INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

www.ijprems.com
editor@ijprems.com

Vol. 03, Issue 04, April 2023, pp : 167-174

e-ISSN : 2583-1062

Impact Factor : 5.725

GENERATIVE ADVERSARIAL NETWORKS IN IOT NETWORKS SECURITY FOR SECURITY DATA TRANSMISSION INTEGRATING IMAGE DATA HIDING

I. Sujiban[1], Mr. E.R. Ramesh[2], Ms. Sarika Jain[3], Dr. S. Geetha[4]

[1]M.Sc – CFIS, Department of Computer Science and Engineering, Dr. M.G.R Educational and Research Institute, Chennai 600 095, Tamilnadu, India

[2,3]Center of Excellence in Digital Forensics, Perungudi, Chennai 600 089, Tamilnadu, India

[4]Head of the Department, Department of Computer Science and Engineering, Dr. M.G.R Educational and Research Institute, Chennai 600 095, Tamilnadu, India

ABSTRACT

The Internet of Things (IoT) describes the network of physical objects or things that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet. Although several IoT devices are openly accessible to all in the network, it is extremely vital to be aware of the security risks and threats of cyber-attacks; therefore, it should be secured. In Cryptography, plain text is converted to encrypted text before it is sent, and it is converted to plain text after communication on the other side. Steganography is a method of hiding secret data, by embedding it into an audio, video, image, or text file. One technique is to hide data in bits that represent the same colour pixels repeated in a row in an image file. By applying the encrypted data to this redundant data in some inconspicuous way, the result will be an image file that appears identical to the original image but that has "noise" patterns of regular, unencrypted data. In this project it proposes to encrypt the IoT networks data by hiding the message inside an image file using image data hiding technique. We are going to incorporate the usage of convolutional neural networks in traditional image data hiding method to drastically increase the payload that can be transmitted through an image. Different convolutional parameters will be analysed to achieve the highest payload. Encryption and decryption of the data will be performed using the newly developed deep learning algorithm. Thus, in this project the convolutional networks will be trained in such a way to increase the payload of the data to be encrypted as well as safely decrypted to view the original message.

Keywords: IoT, Convolutional neural network, steganography, encryption.

1. INTRODUCTION

The Internet of Things (IoT’s) is the concept of connecting any device (so long as it has an on/off switch) to the Internet and to other connected devices. The IoT is a giant network of connected things and people – all of which collect and share data about the way they are used and about the environment around them. IoT devices contain sensors and mini-computer processors that act on the data collected by the sensors via machine learning. IoT devices share the sensor data they collect by connecting to an IoT gateway or other edge device where data is either sent to the cloud to be analyzed or analyzed locally. Sometimes, these devices communicate with other related devices and act on the information they get from one another. The devices do most of the work without human intervention, although people can interact with the devices -- for instance, to set them up, give them instructions or access the data. The internet of things helps people live and work smarter, as well as gain complete control over their lives. In addition to offering smart devices to automate homes, IoT is essential to business. IoT provides businesses with a real-time look into how their systems really work, delivering insights into everything from the performance of machines to supply chain and logistics operations. IoT enables companies to automate processes and reduce labor costs. It also cuts down on waste and improves service delivery, making it less expensive to manufacture and deliver goods, as well as offering transparency into customer transactions. As such, IoT is one of the most important technologies of everyday life, and it will continue to pick up steam as more businesses realize the potential of connected devices to keep them competitive. Essentially, IoT devices are mini computers, connected to the internet, and are vulnerable to malware and hacking.

2. LITERATURE SURVEY

(Steganalysis of Digital Images Using Deep Fractal Network) In the recent literature on steganalysis, it has been observed that a deeper network is, in general, preferred for detecting low tone embedding noise, e.g., SRNet. However, very recently, a deep model, called Fractal Net, became popular, which is based on self-similarity and grows deeper and wider by maintaining a balance between depth and width using a recurrent adaptation of a fundamental building block. In this work, the concept of the Fractal Net model has been exploited for steganalytic detection, where the embedded image has been used as input. In a practical scenario, it has been observed that steganalytic detection for test images is increasing if the width of the network can be increased with a certain proportion to the depth. The proposed deep network is designed by repeating a basic fractal block in such a way that a balance between the depth and width of the overall network can be maintained. A comprehensive set of experiments reveals that the proposed model outperformed the state-of-the-art results. An ablation study is also included to justify the proposed architecture in favour of its performance. (A Novel Image Steganography Method for Industrial Internet of Things Security) the rapid development of the Industrial Internet of Things (IIoT) and artificial intelligence (AI) brings new security threats by exposing secret and private data. Thus, information security has become a major concern in the communication environment of IIoT and AI, where security and privacy must be ensured for the messages between a sender and the intended recipient. To this end, we propose a method called HHOIWT for covert communication and secure data in the IIoT environment based on digital image steganography. The method embeds secret data in the cover images using a metaheuristic optimization algorithm called Harris hawks optimization (HHO) to efficiently select image pixels that can be used to hide bits of secret data within integer wavelet transforms. The HHO-based pixel selection operation uses an objective function evaluation depending on two phases: exploitation and exploration. The objective function is employed to determine an optimal encoding vector to transform secret data into an encoded form generated by the HHO algorithm. Several experiments are conducted to validate the performance of the proposed method with respect to visual quality, payload capacity, and security against attacks. The obtained results reveal that the HHO-IWT method achieves higher levels of security than the state-of-the-art methods and that it resists various forms of steganalysis. Thus, utilizing this approach can keep unauthorized individuals away from the transmitted information and solve some security challenges in the IIoT. (Securing Data in Internet of Things (IoT) Using Cryptography and Steganography) Internet of Things (IoT) is a domain wherein which the transfer of data is taking place every single second. The security of these data is a challenging task; however, security challenges can be mitigated with cryptography and steganography techniques. These techniques are crucial when dealing with user authentication and data privacy. In the proposed work, the elliptic Galois cryptography protocol is introduced and discussed. In this protocol, a cryptography technique is used to encrypt confidential data that came from different medical sources. Next, a Matrix XOR encoding steganography technique is used to embed the encrypted data into a low complexity image. The proposed work also uses an optimization algorithm called Adaptive Firefly to optimize the selection of cover blocks within the image. Based on the results, various parameters are evaluated and compared with the existing techniques. Finally, the data that is hidden in the image is recovered and is then decrypted.

