【转载】什么是隐写技术？（Kaggle近期竞赛ALASKA2 Image Steganalysis）

翻译

囚生CY 于 2020-05-26 11:26:18 发布 12195 收藏 9
分类专栏： 竞赛 文章标签： python
原文链接：https://www.kaggle.com/tanulsingh077/steganalysis-complete-understanding-and-model


竞赛 专栏收录该内容
5 篇文章 0 订阅
订阅专栏

前言

最近***Kaggle***上有一个很有趣的竞赛（ALASKA2 Image Steganalysis），是与隐写技术相关的。竞赛的目标是需要参赛者给测试集上的**5000**张可能被隐写技术加密过的图片上进行标注，以注明这些图片是否被使用***JMiPOD***, ***JUNIWARD***, ***UERD***三种之一的隐写技术加密。截至本文发布，***Leadboard***上的前**50**的排名已经普遍达到了***AUC***值**0.9**向上的高水平，毕竟机器学习在零一分类的领域还是非常高效的。

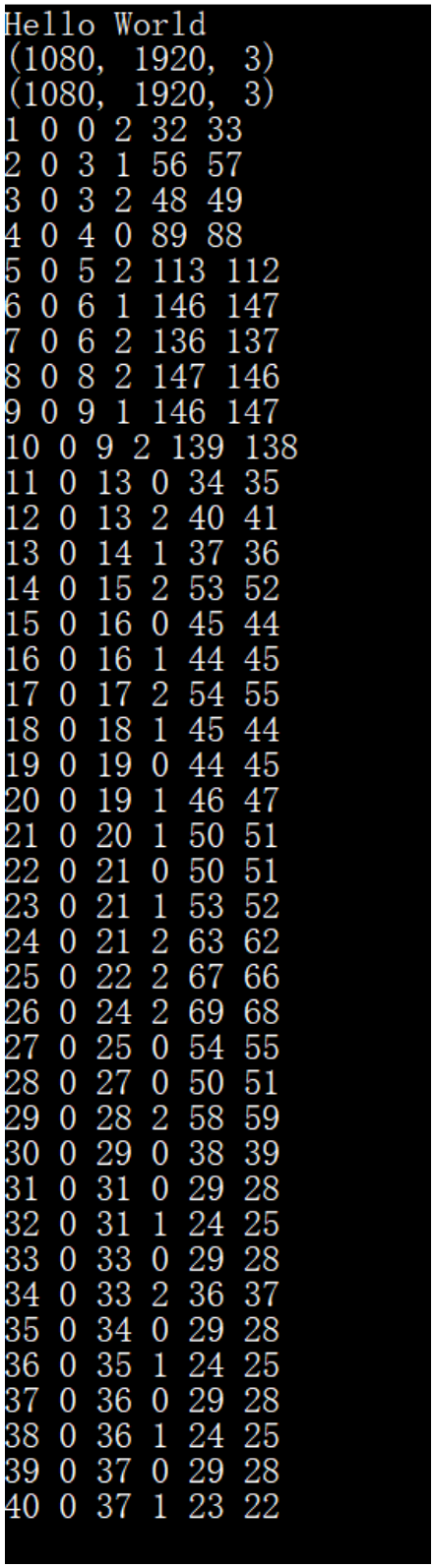
笔者对图片数据处理领域涉足很少，关于隐写技术就更是一窍不通。关于隐写技术，***python***已经有相关***package***可用

```
pip install stegano
```

这个包倒是意外的很小， 简单选一张图片，将一句话嵌入其中进行加密，看看加密后的结果如何

```
import numpy as np
from PIL import Image

# 加密并保存
from stegano import lsb
secret = lsb.hide("4.png", "Hello World")
secret.save("4_new.png")
clear_message = lsb.reveal("4_new.png")
print(clear_message)

# 看看加密前与加密后的图片有什么区别吧
old = Image.open("4.png")
new = Image.open("4_new.png")

old_array = np.asarray(old)
new_array = np.asarray(new)

print(old_array.shape)
print(new_array.shape)

index1 = -1
count = 0
for i,j in zip(old_array,new_array):
 index1 += 1
 index2 = -1
 for i1,j1 in zip(i,j):
  index2 += 1
  index3 = -1
  for i2,j2 in zip(i1,j1):
   index3 += 1
   if not i2==j2:
    count += 1
    print(count,index1,index2,index3,i2,j2)
```

运行输出



可以看到将**Hello World**这段文本嵌入其中后，前后图片在**40**个像素点上的***RGB***值之一加1或者减1，以达到了信息隐写的加密的效果，不过显然仅**40**个像素点上的微小变化是不足以用肉眼能看出区别的，这也就是隐写技术的妙处。

本文翻译了这次竞赛目前***kernal***中***Hottest***的一篇***Notebook***，它详细介绍了本次竞赛的数据集以及隐写技术，非常适合入门的朋友进行学习。笔者看完觉得受益匪浅，决定转载以做记录保存。

比赛链接：https://www.kaggle.com/c/alaska2-image-steganalysis

原文链接：https://www.kaggle.com/tanulsingh077/steganalysis-complete-understanding-and-model

---

关于本次比赛

“等等！什么？另一个，这不可能发生的人”，冷静下来，好朋友！ ，这只是隔离区，它与Kaggle混在一起。 Kaggle即将在本周举行的第4项比赛中竞争，但我们需要作为良好的社区成员来支持Kaggle，对吗？ 我的意思是这是艰难的时刻。 不过，这是一场有趣的比赛，我的意思是Stegography，现在我们在谈论，是时候给我们所有我们长期以来被压制的间谍幻想，一些空气，然后潜入水中。

这项竞赛希望我们创造一种有效且可靠的方法来检测隐藏在无害的数字图像中的秘密数据。

关于本文

大多数人可能会认为：这场比赛不适合我，我对笔法和笔法分析的了解和知识为零，但是我们不喜欢它们，是吗？

好吧，我留给您决定，就我而言，我不是，我渴望学习，探索，并且最重要的是获得了NSA特工的感觉。因此，我将深入研究本笔记本，并以此为基础从零到发布解决方案。如果您愿意，我们可以一起做，因为它只需要学习的意愿。

我们将在此笔记本中看到以下内容：

1. 什么是隐写术和隐写分析？ ：基本定义
2. 隐写分析的知识方法
3. 我们做错了吗？
4. 隐写术：完全理解它是什么，为什么我们做错了一切？
5. 解决方案的新改进

