MDRED: MULTI-MODAL MULTI-TASK DISTRIBUTED RECOGNITION FOR EVENT DETECTION

A THESIS IN
Computer Science

Presented to the Faculty of the University
Of Missouri-Kansas City in Partial Fulfillment
of the Requirements for the Degree

MASTER OF SCIENCE

By
NAGESWARA RAO NANDIGAM

B.E, Vasavi College of Engineering,
affiliated to Osmania University, AP, India, 500031

Kansas City, Missouri
2018

MDRED: MULTI-MODAL MULTI-TASK DISTRIBUTED RECOGNITION FOR EVENT DETECTION

Nageswara Rao Nandigam, Candidate for the Master of Science Degree

University of Missouri-Kansas City, 2018

ABSTRACT

Understanding users' context is essential in emerging mobile sensing applications, such as Metal Detector, Glint Finder, Facefirst. Over the last decade, Machine Learning (ML) techniques have evolved dramatically for real-world applications. Specifically, Deep Learning (DL) has attracted tremendous attention for diverse applications including speech recognition, computer vision. However, ML requires extensive computing resources. ML applications are not suitable for devices with limited computing capabilities. Furthermore, customizing ML applications for users' context is not easy. Such a situation presents real challenges to mobile-based ML applications. We are motivated to solve this problem by designing a distributed and collaborative computing framework for ML edge computing and applications.

In this thesis, we propose the Multi-Modal Multi-Task Distributed Recognition for Event Detection (MDRED) framework for complex event recognition with images. The MDRED framework is based on a hybrid ML model that is composed of Deep Learning (DL) and Shallow Learning (SL). The lower level of the MDRED framework is based on the DL models for (1) object detection, (2) color recognition, (3) emotion recognition, (4) face detection, (5) text detection with event images. The higher level is based on the SL-based fusion techniques for the event detection based on the outcomes from the lower level DL models. The fusion model is designed as a weighted feature vector generated by a modified Term Frequency and Inverse Document Frequency (TF-IDF) algorithm, considering common and unique multi-modal features that are recognized for event detection. The prototype of the MDRED framework has been implemented: A master-slave architecture was designed for coordinating the distributed computing among multiple mobile devices at the edge while connecting the edge devices to the cloud ML servers. The MDRED model has been evaluated with the benchmark event datasets and compared with the state-of-the-art event detection models. The MDRED accuracy of 90.5%, 98.8%, 78% for SocEID, UIUC Sports, RED Events datasets, respectively, outperformed the baseline models of AlexNet-fc7, WEBLY-fc7, WIDER-fc7 and Event concepts. We also demonstrate the MDRED application running on Android devices for the real-time event detection.

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Computing and Engineering, have examined a thesis titled "Multi-Modal Multi-Task Distributed Recognition for Event Detection" presented by Nageswara Rao Nandigam, candidate for the Master of Science degree, and hereby certify that in their opinion, it is worthy of acceptance.

Supervisory Committee

Yugyung Lee, Ph.D., Committee Chair
Department of Computer Science Electrical Engineering

Zhu Li, Ph.D.
Department of Computer Science Electrical Engineering

Baek-Young Choi, Ph.D.
Department of Computer Science Electrical Engineering

CONTENTS

ILLUSTRATIONS

Figure Page

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank Dr. Yugyung Lee for her continuous support throughout my thesis project. Her valuable advises and suggestions helped me to defense my work within deadline. Her experience and pragmatic feedback with new approaches helped me in strengthening this thesis work. She is open to listen all new ideas and challenged me to think outside of the box. It is my pleasure to work with her throughout the Master's program.

My research work could not have been completed without the support from the University of Missouri-Kansas City. It provided me with great world-class lab facilities and the best environment to conduct research work with GPU machines.

Finally, I would like to appreciate my family and friends for their continuous support to pursue my dreams without any problems.

CHAPTER 1

INTRODUCTION

Multimedia events are generally consisting of low-level components of objects, scenes, and actions. The mobile applications such as disaster detector, video annotation, and surveillance systems are some of the applications which require analysis of events. For example, automatic detection of events like traffic accidents, violent crimes, and incidents reporting to the particular department makes the less human intervention.

Event detection models require a lot of training examples with labeled data. These models trained with Supervised Learning algorithms to predict the similar examples. One of the event detection analysis [1] paper published in 2017 for multimedia event detection (MED) with promising results which caught the attention of me in the field of deep learning.

Increase in usage of deep learning systems gone higher over the years. Figure 1 shows the google trend for people interest in deep learning systems. The Rapid development of deep learning systems with TensorFlow [2], Caffe [3], Theano [4], etc. have made possible to solve the real word problems like event detection, voice generation and etc. Computer vision [5] is one of the crucial areas where deep learning is yielding notable results.

1.1 Problem Statement

There were some of the efficient event detection systems proposed in recent times. However, those work is either based on object features or changing the convolutional or pooling players, transferal learning from Convolutional Neural Network (CNN) [6]. Not many systems utilized all visual recognition features of an image to train the models efficiently. Traditional methods of using only object features or manually designed features not efficient to adopt different event detections. Specifically, Deep Learning (DL) has attracted tremendous attention for diverse applications including speech recognition, computer vision. However, Machine Learning (ML) requires extensive computing resources. ML applications are not suitable for devices with limited computing capabilities. Furthermore, customizing ML applications for users' context is not easy. Such a situation presents real challenges to mobile-based ML applications.

1.2 Proposed Solution

In this thesis, we proposed a distributed and collaborative computing framework for ML edge computing and applications. The proposed solution includes examining the current strategies for making event detection to users and contriving a more semantic event detection framework. To accomplish this, we fabricated a model that can take a picture as input and gives the meaningful event as an output for that image.

The proposed framework based on hybrid ML model which consist of Deep Learning (DL) and Shallow Learning (SL) Models. It has two levels of abstraction. The lower level is based on DL recognition models. Recognitions include object detection, color recognition, faces recognition, emotion detection, and text recognition. The higher level is based on the SL-based fusion techniques for the event detection based on the outcomes from the lower level DL models. The fusion model is implemented with the weighted feature vector by a modified Term Frequency and Inverse Document Frequency (TF-IDF) algorithm [7]. This fusion vector which consists of common and unique features that are used to recognize for event detection. The prototype of this framework implemented as follows: A master-slave architecture was designed for coordinating the distributed computing among multiple mobile devices at the edge while connecting the edge devices to the cloud ML servers.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter provides background information on various key terms and techniques used to implement the proposed framework and provides an overview of related work that gives the better understanding of the problem statement.

2.1 Terminology and Technology

2.1.1 Machine Learning

Machine Learning (ML) algorithms use computational resources to study and learn from data without depending on the fixed algorithm. In today's time, Machine Learning applied to broader areas. Some of the areas are medicine, computer vision, cybersecurity, aerospace and etc. Machine Learning is a subcategory of Artificial Intelligence [8]. Machine Learning algorithms allow an application to use the statistical approach to predict the outcomes without being programmed explicitly. Unlike traditional computer programs, the Machine Learning algorithms adopt themselves from experience. The traditional programs have a set of commands to follow and deliver output based on the programmed algorithm but ML algorithms trained a model based on the set of input examples and predict the output by themselves. The ML algorithms improve the accuracy of models to make decisions if they are trained with more no input examples.

Figure 1 shows the steps involved in Machine Learning process which requires data to predict patterns and adopt the algorithms by themselves. Machine Learning algorithms divided based on the nature of data and type of feedback looking from the learning system. Primarily there are two types of algorithms widely used to solve real-world challenges. They are Supervised learning [9]and Unsupervised learning [10] which shows in Figure 2.

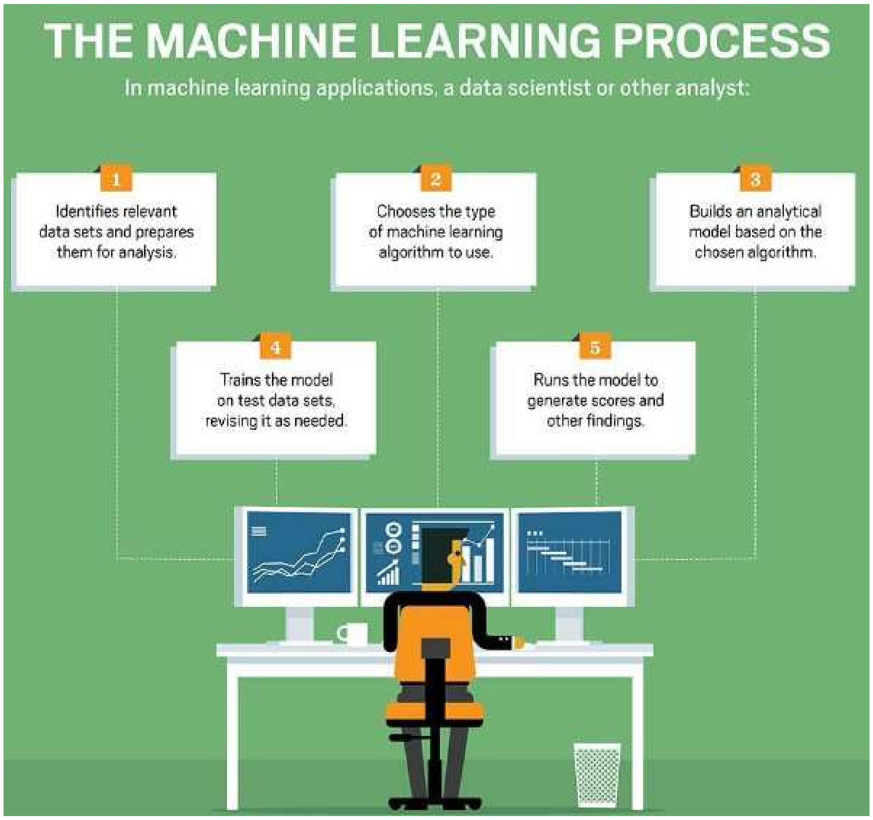

THE MACHINE LEARNING PROCESS
In machine learning applications, a data scientist or other analyst:
1
Identifies relevant data sets and prepares them for analysis.
2
Chooses the type of machine learning algorithm to use.
3
Builds an analytical model based on the chosen algorithm.
4
Trains the model on test data sets, revising it as needed.
5
Runs the model to generate scores and other findings.

Figure 1: Machine Learning Process [30]

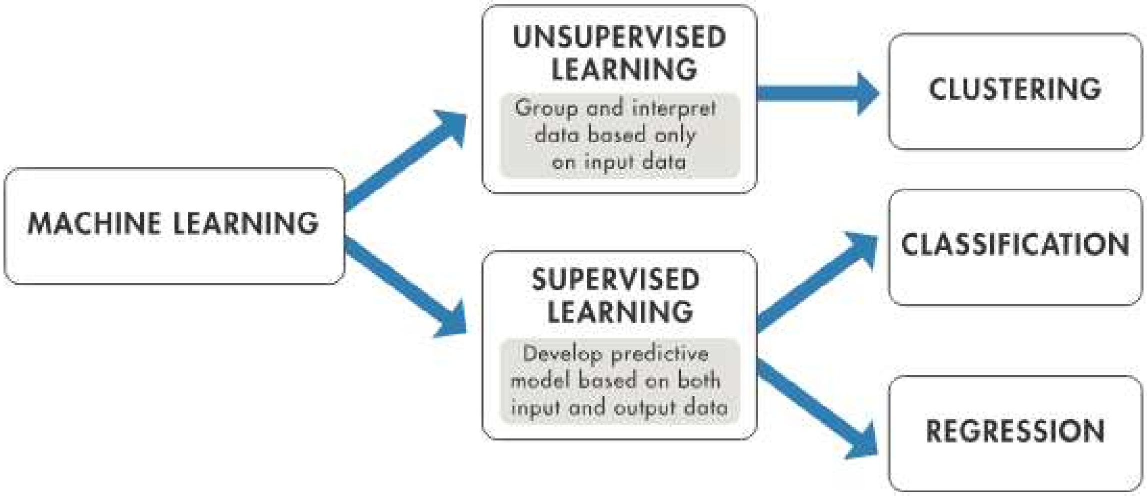

MACHINE LEARNING
UNSUPERVISED LEARNING
Group and interpret data based only on input data
SUPERVISED LEARNING
Develop predictive model based on both input and output data
CLUSTERING
CLASSIFICATION
REGRESSION

Figure 2: Machine Learning techniques include both Unsupervised and Supervised Learning [31]

A supervised algorithm takes a set of input data with mapped label output for each data point and trained the model with input-output mapped pairs. This algorithm improves the performance by checking the actual label with the predicted label to reduce the error. The trained model predicts the response for new unseen data. Supervised algorithms use two kinds of models for predictive analysis.

- Classification: Classification algorithms produce outcomes of discrete or category values. For example, whether the email is spam or not.
- Regression: Regression algorithms produce outcomes of continuous values. For example, Google stock price, changes in temperature and etc.

An unsupervised algorithm doesn't need labeled data, it is trained with input data without corresponding output labels. Unsupervised algorithm trained model by inferencing semantics from input datasets. It finds hidden patterns and inherent structure from data. Like supervised learning, unsupervised algorithms also have two kinds of techniques for predictive analysis.

- Cluster analysis: Grouping of similar objects in the same group which are way from the different group.
- Association: It rules-based learning and finds interesting relations between variables in datasets.

2.1.2 Shallow Learning

Shallow Learning is a type of Machine Learning algorithms which uses few layers to generate good predictive models. It requires feature selection and extraction techniques applied by experts on samples to create feature vectors. Shallow Learning algorithms trained with those features. It can perform well even though only a limited number of samples is available.

2.1.3 Deep Learning

Deep Learning is a subclass of Machine Learning algorithms. It uses multiple layers of nonlinear functionality for feature extraction and transformation. Each layer is using the output from previous layers as input. Figure 3 of Deep Learning network can be trained as supervised and/or unsupervised learning way. It has multiple levels of abstraction to learn from each other and levels can be formed as a hierarchy structure. The network follows a heuristic approach to improve the performance by doing backpropagation to update the weights for minimizing the error.

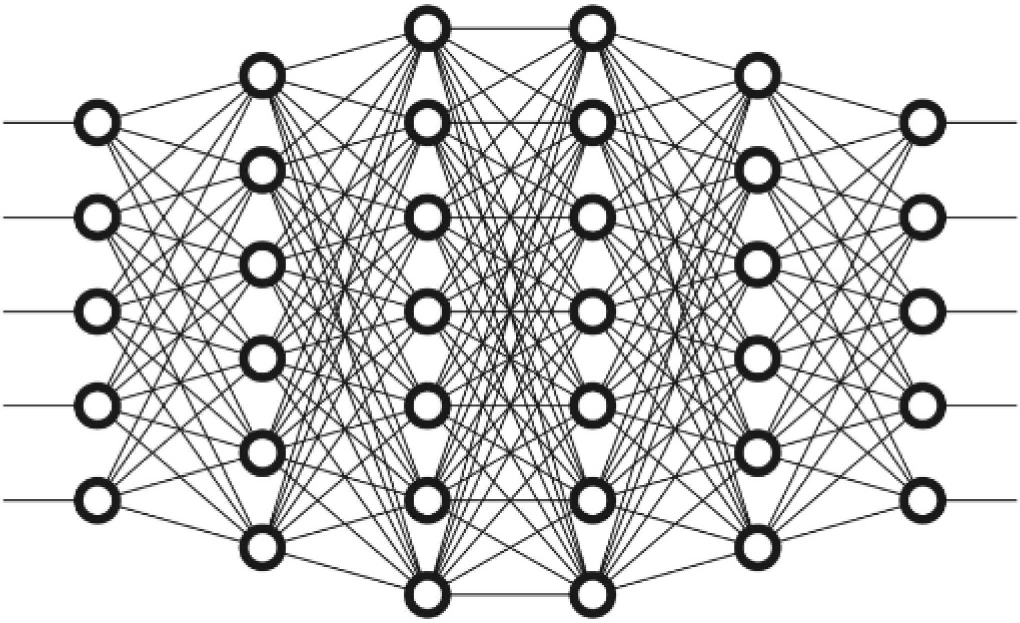


Figure 3: Deep Learning Network [32]

2.1.4 Distributed Computing

Figure 4 shows Distributed computing which uses distributed systems to solve computational problems. A distributed system is systems which are located over the network and interconnected each other with network protocols. These systems communicate and pass information to each other over a network. They all work together for a common goal and divide the tasks among each other to run parallel and distributed manner.

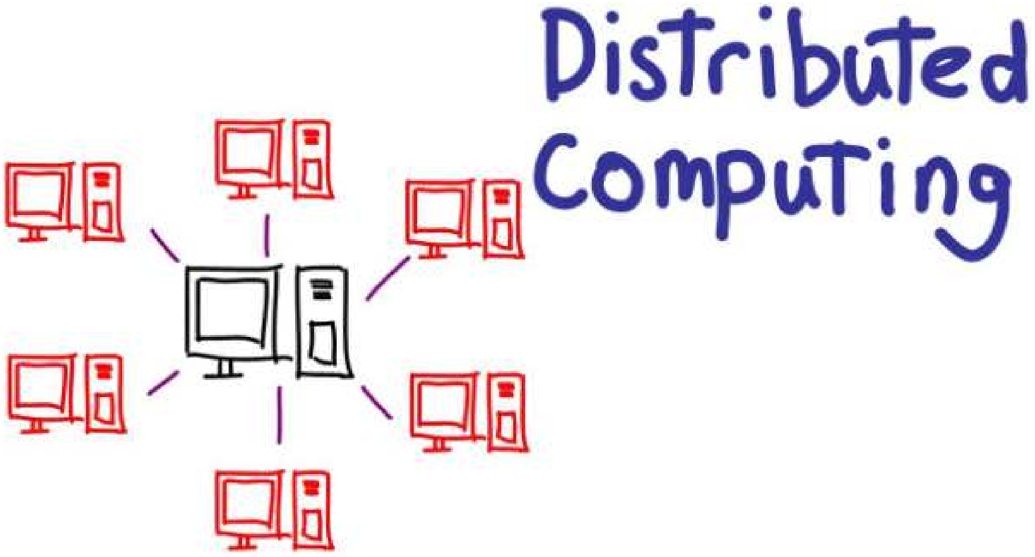


Figure 4: Distributed Computing [33]

2.2 Tools

2.2.1 Apache Spark

Apache spark [11] is an in-memory distributed data processing engine which runs on Hadoop [12]. Spark runs 100x faster than Hadoop for both batching and streaming data by leveraging the in memory and several optimization techniques. Spark provides a high-level API's for Scala, Python, R, and SQL shells to write applications and get insights from data science tools by applying 80 different kinds of transformation and action operations on data.

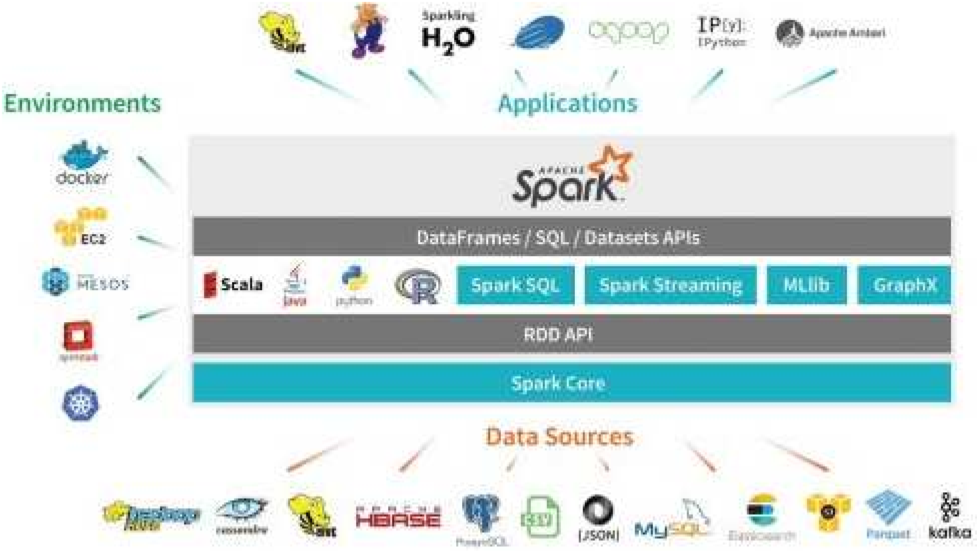


Figure 5: Apache Spark [34]

Spark has coupled with the core engine and acts like a distributed processing engine to schedule the tasks. The tasks include dispatching, scheduling, I/O functionality, etc. On the top of spark core, it provides API level abstraction to programming languages to run Resilient Distributed Datasets in parallel on the cluster. Apart from that Spark provides high-level libraries to create complex workflows:

Spark SQL – It is component on top of Spark core and provides data abstraction called data frames which are used to process structured and unstructured data with SQL queries.

