CSE4019 - Image Processing

Project Report

Destruction of Steganography using EDSR and SRGAN

By

19BCE1290	Sabrina Marshal
19BCE1630	Sharon Immanuel

B. Tech Computer Science and Engineering

Submitted to

Dr.S.Geetha

School of Computer Science and Engineering

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

April 2022

ABSTRACT

The art of hiding secret messages beneath seemingly innocent digital media is known as digital steganography. Steganalysis is a method that seeks to counter steganography by identifying and extracting or deleting the secret information. There are a variety of steganalysis algorithm available nowadays. Many algorithms are limited to only a subset of available photos or produce a high proportion of false positives. These algorithms obstruct the transmission of suspicious images and then filter the image to remove steganographic content. This technique may result in a high number of false positives and may also degrade the image. In our project, we use super resolution to remove the steganographic content from the image. To enhance the steganographic image, we employ EDSR (Enhanced deep super resolution network). The image is then fine-tuned using SRGAN (Super Resolution Generative Adversarial Network). Image Super Resolution is the process of enhancing an image's resolution from low-resolution (LR) to high-resolution (HR). The steganographic content from the image is removed when applying super resolution.

INTRODUCTION

The basic purpose of steganography is to communicate securely while being fully invisible, avoiding suspicion of covert data transmission. Unfortunately, the majority of steganography applications and research revolves around illegitimate purposes. Illegal use of steganography was also discovered on a global scale, involving national security. Digital tools for steganography are now freely accessible to average computer users. Both criminal and legitimate users can use steganography software to conceal messages so that they are not discovered in transit.

Steganalysis is the art and science of detecting hidden messages. It can be used to prevent the transmission of illegal hidden messages. Steganalysis can be divided into two types: targeted and universal. When the steganographic algorithm is known, targeted steganalysis can be used. Universal steganalysis does not require knowledge of the method in use, therefore it will use many attributes to analyze a variety of well-known stego algorithms.

These algorithms are used to detect traditional steganography algorithms, such as Least Significant Bit (LSB) steganography. However, these methods struggle to detect advanced steganography algorithms. Many algorithms are limited to only a subset of available photos or produce a high proportion of false positives. A more feasible policy is to filter suspicious images prior to reception by the host machine. We need to optimally destroy steganographic content without compromising the perceptual quality of the original image.

So in our project we remove the steganographic content in the image by using super resolution. We use EDSR (Enhanced deep super resolution network) to enhance the steganographic image. In this process the steganographic content is removed. Then the image is fine tuned with SRGAN.

The technique of recovering high-resolution (HR) images from low-resolution (LR) photos is known as image super-resolution (SR). Deep learning-based SR models have been actively explored in recent years as deep learning techniques have advanced, and they typically achieve state-of-the-art performance on several SR benchmarks. EDSR is based on SRResNet architecture. It is SRResNet optimized by analyzing and removing unnecessary modules to simplify the network architecture.ResNet, short for Residual Network, is a specific type of neural network. ResNet allows you to train hundreds or even thousands of layers while still achieving excellent results.

The goal of SRGAN is to recover finer textures from the image so that the image's quality is not damaged when it is upscaled. It is similar to GAN architectures, the Super Resolution GAN also contains two parts Generator and Discriminator.

LITERATURE SURVEY

LSB based steganography is a simple and straightforward approach to steganography. In this technique the message is embedded into the least significant bit plane of the image. Since this will affect each pixel by +-1, it is generally assumed with good reason that the degradation caused by this embedding process would be perceptually transparent. Hence there are a number of LSB based steganography.[1]

Corley et al proposed an intelligent image steganography destruction model, Deep Digital Steganography Purifier (DDSP), which utilizes a Generative Adversarial Network (GAN) trained to remove steganographic content from images while maintaining high perceptual quality.DDSP model removes the greatest amount of steganographic content from images while maintaining the highest visual image quality in comparison to other state-of-the-art image steganography destruction methods. Deep Digital Steganography Purifier (DDSP) consists of a similar architecture to SRGAN. However instead of using a large ResNet, DDSP utilizes a pretrained autoencoder as the generator network in the GAN framework to remove the steganographic content from images without sacrificing image quality.[2]

