python lsb_使用Python的LSB图像隐写术

翻译

weixin_26737625 于 2020-09-10 08:06:05 发布 2551 收藏 24
文章标签： python opencv 人工智能 机器学习 计算机视觉
原文链接：https://medium.com/swlh/lsb-image-steganography-using-python-2bbbee2c69a2

python lsb

Hello there!

你好！

In this article, we will understand how to implement **L**east-**S**ignificant **B**it Steganography using Python.

在这篇文章中，我们将了解如何实现**l**使用Python的朝东**šignificant B检查**隐写术。

什么是隐写术？ (WHAT IS STEGANOGRAPHY?)

Steganography is the science that involves communicating secret data in an appropriate multimedia carrier, e.g., image, audio, and video files. It comes under the assumption that if the feature is visible, the point of attack is evident, thus the goal here is always to conceal the very existence of the embedded data.

隐写术是一门涉及在适当的多媒体载体(例如，图像，音频和视频文件)中传输秘密数据的科学。 它是基于以下假设：如果该功能可见，则攻击点显而易见，因此此处的目标始终是隐藏嵌入式数据的存在。

LSB图像密码学 (LSB IMAGE STEGANOGRAPHY)

LSB Steganography is an image steganography technique in which messages are hidden inside an image by replacing each pixel's least significant bit with the bits of the message to be hidden.

LSB隐写术是一种图像隐写术技术，其中消息被隐藏在图像内部，方法是将每个像素的最低有效位替换为要隐藏的消息位。

To understand better, let's consider a digital image to be a 2D array of pixels. Each pixel contains values depending on its type and depth. We will consider the most widely used modes — **RGB(3x8-bit pixels, true-color)** and **RGBA(4x8-bit pixels, true-color with transparency mask).** These values range from 0–255, (8-bit values).

为了更好地理解，我们将数字图像视为像素的2D阵列。 每个像素包含的值取决于其类型和深度。 我们将考虑使用最广泛的模式**-RGB(3x8位像素，真彩色)**和**RGBA(4x8位像素，真色带透明蒙版)。** 这些值的范围是0到255(8位值)。

Image for post

Representation of Image as a 2D Array of RGB Pixels

将图像表示为RGB像素的2D阵列

We can convert the message into decimal values and then into binary, by using the ASCII Table. Then, we iterate over the pixel values one by one, after converting them to binary, we replace each least significant bit with that message bits in a sequence.

我们可以使用ASCII表将消息转换为十进制值，然后转换为二进制。 然后，我们逐个迭代像素值，将其转换为二进制值后，将每个最低有效位替换为序列中的消息位。

To decode an encoded image, we simply reverse the process. Collect and store the last bits of each pixel then split them into groups of 8 and convert it back to ASCII characters to get the hidden message.

要解码编码的图像，我们只需将过程逆转即可。 收集并存储每个像素的最后一位，然后将它们分成8组，然后将其转换回ASCII字符以获取隐藏消息。

PYTHON实现 (PYTHON IMPLEMENTATION)

Now, we will try to implement the above concept step-by-step with the help of Python Libraries — PIL and NumPy.

现在，我们将尝试在Python库PIL和NumPy的帮助下逐步实现上述概念。

Step 1: Import all the required python libraries

第1步： 导入所有必需的python库

```
import numpy as np
from PIL import Image
```

Step 2: Make the Encoder Function

步骤2： 启用编码器功能

Firstly, we write the code to convert the source image into a NumPy array of pixels and store the size of the image. We check if the mode of the image is RGB or RGBA and consequently set the value of **n.** We also calculate the total number of pixels.

首先，我们编写代码以将源图像转换为NumPy像素数组并存储图像的大小。 我们检查图像的模式是RGB还是RGBA，然后设置**n**的值。 我们还计算像素总数。

```
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
        m = 0
    elif img.mode == 'RGBA':
        n = 4
        m = 1
    total_pixels = array.size//n
```

Secondly, we add a delimiter **(“$t3g0")** at the end of the secret message, so that when the program decodes, it knows when to stop. We convert this updated message to binary form and calculate the required pixels.

其次，在秘密消息的末尾添加定界符**(“ $ t3g0”)** ，以便在程序解码时知道何时停止，然后将更新后的消息转换为二进制形式并计算所需的像素。

```
message += "$t3g0"
b_message = ''.join([format(ord(i), "08b") for i in message])
req_pixels = len(b_message)
```

Thirdly, we make a check if the total pixels available is sufficient for the secret message or not. If yes, we proceed to iterating the pixels one by one and modifying their least significant bits to the bits of the secret message until the complete message including the delimiter has been hidden.

第三，我们检查可用总像素是否足以满足秘密消息的要求。 如果是，我们将一一迭代像素并将其最低有效位修改为秘密消息的位，直到隐藏了包括定界符的完整消息为止。

```
if req_pixels > total_pixels:
    print("ERROR: Need larger file size")
else:
    index=0
    for p in range(total_pixels):
        for q in range(m, n):
            if index < req_pixels:
                array[p][q] = int(bin(array[p][q])[2:9] + b_message[index], 2)
                index += 1
```

Finally, we have the updated pixels array and we can use this to create and save it as the destination output image.