MLlib - It is a distributed machine learning component present on top of Spark core. It is providing a lot of machine learning and statistical algorithms include Classification, Regression, and Clustering techniques. Machine learning utilities include Feature transformations, Model evaluation, hyper-parameter tuning, and model persistence tools.

Spark Streaming – It uses the spark core fast scheduling capability. It enables to combine streaming with batch and interactive queries.

GraphX – It is a distributed graph processing framework present on top of spark core. It issued to support graph and graph-parallel computations.

2.2.2 Android

Android [13] is a mobile operating system which is a modified version of Linux kernels and other software's which enabled to provide touchscreen capability for smartphone devices. Figure 6 shows the Android architecture and its components. It has four main layers and five sections:

Linux Kernel- It provides the abstraction between hardware and device drives like camera, display and etc.

Libraries – Libraries present on top of the kernel which as libraries for web browser web kit, SQLite for storage, Media framework for plug and play of audio and video.

Android Runtime- which has a Java runtime environment which designed and optimized for Android.

Application Framework – It provides many services to applications in the form of Java classes to access low-level API's.

Applications- It is the top layer and used to install applications in this layer

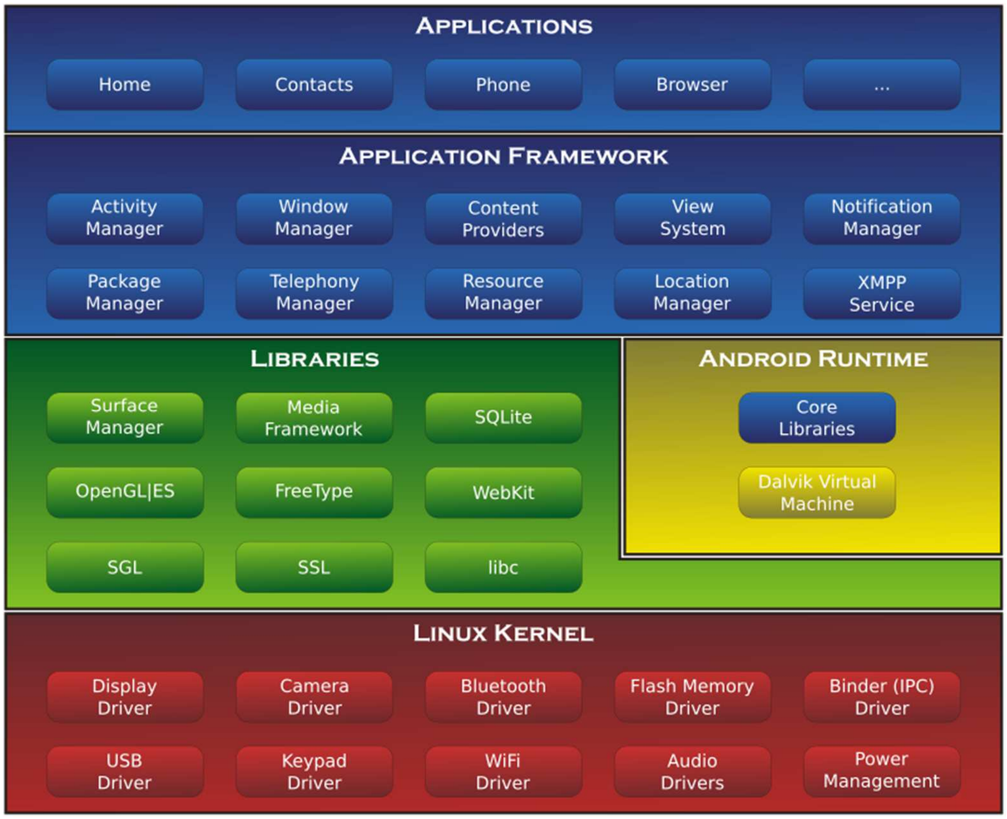


Figure 6: Android Architecture [35]

2.3 RELATED WORK

Doing the event detection in the device became a hard problem for several reasons. Deep Learning techniques are employed for event detection but those approaches need heavy lifting in terms of complex calculations needed for event detection models which were the main roadblocks. There are numerous variants designed for improving the event detection with images and deep learning techniques. Some recent works on Event detection include changing pooling layers in CNN and transfer learning. One of the interesting variants of event detection [14] provides the option to generate events for multimedia and noisy images.

Event detection using text data which comes from Deep Learning recognition models is a new variant approach which we followed in our work. Some of the works, which are in line with our work, were based on text data or event concepts [15] for event classification. Proposed fusion of multiple texts features to understand all visual contents in an image for better event detection. The similar fusion techniques for twitter text and Image features [16] were used for event detection. Table 1 shows the overview of some works in event detection we referred to differentiate their works from our work. Table 2 shows the overview of some works that are related to fusion techniques. Fusion techniques are based on only text features or image features or both text and image features.

In the work proposed by Samar M. Alqhtani et al. [17], they used Twitter data, which contain text and Image data. They built separate models for text event classification and image event classification which is extra overhead to run both models same time. In the end, they are deciding the event based on the outcome from two models of whichever having higher accuracy. They are relying on external tweet text in addition to image features.

Yuanjun Xiong et al. [18], introduced complex event recognition model through the fusion of deep learning channels to recognize the different aspects of images to study event in the image. They proposed the deep multi-layer framework to tackle the problem of capturing the visual features and interaction among humans and objects and combine them to semantic fusion for event recognition. This requires a lot of deep layers to learn different aspects of a single network which takes a lot of time to train and even takes time to predict the event.

In the work did by Unaiza Ahsan et al. [19], they proposed to leverage concept-level representations for complex event recognition with few training examples. Initially, they built an event concept model which they got from the external source of flicker tags and Google Wordnet [39]. This model identified the list of significant terms for each event. They generated the feature vector (text and score) with a logistic regression classifier using CNN-Fc7 layer output for each concept of each event. After training all concept classifiers, the feature vector feeds to support vector machine to predict the event. This model involved a lot of complicated steps and Machine Learning algorithms required to train it and also relying on the external source for text semantics which is not required in our proposed model.

In the work proposed by Chuang Gan et al. [20], they introduced the video event detection using keyframes. They proposed a deep CNN network, simultaneously distinguishes pre-characterized events and provides key spatial-temporal evidence instead of extracting features with Shallow Learning techniques. Extracting the keyframes as input, recognizing the event of interest at the video level by conglomerating the CNN features of the keyframes. Again, this process requires heavy computational resources and will take a longer time to train the models.

The work of Dan Xu et al. [21] on video event detection presented an unsupervised deep learning framework. They introduced the motion and appearance features to automatically learn feature semantics and proposed a new approach of late fusion and early fusion to combine both appearance and motion features. They used autoencoders for early fusion and later multi-class SVM for late fusion to predict the event. This involved a lot of steps and algorithms required to compute the event.

The proposed work done by Xiaojun Chang et al. [22] aims to detect complex events in the video. They determine the semantic saliency score for each shot to relate each keyframe for the specified event and then prioritize shots based on saliency score which contribute to final event detection. Skip-gram model used to calculate a relevance and probability vector which combined to yield semantic saliency score. This score is used for prioritizing the shots. This whole process makes the system more heavyweight and complex to run near real time.

In the work did by Shuang Wu et al. [36], they proposed multi-modal fusion technique for event detection. Leveraging the multi-modal features of fusing the audio features, visual features, and event text descriptors which comes from the external source. Extracting each feature as a separate process and combine the all features as fusion vector to train the event detection model. This model relying on additional features of audio and event concepts to detect the event.

Gregory K. Myers et al. [37] introduced the multimedia event detection framework (SESAME) to solve the problem of heterogeneous content in images and video. SESAME uses a bag of words of a technique of multiple features of event classifiers based on image type. Those features include visual, motion and audio features. They fusion the all event detection scores from different classifiers and decide the event based on the final score.

The fusion of visual features and acoustic features introduced by Vijayakumar et al. [38], they proposed the method for cricket event detection using fusion features approach. On the low level, they are extracting the visual, motion, and audio features. After that, they are applying a specific filter to take only color features from visual aspects, motion vectors from motion features and audio features of MFCC, ZCR from audio. They applied the heuristic rule-based approach to these features to detect the event.

Table 1: Comparison of Different Works on Event Detection

	Distributed framework	Event classification with text	Hybrid Model (DL+SL)	External Source for text semantics	Mobile/Cloud Platforms	Fusion (Text + Image)
Samar M. Alqhtani (2015) et al. [17]	No	Yes	No	Yes	No	Yes
Yuanjun Xiong (2015) et al. [18]	No	No	No	No	No	No
Unaiza Ahsan (2017) et al. [19]	No	Yes	Yes	Yes	No	Yes
Chuang Gan (2015) et al. [20]	No	No	No	No	No	No
Dan Xu (2015) et al. [21]	No	No	Yes	No	No	No
Xiaojun Chang (2015) et. al [22]	No	No	Yes	No	No	No
MDRED (our model)	Yes	Yes	Yes	No	Yes	Yes

Table 2: Comparison of Different Works on Fusion Technique

	Fusion Text Features	Fusion Image Features	Fusion Image+ Text Features
Shuang Wu (2014) et al. [36]	No	No	Yes
Gregory K. Myers (2014) et al. [37]	No	Yes	No
Unaiza Ahsan (2017) et al. [19]	No	No	Yes
Yuanjun Xiong (2015) et al. [18]	No	Yes	No
Vijayakumar (2012) et al. [38]	No	Yes	No
MDRED (our model)	No	No	Yes

CHAPTER 3

PROPOSED WORK

3.1 Introduction

This chapter gives more details about the architecture and the components used in the Event detection system. As discussed in Chapter 1, the proposed framework has the following components:

1. Hybrid ML Model

    1.1 Lower Level- Deep Learning (DL) Model

    1. Distributed Framework for multi-task/multi-modality
    2. multi-task/multi-modality: Object detection, Color detection, Emotion detection, Faces detection and etc.

    1.2 Higher Level- Shallow Learning (SL) Model

    1. Use the text outcomes from DL models of the lower level
    2. Fusion Feature Vector: Extract the features with TF-IDF algorithm
    3. Optimized features with weighted factor
    4. Classify events using SL models

2 Cloud-Edge Computing

1. Cloud ML Servers
2. Master/Slave Architecture: Distributed ML with Mobile Devices

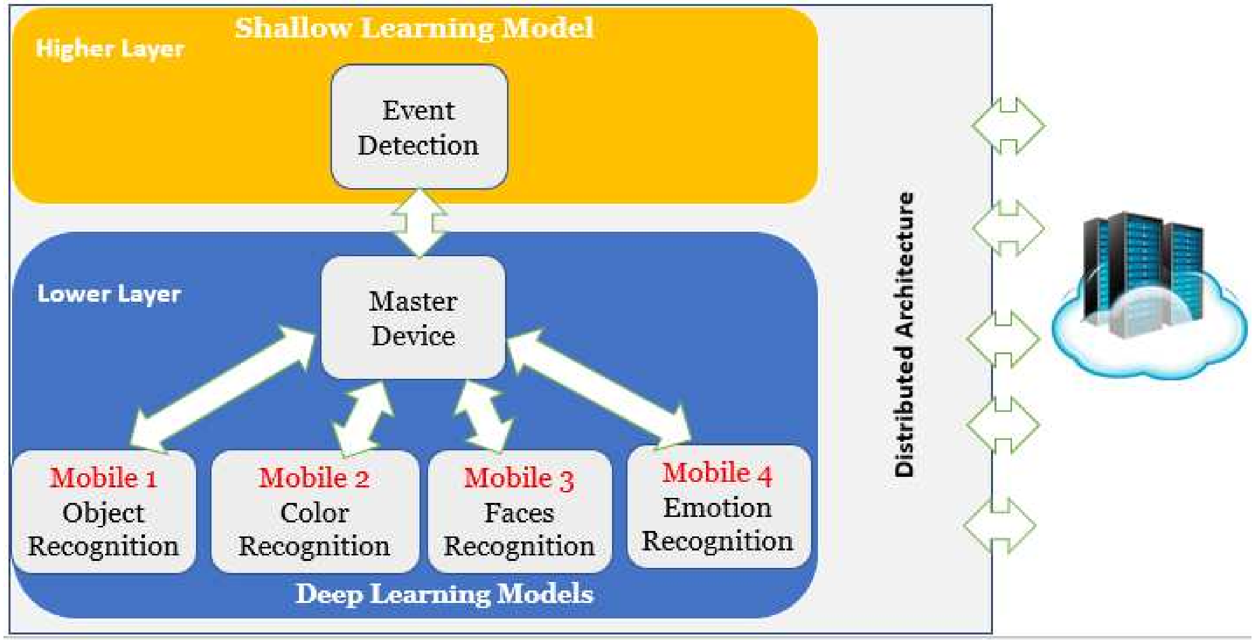


Figure 7: Proposed Framework

In further sections, I will discuss all components in detail.

3.2 Hybrid Machine Learning Model

3.2.1 Lower Level

In the lower level, a distributed framework was implemented where each edge device is responsible for the particular task. The tasks include object recognition, color recognition, emotion recognition, and faces recognition. It follows multi-task/multi-modality recognitions technique to identify the image attributes. The models we used for recognition are cloud deep learning models. Figure 8 depicts the example of multi-task/multi-model recognitions.

Multi-Task: Doing more than one task at the same time. In this context, dealing with multiple recognitions at the same time to identify different image attributes.

Multi-Modality: Multimodality describes the use of multiple models collectively to a common goal. In this context, it is used deep learning models such as an object, color, emotion, faces for every single image to identify the all possible Image attributes.

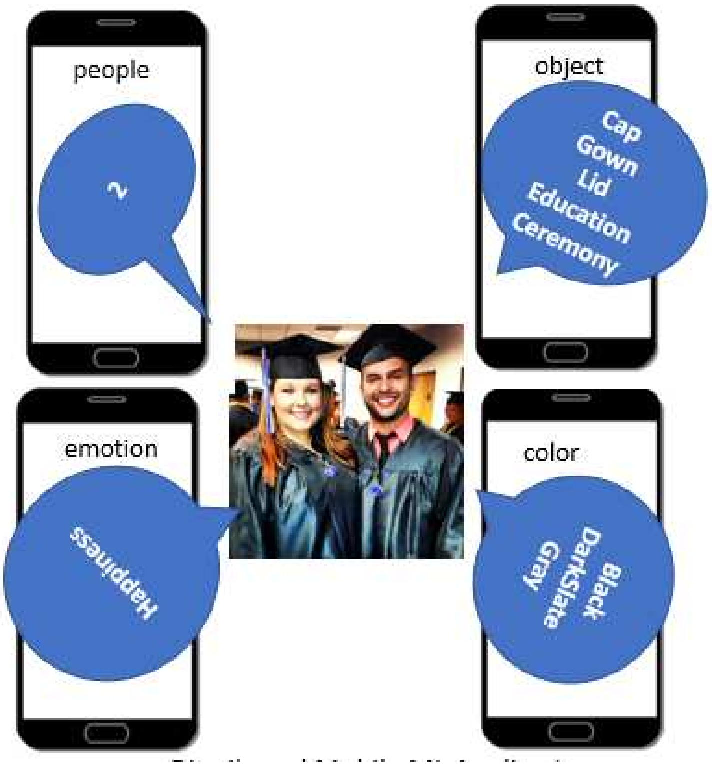


Figure 8: Multi-model/Multi-task Recognitions

3.2.2 Higher Level

In higher level, the Shallow Learning event detection model was implemented. The higher layer relying the outcomes from the lower level. The resulted text goes to data processing steps to extract features. Applied feature extraction technique of Term-Frequency and Inverse Document Frequency (TF-IDF) to identify the significant terms and implemented optimization technique of Weighted factor on top of TF-IDF values to distinguish unique terms and common terms. Once the feature vector was created from recognition results and then fusion the vectors as one to train with Shallow Learning models. After a trained model is ready, it will identify the event in an image.

3.3 Cloud-Edge Computing

The proposed framework prototype implemented in Master/Slave architecture with mobile devices. One mobile device acts master device and rest of mobiles act as slave devices which are connected to master device. The slave devices internally connected to cloud ML servers in a distributed manner. Each slave device responsible for each model recognition task. These slaves run in parallel to achieve the common goal. Once recognitions did from slave devices, the result returned to the master device and after collecting the result, the master device sends to event detection server to predict the event. Intern this result sent back to Master device.

3.4 Event Detection Architecture

An overview of the architecture is shown in Figure 9. The Event detection present in the higher layer. The higher level is based on the SL-based fusion techniques for the event detection based on the outcomes from the lower level DL models. The fusion model is designed as a weighted feature vector generated by a modified Term Frequency and Inverse Document Frequency (TF-IDF) algorithm, considering common and unique multi-modal features that are recognized for event detection. There are following steps involved steps which shown in figure 10 in the event detection system.

1) Multi-task/Multi-modality Recognition
2) Generate datasets as documents
3) Fusion Feature Vector
4) Optimization (Weighted Factor)

I will discuss in detail in the next sections for each step.

