RAMAIAH
Institute of Technology

Department of Computer Science and Engineering

Mini-Project Report

on

Lie Detection Using CNN Video Analytics

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering

in

Computer Science & Engineering

By

Sl. No.	Name	USN
1.	**Akarsh N**	**1MS20CS009**
2.	**Bhargava M**	**1MS20CS030**
3.	**D Vignesh Reddy**	**1MS20CS040**
4.	**B Venkat Rahul**	**1MS20CS145**

Under the guidance of

Name of the internal guide

Designation

M S RAMAIAH INSTITUTE OF TECHNOLOGY

(Autonomous Institute, Affiliated to VTU)

BANGALORE-560054

www.msrit.edu

2021- 2022

Abstract

We have chosen the topic of 'Lie Detection using CNN Video Analytics' as this is an interesting as well as a fun topic to work with. The implementation is not yet done any where and wish to do the same. Detecting whether a person is lying just by analyzing the facial expression is what we want to achieve.

To begin with, we created our own dataset as 'Lie Detection' is an unexplored topic and free open-source datasets are not available. These video clips are 5-8 sec long and contain people who are lying and telling the truth. Then we used the frame extraction method to extract frames. These frames are used to train the CNN based 'Lie Detection' algorithm. The algorithm uses Rectified Linear Unit (ReLU) as the activation function and Softmax as the optimizer. Our model was trained with 6712 images containing both 'True' as well as 'False' frames using 5 epochs, that is the number of iterations we are passing the dataset into our model.

At first, we got an accuracy of 1.0, but this was caused due to overfitting, i.e., the algorithm was working fine on the trained data, but not the unseen data. So, we investigated methods to reduce overfitting. We used cross-validation with 5 KFolds (number of partitions on the dataset) and finally achieved a model with an agreeable accuracy of 0.90.

We integrated this model into a mobile application that dynamically captures videos and predicts the categories with probability percentage.

Table of Contents

Sl. No.	Topics	Page No
1.	Introduction	**4**
2.	Literature Review	**5**
3.	Design and Implementation	**7**
4.	Evaluation Results	**11**
5.	Conclusion	**13**
6.	References	**14**

Chapter 1 – Introduction

Getting to know whether a person is lying or not in any situation is important. Hundreds of evidences get manipulated when people lie and solving the concerned problem becomes difficult. For example, setbacks occur when witness lies which can cause wrong judgment. To eliminate such an issue, even though we have a polygraph machine, it must be wired to people and they must be medically tested before doing so.

Hence our proposed system helps to get to know whether a person is lying or not just by capturing the facial expression of the person. Using our mobile application, we can predict whether a person is lying or not. You can focus the mobile camera to the face of the person. The model dynamically captures the video and predicts the categories with probability percentage.

The topic of 'Lie Detection' is challenging and fun to work with. Even though there are research articles about facial expression detection, implementation is not done yet. So, we felt that we can develop a mobile application that captures videos and the background algorithm will be trained to extract frames and predict the result. After discussing and taking inputs from teachers, we finally decided to use Convolution Neural Network (CNN) for the image classification. It would use ReLU as the activation function and Softmax as the optimizer function.

Since our selected topic is not implemented and not thoroughly explored collection, formation and creation of the dataset was challenging. We had to collect video clips of people lying as well as people who are telling the truth. We carefully analyzed the facial expressions and categorized them into two folders: True and False. Then we passed this dataset into the frame extractor to extract the frames from videos.

For training the model, we used these extracted images as the input. The CNN algorithm works by dividing the RGB image into small squares by using the ReLU and Convolution functions. Later flattening is done and finally Softmax is used as the optimizer.

When an input image is given, a facial expression in this case, is tested by the model. Results with probability percentage is predicted.

Chapter 2 - Literature Review

1. In the research article, the FERC team (2020) used computer algorithms to detect human emotions using facial expressions. They have used FERC model, which is CNN based. They have trained their model using 10000 images of 154 persons. An accuracy of 96% is achieved using length of 24 in expressional vector (EV) calculations.