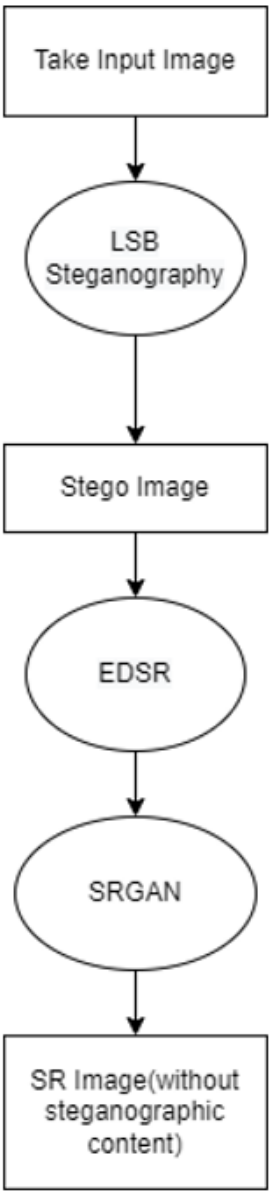
Recently, several models based on deep neural networks have achieved great success in terms of both reconstruction accuracy and computational performance for single image super-resolution. In these methods, the low resolution (LR) input image is upscaled to the high resolution (HR) space using a single filter, commonly bicubic interpolation, before reconstruction. This means that the super-resolution (SR) operation is performed in HR space. The recovery of a high resolution (HR) image or video from its low resolution (LR) counter part is topic of great interest in digital image processing. This task, referred to as super-resolution (SR), finds direct applications in many areas such as HDTV, medical imaging, satellite imaging, face recognition and surveillance. [3]

Ledig, et al proposed SRGAN which is a GAN-based network optimized for a new perceptual loss. They replace the MSE-based content loss with a loss calculated on feature maps of the VGG network, which are more invariant to changes in pixel space. Their ultimate goal is to train a generating function G that estimates for a given LR input image its corresponding HR counterpart. To achieve this, they train a generative model G with the goal of fooling a differentiable discriminator D that is trained to distinguish super-resolved images from real images. With this approach the generator can learn to create solutions that are highly similar to real images and thus difficult to classify by D.[4]

Lim et al proposed an enhanced super-resolution algorithm. By removing unnecessary modules from conventional ResNet architecture, they achieve improved results while making their model compact. They also employ residual scaling techniques to stably train large models. They proposed a single-scale model which surpasses current models and achieves state-of-the-art performance. Furthermore, they developed a multi-scale super-resolution network to reduce the model size and training time. With scale-dependent modules and shared main network, the multi-scale model can

effectively deal with various scales of super-resolution in a unified framework. The proposed single-scale and multi-scale models have achieved the top ranks in both the standard benchmark datasets and the DIV2K dataset.[5]

SYSTEM DESIGN



IMPLEMENTATION OF SYSTEM

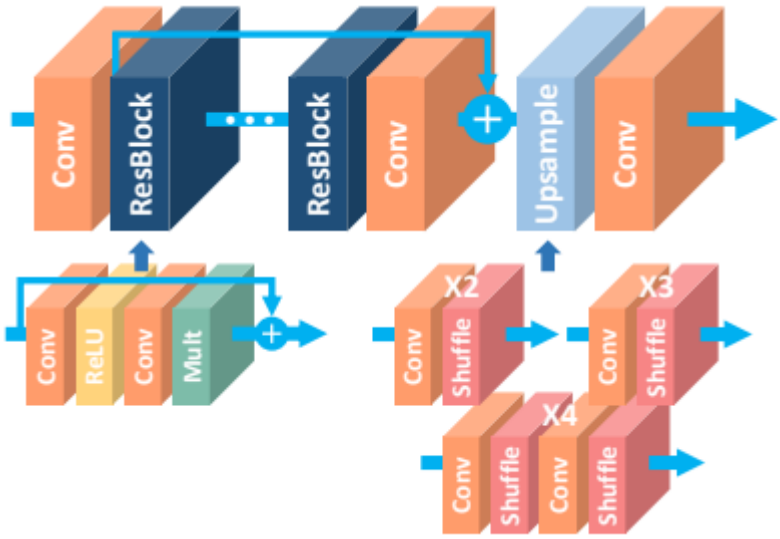
Super-resolution is the process of recovering a high-resolution (HR) image from a low-resolution (LR) image. Super-resolution is an ill-posed problem since a large number of solutions exist for a single pixel in an LR image. Simple approaches like bilinear or bicubic interpolation use only local information in an LR image to compute pixel values in the corresponding SR image.