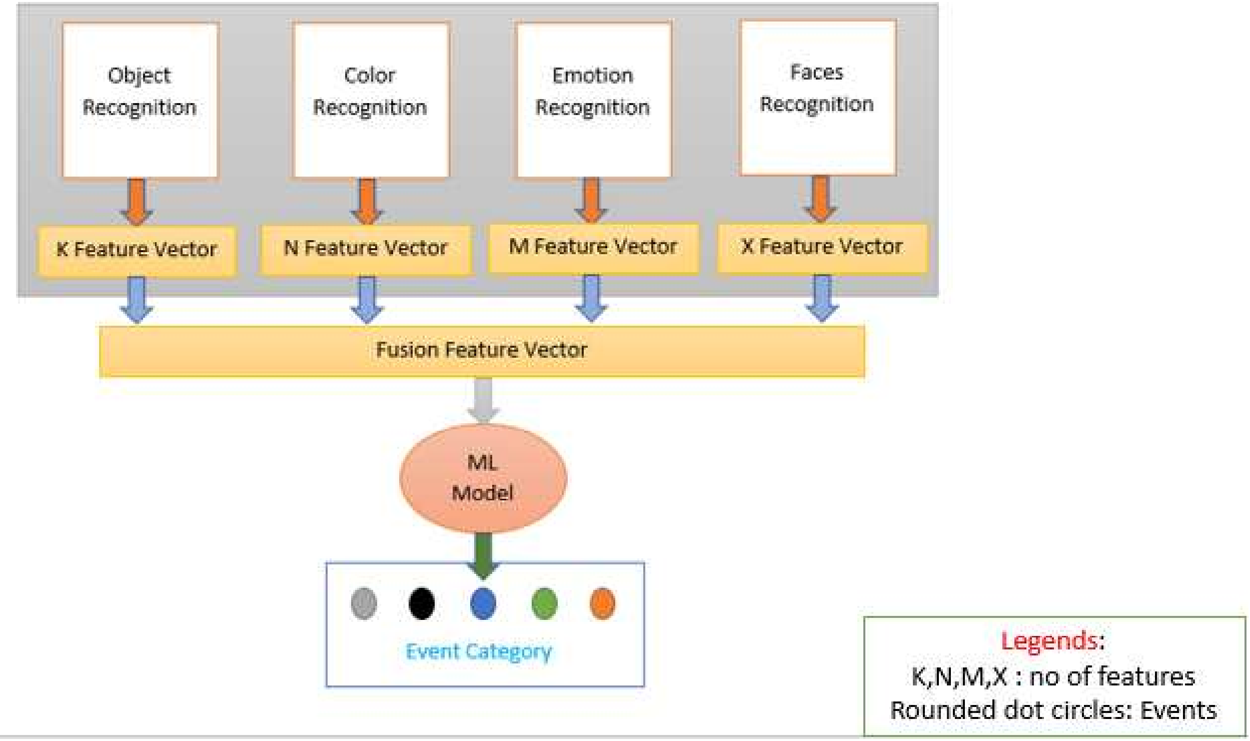
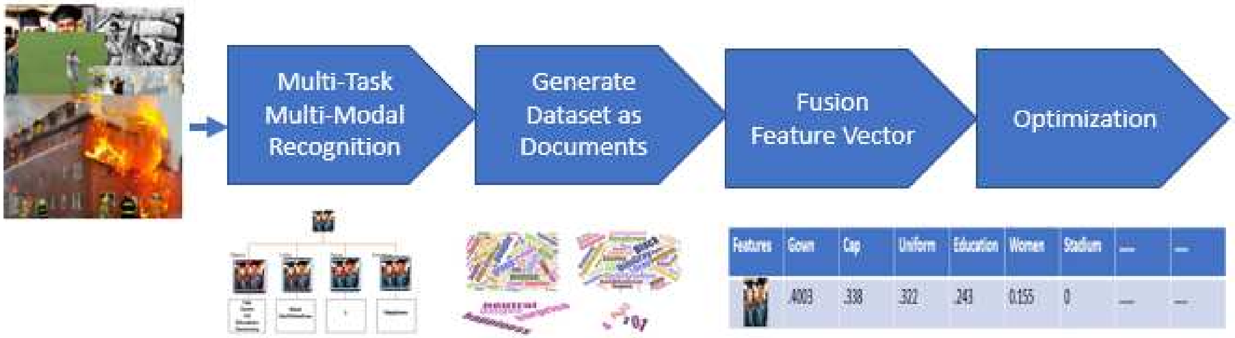


Figure 9: Event Model Architecture


Multi-Task Multi-Modal Recognition
Generate Dataset as Documents
Fusion Feature Vector
Optimization

Features	Gown	Cap	Uniform	Education	Women	Stadium	—	—
	.4003	.338	.322	.243	0.155	0	—	—

Figure 10: Data Preprocessing Step

3.4.1 Multi-task/Multi-modality Recognition

In this step, the input is image and do the recognition to identify all Image attributes. Divided each task based on a type of recognition. These recognitions are each individual Deep Learning model. The outcome of this step is multi-modality recognitions text results. Those are

1) Object Recognition- Identify the objects/scenes/actions in Image
2) Color Recognition- Identify the color property in Image
3) Faces Recognition- No of faces identify in Image
4) Emotion Recognition- Identify the feeling of persons in Image

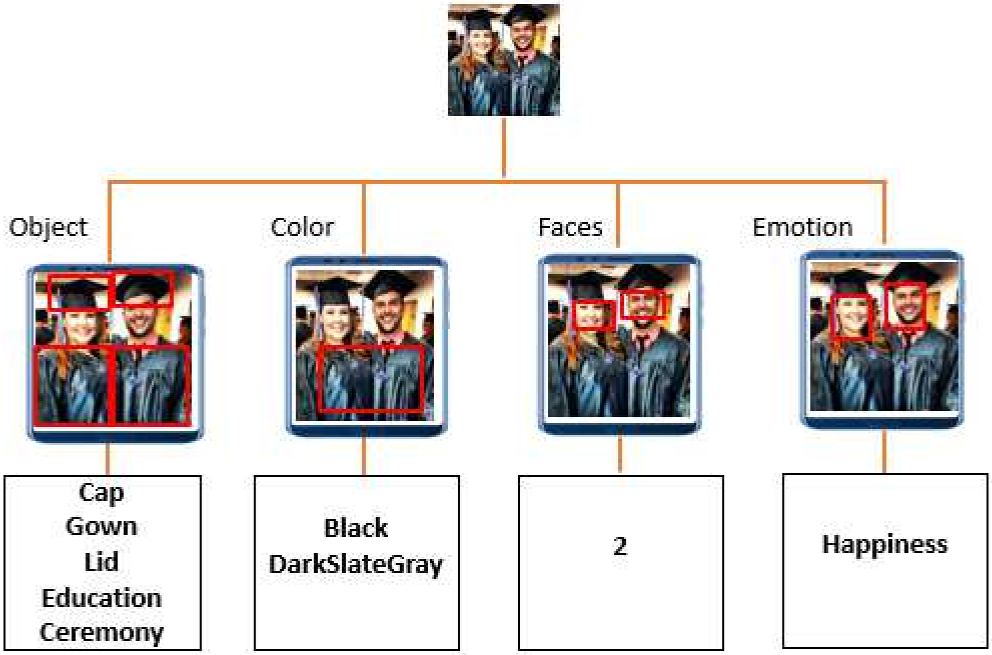
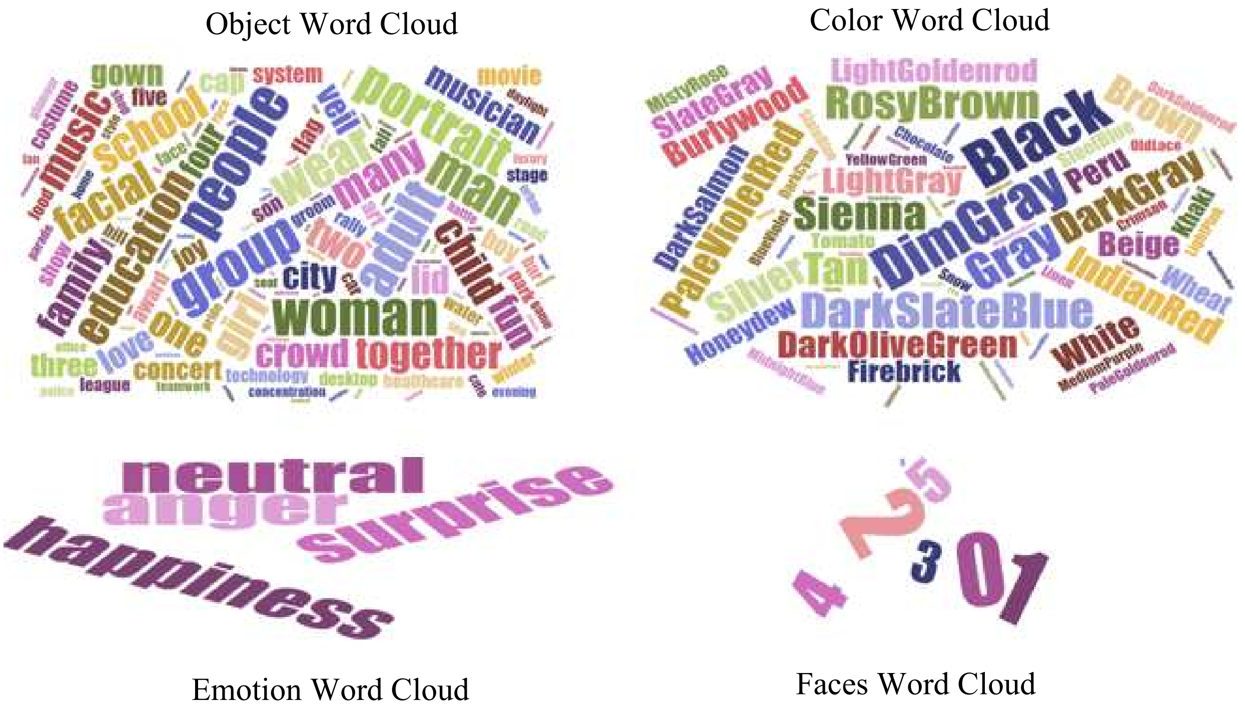


Figure 11: Multi-task/Multi-modality Recognition Results

Figure 11 depicts the example output for a Graduation picture. These results obtained based on multi-modality recognition models we choose.

3.4.2 Generate Dataset as Documents

In this step, the input is text result that is coming from the previous step. The text result has combinations of objects, colors, emotion, and faces. Each image has four category results and created a dataset where each row is each image result and columns are image recognition results. Figure 12 shows a word cloud for each recognition result and the result is a combination of those words.


Object Word Cloud

Color Word Cloud

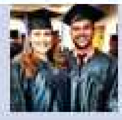
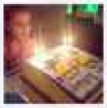
Emotion Word Cloud

Faces Word Cloud

Figure 12: Word Cloud for Multi-modality Recognition Results

The dataset created for each image based on results get from the previous step. Table 3 shows an example dataset for two images. This kind of dataset uses in the next step for creating a feature vector.

Table 3: Example Dataset

	Object Modality	Color Modality	Face Modality	Emotion Modality
	education college gown cap lid woman uniform	Black DimGray Brown	2	Happiness
	candle child cake Light celebration	WhiteSmoke white Brown	1	Neutral

3.4.3 Fusion Feature Vector

Each multi-modality recognition text results follows feature extraction techniques to create fusion feature vector. We used Term frequency and Inverse document frequency (TF-IDF) to create a feature vector. First, we calculate the term frequency and do the Inverse document frequency, then we compute TF*IDF to get unique features and common features. This same process will be applied to each model and at the end combine all feature vectors as one that will be used to train with Shallow Learning model. Figure 13 shows the general steps involved from dataset processing to training the model. Initially, we get training data and preprocess the data for word segmentation. Once we have individual terms and then apply TF-IDF technique to extract features, this feature vector will be used to train the model. Example TF-IDF process described below

a) Tokenization:

In this process, the input text in each document converted into words and tokens. This process is known as lexical analysis.

Input sentence: “education college gown cap lid woman uniform”

Output words: [education, college, gown, cap, lid, woman, uniform]

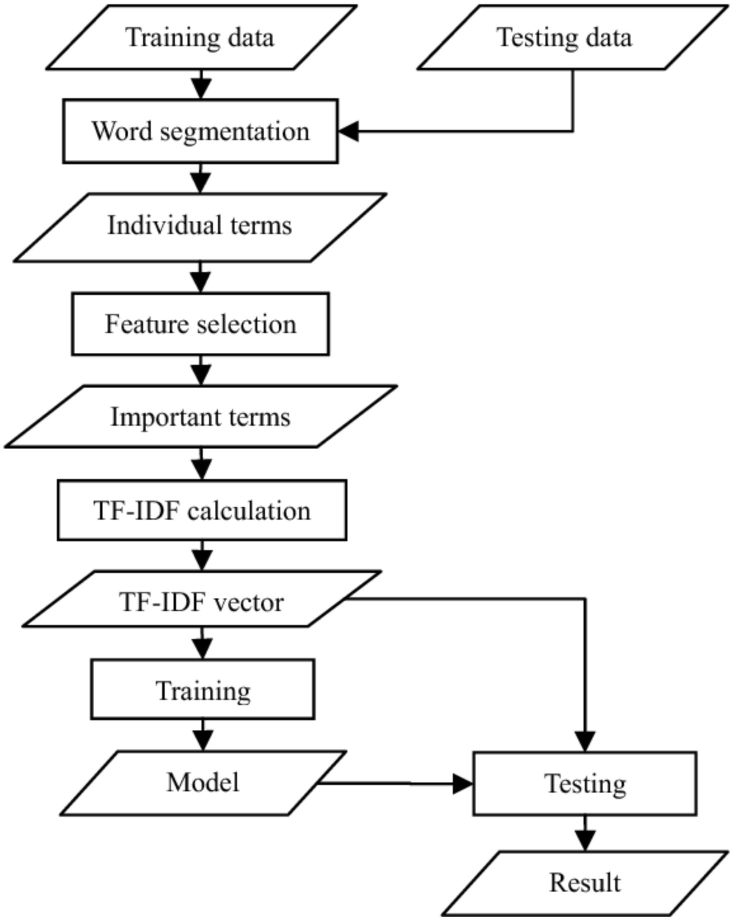


Figure 13: General Steps for training with TF-IDF Features

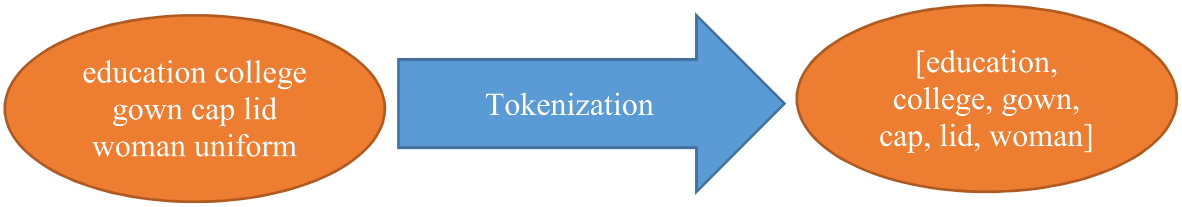

education college gown cap lid woman uniform
Tokenization
[education, college, gown, cap, lid, woman]

Figure 14: Tokenization

b) Term Frequency and Inverse document frequency (TF-IDF):

TF-IDF gives a weight of term that is often used in the document and this weight calculate by statistical methods to evaluate the importance of the term in documents. The weight tells you how many times term appeared in documents and offset by how many times it has appeared in the corpus.TF-IDF calculation is shown below:

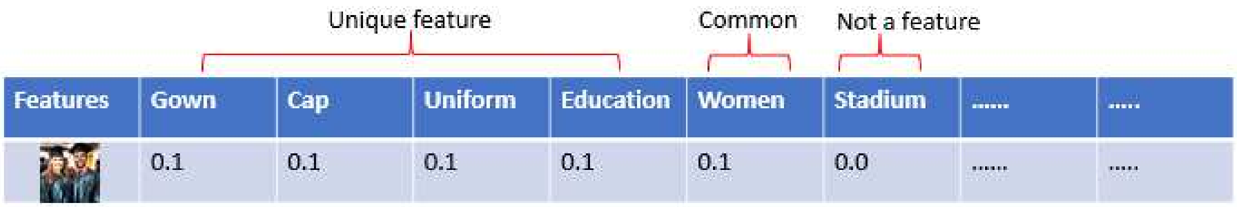
1) Term(t) Frequency (TF):

It tells you how frequently term appeared in a document. Since each document length varies with a number of terms, it is devised with a number of terms in the document.

The formula TF(t) = (No of times term t appears in a document) / (Total number of terms in the document).

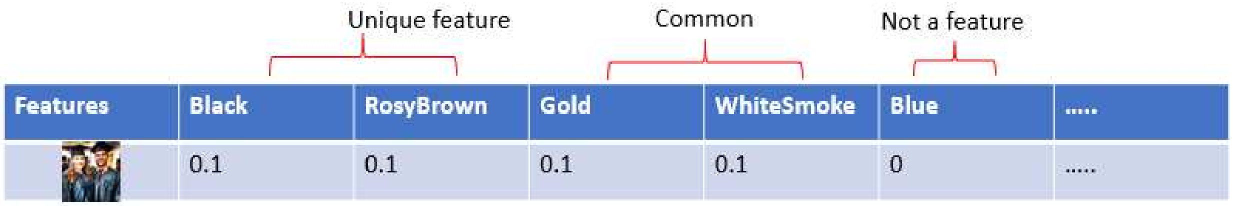
Tables 4-7 show the example TF values calculated based on the formula for each category data that is for the object, color, faces and emotion results respectively.

Table 4: TF Calculation for Object Results


Unique feature
Common
Not a feature

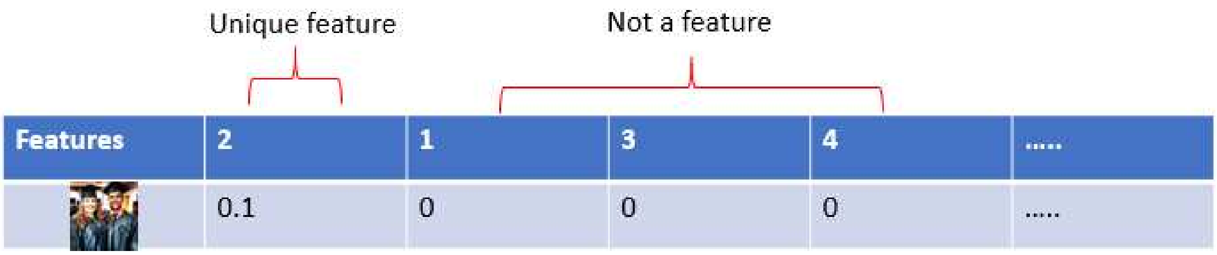
Features	Gown	Cap	Uniform	Education	Women	Stadium	......	.....
	0.1	0.1	0.1	0.1	0.1	0.0	......	.....

Table 5: TF Calculation for Color Results


Unique feature
Common
Not a feature

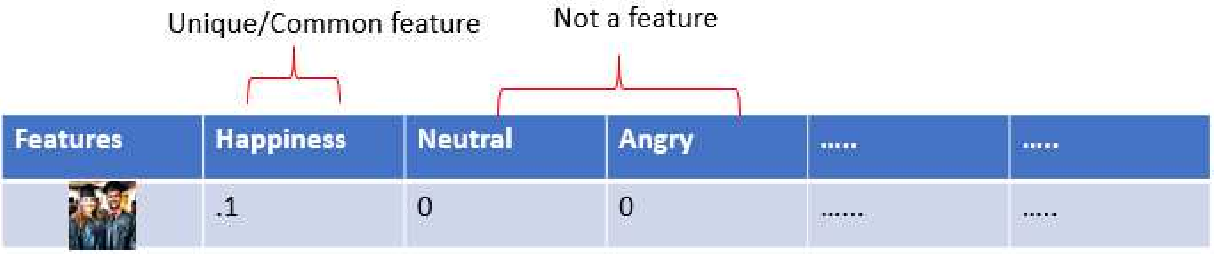
Features	Black	RosyBrown	Gold	WhiteSmoke	Blue	.....
	0.1	0.1	0.1	0.1	0	.....

Table 6: TF Calculation for Faces Results


Unique feature
Not a feature

Features	2	1	3	4	.....
	0.1	0	0	0	.....

Table 7: TF Calculation for Emotion Results


Unique/Common feature
Not a feature

Features	Happiness	Neutral	Angry	.....	.....
	.1	0	0	......	.....

2) Inverse Document Frequency (IDF):

It gives how important the term is. TF generally gives you all terms are equally important. Some of the terms like "is", "are" "of" appeared all most all documents. To reduce the important those terms we need to calculate the IDF values. The IDF value is high if term appeared in some of the documents and IDF value is low if term appeared all of the documents or few of the documents.

The formula for IDF(t) = log_e (Total no of documents / No of documents with term t in it).

Tables 8- 11 show the example IDF values calculated based on the formula for each category data that is for the object, color, faces and emotion results respectively.

Table 8: IDF Calculation for Object Results

Features	Gown	Cap	Uniform	Education	Women	Stadium	......	.....
	4.003	3.38	3.22	2.43	1.55	3.63	......	.....

Table 9: IDF Calculation for Color Results

Unique feature Common Not a feature

Features	Black	RosyBrown	Gold	WhiteSmoke	Blue	.....
	3.8293	2.977	1.66	1.42	2.5	.....

Table 10: IDF Calculation for Faces Results

Unique feature Not a feature

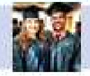
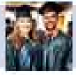
Features	2	1	3	4	.....
	2.1	4.7	1.28	0.5	.....

Table 11: IDF Calculation for Emotion Results

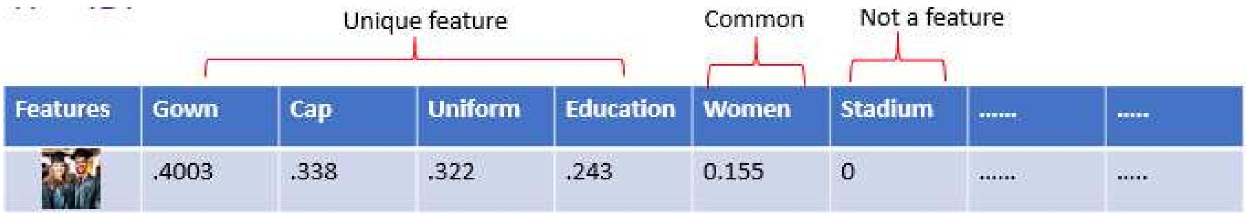
Unique/Common feature Not a feature

Features	Happiness	Neutral	Angry	.....	.....
	5.548	3.5	4.4	......	.....

