Report on Mini Project :-

DeepFake and Steganography in an Image Using Deep Learning

By

Rahul Tandel (2019130063)
Omkar Padir (2019130046)
Sumit Thakare (2020301079)

under the guidance of

Prof. Pramod Bide

Department of Computer Engineering

Bharatiya Vidya Bhavan's
Sardar Patel Institute of
Technology

ACKNOWLEDGEMENT

We are glad to publish "Mini Project" as our report and would like to take this opportunity to express our gratitude to all those who assisted us in building our project. We appreciate our college faculties and mentors for their consistent efforts in making us learn and use new concepts along with the implementation.

Very grateful to Dr. Pramod Bhide, our project guide, for his insightful and timely assistance throughout the project. We appreciate his encouragement, patience, and belief in our team.

We are thankful to all of the people who believed in us and kept pushing us to the edge for completing our project.

Rahul Tandel
Omkar Padir
Sumit Thakare

TABLE OF CONTENT

Problem Definition

- The most important part of capturing photos is the work that happens AFTER they’re shot. This desktop application we created consists of image editing tools and effects useful to both make up for effects you couldn’t create during the shoot and enhance the beautiful feature you did capture.

- Steganography: Normally we send messages through online messaging services. But what if someone is watching your messages in and out and you don't want to get caught. You need a mask to hide the message so that it actually looks like a normal exchange. Here comes the steganography, used to hide messages in normal looking things. In this application we provide an image steganography tool.

- Deep-fake detection: imagine an individual watching a speech of his favourite political figure but, this content is deep-fake. This can mislead someone's ideologies. Moreover, deep-fake cyber-attacks can be easily anonymised, making victims of dirty disinformation wars unsolved deep in history. To solve this issue to some extent one can have his own deep-fake detector to check content.

Market Survey

Considering open source free image editing software we found GIMP, however the best one remains Adobe photoshop but it is paid.

Then in steganography we found "steghide" which is primarily a command shell based package. And many more steganography tools both online and offline.

In deep-fake detection, deepware is far better and currently considered one of the best model available on android and web.

Observations:

- No application offering all 3 together.
- Deepware is one of the best currently available easy to use applications for deep-fake detection, but still it fails to detect deep-fakes created using the latest deep-fake AI models.
- We need more open source image editing softwares to compete with paid softwares.

Literature Survey

We looked over the internet to find any research paper that supports or is similar to our idea. Following are the research papers we read to understand the topic for our project.

- "Steganography in Images Using LSB Technique"
- "MesoNet: a Compact Facial Video Forgery Detection Network",
- "Exploiting Visual Artifacts to Expose Deepfakes and Face Manipulations"
- "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks"

Scope of Work & Objectives

We have started this project to make it one of the best open source image editors. As the paid softwares is most of the time out of bound for many art enthusiasts. Currently

- Constraints:
    1) The model does not verify any given inputs. The model accepts wrong or hypothetical data too and gives its prediction.
    2) The model does not consider interior issues for price prediction. 3) The suggestions for the car are selected such that they guarantee profit. 4) Car brands and models are limited based on the dataset.
    5) All input fields are mandatory for prediction

- Boundaries:
    1) This model covers those users who are concerned with buying or selling used cars.
    2) The model is trained only once.

- Assumptions:
    1) The car is to be sold by the first-owner of the car.
    2) The seller type is Individual. (Seller is not considered as

Dealer) Objectives-

- The Objective of this project is to help the users who don't want to spend the money for their image editing activity and use most of the advanced image editing tools.
- We have also provided the option for the users who wants to perform the steganography
- Also nowadays deep-fake detection is a very famous thing that users want to do for no cost.

Block Diagram :

