基于可逆九卷积变换的鲁棒盲水印嵌入

郭嘉

摘要：

小波变换是鲁棒水印嵌入的一种常用方法，在小波域嵌入水印只能选择四个域中的其中一个或几个进行嵌入，提供的选择太少，尽管多分辨率小波分析能够进一步的将小波域进行分解，提供更多嵌入选择，但这往往伴随着更大的计算量。深度学习的卷积核是进行特征提取的重要工具，但深度学习的卷积往往由于其后非线性激活函数模块不可逆，若要进行水印嵌入，必须通过重复训练一个解码器达到"伪可逆"效果，为了嵌入一个水印大费周折训练一个网络也不是我们想要的。本文希望结合以上两种方法的优点，提出基于可逆九卷积变换的鲁棒盲水印嵌入。

一、相关工作介绍以及它们与本文的联系

1.1 基于 DCT 变换的鲁棒水印嵌入

我们的方法并没有在 DCT 变换方面着手改进，但用到 DCT 变换作为我们方法的一个子模块，因此这里对一些重要的工作进行总结，这些工作的主要重心在于讨论在频域的哪个频率嵌入以及是否使用分块嵌入。

Cox 等人[1]对整个图像进行 DCT 操作，他们利用已有的对人视觉系统 HVS 的量化建模的研究，根据特定参数确定 DCT 域对视觉系统影响较大的频率，并在视觉系统感受明显的频率区域嵌入水印。Huang 等人直接在 DCT 直流位嵌入水印，这是因为他们认为：由于直流位数量一般很大,在这个频率嵌入水印更加鲁棒。还有一些研究[2][3][4][5]也是利用 HVS（人类视觉系统）的量化建模，仅在 DCT 频率中选择符合一定 HVS 指标的频率嵌入水印，使图像在嵌入水印后的人在视觉上感觉的变化不会过于明显。

1.2 基于 DWT 变换的鲁棒水印嵌入

我们的方法主要延伸 DWT 变换的思想，DWT 变换一般把图像变换到 4 个频域，多分辨的 DWT 变换取其中 1 个频域继续分解到四个更精细的频域。我们的方法有所不同，利用 9 个卷积核将图像变换到 9 个域。下面对基于 DWT 的鲁

棒水印嵌入的一些重要工作进行总结，这些工作的重心三点：一是是否使用多分辨率小波，二是在哪个小波域或哪些小波域嵌入，三是是否与 DCT 变换或 SVD 分解组合在一起使用。

Zhu 等人[6]使用多分辨小波进行水印嵌入，他们把水印全部嵌入到多个分辨率上的高频分量上，但他们没有考虑到这种方式对人类视觉系统的影响，Kaewkamnerd 和 Rao[7][8]则把这种影响考虑了进去。Raval 和 Rege[9]在 LL 和 HH 分量上水印，这是因为他们认为这两种频率分别对于一些攻击是鲁棒的：低频分量对于一些低通滤波攻击，有损压缩攻击以及一些几何形变是攻击十分鲁棒的，而高频分量对于亮度攻击，裁剪攻击等是鲁棒的，所以在这两个分量上同时嵌入水印能做到特别鲁棒。Tao 和 Eskicioglu[10]在所有四个分量，即 LL,HL,LH,HH 上都嵌入水印，但在这四个分量上嵌入水印的强度因子不同：在 LL 上嵌入的水印乘一个较大的因子，其他分量上水印乘的因子小些。Ganic and Eskicioglu[13]在进行 DWT 变换后，继续对一个分量作 SVD 分解，在对角矩阵中最大的奇异值上嵌入水印，与他们稍有不同，我们嵌入的是盲水印，他们嵌入的是非盲水印。

1.3 基于深度学习的鲁棒水印嵌入

由于卷积核常常与深度学习联系在一起，我们这里指出深度学习的卷积核数量往往是十万，百万的数量级，而我们这里仅仅使用 9 个卷积核，而且深度学习的卷积往往是不可逆的，因为他们采取降采样操作，且除了卷积还要考虑偏置以及非线性的激活，我们这里采用的是可逆的卷积操作，目的是为了嵌入水印后还能返回原域。尽管一些工作声称它们的卷积可逆，但这种可逆性是通过训练获得的。下面是一些基于深度学习的鲁棒水印嵌入，后面会提出对这种方法的一些质疑。

Zhu 等人[11]提出利用深度神经网络嵌入水印，他们先用一个 Encoder 编码器将水印嵌入图像，然后利用一个 decoder 解码器将嵌入了水印的图像。为了使得解码器能够更加鲁棒，中间有一个噪声层给已嵌入水印的图像添加噪声，还训练了一个对抗网络判断图像是否嵌入水印。Zhang 等人[12]给出了一篇在深度学习下的鲁棒水印的综述。