Supervised machine learning approaches, on the other hand, learn mapping functions from LR images to HR images from a large number of examples. Super-resolution models are trained with LR images as input and HR images as target. The mapping function learned by these models is the inverse of a downgrade function that transforms HR images to LR images. Downgrade functions can be known or unknown.

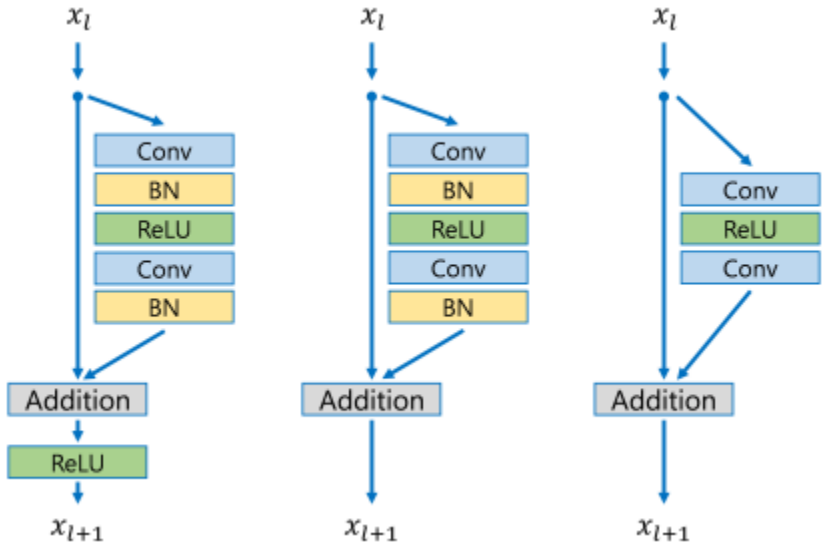
Known downgrade functions are used in image processing pipelines, for example, like bicubic downsampling. With known downgrade functions, LR images can be automatically obtained from HR images. This allows the creation of large training datasets from a vast amount of freely available HR images which enables self-supervised learning.

Super-resolution requires that most of the information contained in an LR image must be preserved in the SR image. Super-resolution models therefore mainly learn the residuals between LR and HR images. Residual network designs are therefore of high importance. Identity information is conveyed via skip connections whereas reconstruction of high frequency content is done on the main path of the network. These layers are often residual blocks as in ResNet. Local skip connections in residual blocks make the network easier to optimize and therefore support the construction of deeper networks.

Based on SRResNet architecture, EDSR is optimized by analyzing and removing unnecessary modules to simplify the network architecture.



Its residual block design differs from that of ResNet. Batch normalization layers have been removed together with the final ReLU activation.



The EDSR authors argue that batch normalization loses scale information of images and reduces the range flexibility of activations. Removal of batch normalization layers not only increases super-resolution performance but also reduces GPU memory up to 40% so that significantly larger models can be trained.

EDSR uses a single sub-pixel upsampling layer for super-resolution scales (i.e. upsampling factors) ×2 and ×3 and two upsampling layers for scale ×4.

For training EDSR and WDSR models the DIV2K dataset was used. It is a dataset of LR and HR image pairs with a large diversity of contents. LR images are available for different downgrade functions. They use bicubic downsampling. There are 800 training HR images and 100 validation HR images. For data augmentation, random crops, flips and rotations are made to get a large number of different training images. A DIV2K data loader automatically downloads DIV2K images for a given scale and downgrade function and provides LR and HR image pairs as tf.data.Dataset.

The pixel-wise L2 loss and the pixel-wise L1 loss are frequently used loss functions for training super-resolution models. They measure the pixel-wise mean squared error and the pixel-wise mean absolute error, respectively, between an HR image IHR and an SR image ISR. The pixel-wise. L2 loss directly optimizes PSNR, an evaluation metric often used in super-resolution competitions. Experiments have shown that the pixel-wise L1 loss can sometimes achieve even better performance and is therefore used for EDSR and WDSR training. A major problem with pixel-wise loss functions is that they lead to poor perceptual quality. Generated SR images often lack high-frequency content, realistic textures and are perceived blurry. This problem is addressed with perceptual loss functions.