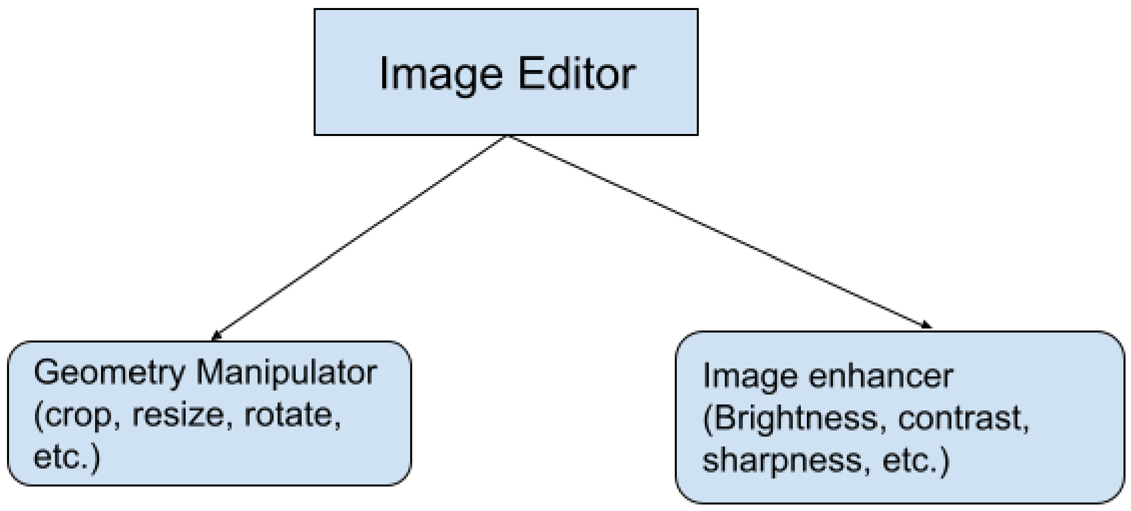
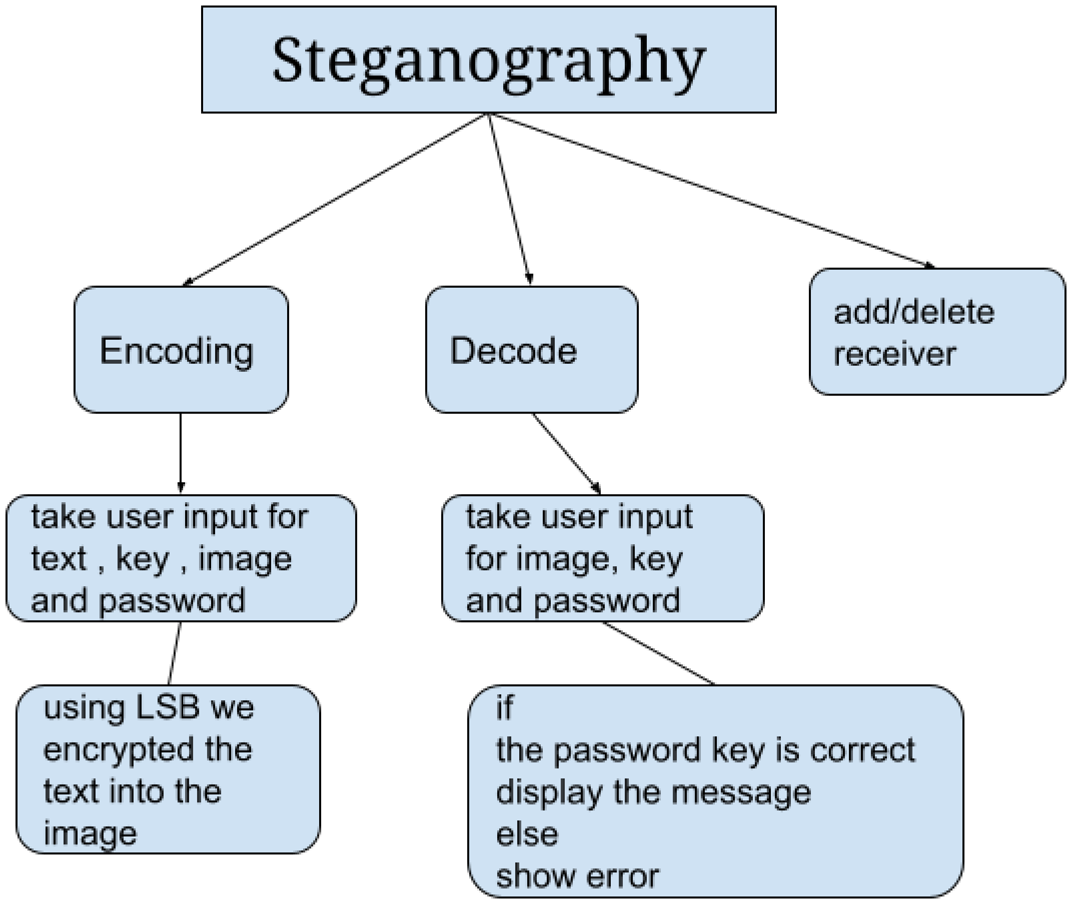