(Secure Halftone Image Steganography Based on Feature Space and Layer) Syndrome-trellis codes (STCs) are commonly used in image steganographic schemes, which aim at minimizing the embedding distortion, but most distortion models cannot capture the mutual interaction of embedding modifications (MIEMs). In this article, a secure halftone image steganographic scheme based on a feature space and layer embedding is proposed. First, a feature space is constructed by a characterization method that is designed based on the statistics of 4 × 4-pixel blocks in halftone images. Upon the feature space, a generalized steganalyzer with good classification ability is proposed, which is used to measure the embedding distortion. As a result, a distortion model based on a hybrid feature space is constructed, which outerforms some state-of-the-art models. Then, as the distortion model is established on the statistics of local regions, a layer embedding strategy is proposed to reduce MIEM. It divides the host image into multiple layers according to their relative positions in 4 × 4 blocks, and the embedding procedure is executed layer by layer. In each layer, any two pixels are located at different 4×4 blocks in the original image, and the distortion model makes sure that the calculation of pixel distortions is independent. Between layers, the pixel distortions of the current layer are updated according to the previous embedding modifications, thus reducing the total embedding distortion. Comparisons with prior schemes demonstrate that the proposed steganographic scheme achieves high statistical security when resisting the state-of-the-art steganalysis. (Secure Data Delivery with Identity-based Linearly Homomorphic Network Coding Signature Scheme in IoT) with the appearance and flourishing development of the Internet of Things (IoT), wireless sensor networks technology has been attracting increasing attention. Network coding is an indispensable technology in the wireless sensor networks, which can improve network transmission throughput. However, a pollution attack is a serious security problem that must be faced in the process of data coding. Although the homomorphic network coding signature schemes can solve this troublesome, the high signature generation and verification cost of these schemes will reduce the transmission efficiency. In this paper, we propose an efficient identity-based linearly homomorphic network coding signature scheme for wireless sensor networks to guarantee data integrity and authenticity. In our scheme, the computation cost of signature generation and verification are both independent of the size of the data packet. The scheme is proved secure against existential forgery under adaptive chosen identity and adaptive chosen subspace attacks in random oracle model. Using Java pairing-based

Cryptography Library (JPBC), the simulation results illustrate that our scheme is more efficient in practical application.

3. EXISTING SYSTEM

In the recent literature on steganalysis, it has been observed that a deeper network is, in general, preferred for detecting low tone embedding noise, e.g., SRNet. However, very recently, a deep model, called Fractal Net, became popular, which is based on self-similarity and grows deeper and wider by maintaining a balance between depth and width using a recurrent adaptation of a fundamental building block. In this work, the concept of the Fractal Net model has been exploited for steganalytic detection, where the embedded image has been used as input. In a practical scenario, it has been observed that steganalytic detection for test images is increasing if the width of the network can be increased with a certain proportion to the depth. The proposed deep network is designed by repeating a basic fractal block in such a way that a balance between the depth and width of the overall network can be maintained. A comprehensive set of experiments reveals that the proposed model outperformed the state-of-the-art results. An ablation study is also included to justify the proposed architecture in favour of its performance.

4. PROPOSING SYSTEM

The Internet of Things (IoT) describes the network of physical objects or things that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet. Although several IoT devices are openly accessible to all in the network, it is extremely vital to be aware of the security risks and threats of cyber-attacks; therefore, it should be secured. In Cryptography, plain text is converted to encrypted text before it is sent, and it is converted to plain text after communication on the other side. Steganography is a method of hiding secret data, by embedding it into an audio, video, image, or text file. One technique is to hide data in bits that represent the same colour pixels repeated in a row in an image file. By applying the encrypted data to this redundant data in some inconspicuous way, the result will be an image file that appears identical to the original image but that has "noise" patterns of regular, unencrypted data. In this project it proposes to encrypt the IoT networks data by hiding the message inside an image file using image data hiding technique. We are going to incorporate the usage of convolutional neural networks in traditional image data hiding method to drastically increase the payload that can be transmitted through an image. Different convolutional parameters will be analysed to achieve the highest payload. Encryption and decryption of the data will be performed using the newly developed deep learning algorithm. Thus, in this project the convolutional networks will be trained in such a way to increase the payload of the data to be encrypted as well as safely decrypted to view the original message.

5. ARCHITECTURE DIAGRAM

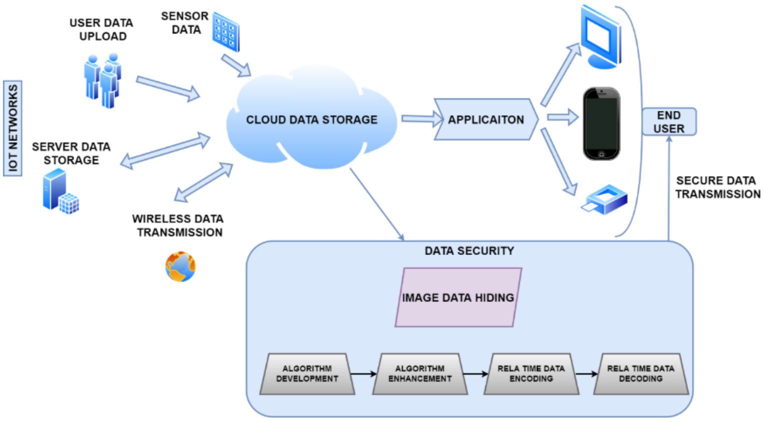


Figure 5.1. Architecture Diagram

5.1 List of phases

There are 5 phases:

- Image reading and writing
- Text to bits conversion
- Deep learning algorithm
- Metrics calculation
- Encoding and decoding of image using image data hiding

Image Reading and Writing

Reading and writing an image is required to read the image, do the inbuilt process and write back the processed data and save the image file. Computer vision is an open source library which has inbuilt various functions to execute. The images which will be performed with image data hiding algorithm will be read and write using computer vision libraries.

