IMAGE STEGANOGRAPHY

USING MACHINE LEARNING

PAPER NAME: PROJECT 2
PAPER CODE: CS 892

• ABHINABA PODDAR • ABHISEK DAS • ABHISHEK CHAKRABORTY •
• ANSHUMAN SWAROOP DAS •

Under the Supervision of Prof. PRATAP CHANDRA MANDAL

OUR OBJECTIVE

- Building a tool that can can be used to hide a text in an image


REALSTEGA
PLAN OF ACTION
COVER SELECTION
TRAINING
ML MODEL
EDGE DETECTION
EMBEDDING

REQUIREMENTS

System Requirements

- CPU: Dual core 64-bit 2.8 GHz 8.00GT/s CPUs.
- RAM: 2GB RAM (recommended 4GB RAM).
- Storage: 2GB for installation of Anaconda Navigator.

Software Requirements

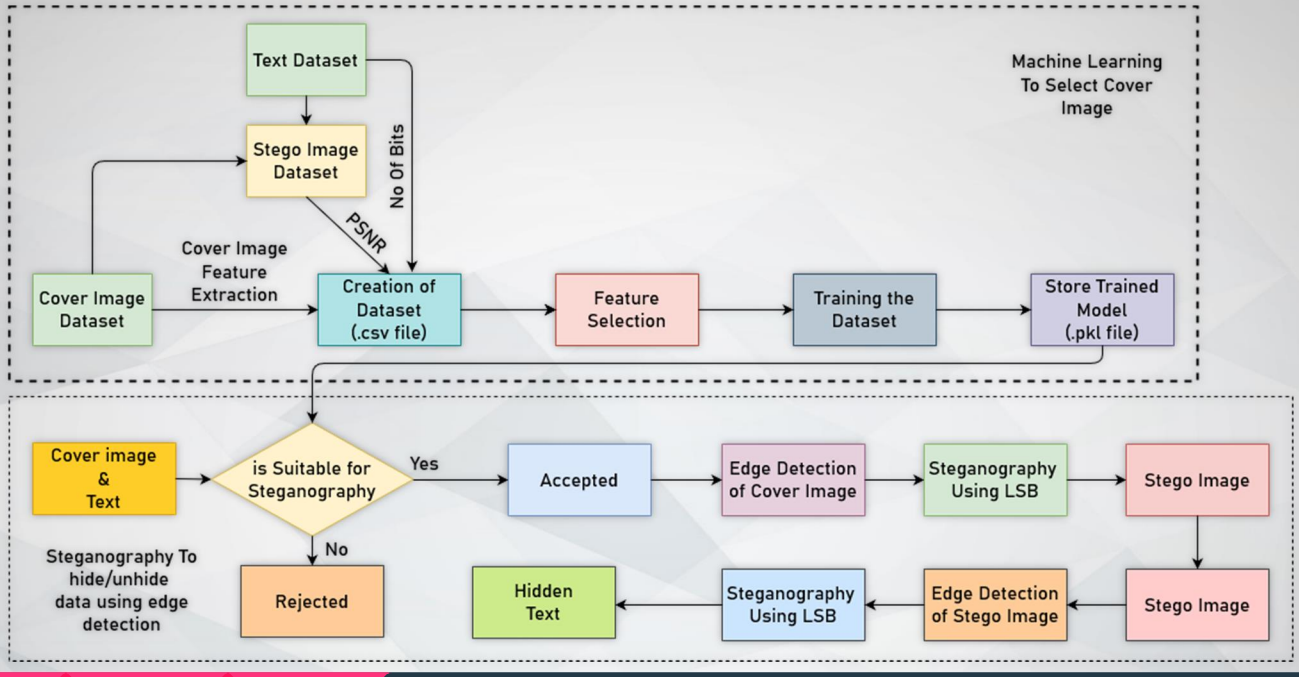
- OS: Windows 10, 8, 7.
- Anaconda Navigator v3.6.4 or higher.
- PyCharm and Jupyter Notebook IDEs.
- Any Browser like Google Chrome or Firefox.

