# python图片隐写术_使用Python的视频隐写术指南

翻译

weixin_26722031 于 2020-08-30 23:57:36 发布 976 收藏 9
文章标签： python 人工智能 机器学习 深度学习
原文链接：https://medium.com/better-programming/a-guide-to-video-steganography-using-python-4f010b32a5b7

python图片隐写术

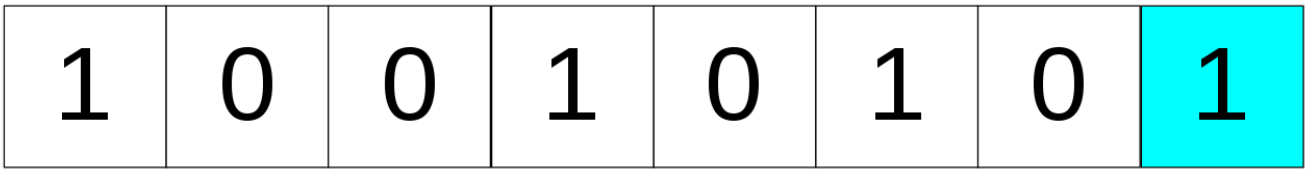
Steganography is the practice of concealing a file, message, image, or video within another file, message, image, or video. It has existed for a long time, and nowadays, digital steganography is used to hide data inside images. We can hide all kinds of data by using different digital steganographic methods.

隐秘术是在另一个文件，消息，图像或视频中隐藏文件，消息，图像或视频的做法。 它已经存在了很长时间，如今，数字隐写术被用于隐藏图像中的数据。 通过使用不同的数字隐写方法，我们可以隐藏各种数据。

In this article, we are going to use a method known as Least Significant Bit (LSB) transformation to hide a text inside a video.

在本文中，我们将使用一种称为最低有效位(LSB)转换的方法来隐藏视频中的文本。

## 什么是最低有效位？ (What Is the Least Significant Bit?)




Wikipedia. 维基百科 。

Above is a representation of the digit 149 as an 8-bit binary digit, with its rightmost bit highlighted. If we change it to 0, we will get 10010100 (decimal equivalent of 148). If we change any other bit in the binary number above, the change would be much greater. So in this example, the rightmost bit is the least significant — changing it results in the least change to the original number.

上面是数字149的表示形式，是一个8位二进制数字，其最右边的一位突出显示。 如果将其更改为0，我们将获得10010100 (相当于148的十进制数)。 如果我们更改上述二进制数中的任何其他位，则更改将更大。 因此，在此示例中，最右边的位是最低有效位-对其进行更改将导致对原始编号的更改最少。

## 我们如何使用LSB转换来隐藏图像中的数据？ (How Can We Use LSB Transformation to Hide Data Inside an Image?)

A digital image is a representation of pixel values, and every pixel value will have numbers containing information regarding the pixel. A digital color image will have red, green, and blue channels and eight bits to represent each channel, so every channel can take a value from 0-255, and this value represents the intensity of the pixel. `(R, G, B)=(0,0,0)` is the representation of the color black and `(255,255,255)` represents the color white.

数字图像代表像素值，每个像素值都有一个数字，其中包含有关像素的信息。 数字彩色图像将具有红色，绿色和蓝色通道以及代表每个通道的八个位，因此每个通道可以采用0-255之间的值，并且该值表示像素的强度。 `(R, G, B)=(0,0,0)`是黑色的表示，而`(255,255,255)`表示白色。

Take an array of pixels as an example and suppose we want to hide the character *A* in it. Let's see how it's done:

以一个像素数组为例，假设我们要在其中隐藏字符*A*。 让我们看看它是如何完成的：

```
(R, G, B)= (11101010 11101001 11001010),(10111001,11001011,11101000),(11001001 00100100 11101001)
```

This is a pixel array, and we want to hide *A* in it. The ASCII value of *A* is 65. If we convert it to binary, we get 01000001. So if we use LSB transformation, we can change the LSB of all the numbers in our array and get:

这是一个像素阵列，我们要在其中隐藏*A*。 *A*的ASCII值为65。如果将其转换为二进制，则将得到01000001。因此，如果使用LSB转换，则可以更改数组中所有数字的LSB并得到：

```
(R,G, B)= (11101010 11101001 11001010),(10111000,11001010,11101000),(11001000 00100101 11101001)
```

Now that we have hidden *A* in the pixel array, let's use Python to hide a text inside an image.

现在我们已经在像素数组中隐藏了*A*，现在让我们使用Python在图像内隐藏文本。

## Python中的图像隐写术 (Image Steganography in Python)

Sourceforge. Sourceforge 。

This is the image that we are going to use. First, we will install the dependencies and then look at the code step by step.