A milestone paper for generating SR images with better perceived quality is Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network (SRGAN). The authors use a perceptual loss function composed of a content loss and an adversarial loss. The content loss compares deep features extracted from SR and HR images with a pre-trained VGG network.
They also train their super-resolution model as generator G in a generative adversarial network (GAN). The GAN discriminator D is optimized for discriminating SR from HR images whereas the generator is optimized for generating more realistic SR images in order to fool the discriminator. They combine the generator loss with the content loss to a perceptual loss which is used as optimization target for super-resolution model training. Instead of training the super-resolution model i.e. the generator from scratch in a GAN, they pre-train it with a pixel-wise loss and fine-tune the model with a perceptual loss.

The running of the model -

We start with an image of size 500x500.

We then resize the image to 128x128 to better show the effect of super resolution.

```
import cv2
import numpy as np
img_array = cv2.imread("im63.png")
img_array = cv2.resize(img_array, (128,128))
cv2.imwrite("image.png", img_array)
```

```
True
```

We use this image as the input cover image to hide our steganographic content.

```python
def Encode(src, message, dest):

    img = Image.open(src, 'r')
    width, height = img.size
    array = np.array(list(img.getdata()))

    if img.mode == 'RGB':
        n = 3
    elif img.mode == 'RGBA':
        n = 4
    total_pixels = array.size//n

    b_message = ''.join([format(ord(i), "08b") for i in message])
    req_pixels = len(b_message)

    if req_pixels > total_pixels:
        print("ERROR: Need larger file size")

    else:
        index=0
        for p in range(total_pixels):
            for q in range(0, 3):
                if index < req_pixels:
                    array[p][q] = int(bin(array[p][q])[2:9] + b_message[index], 2)
                    index += 1

        array=array.reshape(height, width, n)
        enc_img = Image.fromarray(array.astype('uint8'), img.mode)
        enc_img.save(dest)
        print("Image Encoded Successfully")
```

```
def Stego():
    print("1: Encode")
    print("2: Decode")

    func = input()

    if func == '1':
        print("Enter Source Image Path")
        src = input()
        print("Enter Message to Hide")
        message = input()
        print("Enter Destination Image Path")
        dest = input()
        print("Encoding...")
        Encode(src, message, dest)

    elif func == '2':
        print("Enter Source Image Path")
        src = input()
        print("Decoding...")
        Decode(src)

    else:
        print("ERROR: Invalid option chosen")
```

```
Stego()
```

```
1: Encode
2: Decode
1
Enter Source Image Path
image.png
Enter Message to Hide
Testing out the destruction of steganography using super resolution
Enter Destination Image Path
secret_image.png
Encoding...
Image Encoded Successfully
```

Image with secret message embedded:

Decoding the image and checking for message:

```
def Decode(src):

    img = Image.open(src, 'r')
    array = np.array(list(img.getdata()))

    if img.mode == 'RGB':
        n = 3
    elif img.mode == 'RGBA':
        n = 4
    total_pixels = array.size//n

    hidden_bits = ""
    for p in range(total_pixels):
        for q in range(0, 3):
            hidden_bits += (bin(array[p][q])[2:][-1])

    hidden_bits = [hidden_bits[i:i+8] for i in range(0, len(hidden_bits), 8)]

    message = ""
    for i in range(len(hidden_bits)):
        message += chr(int(hidden_bits[i], 2))
    with open("Output.txt", "w") as text_file:
      text_file.write("Message: {0}".format(message))
```

```
Stego()
```

```
1: Encode
2: Decode
2
Enter Source Image Path
secret_image.png
Decoding...
```

First few LSB bits of image:

Output.txt ×

```
1 Message: Testing out the destruction of steganography using super resolution¯f□□âA□□□éôÀÅ□È□Z□ËGØ□ÇÔänôYVëÛ□□H□DÊ
2 ëg□°□ç□□ÊPE□<![ýk□¢□¶□èô@/□aû±□□-#S¼/5□²□□xWPÐó¥®i□§R<Bã□ûo\□□ÌÅv=□:(□)·□È/ú□áÙÇe±ªü(□□îËúuÆ1U
3 4á□□§÷s¶Ñ□<²□ær□û!H□□s□xc8Qô□ø¦«Æ□¼Nò@ÔßÃ□□□öà□um]ÓBñía9§!dVy□□  BóÕ¡'ÀL□□TV□0i¯72□q¸¸¼µ#□»O%□□9õO□ÌS¼□\¸·]&à+È Ö□á□□Éq□Ò[
4 □0Ë□§±®□W□bÏì;UóÇí□|6¢f ø□÷%³□ð□□Zé5'û^08'Æ□¢®P□D^æ□□¦s□□x□LnÙ□NÖZQ·}èÙ□":□□§&é□wS»4□déR?áEå!|ájÁHð□Â}½Ý*□aÖu□ÆXä□d÷□l□FÓil
5 ë□□□°□mÊb□þ)ûÔ¯□□<²¢6Ãé□ªë□éÜßÃ~Z.□
6 □B$c|áûª^h□Sú+{V□çÙÒ□ì
```