TECHNOLOGY STACK

FRONT END

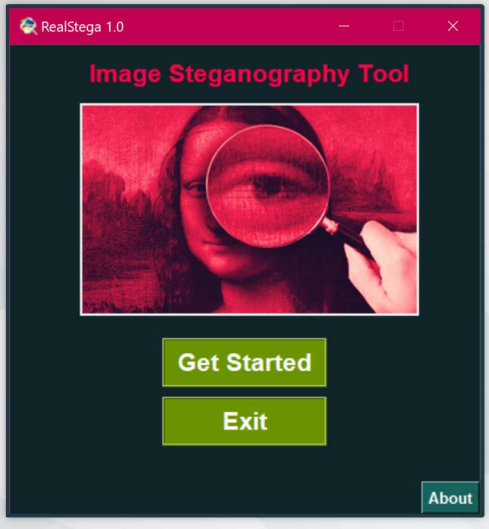
BACK END

matplotlib

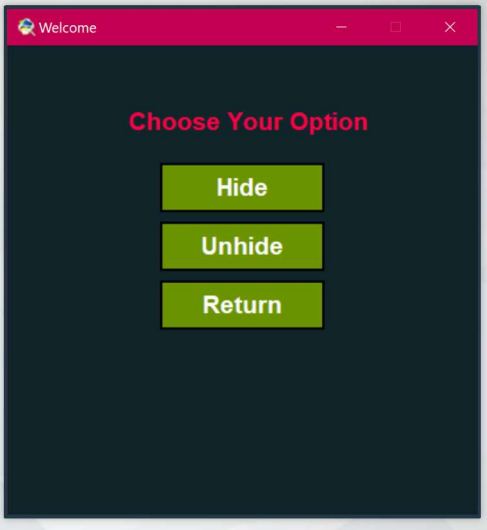


SCHEMATIC DIAGRAM

of our working model

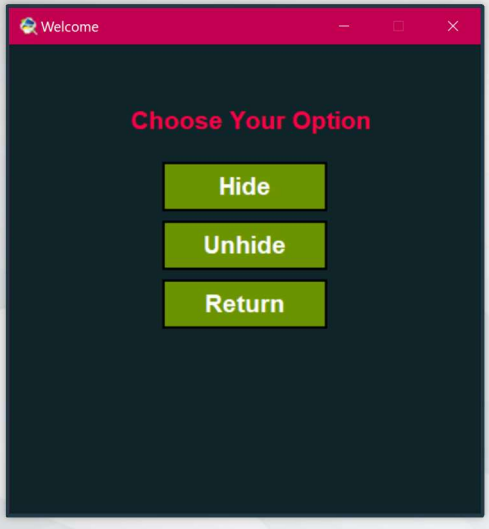


Start Window

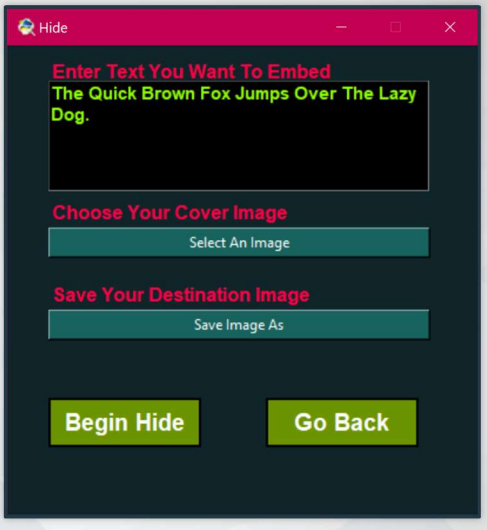


Options Window

FRONT END
of application

