StegIm: Image in Image Steganography

Chapter · January 2023

DOI: 10.1007/978-3-031-22792-9_2

Jovana Dobreva
Ss. Cyril and Methodius University in Skopje
11 PUBLICATIONS **7** CITATIONS
SEE PROFILE

Stefan Andonov
Ss. Cyril and Methodius University in Skopje
5 PUBLICATIONS **5** CITATIONS
SEE PROFILE

Hristina Mihajloska
Ss. Cyril and Methodius University in Skopje
18 PUBLICATIONS **60** CITATIONS
SEE PROFILE

Aleksandra Popovska-Mitrovikj
Ss. Cyril and Methodius University in Skopje
35 PUBLICATIONS **141** CITATIONS
SEE PROFILE

StegIm: Image in Image Steganography

Ivo Tasevski[1[0000-0003-2068-1635]], Jovana Dobreva[1[0000-0001-7382-2062]], Stefan Andonov[1[0000-0002-2025-9314]], Hristina Mihajloska[1[0000-0002-2537-8697]], Aleksandra Popovska-Mitrovikj[1[0000-0002-1578-6275]] and Vesna Dimitrova[1[0000-0003-4393-5589]]

[1] Faculty of Computer Science and Engineering, University Ss. Cyril and Methodius, Skopje, Republic of North Macedonia
ivo.tasevski@students.finki.ukim.mk

Abstract. Today, data security is a significant problem in data communication. Sometimes we do not realize how susceptible we are to cyberattacks and the theft of critical information, including everything, from social network passwords to complete identities. Any system, no matter how sophisticated it is, is vulnerable and susceptible to attack. Steganography is a technique for hiding secret information by adding it in a non-secret file. Also, it can be used in combination with encryption to further conceal or safeguard data. Therefore, in this paper, we present the StegIm model for hiding, retrieving, and detecting images using variants of LSB Encoding Steganography. We present the implementation of the model with different algorithms for hiding the image in another image (shape steganography algorithm and standard delimiter-based steganography algorithm) and give comparison analysis of used algorithms. To further strengthen the security, we implemented image encryption for this type of image steganography and analyzed the improvements, benefits, advantages, and disadvantages of this model in each phase of hiding/retrieving. Also, StegIm can detect hidden data in given images with the help of machine learning methods. This model will strive to reach the highest level of confidentiality, authenticity, and integrity of the user's confidential and sensitive data.

Keywords: StegIm, Crypt-Steganography, Machine Learning, Stego-image Encryption.

1 Introduction

Data security is regarded as a severe issue when data communication is carried out via the Internet, and everyone expects their data to be secure during the communication process. The process of concealing a message, image, audio, or video by embedding it in another message, image, audio, or video is known as steganography.

Steganography is a technique for hiding secret information by adding it in a regular, non-secret file or communication. The hidden information is subsequently extracted at the intended location. Steganography can be used to hide almost any type of digital content, including text, image, audio, or video content, and the data can be hidden in almost any other type of digital content. When we want to send some secret information, steganography can be seen as more discreet than cryptography, but in this case, the hidden message is much easier to find out.

Steganography can be combined with encryption to conceal further or safeguard data. The content we want to hide with steganography, known as hidden text, is often encrypted before being integrated into the carrier text file or data stream. This approach is known as crypt steganography. If the hidden text is not encrypted, it is commonly processed in some way to make it more difficult to detect the secret content.

There are numerous techniques to insert the hidden message into regular data files. One method is to conceal data in bits that correspond to consecutive rows of the same color pixels in a picture file. The output will be an image file that looks just like the original image but has "noise" patterns of regular, unencrypted data.

While there are many various applications for steganography, such as hiding sensitive data within certain file formats, one of the most widely used methods is to incorporate a text file within an image file. This is performed by putting the message with less significant bits in the data file so that anyone viewing the picture file should not be able to distinguish between the original image file and the stego-image. There are a lot of ways to hide information inside an image and some common approaches include Least Significant Bit Insertion, Masking and Filtering, Redundant Pattern Encoding, Encrypt and Scatter, Coding, and Cosine Transformation [1]. The pixels of an image consist of values R, G, and B for Red, Green, and Blue, respectively. Each value is in the 0-255 range and image’s least-significant bits are the final bits of each byte The pixel's smallest value for each of the R, G, and B values are stored in the least-significant bits. In the Least Significant Bit (LSB) approach, these least significant bits of the image are changed [2]. Therefore, it is difficult to distinguish between the original and the stego-image.