We take the secret message image and pass it as input to a pretrained EDSR model and fine tune it with a pretrained SRGAN model.

```python
import os
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np
import cv2
import tensorflow as tf
from tensorflow.keras.layers import Add, Conv2D, Input, Lambda
from tensorflow.keras.models import Model

DIV2K_RGB_MEAN = np.array([0.4488, 0.4371, 0.4040]) * 255

def load_image(path):
    return np.array(Image.open(path))

def resolve_single(model, lr):
    return resolve(model, tf.expand_dims(lr, axis=0))[0]

def resolve(model, lr_batch):
    lr_batch = tf.cast(lr_batch, tf.float32)
    sr_batch = model(lr_batch)
    sr_batch = tf.clip_by_value(sr_batch, 0, 255)
    sr_batch = tf.round(sr_batch)
    sr_batch = tf.cast(sr_batch, tf.uint8)
    return sr_batch

def normalize(x, rgb_mean=DIV2K_RGB_MEAN):
    return (x - rgb_mean) / 127.5

def denormalize(x, rgb_mean=DIV2K_RGB_MEAN):
    return x * 127.5 + rgb_mean
```

```python
def edsr(scale, num_filters=64, num_res_blocks=8, res_block_scaling=None):
    """Creates an EDSR model."""
    x_in = Input(shape=(None, None, 3))
    x = Lambda(normalize)(x_in)

    x = b = Conv2D(num_filters, 3, padding='same')(x)
    for i in range(num_res_blocks):
        b = res_block(b, num_filters, res_block_scaling)
    b = Conv2D(num_filters, 3, padding='same')(b)
    x = Add()([x, b])

    x = upsample(x, scale, num_filters)
    x = Conv2D(3, 3, padding='same')(x)

    x = Lambda(denormalize)(x)
    return Model(x_in, x, name="edsr")

def res_block(x_in, filters, scaling):
    """Creates an EDSR residual block."""
    x = Conv2D(filters, 3, padding='same', activation='relu')(x_in)
    x = Conv2D(filters, 3, padding='same')(x)
    if scaling:
        x = Lambda(lambda t: t * scaling)(x)
    x = Add()([x_in, x])
    return x
```

```python
def upsample(x, scale, num_filters):
    def upsample_1(x, factor, **kwargs):
        """Sub-pixel convolution."""
        x = Conv2D(num_filters * (factor ** 2), 3, padding='same', **kwargs)(x)
        return Lambda(pixel_shuffle(scale=factor))(x)

    if scale == 2:
        x = upsample_1(x, 2, name='conv2d_1_scale_2')
    elif scale == 3:
        x = upsample_1(x, 3, name='conv2d_1_scale_3')
    elif scale == 4:
        x = upsample_1(x, 2, name='conv2d_1_scale_2')
        x = upsample_1(x, 2, name='conv2d_2_scale_2')

    return x
```

```
%matplotlib inline

def resolve_and_plot(model_pre_trained, model_fine_tuned, lr_image_path):
    lr = load_image(lr_image_path)

    sr_pt = resolve_single(model_pre_trained, lr)
    sr_ft = resolve_single(model_fine_tuned, lr)

    sr_ft = np.array(sr_ft)
    cv2.imwrite("srimage.png",cv2.cvtColor(sr_ft, cv2.COLOR_RGB2BGR))

    plt.figure(figsize=(20, 20))

    model_name = model_pre_trained.name.upper()
    images = [lr, sr_pt, sr_ft]
    titles = ['Original image', f'SR ({model_name}, pixel loss)', f'SR ({model_name}, perceptual loss)']
    positions = [1, 3, 4]

    for i, (image, title, position) in enumerate(zip(images, titles, positions)):
        plt.subplot(2, 2, position)
        plt.imshow(image)
        plt.title(title)
        plt.xticks([])
        plt.yticks([])

weights_dir = ''
```