这是我们将要使用的图像。 首先，我们将安装依赖项，然后逐步查看代码。

### 1.安装依赖项 (1. Install the dependencies)

We are going to use stegano library in Python. We can install it by running:

我们将在Python中使用stegano库。 我们可以通过运行以下命令进行安装：

```
$pip install stegano
```

Now we need to import the library and the `lsb` class from it:

现在我们需要从中导入库和`lsb`类：

```
from stegano import lsb
```

Now that we have imported the `lsb` class, let's use its `hide()` method to hide our text inside the image:

现在我们已经导入了`lsb`类，让我们使用它的`hide()`方法将我们的文本隐藏在图像中：

```
secret=lsb.hide('./lenna.png','hello world')
```

This will transform the LSB of our image and hide our message in it. Now we have to save the image:

这将改变我们图像的LSB，并在其中隐藏我们的信息。 现在我们必须保存图像：

```
secret.save('./encoded_image.png')
```

That's it. With just three lines of code, we have hidden the message.

而已。 仅用三行代码，我们就隐藏了该消息。

## 图像比较 (Image comparison)

Encoded image (left) vs. original image (right)

编码图像(左)与原始图像(右)

As you can see, there is no visible difference between these two images.

如您所见，这两个图像之间没有可见的差异。

To reveal the secret message, simply use the `lsb.reveal()` method with the image as the argument and the message will be printed:

要显示秘密消息，只需使用`lsb.reveal()`方法(以图像作为参数)，将显示消息：

```
lsb.reveal('./encoded_image.png')Output:- hello world
```

We have used LSB transformation to hide data inside an image, but how can we use this to hide data inside a video?

我们已经使用LSB转换来隐藏图像内的数据，但是如何使用它来隐藏视频内的数据？

## 视频中的隐写术 (Steganography in Video)

A video is a collection of frames, and each frame is an image. So if we pull out all the frames from a video, we can use this method to store our data using LSB steganography and stitch those frames back into a video with the secret message.

视频是帧的集合，每个帧都是图像。 因此，如果我们从视频中提取所有帧，则可以使用此方法通过LSB隐写术存储数据，然后将这些帧缝在一起并发送带有秘密消息的视频。

Video by Roman Koval on Pexels.

罗曼·科瓦尔（Roman Koval)在 Pexels上 拍摄的视频。

The original source video is quite long, so I have just used an eight-second portion of it.

原始源视频很长，因此我只用了八秒钟的时间。

## 1.从视频中提取帧 (1. Extracting frames from a video)

To extract frames from a video, we can use the most popular computer vision library, OpenCv.

要从视频中提取帧，我们可以使用最受欢迎的计算机视觉库OpenCv 。

First, install OpenCv and import it:

首先，安装OpenCv并将其导入：

```
import cv2
```

We have to read the video. Go through the video frame by frame and save all the images into a new directory.

我们必须看视频。 逐帧浏览视频，然后将所有图像保存到新目录中。

To load the video, we run:

要加载视频，我们运行：

```
vidcap = cv2.VideoCapture("video.mp4")
```

This loads the video into vidcap. Now we can use the `read` method to read the frames from “video.mp4:"

这会将视频加载到vidcap。 现在，我们可以使用`read`方法从“ video.mp4”中读取帧：

```
success, image = vidcap.read()
```

We can define a loop to go through all the frames and save it with a unique filename that we can easily sort:

我们可以定义一个循环遍历所有框架，并将其保存为可以轻松排序的唯一文件名：

Once this is done, we are left with all the frames from the video.

完成此操作后，我们将保留视频中的所有帧。

But a video is not just a collection of images. There is audio as well. To extract that audio, we will use FFmpeg.

但是，视频不仅仅是图像的集合。 也有音频。 要提取该音频，我们将使用FFmpeg 。

## 2.从视频中提取音频 (2. Extracting audio from a video)

FFmpeg is a free and open-source command-line tool for transcoding multimedia files. It contains a set of shared audio and video libraries such as libavcodec, libavformat, and libavutil. With FFmpeg, you can extract audio files from a video, convert your PNG image files into video, and much more.