3) TF- IDF:

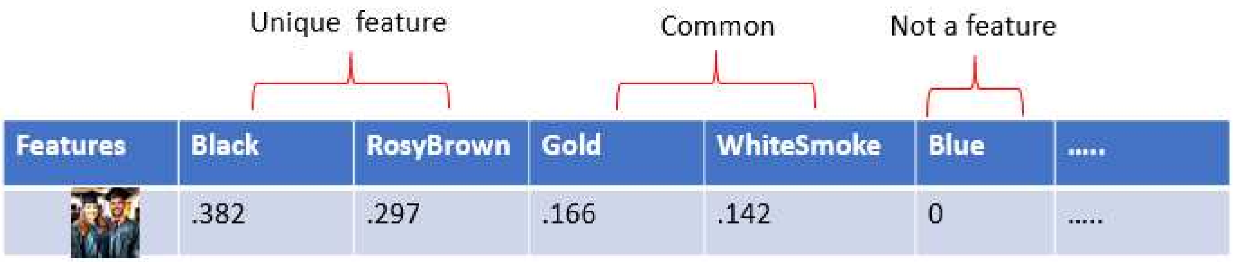
Once the TF and IDF values calculated separately, then TF multiply with IDF gives TF-IDF feature vector. TF-IDF values tell us a weight of terms. Table 12 shows an example feature vector for Graduation Image. The unique terms have a higher weight. So "gown", "cap", "uniform" has higher values for graduation and "women" is a common term which appeared many events has a lower weight. "Stadium" does not feature for Graduation which is why weight is 0. The same analogy applied to Tables 13-15 which show the color, faces, and emotion TF-IDF vector of unique and common terms respectively.

Table 12: TF-IDF Calculation for Object Results


Unique feature
Common
Not a feature

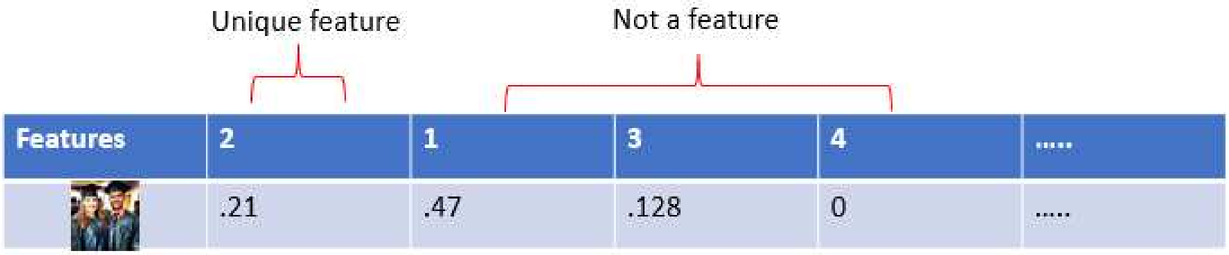
Features	Gown	Cap	Uniform	Education	Women	Stadium	......	.....
	.4003	.338	.322	.243	0.155	0	......	.....

Table 13: TF-IDF Calculation for Color Results


Unique feature
Common
Not a feature

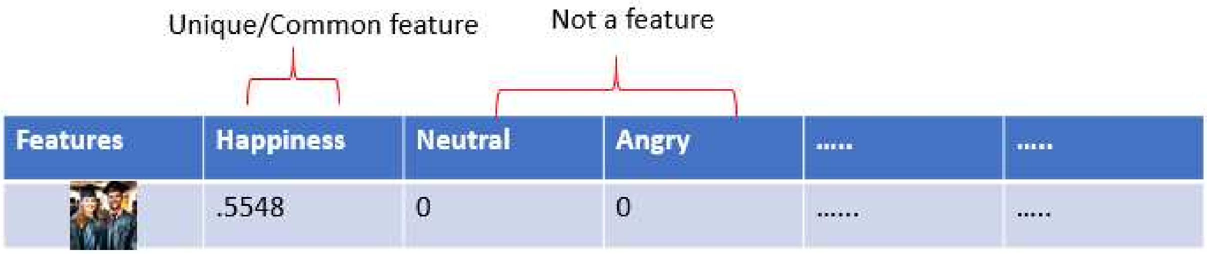
Features	Black	RosyBrown	Gold	WhiteSmoke	Blue	.....
	.382	.297	.166	.142	0	.....

Table 14: TF-IDF Calculation for Faces Results


Unique feature
Not a feature

Features	2	1	3	4	.....
	.21	.47	.128	0	.....

Table 15: TF-IDF Calculation for Emotion Results


Unique/Common feature
Not a feature

Features	Happiness	Neutral	Angry	.....	.....
	.5548	0	0	......	.....

c) Fusion Feature Vector:

For every recognition, text result follows the same process of taking input a document, convert into tokens and apply TF-IDF feature extraction technique to create feature vector which consists of unique, common and not a feature in it. This is done for the same for every other recognition for color, emotion, and faces for each image results. At the end, all feature vectors will be combined as a single feature vector, which consists of object, emotion, faces, and color features.

Table 16: Fusion Feature Vector

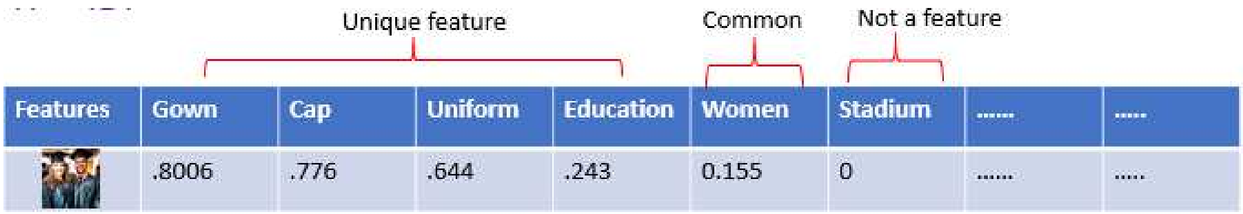
Object features	Color Features	Emotion Features	Faces features
1150 features **(education, cap, adult, gown, flame, smoke, rebellion, heat, battle, sport, game, match, goal, ball, candle, child, family, cake, bride, woman, fashion, love, music, people performance, musician, singer, guitar, competition runner, race, track, etc.)**	**100 features** **(Gold, RosyBrown, WhiteSmoke, Black, White, Green, Gray, DimGray, SaddleBrown, Silver, etc.)**	**6 features** **(happiness, angry, neutral, fear, surprise, sadness)**	**10 features** **(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)**

3.4.4 Optimization

The optimization used to reduce the confusion between images by making insignificant common terms and significant unique terms. This is done by modifying the TF-IDF values. The TF-

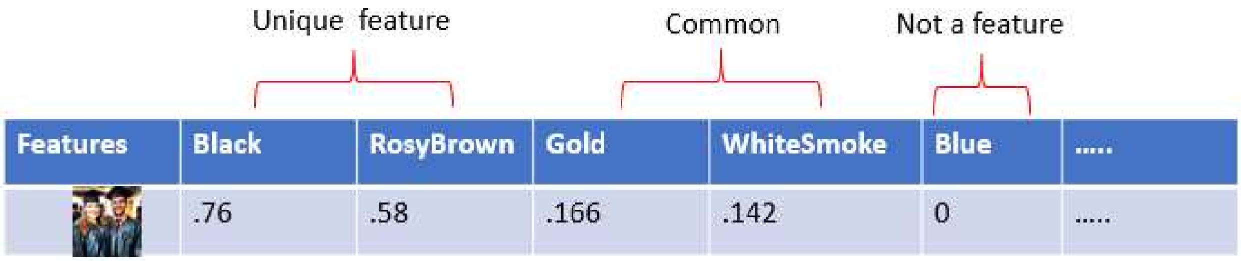
IDF values changed based on weighted factor (W) approach. Initially identified top k unique features based on higher TF-IDF values. Multiply the TF-IDF values with value W to increase the weight of unique terms for making significant terms. So that we get huge value difference between unique terms versus common terms to make difference. Since we can't deice what value of w has better accuracy, followed heuristic approach to decide the value of w. The updated TF-IDF score for K features = TF-IDF (K- features) *w. Tables 17-20 show the example TF-IDF score after updating values with W=2 when k=3 for each category data.

Table 17: Updated TF-IDF Values for Object Results


Unique feature
Common
Not a feature

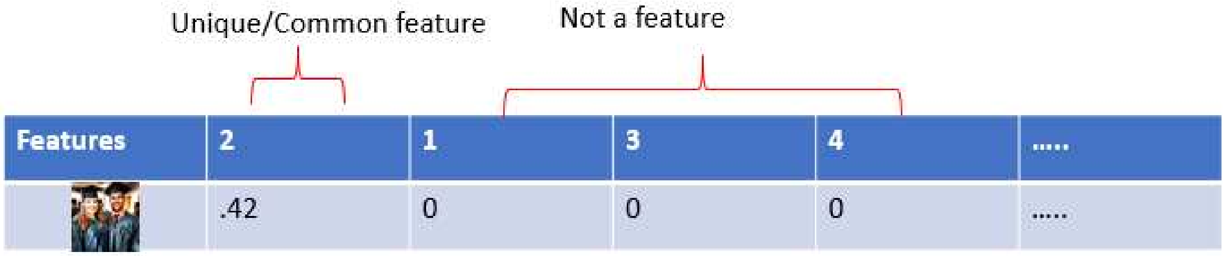
Features	Gown	Cap	Uniform	Education	Women	Stadium	……	…..
	.8006	.776	.644	.243	0.155	0	……	…..

Table 18: Updated TF-IDF Values for Color Results


Unique feature
Common
Not a feature

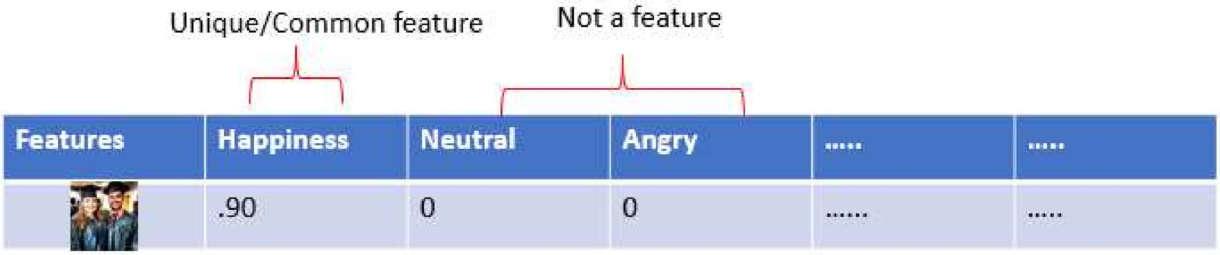
Features	Black	RosyBrown	Gold	WhiteSmoke	Blue	…..
	.76	.58	.166	.142	0	…..

Table 19: Updated TF-IDF Values for Faces Results


Unique/Common feature
Not a feature

Features	2	1	3	4	…..
	.42	0	0	0	…..

Table 20: Updated TF-IDF Values for Emotion Results


Unique/Common feature
Not a feature

Features	Happiness	Neutral	Angry	.....	.....
	.90	0	0	......	.....

Figure 15 shows the flow diagram of how weight factor(W) was decided. The W value is initialized to 2 and the threshold value is assigned as accuracy, which was obtained from the original TF-IDF vector. The W is iterated till it becomes 10 and at each iteration identifying the top K unique features. Once we get those features to multiply TF-IDF (K features) with W, we get the updated TF-IDF vector. The ML model will be trained with the updated feature. The accuracy will be checked with the threshold accuracy. If it is greater than the threshold value, then we update the threshold value with current accuracy and save the model. We repeat the same process till it becomes 10. At the end of the last iteration, we will have a better-saved model which gives a higher accuracy. Figure 16 shows the pseudo-algorithm for a heuristic approach. The same steps we followed is described in the flow diagram to implement the weighted factor.

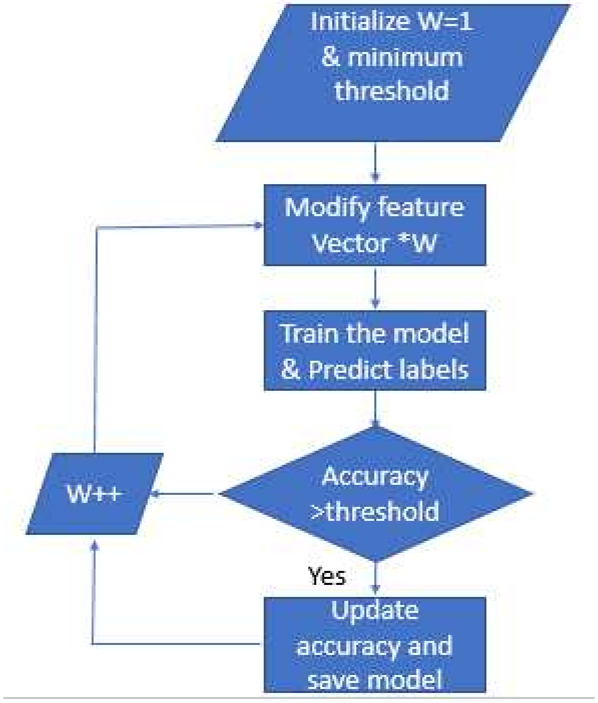
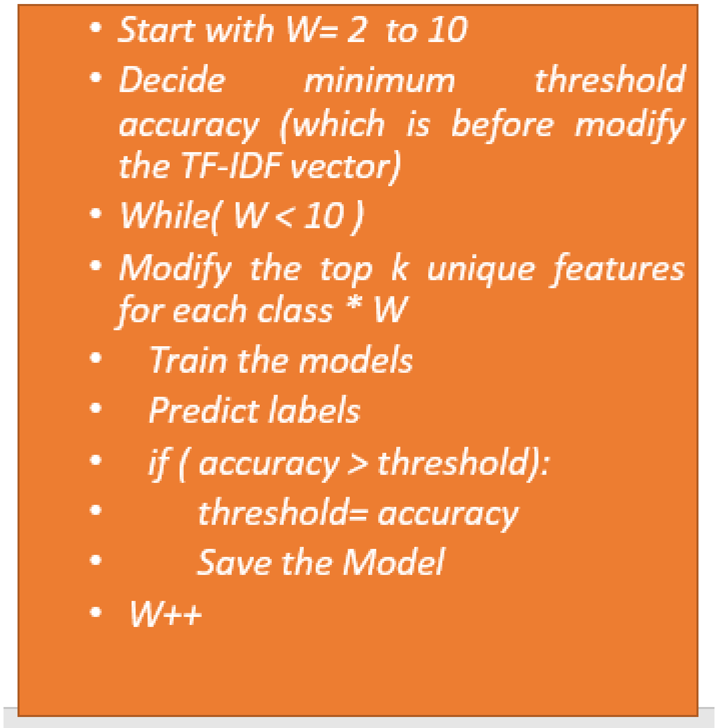

Initialize W=1 & minimum threshold
Modify feature Vector *W
Train the model & Predict labels
Accuracy >threshold
W++
Yes
Update accuracy and save model

Figure 15: Workflow for Optimization Technique



```
• Start with W= 2 to 10
• Decide minimum threshold accuracy (which is before modify the TF-IDF vector)
• While( W < 10 )
• Modify the top k unique features for each class * W
•    Train the models
•    Predict labels
•    if ( accuracy > threshold):
•        threshold= accuracy
•        Save the Model
• W++
```

Figure 16: Pseudo Algorithm for Optimization Technique

3.4.5 Event Classification

We use the following Shallow learning algorithms used to classify the event in images.

a) Naïve Bayes classifier [23]

Naïve Bayes classifier is based on the probabilistic model of Bayes theorem which assumes that features are independent of each other. It works well if the dimensionality of features is high. Naïve Bayes is extremely useful for text categorization methods. In this classifier, the label is decided based on probability it belongs. It will return the probabilities of each it belongs to and classifies the event based on the highest probability event.

Table 21: Example Naïve Bayes Classification

Event	Probability
Graduation	0.98
Wedding	0.21
Birthday Party	0.01

b) Random Forest [24]

Random forest used either for classification or regression tasks. Unlike another algorithm, the random forest grows like multiple decision trees. It uses multiple decision trees and merges them together to yield a better accuracy and stable predictions.

CHAPTER 4

RESULTS AND EVALUATION

4.1 Introduction

In this chapter, we used datasets to evaluate in our experiments, shown individual model metrics with an intuitive explanation. We will also look into details of the hardware and software configurations upon which the experiments were run.

4.2 Configuration

The details of the configurations used are as shown in Table 22,23

Table 22: System Configuration

Property	**Value**
Operating System	Ubuntu 16.04 LTS
Memory	32 GB
Processor	Intel® Xeon ® CPU E5-2603 v4 @ 1.0Ghz
GPU	GeForce GTX 1080Ti (4*11 Gb)
OS type	64- bit
Disk	1 TB

Table 23: Mobile Configuration

Property	Value
Operating System	Android 7.1.1
Memory	2 GB
CPU	Qualcomm Snapdragon 625
OS type	32- bit
Disk	16 GB

We used the following languages, libraries, and technologies

1. Android SDK’s
2. Java
3. Scala
4. Spark

4.3 Datasets

In this experiment, we have used three different datasets in our evaluation. Following datasets used for the implementation.

1) Social Event Image Dataset (SocEID) [25] dataset consist of some images from the NUS-WIDE dataset [26] and the Social Event Classification subtask from MediaEval 2013 [27]. It has 8 social events: protests, parades, soccer matches, birthdays, graduations, weddings, marathons/races, and concerts. The dataset has 27,718 training images and 10,100 test images. Figure 17 shows example images for the SocEID dataset.

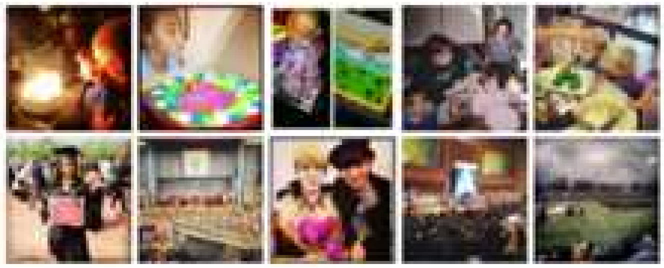


Figure 17: SocEID Dataset

2) UIUC Sports Event Dataset [28] contains 8 sports event categories: snowboarding (190 images), croquet (236 images), sailing (190 images), rowing (250 images), badminton (200 images), polo (182 images), bocce (137 images), and rock climbing (194 images). Total of 1579 images. Figure 18 shows example images for UIUC dataset.

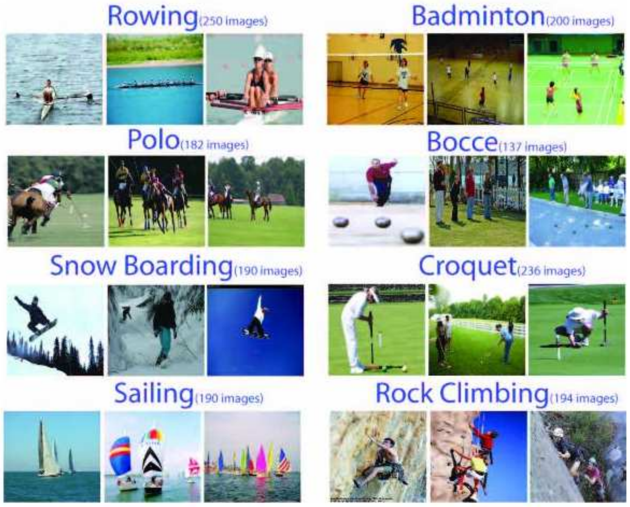


Figure 18: UIUC Sports Dataset

3) Rare Events Dataset (RED) [29] consist of 21 rare event categories. We call them not because of how often they happen in the world but because of how seldom they appear in datasets. These consist of natural disasters like Nepal earthquake, Hurricane Sandy and recent news such as Justin Trudeau elected, election campaign Trump. Figure 19 shows example images for UIUC dataset.

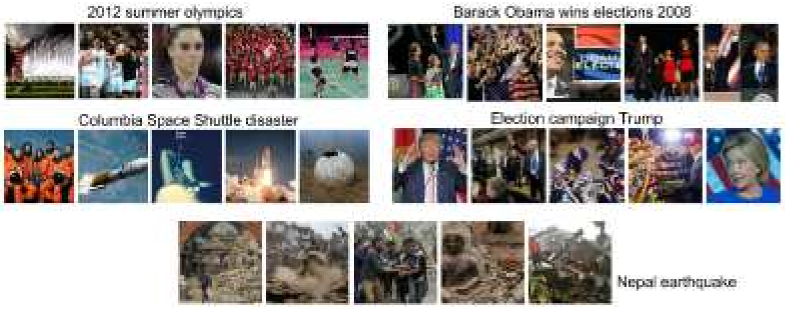

2012 summer olympics
Barack Obama wins elections 2008
Columbia Space Shuttle disaster
Election campaign Trump
Nepal earthquake

Figure 19: RED Events Dataset

4.4 Mobile Performance Evaluation

We are running multi-task/multi-modality recognitions in edge devices. We have evaluated the time taken from sending an image to ML server and send back the results to the device. Each mobile device is doing each recognition task. Figure 20 shows the prediction time for each recognition. Since all recognitions running parallelly on each device the overall prediction is the time which runs last. This case prediction time 270ms, 1000ms, 560ms, 920ms for object, color, face and emotion recognition respectively. The overall prediction time 1000ms which execute last. After doing multi-modality recognition we use the result to do event detection. The time taken for event detection is 500ms. So overall event prediction time is 1500ms (excludes network delay) that is the time image input to master device till it does event classification.