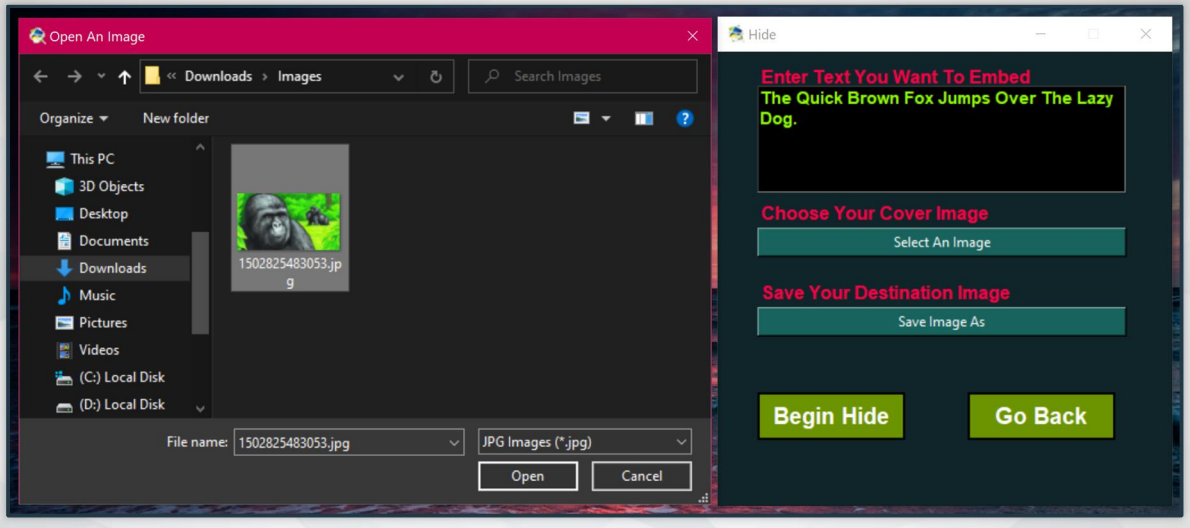


Options Window

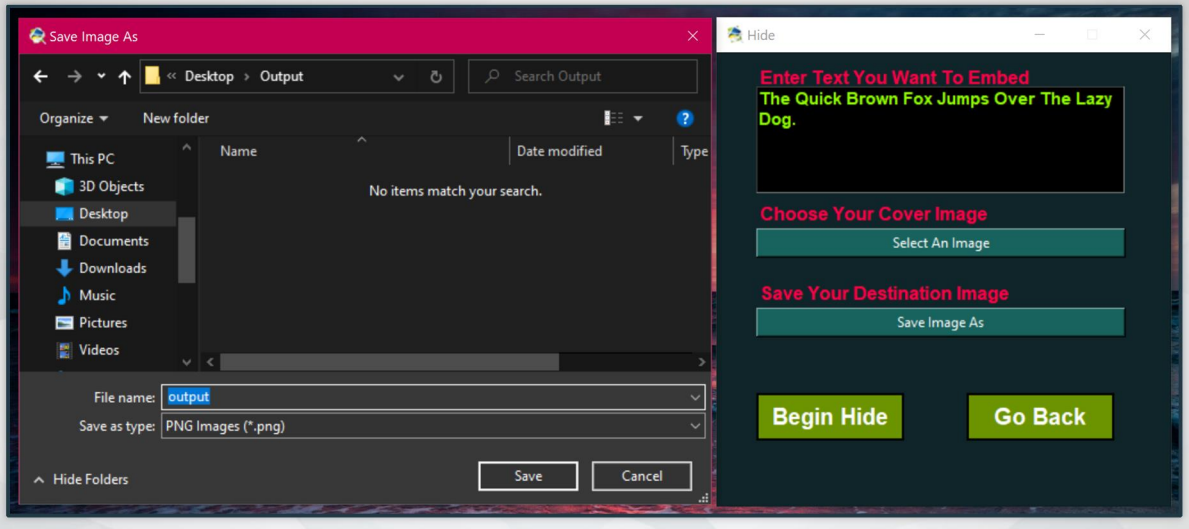


Hide Window

FRONT END
of application

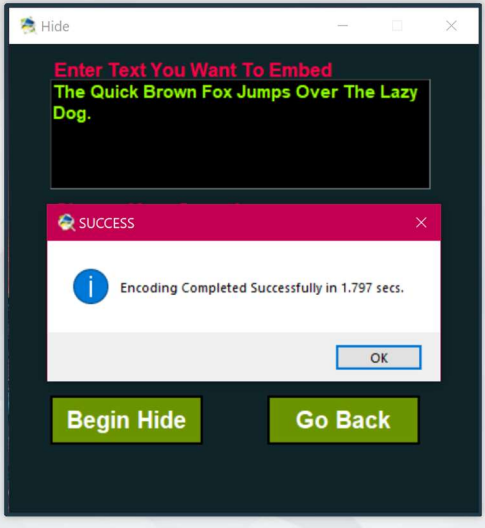


Choosing an Image for Hiding the Text



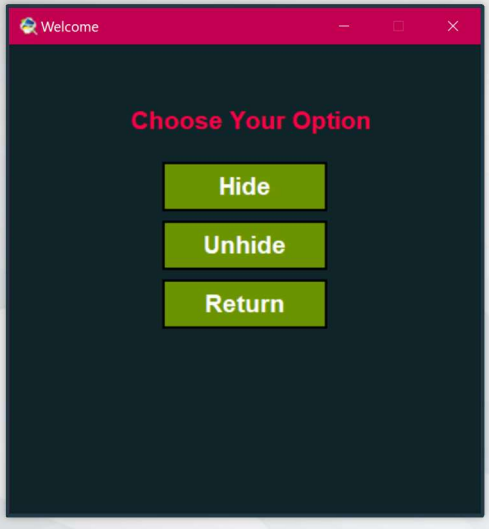
Choosing the path for saving the Image with hidden Text