FFmpeg是一个免费的开源命令行工具，用于对多媒体文件进行代码转换。 它包含一组共享的音频和视频库，例如libavcodec，libavformat和libavutil。 使用FFmpeg，您可以从视频中提取音频文件，将PNG图像文件转换为视频，等等。

To install FFmpeg in Ubuntu, first update the package list:

要在Ubuntu中安装FFmpeg，请首先更新软件包列表：

```
$sudo apt update
```

Then run the command below to install FFmpeg:

然后运行以下命令以安装FFmpeg：

```
$
```

To validate that FFmpeg is installed properly, run:

要验证FFmpeg是否已正确安装，请运行：

```
ffmpeg -version
```

If you get something like the message below, everything is OK:

如果您收到类似以下消息的信息，则一切正常：

```
ffmpeg version n4.1.4 Copyright (c) 2000-2019 the FFmpeg developers
```

To use FFmpeg in Python, we have to import call and STDOUT from the subprocess library:

要在Python中使用FFmpeg，我们必须从子进程库中导入call和STDOUT：

```
from subprocess import call,STDOUT
```

And then we can run the code below in Python:

然后，我们可以在Python中运行以下代码：

```
call(["ffmpeg", "-i","video.mp4" , "-q:a", "0", "-map", "a", "tmp/audio.mp3", "-y"],stdout=open(os.de
```

This code will extract the audio from the given video file and save it as “audio.mp3” in the tmp folder. After we encode the frames with our text, we can then use this audio file to give our encoded video file the proper audio.

此代码将从给定的视频文件中提取音频，并将其另存为tmp文件夹中的“ audio.mp3”。 用文本对帧进行编码后，我们可以使用此音频文件为编码的视频文件提供适当的音频。

## 3.在框架内编码文本 (3. Encoding the text inside frames)

Now that we have all the frames, we can divide the strings into small chunks and hide each chunk of the message inside a frame using the `lsb.hide()` method:

现在我们有了所有框架，我们可以将字符串分成小块，并使用`lsb.hide()`方法将消息的每个块隐藏在框架内：

`split_string` is a helper function to split strings into small portions. As we do not need to hide all the text in the first frame itself, we divide the frames and hide it in many frames. The full code can be found on GitHub.

`split_string`是一个辅助函数，用于将字符串分成小部分。 由于我们不需要在第一帧本身中隐藏所有文本，因此我们将这些帧划分为多个帧并将其隐藏。 完整的代码可以在GitHub上找到 。

We have loaded a video, pulled it to as many frames as possible, and encoded it. Now we have to put together the frames together into a video.

我们已经加载了视频，将其拉到尽可能多的帧并进行了编码。 现在，我们必须将这些帧放到一个视频中。

## 4.从帧制作视频 (4. Making video from frames)

We can use FFmpeg to stitch together all our frames with a hidden message to form a video and then lay out the audio:

我们可以使用FFmpeg将所有帧与一条隐藏消息拼接在一起，以形成视频，然后布置音频：

```
call(["ffmpeg", "-i", "tmp/frame%d.png" , "-vcodec", "png", "video.mov", "-y"],stdout=open(os.d
```

Running the code above creates the video with our secret message hidden in it.

运行上面的代码会在视频中隐藏我们的秘密消息。

We can run the code below if we want sounds in our video:

如果我们想要在视频中发出声音，我们可以运行以下代码：

```
call(["ffmpeg", "-i", "temp/video.mov", "-i", "temp/audio.mp3", "-codec", "copy","data/enc-" +
```

And we have done it. The output video is almost 900 MB. I have uploaded a small portion of the video to show that video quality is not affected by this method.

我们已经做到了。 输出的视频将近900 MB。 我上传了视频的一小部分，以表明视频质量不受此方法的影响。

Now that we have done the hard part of encoding, let's look at how a person can decode our video and read the message.

既然我们已经完成了编码的困难部分，那么让我们看一个人如何解码我们的视频并阅读消息。

### 解密视频中的消息 (Decrypting the message from the video)

We have to do the same steps as with encryption, but in the opposite order. So we extract all the frames from the video and extract the information from the LSB:

我们必须执行与加密相同的步骤，但顺序相反。 因此，我们从视频中提取所有帧，并从LSB中提取信息：

## 结论 (Conclusion)

As you can see, with modern libraries and languages like Python, it is really simple to do steganography and cryptography.

如您所见，使用现代库和Python之类的语言，进行隐写术和加密非常简单。

The only problem with this method is that the new video is huge in size when compared to the original video. If we could reduce the size, it would be a good addition to this project.

这种方法的唯一问题是，与原始视频相比，新视频的尺寸很大。 如果我们可以减小尺寸，那将是对该项目的一个很好的补充。

The full code for this project can be found on GitHub.

该项目的完整代码可以在GitHub上找到 。

> 翻译自: https://medium.com/better-programming/a-guide-to-video-steganography-using-python-4f010b32a5b7

python图片隐写术