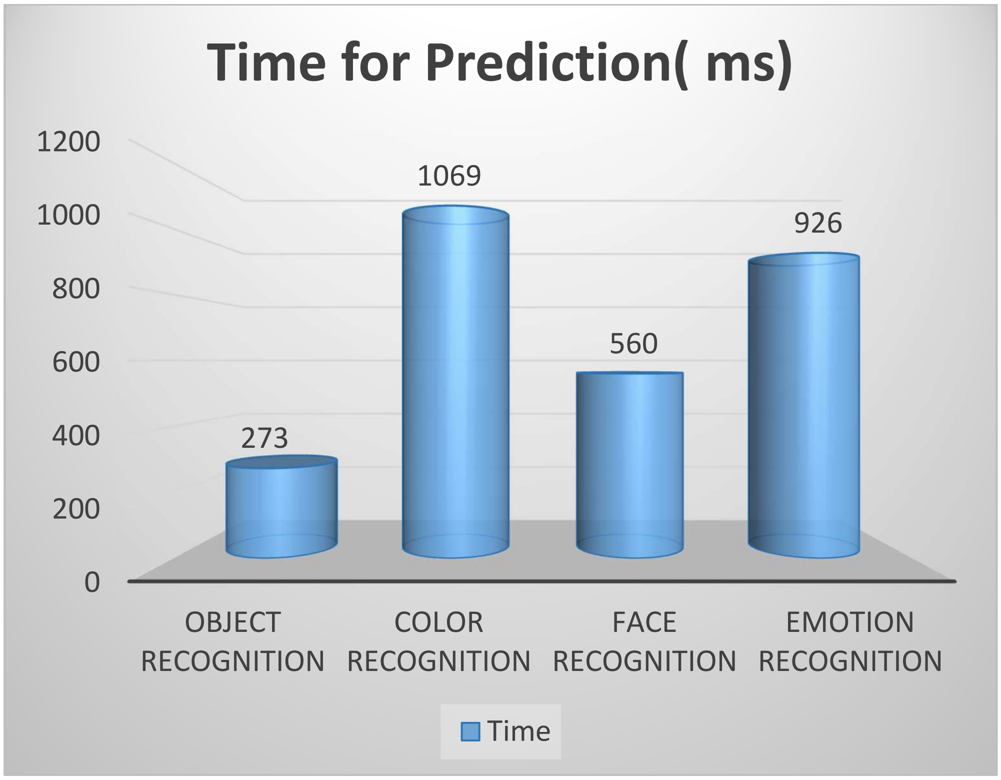


Figure 20: Prediction Time for Multi-Modality Recognitions

We also evaluated CPU utilization for each model recognition. Face recognition taking highest CPU utilization compared to other recognitions. Figure 21 shows the CPU utilization for each model recognition. Object, Color, Face and Emotion recognition taking 42%, 20%, 50%,32% CPU utilization respectively.

We also evaluated Memory utilization for each model recognition. Face recognition taking highest Memory utilization compared to other recognitions. Figure 22 shows the Memory utilization for each model recognition. Object, Color, Face and Emotion recognition taking 53MB, 48MB, 80MB,62MB Memory capacity respectively.

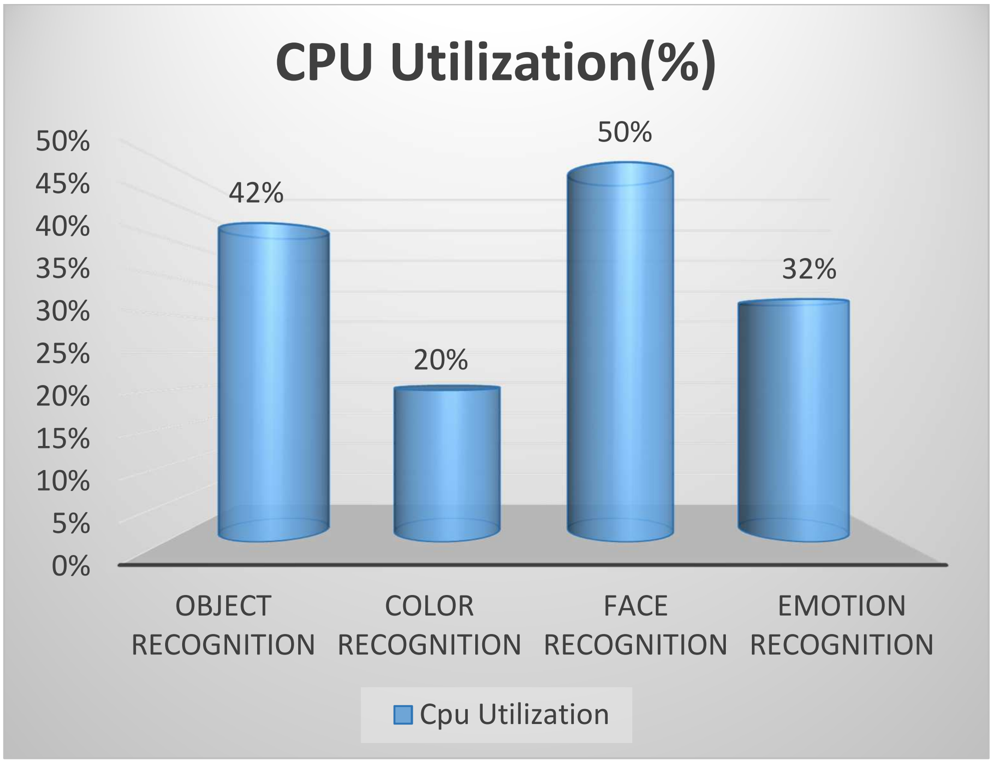


Figure 21: CPU Utilization for Multi-Modality Recognitions

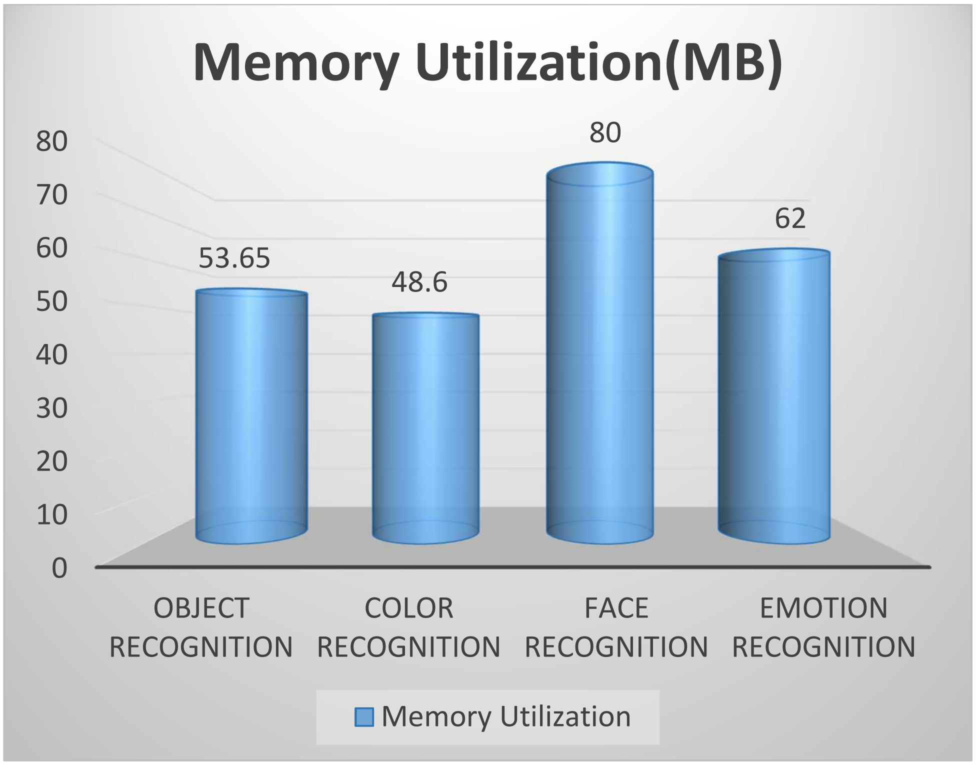


Figure 22: Memory Utilization for Multi-Modality Recognitions

4.5 Deep Learning Models Evaluation

We evaluated the lower level Deep Learning model's accuracy to understand event model accuracy. This accuracy varies from each individual Deep Learning models. We evaluated with benchmark dataset to show accuracy. Since event detection model accuracy relying on output from lower level results that are coming from Deep Learning models. So, if there is a change(α) in accuracy for Deep Learning models then It will impact (β) value in event detection accuracy.

Deep Learning models + α => Event detection model + β where α directly proportional to β.

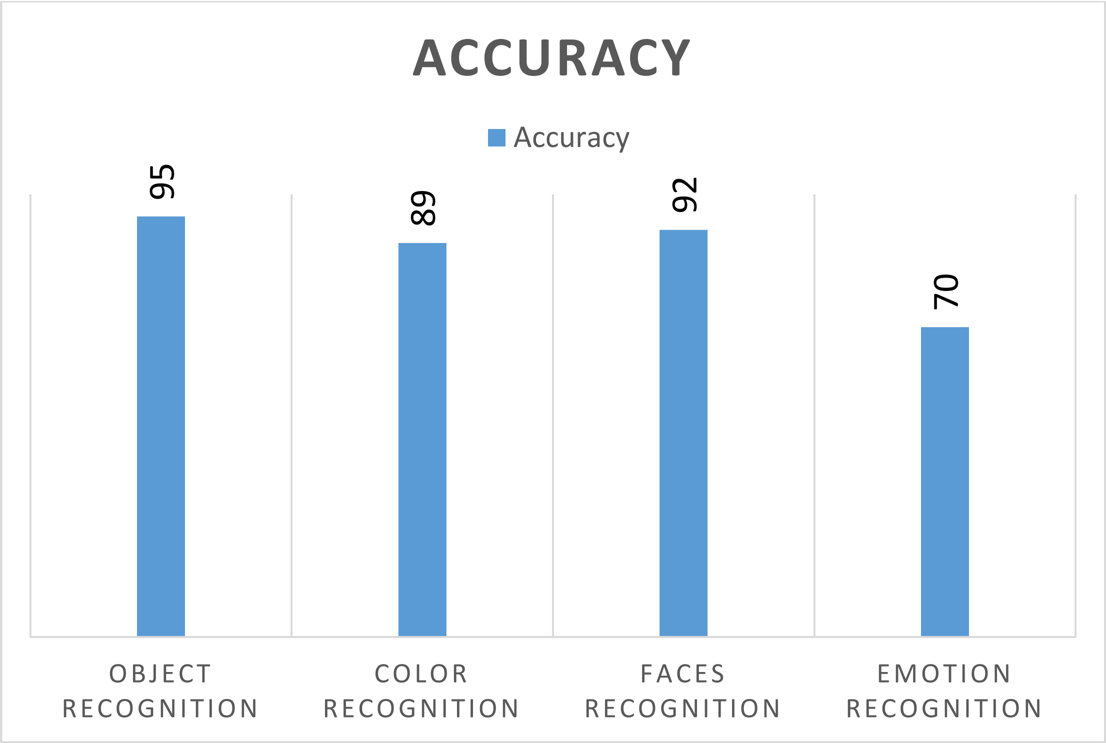


Figure 23: Deep Learning Models Accuracy

4.6 Features Size Evaluation

Since accuracy depends on feature size selection, we evaluated the accuracy with Naïve Bayes, Random Forest training model by varying the features size. In this case, we choose the feature dimensions ranging from 50 to actual features (1250) for SocEID dataset. For each feature selection, we evaluated with Naïve Bayes and Random Forest algorithms. Till 200 features Random Forest giving better accuracy over Naïve Bayes which is 79% maximum. After 200 features to 1250 features (Actual features) Naïve Bayes beating the Random Forest accuracy. In our evaluation, we found that Naïve Bayes achieves the best accuracy of 86.37% for our datasets. Figure 24 shows the feature size evaluation accuracy.

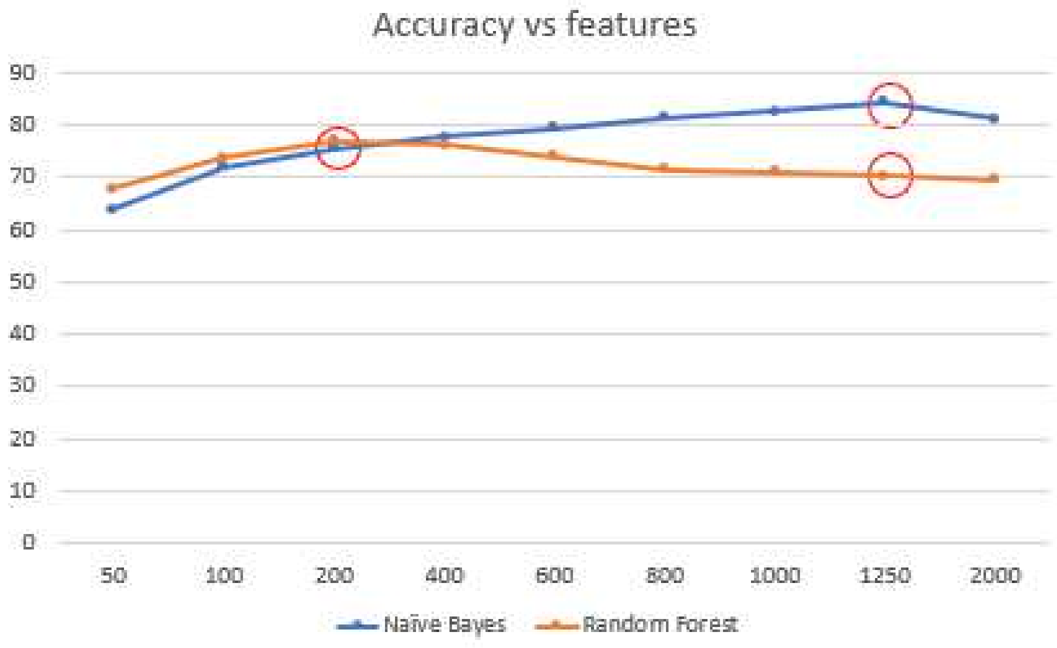


Figure 24: Features Size Evaluation

4.7 Weighted Factor Evaluation

In our training phase, we Introduced the optimization technique for modifying the TF-IDF values using a weighted factor approach. Since we can't decide the random number to multiply with top K unique features for TF-IDF values, we implemented the heuristic approach to decide the value range from 2 to 10. In our experiments, we evaluated accuracy for values ranging from 2 to 10. At weighted factor 2, the accuracy is high for the SocEID dataset. Figure 25 shows the accuracy varying based on the weighted factor. After value 2, there is a gradual decrease in accuracy because we are making it less significant than the other features(n-k).

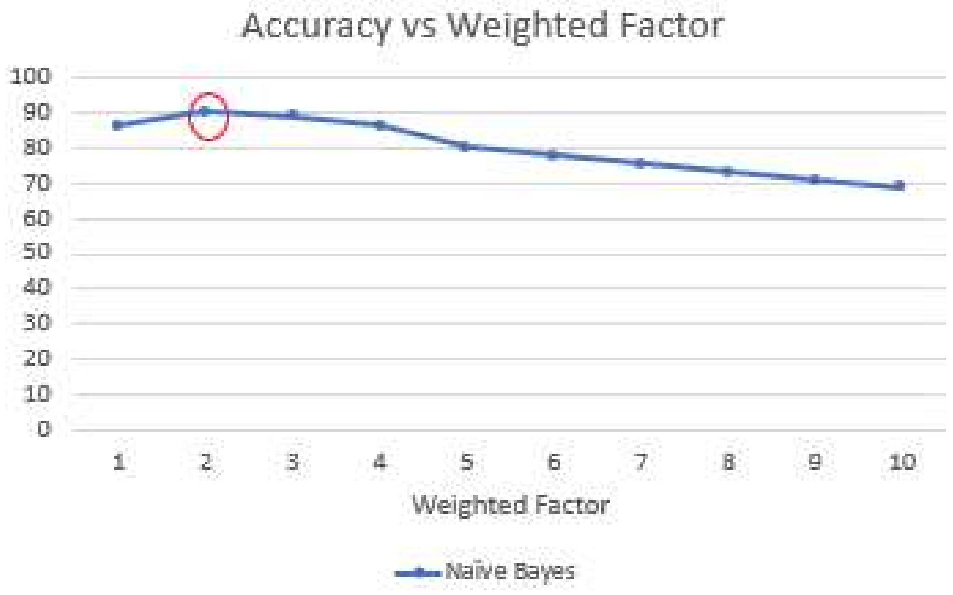


Figure 25: Weighted Factor Evaluation

4.8 SocEID Dataset Evaluation

The SocEID dataset evaluated with the different combination of model recognition results. The accuracy is changing with different combinations. If we use only the object recognition results and predict the event with those features, then the accuracy yields to 79.5%. But if we use all combinations with those features, then the accuracy was improved from 79.5% to 86.37%, which is the highest accuracy with the original TF-IDF features. Table 24 shows the accuracy of the different combinations which are also shown in Figure 26.

Table 24: SocEID Model Combinations Accuracy

Model combinations	**Accuracy**
Object	79.5%
Object + Face	80.3%
Object + Color	81.3%
Object + Emotion	83.6%
Object + Color + Face	80.9%
Object + Emotion + Face	81.1%
Object + Color + Emotion	83.8%
Object + Color + Emotion + Face	**86.37%**

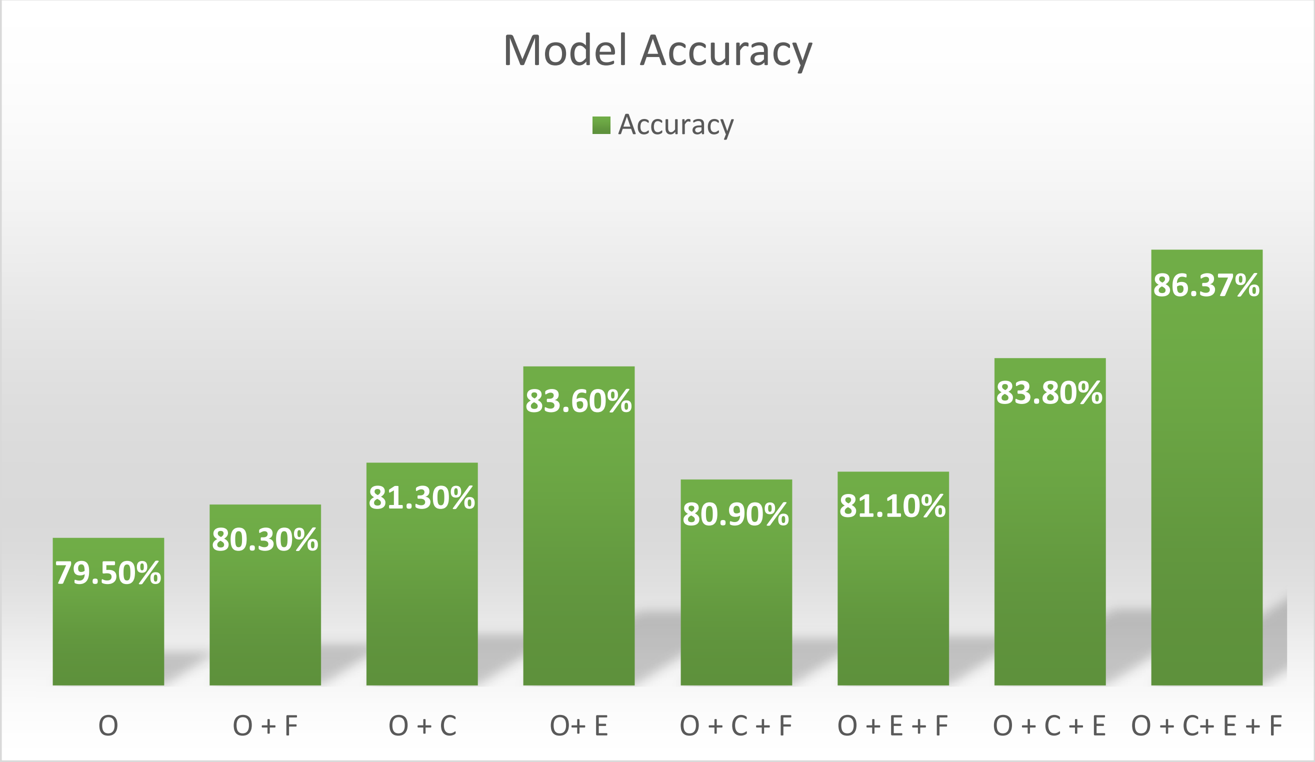


Figure 26: SocEID Model Combinations Accuracy

The model showed the higher accuracy of 86.37% when all multi-modal recognition was used in the feature vector to predict the event. We also evaluated the individual event accuracy from this higher accuracy model. Out of 8 events, 6 events have greater than 80% accuracy and rest two of them having 75% accuracy. Figure 27 shows each event accuracy for the SocEID dataset.