2. In the research paper published in 2019 14th IEEE International Conference, they have considered a method using recurrent neural network. Using this method, micro facial expressions are captured. To encode the temporal changes in facial regions, they have made use of Histogram of Oriented Optical Flow features.

3. In this journal the research team has mentioned how facial expressions are recognized using shallow convolution neural network. They have classified the facial expressions into two categories namely static and micro expressions. With the help of shallow CNN which was applied on 5 open datasets like FER2013 etc., they achieved a model with accuracy 0.853 with 37 epochs.

4. In the following Conference paper published in 2020 by Xiaofang Wang, they have mentioned about the limitations of convolution operations. Here they are considering a new method called Spatiotemporal Attention Cell Search. This method improved video classification accuracy by 2%.

5. In this research article published in 2019, the authors have spoken about the application of CNN in image sensing. They have focused on CNN-based deep learning, multiple scales, and non-stationary methods to extract useful information from images.

6. In the corresponding journal published in 2020 by G.G.Yen, they proposed an algorithm that had taken some population and image datasets. They tried to applied different architecture of CNN. They tried to compare the different classification accuracy with respect to number of parameters. Finally based on the accuracy they discovered best CNN architecture.

7. In this research paper published in 2019 by Ammar Ismael Kadhim, they conducted a survey on text classification where they had taken a data and used different classification techniques to measure the performance among the different text classifiers. The merging of different information into classification process can improve the results.

8. In this conference paper published in 2019 4th International Conference, Convolution Denoising Autoencoder-convolution neural network is used to predict crowd density in image frames. Experimentally, they have proved that CDAE-CNN outperforms CNN across various data sets. The CDAE reconstructs noisy images and thus improving the accuracy.

9. In this research paper they test video classifier with large video set. Logistic Regression is used for high level videos extraction from the large dataset. They combined the Logistic Regression and Dense Layer together to their MoE model and extracted the output

10. In the following conference paper published in 2022 by Kwanee Choi. They developed an on-device model to transfer the knowledge from teacher model to student model. Their proposed method gives an exact idea of the respective architectures they have chosen. Based on noise level they can easily differentiate the teacher model from attention based and non-attention-based student models.

Chapter 3 – Design and Implementation

The proposed system uses CNN on the video frames to categorize them as “TRUE” or “FALSE.” As shown in Fig.1, the proposed system takes video as the input. Frames are extracted from the video. Then the trained CNN model classifies these frames into 2 categories as mentioned above. The model training is done using our own data-set. CNN model uses ReLU function and Softmax function. Finally, the maximum number of frames matching with a particular category is considered as the output.

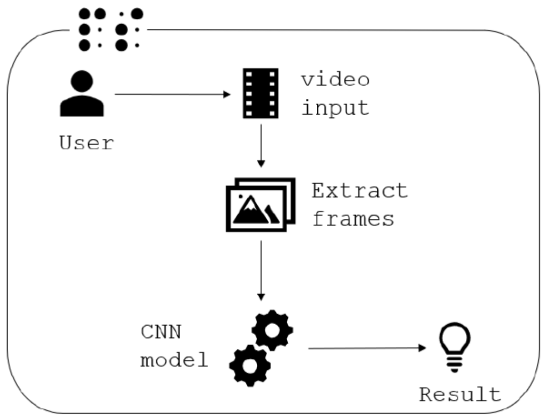
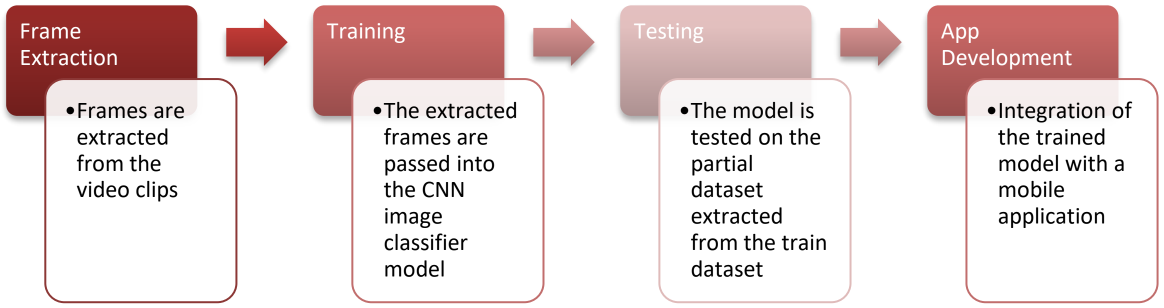


