Voice-Interactive, Immersive Sports Experience

A Question and Answering system for sports domain using video analysis.

Team 6 - Affinity

Gulnoza Khakimova
Ganesh Taduri
Raghava Kundavajjala

1.PROJECT OBJECTIVES:

1.1 Background

Since the popularization of virtual reality, the focus has predominantly centered around gaming. This can be attributed to a variety of factors such as the gaming community's tendency to embrace new technologies and the familiarity of computer-generated graphics to gamers. Now that virtual reality has become more familiar to mainstream consumers there is a push to expand its use into other areas of entertainment, including professional sports. Over the past year or so major sports broadcasting companies, such as FOX Sports and NBC, have partnered with tech startups to release 360°, immersive VR coverage of select sporting events like the 2016 Summer Olympic Games and the 2015 NBA season opener.

While VR for sports content has proven to be a promising market, there continues to be a number of aspects that can be enhanced to provide a more enjoyable user experience. One problem is where and how to display relevant information to the user. In a 360° environment care must be taken not to obstruct the user's view while displaying information. Another opportunity for improvement is the lack of control for users to determine what information they want and when they want it. Finally, a third concern is the limited methods of interaction between the user and the environment. The aim of this project is to address these issues by allowing the user to interact with the VR environment via voice commands. Voice interaction would improve the first issue by allowing the system to respond to voice commands with audio feedback, rather than visual, thus reducing the amount of screen clutter. User's would also gain control of the information they receive by using voice queries to request the information they need. By utilizing voice commands users would have another, more natural, method of interaction with system in addition to headset buttons and remote controls that are included with many current VR headsets.

1.2 Significance

Voice controlled computer systems have been in existence for some time and used in a variety of academic and commercial fields. However, the use of voice commands within virtual reality environments has appeared only recently. For virtual reality professional sports coverage, the ability for viewers to use voice-interaction is non-existent. This can most likely be attributed to relatively recent emergence of the VR sports market. The majority of effort in early development was placed on capturing high-quality footage in a format that could be viewed in 360°. Now that the concept of VR sports has been established and shown promise in marketability, efforts should shift to enhancing the viewer's experience by providing new features.

1.3 Objectives

Although this project will focus on improving interactions with VR sports content, the ideas and results would apply to a variety of other media. Our primary purpose is to take an experience that has traditionally been static and enable it to be user controlled in a natural way. In doing so, we expect the outcome to improve the user's visual experience by reducing the amount of data displayed over visual content, enhance the overall experience by providing relevant contextual information when the user wants it, and promote increased adoption of VR for entertainment purposes outside of gaming through intuitive user interactions.

1.4 System Features

The main feature of the system is to answer a specific question asked by a user related to a sport video. A few of the examples that user could ask the system are:

- What sport is it?
- *Game info - Who's winning? What quarter is it? How many hard until first down?*
- *Player statistics - What's the pitcher's ERA? How many assists has Kobe made so far?*
- *Switch view perspectives - Take me to the fifty-yard line.*

2. APPROACH

2.1 Data Sources

We collected large amount of data sets - images and videos for training and testing the model. The major data sources that we used for collecting the data are google images and videos from youtube. For image classification it was a little difficult to find better images on all different categories of sports so instead we extracted main frames from the video we collected from youtube and used these as images for training the model.

2.2 expected Inputs/Outputs

This application is a typical Question and Answering system where a user can ask a question about a sport Image/video and system answers the question. A few of the expected questions and answers from a user and the system respectively are

1. Q: What game is it?
   A: This is Basketball
2. What is the current score?
   A: The current score is Nadal:3; Federer : 5
3. Who would win the game?
   A: According to previous Statistics seems like Federer would win the Game.

2.3 Algorithms

The major algorithms we used for this project are related to Machine Learning algorithms. For phase 1 to implement the image classification we Decision Tree and Random Forest algorithms. The Main intention of this work is to study how different machine learning algorithms are performing on different sets. In the future phases of the project we are also planning to implement Naive Bayes Algorithm. Also, we are planning to implement Neural Networks for Deep learning in the future.

3. RELATED WORK

Currently, there are several tech startups which partner with broadcasting companies and major league sports. A few of the projects that are related to this project are:

- *Fox Sports VR*[7]
- *NextVR*[11]
- *EON Sports*[12]
- *LiveLikeVR*[13]
- Virtually Live[14] - virtual reconstruction, social aspect

Research and products incorporating voice control and VR:

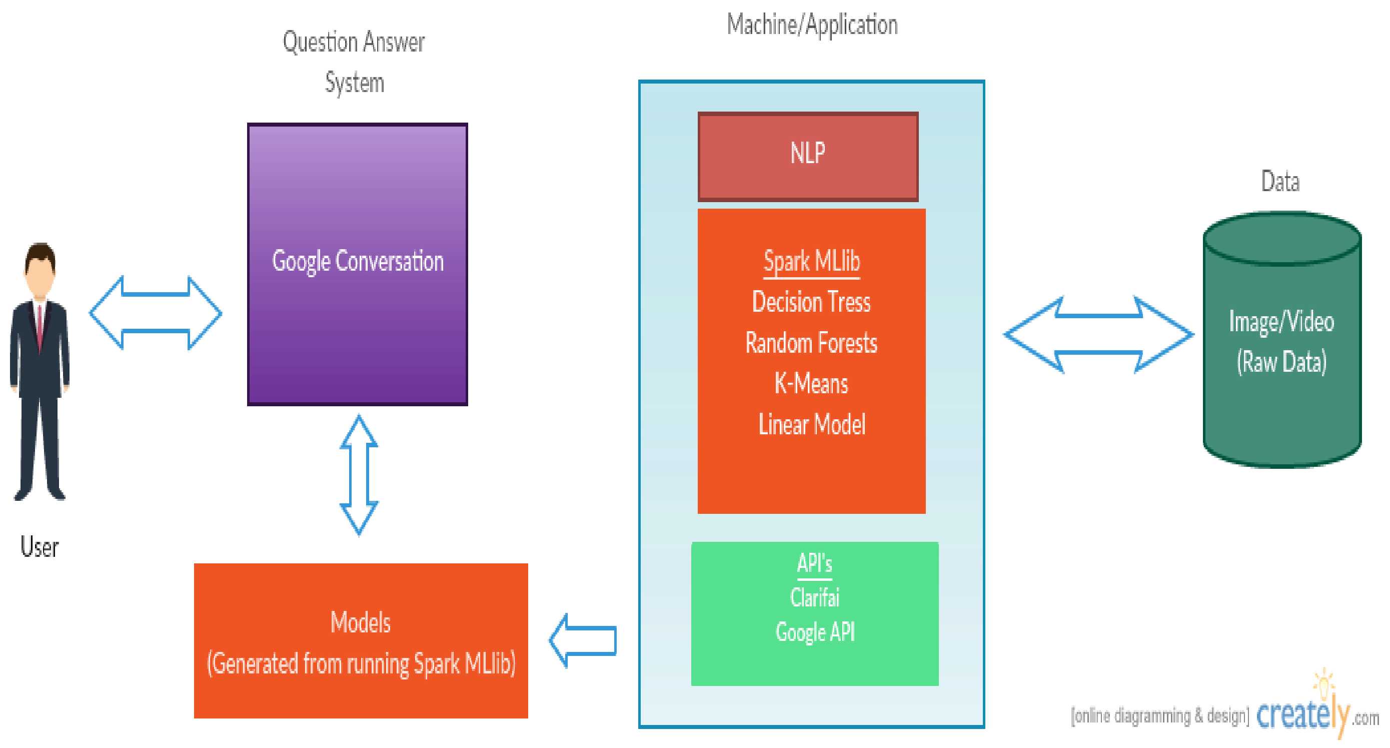
- *VoiceBot[8] - voice powered game control*
- *Modbox[9] - VR multiplayer sandbox game*
- *VRIO[10] - speech processing unit for VR and real-world scenarios*
- *VR Speech Synthesis[6]*

4. APPLICATION SPECIFICATION

4.1 System Specification

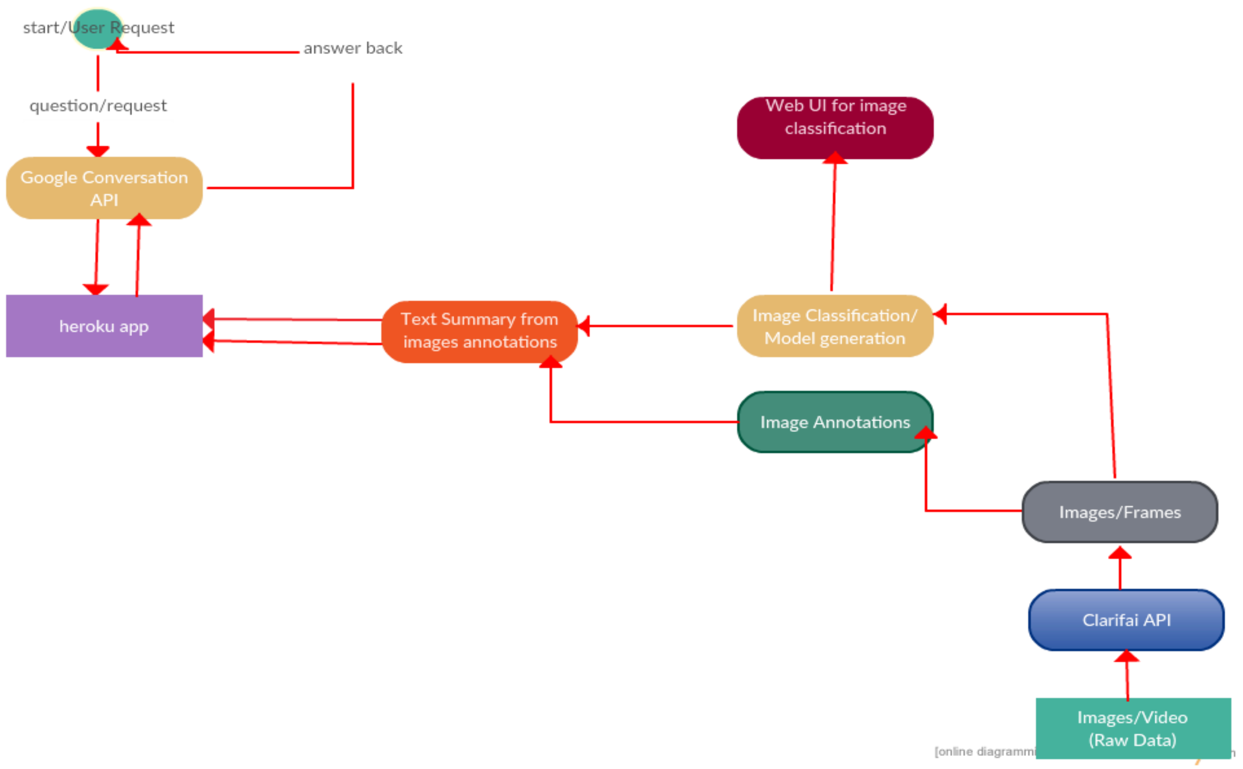
Operating System: Windows 10/ IOS
Memory: 8GB
Processor: intel i5 6th gen/ AMD

4.1.1 Software Architecture

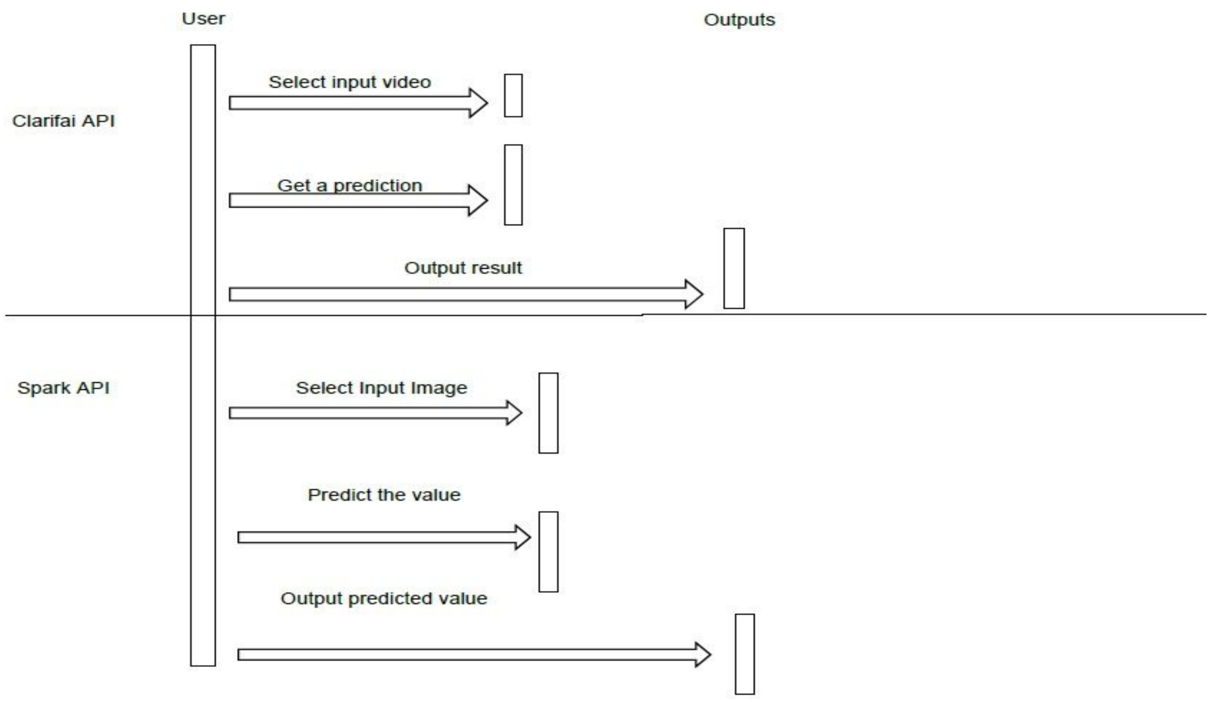


4.1.2 Features, workflow, technologies

- **Activity Diagram**



- **Sequence Diagram**



- **Feature specification**

1. Identify what kind of game it is by analyzing input video.
2. Identify what kind of game it is by analyzing input image using Decision tree and Random Forest Models.