到最后，您将对数据和比赛有所有的了解，并且将有足够的能力在这次比赛中表现出色。

该笔记本正在进行中，如果该笔记本得到支持，我将继续用我在这次比赛中取得的最新进展对其进行更新。

如果您喜欢我的努力和贡献，请通过支持我的kernel表示感谢。

```
# PRELIMINARIES
import os
from stegano import lsb #USED FOR PNG IMAGE
import skimage.io as sk
import matplotlib.pyplot as plt
from scipy import spatial
from tqdm import tqdm

from PIL import Image
from random import shuffle


import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

Q1）什么是隐写术？

让我们深入了解我们的第一点

用纯英语讲，它具有以下含义：在其他非秘密文本或数据中隐藏消息或信息的做法。

以下是一篇研究论文的描述：“隐写术通常是指可以在无害的掩盖对象中隐藏信息的技术和方法。由此产生的隐身对象尽可能地类似于原始掩盖对象。因此，它可以通过可能受到窃听者窃听的不安全通信信道发送”

在这场竞争中，它意味着将信息隐藏在数字图像中。用于数字水印和数字标记/版权的技术相同。但是不限于此，任何人都可以使用它来隐藏信息，绕过证券，联系特定人员，发送恶意内容等。现在您明白了，为什么我要谈论所有间谍工作。

如今，隐写术主要用于数字图像，这是因为它们在Internet上的广泛存在，普遍采用的JPEG压缩方案及其相对简单的方式都可以修改。

黑白差异隐写分析和隐写术

隐写术是对可行对象源上的信息进行编码，而隐写术是对来自已编码对象的信息进行解码。

在开始原始数据集和隐写分析问题之前，让我们首先尝试以有趣的方式执行隐写术，使用一个称为stegano的python模块，在完成隐写术之后，我们将了解如何完成隐写术。

可以从这里的文档中阅读更多内容

我添加了一些png图片，您可以添加自己的图片并在其中玩耍

```
image = sk.imread("/kaggle/input/png-for-steg/bald-eagle-png-transparent-image-pngpix-7.png")
```

```
secret = lsb.hide("/kaggle/input/png-for-steg/bald-eagle-png-transparent-image-pngpix-7.png", "I will be th
secret.save("encoded.png")
```

我们只是在图像中用两行代码隐藏了一条消息，很酷吧？
让我们尝试看一下图像以观察是否有任何差异

```
img1 = sk.imread("/kaggle/input/png-for-steg/bald-eagle-png-transparent-image-pngpix-7.png")
img2 = sk.imread("/kaggle/working/encoded.png")

fig,ax = plt.subplots(1,2,figsize=(18,8))

ax[0].imshow(img1)
ax[1].imshow(img2)
```

惊讶吗 无法分辨出差异吗？ 这很特殊，现在我们为什么这么难
Stegano还提供了对隐藏在我们图像中的消息进行解码的功能，让我们尝试一下

```
print(lsb.reveal("/kaggle/working/encoded.png"))

# I will be there but you can't find me even if I'm a very very very long sentence
```

这样我们就可以看到我们隐藏的文本，但是幕后到底是怎么回事？

该模块使用一种技术在封面图像的各个部分中创建一个隐蔽通道，与人类视觉系统（HVS）相比，其中的更改可能会显得很少，它将信息隐藏在最低有效位（LSB）中。图像数据。这种嵌入方法基本上基于以下事实：可以将图像中的最低有效位视为随机噪声，因此，它们变得对图像的任何变化均无响应

这是在图像上进行隐写术的第一种经典方法。众所周知的基于LSB嵌入的隐写术工具在隐藏信息方面有所不同。他们中的一些人随机更改像素的LSB，其他人则不是在整个图像中而是在其选定区域中更改像素，还有一些人则增加或减少LSB的像素值，而不是更改值

因此，现在我们了解了stegano的工作原理以及最早的，最古老的stegnanalyis方法之一，我们可以使用knaive方法来检测消息是否已在图像文件中编码。以下是一种简单的方法：

RGB图像以3-D numpy数组的形式存储，其中每个维度包含红色，绿色和蓝色的像素。其值在0到255之间变化。我们可以将numpy数组展平为rdim x的变容体gdim x bdim。这将为我们提供img的特征向量

现在我们可以找到正常图像和编码图像的两个向量之间的余弦相似度，如果它们相似，则余弦不相似度（1-相似度）必须为另一个，小于1，我们可以得出这样的想法：错误

让我们做这个练习，看看结果

```
vec1 = np.reshape(img1,(1151*1665*4))
vec2 = np.reshape(img2,(1151*1665*4))
```

首先验证一下我们的论点，即对于完全相同的向量，余弦相似度为1

```
print(1 - spatial.distance.cosine(vec1,vec1))
print(1 - spatial.distance.cosine(vec1,vec2))
print(spatial.distance.cosine(vec2,vec1))

# 1.0
# 0.9999991304853004
# 8.695146995751912e-07
```

因此，我们可以看到，无需付出太多努力，就可以使用幼稚的方法来查找信息是否隐藏。

我尝试将此knaive与主要数据集一起使用，但是组织者意识到这种knaive方法没有错，但并不完全正确，因为信息隐藏在图像的DCT系数中，而不是RGB像素中。

如果这一切都没有意义，请不要担心，我们将研究所有内容，但首先请看一些可视化

数据挖掘

既然我们对隐写术的技术很熟悉，那么我们现在可以转到数据探索和隐写分析部分

```
BASE_PATH = "/kaggle/input/alaska2-image-steganalysis"
train_imageids = pd.Series(os.listdir(BASE_PATH + '/Cover')).sort_values(ascending=True).reset_index(drop=T
test_imageids = pd.Series(os.listdir(BASE_PATH + '/Test')).sort_values(ascending=True).reset_index(drop=Tru
```