Fig. 1. Lie Detection Software Workflow

Pseudo Code:

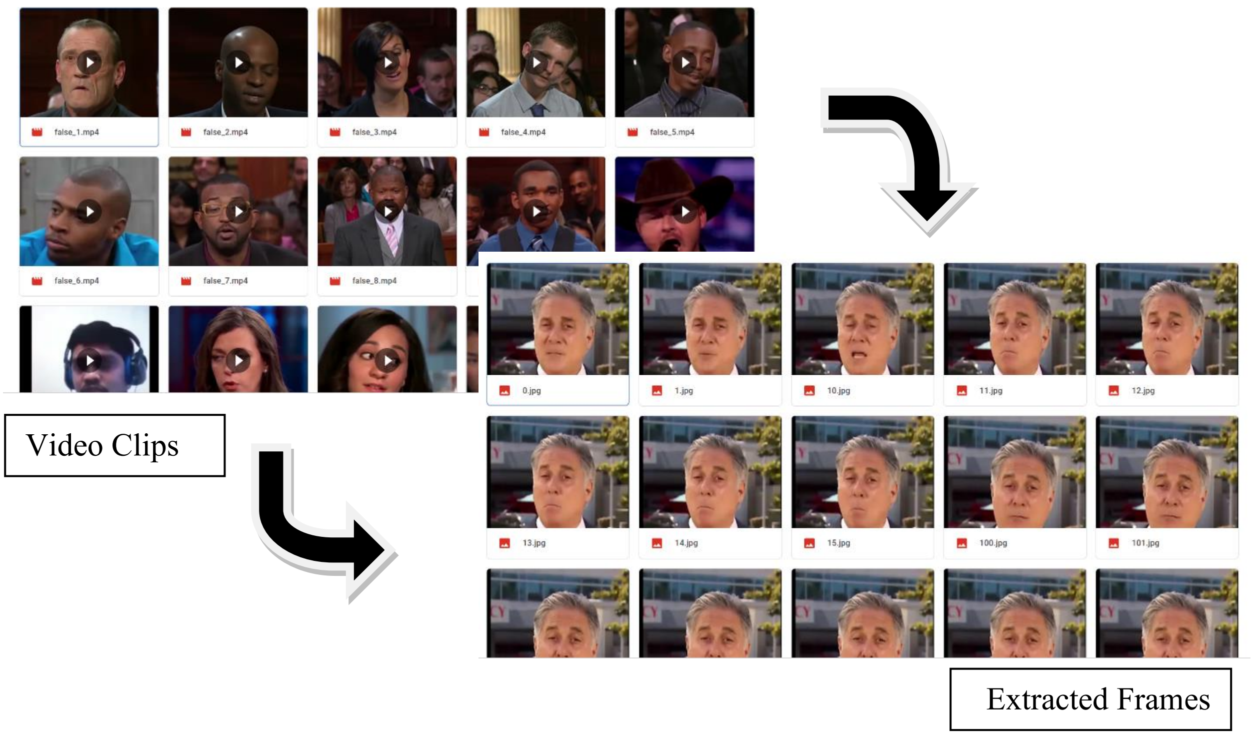
```
Input: video=Input()
Output: Predicted category with probability percentage
do{
    T_Frames=0;//Initially number of true and false frames will be zero
    F_Frames=0;
    Frames[] = ExtractFrames(video); //Frames or still images are
    extracted from the given video input
    for(int i=0; i<sizeof(Frames); i++)
        if(True==CNNModel(Frames[i])
            T_Frames+=1;
        else
            F_Frames+=1;
    Output(T_Frames, F_Frames, Frames);//to calculate the probability percentage of
    the respective categories
}while(True);
```