- **Operation Specification**

1. Videos no more than 2MB in size.
2. Do not use huge dataset for training and testing.

4.1.3 Existing Applications/Services Used: Name, Description, UR

- Clarifai API
- Spark API

4.2 Machine Learning Algorithms/Models

4.2.1 Shallow Learning.

One of the important tasks in our project is image classification. For answering which type of the game it is first machine needs to read the understand the content it and should be able to differentiate it from other images. This can be achieved by implementing the Classification algorithms in Machine Learning. For this Project, we implemented Decision Tree and Random Forest Models in Spark MLlib for classification of sports images. The detailed implementation of these algorithms is explained in later sections (Section 5.4)

4.2.2 Deep Learning

In addition to the traditional shallow learning algorithms we also implemented deep learning for classification of the images. Deep learning is one of the hot topics lately in the machine learning. The results achieved through deep learning are so effective and accuracy of the prediction from the model is as close to 98%. We used TensorFlow library for generating a Deep learning model for image classification. In addition to our model we also used Google's Inception 3 model in order to acheive better results and higher accuracy.

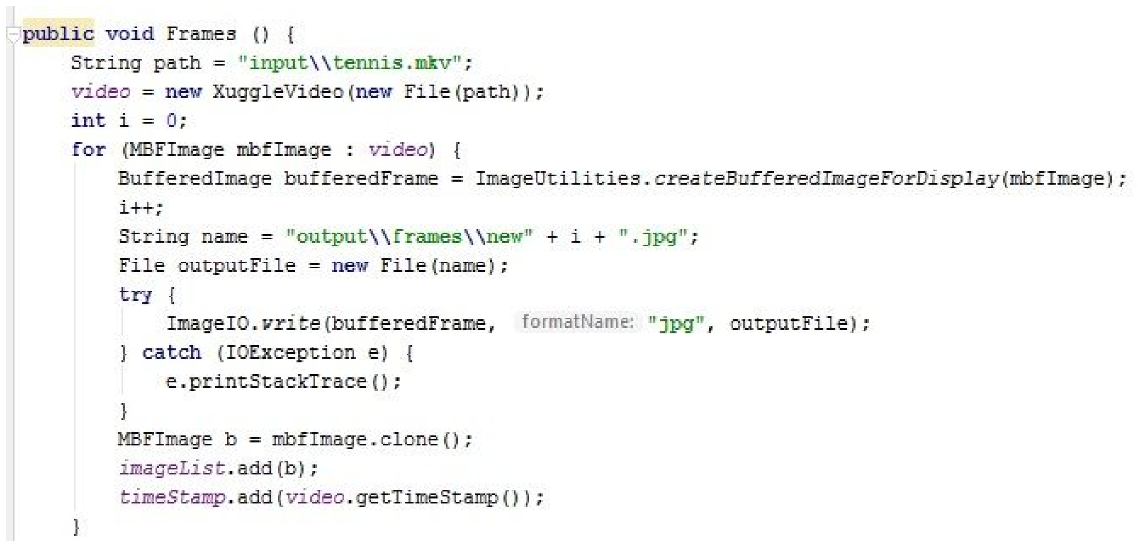
5. IMPLEMENTATION

5.1 Implementation of the application using Clarifai API

Clarifai APi offers video and image recognition. Clarifai API is built around a simple idea – you feed input file and Clarifai API service returns prediction. There are several models available in Clarifai API - for example food, travel, wedding, etc. In our project we are using the General Model.

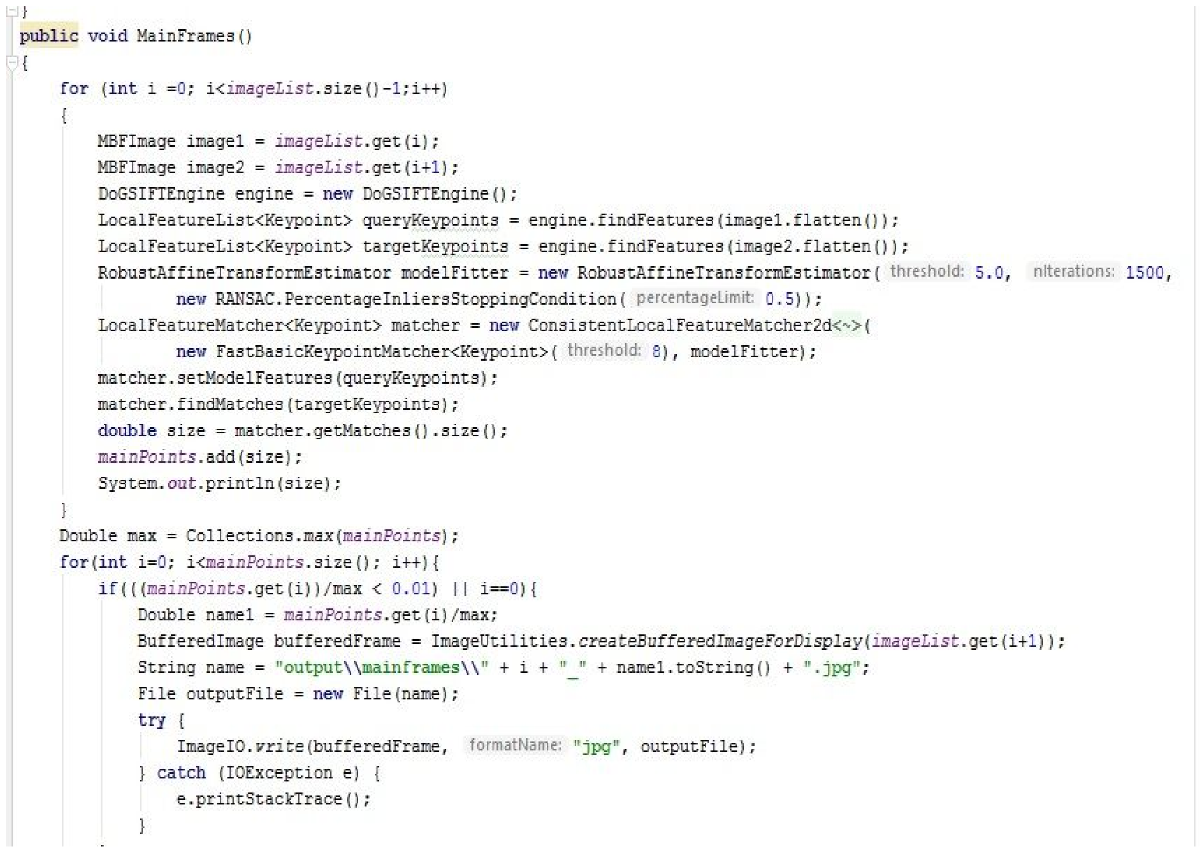
This part of the project predicts what type of sport game is played on the provided video file. Below are steps used to predict video game using external API:

1. Key Frames detection- take video file as input and iterate over video frames to detect key frames:



```
public void Frames () {
    String path = "input\\tennis.mkv";
    video = new XuggleVideo(new File(path));
    int i = 0;
    for (MBFImage mbfImage : video) {
        BufferedImage bufferedFrame = ImageUtilities.createBufferedImageForDisplay(mbfImage);
        i++;
        String name = "output\\frames\\new" + i + ".jpg";
        File outputFile = new File(name);
        try {
            ImageIO.write(bufferedFrame,  formatName: "jpg", outputFile);
        } catch (IOException e) {
            e.printStackTrace();
        }
        MBFImage b = mbfImage.clone();
        imageList.add(b);
        timeStamp.add(video.getTimeStamp());
    }
```

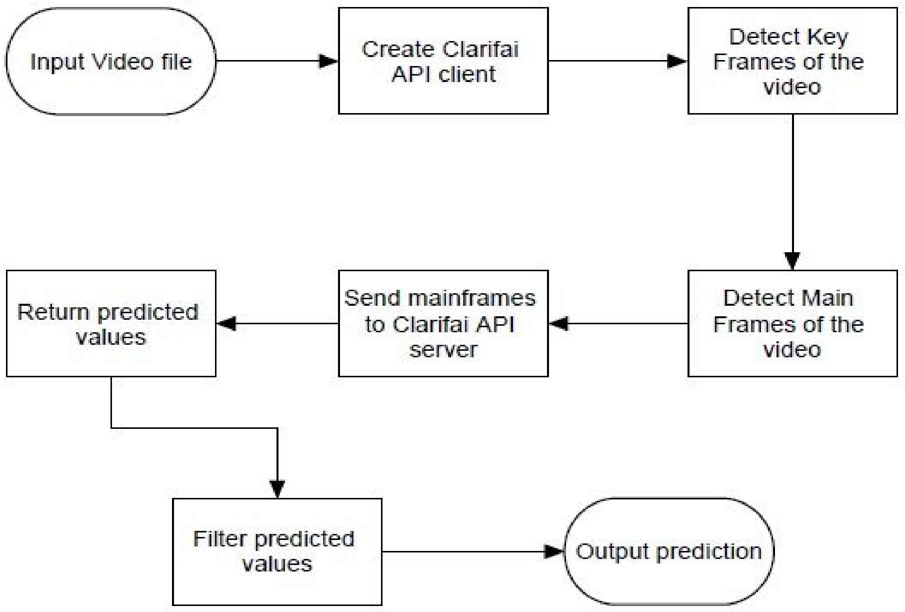
2. Detect main frames – after detecting key frames, main frames need to be pointed out so they could be uploaded to Clarifai API server to get predictions.



```
}
public void MainFrames()
{
    for (int i =0; i<imageList.size()-1;i++)
    {
        MBFImage image1 = imageList.get(i);
        MBFImage image2 = imageList.get(i+1);
        DoGSIFTEngine engine = new DoGSIFTEngine();
        LocalFeatureList<Keypoint> queryKeypoints = engine.findFeatures(image1.flatten());
        LocalFeatureList<Keypoint> targetKeypoints = engine.findFeatures(image2.flatten());
        RobustAffineTransformEstimator modelFitter = new RobustAffineTransformEstimator( threshold: 5.0,  nIterations: 1500,
                new RANSAC.PercentageInliersStoppingCondition( percentageLimit: 0.5));
        LocalFeatureMatcher<Keypoint> matcher = new ConsistentLocalFeatureMatcher2d<~>(
                new FastBasicKeypointMatcher<Keypoint>( threshold: 8), modelFitter);
        matcher.setModelFeatures(queryKeypoints);
        matcher.findMatches(targetKeypoints);
        double size = matcher.getMatches().size();
        mainPoints.add(size);
        System.out.println(size);
    }
    Double max = Collections.max(mainPoints);
    for(int i=0; i<mainPoints.size(); i++){
        if(((mainPoints.get(i))/max < 0.01) || i==0){
            Double name1 = mainPoints.get(i)/max;
            BufferedImage bufferedFrame = ImageUtilities.createBufferedImageForDisplay(imageList.get(i+1));
            String name = "output\\mainframes\\" + i + "_" + name1.toString() + ".jpg";
            File outputFile = new File(name);
            try {
                ImageIO.write(bufferedFrame,  formatName: "jpg", outputFile);
            } catch (IOException e) {
                e.printStackTrace();
            }
```

3. Annotation – after detection main frames and uploading them to Clarifai server for processing we get back predictions on what sport game is shown on the main frame. At this level we need to filter some of the predicted values since we are interested only in what sport is shown on the video. We

created the program which can predict the following sport games: soccer/football, tennis, basketball, volleyball, swimming, boxing, badminton and cricket. Other games can be added as needed.
Diagram:


Input Video file
Create Clarifai API client
Detect Key Frames of the video
Detect Main Frames of the video
Send mainframes to Clarifai API server
Return predicted values
Filter predicted values
Output prediction

5.2 Implementation of the application using Spark Client API
In the phase 1 of the project, we implemented a simple application using a spark Client API. This application , given an image will identify what kind of sport it is. We considered five different categories of sports as five different classes while training the model. Cricket, BasketBall, Swimming, Boxing and Tennis are the five sport categories used for this application. We collected large amount of data set(Sports images) and categorized them into five different classes with respect to the sport the belong and used decision tree model to generate the model for image classification. SIFT features of the images are used for implementing the classification algorithm. Then, we used this model from server end to test an image from client side. Basically, We implemented the following architecture shown in the figure.

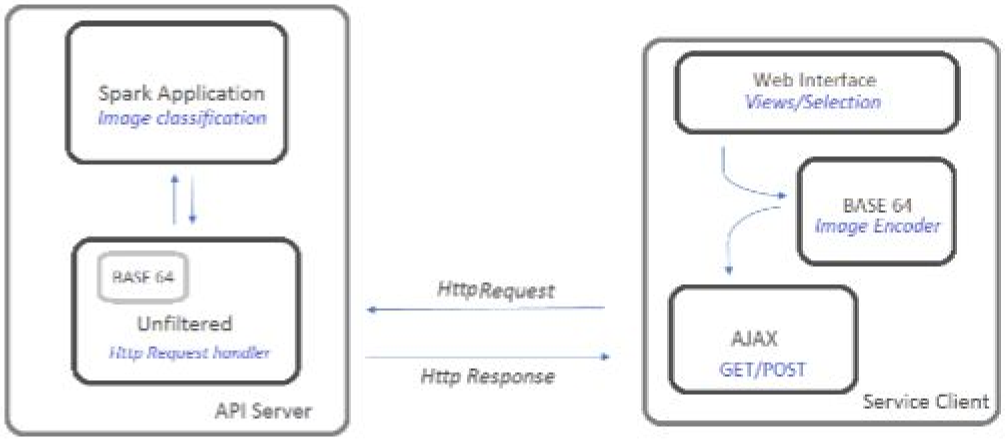
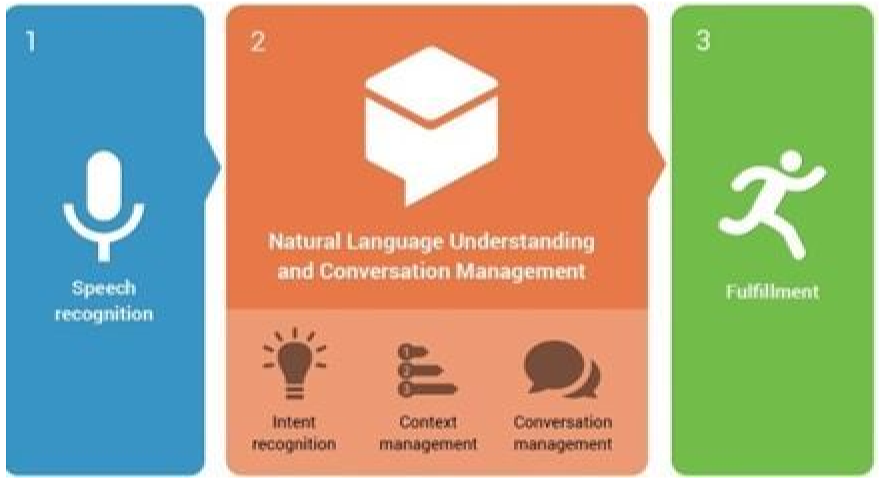


Fig. Architecture of Spark Client API.

