Sherlock's Eye Using Machine Learning

Azhar Mithani, Saurabh Bade, Yash Chavhan, Swapnil Vaidya

Computer Department, All India Shri Shivaji Memorial Society's Institute of Information Technology,Pune, Maharashtra, India

***Abstract*—** Unparalleled growth in the sharing of media via networks has prompted a great deal of research into issues pertaining to image labelling and classification. The training and classification of image retrieval systems requires a large number of labelled images; however, most researchers employ synthetic datasets for their experiments. Our proposed system is simple and easy to use because of it's multi-purpose use. Experiments demonstrate the excellent performance of the proposed system with regard to accuracy and response time. To end with we provide machine learning in our app to solve some real-world problems and make some lives a little bit easier.

***Keywords*—** Object Recognition, Data Collection, Face Recognition.

I. INTRODUCTION

To obtain image related information object recognition can be used; however, images with data are required for training such a system. It will take a great deal of time and incur high costs for recruiting individuals to identify objects in images. Public datasets such as that provided by the Pascal VOC 2012 Challenge are opted by researchers to use. The size and content of the dataset restrict the result. For the development other researchers take help from internet users. For the establishment of such a database we adopt a similar approach in this study. Creating a large dataset comprising images with detailed information pertaining to the items included in the images as well as their location within the frame is our aim. We propose engaging mobile phone users in a gaming online application (App) rather than recruiting a large number of paid workers to manually label images. In which using the touch function of a mobile device to draw the contours of the object participants report whether certain object exists in a photo and mark the position. To minimize the negative effects of erroneous responses and to identify users who consistently provide correct responses we have developed a method. Convincing a greater number of users to employ the system is achieved by the ease with which the system can be used.

II. Literature Review

Getting started with machine learning can be difficult for many developers. Typically, new ML developers spend countless hours learning the intricacies of implementing low-level models, using frameworks, and more. Even for the seasoned expert, adapting and optimizing models to run on mobile devices can be a huge undertaking. Beyond the machine learning complexities, sourcing training data can be an expensive and time consuming process, especially when considering a global audience. With ML Kit, you can use machine learning to build compelling features, on Android and iOS, regardless of your machine learning expertise. ML Kit gives you both on-device and Cloud APIs, all in a common and simple interface, allowing you to choose the ones that fit your requirements best. The on-device APIs process data quickly and will work even when there's no network connection, while the cloud-based APIs leverage the power of Google Cloud Platform's machine learning technology to give a higher level of accuracy. If you're seasoned in machine learning and you don't find a base API that covers your use case, ML Kit lets you deploy your own TensorFlow Litemodels. You simply upload them via the Firebase console, and we'll take care of hosting and serving them to your app's users. This way you can keep your models out of your APK/bundles which reduces your app install size. Also, because ML Kit serves your model dynamically, you can always update your model without having to re-publish your apps.

But there is more. As apps have grown to do more, their size has increased, harming app store install rates, and with the potential to cost users more in data overages. Machine learning can further exacerbate this trend since models can reach 10's of megabytes in size. So we decided to invest in model compression. Specifically, we are experimenting with a feature that allows you to upload a full TensorFlow model, along with training data, and receive in return a compressed TensorFlow Lite model.

ML Kit makes it easy to apply ML techniques in your apps by bringing Google's ML technologies, such as the Google Cloud Vision API, Mobile Vision, and TensorFlow Lite, together in a single SDK. Whether you need the power of cloud-based processing, the real-time capabilities of Mobile Vision's on-device models, or the flexibility of custom TensorFlow Lite models, ML Kit makes it possible with just a few lines of code.

At Google I/O this year we saw the introduction of Firebase MLKit, a part of the Firebase suite that intends to give our apps the ability to support intelligent features with more ease. The SDK currently comes with a collection of pre-defined capabilities that are commonly required in applications—you'll be able to implement these in your application regardless of whether you are familiar with machine learning or not. Now, what Firebase ML Kit offers to us is already possible to implement yourself using various machine-learning technologies. The thing with Firebase ML is that as well as offering these capabilities underneath a form of wrapper, it also takes these technologies and offers their capabilities inside of a single SDK.

Whilst we can implement these things without Firebase ML, some reasons why we may not be able to do so may be due to:

- Lack of machine learning knowledge may hold us back from being able to implement such features—maybe we find it overwhelming or just don't have the time to be able to ramp up in these areas

- Finding machine learning models that are super accurate and well trained can be not only difficult, but at the same time hard to choose which ones to use and then optimise for your platform.

- Hosting your ML model for cloud access may also be something to bring difficult to your ML implementation. Packaging it within your app can sometimes be a more straightforward approach, but that itself comes with some drawbacks.