FRONT END
of application

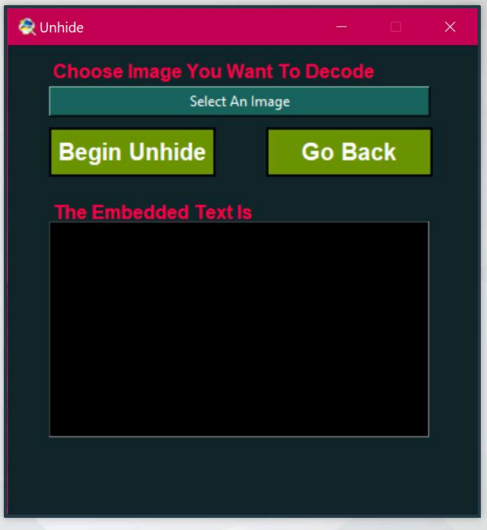


After Successful Hiding of Text in the Image

FRONT END
of application

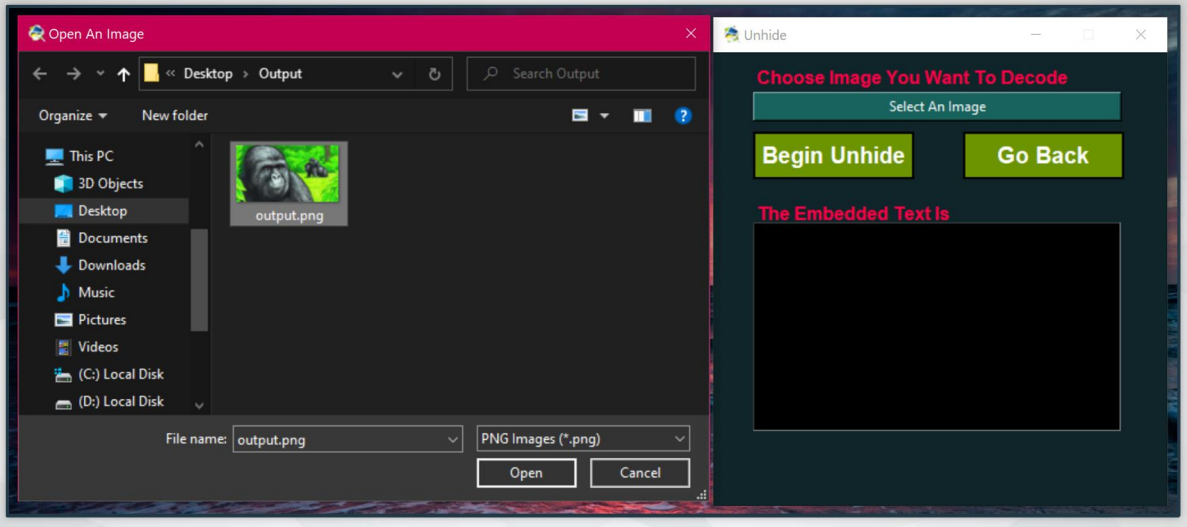


Options Window



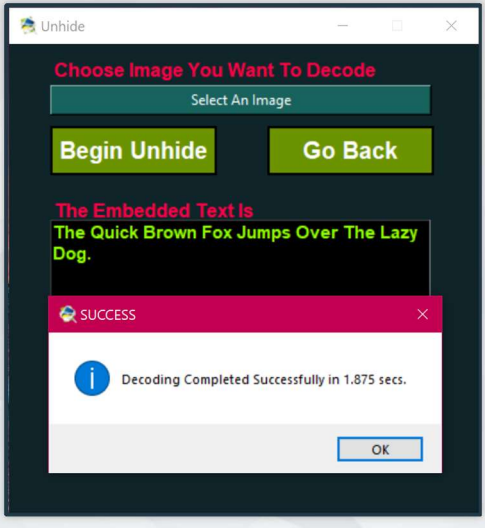
Unhide Window

FRONT END
of application



Choosing an Image with a Text Embedded

FRONT END
of application

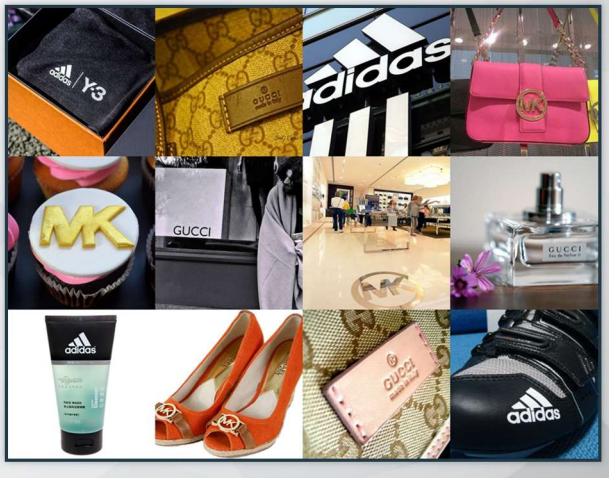


After Successful Unhiding of Text in the Image

FRONT END
of application

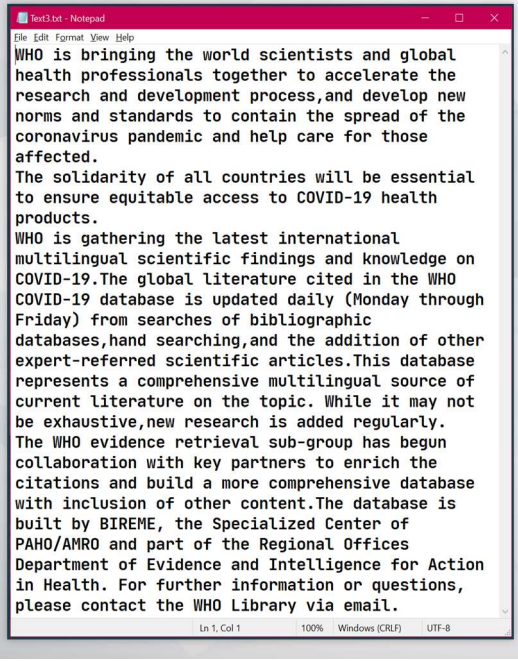
OUR DATASET
of cover images

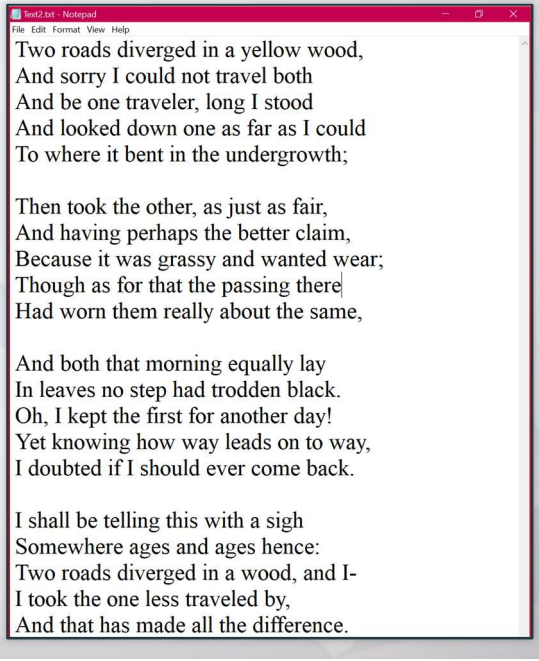