Workflow:


Frame Extraction
•Frames are extracted from the video clips
Training
•The extracted frames are passed into the CNN image classifier model
Testing
•The model is tested on the partial dataset extracted from the train dataset
App Development
•Integration of the trained model with a mobile application

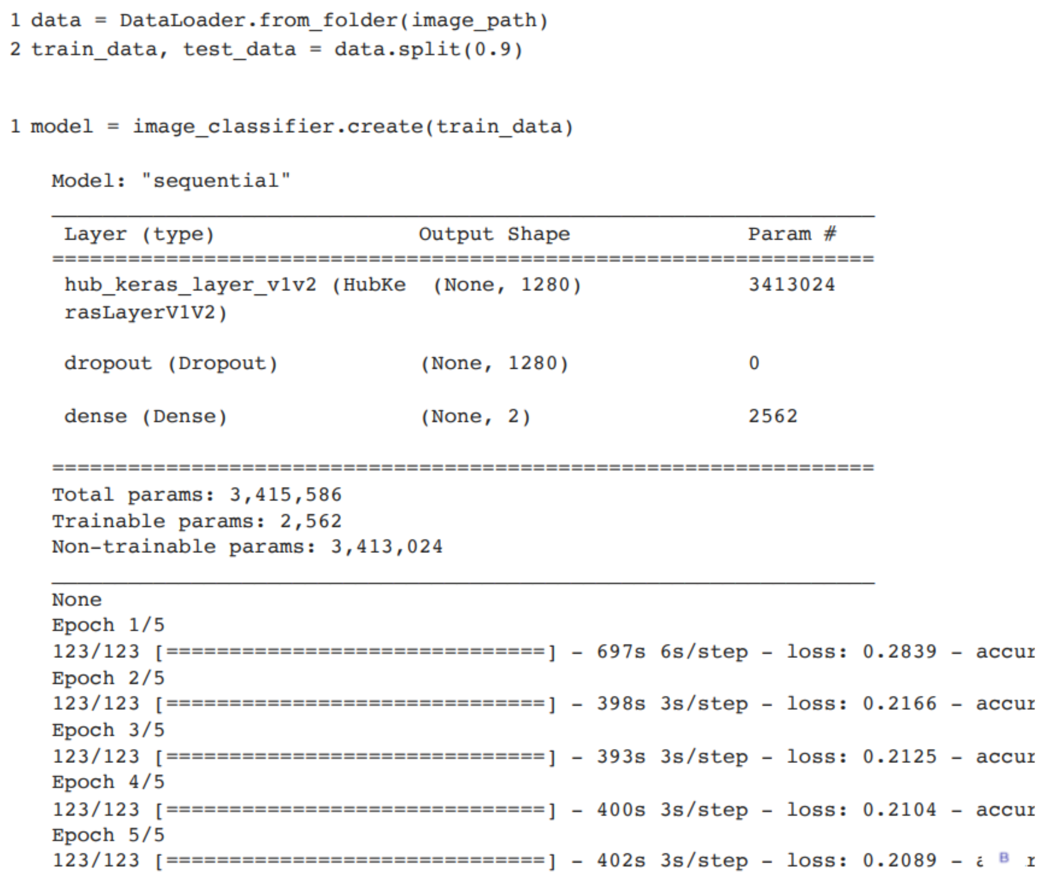
1. Frame Extraction:

```
count = 0
os.makedirs("Dataset/train/true")
for i in range(0,len(videonames_list)):
  video_data = videonames_list[i]
  video = cv2.VideoCapture(video_data)
  success = True
  while success:
    success,image = video.read()
    name = './Dataset/train/true/'+str(count)+'.jpg'
    if success == True:
      cv2.imwrite(name,image)
      print('Frame {} Extracted Successfully'.format(count))
      count+=1
    else:
      i = i+1
    i = i+1
  print('\n\n\nVideo {} Extracted Successfully\n\n\n'.format(video_data))
```