On the other hand, steganalysis is the process of identifying the presence of hidden messages using steganography. This process uses methods for extracting hidden communication from stego-objects [3,4]. For the detection of hidden data, in paper [5], authors present methods for steganography and steganalysis methodologies with machine learning frameworks. They show how machine learning frameworks can be used to uncover secret data concealed in images using steganography algorithms. In order to detect real-world stego-images, in [6] authors provide a deep learning system that employs Convolutional Neural Networks (CNN). Two pre-trained models with 73.33% and 79.43% accuracy are presented.

In paper [7] we propose a StegYou model for hiding, retrieving, and detecting digital data in images. The implementation of hiding and retrieving data is based on a custom-made crypt-steganography algorithm for one or more images. Also, StegYou can detect hidden data in given images using the Machine Learning approach and achieve an accuracy of 87% for Single-Image Steganography and accuracy of 98% for Single-Image Crypt-Steganography.

In this paper, we propose a modification of the StegYou model, called StegIm where we consider hiding and retrieving a secret image in a cover image. While steganography has many legitimate applications, malware developers have been found to use it to conceal the transmission of malicious code. Therefore, in our StegIm model, we incorporate a detection of hidden image in image using the machine learning methods.

The rest of the paper is organized in the following way. In Section 2 is given the implementation of StegIm for hiding and retrieving digital images using different encoding methods. In the same section is presented the implementation of the model with different algorithms for hiding the image in another image (image shape steganography algorithm and standard delimiter-based steganography algorithm). Also, we present comparison analysis of used algorithms and to further strengthen the security, we have also implemented image encryption for this type of image steganography. The improvements, benefits, advantages, and disadvantages of our model in each phase of hiding/retrieving are analyzed. The detection of hidden data in images for the StegIm model, using the machine learning method, is described in Section 3. Finally, in Section 4, we give some conclusions and further work on the proposed StegIm model.

2 Implementation of StegIm for hiding and retrieving digital images

In this section, we consider the implementation of StegIm for hiding and retrieving digital images. Now we will dive into the concept of hiding an image inside another image. Still, before we do, we should again mention some noteworthy things relevant to this concept of image steganography. As it is already known, a digital image can be represented as a finite set of digital values called pixels. Pixels are the smallest units of the digital image that can be displayed, and they contain values representing digital numeric RGB color data. Therefore, we can see an image as a two-dimensional array - a matrix of pixels that includes a fixed number of rows and columns.. Each pixel has three decimal R, G, and B values, and each RGB value can be represented in binary format with 8 bits. In this section, we have defined three encoding methods, each with its own unique characteristics, benefits, and downsides. Since we are working with more extensive and denser data (images), it is crucial to have enough encoding capacity. This is the primary principle of the L4SB and L2SB encoding methods. As the name indicates, the L4SB encoding method uses the last four rightmost bits or last four LSBs (least-significant bits) to encode a secret image, while the latter uses the last two LSBs. This increases the encoding capacity and brings more data to be encoded in the carrier images. The third one is the standard LSB encoding method used in steganography, where data is hiding in images and use the last LSB of the carrier image for secret message encoding.

2.1 The LSB encoding method

The LSB encoding method is used to encode secret image into a carrier image, using the last rightmost bit of the carrier image’s pixel. It allows us to encode a secret image which is eight times smaller than the carrier image. A carrier image of size N bytes can hold a secret image of at most N/8 bytes since we are encoding one R, G, or B value of the secret image inside eight R, G, and B values of the carrier image. This encoding method does not compromise the quality of the carrier stego-images. Also, the image quality loss is not visible to the bare eye, meaning that stego-images are very hard to detect, making this encoding method the safest and best in terms of quality retention. The encoding capacity is its biggest problem, but it can be surpassed by simply using bigger carrier images or smaller secret images.