Samples of Kaggle Steganography Dataset



Samples of TopLogo-10 Dataset

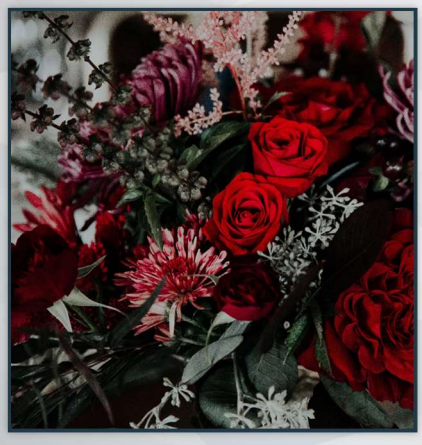
TEXT FILE
samples



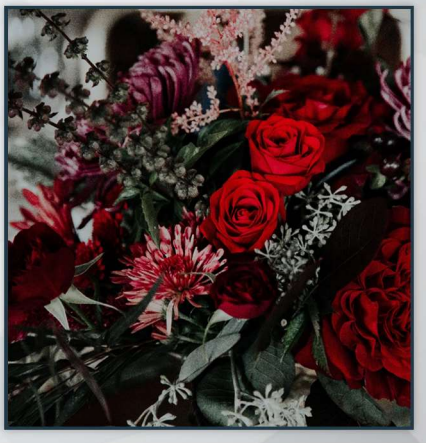


CREATING A STEGO IMAGE

from cover image



Cover Image



Stego Image

FEATURES EXTRACTION

01	Brightness
02	Contrast
03	Pixels
04	Number of Bits
05	JPEG Quality

PSNR CALCULATION

- We use these formulae to calculate the PSNR values using cover and stego images.

$$MSE = \left(\frac{1}{MN}\right) \sum_{i=1}^{M} \sum_{j=1}^{N} \left(X_{ij} - \overline{X_{ij}}\right)^2$$