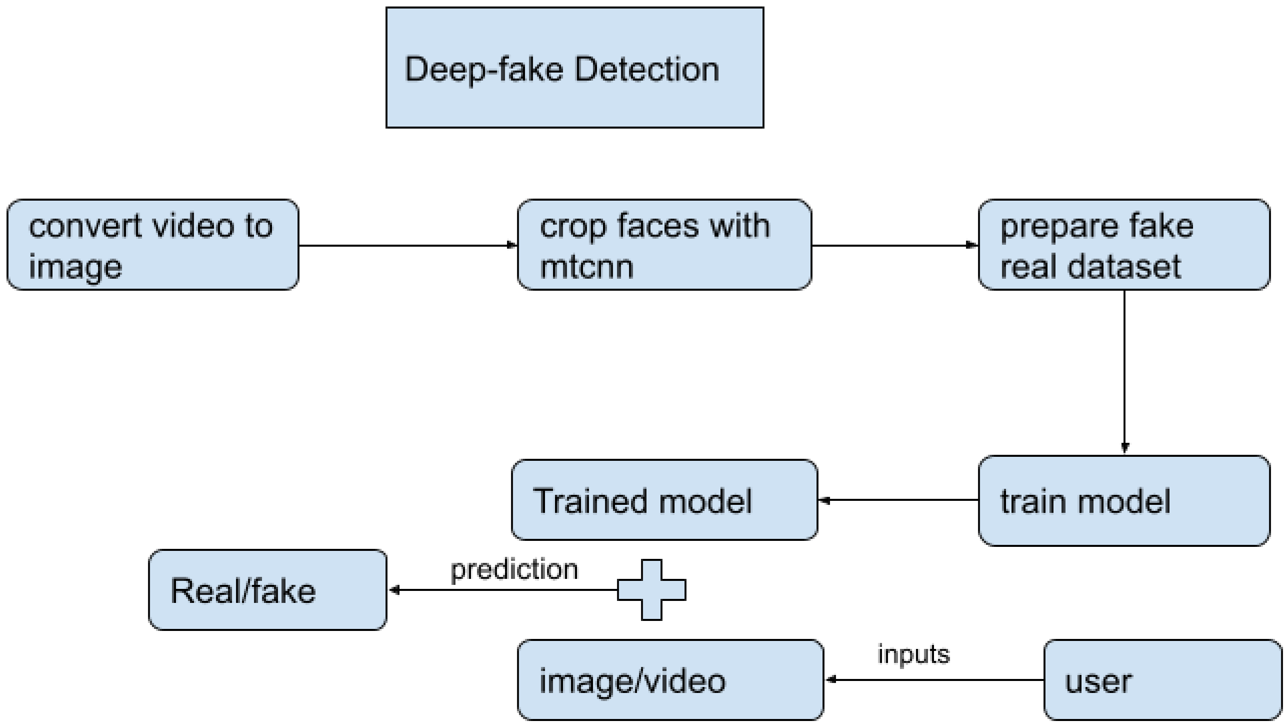
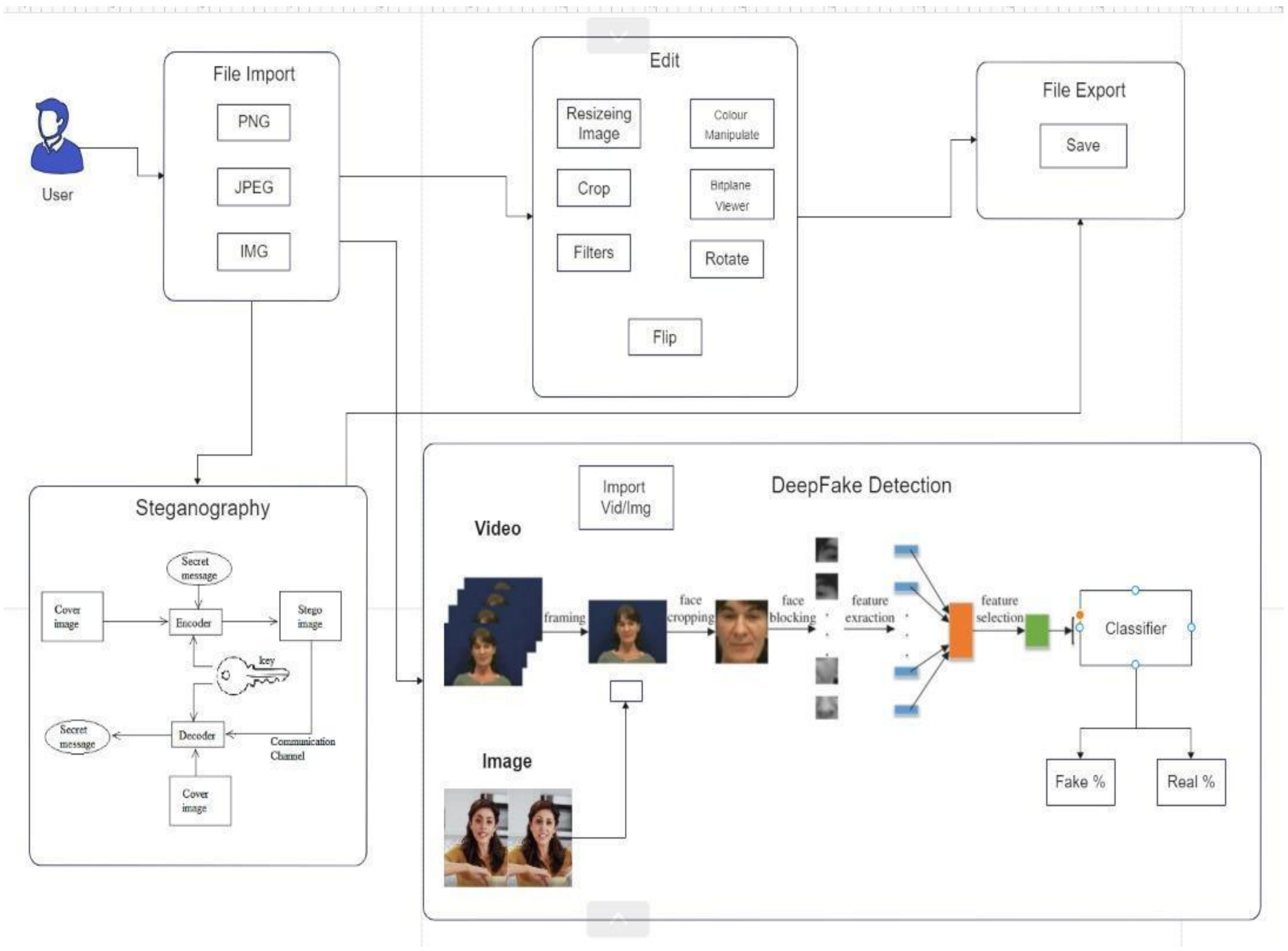