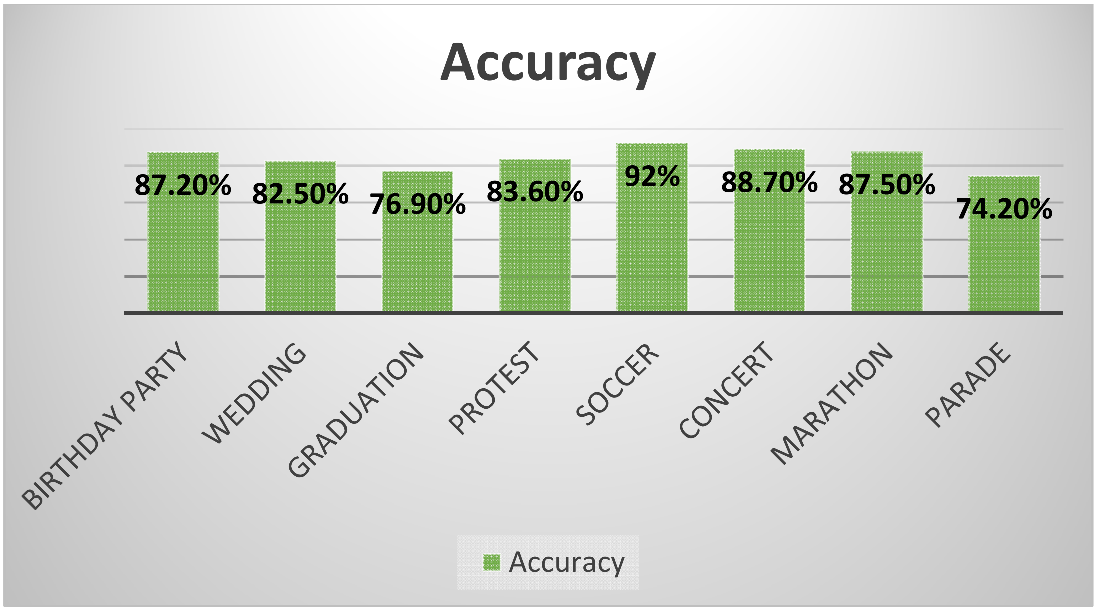


Figure 27: SocEID Dataset Events Accuracy

In the real world, images are skewed, not clear and highly confused each other. We found the same issue among Birthday_Party, Wedding and Graduation images which are confused each other. Figure 28 shows the example images which are confusing between wedding and graduation images because of similar object properties and color properties. The confusion matrix shown in Figure 29 gives a better understanding of the images, which are classified wrongly. The red block shows the number of images, which are wrongly classified. In this case birthday party, wedding and graduation have the higher number of wrongly classified events.

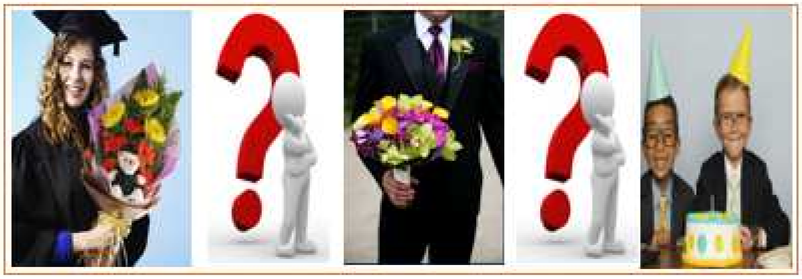
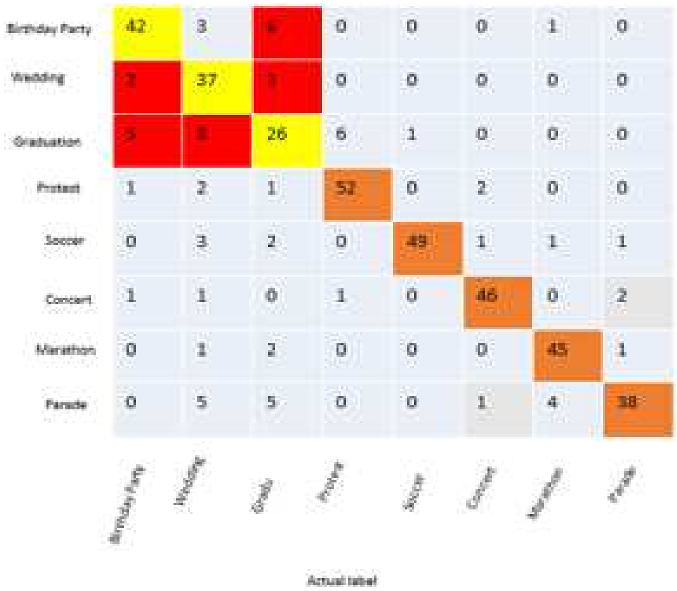


Figure 28: Confusion Images for SocEID Dataset

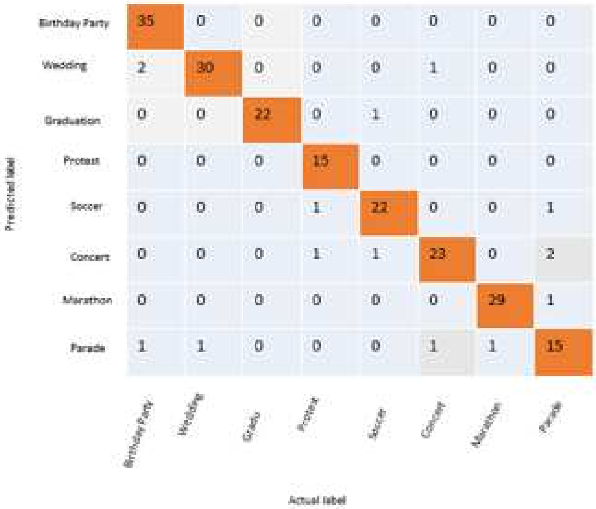


	Birthday Party	Wedding	Gradu	Protest	Soccer	Concert	Marathon	Parade
Birthday Party	42	3	6	0	0	0	1	0
Wedding	2	37	7	0	0	0	0	0
Graduation	5	8	26	6	1	0	0	0
Protest	1	2	1	52	0	2	0	0
Soccer	0	3	2	0	49	1	1	1
Concert	1	1	0	1	0	46	0	2
Marathon	0	1	2	0	0	0	45	1
Parade	0	5	5	0	0	1	4	38

Actual label

Figure 29: SocEID Dataset Confusion Matrix

We have implemented the optimization of weighted feature technique to reduce the confusion between images. We have given more weight to top k unique features to improve the accuracy. Accuracy improved from 86.37% to 90.5%. Figure 30 shows a smaller number of wrongly classified events compared to Figure 29.



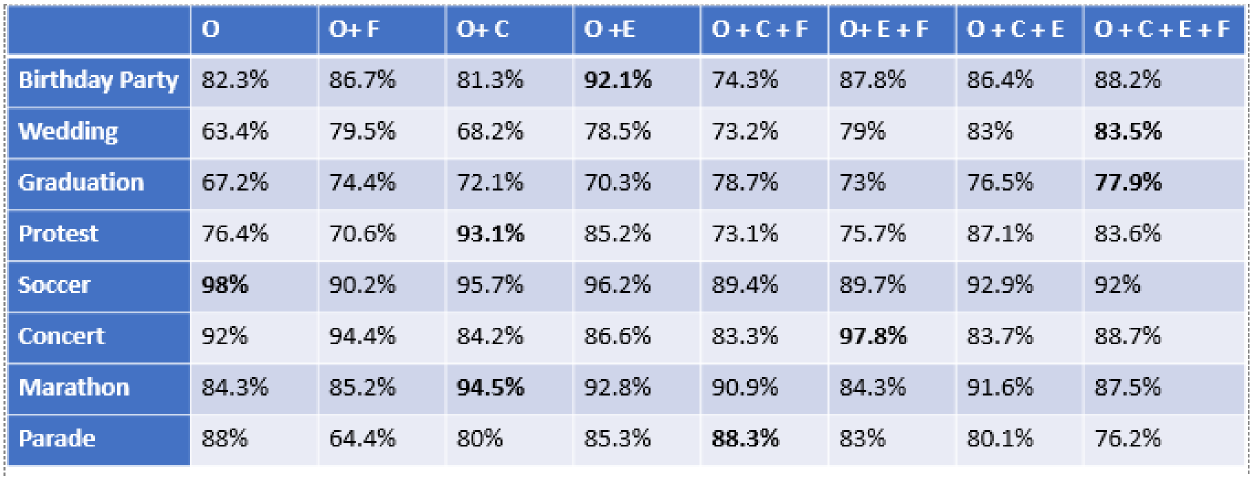
	Birthday Party	Wedding	Gradu	Protest	Soccer	Concert	Marathon	Parade
Birthday Party	35	0	0	0	0	0	0	0
Wedding	2	30	0	0	0	1	0	0
Graduation	0	0	22	0	1	0	0	0
Protest	0	0	0	15	0	0	0	0
Soccer	0	0	0	1	22	0	0	1
Concert	0	0	0	1	1	23	0	2
Marathon	0	0	0	0	0	0	29	1
Parade	1	1	0	0	0	1	1	15

Predicted label
Actual label

Figure 30: SocEID Dataset Confusion Matrix after Optimization

We also evaluated each event accuracy for each combination recognition results. Each event accuracy works well for different combinations. The birthday party has the highest accuracy of 92.1% if it uses only object and emotion features. The protest has the highest accuracy of 93.1% if it uses only object and color features. Like that each event is best for different combination features. Table 25 shows the overall results for each event accuracy with each combination for the SocEID dataset.

Table 25: SocEID Dataset Events Accuracy for Each Combination



	O	O+ F	O+ C	O +E	O + C + F	O+ E + F	O + C + E	O + C + E + F
Birthday Party	82.3%	86.7%	81.3%	**92.1%**	74.3%	87.8%	86.4%	88.2%
Wedding	63.4%	79.5%	68.2%	78.5%	73.2%	79%	83%	**83.5%**
Graduation	67.2%	74.4%	72.1%	70.3%	78.7%	73%	76.5%	**77.9%**
Protest	76.4%	70.6%	**93.1%**	85.2%	73.1%	75.7%	87.1%	83.6%
Soccer	**98%**	90.2%	95.7%	96.2%	89.4%	89.7%	92.9%	92%
Concert	92%	94.4%	84.2%	86.6%	83.3%	**97.8%**	83.7%	88.7%
Marathon	84.3%	85.2%	**94.5%**	92.8%	90.9%	84.3%	91.6%	87.5%
Parade	88%	64.4%	80%	85.3%	**88.3%**	83%	80.1%	76.2%

In this evaluation, we also have shown the comparison between other models with our models. Other models include Alexnet-Fc7, Webly- Fc7, Wider-Fc7 and Event concepts. We compared with other models of one short learning with our model. Their one short learning was implemented with a single positive image for each event. Our model outperformed the mentioned models for each event. Figure 31 shows the overall comparison for each event with other models.

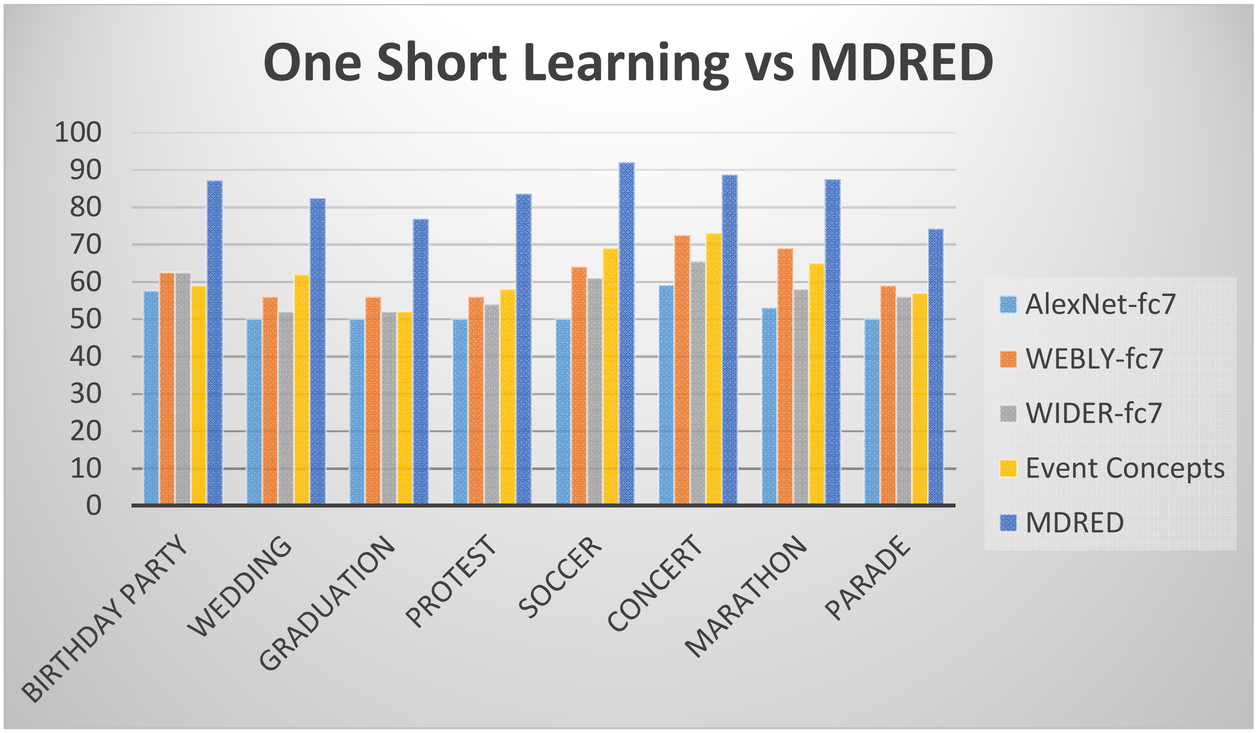


Figure 31: Accuracy Comparison 1 with Other Models for SocEID Dataset

Figure 32 shows the overall accuracy in the comparison with other models. The accuracy for Alexnet-Fc7, Webly-Fc7, Wider-Fc7 and Event concepts is 86.42%, 83.66%, 80.42% and 85.39%, respectively. Our model (MDRED) showed the accuracy of 86.37%, which is higher than the event concepts model and the other two models. After the optimization of weighted features, the MDRED model showed an improved accuracy of 90.5%. This model outperformed the baseline models.

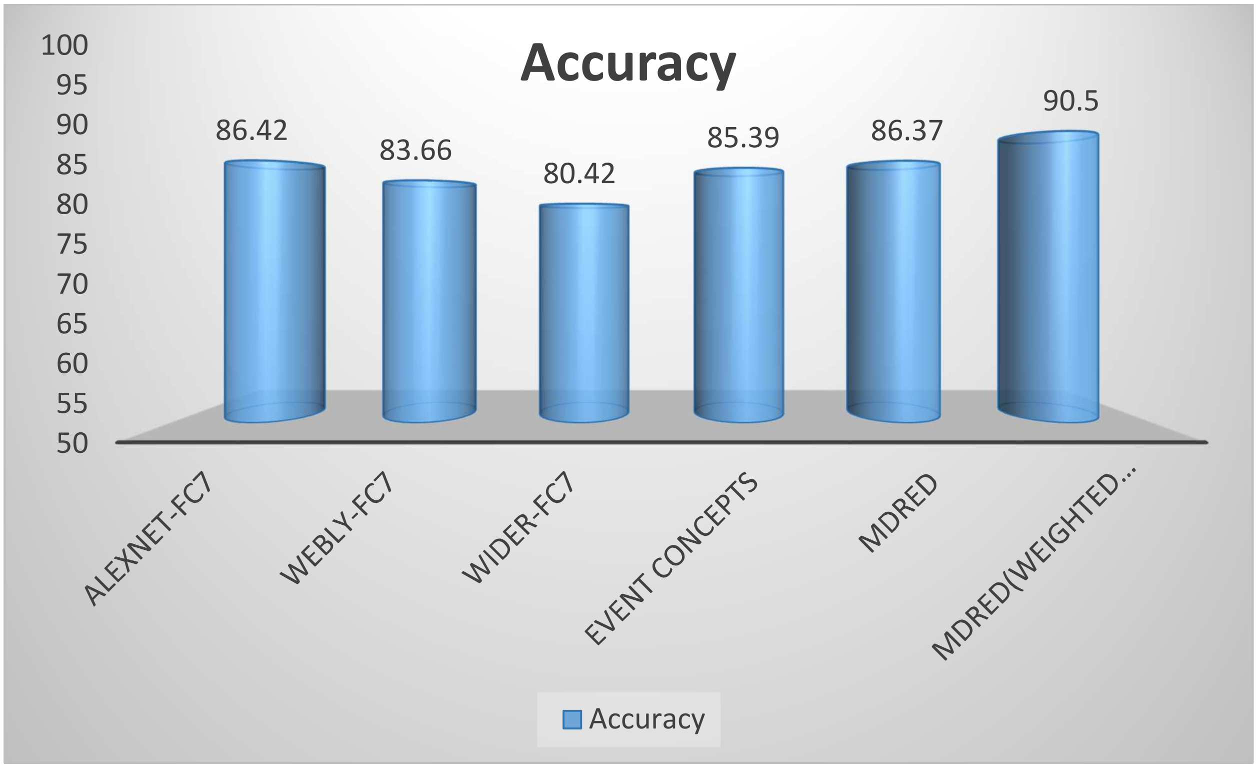


Figure 32: Accuracy Comparison 2 with Other Models for SocEID Dataset

4.9 UIUC Sports Dataset Evaluation

The UIUC Sports dataset was evaluated with a different combination of model recognition results. The accuracy was changing with different combinations. If we use only object recognition results and predict the event with those features, then the accuracy yields to 93.9%. If we use all combinations as the features, then the accuracy was improved from 93.9% to 96.1%, which is the highest accuracy with all TF-IDF features. Table 26 shows the accuracy of different combinations which is also shown in Figure 33.