$$PSNR = 10.\log_{10} \frac{I^2}{MSE} \quad db$$

- Where,

X_{ij} is the i^{th} row and the j^{th} column pixel in the original image,

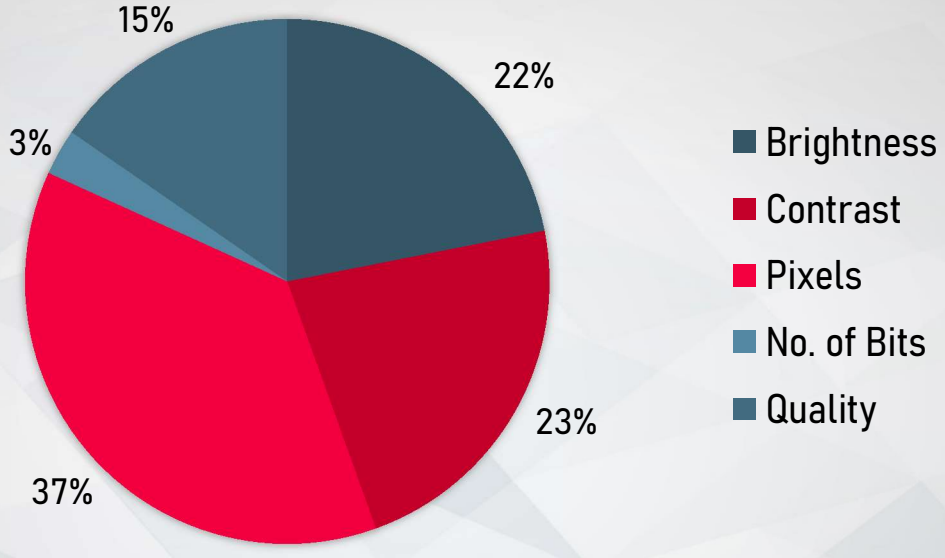
$\overline{X_{ij}}$ is the i^{th} row and the j^{th} column pixel in the reconstructed (stego) image,

- M and N are the height and the width of the image.
- I is the dynamic range of pixel values, or the maximum value that a pixel can take (equals to (255) for 8-bit images).

CREATING THE DATASET

Brightness	Contrast	Pixels	NoOfBits	Quality	Ans
0.270653	58.566865	1705600	1488	90	1
0.382019	88.083912	480200	1488	80	1
0.809317	81.171926	640000	1488	75	1
0.280167	53.957373	920320	1488	80	1
0.240583	44.726679	618240	1488	80	1

First Five Rows Generated from our CSV Dataset



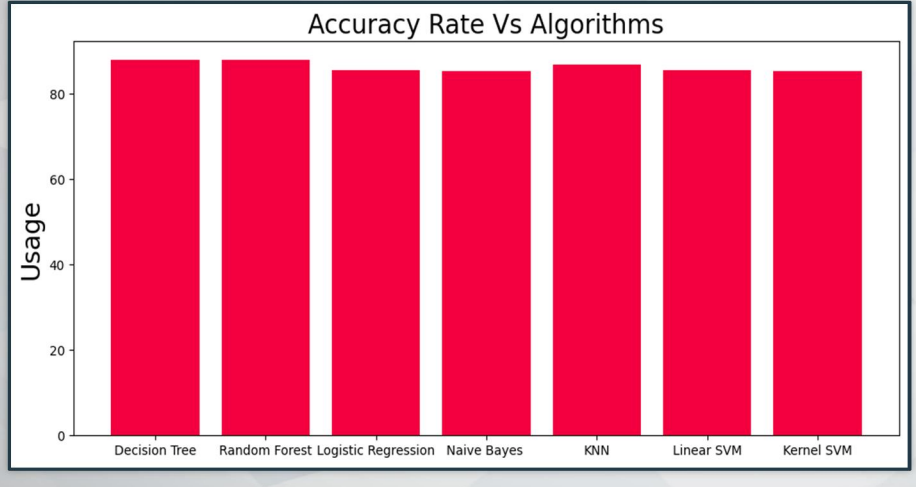
IMPORTANCE CHART

of features

MACHINE LEARNING

- **Decision Tree**
- Random Forest
- Logistic Regression
- Naive Bayes
- K Nearest Neighbours
- Support Vector Machine (Linear)
- Kernel SVM (Radial Basis Function)

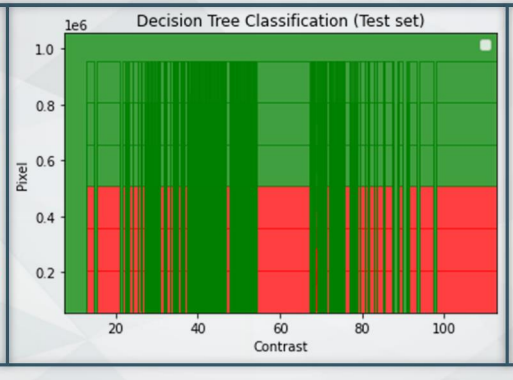
ACCURACY CHART
of algorithms

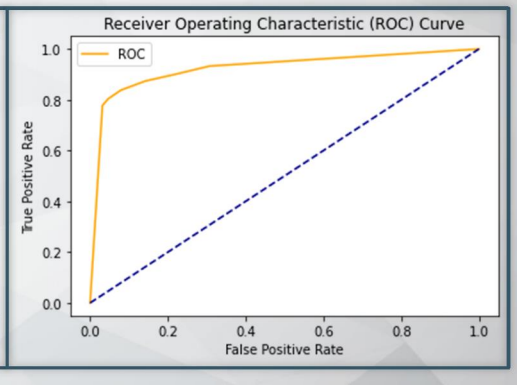


Algorithms	Accuracy
Decision Tree	**87.85%**
Random Forest	87.84%
Logistic Regression	85.39%
Naive Bayes	85.36%
K Nearest Neighbours	86.87%
Linear SVM	85.39%
Kernel SVM	85.31%