2.2 The L2SB encoding method

The L2SB encoding method allows us to encode a secret image that is a fourth of the size of the carrier image in bytes, at most. A carrier image of N bytes can hold a secret image of nearly N/4 bytes, since we are encoding one R, G or B value of the secret image inside four R, G, and B values of the carrier image. This encoding method delivers a twice bigger encoding capacity than the previous one. It doesn't compromise the quality of the carrier stego-image, but some of the changes still can be visible to the bare eye. The L2SB encoding method breaks the ice between the L4SB and LSB encoding methods and strikes a compromise between the two when viewing from image quality and encoding capacity point of view.

2.3 The L4SB encoding method

Since we are encoding data in the last four bits of the values of the pixels in the carrier image, the L4SB encoding method allows us to encode an image that is half the size of the carrier image in bytes at most. Suppose an image has N values (every pixel contains three values, and each value is one byte). In that case, an image of nearly N/2 bytes can be encoded in such a carrier image since we are technically replacing half of the bits of the carrier image with the bits of the secret image. If nothing else needed to be encoded, then the encoding capacity would be exactly N/2 bytes.

The L4SB encoding method still follows the LSB encoding principles, with the only difference being that it is a modified version of it. Instead of using only the last LSB, it uses the last four LSBs, with the sole purpose of increasing the encoding capacity to fulfill the needs of encoding more dense and larger data formats, like JPG or PNG images.

However, it is crucial to mention that this encoding method drastically compromises the quality of the carrier stego-image, making it easily detectable and thus making the anonymity and privacy of secret images obsolete. It is the most insecure of all three encoding methods and simultaneously makes the whole steganography concept pointless.

The L4SB encoding method is very risky, and we do not recommend using it for textual or other smaller data formats.

To summarize, in Table 1, we give all advantages and disadvantages of previously presented encoding methods.

Table 1. Comparison analysis of LSB, L2SB and L4SB encoding methods

Encoding method	Advantages	Disadvantages
LSB	Safest encoding method - carrier stego-image quality is slightly compromised, but not visible to the bare eye	Smallest encoding capacity - twice smaller than the one when using L2SB, and four times smaller than the one when using L4SB
L2SB	Encoding capacity is still big - twice bigger than the one when using the LSB encoding method Compromise between quality retention and encoding capacity	Carrier stego-image quality is slightly compromised, but visible to the bare eye Encoding capacity is twice smaller than the one when using L4SB
L4SB	Biggest encoding capacity - two times bigger than the one when using L2SB, four times bigger than the one when using LSB	Most insecure encoding method - carrier stego-image quality is drastically compromised Stego-image is easily detectable

2.4 Used algorithms for hiding/retrieving images

In this subsection, we compare two algorithms for hiding an image inside another image - the image shape steganography algorithm and the standard delimiter-based steganography algorithm, which both carry unique principles and features. But, before we begin the analysis, we need to clarify a few things and explain some key terms used in the following.

A carrier image is an image in which the secret image is encoded/hidden, i.e., the image that carries our secret JPG or PNG data. A carrier image is the same as a stego-image; to be more precise, the carrier image is the stego-image after the image has been encoded. Image shape represents the dimensions of an image, i.e., the height and width of an image. A shape flag is an indicator for the encoded image shape, done in the first two, four, or eight pixels of the carrier image, depending on the encoding method used. The shape flag is used to carry the dimensions of the secret image, just in order to make decoding process easily.

Essential functions used in the algorithms are the functions encode (for hiding data) and decode (for retrieving data). These functions are used as entry points to the actual functions that represent hiding in and retrieving the image from an image. Their variation depends on which algorithm is used for producing the stego-image.In the following we will give a brief explanation of the used algorithms.