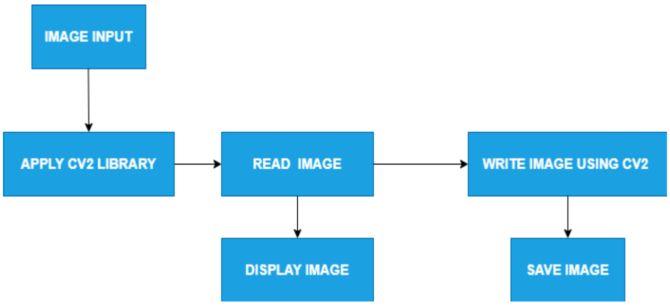


Figure 6.1.1 Image Reading and Writing diagram

Text to Bits Conversion

Converting a text to bits is a process by which any asci character can be changed to binary bits for further processing with image data hiding. Bits are the basic unit of any image which stores all the information of an image. In order to store the information in an image the data should first be converted to bits format so that the data can be merged in between the pixel bits. Hence, the data will be converted to bit form for further processing.

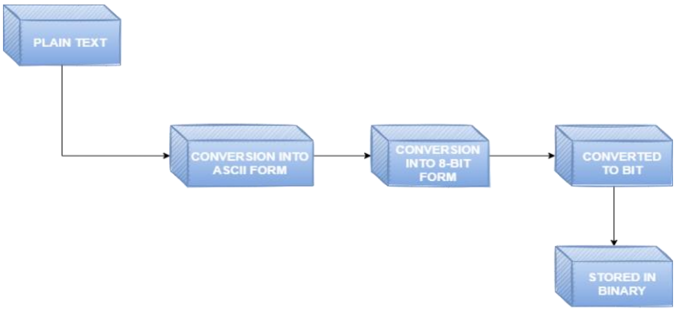


Figure 6.2.1 Text to Bits Conversion Diagram

Deep Learning Algorithm

Deep learning can be considered as a subset of machine learning. It is a field that is based on learning and improving on its own by examining computer algorithms. Deep learning uses artificial neural networks to perform sophisticated computations on large amounts of data. Whereas in image data hiding field still traditional techniques are made use. Hence, we are going to develop a deep learning algorithm which can be used in the secret data transfer with the emerging techniques. Since Deep learning is a trial and error technique to apply on machines, to enhance it changes can be done accordingly. Parameters such as bits per pixel, signal to noise ratio, etc. can be measured to see the variation and measure the performance of the model.

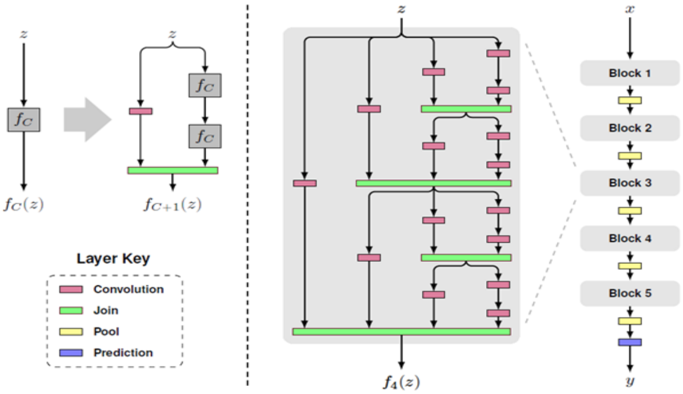


Figure 6.3.1 Deep Learning Algorithm diagram

Metrics Calculation

For comparing stego image with cover image results requires a measure of image quality, commonly used,

- Payload

- Loss
- Accuracy
- Peak Signal to Noise Ratio (PSNR)
- Structural Similarity Index (SSIM)

Depending on the above metric values the deep learning algorithm can be even fine-tuned to obtain higher values.

Encoding and decoding of image using image data hiding

After successful development of the deep learning algorithms and trained model generation encoding and decoding of data will be performed. Encoding is the process of putting a sequence of characters such as letters, numbers and other special characters into a specialized format for efficient transmission. Decoding is the process of converting an encoded format back into the original sequence of characters. The data will be given as an input to encode and decode the data in the image we select using the trained model.

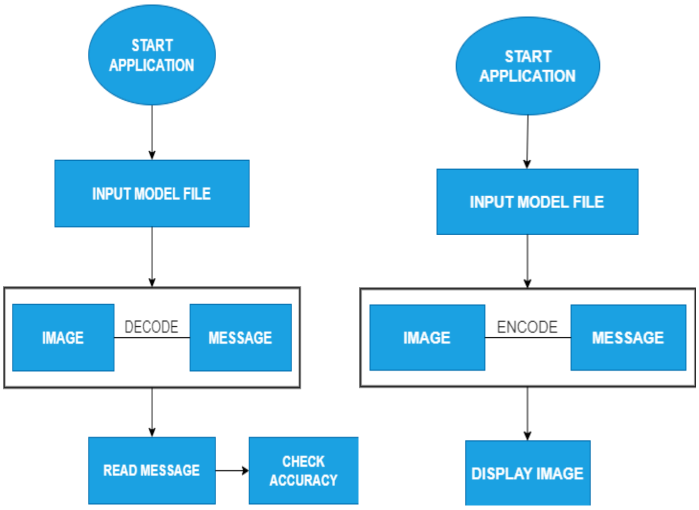
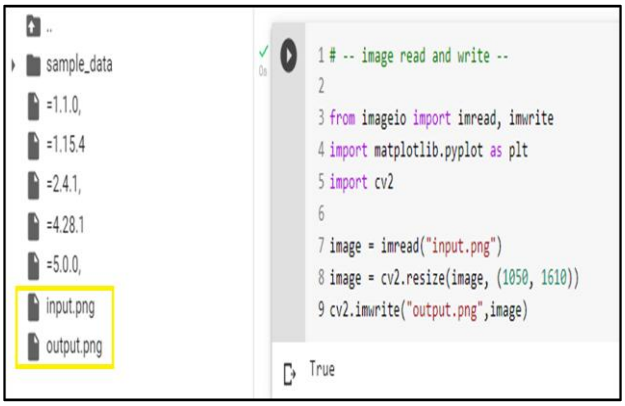


Figure 6.5.1 Encoding and Decoding diagram

Screen Shots

Image Read and Write

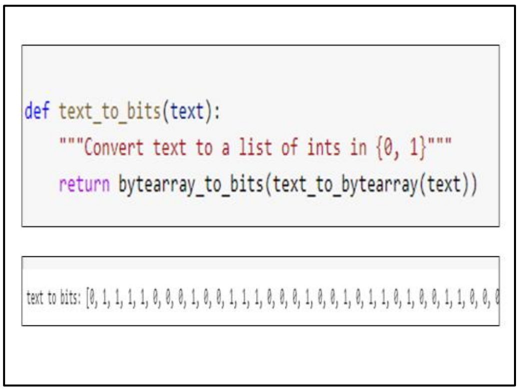


```
..
sample_data
=1.1.0,
=1.15.4
=2.4.1,
=4.28.1
=5.0.0,
input.png
output.png
```

```
# -- image read and write --

from imageio import imread, imwrite
import matplotlib.pyplot as plt
import cv2

image = imread("input.png")
image = cv2.resize(image, (1050, 1610))
cv2.imwrite("output.png",image)
```

```
True
```