```
cover_images_path = pd.Series(BASE_PATH + '/Cover/' + train_imageids ).sort_values(ascending=True)
JMIPOD_images_path = pd.Series(BASE_PATH + '/JMiPOD/'+train_imageids).sort_values(ascending=True)
JUNIWARD_images_path = pd.Series(BASE_PATH + '/JUNIWARD/'+train_imageids).sort_values(ascending=True)
UERD_images_path = pd.Series(BASE_PATH + '/UERD/'+train_imageids).sort_values(ascending=True)
test_images_path = pd.Series(BASE_PATH + '/Test/'+test_imageids).sort_values(ascending=True)
ss = pd.read_csv(f'{BASE_PATH}/sample_submission.csv')
```

因此，组织者已使用四种算法将数据编码到封面图像中，它们是[JUNIWARD，JMiPOD和UERD]

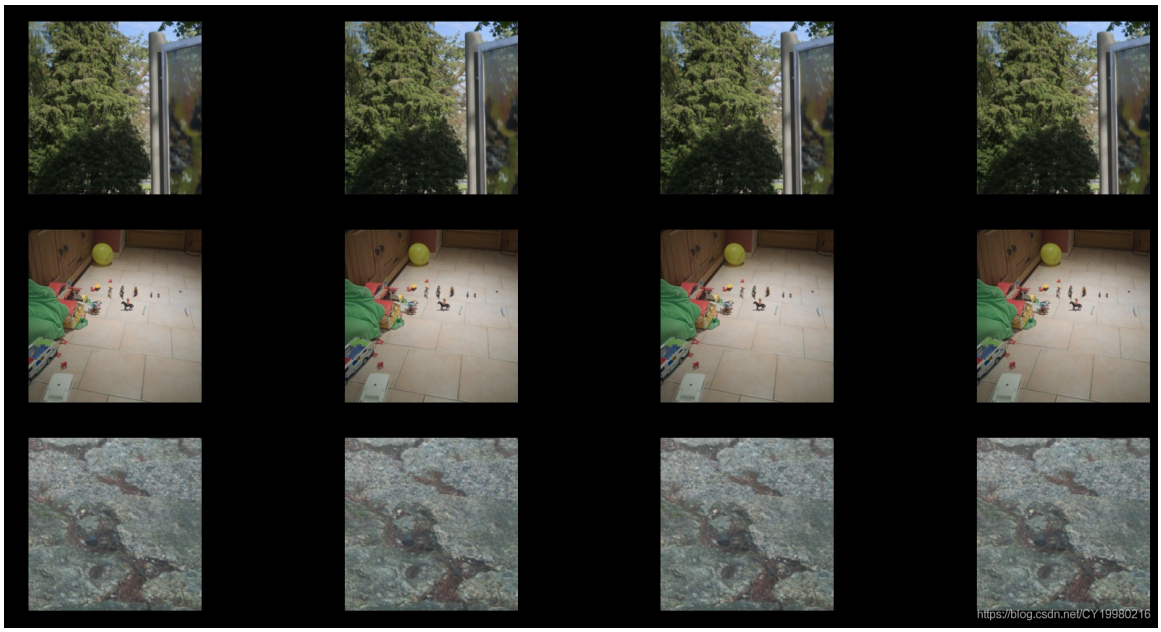
1. UNIWARD：通用小波相对失真，描述CNN用于隐匿JUNIWARD图像的论文https://arxiv.org/ftp/arxiv/papers/1704/1704.08378.pdf
2. JMiPOD：描述CNN用于隐匿JMiPOD图像的论文https://hal-utt.archives-ouvertes.fr/hal-02542075/file/J_MiPOD_vPub.pdf
3. UERD：均匀嵌入再次失真，尚未找到任何资源，我会在发现问题后立即进行更新

现在，仅假设它们是用于将数据编码为图像的某些算法，我们将很快了解所有内容

```
#VISUALIZING SOME IMAGES FROM COVER SECTION
fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(30, 15))
k=0
for i, row in enumerate(ax):
    for j, col in enumerate(row):
        img = sk.imread(cover_images_path[k])
        col.imshow(img)
        col.set_title(cover_images_path[k])
        k=k+1
plt.suptitle('Samples from Cover Images', fontsize=14)
plt.show()
```

让我们并排可视化封面图像和编码图像

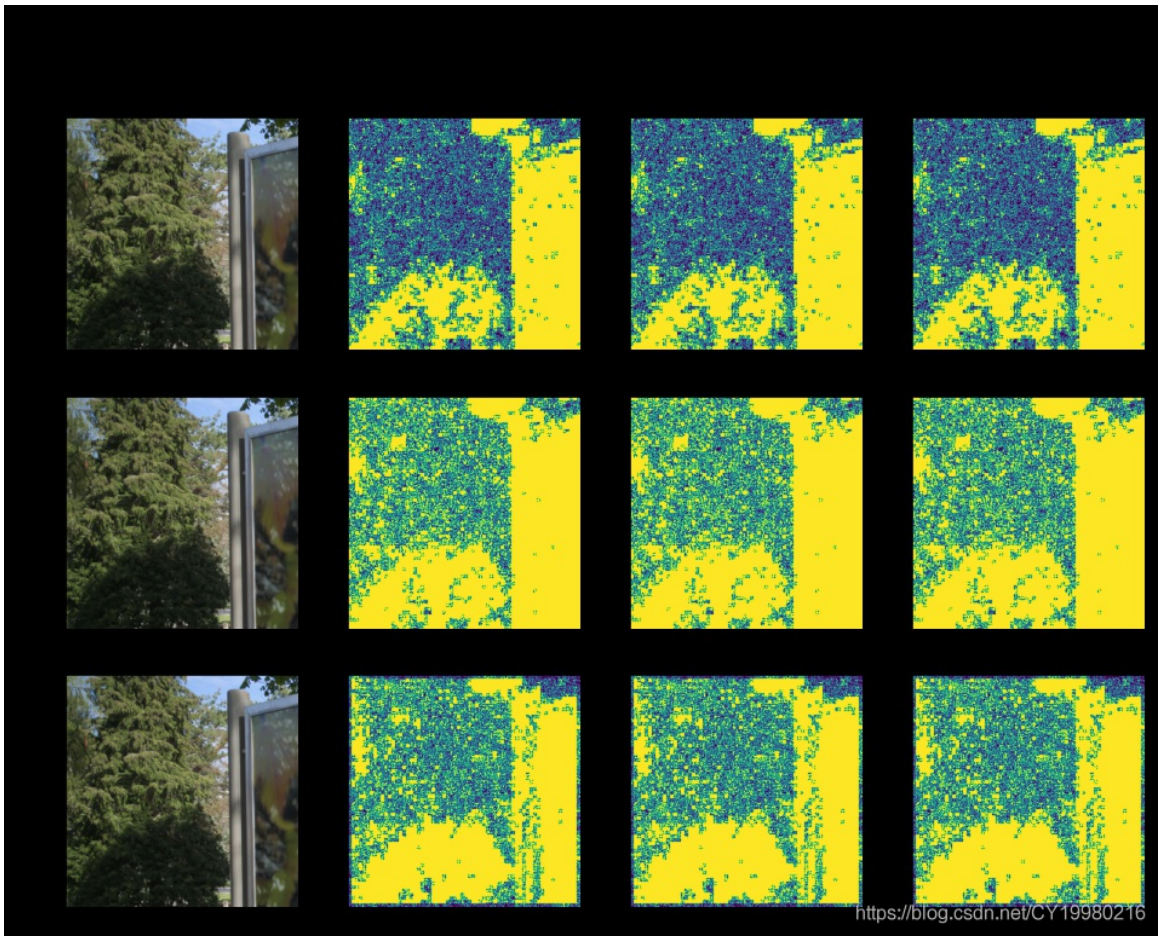
```
fig, ax = plt.subplots(nrows=3, ncols=4, figsize=(30, 15))
for i in range(3):
    '''
    If you want to print more images just change the values in range and ncols in subplot