A client can request a service from the server using GET and POST calls for sending an image to the server and to get a response of the prediction of the Image. Once the model gets an image it predicts the image sports category using SIFT features and displays the result at the client end.
We, also considered Random Forest algorithm for generating the model for image classification expecting it would fetch more accuracy than Decision tree. On Contrary Decision tree model has more Accuracy of 84% than Random forest which has around 72%. Screenshots of the results are included in the documentation.

5.3 Implementation of the application using Google Conversation API



[15]API.AI is a natural language understanding platform that makes it easy for developers (and non-developers) to design and integrate intelligent and sophisticated conversational user interfaces into mobile apps, web applications, devices, and bots.

We used api.api to create an agent VoiceInteractiveISE. We have created intent and entities as part of the conversation. We have given the summary of the video and several other data as inputs and trained the agent.

5.4 Implementation of Decision Tree And Random Forest

Image classification is one of the important task in our project. The major algorithms implemented are Decision Tree and Random Forest,. We used Spark MLlib to implement these algorithms on our dataset.

Training the model:

We collected large amount of dataset basically sports images. We classified these images into 5 classes each class specifying a specific sport. And trained the model using both Decision Tree and Random Forest algorithms. The five Classes classes of sports are:

1. Basket Ball
2. Boxing
3. Cricket
4. Swimming
5. Tennis

Testing the model and Calculating the Accuracy:

We tested the models we generated in the training phase on new set of images. By comparing the results from both the models, we observed that Decision tree model has higher accuracy rate than Random forest model for Image Classification. The Decision Tree model has recorded around 86% accuracy rate whereas the Random Forest model has achieved around 77% accuracy.

5.5. Implementation of application using Tensorflow

In this phase we needed to implement application for video annotation using Tensorflow. We began with image classification. First we needed to find the dataset which contains images related to sport, however we could not find it, so instead of it we tried to use Cifar-10 dataset. We are still working on our model. Our overall plan is:

Get a dataset of spot images and divide them into batches, first we need to process images and then use CNN on all the samples. The images need to be normalized and the labels need to be one-hot encoded. Also will have to build a convolutional, max pooling, dropout, and fully connected layers. At the end, we should see your neural network's predictions on the sample images.

· Explore the data

We are dividing dataset into batches which will prevent our machine from running out of memory. Each batch should contains the labels and images.

· Normalization

Normalization function should take in image data, x, and return it as a normalized Numpy array. The values should be in the range from 0 to 1, inclusive. The return object should be the same shape as x.

· Randomize the data

In this phase we need to randomize the data.

· Process all data and save it

· Checkpoints

This will be our first checkpoint.

· Built the network

For the neural network, we should built each layer into a function. To test the code more thoroughly, we require that to put each layer in a function. This allows us to give better feedback and test for simple mistakes using our unit tests.

· Input

The neural network needs to read the image data, one-hot encoded labels, and dropout keep probability.

· Convolutional and max pulling layer

We need to create a weight, apply convolution to x_tensor using the weight, add bias, add a nonlinear activation to the convolution, apply Max Pooling.

· Flatten layer

Implement the fully_conn function to apply a fully connected layer to x_tensor with the shape (*Batch Size*, *num_outputs*). We can use TensorFlow Layers or TensorFlow Layers (contrib) for this layer.

· Output layer

Implement the output function to apply a fully connected layer to x_tensor with the shape (*Batch Size*, *num_outputs*). We can use TensorFlow Layers or TensorFlow Layers (contrib) for this layer.

· Create convolutional layer

Implement the function conv_net to create a convolutional neural network model. The function takes in a batch of images, x, and outputs logits. Should use the layers that we created above to create this model: Apply 1, 2, or 3 Convolution and Max Pool layers, Apply a Flatten Layer, Apply 1, 2, or 3 Fully , connected Layers, Apply an Output Layer, Return the output, Apply TensorFlow's Dropout to one or more layers in the model using keep_prob.

· Train Neural network

Implement the function train_neural_network to do a single optimization. The optimization should use optimizer to optimize in session with a feed_dict of the following: x for image input, y for labels, keep_prob for keep probability for dropout. This function will be called for each batch, so tf.global_variables_initializer() has already been called.

· Show stats

Implement the function print_stats to print loss and validation accuracy. Should use the global variables valid_features and valid_labels to calculate validation accuracy. Also should use a keep probability of 1.0 to calculate the loss and validation accuracy.

· Fully train the model

Train all batches.

· Test model

Test our model against the test dataset. This will be our final accuracy. We should have accuracy greater than 50%.

5.6 Implementation of Google Inception for Image Classification

For this phase we used transfer learning i.e., implementing the model that has already been generated for another model. Google has created a classification model called Inception V3 which is trained on millions of images for about 2 weeks. This model also ses TensorFlow library. This model can classify the images into about 1000 classes. We used the same network by limiting the number classes to 5 based on our own data set. Our model can classify an image into one of the 5 sports categories.

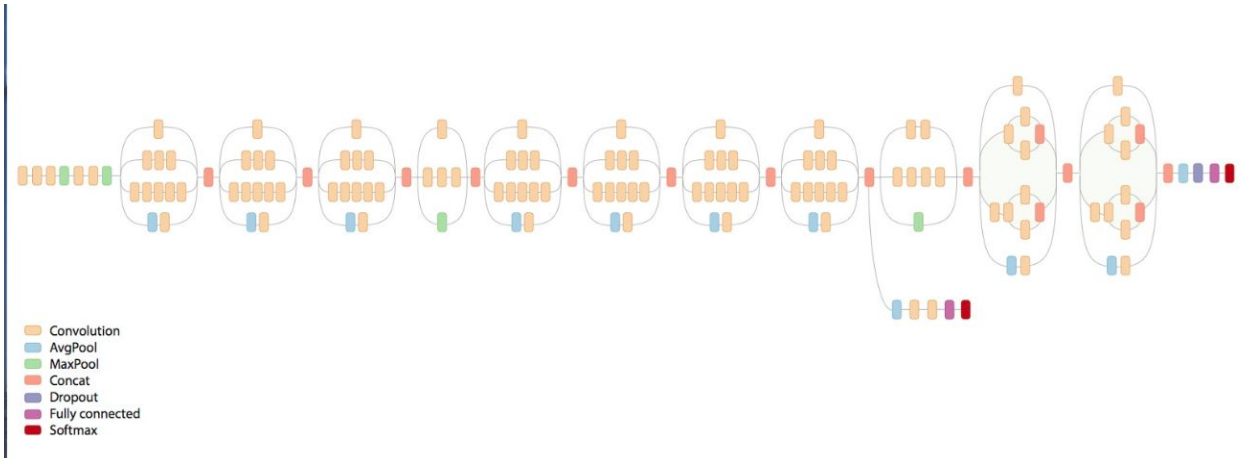


Fig. Convolutional Neural Net for Inception V3.

This network used used by Google for building Inception V3. First part involves extracting the features from the images in the training data and second part classifies the image based on those features. In this transfer learning we built a new model to classify our own data set by using the feature extraction part and training part from inception V3 own our own data. We collected about 100 images and classified them into 5 different categories such as Cricket, BasketBall, Swimming, Boxing and Tennis. Once the model is generated after the training part we tested the model on a new sports image and the results were very effective.

5.6 Implementation of Google CardBoard API.

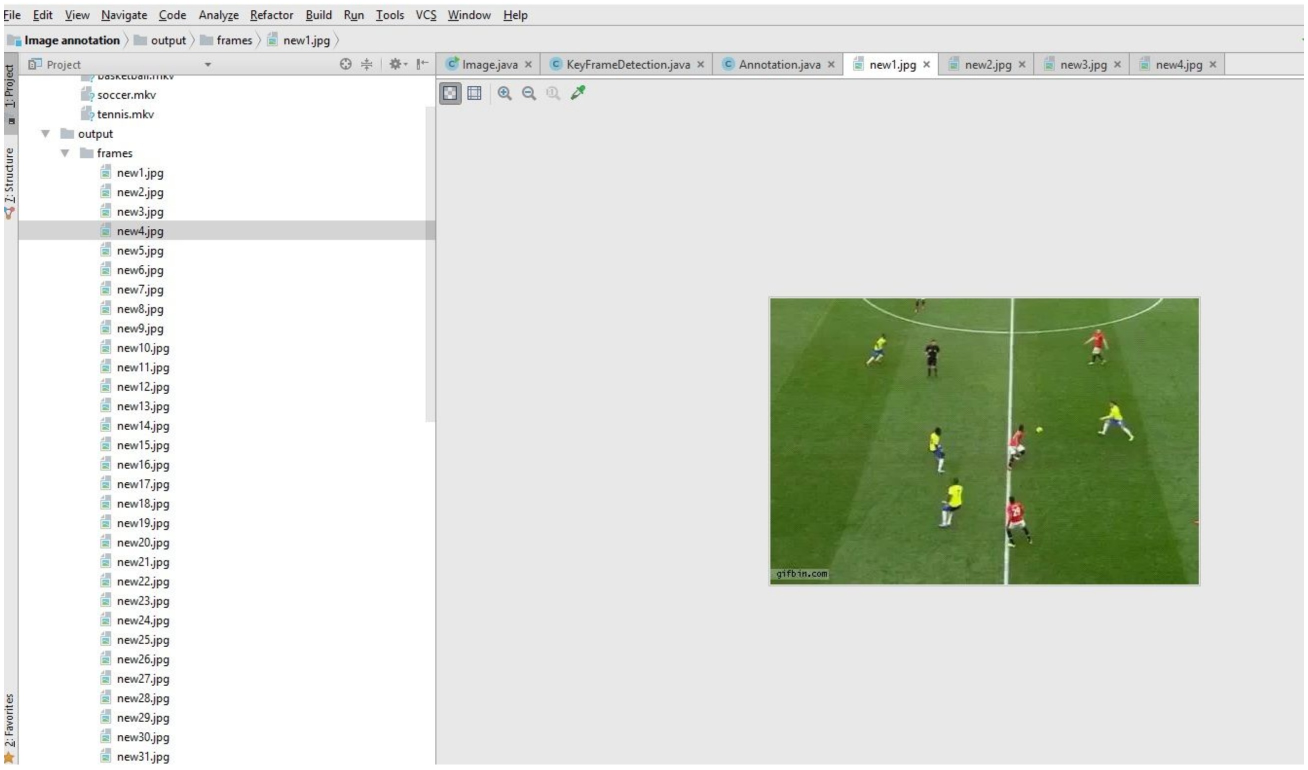
The google cardboard API is implemented taking the references from google's own github gvr-android-sdk. The google cardboard API app is developed as an application to view sports videos with a 360 degree view on google cardboard. The cardboard app provides an application interface to view a 360 degree video in google carboard VR mode. More on this application, the screenshots are provided in the documentation part of the report. This cardboard app, in the future can use for the 360 degree video annotations. The annotations for 360 degree video may be a possibility in the coming phases.

6. DOCUMENTATION

6.1 Phase1

- **Screenshots of application using clarifai API**

Detected key Frames:



Detected MainFrames:

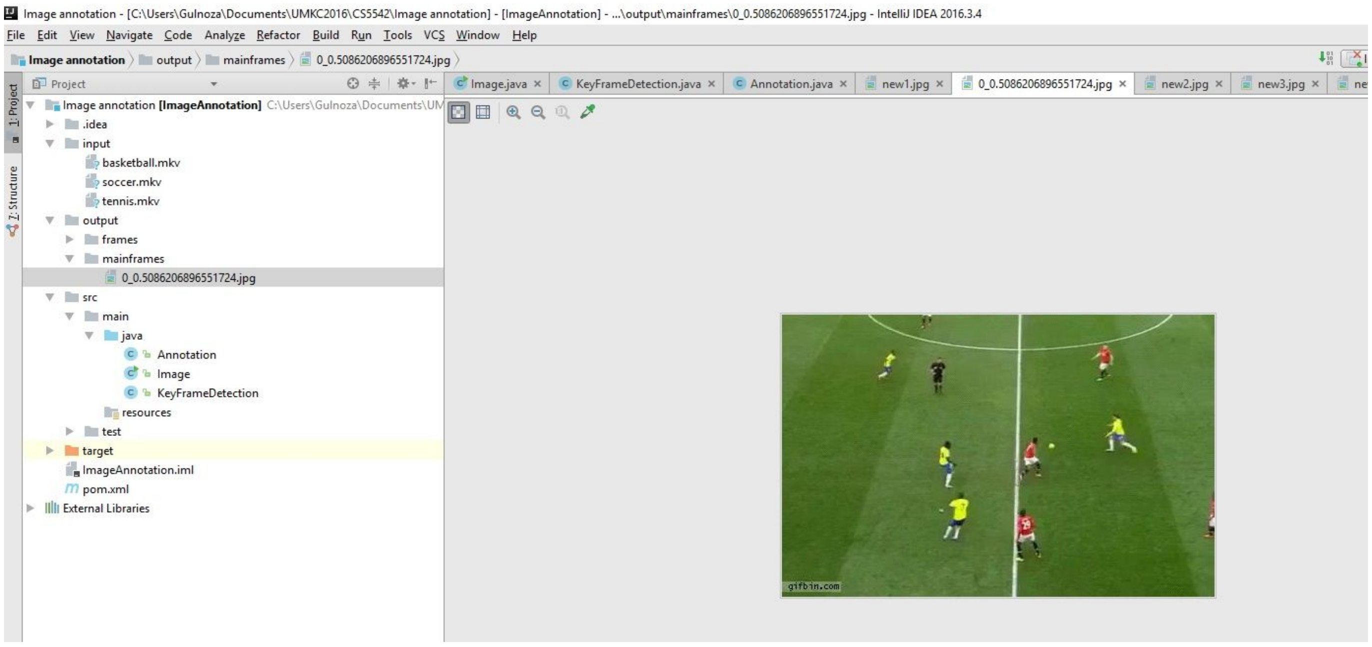
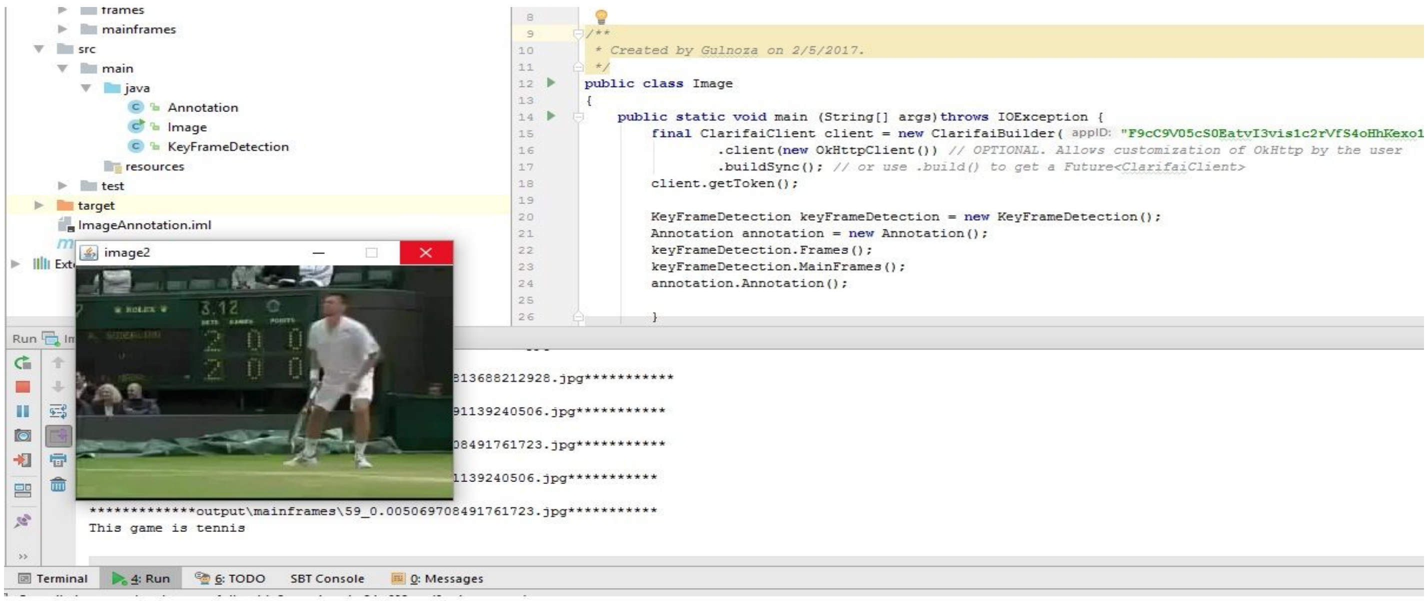


Image annotation - [C:\Users\Gulnoza\Documents\UMKC2016\CS5542\Image annotation] - [ImageAnnotation] - ...\output\mainframes\0_0.5086206896551724.jpg - IntelliJ IDEA 2016.3.4
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Image annotation > output > mainframes > 0_0.5086206896551724.jpg
Project
Image.java
KeyFrameDetection.java
Annotation.java
new1.jpg
0_0.5086206896551724.jpg
new2.jpg
new3.jpg
Image annotation [ImageAnnotation] C:\Users\Gulnoza\Documents\UM
.idea
input
basketball.mkv
soccer.mkv
tennis.mkv
output
frames
mainframes
0_0.5086206896551724.jpg
src
main
java
Annotation
Image
KeyFrameDetection
resources
test
target
ImageAnnotation.iml
pom.xml
External Libraries
1: Project
7: Structure
gifbin.com

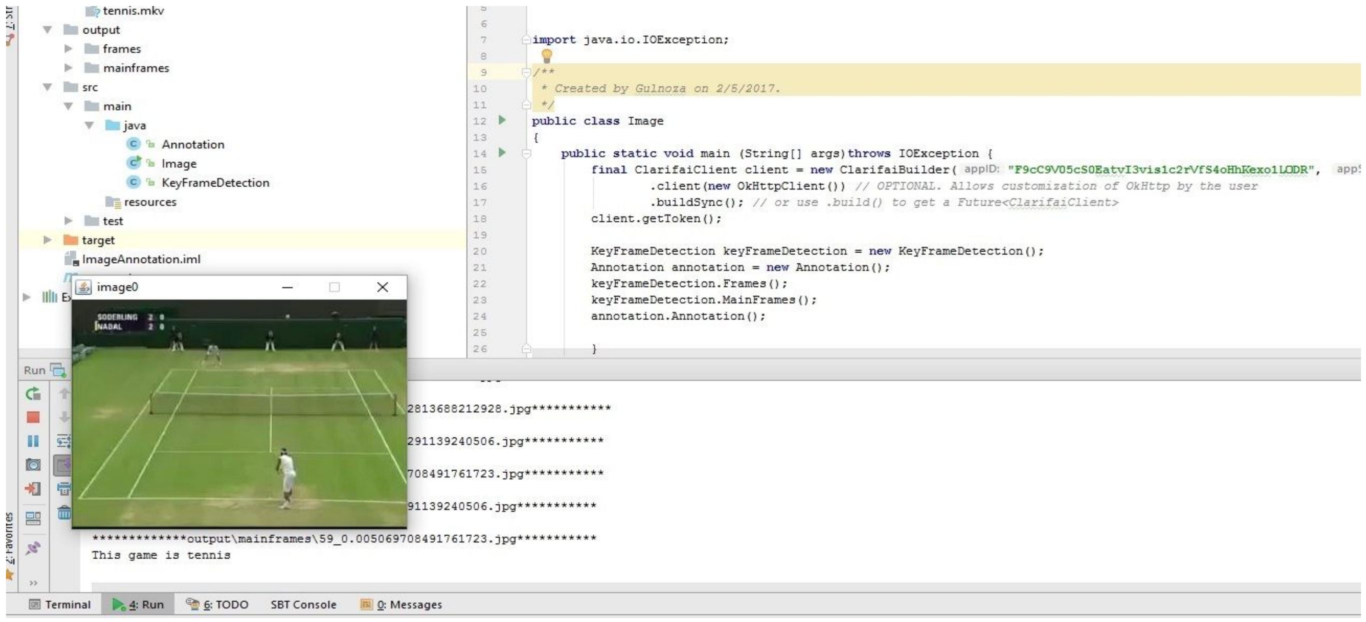
Predicting types of games:

Tennis:



```
/**
 * Created by Gulnoza on 2/5/2017.
 */
public class Image
{
    public static void main (String[] args)throws IOException {
        final ClarifaiClient client = new ClarifaiBuilder( appID: "F9cC9V05cS0EatvI3vis1c2rVfS4oHhKexo1
                .client(new OkHttpClient()) // OPTIONAL. Allows customization of OkHttp by the user
                .buildSync(); // or use .build() to get a Future<ClarifaiClient>
        client.getToken();

        KeyFrameDetection keyFrameDetection = new KeyFrameDetection();
        Annotation annotation = new Annotation();
        keyFrameDetection.Frames();
        keyFrameDetection.MainFrames();
        annotation.Annotation();

    }
```

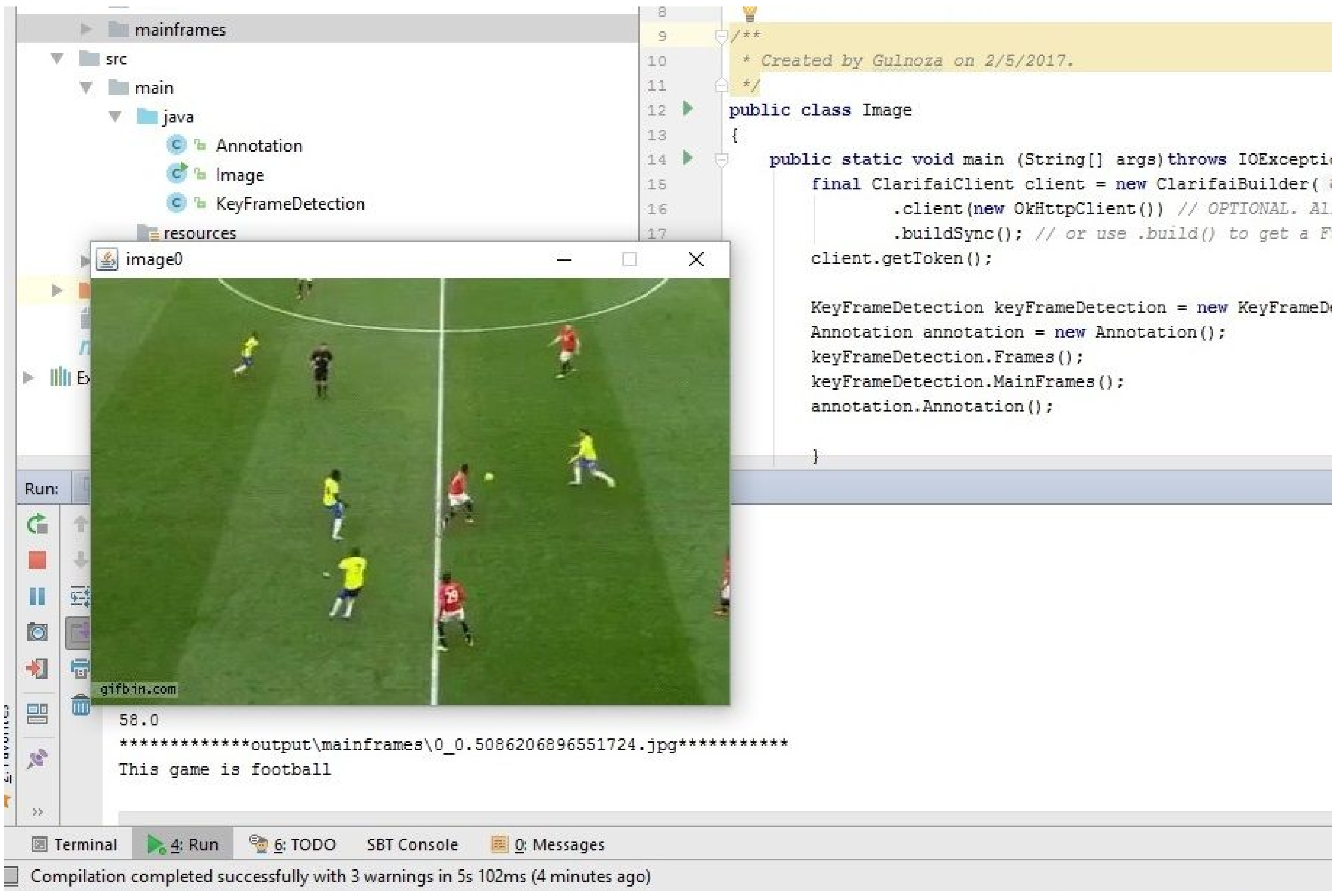
```
813688212928.jpg***********

91139240506.jpg***********

08491761723.jpg***********

1139240506.jpg***********

*************output\mainframes\59_0.005069708491761723.jpg***********
This game is tennis
```



```
import java.io.IOException;

/**
 * Created by Gulnoza on 2/5/2017.
 */
public class Image
{
    public static void main (String[] args)throws IOException {
        final ClarifaiClient client = new ClarifaiBuilder( appID: "F9cC9V05cS0EatvI3vis1c2rVfS4oHhKexo1LODR",  app
                .client(new OkHttpClient()) // OPTIONAL. Allows customization of OkHttp by the user
                .buildSync(); // or use .build() to get a Future<ClarifaiClient>
        client.getToken();

        KeyFrameDetection keyFrameDetection = new KeyFrameDetection();
        Annotation annotation = new Annotation();
        keyFrameDetection.Frames();
        keyFrameDetection.MainFrames();
        annotation.Annotation();

    }
```

```
2813688212928.jpg***********

291139240506.jpg***********

708491761723.jpg***********

91139240506.jpg***********

*************output\mainframes\59_0.005069708491761723.jpg***********
This game is tennis
```

Soccer/football:

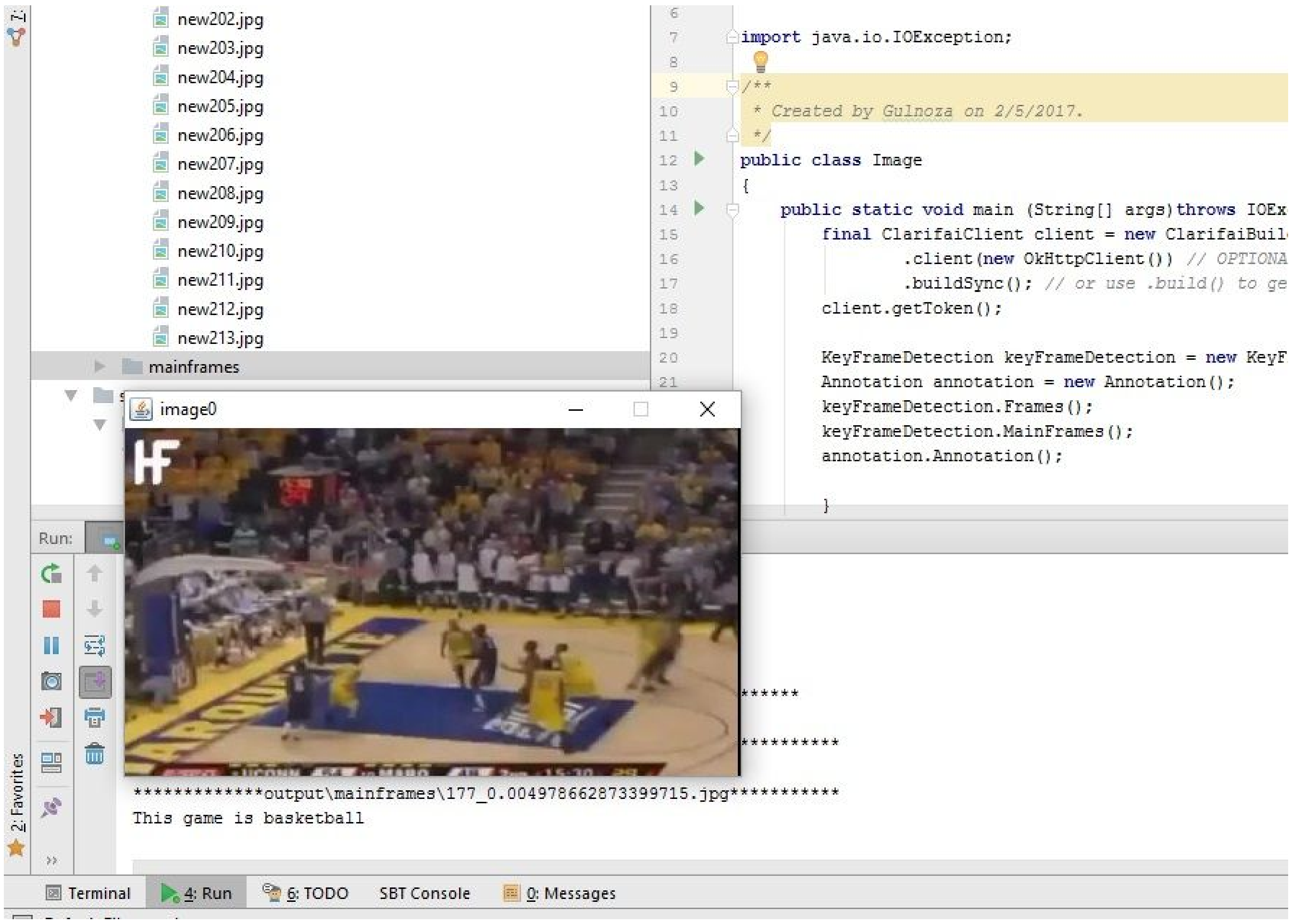


```
/**
 * Created by Gulnoza on 2/5/2017.
 */
public class Image
{
    public static void main (String[] args)throws IOExcepti
        final ClarifaiClient client = new ClarifaiBuilder(
                .client(new OkHttpClient()) // OPTIONAL. Al
                .buildSync(); // or use .build() to get a F
        client.getToken();

        KeyFrameDetection keyFrameDetection = new KeyFrameD
        Annotation annotation = new Annotation();
        keyFrameDetection.Frames();
        keyFrameDetection.MainFrames();
        annotation.Annotation();

        }
```

```
58.0
*************output\mainframes\0_0.5086206896551724.jpg***********
This game is football
```

Compilation completed successfully with 3 warnings in 5s 102ms (4 minutes ago)

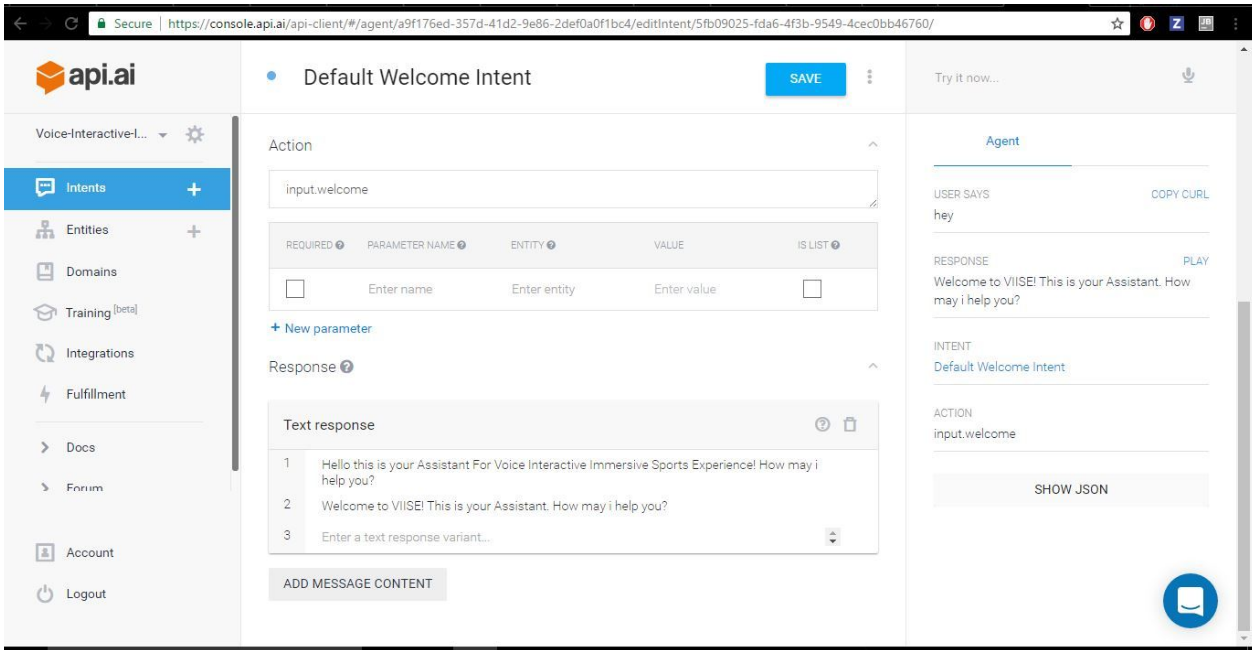
Basketball:



```
import java.io.IOException;

/**
 * Created by Gulnoza on 2/5/2017.
 */
public class Image
{
    public static void main (String[] args)throws IOEx
        final ClarifaiClient client = new ClarifaiBuil
                .client(new OkHttpClient()) // OPTIONA
                .buildSync(); // or use .build() to ge
        client.getToken();

        KeyFrameDetection keyFrameDetection = new KeyF
        Annotation annotation = new Annotation();
        keyFrameDetection.Frames();
        keyFrameDetection.MainFrames();
        annotation.Annotation();

        }
```

```
******
**********
*************output\mainframes\177_0.004978662873399715.jpg***********
This game is basketball
```

Screenshots of Google Conversation API:



Secure | https://console.api.ai/api-client/#/agent/a9f176ed-357d-41d2-9e86-2def0a0f1bc4/editIntent/5fb09025-fda6-4f3b-9549-4cec0bb46760/

Default Welcome Intent

Action

input.welcome

Response

Text response

1 Hello this is your Assistant For Voice Interactive Immersive Sports Experience! How may i help you?

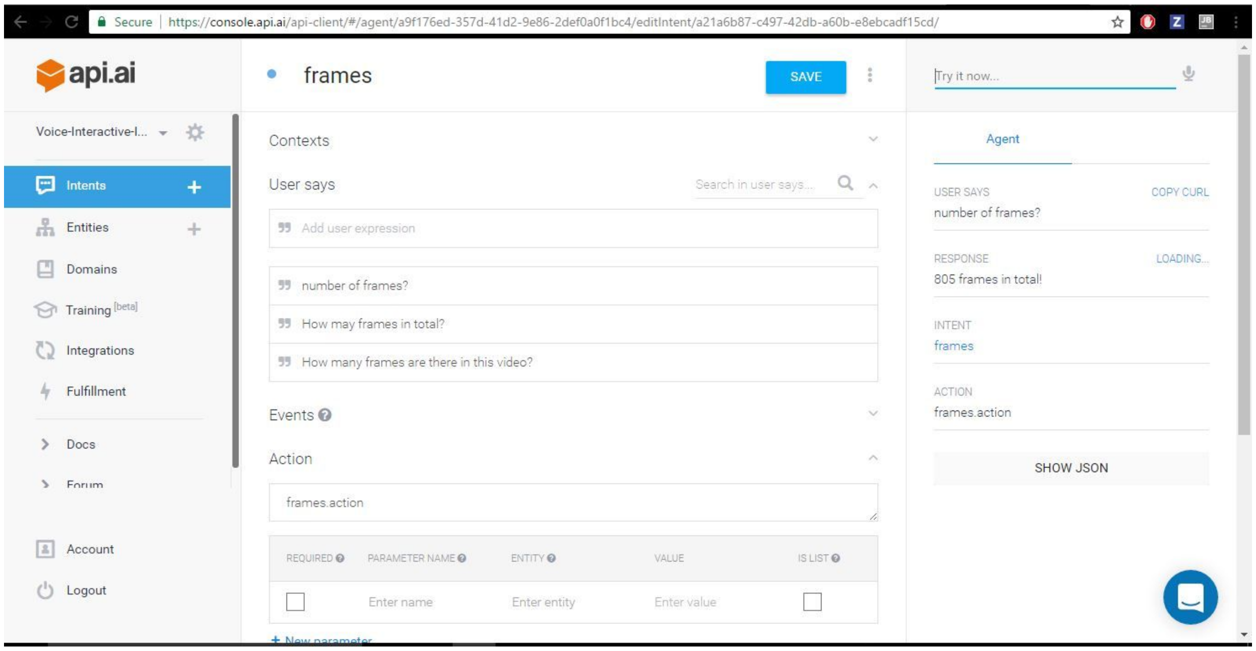
2 Welcome to VIISE! This is your Assistant. How may i help you?

USER SAYS
hey

RESPONSE
Welcome to VIISE! This is your Assistant. How may i help you?

INTENT
Default Welcome Intent

ACTION
input.welcome



Secure | https://console.api.ai/api-client/#/agent/a9f176ed-357d-41d2-9e86-2def0a0f1bc4/editIntent/a21a6b87-c497-42db-a60b-e8ebcadf15cd/

frames

Contexts

User says

number of frames?

How may frames in total?

How many frames are there in this video?

Events

Action

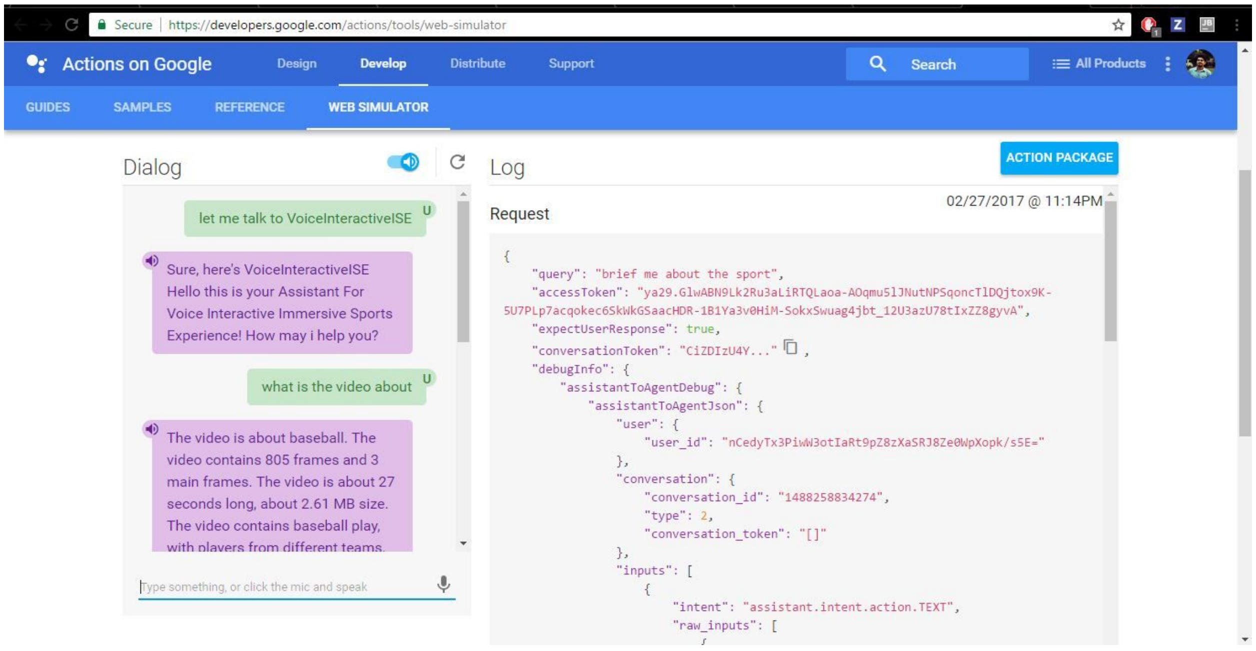
frames.action

USER SAYS
number of frames?

RESPONSE
805 frames in total!

INTENT
frames

ACTION
frames.action



let me talk to VoiceInteractiveISE

Sure, here's VoiceInteractiveISE
Hello this is your Assistant For Voice Interactive Immersive Sports Experience! How may i help you?