2. Training:

   We have trained the model using the above-mentioned dataset with 5 epochs.



```
data = DataLoader.from_folder(image_path)
train_data, test_data = data.split(0.9)


model = image_classifier.create(train_data)

  Model: "sequential"
  _________________________________________________________________
   Layer (type)                Output Shape              Param #
  =================================================================
   hub_keras_layer_v1v2 (HubKe  (None, 1280)             3413024
   rasLayerV1V2)

   dropout (Dropout)           (None, 1280)              0

   dense (Dense)               (None, 2)                 2562

  =================================================================
  Total params: 3,415,586
  Trainable params: 2,562
  Non-trainable params: 3,413,024
  _________________________________________________________________
  None
  Epoch 1/5
  123/123 [==============================] - 697s 6s/step - loss: 0.2839 - accur
  Epoch 2/5
  123/123 [==============================] - 398s 3s/step - loss: 0.2166 - accur
  Epoch 3/5
  123/123 [==============================] - 393s 3s/step - loss: 0.2125 - accur
  Epoch 4/5
  123/123 [==============================] - 400s 3s/step - loss: 0.2104 - accur
  Epoch 5/5
  123/123 [==============================] - 402s 3s/step - loss: 0.2089 -
```

3. Testing:

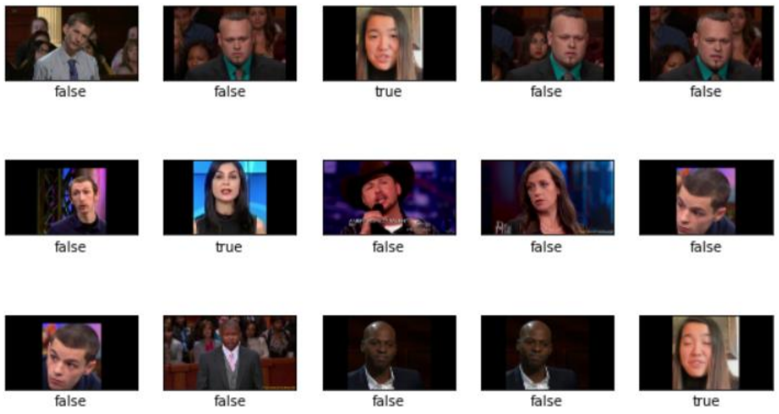
Initially our CNN based model was showing an accuracy of 1.0. This was due to the overfitting. Overfitting is caused when the model predicts accurately for the trained dataset but not on unseen data.

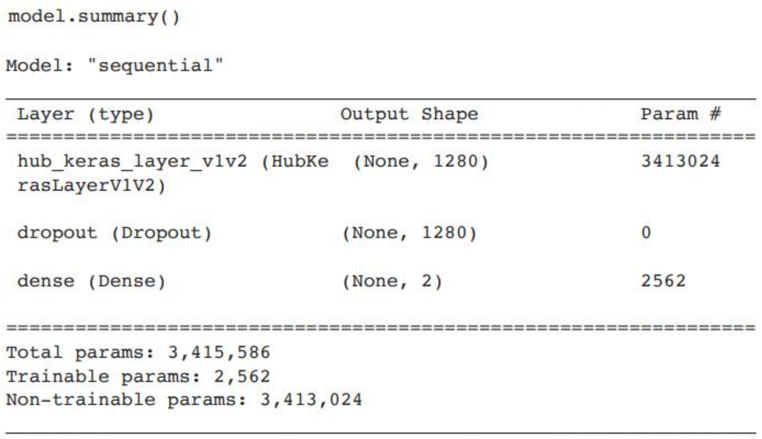
We used the method of cross-validation to reduce overfitting. We divided the data into 5 subcategories and re-trained the model. Then we were able to achieve an accuracy of 0.90.