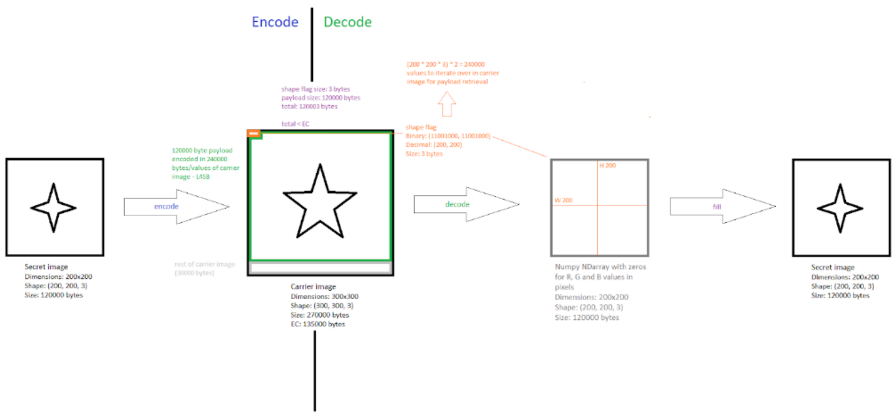

Fig. 1. The visual representation of the image shape steganography algorithm

2.4.1. Image shape algorithm

The core concept of this method lies purely in the image shape, i.e., the image dimensions of the secret image that are hidden in the carrier image. Knowing the dimensions of the secret image during the decoding process helps us find the exact shape of the secret image. It also opens up the possibility of extracting only the relevant information from the carrier image by calculating the exact number of values used for encoding. This is the core concept of our image shape algorithm.

This algorithm works by encoding the image shape in the carrier image's first two, four, or eight pixels, depending on the encoding technique. According to the fact that the encoding capacity of a pixel of a carrier image is 12 bits at most when using the L4SB encoding method, 6 when using the L2SB encoding method, and 3 when using the LSB encoding method, we decided to use 12 bits for both the height and width of an image to be encoded, resulting in maximum width/height of a secret image to be 4095. An image with dimensions 4095x4095 is the maximum allowed size of a secret image. Following the shape flag is the actual payload, i.e. the secret data that we encode in the carrier image. During the encoding process, the first step is determining the secret image dimensions. After that, the secret image is converted into binary format, i.e., the values of the pixels of the secret image for width and height are converted to binary and then concatenated together. The shape flag is encoded in the first two, four, or eight pixels of the carrier image, after which the binary bitstring of concatenated, flattened binary values are encoded in the carrier image. During the decoding process, the shape flag is extracted from the carrier image first, and the secret image size is calculated. By doing this, we now know the exact width and height of our secret image and the number of relevant R, G, and B values of the carrier image that have data encoded in them. Afterward, we iterate over the exact number of pixels and values of the carrier image and retrieve our payload, i.e., the bits of our secret image. The next step is grouping the extracted bits into groups of 8, representing bytes, and converting them into decimal format. Using the extracted width and height of the secret encoded image, we can reconstruct the exact secret image passed in the encoding function. In Fig. 1 we try to consider the simplified visual representation of the image shape steganography algorithm with the L4SB encoding technique.

To summarize, in Table 2, all advantages and disadvantages of this algorithm are given.

Table 2. Advantages and disadvantages for the image shape steganography algorithm.

Advantages	Disadvantages
• Slightly bigger encoding capacity and simpler workflow • Only data from relevant R, G and B values from carrier image is extracted - relevant value calculation using shape flag • No irrelevant, unimportant data extracted - speed and efficiency • Faster when it comes down to small secret images and large carrier images	• Carrier stego-image quality might be drastically compromised • Secret image size must be up to eight times smaller than carrier image, depends on encoding technique used • Non-stego images can’t be detected

2.4.2. Delimiter-based algorithm

The delimiter-based algorithm works the same way as in the case of hiding secret data explained in [7]. To summarize quickly, a delimiter is a fixed-length string containing random punctuation marks. Its purpose is to act as an indicator for accurate data retrieval during the decoding process. It is first converted into binary format and then appended onto the end of the binary bitstring that represents secret data. It is the part that is last encoded in the carrier image after the shape flag and payload have been encoded. It is similar to the image shape algorithm but with some key differences.