Image Editor
Geometry Manipulator (crop, resize, rotate, etc.)
Image enhancer (Brightness, contrast, sharpness, etc.)


Steganography
Encoding
Decode
add/delete receiver
take user input for text , key , image and password
take user input for image, key and password
using LSB we encrypted the text into the image
if
the password key is correct
display the message
else
show error


Deep-fake Detection
convert video to image
crop faces with mtcnn
prepare fake real dataset
train model
Trained model
prediction
Real/fake
inputs
image/video
user

Architecture Diagram:



Project Plan and Timeline

Week-1	**Started listing out what we want to include in our project. Created ppt for phase 1.**
Week-2	**We wanted to make this project as a desktop application as we have not done it before, so that new things would be learned. Started reading documentation and watching YouTube videos for tkinter, pillow and OpenCV. Our first thought was to complete the image editor part. We aimed for a basic image editor as we didn’t have much time to make it grand.**
Week-3	**Normal desktop application with feature for basic image filters, with open and display image, was done.**
Week-4	**As we gained confidence in the image editing part. We want to implement steganography. We read about the lsb method for steganography. (research paper). Then on the weekend we wrote code to implement the steganography. It was basic steganography, just as script (not implemented as GUI). Also started reading about deepfake and various methods implemented to detect it.**
Week-5	**The Tkinter gui part was having a lot of bugs while implementing. So solving those issues took a lot of time. In this week only we actually started practically implementing the deep-fake detection in google colab. This week we were also searching for a deepfake dataset and tried to implement the code we wrote to train the model. Most of the dataset available were too big in size. Finally in the last days we found a perfect dataset containing real and deep-fake cropped face images.**

Week-6	**We also completed steganography GUI implementation. We trained the model with a small dataset (Containing the 25 images). It was working fine compared to very few images provided. 10 Apr we had the 2nd phase.**
Week-7	**70% implementation was done but, GUI was still filled with bugs. Started working on GUI. image editor features were added.**
Week-8	**GUI. Major flow in steganography code was detected. Solved those issues.** **Trained model with dataset containing 11000+ cropped face images. Model accuracy was 90 % on validation dataset containing 5000+ cropped face images.**
ESE	**Last few changes in GUI were done.**