Table 26: UIUC Model Combinations Accuracy

Model combinations	Accuracy
Object	93.9%
Object + Face	95.6%
Object + Color	93.06%
Object + Emotion	94.6%
Object + Color + Face	94.8%
Object + Emotion + Face	94.01%
Object + Color + Emotion	94.2%
Object + Color + Emotion + Face	**96.1%**

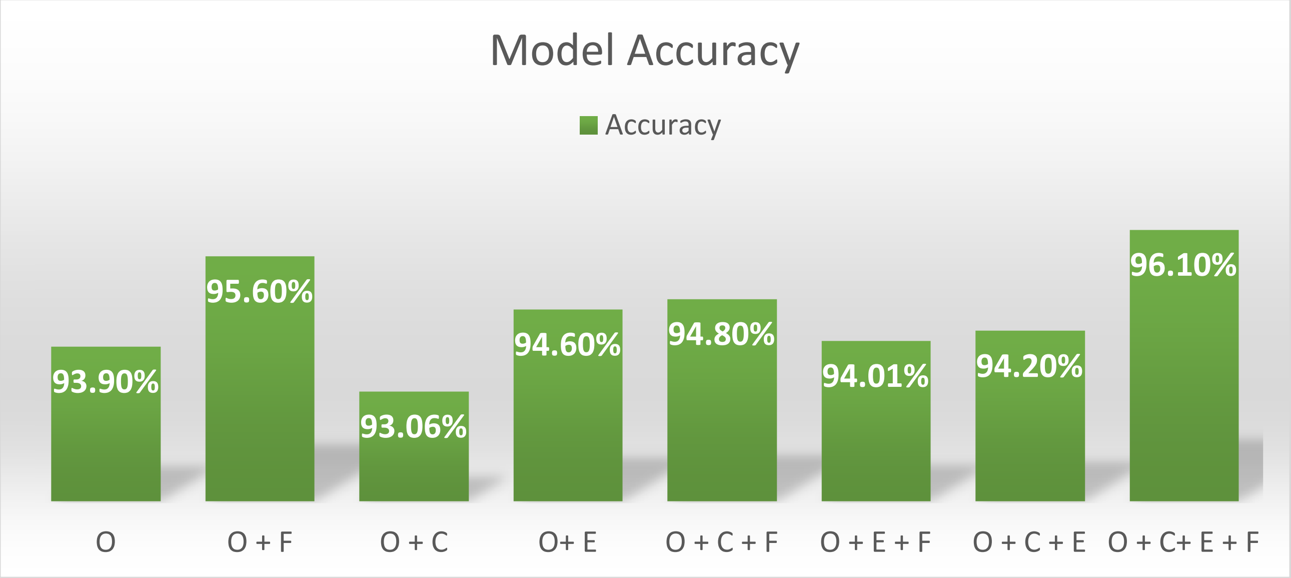


Figure 33: UIUC Sports Model Combinations Accuracy

In the model which is having the high accuracy of 96.1%, all multi-modal recognition was used with the feature vector to predict the event. Out of 8 events, 4 events having 100% accuracy, 2 events have greater than 95% accuracy, Bocce Sports event shows 73% accuracy and Croquet event has 91 %. Figure 34 shows each event accuracy for the UIUC Sports dataset.

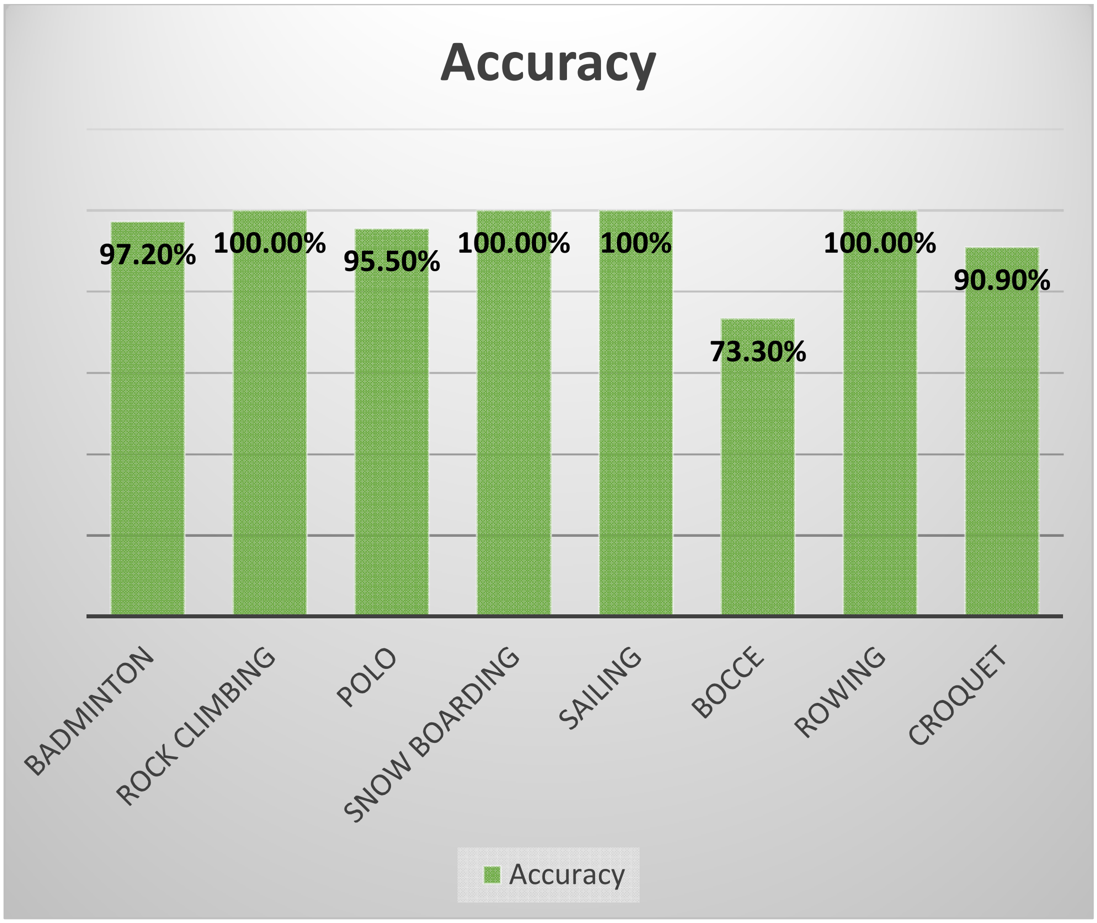


Figure 34: UIUC Dataset Events Accuracy

In the real world, images are not clear and highly confused each other. We found the same issue between Bocce and Croquet Sports event images, which are confused each other. Figure 35 shows the example images which are confused between Bocce and Croquet images because of similar object properties and face properties. The confusion matrix shown in Figure 36 gives a better understanding of the images, which are classified wrongly. The red block shows the number of the images, which are wrongly classified. In this Bocce and Croquet event, the more number of events are wrongly classified.

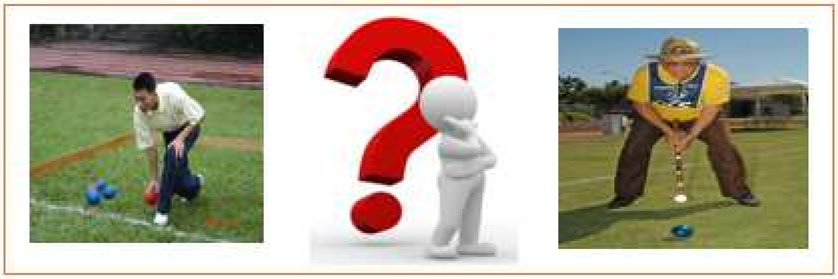
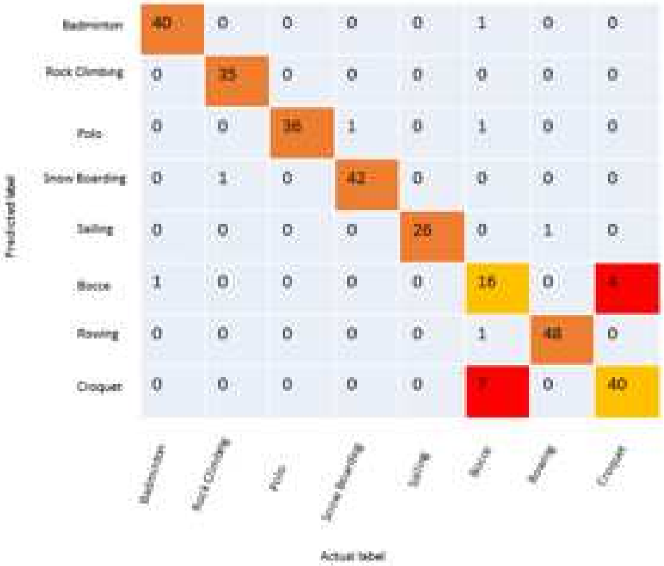


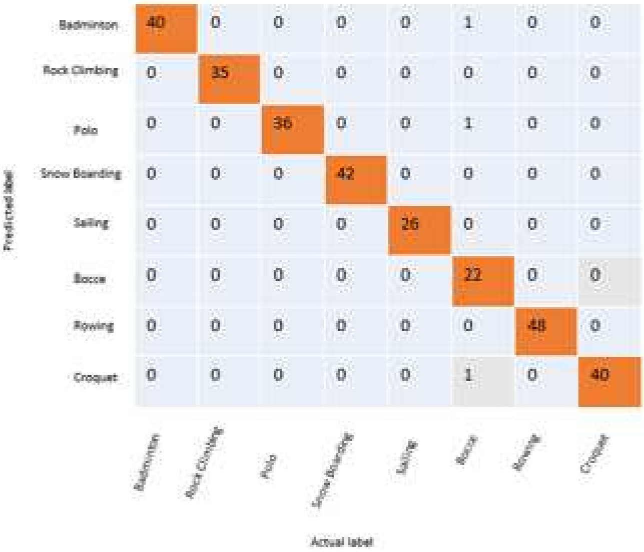
Figure 35: Confusion Images for UIUC Sports Dataset



	Badminton	Rock Climbing	Polo	Snow Boarding	Sailing	Bocce	Rowing	Croquet
Badminton	40	0	0	0	0	1	0	0
Rock Climbing	0	35	0	0	0	0	0	0
Polo	0	0	36	1	0	1	0	0
Snow Boarding	0	1	0	42	0	0	0	0
Sailing	0	0	0	0	26	0	1	0
Bocce	1	0	0	0	0	16	0	6
Rowing	0	0	0	0	0	1	48	0
Croquet	0	0	0	0	0	7	0	40

Figure 36: UIUC Sports Dataset Confusion Matrix

We have implemented the optimization of a weighted feature that aims to reduce the confusion between images. We have given a higher weight to top k unique features to improve the accuracy. The accuracy was improved from 96.1% to 98.8%. Figure 37 shows the confusion matrix after applied the optimization technique. Figure 37 shows the smaller number of wrongly classified events compared to Figure 36.

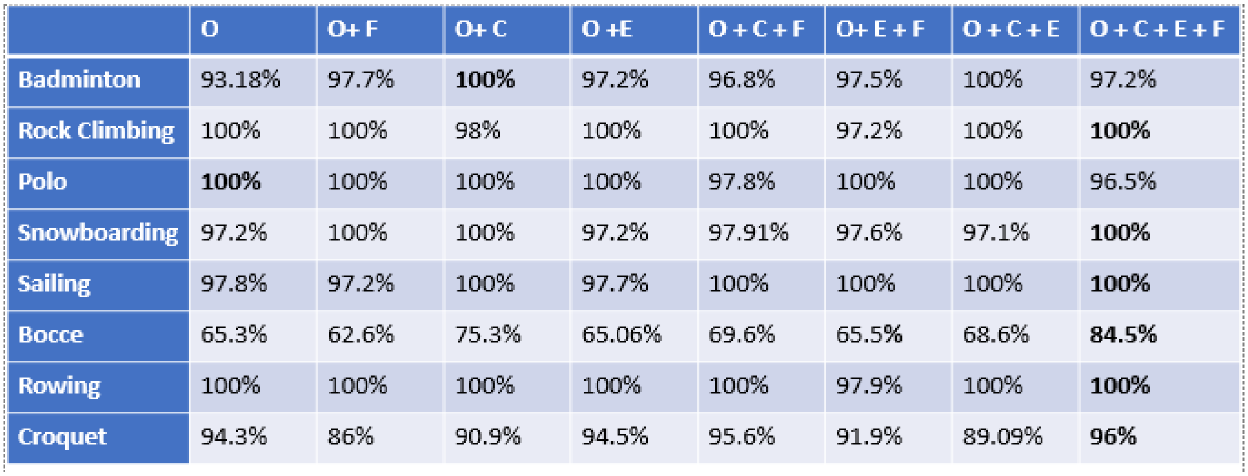


	Badminton	Rock Climbing	Polo	Snow Boarding	Sailing	Bocce	Rowing	Croquet
Badminton	40	0	0	0	0	1	0	0
Rock Climbing	0	35	0	0	0	0	0	0
Polo	0	0	36	0	0	1	0	0
Snow Boarding	0	0	0	42	0	0	0	0
Sailing	0	0	0	0	26	0	0	0
Bocce	0	0	0	0	0	22	0	0
Rowing	0	0	0	0	0	0	48	0
Croquet	0	0	0	0	0	1	0	40

Figure 37: UIUC Sports Dataset Confusion Matrix after Optimization

We also evaluated the accuracy of each event for the combined recognition. Each event accuracy works well for different combinations. Badminton has the highest accuracy of 100% if it uses only object and color features. Polo has the highest accuracy of 100% if it uses only object features. Each event shows the different performance for the different combination of the features. Table 27 shows the overall results for each event with the different combination of the UIUC sports dataset.

Table 27: UIUC Sports Dataset Events Acuuracy for Each Combination



	O	O+ F	O+ C	O +E	O + C + F	O+ E + F	O + C + E	O + C + E + F
Badminton	93.18%	97.7%	**100%**	97.2%	96.8%	97.5%	100%	97.2%
Rock Climbing	100%	100%	98%	100%	100%	97.2%	100%	**100%**
Polo	**100%**	100%	100%	100%	97.8%	100%	100%	96.5%
Snowboarding	97.2%	100%	100%	97.2%	97.91%	97.6%	97.1%	**100%**
Sailing	97.8%	97.2%	100%	97.7%	100%	100%	100%	**100%**
Bocce	65.3%	62.6%	75.3%	65.06%	69.6%	65.5%	68.6%	**84.5%**
Rowing	100%	100%	100%	100%	100%	97.9%	100%	**100%**
Croquet	94.3%	86%	90.9%	94.5%	95.6%	91.9%	89.09%	**96%**

In this evaluation, we also have shown the comparison between other models with our models. Other models include Alexnet-Fc7, Webly- Fc7, Wider-Fc7 and Event concepts. We compared with other models of one short learning with our model. Their one short learning was implemented with a single positive image for each event. Our model outperformed the mentioned models for each event. Figure 38 shows the overall comparison for each event with other models.

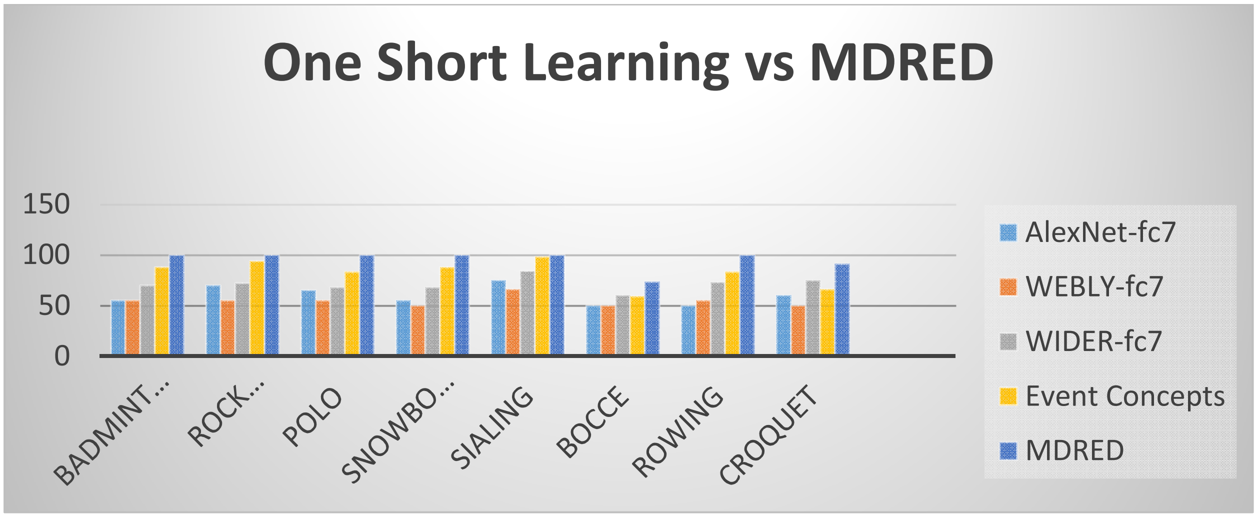


Figure 38: Accuracy Comparison 1 with Other Models for UIUC Sports Dataset

Figure 39 shows the overall accuracy in comparison with other models. Accuracy for Alexnet-Fc7, Webly-Fc7, Wider-Fc7 and Event concepts has 96.47%, 95.16%, 93.85% and 96.68% respectively. Our model (MDRED) shows 96.1% which is lower than the event concepts model and higher than the other two models. After the optimization of the weighted features, the MDRED model has improved its accuracy to 98.8%. This model outperformed all the baseline models.

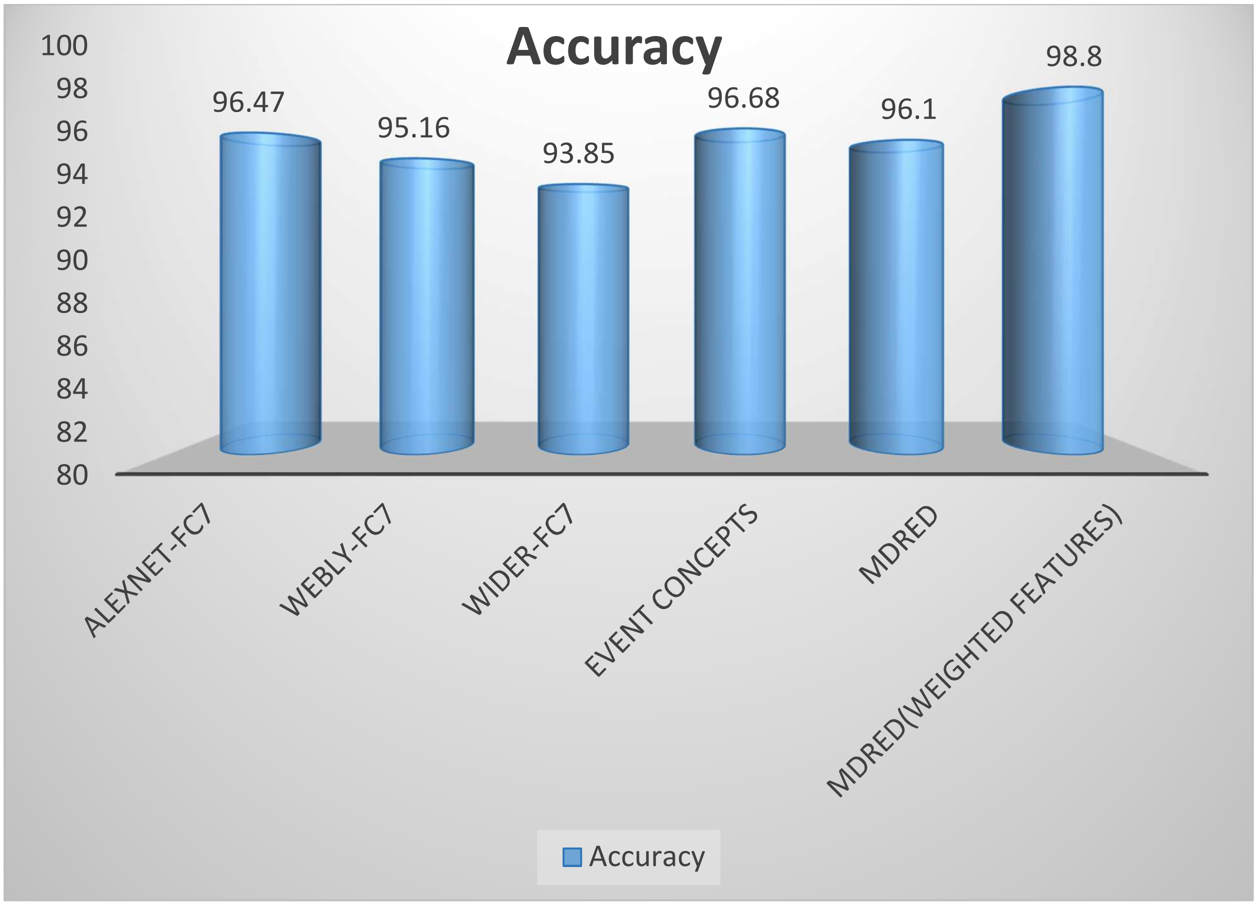


Figure 39: Accuracy Comparison 2 with Other Models for UIUC Sports Dataset

4.10 RED Events Dataset Evaluation

The RED events dataset was evaluated with a different combination of two model recognition results. The accuracy is changed with the combinations of different recognition models. If we use only object recognition results and predict the event with those features, then the accuracy yields to 58.5%. But if we use both object and color features, then accuracy was improved from 58.5% to 65.5%. Since the RED event dataset has natural disasters which don't contain people images, so the face and emotion attributes did not contribute for event classification. Figure 40 shows the accuracy of a different combination of models.