During the encoding process, identical operations are performed like the ones from the previous algorithm, except that additional 10 bytes are added to the payload, corresponding to the delimiter. Firstly, the shape flag is encoded in the first two, four, or eight pixels of the carrier image (depending on the encoding technique), followed by the payload (the secret bits of the secret image and the delimiter). Since the delimiter is located at the end of the payload, it is encoded last.

During the decoding process, the shape flag is first extracted from the carrier stego-image. After that, the process is identical to the one in the previous phase, except for the part where every extracted byte, apart from being converted to decimal, is converted to its corresponding ASCII characters. This is done for delimiter detection, and once the last ten extracted characters match the delimiter, the delimiter is removed from the final extracted values, leading to valid image retrieval. A simplified visual representation of the delimiter-based workflow with the L4SB encoding method is given in Fig. 2.

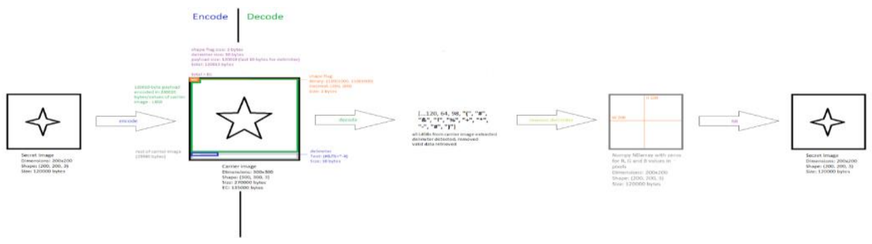

Fig. 2. The visual representation of the delimiter-based steganography algorithm.

To summarize, in Table 3, all advantages and disadvantages of this algorithm are given.

2.4.3. General in-depth analysis

According to Table 2 and Table 3 from the previous algorithms, we can conclude that using one of the algorithms has its own benefits.

Using an image shape algorithm makes the decoding process easy, but it lacks in the way that any image can be passed to the decoding function, and there is not a way to determine if that image is truly a stego-image or not. The decoding function will throw an error in most cases like these because of inconsistent values, but not for all. Also, an image shape algorithm is carrying the image size in the carrier image, so it is the best practice not to decode the whole stego-image and improves the implementation performances.

However, using and encoding a delimiter in the carrier image, stego-images can be easily detected when passed to the decoding function. In the previous case, the user could send any image and whether that image is a stego-image or not, there wasn't a way to determine that. Here, passed images can instantly be classified as stego or non-stego images simply by checking the presence of the delimiter in the image. This eliminates those very small number of cases where the user could pass a non-stego image and successfully decode data from it, even if it is not a stego-image. The delimiter-based algorithm will eliminate the presence of non-stego images during the decoding process, which makes it in a way a bit more reliable than the other algorithm. Nevertheless, that doesn't make it better than the image shape algorithm. This algorithm is inefficient to use, makes the workflow slightly more complex and might drastically increase processing time. With that said, the image shape algorithm takes the crown for this type of image steganography.

Table 3. Advantages and disadvantages for the delimiter-based steganography algorithm.

Advantages	Disadvantages
• Non-stego images can be detected • It does not depend on the size of the secret image	• Carrier stego-image quality might be drastically compromised • Secret image size must be up to eight times smaller than carrier image, depends on encoding technique used • Need for a delimiter to be encoded along with the secret data - uses carrier image encoding capacity and makes workflow slightly more complex • Need for all least-significant bits to be extracted for secret image retrieval - creation of irrelevant and unimportant data, makes workflow

Also, the most important metric for stego-images is the Peak Signal-to-Noise Ratio (PSNR) [8] as a way to prove the quality of a stego-image representation. The distortion in the stego-image can be measured by PSNR, and a higher value of PSNR indicates a lesser image distortion. We did an experiment with 400 color carrier images with different sizes and just one secret image of size 100x100 pixels. We choose to work with some popular image sizes like one for Viber image (1600x1200), Facebook and Instagram post images (1080x1080), Instagram Story (1080x1920) and Twitter post image (1200x675) [9]. We computed PSNR numbers to all stego-images obtained with the image shape algorithm, and the delimiter-based algorithm. All the stego-images are encoded using the previously explained encoding methods (LSB, L2SB, and L4SB). We found that some of the stego-images encoded with L4SB method had a PSNR number less than 50, so we decided to remove them from further analyses.