With these in mind, it can be difficult to know where to start. This is one of the main goals of Firebase ML Kit—to make Machine Learning to our Android and iOS applications more accessible to developers and available in more apps. Currently ML Kit offers the ability to:

- Recognize text
- Recognize landmarks
- Face recognition
- Scan bar codes
- Label images

To be able to utilize these features all we need to do is pass our desired data to the SDK and in return we will receive the data back dependant on what part of ML Kit we are using. The data returned will be dependant on the machine learning capability being used, you will just need to extract the data from the response that is returned to you.

And if one of these above does not satisfy your machine learning requirements, Firebase MLKit offers the ability for you to upload your own custom tensorflow lite models so that you don't need to worry about the hosting of these models or the serving of them to your users devices.

One of the nice things about Firebase ML is that it offers it's machine learning abilities both on the device and on the cloud, this allows you to be creative and mindful of how and when you use machine learning. For example, some operations may be intensive so we have to bear this in mind—luckily though we have this choice of whether we want to use on-device or cloud learning for most of the firebase ML capabilities. On-device APIs in Firebase MLKit are designed to work fast and will be able to provide results even when a network connection isn't present. On the other-hand, cloud-based APIs utilise the Google Cloud platforms
Machine Learning technology to provide a increased level of accuracy.Each of the ML capabilities will require this dependency, some require an extra individual one but we will cover those through this series of articles. At this point, you will have the vision tools available to your app. This is where you will provide the input data to the model from your application, maybe this will be content for barcode scanning or face recognition etc, then MLKit will provide back some result value based off of this data which you can then apply to your application. By default for on-device training, the ML model will be downloaded when it is first used by your application. However, if you wish to download the required models at install time then you can do so by adding meta data to your manifest. This should really depend on your application. If the ML model is a core part of your application experience then this would make sense, otherwise the models should be downloaded as they are required. ML Kit's Text Recognition provides both on-device and cloud-based APIs. You can choose which one to use depending on your use case.

The ML Kit's Text Recognizer segments text into blocks, lines, and elements.

- **Block** is a contiguous set of text lines, such as a paragraph or column.
- **Line** is a contiguous set of words on the same vertical axis.
- **Element** is a contiguous set of alphanumeric characters on the same vertical axis.

ML Kit makes it easy to apply ML techniques in your apps by bringing Google's ML technologies, such as the Google Cloud Vision API, Mobile Vision, and TensorFlow Lite, together in a single SDK. Whether you need the power of cloud-based processing, the real-time capabilities of Mobile Vision's on-device models, or the flexibility of custom TensorFlow Lite models, ML Kit makes it possible with just a few lines of code.

III. System Architecture:-

Scope of the project is that The image taken by the mobile camera by using artificial intelligence and machine learning it will predict object.

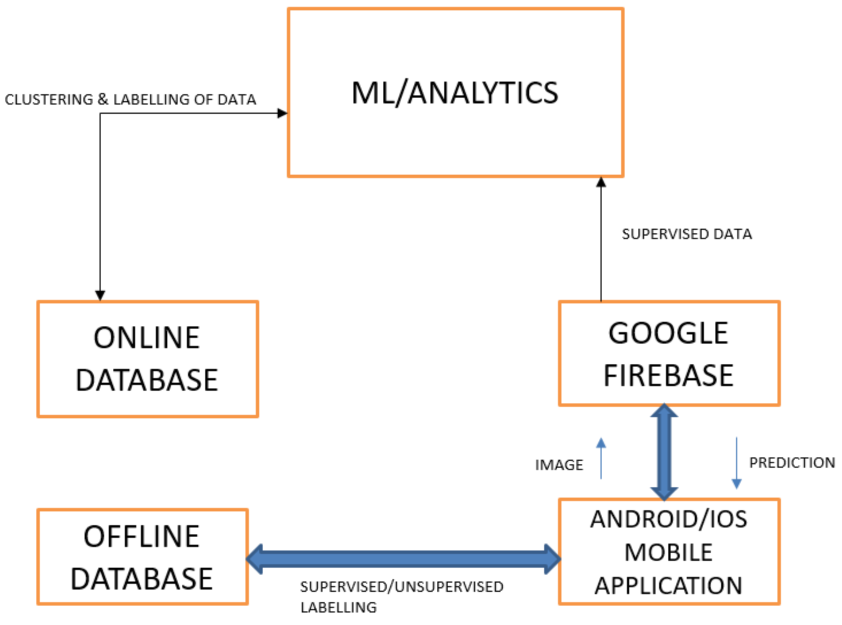
ML/ANALYTICS
CLUSTERING & LABELLING OF DATA
SUPERVISED DATA
ONLINE DATABASE
GOOGLE FIREBASE
IMAGE
PREDICTION
OFFLINE DATABASE
ANDROID/IOS MOBILE APPLICATION
SUPERVISED/UNSUPERVISED LABELLING