    '''
    cvimg = sk.imread(cover_images_path[i])
    uniimg = sk.imread(JUNIWARD_images_path[i])
    jpodimg = sk.imread(JMIPOD_images_path[i])
    uerdimg = sk.imread(UERD_images_path[i])

    ax[i,0].imshow(cvimg)
    ax[i,0].set_title('Cover_IMG'+train_imageids[i])
    ax[i,1].imshow(uniimg)
    ax[i,1].set_title('JNIWARD_IMG'+train_imageids[i])
    ax[i,2].imshow(jpodimg)
    ax[i,2].set_title('JMiPOD_IMG'+train_imageids[i])
    ax[i,3].imshow(uerdimg)
    ax[i,3].set_title('UERD_IMG'+train_imageids[i])
```



不出所料，我们无法用肉眼看到任何差异，我们知道使用失真函数可以将信息隐藏在某个地方

我将尝试对图像通道中的像素偏差进行最后的可视化，并查看其是否有效，想法取自：https://www.kaggle.com/iamleonie/alaska2-first-look-into-the-wild-data

```
img_cover = sk.imread(cover_images_path[0])
img_jmipod = sk.imread(JMIPOD_images_path[0])
img_juniward = sk.imread(JUNIWARD_images_path[0])
img_uerd = sk.imread(UERD_images_path[0])


fig, ax = plt.subplots(nrows=3, ncols=4, figsize=(16, 12))
ax[0,0].imshow(img_jmipod)
ax[0,1].imshow((img_cover == img_jmipod).astype(int)[:,:,0])
ax[0,1].set_title(f'{train_imageids[k]} Channel 0')

ax[0,2].imshow((img_cover == img_jmipod).astype(int)[:,:,1])
ax[0,2].set_title(f'{train_imageids[k]} Channel 1')
ax[0,3].imshow((img_cover == img_jmipod).astype(int)[:,:,2])
ax[0,3].set_title(f'{train_imageids[k]} Channel 2')
ax[0,0].set_ylabel('JMiPOD', rotation=90, size='large', fontsize=14)


ax[1,0].imshow(img_juniward)
ax[1,1].imshow((img_cover == img_juniward).astype(int)[:,:,0])
ax[1,2].imshow((img_cover == img_juniward).astype(int)[:,:,1])
ax[1,3].imshow((img_cover == img_juniward).astype(int)[:,:,2])
ax[1,0].set_ylabel('JUNIWARD', rotation=90, size='large', fontsize=14)

ax[2,0].imshow(img_uerd)
ax[2,1].imshow((img_cover == img_uerd).astype(int)[:,:,0])
ax[2,2].imshow((img_cover == img_uerd).astype(int)[:,:,1])
ax[2,3].imshow((img_cover == img_uerd).astype(int)[:,:,2])
ax[2,0].set_ylabel('UERD', rotation=90, size='large', fontsize=14)

plt.suptitle('Pixel Deviation from Cover Image', fontsize=14)