Table 4, shown the average values of PSNRs obtained by the three encoding methods on 400 colored images using the image shape algorithm, and the delimiter-based algorithm. The experimental results showed that LSB method has higher peak signal to noise ratio compared to others for both algorithms.

2.5 Image Encryption

To further strengthen the security, we have also implemented image encryption with authentication for this type of image steganography. We are using the PyNaCl python library for symmetric encryption with authentication. This library uses 32-byte symmetric keys, along with a 24-byte nonce and makes use of the XSalsa20 stream cipher [10], which has been proven to be a reliable and fast stream cipher. It also used Poly1305 MAC [11] authentication for data integrity and authenticity. Since we cannot encrypt the secret image directly, we need to first convert it to a byte-array. After this, we can encrypt the resulting byte-array using PyNaCl, by passing in a custom-generated nonce and symmetric encryption key. The output of the authenticated encryption process is another cipher byte-array, which contains the cipher-text, nonce and authentication data. We then convert that cipher byte-array into binary format by using our custom function and encode the resulting bitstring into our carrier image. During the decoding process, after the data is extracted, the ASCII decimal representation of every extracted byte is retrieved and stored in a byte-array. The byte-array is then decrypted using PyNaCl and the previously generated symmetric key and nonce. After the decryption is over, we load the resulting, decrypted byte-array and return the data and fully retrieve the secret image.

Table 4. PSNR analysis for image shape and delimiter-based algorithms

Encoding method	PSNR (average value)	
	Stego-image obtained with image shape alg.	Stego-image obtained with delimiter-based alg.
LSB	63.55	63.55
L2SB	59.57	59.61
L4SB	51.83	51.85

It is important to mention that image encryption does not work with the image shape algorithm, since the size of the resulting cipher byte-array is dynamic and diverse when it comes down to secret images of different sizes and dimensions. Thus, image encryption only works with the delimiter-based algorithm.

There is still an option to not use image encryption at all. The code has been adjusted and optimized to enable selection of desired algorithm and encryption, by simply passing additional parameters to the encoding and decoding functions. The reason for this is that we don't need the secret image dimensions for image retrieval and reconstruction when we use image encryption, since we retrieve byte data that already has that information encoded in it. When Pickle deserializes that data, the secret image is automatically reconstructed and returned, leaving us with the opportunity of not having to deal with an additional property, that being the shape flag.

3 Machine Learning Implementation of StegIm for Detecting Digital Data in Images

Our StegIm model for detection of hidden images in images is based on the Machine Learning approach, where with the help of supervised learning we train several machine learning models for recognizing stego and non-stego images that were created with the usage of various carriers. We are training and evaluating different classification models, with the purpose of discovering the image dimensions and stego encoding algorithms which produce the best security for hiding information.

We conducted 4 experiments using the LSB encoding method for the following resolutions of the carrier images:

- 1080x1080 pixels (resolution used in posts on Facebook and Instagram)
- 1080x1920 pixels (resolution used in stories on Instagram)
- 1200x675 pixels (resolution used in posts on Twitter)
- 1600x1200 pixels (max resolution used on Viber for sharing images)

In subsection 3.1 we describe the phase of data sets generation and preprocessing. In subsection 3.2 we present the machine learning model architecture that resulted in the best results while in subsection 3.3 we present the best results from the evaluation of the chosen model.

3.1 Data sets Generation and Preprocessing

For all previously defined 4 experiments, we created different datasets. Each dataset corresponds to a different carrier resolution as defined above. We used 100 carriers in all 4 experiments and accordingly 100 secret images with a resolution of 100x100 pixels so that each secret image was hidden in a different carrier. After the encoding was done using the LSB encoding and the image shape algorithm, in a separate textual file we labeled all the stego images with a binary label 1. Lastly, we included in the dataset, the original 100 carriers that were used to hide images, with a binary label 0 (non-stego image).