figure 3.1:system architecture

The wellness capacity will mark the wellness estimations of every chromosomes and apply the clustering rule for top principles development. At the point when variety is finished then GA will close and we will get the GA result at last. In second stage we represent the fuzzy validation, based on likelihood capacity. Here the collective will establish base on every characteristic qualities for ordinary interruption base rule.

In the third stage affiliation methodology will create the continuous principles for better identification. Help of these frameworks can effectively recognize the both attacks like inward or outer. The figure 1 demonstrates the general framework structure and information stream.

IV. Key Capabilities

table 4.1: Key Capabilities

Production-ready for common use cases	ML Kit comes with a set of ready-to-use APIs for common mobile use cases: recognizing text, detecting faces, identifying landmarks, scanning barcodes, labeling images, and identifying the language of text. Simply pass in data to the ML Kit library and it gives you the information you need.

On-device or in the cloud	ML Kit's selection of APIs run on-device or in the cloud. Our on-device APIs can process your data quickly and work even when there's no network connection. Our cloud-based APIs, on the other hand, leverage the power of Google Cloud Platform's machine learning technology to give you an even higher level of accuracy.
Deploy custom models	If ML Kit's APIs don't cover your use cases, you can always bring your own existing TensorFlow Lite models. Just upload your model to Firebase, and we'll take care of hosting and serving it to your app. ML Kit acts as an API layer to your custom model, making it simpler to run and use.

V. Algorithms

1. Image Labelling Algorithm

Input: provide live or offline image.

Output: Classification of live or offline image provided in the input.

Step 1:Setup Firebase in your project and add the vision dependency
This is a simple one, simply setup firebase in your project. You can find a good tutorial here. In order to use this API, you also need to add the relevant dependencies.

Step 2: Implement Camera functionality in your app
The Vision API needs an image to extract the data from, so either create an app that lets you upload images from the gallery or create an app that uses the Camera APIs to click a picture and use it instead. I found this library to be pretty handy and easy to use instead of the framework Camera APIs so this is what I end up using.

Step 3:Use the Bitmap to make the API call
If you used the library above, it directly provides you with a Bitmap of the captured image which you can use to make an API call.

Step 4:In the above code snippet, we first create a FirebaseVisionImage from the bitmap. Then we create an instance of the FirebaseVisionLabelDetector which goes through the FirebaseVisionImage and finds the appropriate FirebaseVisionLabels (objects) it notices in the supplied image.

Step 5:Lastly we pass the Image to the detectInImage() method and let the detector label the Image.

Step 6:We have a success and a failure callback which contains a list of labels and an exception respectively. You can go ahead and loop through the list to get the Name, Confidence and Entity ID for every label that was detected in the image.

2. Pattern Matching Algorithm for sub attack classification

Input: network connection N which is belongs from KDD set or network adapter, network rules R, Threshold T.

Output: Classify all network instances with label

Step 1:Setup Firebase in your project and add the vision dependency
This is a simple one, simply setup firebase in your project. You can find a good tutorial here. In order to use this API, you also need to add the following dependencies.

Step 2:Implement Camera functionality in your app
The Vision API needs an image to extract the data from, so either create an app that lets you upload images from the gallery or create an app that uses the Camera APIs to click a picture and use it instead.
I found this library to be pretty handy and easy to use instead of the framework Camera APIs so this is what I end up using.

Step 3:Use the Bitmap to make the API call
If you used the library above, it directly provides you with a Bitmap of the captured image which you can use to make an API call.

In the above code snippet, we first create a **FirebaseVisionImage** from the bitmap.Followed by that, we create an instance of **FirebaseVisionBarcodeDetector**which is used to recognize barcodes in a supplied Image.Lastly we pass the Image to the **detectInImage()** method and let the detector detect text from the image.

We have a success and a failure callback which contains a list of **FirebaseVisionBarcode** objects containing all the detected barcodes and an exception respectively.
Further, we can loop over the above list and get information contained in each and every Barcode that was detected.

VI. Results

We were able to classify the real time objects such as chair,glasses, helmet,etc. We were also able to detect face and textual data using the firebase dataset.The following figures show the actual implementation of the system.