Text to Bits Conversion

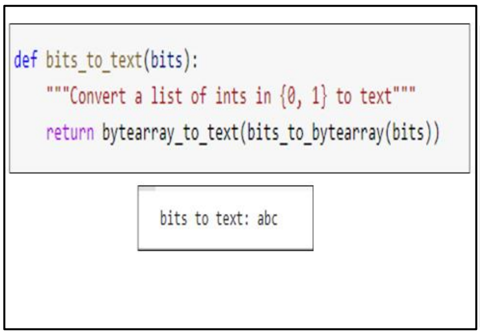


```
def text_to_bits(text):
    """Convert text to a list of ints in {0, 1}"""
    return bytearray_to_bits(text_to_bytearray(text))
```

```
text to bits: [0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0
```

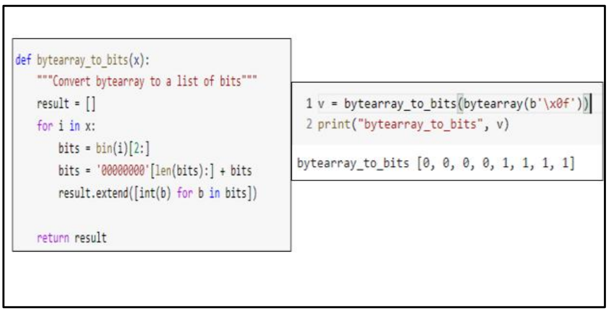
Bits to text conversion

Byte array to bits conversion



```
def bits_to_text(bits):
    """Convert a list of ints in {0, 1} to text"""
    return bytearray_to_text(bits_to_bytearray(bits))
```

```
bits to text: abc
```



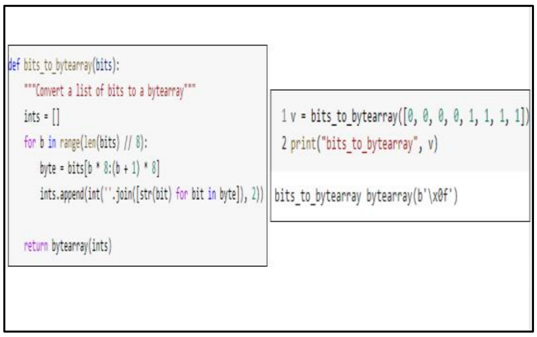
```
def bytearray_to_bits(x):
    """Convert bytearray to a list of bits"""
    result = []
    for i in x:
        bits = bin(i)[2:]
        bits = '00000000'[len(bits):] + bits
        result.extend([int(b) for b in bits])

    return result
```

```
v = bytearray_to_bits(bytearray(b'\x0f'))
print("bytearray_to_bits", v)
```

```
bytearray_to_bits [0, 0, 0, 0, 1, 1, 1, 1]
```

Bits to byte array conversion

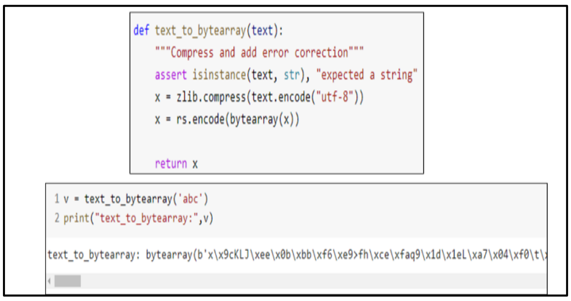
Text to byte array conversion



```
def bits_to_bytearray(bits):
    """Convert a list of bits to a bytearray"""
    ints = []
    for b in range(len(bits) // 8):
        byte = bits[b * 8:(b + 1) * 8]
        ints.append(int(''.join([str(bit) for bit in byte]), 2))

    return bytearray(ints)
```

```
v = bits_to_bytearray([0, 0, 0, 0, 1, 1, 1, 1])
print("bits_to_bytearray", v)
```

```
bits_to_bytearray bytearray(b'\x0f')
```



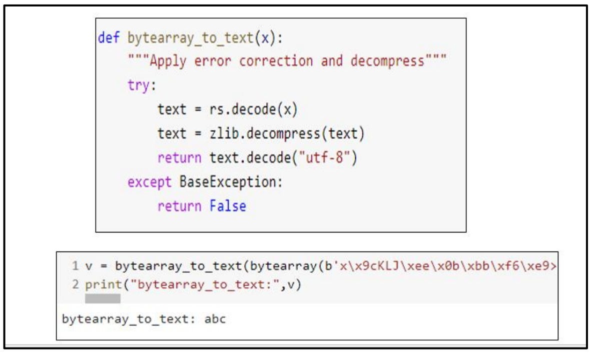
```
def text_to_bytearray(text):
    """Compress and add error correction"""
    assert isinstance(text, str), "expected a string"
    x = zlib.compress(text.encode("utf-8"))
    x = rs.encode(bytearray(x))

    return x
```

```
v = text_to_bytearray('abc')
print("text_to_bytearray:",v)
```

```
text_to_bytearray: bytearray(b'x\x9cKLJ\xee\x0b\xbb\xf6\xe9>fh\xce\xfaq9\x1d\x1eL\xa7\x04\xf0\t\
```

Byte to array to text conversion

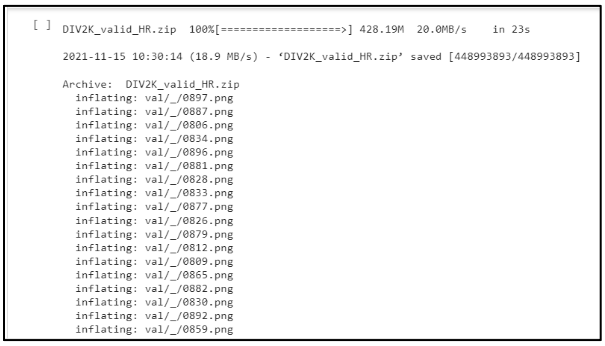
Downloading the dataset for training



```
def bytearray_to_text(x):
    """Apply error correction and decompress"""
    try:
        text = rs.decode(x)
        text = zlib.decompress(text)
        return text.decode("utf-8")
    except BaseException:
        return False
```

```
v = bytearray_to_text(bytearray(b'x\x9cKLJ\xee\x0b\xbb\xf6\xe9>
print("bytearray_to_text:",v)
```