二、方案介绍

本文的主要创新点为图 1，本文将鲁棒水印中常用的小波变换变为卷积变换。注意这里我们不使用深度神经网络，和深度神经网络嵌入水印的技术也完全不同。本章节安排如下：2.1 节提出对利用深度神经网络嵌入水印的质疑，并指出我们的方法与其不同之处。2.2 节指出我们方法的优点。2.3 节回答如何找到卷积变换的逆变换。2.4 节回答如何用卷积对频率进行建模。2.5 节给出整体水印嵌入的流程。2.6 节给出另外一种卷积核选取方案，即正交卷积核，以此提供频率卷积核外的一种选择。由于过于繁杂的公式会使读者难以把握问题本质，因此本文的写作风格偏于直观解释，避免使读者陷入数学细节。

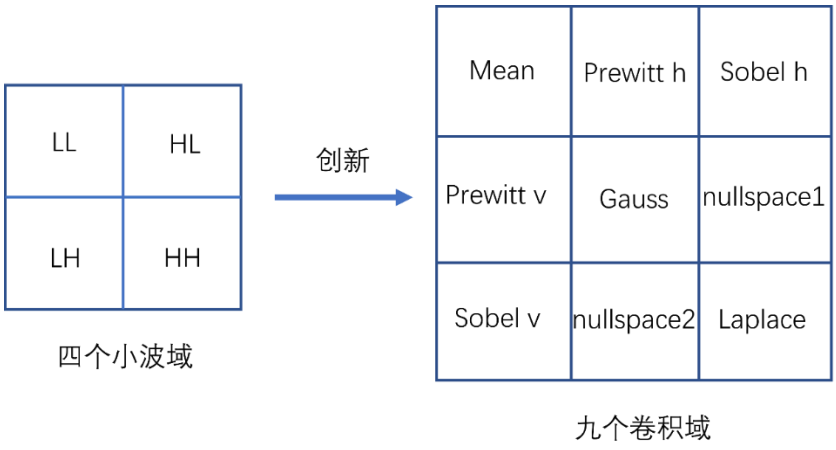


图 1 将小波变换变为卷积变换

2.1 对于用深度神经网络嵌入水印的质疑

尽管 Zhu 等人[11]提出利用深度神经网络嵌入水印，但利用深度神经网络嵌

入水印有以下几个缺点。

第一，时间成本高。神经网络嵌入所需的计算量太大，在现实场景下，如果需要同时对成千上万张图片进行水印嵌入来进行版权保护，而且计算资源比较受限，那么所需的时间是无法忍受的。

第二，也是最根本的一点，这种不具可解释性的嵌入不能作为版权认证手段。设想一个非法团伙训练了一个深度网络，这个网络的功能是对任何输入的图像，输出能认证该非法团伙的水印，这种要求对深度神经网络来说是十分简单的。那么每当该非法团伙受到侵犯版权的控告，他可以明目张胆地说："将你声称你具有版权的图片拿来，我用我这个深度网络直接可以提出我自己的水印"，那么这还有何意义？传统的方法则不同，非法团伙不可能利用可解释的水印提取方式从版权所有方的图像中提取出自己的水印。

第三，深度卷积网络的可逆性基于训练完成，并非严格可逆，本文的九卷积变换是严格可逆的。

2.2 利用可逆卷积核进行水印提取的好处

与深度神经网络需要训练一个 Decoder 来进行水印的提取不同。可逆卷积核仅仅使用逆矩阵就可完成卷积的逆操作。

相比四个小波域不同，本文有九个区域可以用来选择进行水印的嵌入，这提供了更多的嵌入选择。

另外本文提供一种权衡鲁棒性和嵌入容量的另一种选择，如果仅在一个卷积域上面嵌入水印，该方案的嵌入容量为在一个小波域上嵌入的 4/9，但显然这种方案具有更高的鲁棒性。