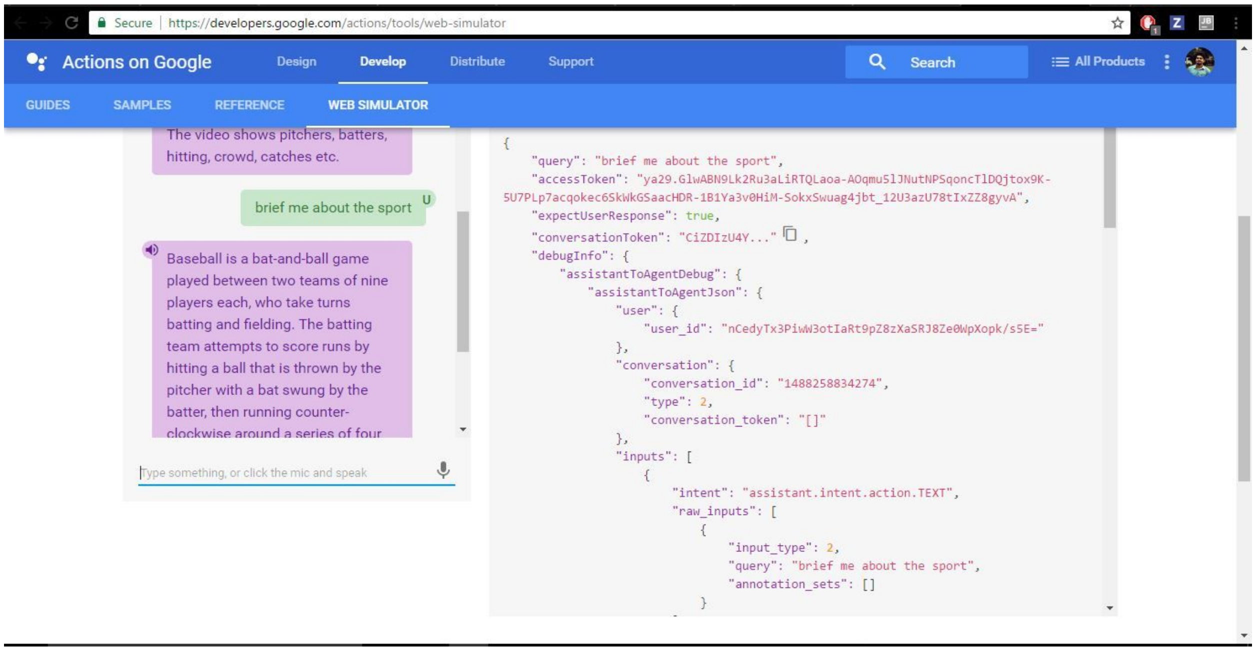
what is the video about

The video is about baseball. The video contains 805 frames and 3 main frames. The video is about 27 seconds long, about 2.61 MB size. The video contains baseball play, with players from different teams

Request

02/27/2017 @ 11:14PM

```
{
    "query": "brief me about the sport",
    "accessToken": "ya29.GlwABN9Lk2Ru3aLiRTQLaoa-AOqmu5lJNutNPSqoncTlDQjtox9K-
5U7PLp7acqokec6SkWkGSaacHDR-1B1Ya3v0HiM-SokxSwuag4jbt_12U3azU78tIxZZ8gyvA",
    "expectUserResponse": true,
    "conversationToken": "CiZDIzU4Y..." ,
    "debugInfo": {
        "assistantToAgentDebug": {
            "assistantToAgentJson": {
                "user": {
                    "user_id": "nCedyTx3PiwW3otIaRt9pZ8zXaSRJ8Ze0WpXopk/s5E="
                },
                "conversation": {
                    "conversation_id": "1488258834274",
                    "type": 2,
                    "conversation_token": "[]"
                },
                "inputs": [
                    {
                        "intent": "assistant.intent.action.TEXT",
                        "raw_inputs": [
```

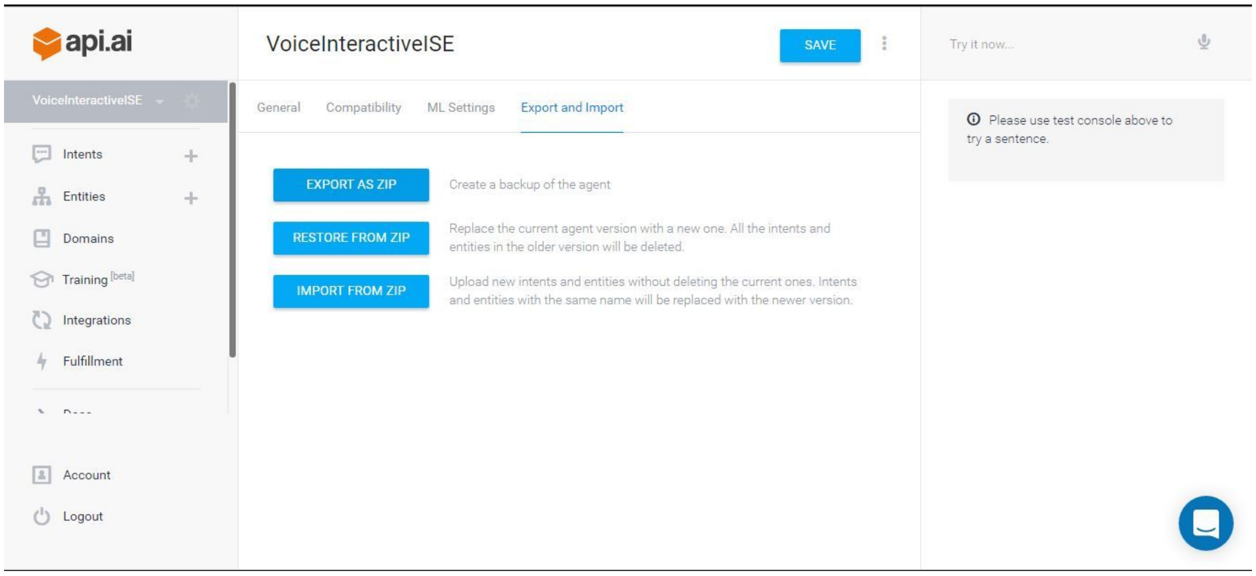

Secure | https://developers.google.com/actions/tools/web-simulator
Actions on Google
Design
Develop
Distribute
Support
Search
All Products
GUIDES
SAMPLES
REFERENCE
WEB SIMULATOR
Type something, or click the mic and speak

The video shows pitchers, batters, hitting, crowd, catches etc.

brief me about the sport

Baseball is a bat-and-ball game played between two teams of nine players each, who take turns batting and fielding. The batting team attempts to score runs by hitting a ball that is thrown by the pitcher with a bat swung by the batter, then running counter-clockwise around a series of four

```
{
    "query": "brief me about the sport",
    "accessToken": "ya29.GlwABN9Lk2Ru3aLiRTQLaoa-AOqmu5lJNutNPSqoncTlDQjtox9K-
5U7PLp7acqokec6SkWkGSaacHDR-1B1Ya3v0HiM-SokxSwuag4jbt_12U3azU78tIxZZ8gyvA",
    "expectUserResponse": true,
    "conversationToken": "CiZDIzU4Y..." ,
    "debugInfo": {
        "assistantToAgentDebug": {
            "assistantToAgentJson": {
                "user": {
                    "user_id": "nCedyTx3PiwW3otIaRt9pZ8zXaSRJ8Ze0WpXopk/s5E="
                },
                "conversation": {
                    "conversation_id": "1488258834274",
                    "type": 2,
                    "conversation_token": "[]"
                },
                "inputs": [
                    {
                        "intent": "assistant.intent.action.TEXT",
                        "raw_inputs": [
                            {
                                "input_type": 2,
                                "query": "brief me about the sport",
                                "annotation_sets": []
                            }
```

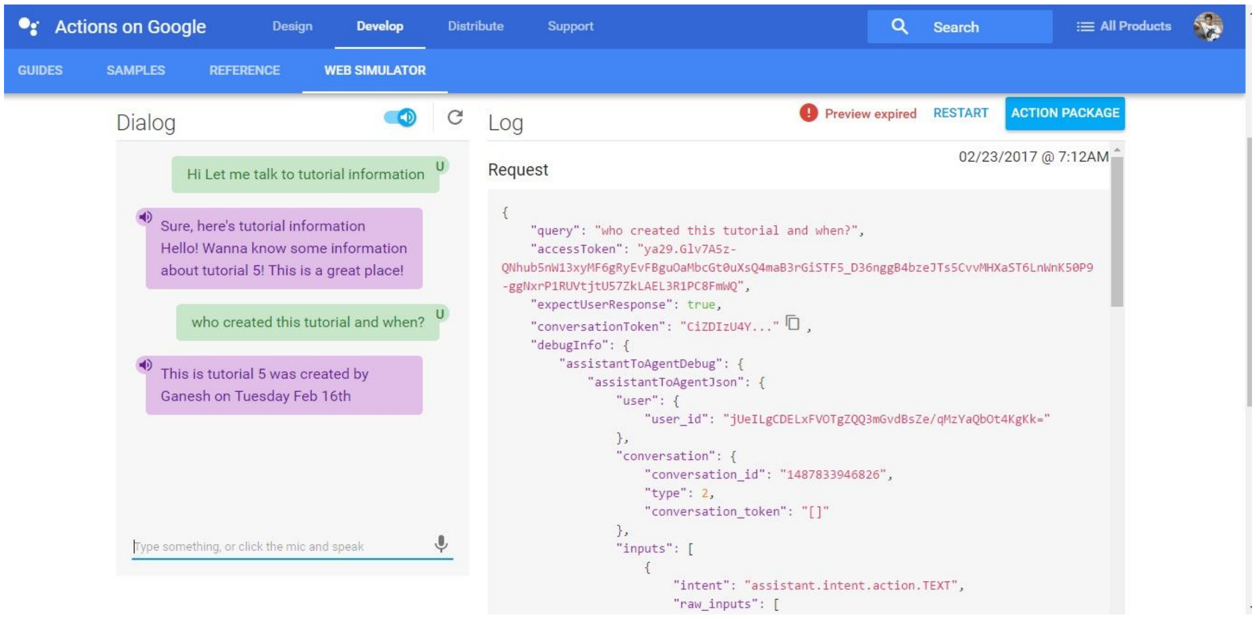


Create a backup of the agent

Replace the current agent version with a new one. All the intents and entities in the older version will be deleted.

Upload new intents and entities without deleting the current ones. Intents and entities with the same name will be replaced with the newer version.

Please use test console above to try a sentence.



Hi Let me talk to tutorial information

Sure, here's tutorial information Hello! Wanna know some information about tutorial 5! This is a great place!

who created this tutorial and when?

This is tutorial 5 was created by Ganesh on Tuesday Feb 16th

Request 02/23/2017 @ 7:12AM

```
{
    "query": "who created this tutorial and when?",
    "accessToken": "ya29.Glv7A5z-
QNhub5nW13xyMF6gRyEvFBguOaMbcGt0uXsQ4maB3rGiSTF5_D36nggB4bzeJTs5CvvMHXaST6LnWnK50P9
-ggNxrP1RUVtjtU57ZkLAEL3R1PC8FmWQ",
    "expectUserResponse": true,
    "conversationToken": "CiZDIzU4Y..." ,
    "debugInfo": {
        "assistantToAgentDebug": {
            "assistantToAgentJson": {
                "user": {
                    "user_id": "jUeILgCDELxFVOTgZQQ3mGvdBsZe/qMzYaQbOt4KgKk="
                },
                "conversation": {
                    "conversation_id": "1487833946826",
                    "type": 2,
                    "conversation_token": "[]"
                },
                "inputs": [
                    {
                        "intent": "assistant.intent.action.TEXT",
                        "raw_inputs": [
```