plt.show()
```



因此，再次进行这些可视化时，组织者再次指出，比较普通图像和隐秘图像的RGB通道的像素值并不完全正确。 https://www.kaggle.com/c/alaska2-image-steganalysis/discussion/147494 ，请在继续阅读之前阅读此内容，如果您不了解它，请不要担心，我们将深入研究

要理解所有内容，我们必须更深入地了解JPEG和Stregnography的基本原理

我们做错了吗？

1）我们将一步一步地了解所有内容，因此让我们从“ JPEG”开始吧？

因此JPEG不是文件格式，它是图像文件的压缩算法，可以减小其大小，而又不会丢失很多信息。因此，让我们看一下使用JPEG图像压缩图像时要执行的步骤：

1. 首先，图像从RGB通道转换为YCbCr。 YCbCr和RGB都是具有不同通道的色彩空间，其中YCbCr包括三个通道，即Luminance（Y），Cb（Cb是蓝色减去亮度（B-Y）），Cr（Cr是红色减去亮度（R-Y））。要了解有关色彩空间的更多信息，请观看此视频
2.
3. 然后使用DCT coeff将DCT应用于这些通道的像素，要了解DCT，请观看此视频：https://www.youtube.com/watch?v=LFXN9PiOGtY&list=PLzH6n4zXuckoAod3z31QEST1ZaizBuNHh&index=1
4. 使用JPEG算法编码的图像会保留在YCbCr颜色空间中，直到由图像查看器软件对其进行解码为止。读取JPEG后，将使用上述视频中描述的技术将其解码并转换回RGB色彩空间，以在屏幕上呈现

既然我们知道了JPEG的工作原理，那么让我们了解一下笔迹学是如何工作的，以及多年来使用的技术是如何发展的

那么您认为JPEG文件中潜在的信息隐藏点是什么？

各个颜色空间的通道，例如RGB和YCbCr。

这些是启动算法中使用的技术，但是正如我们已经看到的，即使使用knaive方法也很容易识别它，因此人们开始寻找一种隐藏信息的新技术。

不同通道的DCT系数

更新的方法将信息隐藏在JPEG图像的不同通道的DCT系数中，并且考虑DCT系数的统计信息，有效载荷（秘密数据）也随机分布在其中

观看此视频后（https://www.youtube.com/watch?v=Q2aEzeMDHMA&list=PLzH6n4zXuckoAod3z31QEST1ZaizBuNHh&index=3），您将清楚地了解到现在为止讨论的所有内容

Knaive方法发生了什么

我已经从头开始重新制作了这个笔记本，至于我对这个问题的专业方法，即基于像素差异将其作为回信问题，如果您想在这里了解，请在这里添加：https://www.kaggle.com/tanulsingh077/creating-labels-for-steganalysis

改进的新方法

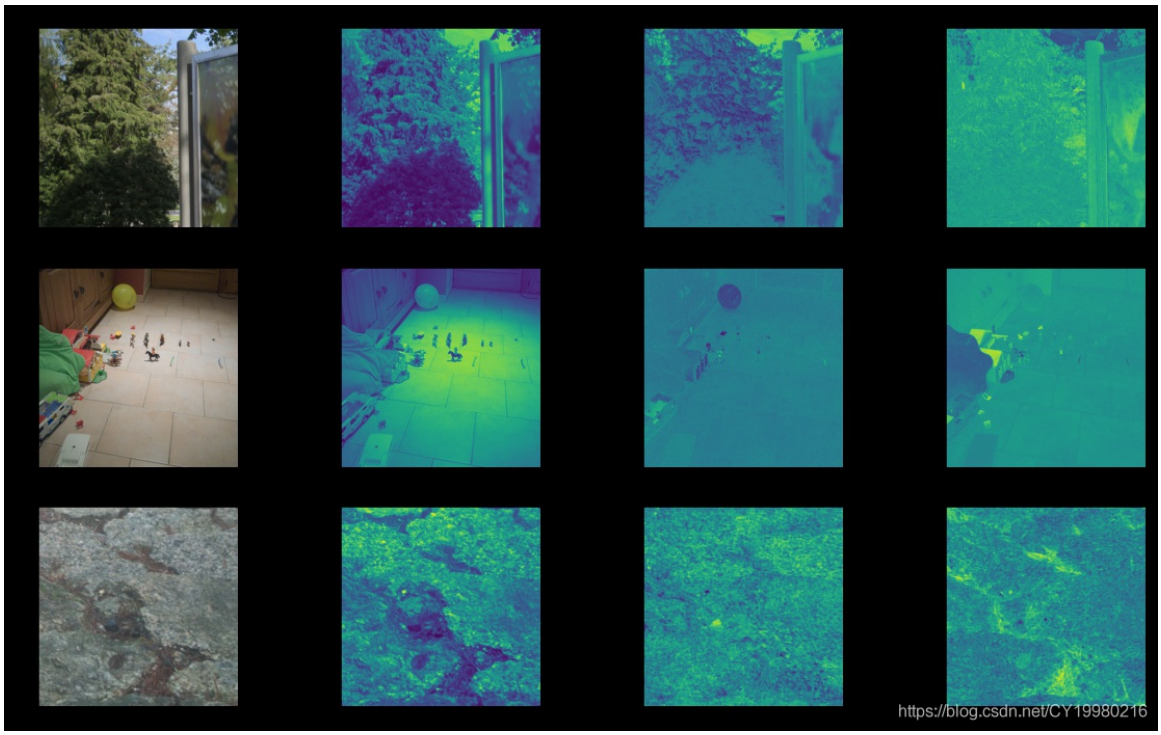
所以现在我们必须着手进行模型构建：

1. 像现在一样，使用像素作为神经网络的输入，我们也可以使用朴素方法
2. 使用DCT coeff代替像素值作为我们神经网络的输入来预测标签

在本节中，我们将尝试并确定什么是好主意（您对此有何想法，让我们在评论中进行讨论）

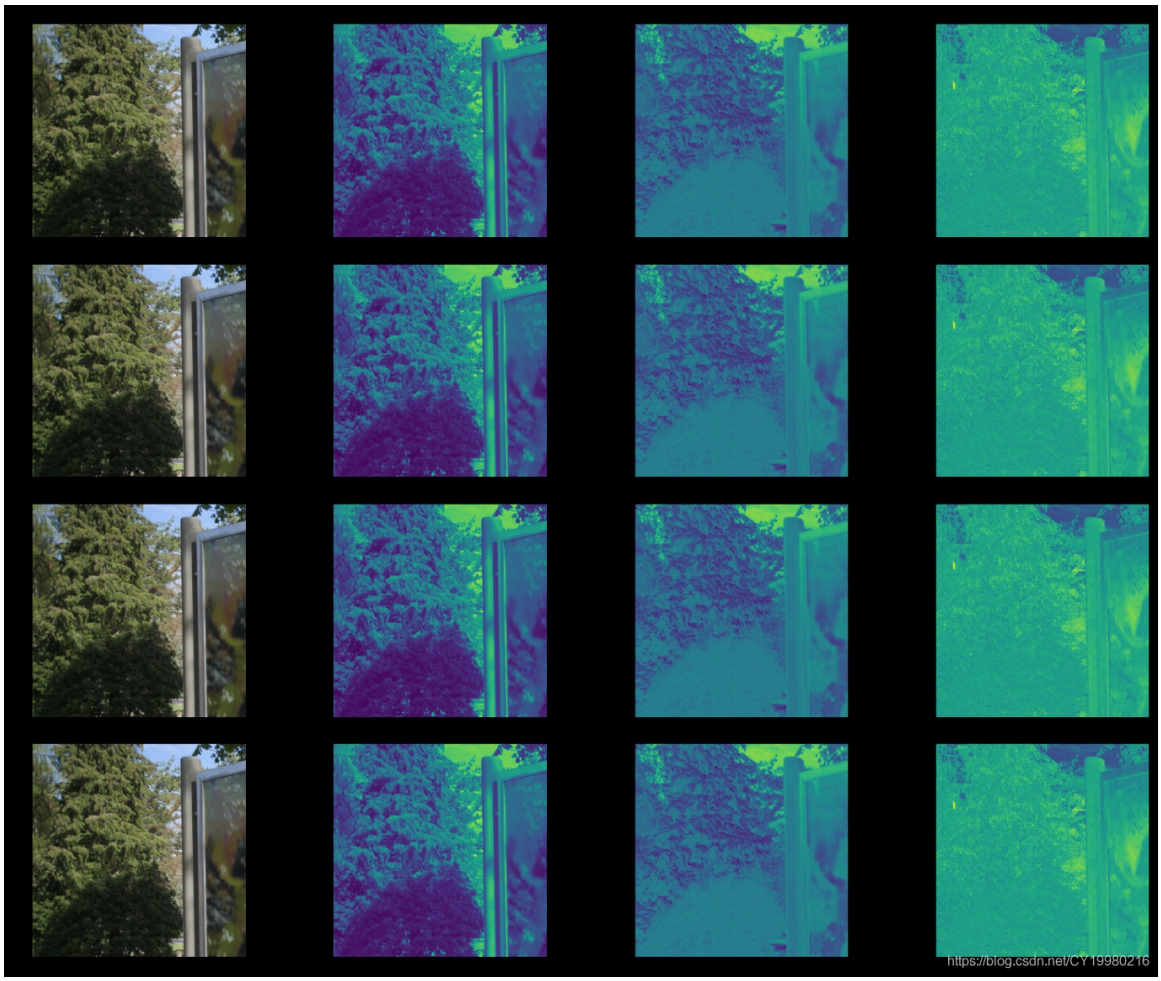
首先，让我们看看如何可视化YCbCr色彩空间图像的不同通道