kfold = KFold(3, True, 1) # prepare cross validation

for train, test in kfold.split(data): # enumerate splits

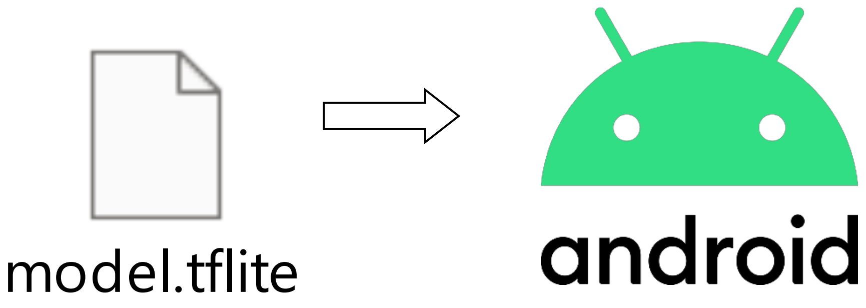
print('train: %s, test: %s' % (data[train], data[test]))





```
model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 hub_keras_layer_v1v2 (HubKe  (None, 1280)             3413024
 rasLayerV1V2)

 dropout (Dropout)           (None, 1280)              0

 dense (Dense)               (None, 2)                 2562

=================================================================
Total params: 3,415,586
Trainable params: 2,562
Non-trainable params: 3,413,024
_________________________________________________________________
```

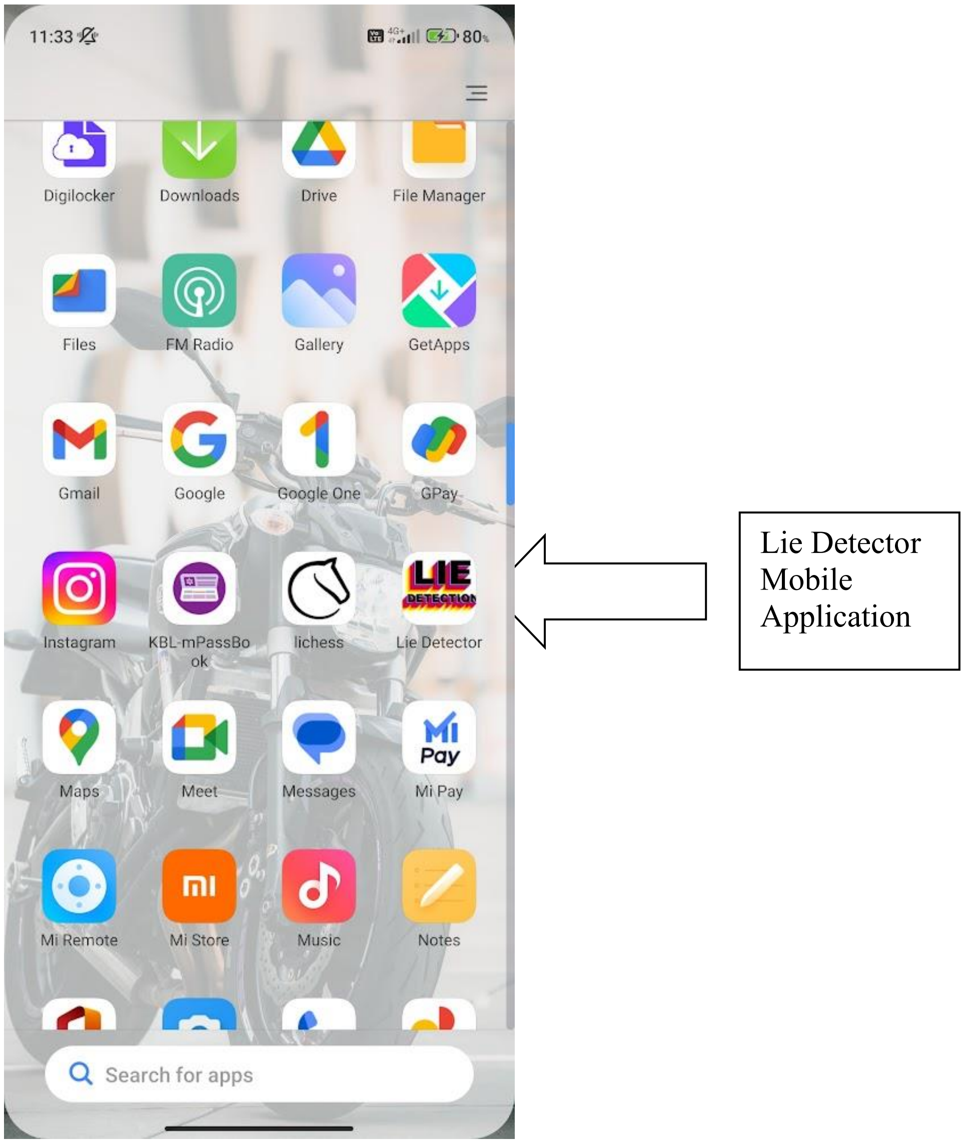
4. App Development:

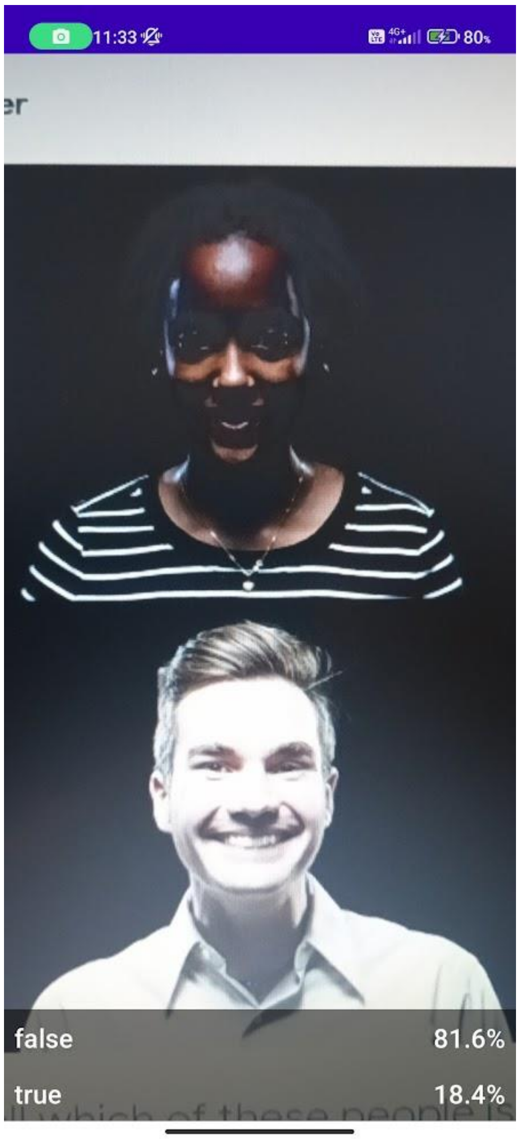


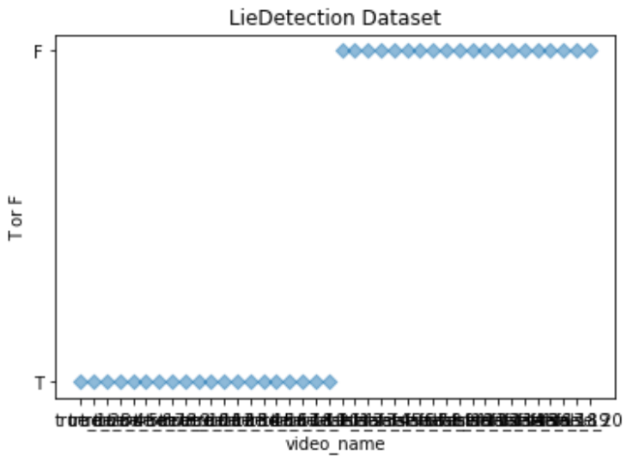
With the help of Android Studio, we integrated the above trained and tested TensorFlow lite model with a mobile application. The app works dynamically by capturing the videos and predicting the results.