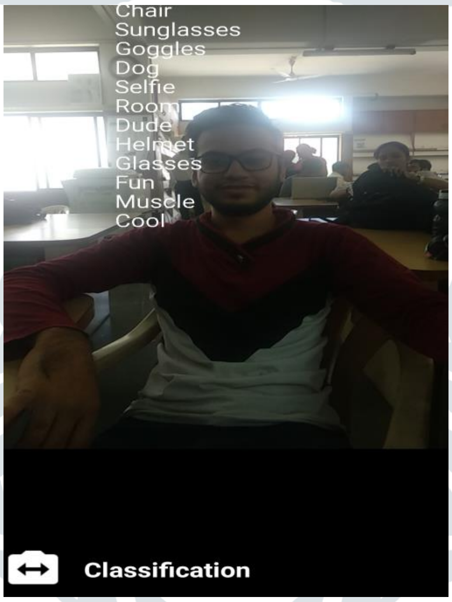

figure 6.1: Classification

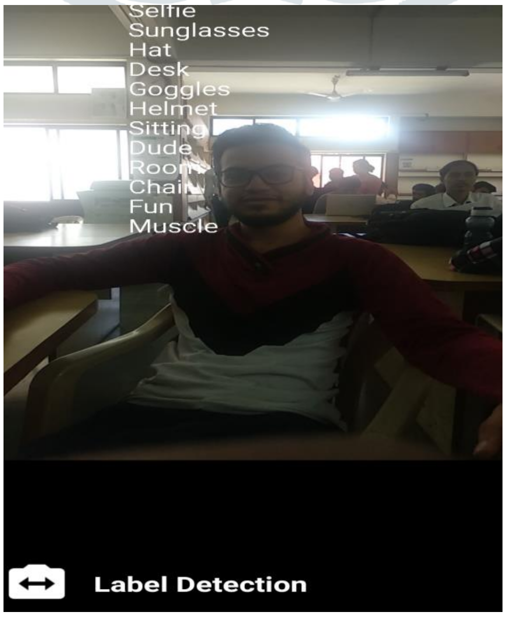

figure 6.2:Label Detection

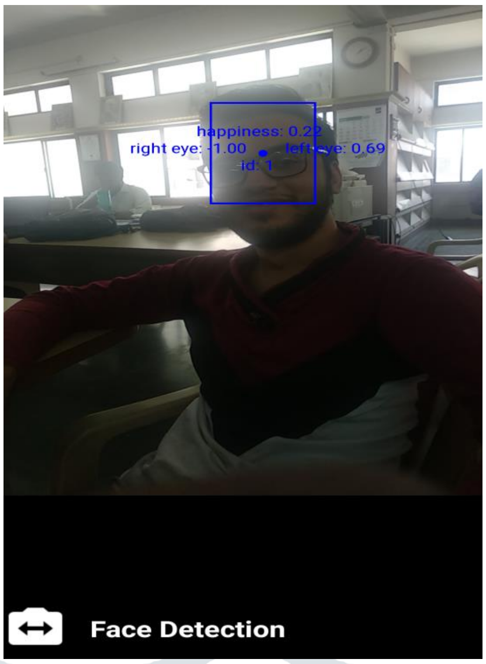

figure 6.3: Face Detection

VII. Conclusion

In the end we can conclude that the application can predict the objects in the image.The future work involves improving its accuracy and making it more advanced and user friendly.

VIII. References

1. Hyuk-Jin Jeong, InChang Jeong, Hyeon-Jae Lee, Soo-Mook Moon, " Computation Offloading for Machine Learning Web Apps in the Edge Server Environment," 2018 IEEE 38th International Conference on Distributed Computing Systems.

2. David Novotny, Diane Larlus, and Andrea Vedaldi" Capturing the Geometry of Object Categories from Video Supervision," .

3. Wen-Yen Tseng, Kai-Hsaign Chen, Jen-Wei Huang,"An Effective Object Recognition System by a Mobile Application," 10Th international Conference on Ubi-media computing and workshops.

4. Shuqiang Jiang, Senior Member, IEEE, Weiqing Min, Member, IEEE, and Shuhuan Mei, " Hierarchy-Dependent Cross-Platform Multi-View Feature Learning for Venue Category Prediction," IEEE TRANSACTIONS ON MULTIMEDIA, VOL. X, NO. XX, MONTH YEAR.

5. Azhar Mithani, "A SMART PARKING SYSTEM USING RPI", International Journal of Advance Research in Science and Engineering, Vol. No. 6, Issue No.9, 09 September 2017.

6. Humaid Alshamsi, Veton Kepuska,Hongyeng Meng, " Automated Facial Expression Recognition App Development on Smart Phones using Cloud Computing,".