最后，我们有更新的像素数组，我们可以使用它来创建并将其保存为目标输出图像。

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

With this, our encoder function is done and should look something like this —

这样，我们的编码器功能就完成了，应该看起来像这样-

```
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
        m = 0
    elif img.mode == 'RGBA':
        n = 4
        m = 1
    total_pixels = array.size//n
    message += "$t3g0"
    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(m, n):
                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")
```

Step 3: Make the Decoder Function

步骤3：启用解码器功能

Firstly, we repeat a similar procedure of saving the pixels of the source image as an array, figuring out the mode, and calculating the total pixels.

首先，我们重复一个类似的过程，将源图像的像素保存为一个数组，确定模式，然后计算总像素。

```
def Decode(src):
    img = Image.open(src, 'r')
    array = np.array(list(img.getdata()))
    if img.mode == 'RGB':
        n = 3
        m = 0
    elif img.mode == 'RGBA':
        n = 4
        m = 1
    total_pixels = array.size//n
```

Secondly, we need to extract the least significant bits from each of the pixels starting from the top-left of the image and store it in groups of 8. Next, we convert these groups into ASCII characters to find the hidden message until we read the delimiter inserted previously completely.

其次，我们需要从图像的左上角开始，从每个像素中提取最低有效位，并将其存储为8组。接下来，我们将这些组转换为ASCII字符以找到隐藏的消息，直到读取之前完全插入的定界符。

```
hidden_bits = ""
for p in range(total_pixels):
    for q in range(m, n):
        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)):
    if message[-5:] == "$t3g0":
        break
    else:
        message += chr(int(hidden_bits[i], 2))
```

Finally, we do a check if the delimiter was found or not. If not, that means there was no hidden message in the image.

最后，我们检查是否找到分隔符。 如果不是，则表示图像中没有隐藏的消息。

```
if "$t3g0" in message:
    print("Hidden Message:", message[:-5])
else:
    print("No Hidden Message Found")
```

With this, our decoder function is done and should look something like this —

这样，我们的解码器功能就完成了，应该看起来像这样-

```
def Decode(src):
    img = Image.open(src, 'r')
    array = np.array(list(img.getdata()))
    if img.mode == 'RGB':
        n = 3
        m = 0
    elif img.mode == 'RGBA':
        n = 4
        m = 1
    total_pixels = array.size//n
    hidden_bits = ""
    for p in range(total_pixels):
        for q in range(m, n):
            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)):
        if message[-5:] == "$t3g0":
            break
        else:
            message += chr(int(hidden_bits[i], 2))
    if "$t3g0" in message:
        print("Hidden Message:", message[:-5])
    else:
        print("No Hidden Message Found")
```

Step 4: Make the Main Function

步骤4： 设定主要功能

For the main function, we ask the user which function they would like to perform — Encode or Decode.

对于主要功能，我们询问用户他们想执行哪个功能-编码或解码。

For Encode, we ask the user the following inputs — source image name with extension, secret message, and destination image name with extension.

对于Encode，我们要求用户输入以下内容-带扩展名的源图像名称，秘密消息和带扩展名的目标图像名称。

For Decode, we ask the user for the source image, that has a message hidden.

对于解码，我们要求用户提供隐藏了消息的源图像。

```
def Stego():
    print("--Welcome to $t3g0--")
    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")
```

Step 5: Put all the above functions together and our own LSB Image Steganography program is ready. Check out my GitHub repository for the complete code.

步骤5： 将以上所有功能放在一起，我们自己的LSB图像隐写程序已准备就绪。 查看我的GitHub存储库以获取完整的代码。

> **NOTE**
>
> **注意**
>
> In a study, it was observed that conventional LSB is not effective in the case of JPEG as the data gets manipulated on compression due to its lossy nature. Whereas for a PNG image a simple LSB is applicable without any loss of data on compression. So, try running your program on PNG images only.
>
> 在一项研究中，观察到常规LSB在JPEG情况下无效，因为数据由于其有损性质而在压缩时受到操纵。 而对于PNG图像，可以使用简单的LSB，而压缩时不会丢失任何数据。 因此，请尝试仅在PNG图像上运行程序。

例 (EXAMPLE)

Image for post

1. Encode Message
   编码信息

Image for post

2. Decode Message

2.解码消息

Image for post

I extensively use this Online Steganography Tool in CTFs, was intrigued to know the code behind it's functioning and that's all the story behind approaching this advanced topic for today.

我在CTF中广泛使用了此在线隐写术工具，对它的功能背后的代码很感兴趣，这就是今天探讨此高级主题的全部故事。

Thank you for reading this article, hope you got to learn something.

感谢您阅读本文，希望您能学到一些东西。

Catch you in the next one!

赶上下一个！

参考资料 (REFERENCES)

https://www.ijsr.net/archive/v4i4/29031501.pdf

https://www.ijsr.net/archive/v4i4/29031501.pdf

https://towardsdatascience.com/hiding-data-in-an-image-image-steganography-using-python-e491b68b1372

https://towardsdatascience.com/hiding-data-in-an-image-image-steganography-using-python-e491b68b1372

> 翻译自: https://medium.com/swlh/lsb-image-steganography-using-python-2bbbee2c69a2

python lsb