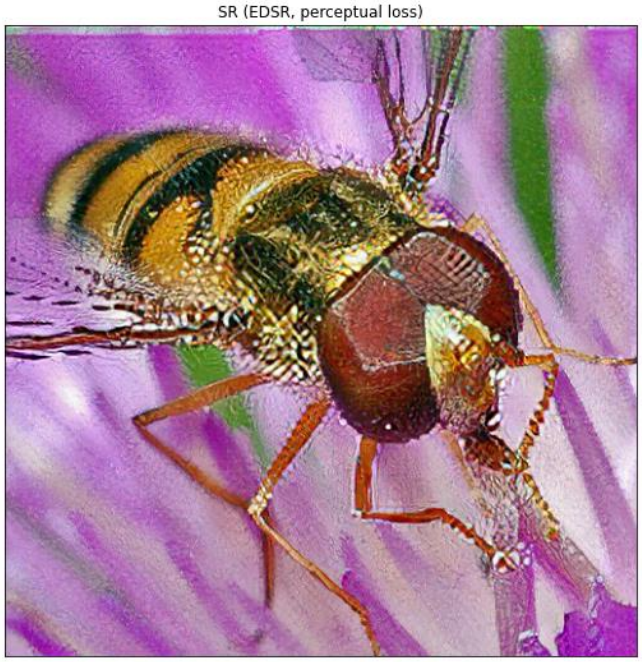
```
edsr_pre_trained = edsr(scale=4, num_res_blocks=16)
edsr_pre_trained.load_weights(os.path.join(weights_dir, 'weights-edsr-16-x4.h5'))

edsr_fine_tuned = edsr(scale=4, num_res_blocks=16)
edsr_fine_tuned.load_weights(os.path.join(weights_dir, 'weights-edsr-16-x4-fine-tuned.h5'))

resolve_and_plot(edsr_pre_trained, edsr_fine_tuned, 'secret_image.png')
```

Original low resolution image-

Super resolution with EDSR-

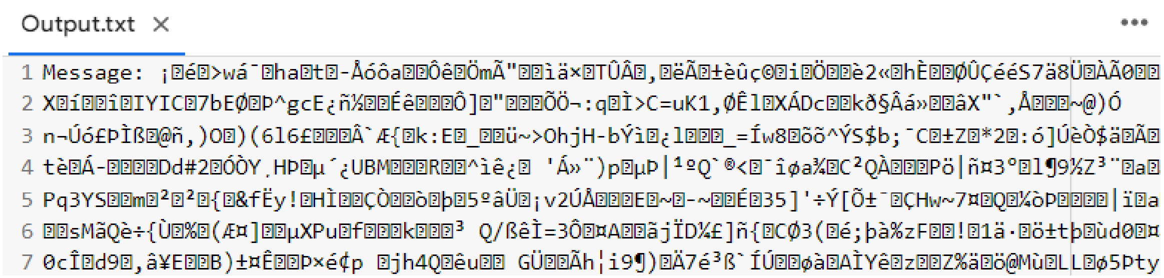
Super resolution by fine tuning with SRGAN-

SR (EDSR, perceptual loss)



RESULTS AND DISCUSSIONS

The super resolution image is obtained using the EDSR and SRGAN models. To check for the secret message, we look at the least significant bits of the image.



As we can see, the steganographic content was completely destroyed in the super resolution process. The SR process can thus be used in the destruction of steganography.

The trained EDSR model can now be used to create SR images from LR images. One can clearly see how fine-tuning with a perceptual loss creates more realistic textures in SR images compared to training with a pixel-wise loss alone.

CONCLUSION AND FUTURE WORK

We created a steganography destruction model that incorporates EDSR and SRGAN. It removes the majority of steganographic content from photos while preserving the best visual image possible.We intend to expand the model in the future to purify inputs of various types, sizes, and color spaces. We can also train the model with more epochs and a larger dataset to make it more robust, allowing it to be implemented for a real-world steganography destruction system.

REFERENCES

[1] R. Chandramouli, Nasir Memon. Analysis of LSB based image steganography techniques. 2001.

[2] Isaac Corley, Jonathan Lwowski, Justin Hoffman, Booz Allen Hamilton. Destruction of Image Steganography using Generative Adversarial Networks. 2019.

[3] Wenzhe Shi , Jose Caballero, Ferenc Huszar´, Johannes Totz, Andrew P. Aitken , Rob Bishop, Daniel Rueckert, Zehan Wang. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. 2016.

[4] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, Kyoung Mu Lee. Enhanced Deep Residual Networks for Single Image Super-Resolution. 2017.

[5] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, ´ Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. 2017