```
fig,ax = plt.subplots(3,4,figsize=(20,12))

for i,paths in enumerate(cover_images_path[:3]):
    image = Image.open(paths)
    ycbcr = image.convert('YCbCr')
    (y, cb, cr) = ycbcr.split()

    ax[i,0].imshow(image)
    ax[i,0].set_title('Cover'+train_imageids[i])
    ax[i,1].imshow(y)
    ax[i,1].set_title('Luminance')
    ax[i,2].imshow(cb)
    ax[i,2].set_title('Cb:Chroma Blue')
    ax[i,3].imshow(cr)
    ax[i,3].set_title('Cr:Chroma Red')
```



现在，让我们并排探索Cover和编码图像的YCbCr的不同通道

```
fig,ax = plt.subplots(4,4,figsize=(20,16))
plt.tight_layout()


im1 = Image.open(cover_images_path[0])
im2 = Image.open(JUNIWARD_images_path[0])
im3 = Image.open(JMIPOD_images_path[0])
im4 = Image.open(UERD_images_path[0])

for i,image in enumerate([im1,im2,im3,im4]):
    ycbcr = image.convert('YCbCr')
    (y, cb, cr) = ycbcr.split()

    ax[i,0].imshow(image)
    ax[i,0].set_title('Image')
    ax[i,1].imshow(y)
    ax[i,1].set_title('Luminance')
    ax[i,2].imshow(cb)
    ax[i,2].set_title('Cb:Chroma Blue')
    ax[i,3].imshow(cr)
    ax[i,3].set_title('Cr:Chroma Red')
```



在涵盖了以上部分中的所有视频之后，我们知道如何使用JPEG压缩图像以及如何使用量化和编码来计算DCT系数

现在我们可以更深入地分析哪种模型更好。 我们将首先获取图像的DCT系数并将其可视化

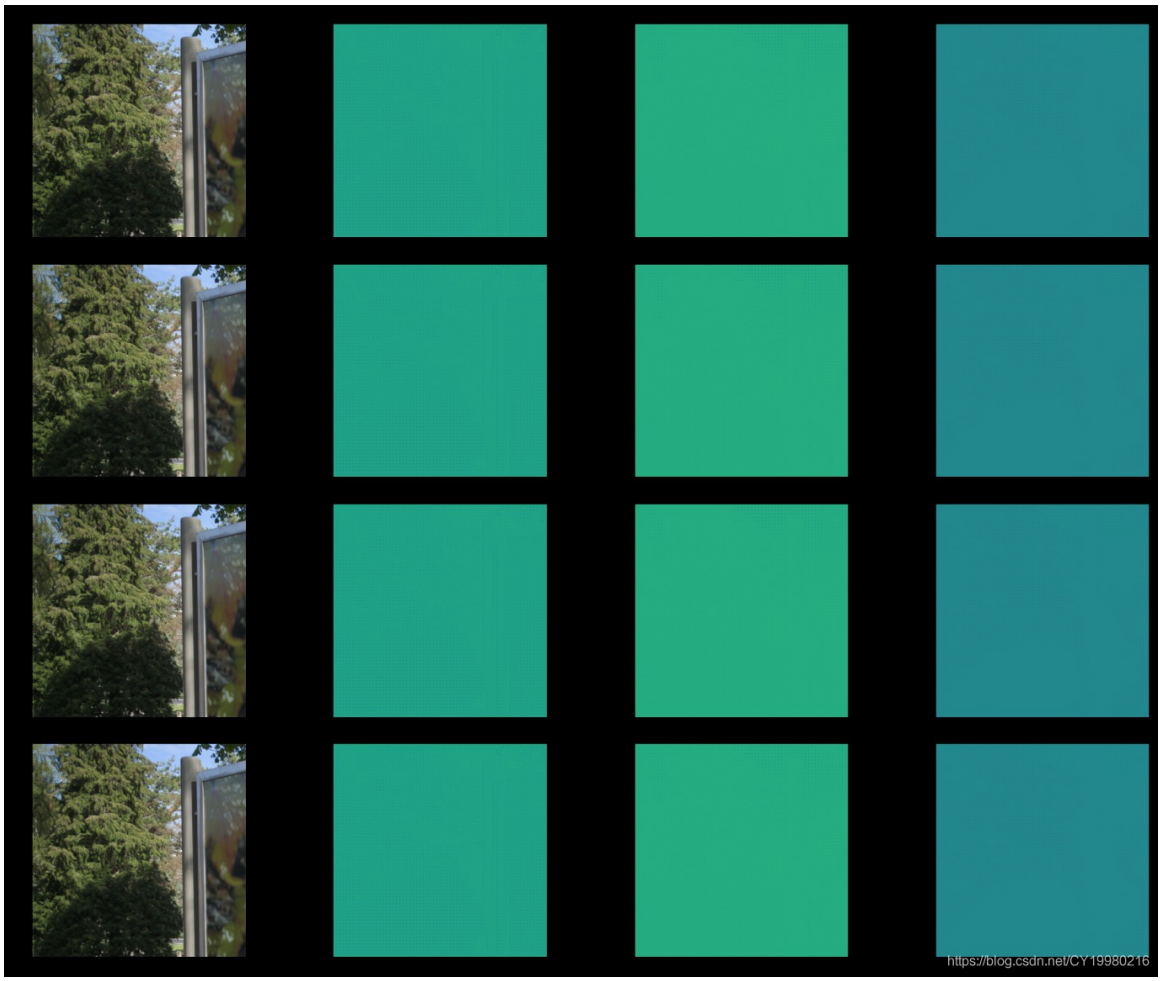
```
! git clone https://github.com/dwgoon/jpegio
```