Screenshots of application using Spark Client API

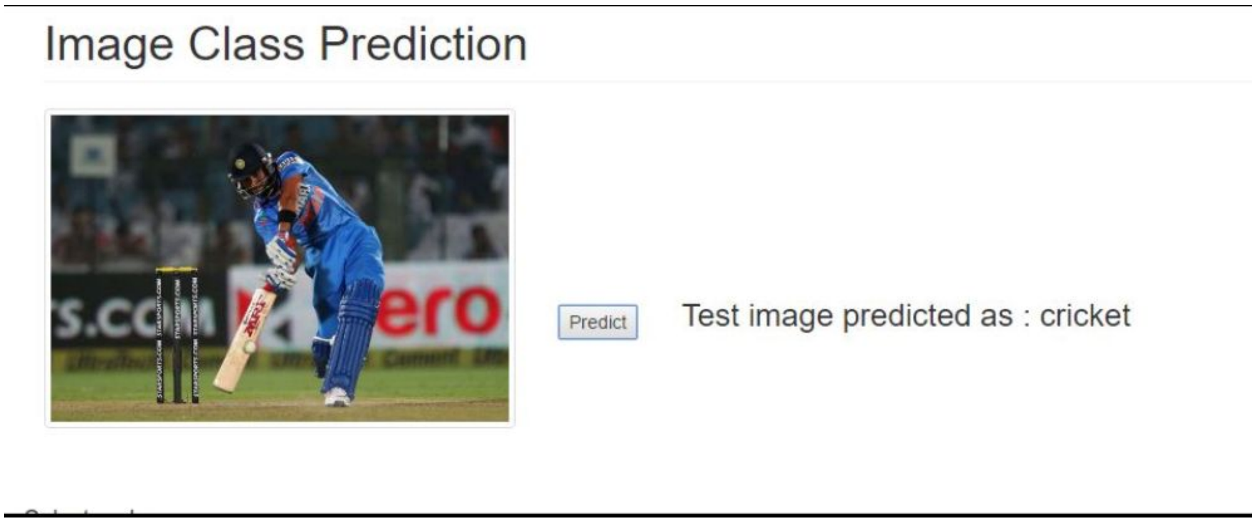


Image Class Prediction

Test image predicted as : cricket

Fig: Model predicted the cricket image as cricket

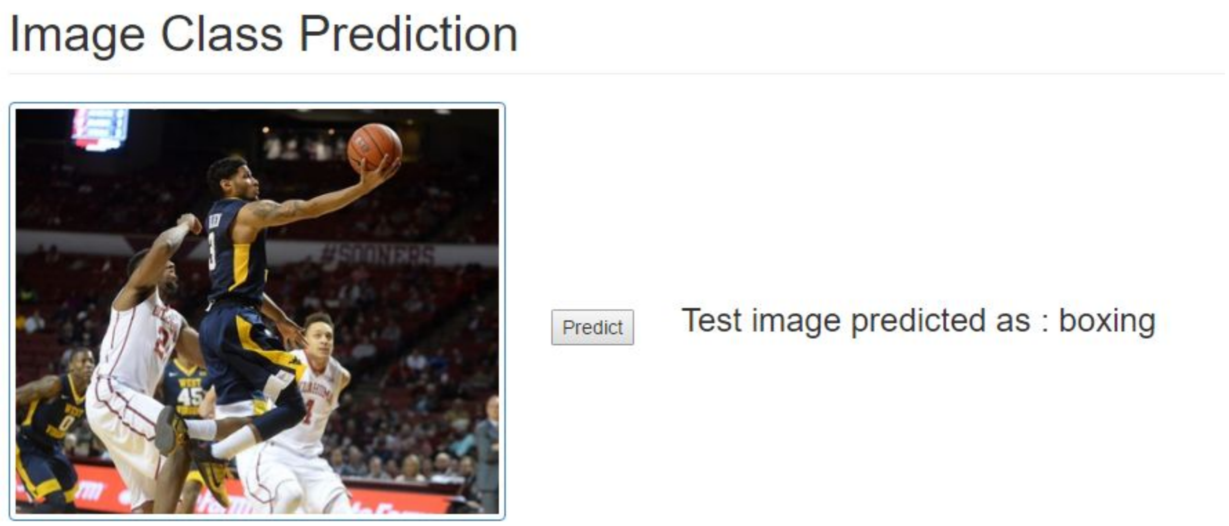


Image Class Prediction

Test image predicted as : boxing

Fig: model predicted the basketball as Boxing

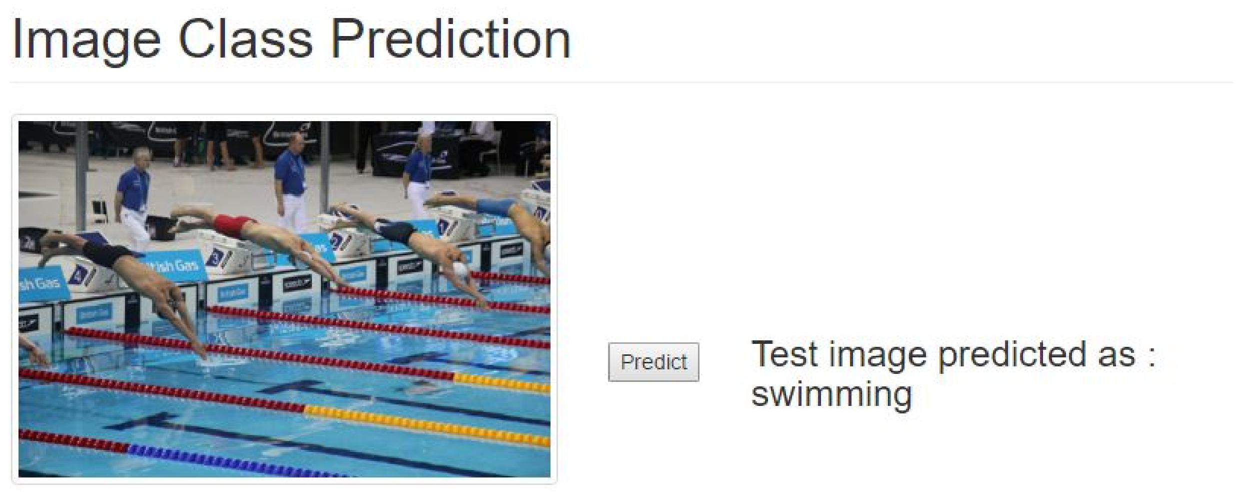


Image Class Prediction

Test image predicted as : swimming

Fig: model predicting swimming as swimming

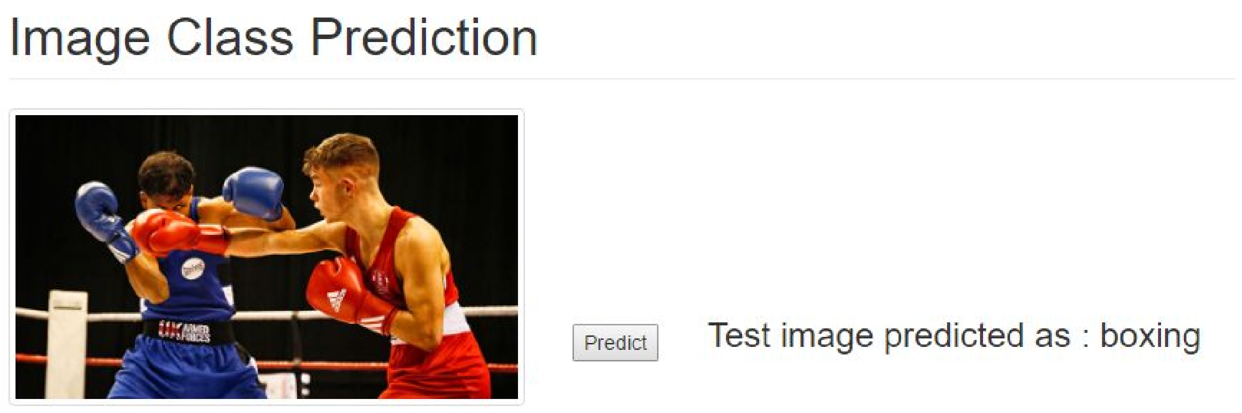
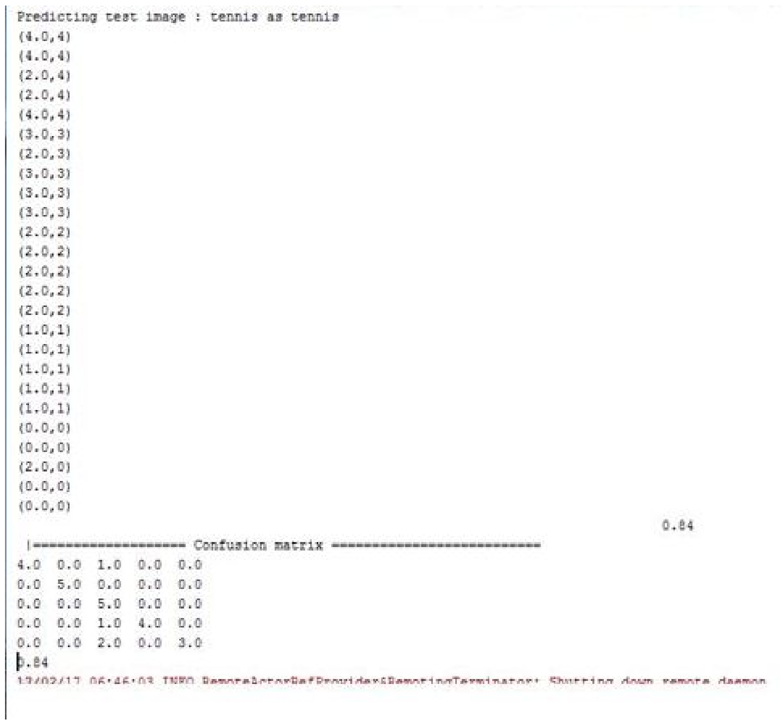


Image Class Prediction

Test image predicted as : boxing

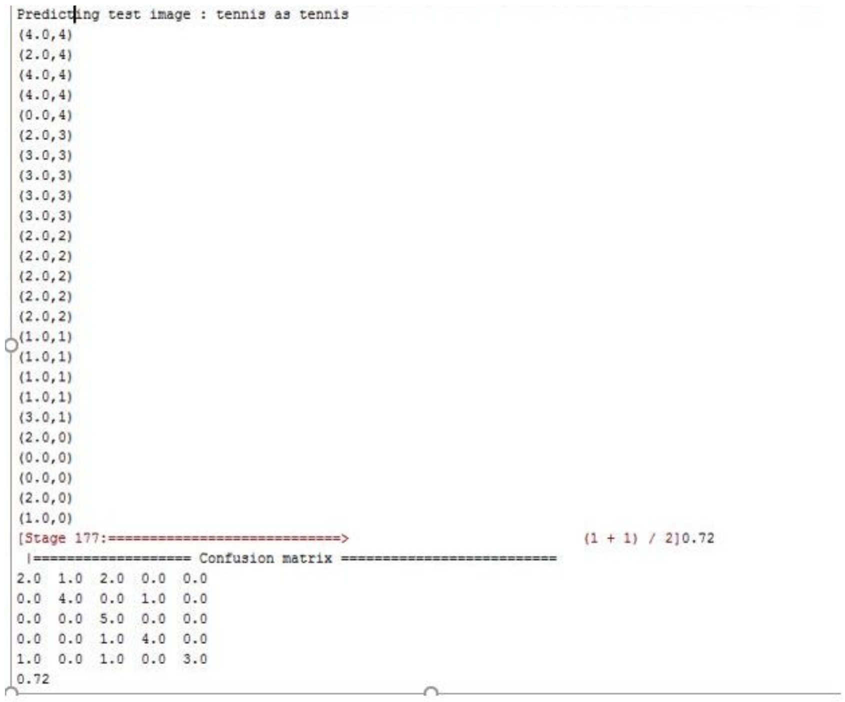
6.2 Phase 2 ScreenShots

Screenshots for Shallow Learning.



```
Predicting test image : tennis as tennis
(4.0,4)
(4.0,4)
(2.0,4)
(2.0,4)
(4.0,4)
(3.0,3)
(2.0,3)
(3.0,3)
(3.0,3)
(3.0,3)
(2.0,2)
(2.0,2)
(2.0,2)
(2.0,2)
(2.0,2)
(1.0,1)
(1.0,1)
(1.0,1)
(1.0,1)
(1.0,1)
(0.0,0)
(0.0,0)
(2.0,0)
(0.0,0)
(0.0,0)
                                                                  0.84
 |=================== Confusion matrix ==========================
4.0  0.0  1.0  0.0  0.0
0.0  5.0  0.0  0.0  0.0
0.0  0.0  5.0  0.0  0.0
0.0  0.0  1.0  4.0  0.0
0.0  0.0  2.0  0.0  3.0
0.84
17/02/17 06:46:03 INFO RemoteActorRefProvider$RemotingTerminator: Shutting down remote daemon
```

Fig. Confusion Matrix result using Decision tree model



```
Predicting test image : tennis as tennis
(4.0,4)
(2.0,4)
(4.0,4)
(4.0,4)
(0.0,4)
(2.0,3)
(3.0,3)
(3.0,3)
(3.0,3)
(3.0,3)
(2.0,2)
(2.0,2)
(2.0,2)
(2.0,2)
(2.0,2)
(1.0,1)
(1.0,1)
(1.0,1)
(1.0,1)
(3.0,1)
(2.0,0)
(0.0,0)
(0.0,0)
(2.0,0)
(1.0,0)
[Stage 177:============================>                       (1 + 1) / 2]0.72
 |=================== Confusion matrix ==========================
2.0  1.0  2.0  0.0  0.0
0.0  4.0  0.0  1.0  0.0
0.0  0.0  5.0  0.0  0.0
0.0  0.0  1.0  4.0  0.0
1.0  0.0  1.0  0.0  3.0
0.72
```

Fig. Confusion Matrix result using Random Forest Model.

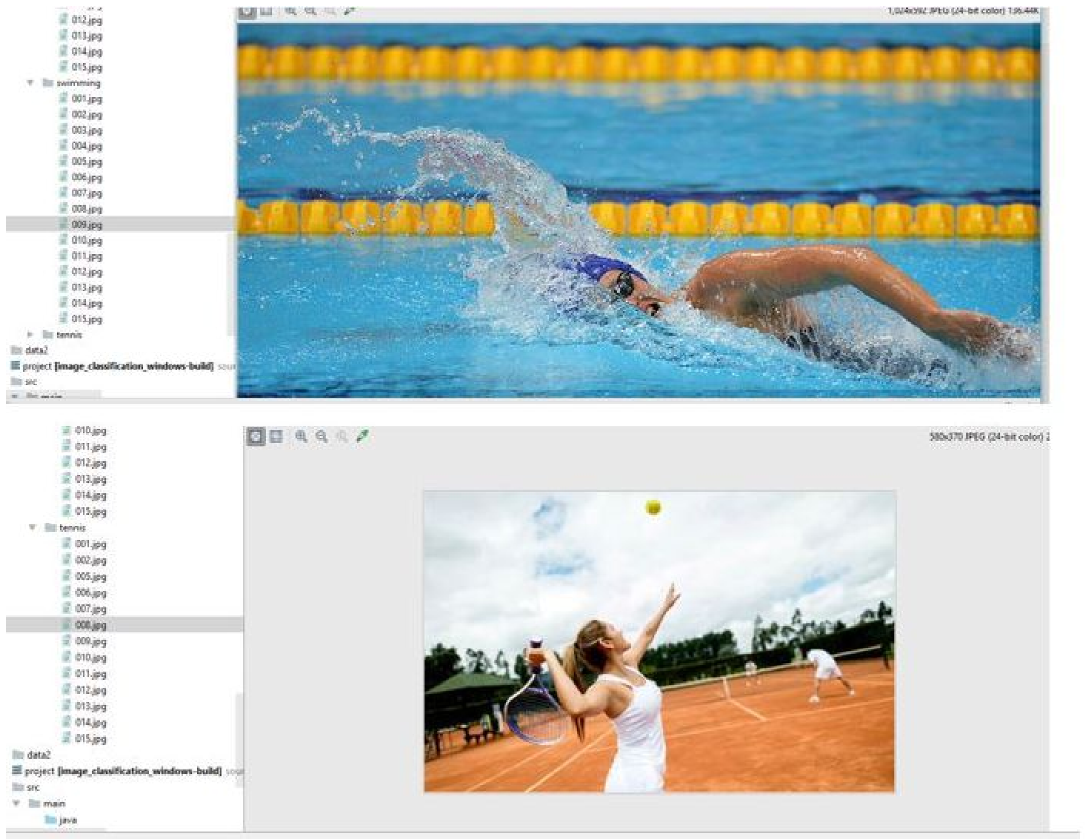


Fig. Sample of training images for image classification using decision tree.