After the datasets were created, we preprocessed each of the datasets with the help of the previously mentioned function for converting a given image into bits, so basically, we mapped all of the image datasets into a tabular dataset (which can be used in traditional ML approaches) where each column represents the last bit in each of the RGB channels of each pixel. Depending on the experiment here is given the list with the input binary vector sizes:

- Twitter Post Image: 2 430 000
- Facebook/Instagram Post Images: 3 499 200
- Viber Images: 5 760 000
- Instagram Story: 6 220 800

In order to perform an evaluation of the trained models, we split each dataset into a training dataset (80% or 160 images) and a testing dataset (20% or 40 images). The split of the dataset is completely balanced i.e the training dataset has 80 stego images and 80 non-stego images and the testing dataset accordingly has 20 stego and 20 non-stego images.

3.2 Model Architecture

To achieve exceptionally high prediction skills, the gradient boosting method [13] was developed. However, the algorithm's application has been constrained since it requires that just one decision tree be constructed at a time in order to minimize the errors of all earlier trees in the model. Because of this, training even the smallest models took a long time. Gradient boosting underwent a revolution when eXtreme Gradient Boosting

(XGBoost) emerged as a cutting-edge method. In XGBoost, individual trees are constructed over many cores, and data is organized to speed up searches [14]. Model performance increased and model training time was cut in half consequently.

For our problem, we use the following hyper-parameters for XGBoost model:

- Maximal depth of the decision tree: 68
- Number of parallelized decision trees: 10
- Learning rate: 0.16

3.3 Results

Due to hardware and software limitations as well as the huge sizes of the datasets defined for Viber images and Instagram stories, we could not complete the experiments and provide results in this paper.

The following metrics were used for evaluation: F1 score, Accuracy and Recall. The precision and recall of the test dataset are used to calculate the F1 score, with precision being the number of true positive results divided by the total number of positive results, and Recall being the number of true positive results divided by the total number of samples that should have been identified as positive. In Table 5, are shown the results from the evaluation of the StegIm model.

Table 5. Results from the Machine Learning Approach

Image on social media Posts	Accuracy	Recall	F1 Score
Twitter	100%	100%	100%
Facebook/Instagram	97%	98%	97%

In Fig. 3 we present the confusion matrix, which represents the number of False positives, False negatives, True positives, and True negatives obtained with Twitter Image posts and Facebook or Instagram Image posts. Where, in our case the positive class is the one labeled with **1**, in which the secret image is hidden.

From this we can conclude that the performance of our model is quite good for the original images with smaller dimensions, while the evaluation metrics for larger dimensions tend to decline. Therefore, we hope to train the classification model on larger images in the future along with the various image-in-image hiding algorithms. And, with this we will demonstrate through the evaluation metrics that image dimensions are important in the stego encoding algorithms.

3. Conclusion and Future Work

StegIm is a model for hiding and retrieving digital images into other images. It uses the LSB encoding method as commonly used in image steganography. Instead, we

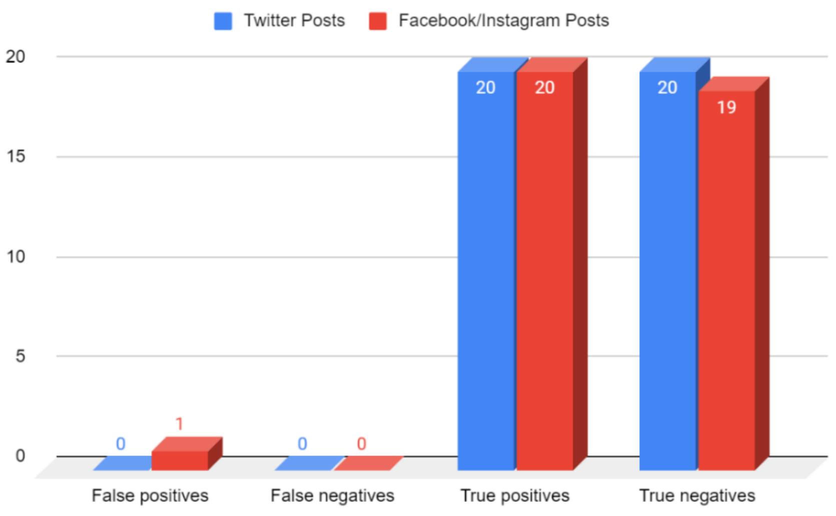

Fig. 3. Prediction of the StegIm model for Twitter, Facebook and Instagram Image Posts

