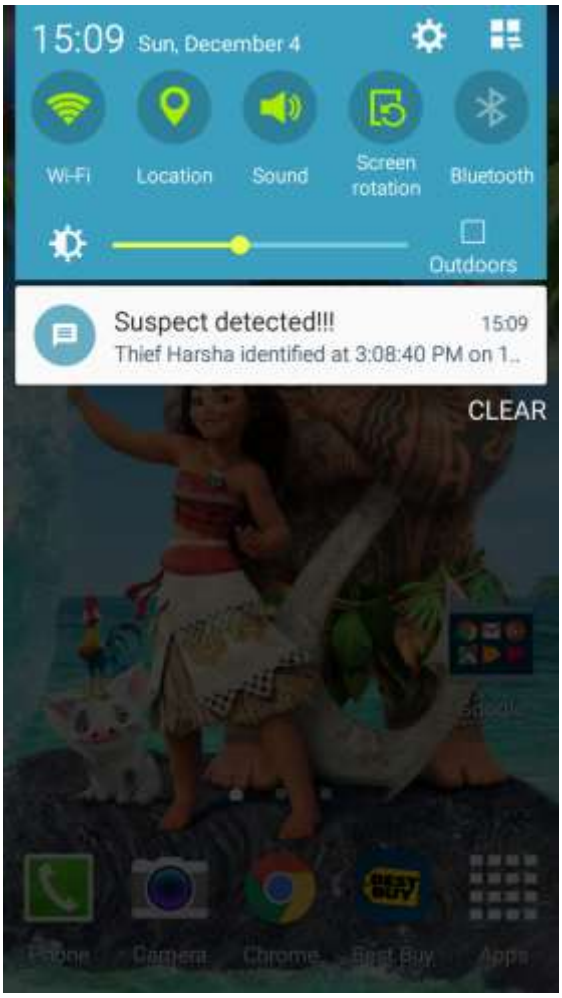


Fig: Alert in notification drawer (Android)

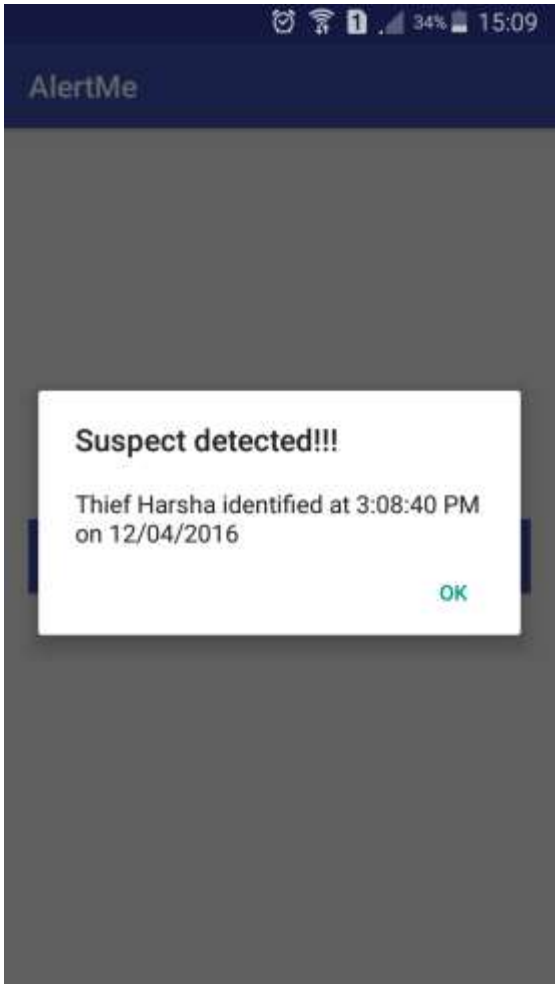


Fig: Alert popup in the application

6.3 Work completed

6.3.1 Phase 1:

We've divided the increment 1 tasks in to three groups, first one is requirements gathering. In this, we've gone through various research papers and understand the basic steps to be followed in implementing the proposed system. We identified the advantages and performance improvements that we can achieve by using Apache Spark and Apache Storm in processing and

analyzing the video frames. Each one from our team has participated actively and after several group meetings we concluded the system architecture mentioned in the earlier section.

Second one is, reading a video file and extracting the key frames from the video. Tej Kumar and Chaitanya worked on this task and they are able to successfully complete the task in less time. Later Sri Harsha and Priyadarsini used these extracted key frames in generating the tags for the objects in these frames using Clarifai API. We've also created a short video using these key frames and the generated tags. This helped us in understanding the framework and workflow of OpenIMAJ.