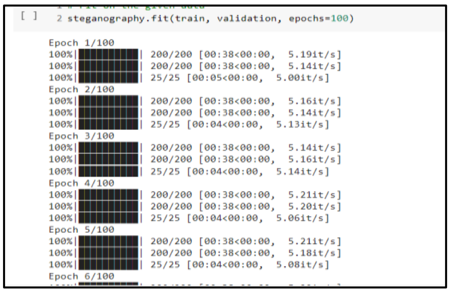
```
bytearray_to_text: abc
```



```
[ ]  DIV2K_valid_HR.zip  100%[===================>] 428.19M  20.0MB/s    in 23s

     2021-11-15 10:30:14 (18.9 MB/s) - ‘DIV2K_valid_HR.zip’ saved [448993893/448993893]

     Archive:  DIV2K_valid_HR.zip
       inflating: val/_/0897.png
       inflating: val/_/0887.png
       inflating: val/_/0806.png
       inflating: val/_/0834.png
       inflating: val/_/0896.png
       inflating: val/_/0881.png
       inflating: val/_/0828.png
       inflating: val/_/0833.png
       inflating: val/_/0877.png
       inflating: val/_/0826.png
       inflating: val/_/0879.png
       inflating: val/_/0812.png
       inflating: val/_/0809.png
       inflating: val/_/0865.png
       inflating: val/_/0882.png
       inflating: val/_/0830.png
       inflating: val/_/0892.png
       inflating: val/_/0859.png
```

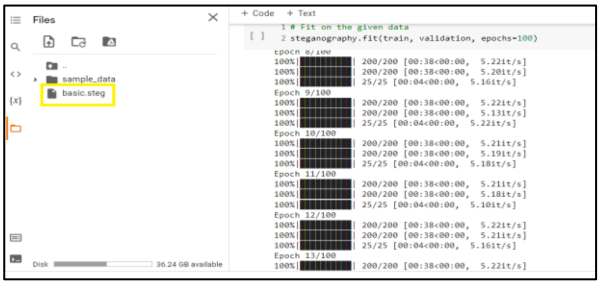
Deep learning algorithm training

Model file generation



```
# Fit on the given data
steganography.fit(train, validation, epochs=100)
```

```
Epoch 1/100
100%|██████████| 200/200 [00:38<00:00,  5.19it/s]
100%|██████████| 200/200 [00:38<00:00,  5.14it/s]
100%|██████████| 25/25 [00:05<00:00,  5.00it/s]
Epoch 2/100
100%|██████████| 200/200 [00:38<00:00,  5.16it/s]
100%|██████████| 200/200 [00:38<00:00,  5.14it/s]
100%|██████████| 25/25 [00:04<00:00,  5.13it/s]
Epoch 3/100
100%|██████████| 200/200 [00:38<00:00,  5.14it/s]
100%|██████████| 200/200 [00:38<00:00,  5.16it/s]
100%|██████████| 25/25 [00:04<00:00,  5.14it/s]
Epoch 4/100
100%|██████████| 200/200 [00:38<00:00,  5.21it/s]
100%|██████████| 200/200 [00:38<00:00,  5.20it/s]
100%|██████████| 25/25 [00:04<00:00,  5.06it/s]
Epoch 5/100
100%|██████████| 200/200 [00:38<00:00,  5.21it/s]
100%|██████████| 200/200 [00:38<00:00,  5.18it/s]
100%|██████████| 25/25 [00:04<00:00,  5.08it/s]
Epoch 6/100
```

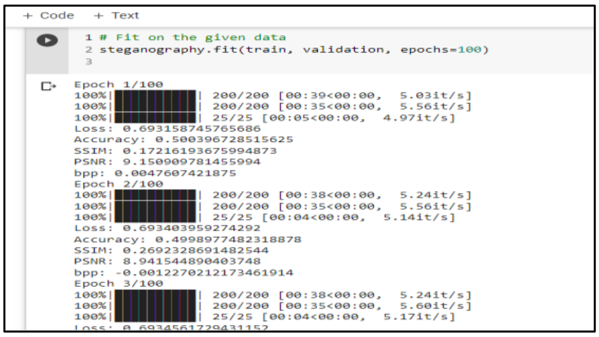


```
# Fit on the given data
steganography.fit(train, validation, epochs=100)
```

```
Epoch 8/100
100%|██████████| 200/200 [00:38<00:00,  5.22it/s]
100%|██████████| 200/200 [00:38<00:00,  5.20it/s]
100%|██████████| 25/25 [00:04<00:00,  5.16it/s]
Epoch 9/100
100%|██████████| 200/200 [00:38<00:00,  5.22it/s]
100%|██████████| 200/200 [00:38<00:00,  5.13it/s]
100%|██████████| 25/25 [00:04<00:00,  5.22it/s]
Epoch 10/100
100%|██████████| 200/200 [00:38<00:00,  5.21it/s]
100%|██████████| 200/200 [00:38<00:00,  5.19it/s]
100%|██████████| 25/25 [00:04<00:00,  5.18it/s]
Epoch 11/100
100%|██████████| 200/200 [00:38<00:00,  5.21it/s]
100%|██████████| 200/200 [00:38<00:00,  5.18it/s]
100%|██████████| 25/25 [00:04<00:00,  5.10it/s]
Epoch 12/100
100%|██████████| 200/200 [00:38<00:00,  5.22it/s]
100%|██████████| 200/200 [00:38<00:00,  5.21it/s]
100%|██████████| 25/25 [00:04<00:00,  5.16it/s]
Epoch 13/100
100%|██████████| 200/200 [00:38<00:00,  5.22it/s]
```

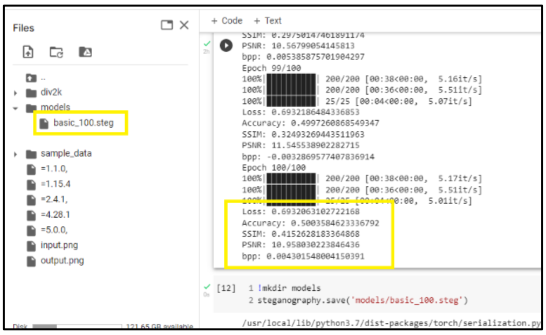
Metrics Calculation

Model file generation with metrics



```
# Fit on the given data
steganography.fit(train, validation, epochs=100)