```
!pip install jpegio/.
import jpegio as jio
```

package的使用和审查可以在这里查看https://github.com/dwgoon/jpegio

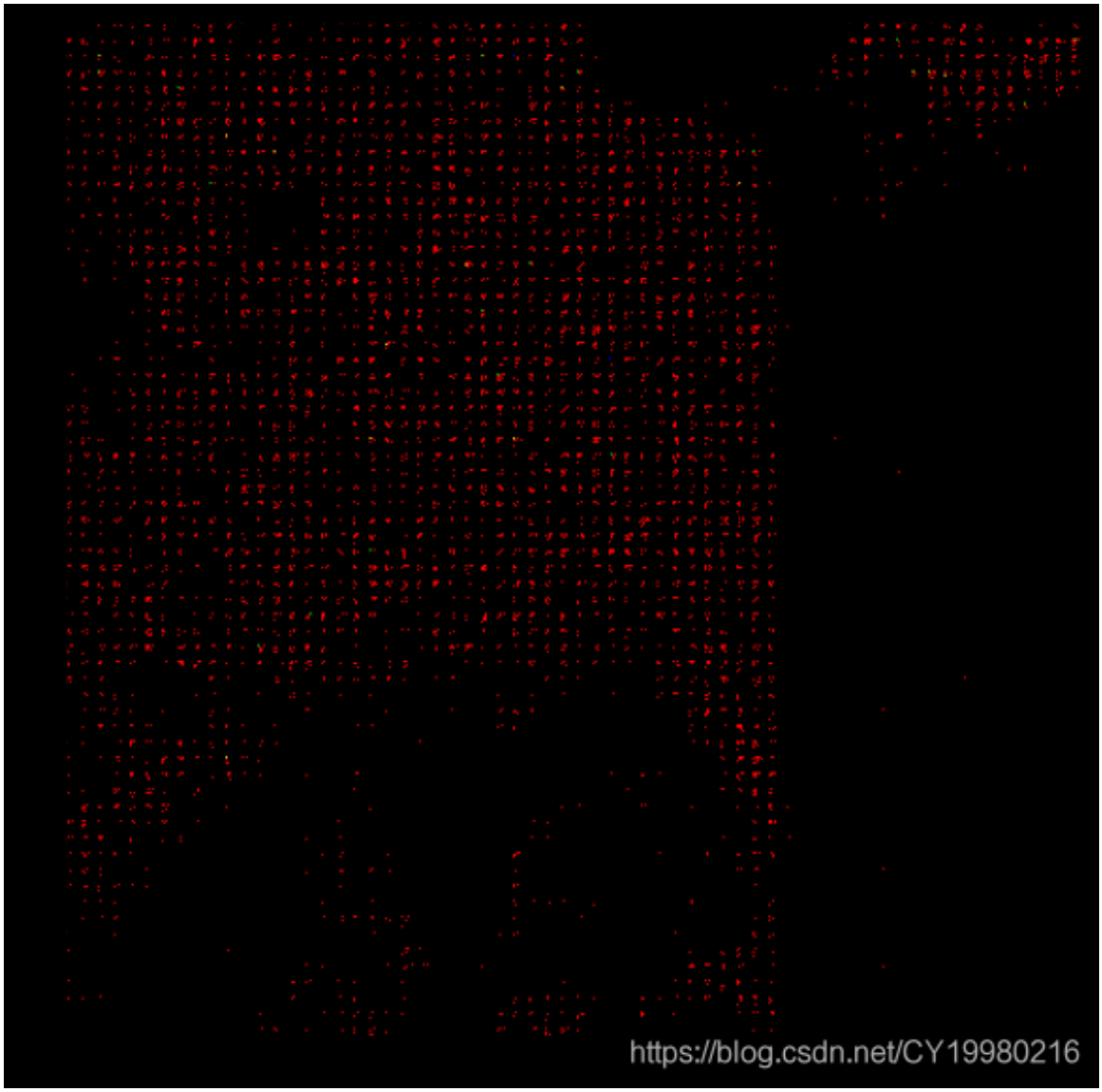
属性coeff_arrays给出图像YCbCr的所有三个通道的DCT系数，让我们绘制Cover和Encoded Image的DCT系数，并尝试查看是否存在任何差异

```
fig,ax = plt.subplots(4,4,figsize=(20,16))
plt.tight_layout()

for i,path in enumerate([cover_images_path[0],JUNIWARD_images_path[0],JMIPOD_images_path[0],UERD_images_pat

    image = Image.open(path)
    jpeg = jio.read(path)
    DCT_Y = jpeg.coef_arrays[0]
    DCT_Cr = jpeg.coef_arrays[1]
    DCT_Cb = jpeg.coef_arrays[2]


    ax[i,0].imshow(image)
    ax[i,0].set_title('Image')
    ax[i,1].imshow(DCT_Y)
    ax[i,1].set_title('Luminance')
    ax[i,2].imshow(DCT_Cb)
    ax[i,2].set_title('Cb:Chroma Blue')
    ax[i,3].imshow(DCT_Cr)
    ax[i,3].set_title('Cr:Chroma Red')
```

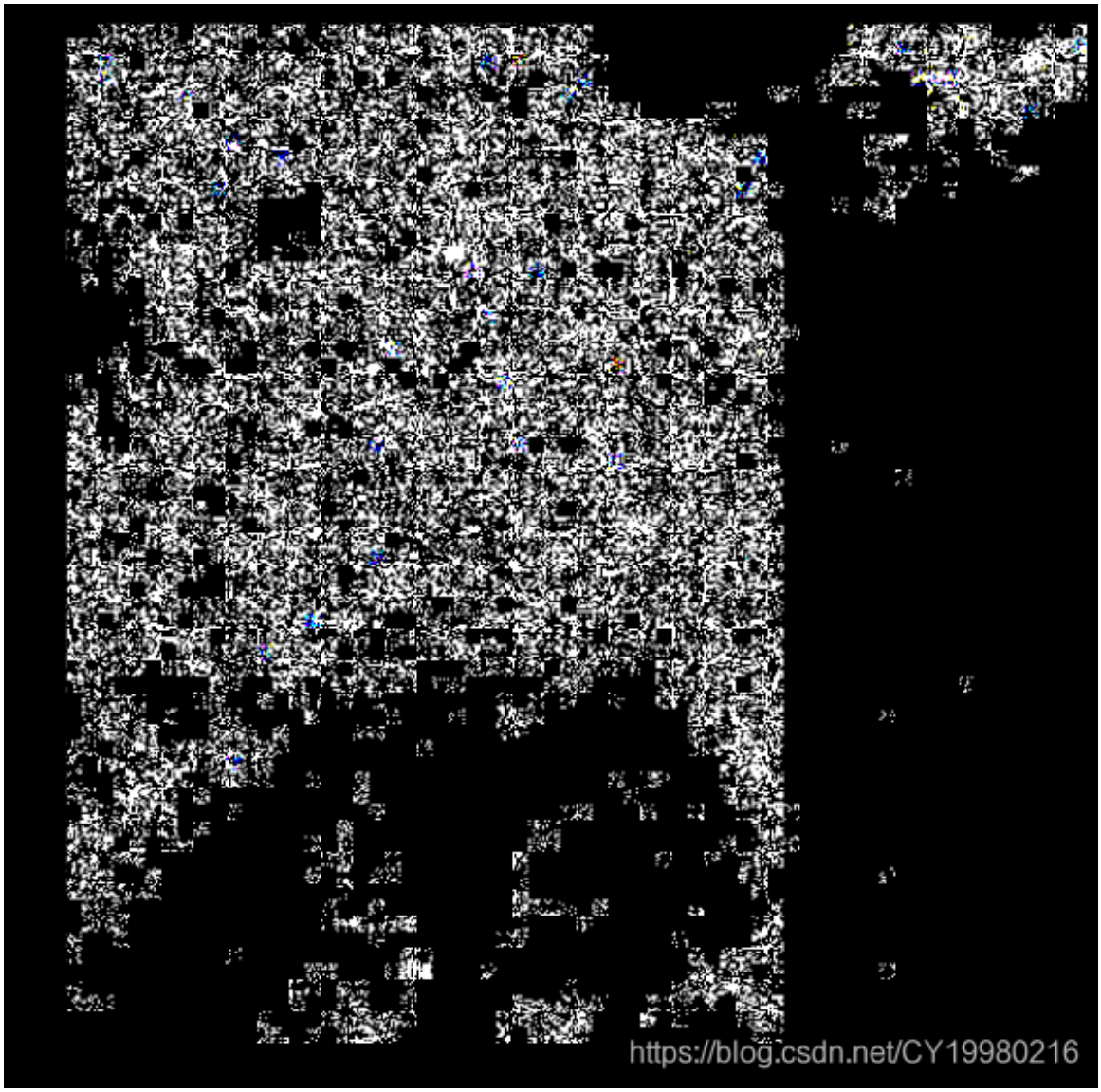


直到现在，我们都使用像素值和正常像素值之间的差异进行分析，并且有人认为这不是正确的方法。现在让我们来看一下DCT值的差异和像素的差异，看看这些差异，这将有助于我们 做出更好的决定