also use L2SB and L4SB encoding methods to increase the encoding capacity but still not compromise the quality of obtained stego-images. The actual parameter for distortion in stego-images is PSNR. The PSNR values obtained from the three methods with the given two algorithms (image shape and delimiter based) are very similar. Besides hiding, StegIm is used for retrieving the secret image from the stego-image. Also, we use the power of machine learning and implement the successful detection of the hidden image in one standard image (without knowing that it is a stego-image).

Our plan for future work is to extend our work to other encoding algorithms and make our comparison analysis broader and more in-depth. Also, we will try to extend StegIm as a tool for hiding/retrieving secret image into multiple images, so that we can focus more on the security part of this methodology.

References

1. Bandyopadhyay, S. K., Bhattacharyya, D., Ganguly, D., Mukherjee, S., Das, P.: "A Tutorial Review on Steganography". University of Calcutta, Senate House, Computer Science and Engineering Department, Heritage Institute of Technology, Anandapur, Kolkata
2. Al-Azzeh, J., Alqadi, Z., Ayyoub, B., Sharadqh, A.: "Improving the Security of LSB Image Steganography". Computer Engineering Department, Al Balqa'a Applied university, Amman, Jordan vol. 3 (4) (2019)
3. Babu, J. , Rangu, S. and Manogna, P.: "A Survey on Different Feature Extraction and Classification Techniques Used in Image Steganalysis".(2017), Journal of Information Security, 8, 186-202. doi: 10.4236/jis.2017.83013.
4. Oludele, A., Sunday,I., Afolashade, K., Uchenna,N.: "Security Test using StegoExpose on Hybrid Deep Learning Model for Reversible Image Steganography" Computer Science Department, Babcock University, Ilishan-Remo, Ogun State, Nigeria.

5. Ki-Hyun J.: “A Study on Machine Learning for Steganalysis”. In Proceedings of the 3rd International Conference on Machine Learning and Soft Computing (ICMLSC 2019). Association for Computing Machinery, New York, NY, USA, 12–15 (2019). DOI:https://doi.org/10.1145/3310986.3311000
6. Lavania, K.: “A Deep Learning Framework to identify real-world stego images”. Masters thesis, Dublin, National College of Ireland (2021).
7. Tasevski, I., Nikolovska, V., Petrova, A., Dobreva, J., Popovska-Mitrovikj, A., Dimitrova, V.: “StegYou: Model for Hiding, Retrieving and Detecting Digital Data in Images”, Future Technologies Conference (FTC 2022), Vancouver, Canada (2022) - accepted
8. Wang Z., Simoncelli, E.P. and Bovik, A. C: "Multiscale structural similarity for image quality assessment," The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003, 2003, pp. 1398-1402 Vol.2, doi: 10.1109/ACSSC.2003.1292216
9. Social Media Image Sizes Guide, https://www.constantcontact.com/blog/social-media-image-sizes/ , last access 2022/06/01.
10. Bernstein, D.J.: Extending the Salsa20 nonce, http://skew2011.mat.dtu.dk/proceedings/Extending%20the%20Salsa20%20nonce.pdf, last access 2022/06/15.
11. Poly1305.com
12. Bernstein, Daniel J. (2005). "The Poly1305-AES Message-Authentication Code". Fast Software Encryption. Lecture Notes in Computer Science. Vol. 3557. pp. 32–49. doi:10.1007/11502760_3. ISBN 978-3-540-26541-2
13. Friedman, J. H.: “Greedy function approximation: a gradient boosting machine”, Annals of statistics. JSTOR, 1189–1232 (2001)
14. Chen, T., Guestrin, C.: “XGBoost: A Scalable Tree Boosting System”. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). Association for Computing Machinery, New York, NY, USA, 785–794. (2016) DOI:https://doi.org/10.1145/2939672.2939785