Third one is, detecting a person's face and the key points on his face (nose, mouth, eyes etc.) using HaarCascadeDetector class of OpenIMAJ framework. Tej Kumar and Chaitanya worked on these tasks. They are able to successfully annotate the detected faces with a rectangular box in a given video stream. Later Sri Harsha and Priyadarsini used these results to extend the functionality by detecting a person in the given video stream. We've tested various built in cascades available in OpenIMAJ (eyes, mouth, upper body, lower body, full body etc.,) and are finally able to highlight the persons with a rectangular box.

6.3.2 Phase 2:

As part of this phase, we have divided the tasks based on the domain expertise. Sri Harsha and Priyadarsini had worked on establishing the communication between the Kafka producer and consumer. Upon extracting the key frames out of the streaming video, the set of images has been sent to the Kafka Consumer from Kafka Producer. The establishment of Kafka system in the local system has been done to avoid issues with the centralized server. A topic is created in the Kafka server and is able to subscribe to that particular topic. The key frames are been tagged to a particular topic and the consumer who is subscribed to that topic will be receiving these images when there is a trigger from the producer.

On the other side, classification on the streaming video using Spark MLlib has been done. We have performed classification using two algorithms namely Decision Tree and Random Forest. Tej Kumar and Chaitanya has performed the classification on both the algorithms by training bthe

model with two different persons and tested for their accuracy against each algorithm. They concluded that Decision Tree has provided better results and gave an accuracy of 74%. Then we reframed the video using the extracted key frames and annotated the video based on the classification. In this phase we identified two persons and annotated the video with the person name.

6.3.3 Phase 3:

As part of this phase, we have collected the sample training set of images of three different people, and extracted the features of those images. These features are sent to Spark through Kafka message broker. Using these features, we are able to build a training model and have stored this model in the MongoDB using REST service call to MLab. There are two kinds of models that are being generated. One is with respect to audio and the other is with respect to images. In order to generate the model in Spark, we have used Decision Tree algorithm and Spark's MLLib machine learning library. We are able to generate the model in terms of Audio and images and is able to store these models to MongoDB.

The model that is built by Spark will be fetched by the Storm from the MongoDB and used for classification. As we know that the video is a combination of Audio and Images, our work is done for the Audio from the Storm side. We are working on the images prediction at the Storm side.

6.3.4 Phase 4:

As part of this phase, we are able to fetch the model from MongoDB by the Storm. This model will have the training model generated by Spark for image classification. Storm, upon receiving the testing stream of data will use this model to predict a person. Upon prediction, the results will be stored into the MongoDB. We have an android application that receives a notification upon MongoDB update. So when the MongoDB is triggered with the Storm, a notification will be sent to the Android application.

Contributions: Sri Harsha 25%, Priyadarsini 25%, Chaitanya 25%, Tej Kumar 25%.

Fig. Project contributions graph in GitHub

6.4 Work to be completed

As of now, we have trained with a simple model and the accuracy is not so guaranteed. So the next task is to generate a more generic model which could result in the accuracy of predictions. Time to be taken: 20 hrs/person.

6.5 Git Issues/Concerns

The cascades available with OpenIMAJ are not reliable in detecting a person. We got false positive results in some scenes of the input video.

We have faced issues while sending the multiple key frames to Kafka consumer. We are able to send only a single image file to the consumer and is able to overcome them.

We are facing a lot of issues on the centralized server, hence we had setup Kafka and Storm in our local machines.

We have faced a lot of issues while sending the bulk of features to the servers via Kafka. Also, the architectural changes made our job a bit more confusing for a while.

7. Future Scope

- To identify the scenario and the location where the activity is being happening.
- To find the expression of a person, so the classification can be done based on the person's expression whether he is going to perform any suspicious activities.
- To establish communication between Spark and Storm directly so as to get the model from the Spark.

8. Bibliography

- Face recognition using Laplacianfaces - http://ieeexplore.ieee.org/document/1388260/?arnumber=1388260
- Face recognition using eigenfaces - http://ieeexplore.ieee.org/document/139758/?arnumber=139758&tag=1
- Facial recognition techinques - https://en.wikipedia.org/wiki/Facial_recognition_system
- Computer Vision - http://opencv.org/
- OpenIMAJ Face Recognition Library - http://openimaj.org/openimaj-image/faces/dependencies.html
- Tags generation for a video frame - https://www.clarifai.com/
- Spark Architecture - http://static.usenix.org/legacy/events/hotcloud10/tech/full_papers/Zaharia.pdf
- Storm Architecture- http://db.cs.berkeley.edu/cs286/papers/storm-sigmod2014.pdf
- Kafka Architecture - http://people.csail.mit.edu/matei/courses/2015/6.S897/readings/kafka.pdf
- Video Streaming - http://ieeexplore.ieee.org.proxy.library.umkc.edu/stamp/stamp.jsp?tp=&arnumber=7412619