ML GRAPHS & DATA

DECISION TREE RESULTS





Confusion Matrix: [[1887 162]
[347 1792]]

Accuracy Score:
87.84622731614136%

AUC: 0.92

EDGE DETECTION

- Edges are significant local changes of intensity in an image.
- Edges typically occur on the boundary between two different regions in an image.
- Produce a line drawing of a scene from an image of that scene.
- Important features can be extracted from the edges of an image (e.g., corners, lines, curves).

STEPS FOR EDGE DETECTION

- **Smoothing**

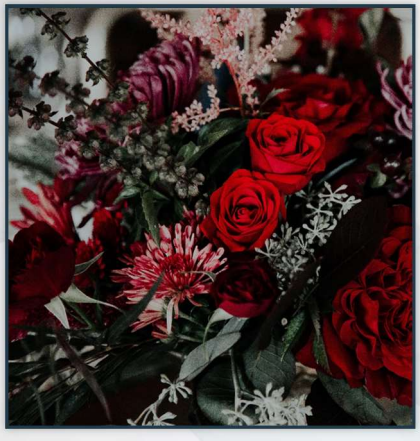
Suppress as much noise as possible, without destroying the true edges.

- **Enhancement**

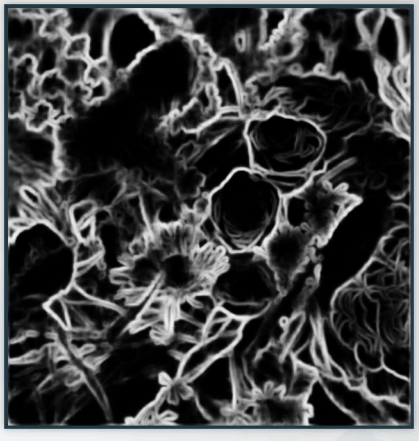
Apply a filter to enhance the quality of the edges in the image.

- **Localization**

Determine the exact location of an edge.