```

```
Epoch 1/100
100%|██████████| 200/200 [00:39<00:00,  5.03it/s]
100%|██████████| 200/200 [00:35<00:00,  5.56it/s]
100%|██████████| 25/25 [00:05<00:00,  4.97it/s]
Loss: 0.693158745765686
Accuracy: 0.500396728515625
SSIM: 0.17216193675994873
PSNR: 9.150909781455994
bpp: 0.0047607421875
Epoch 2/100
100%|██████████| 200/200 [00:38<00:00,  5.24it/s]
100%|██████████| 200/200 [00:35<00:00,  5.56it/s]
100%|██████████| 25/25 [00:04<00:00,  5.14it/s]
Loss: 0.693403959274292
Accuracy: 0.4998977482318878
SSIM: 0.2692328691482544
PSNR: 8.941544890403748
bpp: -0.0012270212173461914
Epoch 3/100
100%|██████████| 200/200 [00:38<00:00,  5.24it/s]
100%|██████████| 200/200 [00:35<00:00,  5.60it/s]
100%|██████████| 25/25 [00:04<00:00,  5.17it/s]
Loss: 0.6934561729431152
```

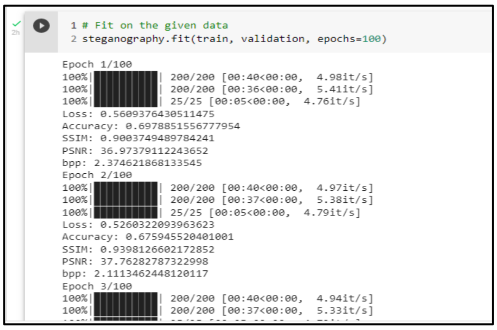


```
SSIM: 0.2975014746189174
PSNR: 10.56799054145813
bpp: 0.005385875701904297
Epoch 99/100
100%|██████████| 200/200 [00:38<00:00,  5.16it/s]
100%|██████████| 200/200 [00:36<00:00,  5.51it/s]
100%|██████████| 25/25 [00:04<00:00,  5.07it/s]
Loss: 0.6932186484336853
Accuracy: 0.4997260868549347
SSIM: 0.32493269443511963
PSNR: 11.545538902282715
bpp: -0.0032869577407836914
Epoch 100/100
100%|██████████| 200/200 [00:38<00:00,  5.17it/s]
100%|██████████| 200/200 [00:36<00:00,  5.51it/s]
100%|██████████| 25/25 [00:04<00:00,  5.01it/s]
Loss: 0.6932063102722168
Accuracy: 0.5003584623336792
SSIM: 0.4152628183364868
PSNR: 10.958030223846436
bpp: 0.004301548004150391
```

```
[12]  !mkdir models
      steganography.save('models/basic_100.steg')

/usr/local/lib/python3.7/dist-packages/torch/serialization.py
```

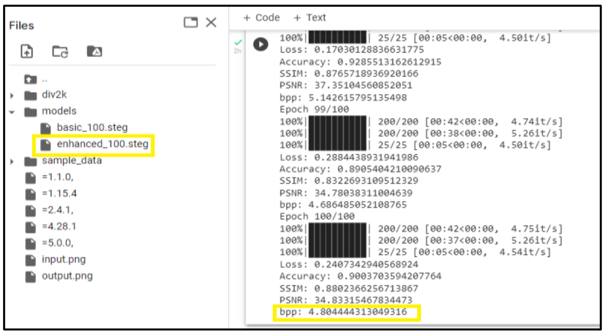
Enhanced deep learning algorithm training

Model File Generation After Enhanced Algorithm Training



```
# Fit on the given data
steganography.fit(train, validation, epochs=100)
```

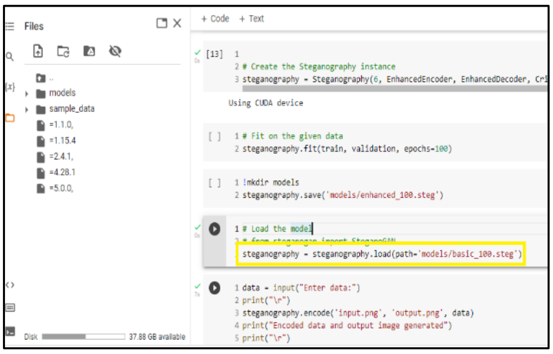
```
Epoch 1/100
100%|██████████| 200/200 [00:40<00:00,  4.98it/s]
100%|██████████| 200/200 [00:36<00:00,  5.41it/s]
100%|██████████| 25/25 [00:05<00:00,  4.76it/s]
Loss: 0.5609376430511475
Accuracy: 0.6978851556777954
SSIM: 0.9003749489784241
PSNR: 36.97379112243652
bpp: 2.374621868133545
Epoch 2/100
100%|██████████| 200/200 [00:40<00:00,  4.97it/s]
100%|██████████| 200/200 [00:37<00:00,  5.38it/s]
100%|██████████| 25/25 [00:05<00:00,  4.79it/s]
Loss: 0.5260322093963623
Accuracy: 0.675945520401001
SSIM: 0.9398126602172852
PSNR: 37.76282787322998
bpp: 2.1113462448120117
Epoch 3/100
100%|██████████| 200/200 [00:40<00:00,  4.94it/s]
100%|██████████| 200/200 [00:37<00:00,  5.33it/s]
```



```
100%|██████████| 25/25 [00:05<00:00,  4.50it/s]
Loss: 0.17030128836631775
Accuracy: 0.9285513162612915
SSIM: 0.8765718936920166
PSNR: 37.35104560852051
bpp: 5.142615795135498
Epoch 99/100
100%|██████████| 200/200 [00:42<00:00,  4.74it/s]
100%|██████████| 200/200 [00:38<00:00,  5.26it/s]
100%|██████████| 25/25 [00:05<00:00,  4.50it/s]
Loss: 0.2884438931941986
Accuracy: 0.8905404210090637
SSIM: 0.8322693109512329
PSNR: 34.78038311004639
bpp: 4.686485052108765
Epoch 100/100
100%|██████████| 200/200 [00:42<00:00,  4.75it/s]
100%|██████████| 200/200 [00:37<00:00,  5.26it/s]
100%|██████████| 25/25 [00:05<00:00,  4.54it/s]
Loss: 0.2407342940568924
Accuracy: 0.9003703594207764
SSIM: 0.8802366256713867
PSNR: 34.83315467834473
bpp: 4.804444313049316
```