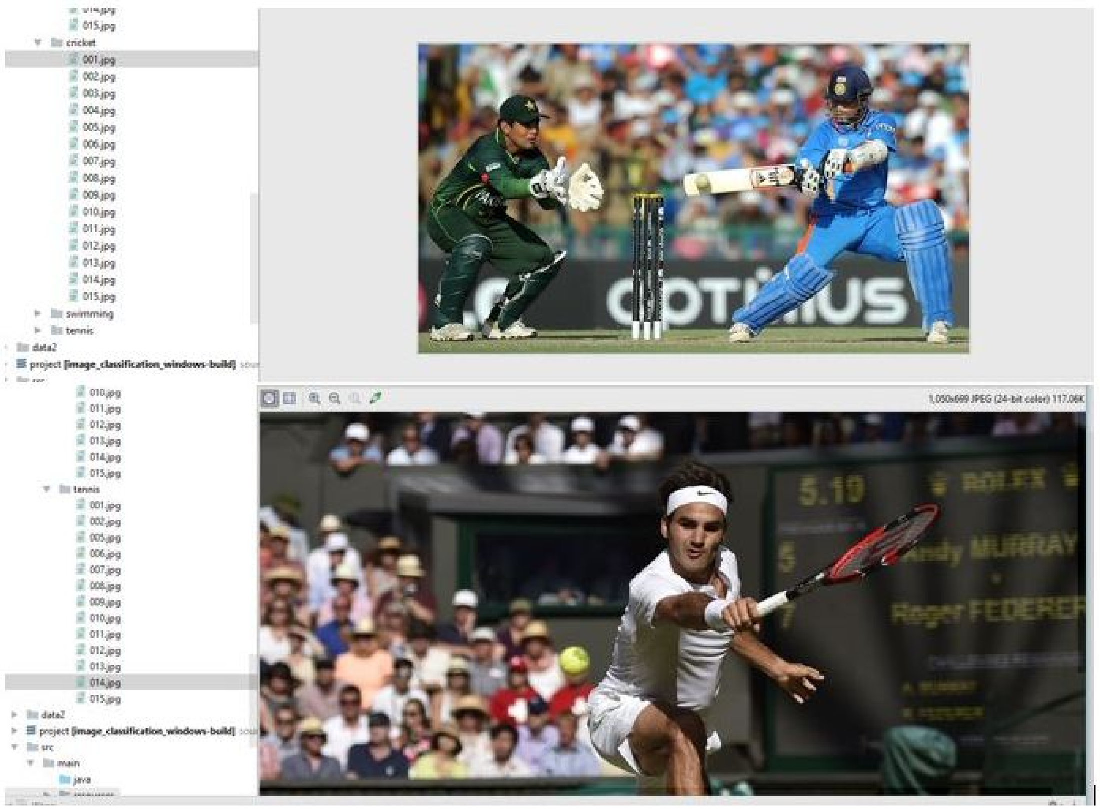
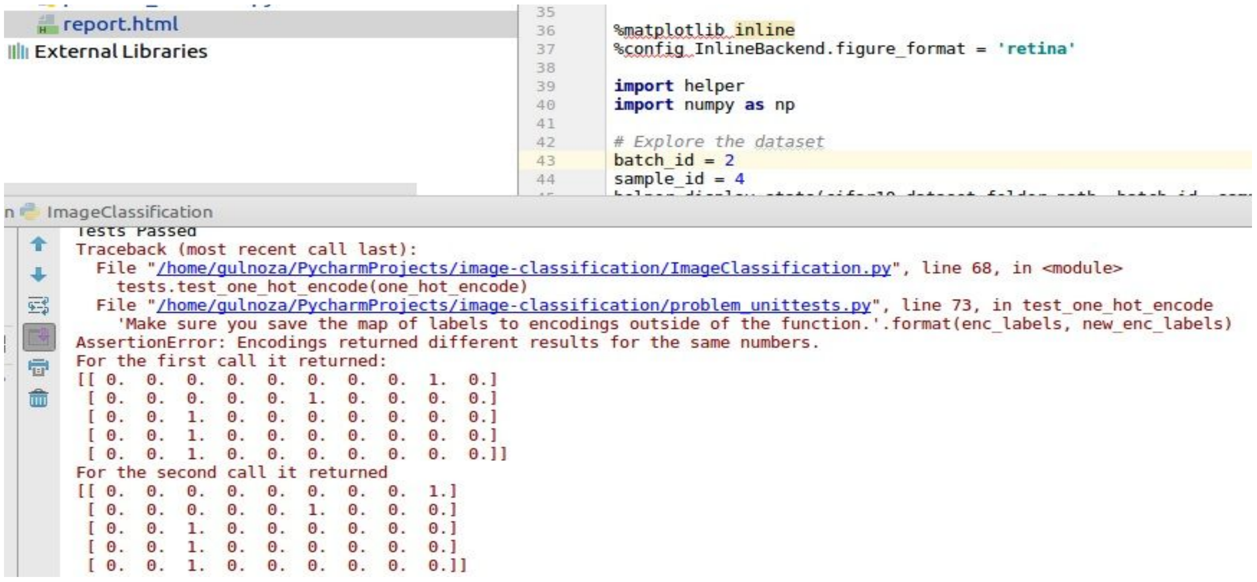

cricket
001.jpg
swimming
tennis
data2
project [image_classification_windows-build]
src
main
java
1,050x699 JPEG (24-bit color) 117.06K

Fig. sample of training/testing images for image Classification

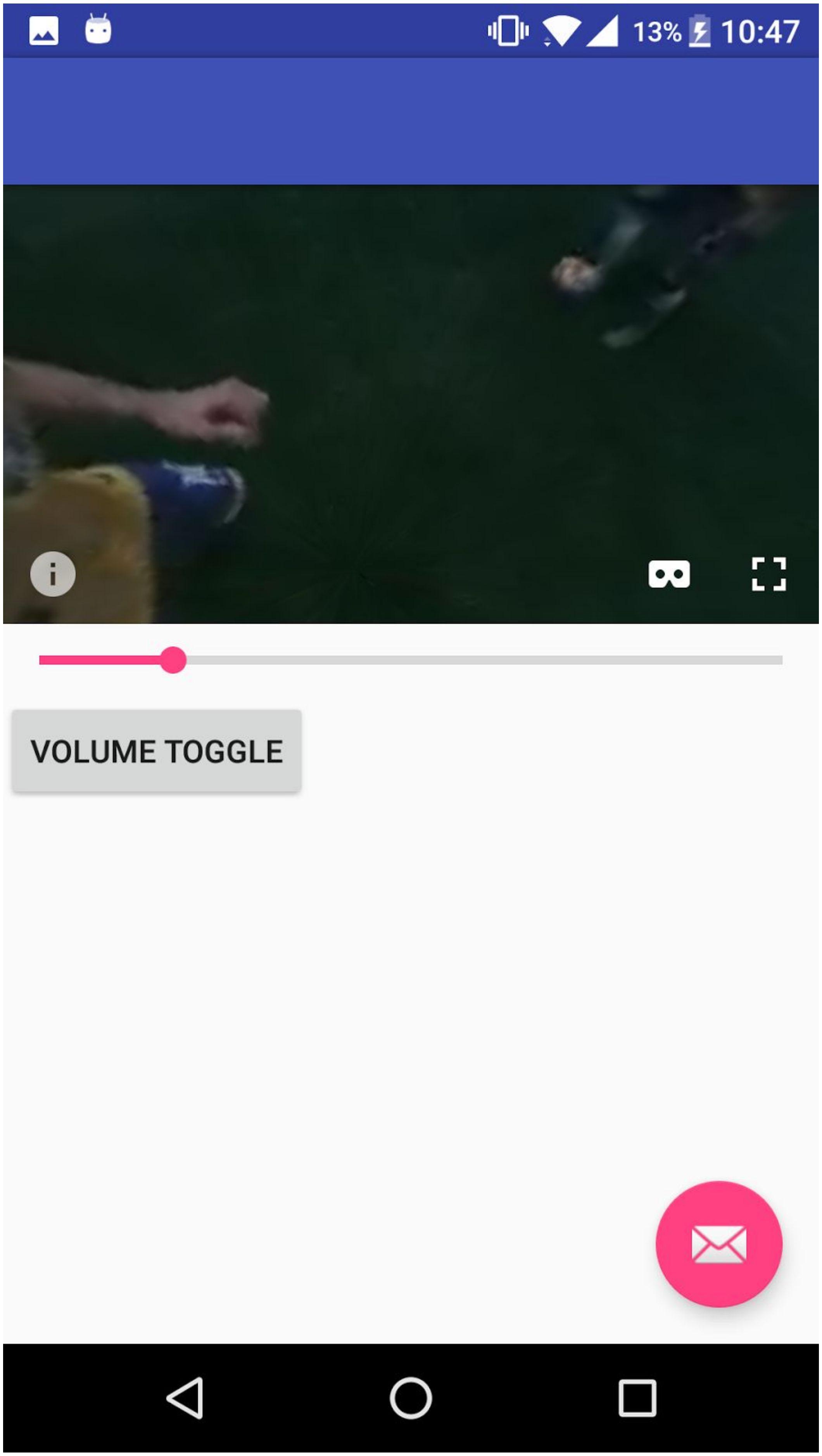
Screen Shot of output (confusion matrix) using Tensorflow:

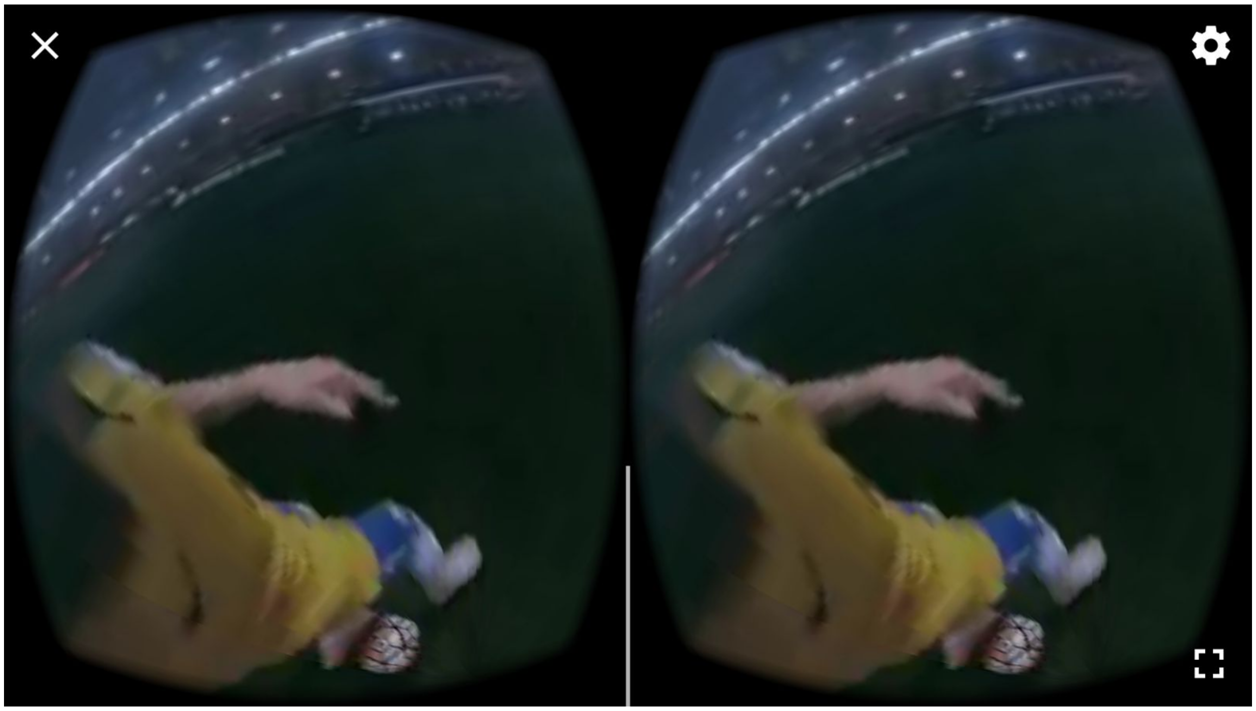


```
%matplotlib inline
%config InlineBackend.figure_format = 'retina'

import helper
import numpy as np

# Explore the dataset
batch_id = 2
sample_id = 4
```

```
Tests Passed
Traceback (most recent call last):
  File "/home/gulnoza/PycharmProjects/image-classification/ImageClassification.py", line 68, in <module>
    tests.test_one_hot_encode(one_hot_encode)
  File "/home/gulnoza/PycharmProjects/image-classification/problem_unittests.py", line 73, in test_one_hot_encode
    'Make sure you save the map of labels to encodings outside of the function.'.format(enc_labels, new_enc_labels)
AssertionError: Encodings returned different results for the same numbers.
For the first call it returned:
[[ 0.  0.  0.  0.  0.  0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  0.  1.  0.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.  0.  0.  0.  0.  0.]]
For the second call it returned
[[ 0.  0.  0.  0.  0.  0.  0.  0.  1.]
 [ 0.  0.  0.  0.  0.  1.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.  0.  0.  0.  0.]]
```

Screenshots for Google cardboard API


13%
10:47
VOLUME TOGGLE



7. PROJECT MANAGEMENT

7.1 Plan & Project Timelines , Members, Task Responsibility

Implementation status report

- **Work Completed:**
    1. **Description:**
       We tried to implement our application using Tensorflow also we implemented application using Spark API (Random Forest and Decision Tree models). For this part of the project we needed to find dataset of sport images, however we could not find it, we are still working on our model.
    2. **Responsibility**
       Gulnoza Khakimova: Implemented application using Tensorflow (Deep Learning) by implementing CNN model, searched for the dataset, worked on Shallow learning, Decision tree and Random forest models.

       Ganesh Taduri: Implementation of application using Tensorflow(Deep Learning) with the help of Inception model, worked on Shallow learning, Decision tree and Random forest models.

       Raghava Kundavajjala: Worked on implementation of Google cardboard, worked on Shallow learning, Decision tree and Random forest models. Also worked on features extraction and object detection using SPARK API.
    3. **Time taken**
       Tensorflow implementation using CNN: Around 30 hours
       Tensorflow impelentation using Inception: Around 30 hours
       Google cardboard Implementation: Around 30 hours.
       Shallow learning, decision tree, random forest: Around 12 hours.
    4. **Contributions**
       Gulnoza Khakimova: 33%
       Ganesh Taduri: 33%
       Raghava Kundavajjala: 33%

- **Work to be completed**
    1. **Description**
       We need to complete implementation of our model by using Deep learning, so our application could answer more questions asked by user.
       Responsibility
       Future Phases work yet to be shared depending on the Work load.

8. References

[1]http://dl.acm.org.proxy.library.umkc.edu/citation.cfm?id=2983530&CFID=894804323&CFTOKEN=31026051

[2]http://dl.acm.org.proxy.library.umkc.edu/citation.cfm?id=2535589&CFID=894804323&CFTOKEN=31026051

[3]http://dl.acm.org.proxy.library.umkc.edu/citation.cfm?id=2830309&CFID=894804323&CFTOKEN=31026051

[4]http://dl.acm.org.proxy.library.umkc.edu/citation.cfm?id=1466673&CFID=894804323&CFTOKEN=31026051

[5]https://techcrunch.com/2016/09/15/how-virtual-reality-is-transforming-the-sports-industry/

[6]http://www.idiap.ch/~mcernak/public/MCernak_RTT02_eng.pdf

[7]http://www.foxsports.com/presspass/latestnews/2016/09/13/fox-sports-and-livelikevr-team-to-present-ohio-state-at-oklahoma

[8]https://www.voicebot.net/

[9]http://store.steampowered.com/app/414120/

[10]https://www.researchgate.net/publication/265917403_VRIO_A_Speech_Processing_Unit_for_Virtual_Reality_and_Real-World_Scenarios_-_An_Experience_Report

[11]http://www.nextvr.com/news

[12]http://fortune.com/2016/04/29/mlb-eon-sports-vr/

[13]http://www.livelikevr.com/#sectionHome

[14]http://virtuallylive.com/
[15] https://docs.api.ai/docs/welcome
[16]https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/?utm_campaign=chrome_series_machinelearning_063016&utm_source=gdev&utm_medium=yt-desc#1
[17]https://github.com/googlevr/gvr-android-sdk/tree/master/samples/sdk-videoplayer/src/main/java/com/google/vr/sdk/samples/videoplayer