Implementation details:

Tech stack used:

- **Google colab (data science notebook)**
- **Tensorflow**
- **Keras**
- **Tkinter (python frame work)**
- **Python(3.8) (programming language)**
- **Os**
- **Json**
- **Pillow (PIL: python imaging library) (for image processing)**
- **Opencv (for image processing)**
- **Numpy**
- **Cryptography**

- **Pandas**
- **Math**
- **Pathlib**
- **Mtcnn (face detection)**

Image editor implementation:

tkinter was used as main GUI framework. So all the GUI was created from scratch.

Crop, resize, rotate, image enhancement(Brightness, contrast, colour, sharpness) and then extra fetures like tree-view for images were handled with tkinter and python pillow library.

Inside the tools option we have colour manipulator and bitplane viewer.

Color manipulator: In color manipulator user can select the threshold of RGB value to which he wants to enhance. This is implemented using opencv. After taking the threshold of color values to enhance in backend we split the perticuler channel and then convert it into binary image. White part in the binary image is the part which would be actually affected and other part of image would remain unaffected.

Bitplane viewer: this is included primarily as a supplement to steganography. Using this user can view each 3 color channel last bit plane.

Steganography implementation:

With this user can encode the image he received, decode the image with message or add / remove receivers.

Encoding image:

For this, we have implemented least significant bit method with changes in the ways it is saved. Following is the algorithm.

- **User provide us the password (satisfying the requirements), key, message to encode and image to contain the message.**
- **User can generate key for different receivers. In the option menu he is only able to see the name he given to particular receiver. Each receiver in user's list have unique key which is stored in json file. When he selects the name of receiver that key is selected In backend.**
- **Now using key we encrypt the message from plane text to cryptographic text. So that we will put the cryptographic text in image and not plane image. If user selects "no key" then we will put plane text in image**
- **After this we need a password to fetch from which row of image we will start filling our data.**

```
If (position(letter in message)%(length(password)) = = position(letter in password))
        then that letter is placed in active row of that password letter.
```

- **We are keeping the array of active row for that password letter and array of all current pixel to fill in each row. If that active row of password letter is filled then we will search for another empty row as we have stored the unfilled row record.**
- **After all this image contains the message and user can save the image.**

Decoding image:

With this user can decode the message from stego message. this can only decode the messages encoded with this application. So after receiving image, you need the key (if used),

password and same piece of application.

- **Algorithm of decoding goes just reverse of encoding message. like at the time of decoding we need image containing message, key (if used) and password, and same application.**

Add/delete receivers:

With this user can add the receivers. Each name of the receiver needs to be different. After providing name of receiver and pressing add receiver new receiver with different key is created.

Implementation Details

Tech Stack Used-

- Jupyter (Data Science Notebook)
- Django (Web Framework)
- Python (Programming language)
- HTML
- CSS
- Javascript
- MySQL (Database)

·

Conclusion

1) Digital image processing consists of the manipulation of images using digital computers. Its use has been increasing exponentially in the last decades. By doing this project we have learned about many new things like tkinter, opencv, steganography, and also about using ML/AI in image processing.

2) We learnt how to integrate our trained model with our GUI for a friendly user interface

Future scope

- We will also try to add more features for editing the Digital image and to convert the singal image to different types of frames.

- Also in this project we will make some advance changes so that user can work on digital image more easily

Reference research papers/links

● https://www.researchgate.net/publication/319306871_Predicting_the_Price_of_Used_Cars_using_Machine_Learning_Techniques seen on 10th March 2021. ● https://www.temjournal.com/content/81/TEMJournalFebruary2019_113_118.pdf, 2019, seen on 9th March 2021.

● https://arxiv.org/pdf/1711.06970, 2018, seen on 9th March 2021.

● 17c Formula- https://dvcheck.com/how-much-diminished-value-do-you-have/

● https://towardsdatascience.com/how-to-reuse-your-python-models-without-ret raining-them-39cd685659a5

● https://pbpython.com/categorical-encoding.html

● https://levelup.gitconnected.com/random-forest-regression-209c0f354c84 ●

https://towardsdatascience.com/3-underrated-strategies-to-deal-with-missing-v alues-a539fb6c0690