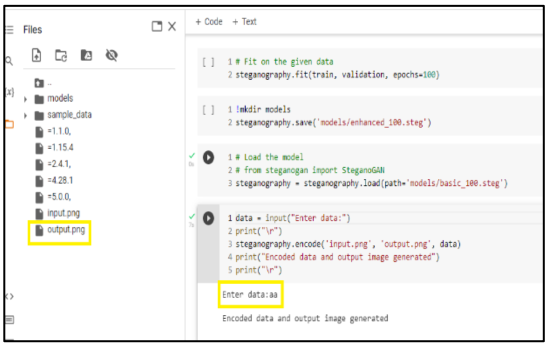
Loading basic model file for encoding

Output image generation



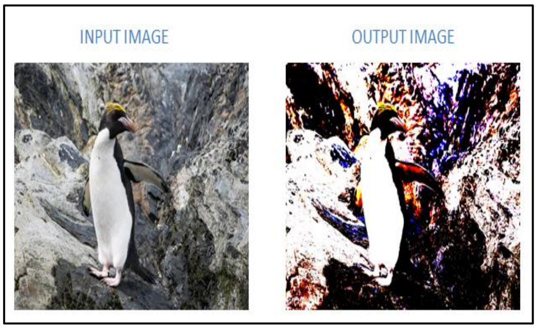
```
[13]
# Create the Steganography instance
steganography = Steganography(6, EnhancedEncoder, EnhancedDecoder, Cri

Using CUDA device

[ ]
# Fit on the given data
steganography.fit(train, validation, epochs=100)

[ ]
!mkdir models
steganography.save('models/enhanced_100.steg')

# Load the model
# from steganogan import SteganoGAN
steganography = steganography.load(path='models/basic_100.steg')

data = input("Enter data:")
print("\r")
steganography.encode('input.png', 'output.png', data)
print("Encoded data and output image generated")
print("\r")
```

Image generation in input and output

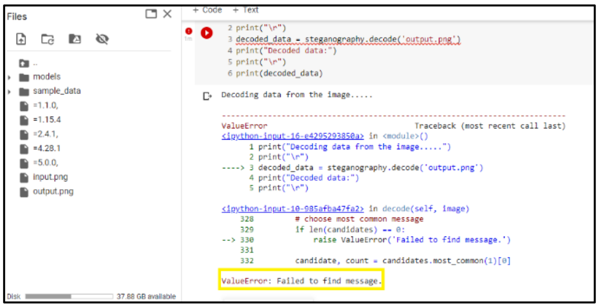


```
[ ]
# Fit on the given data
steganography.fit(train, validation, epochs=100)

[ ]
!mkdir models
steganography.save('models/enhanced_100.steg')

# Load the model
# from steganogan import SteganoGAN
steganography = steganography.load(path='models/basic_100.steg')

data = input("Enter data:")
print("\r")
steganography.encode('input.png', 'output.png', data)
print("Encoded data and output image generated")
print("\r")

Enter data:aa

Encoded data and output image generated
```

Decoding data using basic model

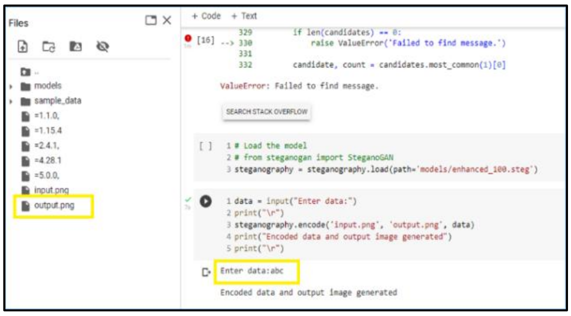


Loading enhanced model file for encoding



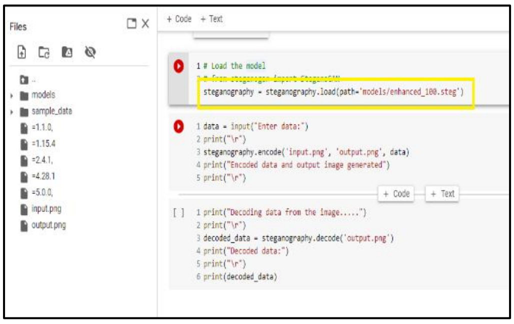
```
print("\r")
decoded_data = steganography.decode('output.png')
print("Decoded data:")
print("\r")
print(decoded_data)

Decoding data from the image.....

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-16-e4295293850a> in <module>()
      1 print("Decoding data from the image.....")
      2 print("\r")
----> 3 decoded_data = steganography.decode('output.png')
      4 print("Decoded data:")
      5 print("\r")

<ipython-input-10-985afba47fa2> in decode(self, image)
    328         # choose most common message
    329         if len(candidates) == 0:
--> 330             raise ValueError('Failed to find message.')
    331
    332         candidate, count = candidates.most_common(1)[0]

ValueError: Failed to find message.
```

Encoding data using enhanced model



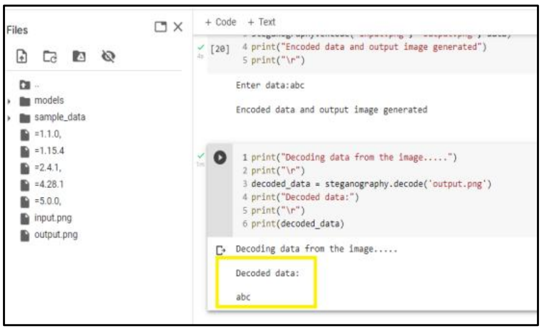
```
[16]
    329         if len(candidates) == 0:
--> 330             raise ValueError('Failed to find message.')
    331
    332         candidate, count = candidates.most_common(1)[0]

ValueError: Failed to find message.

[ ]
# Load the model
# from steganogan import SteganoGAN
steganography = steganography.load(path='models/enhanced_100.steg')

data = input("Enter data:")
print("\r")
steganography.encode('input.png', 'output.png', data)
print("Encoded data and output image generated")
print("\r")

Enter data:abc
Encoded data and output image generated
```