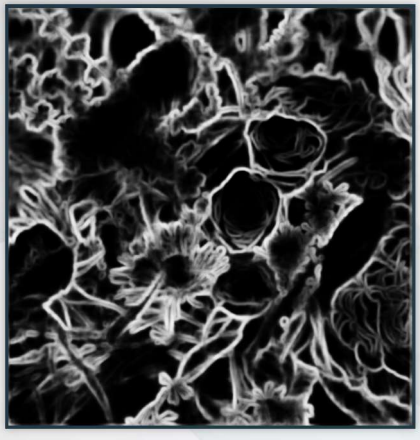
Original Image



HED Image

EDGE DETECTION

of cover image



HED Image

V

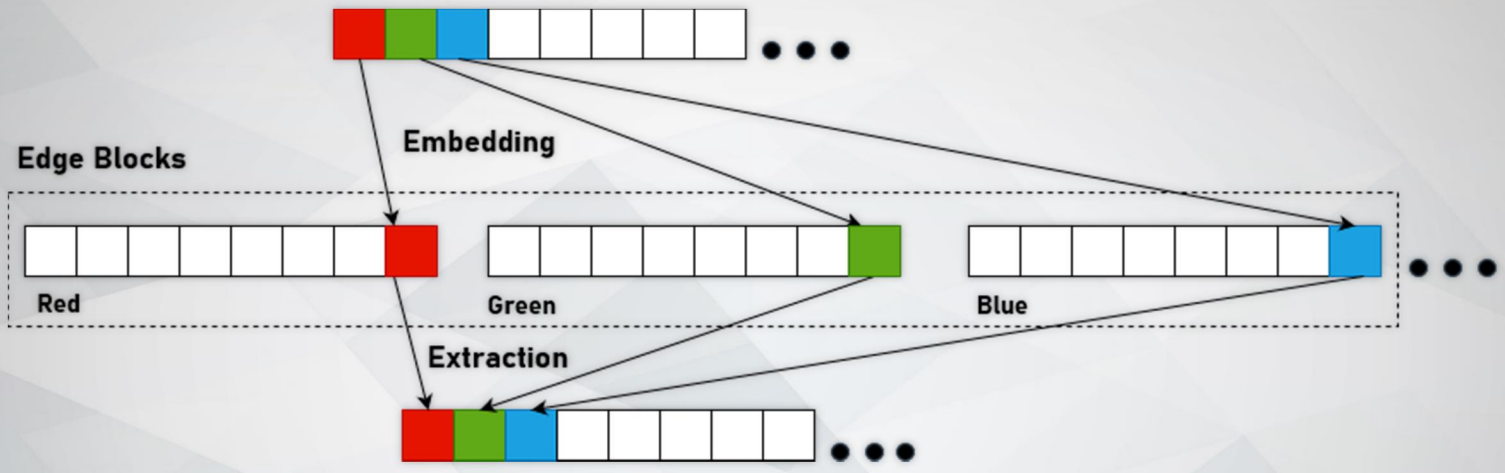
Canny Image

HED v CANNY

comparison

HIDING DATA IN EDGES

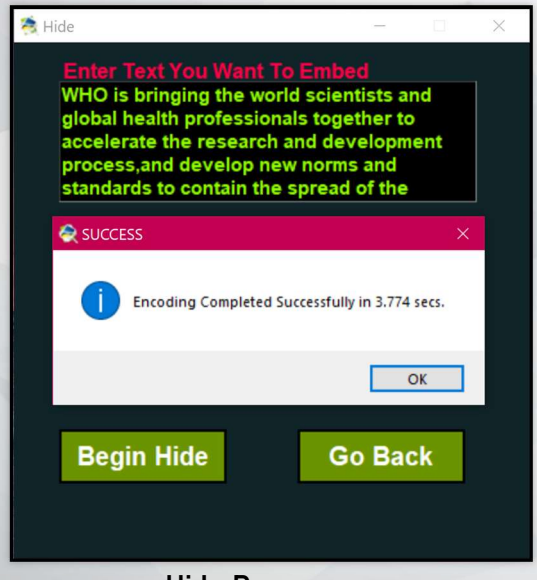
- We used Least Significant Bit (LSB) to embed message bits into the HED output.
- LSB replaces the least significant bit of the pixel with our message bit.
- Hiding the information in the edges of the image helps in non-sequential embedding so that it cannot be easily detected by attackers.



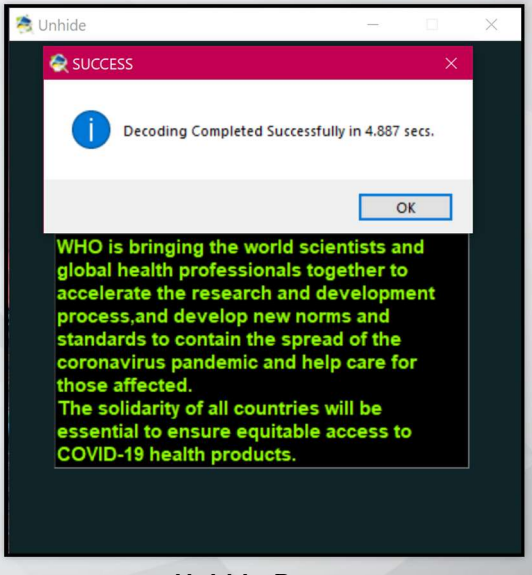
EDGE BASED LSB

illustration

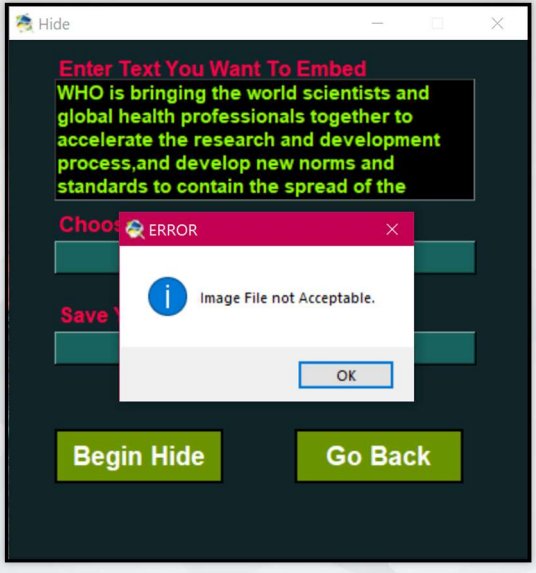
RESULTS
of an acceptable image



Hide Process



Unhide Process



Unsuccessful Hiding Process

RESULTS

of an unacceptable image

FUTURE ENHANCEMENTS

- Improving the accuracy by training a bigger dataset.
- Considering the higher number of features of the image.
- Improving the embedding of special characters in image.
- Enhancing Edge Detection Algorithm to detect Edges better and give better results while hiding information.

THANK YOU