```
coverDCT = np.zeros([512,512,3])
stegoDCT = np.zeros([512,512,3])
jpeg = jio.read(cover_images_path[0])
stego_juni = jio.read(JUNIWARD_images_path[0])



coverDCT[:,:,0] = jpeg.coef_arrays[0] ; coverDCT[:,:,1] = jpeg.coef_arrays[1] ; coverDCT[:,:,2] = jpeg.coef
stegoDCT[:,:,0] = stego_juni.coef_arrays[0] ; stegoDCT[:,:,1] = stego_juni.coef_arrays[1] ; stegoDCT[:,:,2]

DCT_diff = coverDCT - stegoDCT
# So since they are not the same Images the DCT_diff would not be zero
print(len(DCT_diff[np.where(DCT_diff!=0)]))
print(np.unique(DCT_diff))
plt.figure(figsize=(16,10))
plt.imshow( abs(DCT_diff) )
plt.show()

# 5883
# [-1.  0.  1.]
```



```
coverPixels = np.array(Image.open(cover_images_path[0])).astype('float')
stegoPixels = np.array(Image.open(JUNIWARD_images_path[0])).astype('float')

pixelsDiff = coverPixels - stegoPixels

# So since they are not the same Images the pixels_diff would not be zero
print(len(pixelsDiff[np.where(pixelsDiff!=0)]))
print(np.unique(pixelsDiff))
plt.figure(figsize=(16,10))
plt.imshow( abs(pixelsDiff) )
plt.show()

# 161125
# [-4. -3. -2. -1.  0.  1.  2.  3.  4.  5.]
```

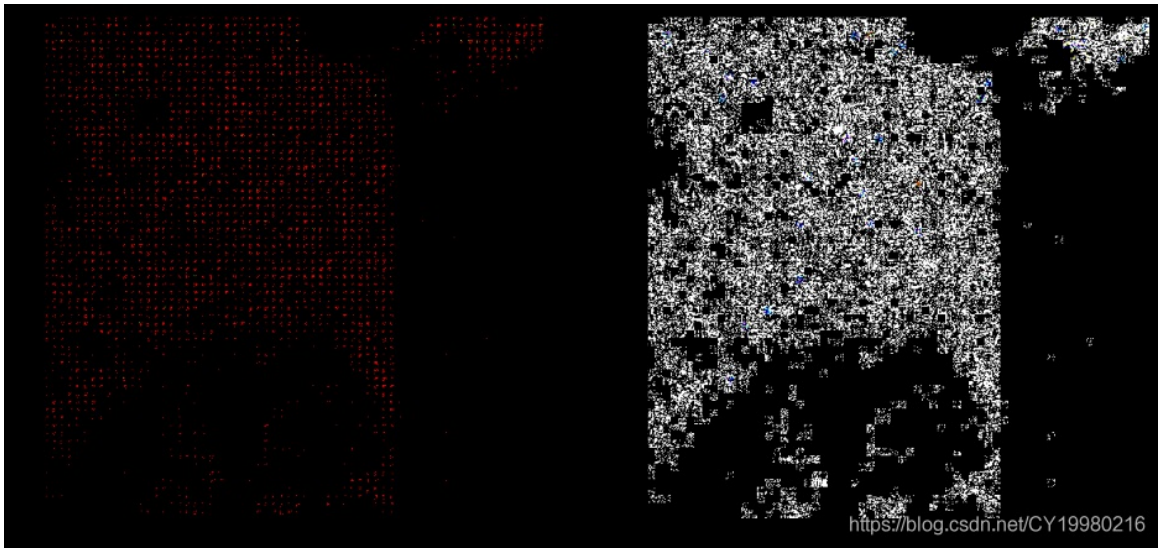


从这些图像很难推断出哪个效果更好，因为DCT和Pixel模式的图像都提供了相同的基础模式。 但是有人可以说像素值包含很多噪声，而DCT值则不包含这些噪声，可以为您提供更好的模型

但是从DCT和像素的非零差异值的数量中我们可以看到，Pixel具有比DCT更多的非零值，因此对于掩盖图像和隐身图像之间的区分更具区分性

我们并排查看图像以获得更好的底纹

```
fig,ax = plt.subplots(1,2,figsize=(16,12))
ax[0].imshow(abs(DCT_diff))
ax[1].imshow(abs(pixelsDiff))
```



下一步

1. 如我在笔记本的第8版中所讨论的，用像素差异和我的幼稚方式构建基线
2. 仅使用像素值和DCT值来构建基线，并查看哪种效果更好
3. 在决定使用SOTA模型实现完美的方法之后
4. 更多有助于我们的见解

如果您已阅读此内核，请分享您的想法，这是在以下注释中建模的更好方法
敬请关注！！！

分享学习，共同进步！