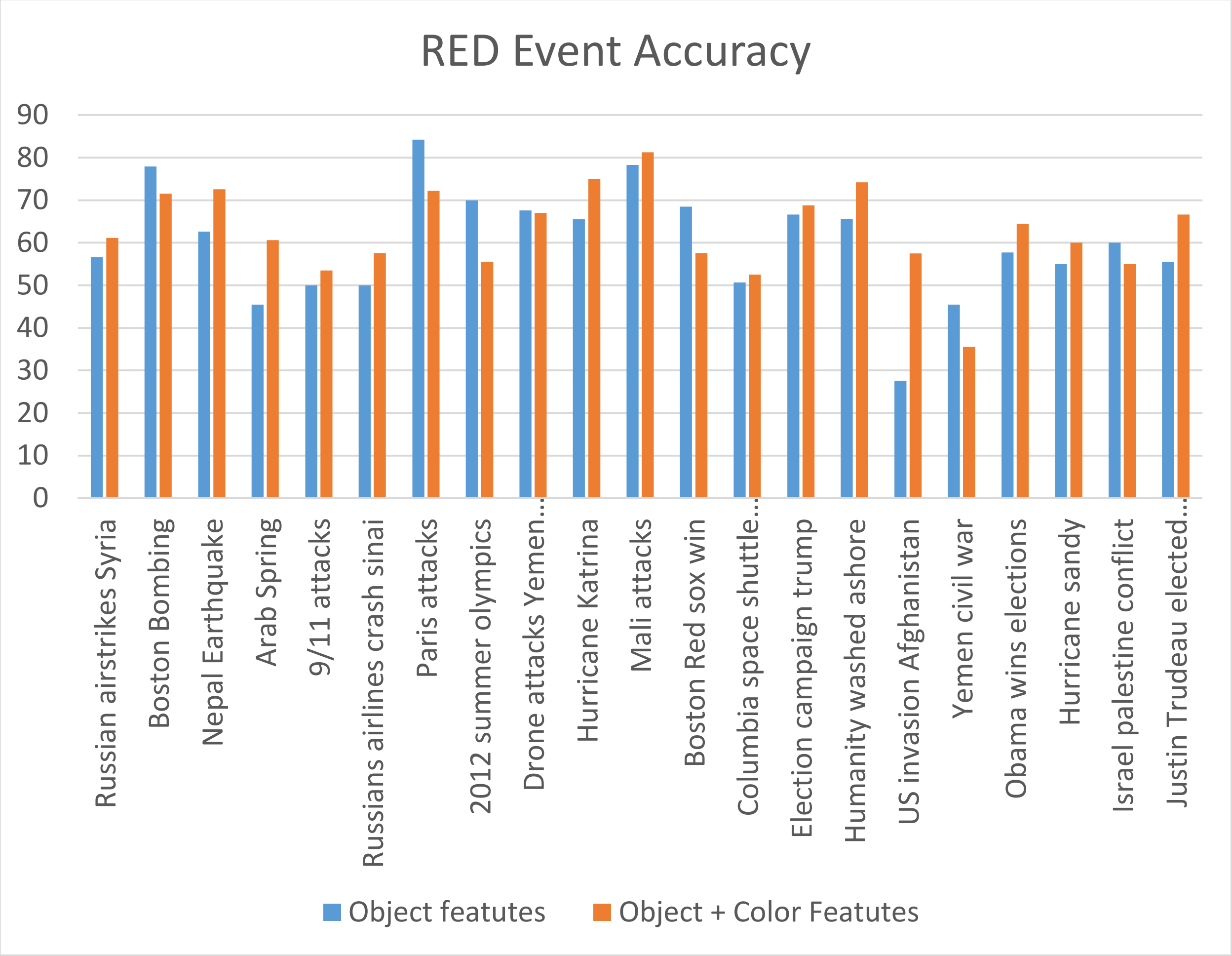


Figure 40: RED Dataset Events Accuracy

In the real world, images are not clear and highly confused each other. We found the same issue between a lot of rare event images, which are confused each other. We have implemented the optimization of the weighted feature technique to reduce the confusion between images. We have given a higher weight to top k unique features to improve the accuracy. The accuracy was improved from 65.5% to 78%.

Figure 41 shows the overall accuracy in comparison with other models. The accuracy for Alexnet-Fc7, Webly-Fc7, Wider-Fc7 and Event concepts was 77.86%, 79.39%, 76.64% and 75.5%, respectively. Our model (MDRED) showed the accuracy of 65.5% which is lower than other models. After the optimization of the weighted features, the accuracy of our model has been improved to 78%. This model outperformed three models of Alexnet-Fc7, Wider-Fc7 and Event Concepts.

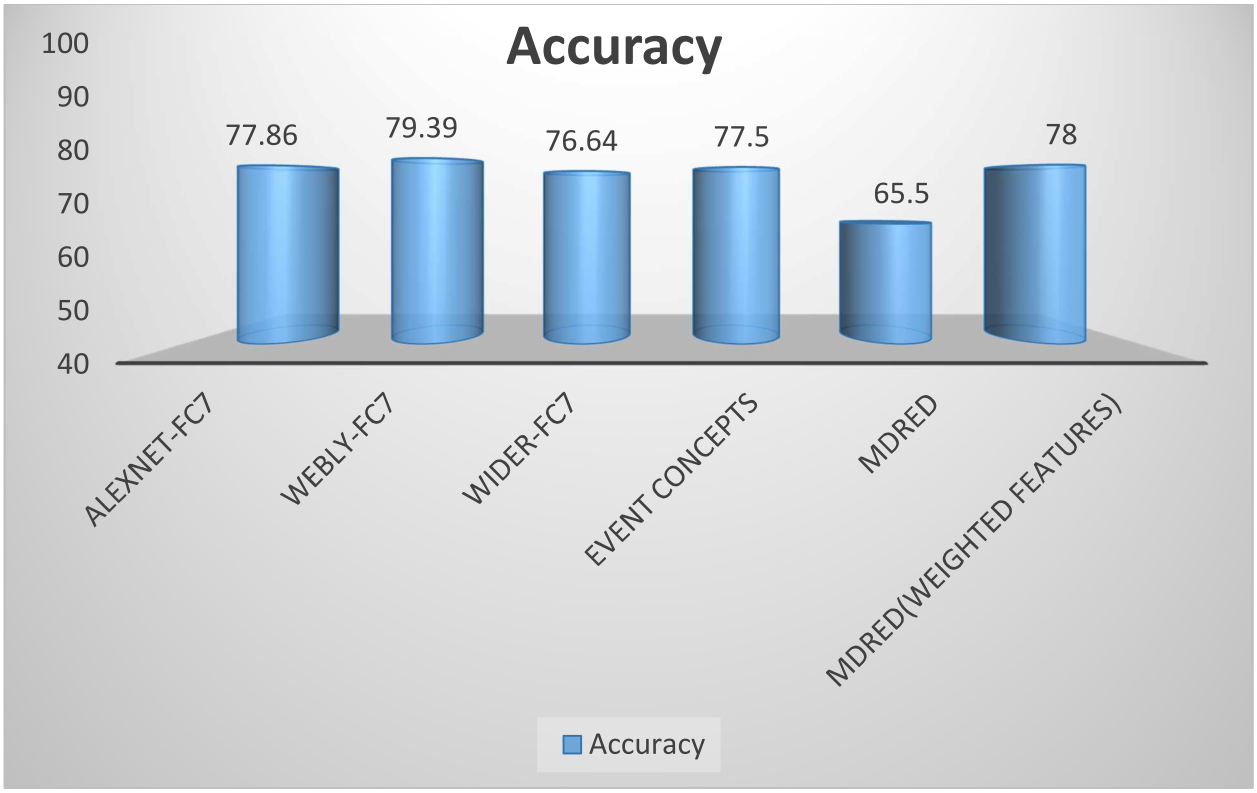


Figure 41: Accuracy Comparison with Other Models for RED Events Dataset

4.11 Model Evaluation using Shallow and Deep Learning

We evaluated the event detection with Deep Learning technique of the CNN and RNN networks. Table 28 shows the network configuration we used in our evaluation. Results showed that Deep Learning performed a less accuracy for all datasets, which we used in our experiments. Since our datasets have small training examples, which are not good for Deep Learning models to train it. Deep Learning expects a large training dataset to have better accuracy. The CNN+RNN model showed 80%, 94%, 62% accuracy for SocEID, UIUC sports, and RED event datasets, respectively. Figure 42 shows that in this context the Shallow Learning model outperforms the Deep Learning model for the event detection.

Table 28: Deep Learning Network Parameters

Property	Values
Embedding Layer Dimension	300
Filter Sizes	[3, 4, 5]
Number of Filters	32
Hidden Units	300
Max Pool Size	4
Dropout Probability	0.5
Batch Size	128
Number of Epochs	100

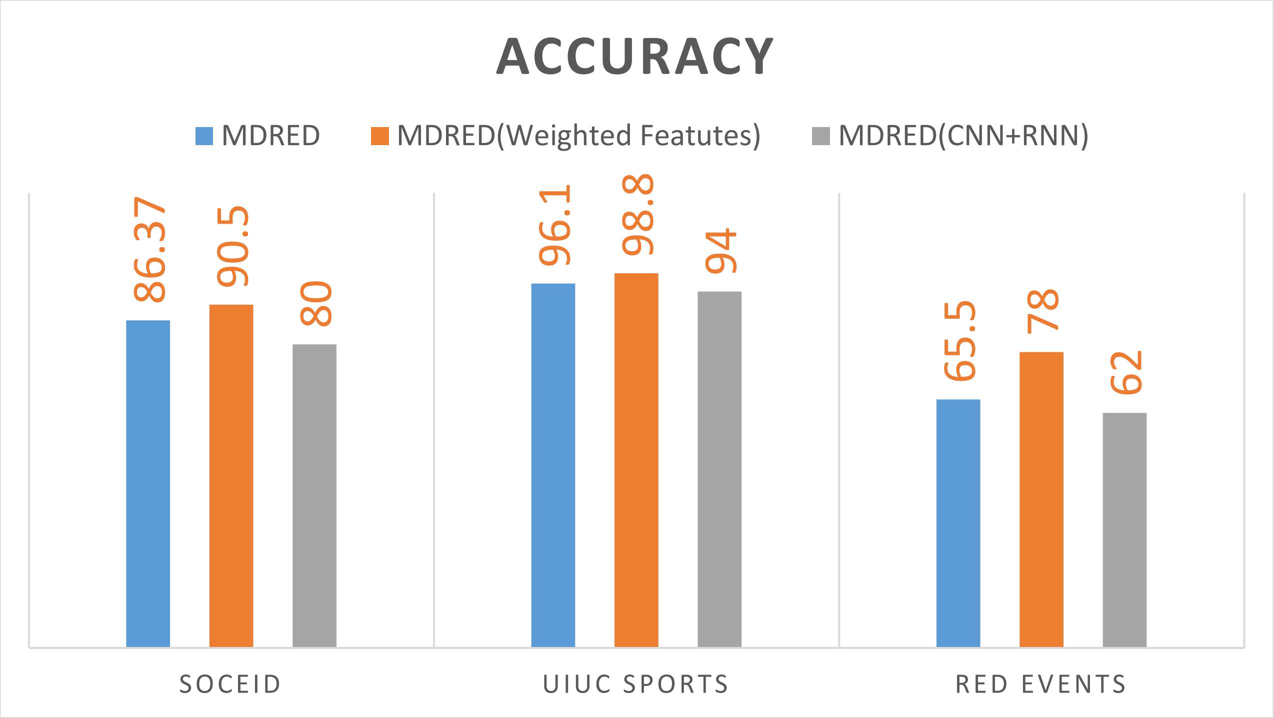


Figure 42: Model Evaluation using Shallow Learning and Deep Learning

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

The Deep Learning becomes a natural choice for visual recognition as it shows an outstanding performance in computer vision tasks. We employed distributed systems with Deep Learning to make event recognitions which can use to classify events. We provided the quantitative results which show the better performance in event predictions. The plug and play model approach is suitable for better scalability. Our quantitative results show that our model outperforms baseline models for diversity event datasets.

5.2 Challenges

Our model results look promising but we can't rely on this system for daily use. This system can be improved if we deal with the following aspects:

- Improving the synchronization for collaborative recognitions: Distributed systems are known for parallel execution on multiple systems and are hard to cop up each other especially if we are expecting a result at the same time
- Improve the execution time: Naturally, cloud-based Deep Learning models have network delay to make a rest call, that needs more time than inferencing

5.3 Future Work

There is some future work that is needed to improve the performance of the proposed model.

- Employing a more powerful event detection model which can work with specific events to get a better accuracy
- Improving the event detection accuracy with a light Deep Learning network
- Extending to use an efficient model that can be deployed in the device to get a better performance instead of the cloud-based Deep Learning model.

BIBLIOGRAPHY

[1] Z. Ma, X. Chang, Z. Xu, N. Sebe and A. G. Hauptmann, "Joint Attributes and Event Analysis for Multimedia Event Detection," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 7, pp. 2921-2930, July 2018.

[2] Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin et al. "Tensorflow: a system for large-scale machine learning." In *Operating Systems Design and Implementation*, vol. 16, pp. 265-283. 2016.

[3] G.Alain, A. Almahairi, C. Angermueller, N. Ballas, Y. Bengio, "Theano: A Python framework for fast computation of mathematical expressions," in *arXiv preprint arXiv:1605.02688,* 2016.

[4] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, Orlando, Florida, USA, 2014.

[5] D. Ballard and C. Brown, *Computer vision*. Englewood Cliffs, New Jersey:Prentice-Hall, 1982, pp. 1-6.

[6] Ujjwalkarn," An Intuitive Explanation of Convolutional Neural Networks", the data science blog, 2016. [Online]. Available: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/. [Accessed: Aug.2018]

[7] Anon, "Term frequency and Inverse Document Frequency",2018. [Online]. Available: http://www.tfidf.com/. [Accessed: Aug. 2018]

[8] Anon. 2018. Artificial intelligence. (August 2018). Retrieved August 4, 2018 from https://en.wikipedia.org/wiki/Artificial_intelligence

[9] Anon. 2018. Supervised learning. (August 2018). Retrieved August 4, 2018 from https://en.wikipedia.org/wiki/Supervised_learning

[10] Anon. 2018. Unsupervised learning. (August 2018). Retrieved August 4, 2018 from https://en.wikipedia.org/wiki/Unsupervised_learning

[11] Anon. What is Apache Spark. Retrieved August 4, 2018 from https://hortonworks.com/apache/spark/

[12] Anon. What is Apache Hadoop. Retrieved August 4, 2018 from https://hortonworks.com/apache/hadoop/

[13] Anon. What is Android. Retrieved August 4, 2018 from https://www.android.com/

[14] Pouyanfar, S., & Chen, S. C. (2017). “Automatic video event detection for imbalance data using enhanced ensemble deep learning”. *International Journal of Semantic Computing*, 11(01), 85-109.

[15] Ahsan, Unaiza, Chen Sun, James Hays, and Irfan Essa. "Complex Event Recognition from Images with Few Training Examples." *In 2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 669-678. IEEE, 2017.

[16] Alqhtani, Samar M., Suhuai Luo, and Brian Regan. "Fusing Text and Image for Event Detection in Twitter." *The International Journal of Multimedia & Its Applications 7*, no. 1 (2015): 27.

[17] Alqhtani, Samar M., Suhuai Luo, and Brian Regan. "Fusing Text and Image for Event Detection in Twitter." *The International Journal of Multimedia & Its Applications 7*, no. 1 (2015): 27.

[18] Xiong, Yuanjun, Kai Zhu, Dahua Lin, and Xiaoou Tang. "Recognize complex events from static images by fusing deep channels." *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1600-1609. 2015.

[19] Ahsan, Unaiza, Chen Sun, James Hays, and Irfan Essa. "Complex Event Recognition from Images with Few Training Examples." *In 2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 669-678. IEEE, 2017.

[20] Gan, Chuang, Naiyan Wang, Yi Yang, Dit-Yan Yeung, and Alex G. Hauptmann. "Devnet: A deep event network for multimedia event detection and evidence recounting." *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2568-2577. 2015.

[21] Xu, Dan, Yan Yan, Elisa Ricci, and Nicu Sebe. "Detecting anomalous events in videos by learning deep representations of appearance and motion." *Computer Vision and Image Understanding* 156 (2017): 117-127.

[22] Chang, Xiaojun, Yi Yang, Eric Xing, and Yaoliang Yu. "Complex event detection using semantic saliency and nearly-isotonic SVM." *In International Conference on Machine Learning*, pp. 1348-1357. 2015.

[23] Anon. 2018. Naive Bayes classifier. (August 2018). Retrieved August 4, 2018 from https://en.wikipedia.org/wiki/ Naive_Bayes_classifier.

[24] Anon. 2018. Random forest. (August 2018). Retrieved August 4, 2018 from https://en.wikipedia.org/wiki/Random_forest.

[25] Anon. SocEID Dataset. Retrieved August 4, 2018 from http://unaizahsan.com/?page_id=64.

[26] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng. Nus-wide: A real-world web image database from national university of singapore. *In Proc. of ACM Conf. on Image and Video Retrieval (CIVR'09), Santorini, Greece*., July 8 -10, 2009.

[27] T. Reuter, S. Papadopoulos, G. Petkos, V. Mezaris, Y. Kompatsiaris, P. Cimiano, C. de Vries, and S. Geva. Social event detection at mediaeval 2013: Challenges, datasets, and evaluation. *In Proceedings of the MediaEval 2013 Multimedia Benchmark Workshop Barcelona, Spain*, October 18-19, 2013, 2013.

[28] Li, Li-Jia, and Li Fei-Fei. "What, where and who? Classifying events by scene and object recognition." In Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, pp. 1-8. IEEE, 2007.

[29] Anon. RED Events Dataset. Retrieved August 4, 2018 from http://unaizahsan.com/?page_id=64.

[30] Anon. Machine Learning Process Image. Retrieved August 4, 2018 from https://searchenterpriseai.techtarget.com/definition/machine-learning-ML

[31] Anon. Supervised and Unsupervised Learning Image. Retrieved August 4, 2018 from https://www.mathworks.com/discovery/machine-learning.html

[32] Anon. Deep learning network Image. Retrieved August 4, 2018 from https://medium.freecodecamp.org/want-to-know-how-deep-learning-works-heres-a-quick-guide-for-everyone-1aedeca88076

[33] Anon. Distributed Computing Image. Retrieved August 4, 2018 from https://i.ytimg.com/vi/YS-QvfCZWvc/maxresdefault.jpg

[34] Anon. Apache Spark Image. Retrieved August 4, 2018 from https://ogirardot.files.wordpress.com/2015/05/future-of-spark.png

[35] Anon. Android architecture Image. Retrieved August 4, 2018 from https://www.tutorialspoint.com/android/images/architecture.jpg

[36] Wu, Shuang, Sravanthi Bondugula, Florian Luisier, Xiaodan Zhuang, and Pradeep Natarajan. "Zero-shot event detection using multi-modal fusion of weakly supervised concepts." *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2665-2672. 2014.

[37] Myers, Gregory K., Cees GM Snoek, Ramakant Nevatia, Ramesh Nallapati, Julien van Hout, Stephanie Pancoast, Chen Sun et al. "Evaluating multimedia features and fusion for example-based event detection." *In Fusion in Computer Vision*, pp. 109-133. Springer, Cham, 2014.

[38] Vijayakumar, V. "Event detection in cricket video based on visual and acoustic features." *Journal of Global Research in Computer Science 3*, no. 8 (2012): 26-29.

[39] George A. Miller (1995). wordNet: A Lexical Database for English. *Communications of the ACM* Vol. 38, No. 11: 39-41.

VITA

Nageswara Rao Nandigam completed his Bachelor’s degree in computer science from Vasavi college of engineering in Hyderabad, affiliated to Osmania University in Hyderabad, India and then he worked for Teradata from May 2014 to December 2016 in India. In January 2017, he joined in University of Missouri-Kansas City (UMKC) to pursue his Master degree in Computer Science, with an emphasis on Data Science. While he was studying at UMKC, he worked as a Data Analytics Intern at CenturyLink from May 2018 to July 2018. After completion of his Master’s Program, he plans to work as a Software Engineer in Cerner.