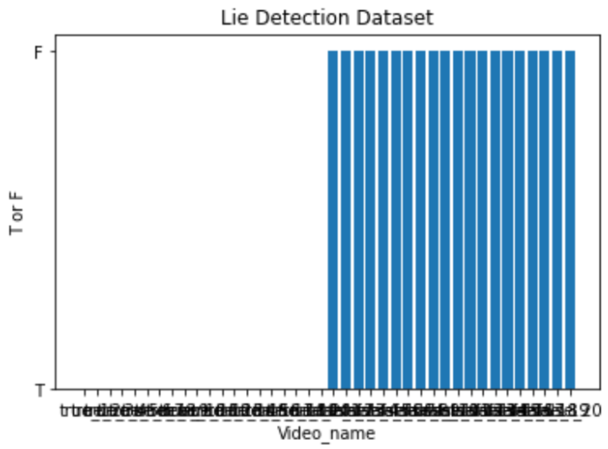
Chapter 4 – Evaluation Results

We have developed a mobile application which uses the above trained model to predict whether the person is lying of not. The user must open the application and focus the camera onto the face of the person to detect the facial expressions. The model dynamically takes the video input and predicts the category with probability percentage.









Predicted Results

The User Interface of the application

```
# plotting the data
plt.scatter(x, y,
            marker='D', alpha=0.5)

# Adding title to the plot
plt.title("LieDetection Dataset")

# Adding label on the y-axis
plt.ylabel('T or F')

# Adding label on the x-axis
plt.xlabel('video_name')

plt.show()
```

Chapter 5 – Conclusion and Future work

Our project is a valuable instrument which can be implemented to detect truthfulness and deceitful behavior in several fields such as:

• The crime investigation departments and national security agencies

• Business and industry meetings

• Court sessions while interrogating victims and people

Limitations of the proposed system:

• The proposed model may not be as accurate as the traditional polygraph method used for lie detection

• The computing time of the algorithm might be an issue in real-time applications

• The algorithm fails to detect lies from people who do not show any facial features while lying

In future, we wish to reduce the above-mentioned limitations, increase the accuracy of the model and implement the proposed project in various fields.

References

1) The Science of Lie Detection by Verbal Cues: What Are the Prospects for Its Practical Applicability? Tim Brennen and Svein Magnussen

2) Assessment criteria indicative of deception (ACID): an integrated system of investigative interviewing and detecting deception. Colwell, K., Hiscock-Anisman, C. K., Memon, A., Taylor, L., and Prewett, J.

3) The analysis of nonverbal communication: The dangers of pseudoscience in security and justice contexts. DePaulo, B. M., Lindsay, J. J., Malone, B. E., Muhlenbruck, L., Charlton,

4) Deception detection: state of the art and future prospects. Masip, J.,Psicothema

5) 'Language of lies': urgent issues and prospects in verbal lie detection research. Nahari, G., Ashkenazi, T., Fisher, R. P., Granhag, P.-A.,

6) Validity of content-based techniques for credibility assessment—how telling is an extended meta-analysis taking research bias into account? Oberlader, V. A., Quinten, L., Banse, R., Volbert, R., Schmidt, A. F.,

7) A meta-analytic review of the timing for disclosing evidence when interviewing suspects. Oleszkiewicz, S., and Watson, S. J. M.S. RAMAIAH INSTITUTE OF TECHNOLOGY (Autonomous Institute, Affiliated to VTU)

8) Discriminability in deception detection is not d: Reporting the overlap coefficient for practitioner-accessible results. Pérez-Rosas, V., Abouelenien, M., Mihalcea, R., and Xiao, Y., Linton

9) A personal model of trumpery: linguistic deception detection in a real-world high-stakes setting. Van Der Zee, S., Poppe, R., Havrileck, A., and Baillon

10) A cognitive approach to lie detection: A meta-analysis. Vrij, A., Fisher, R. P., and Blank, H Levine