Image generation in input and output

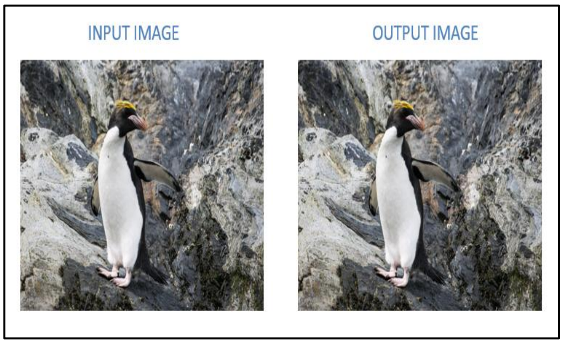


```
# Load the model
# from steganogan import SteganoGAN
steganography = steganography.load(path='models/enhanced_100.steg')

data = input("Enter data:")
print("\r")
steganography.encode('input.png', 'output.png', data)
print("Encoded data and output image generated")
print("\r")

[ ]
print("Decoding data from the image.....")
print("\r")
decoded_data = steganography.decode('output.png')
print("Decoded data:")
print("\r")
print(decoded_data)
```

Decoding data using enhanced model



```
[20]
print("Encoded data and output image generated")
print("\r")

Enter data:abc

Encoded data and output image generated

print("Decoding data from the image.....")
print("\r")
decoded_data = steganography.decode('output.png')
print("Decoded data:")
print("\r")
print(decoded_data)

Decoding data from the image.....

Decoded data:

abc
```



6. CONCLUSIONS

This project proposed that encrypt the IoT networks data by hide the encrypted message inside an image file using image data hiding technique as well increases the number of bits that can be saved within a pixel of an image using the currently prevailing deep learning approach. Using which we have developed an algorithm such as convolutional neural networks effectively protect and secure the data. Encryption and decryption of the data has been performed using the newly developed deep learning algorithm.

7. REFERENCES

[1] Kuri, J.L. (2020). Securing Data in Internet of Things (IoT) using Cryptography and Steganography Techniques. International Journal for Research in Applied Science and Engineering Technology, 8(7), pp.1933--1939. doi:10.22214/ijraset.2020.30485.

[2] Boroumand, M., Chen, M. and Fridrich, J. (2019). Deep Residual Network for Steganalysis of Digital Images. IEEE Transactions on Information Forensics and Security, 14(5), pp.1181–1193. doi:10.1109/tifs.2018.2871749.

[3] Hassaballah, M., Hameed, M.A., Awad, A.I. and Muhammad, K. (2021). A Novel Image Steganography Method for Industrial Internet of Things Security. IEEE Transactions on Industrial Informatics, pp.1–1. doi:10.1109/tii.2021.3053595.

[4] Karati, A., Islam, S.H. and Karuppiah, M. (2018). Provably Secure and Lightweight Certificateless Signature Scheme for IIoT Environments. IEEE Transactions on Industrial Informatics, [online] 14(8), pp.3701–3711. doi:10.1109/TII.2018.2794991.

[5] Li, Y., Zhang, F. and Liu, X. (2020). Secure Data Delivery with Identity-based Linearly Homomorphic Network Coding Signature Scheme in IoT. IEEE Transactions on Services Computing, pp.1–1. doi:10.1109/tsc.2020.3039976.

[6] Liao, X., Yin, J., Chen, M. and Qin, Z. (2020). Adaptive Payload Distribution in Multiple Images Steganography Based on Image Texture Features. IEEE Transactions on Dependable and Secure Computing, [online] pp.1–1. doi:10.1109/TDSC.2020.3004708.

[7] Lin, Z., Huang, Y. and Wang, J. (2018). RNN-SM: Fast Steganalysis of VoIP Streams Using Recurrent Neural Network. IEEE Transactions on Information Forensics and Security, 13(7), pp.1854–1868. doi:10.1109/tifs.2018.2806741.

[8] Lu, W., Chen, J., Zhang, J., Huang, J., Weng, J. and Zhou, Y. (2022). Secure Halftone Image Steganography Based on Feature Space and Layer Embedding. IEEE Transactions on Cybernetics, 52(6), pp.5001–5014. doi:10.1109/tcyb.2020.3026047.

[9] Singh, B., Sur, A. and Mitra, P. (2021). Steganalysis of Digital Images Using Deep Fractal Network. IEEE Transactions on Computational Social Systems, 8(3), pp.599–606. doi:10.1109/tcss.2021.3052520.

[10] Su, W., Ni, J., Hu, X. and Fridrich, J. (2021). Image Steganography With Symmetric Embedding Using Gaussian Markov Random Field Model. IEEE Transactions on Circuits and Systems for Video Technology, 31(3), pp.1001–1015. doi:10.1109/tcsvt.2020.3001122.

[11] Tang, W., Li, B., Tan, S., Barni, M. and Huang, J. (2019). CNN-Based Adversarial Embedding for Image Steganography. IEEE Transactions on Information Forensics and Security, 14(8), pp.2074–2087. doi:10.1109/tifs.2019.2891237.

[12] Tedeschi, P., Sciancalepore, S., Eliyan, A. and Di Pietro, R. (2020). LiKe: Lightweight Certificateless Key Agreement for Secure IoT Communications. IEEE Internet of Things Journal, 7(1), pp.621–638. doi:10.1109/jiot.2019.2953549.

[13] Wang, W., Xu, P., Liu, D., Yang, L.T. and Yan, Z. (2020). Lightweighted Secure Searching Over Public-Key Ciphertexts for Edge-Cloud-Assisted Industrial IoT Devices. IEEE Transactions on Industrial Informatics, 16(6), pp.4221–4230. doi:10.1109/tii.2019.2950295.

[14] Wu, S., Zhong, S. and Liu, Y. (2020). A Novel Convolutional Neural Network for Image Steganalysis with Shared Normalization. IEEE Transactions on Multimedia, 22(1), pp.256–270. doi:10.1109/tmm.2019.2920605.

[15] Zhang, R., Zhu, F., Liu, J. and Liu, G. (2020). Depth-Wise Separable Convolutions and Multi-Level Pooling for an Efficient Spatial CNN-Based Steganalysis. IEEE Transactions on Information Forensics and Security, 15, pp.1138–1150. doi:10.1109/tifs.2019.2936913.