最后本文希望能够通过这种寻找可逆卷积核组的方法促进对卷积网络可逆

性的研究，即可逆性不是通过训练获得，而是通过严格的逆矩阵或逆变换获得。

上述的一些描述现在读来可能有些模糊，下面几节叙述相关细节。

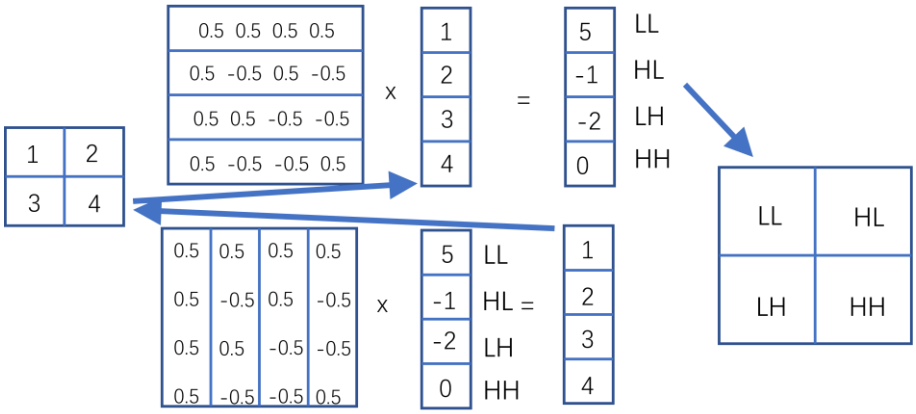


图 2 Haar 小波的矩阵解释

2.3 我们如何寻找可逆卷积核，如何在卷积后进行严格的逆变换？

我们知道，单个卷积操作会将图像中的 9 个像素变为 1 个像素，显然这种变换是不可逆的，当然不能用于水印嵌入。但想像我们有 9 个不同卷积核，那么就会将 9 个像素变为一条 9 维的向量，假如这 9 个卷积核是线性无关的，那么我们就可以通过这条 9 维的向量恢复原来 9 个像素。

这种思想其实与小波变换是相似的，我们以 Haar 小波为例，一个 4 像素的图像通过 Haar 小波变换变到四个频域 HH，HL，LH,LL，由于四个小波函数是线性无关的，所以我们可以仅根据 HH,HL,LH,LL 恢复原来图像。

为了更好地解释如何找到这九个卷积核的逆卷积操作，我们先观察小波变换的矩阵解释。图 2 给出了从矩阵角度解释 Haar 小波变换及其逆变换，这里直接将四个小波核展平成了四个行向量并堆叠成一个小波矩阵 H，对于[[1,2],[3,4]]的小波变换实际上就相当于将其变成列向量$[1,2,3,4]^T$,然后用 H 作用于它。为了对 LL,HL,LH,HH 进行逆小波变换，我们只需求 H 的逆矩阵，由于小波变换的特性，

H 为正交矩阵，我们求得其转置即可。

同样的，如图 3 为了求九个 3x3 的卷积核的逆变换，分别将九个卷积核分别展平成 1x9 的行向量，将其堆叠成 9x9 的矩阵，我们只需求得其逆矩阵即可，其逆矩阵的每行即为逆卷积核的展平形式。

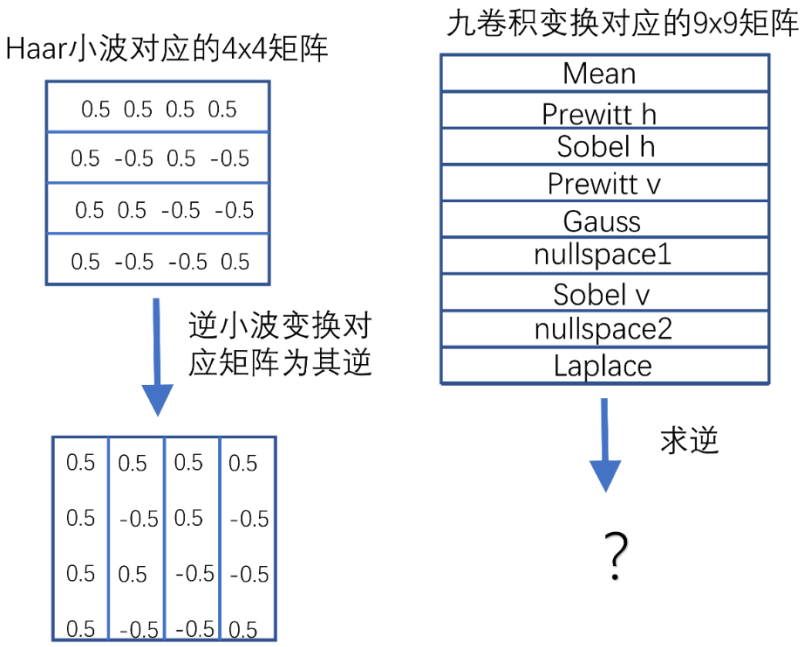


图 3

2.4 如何用卷积核来建模频率

我们看到图 3 中 9x9 矩阵中，除了 Mean 核外还有 prewitt 核，sobel 核,gauss 核和 laplace 卷积核。Prewitt 核和 sobel 核事实上起到了一阶差分的效果，他们有垂直与水平差分两种形式，可以与 HL 与 LH 类比。Laplace 核实际上起到水平二阶差分与垂直二阶差分的效果，可与 HH 类比。Gauss 核与 Mean 核则对应 LL 域。

在找到这 7 个卷积核后我们还需要两个与他们线性无关的卷积核，人工构造剩下两个与他们线性无关且可解释性强的卷积核是非常困难的，我们将利用图 4 所示方式寻找接下来的两个卷积核。此即找到这 7 个卷积核所展成线性子空间

的零空间的一组基。我们希望通过这样的方式使 9 个卷积核所提取的语义各自不同。

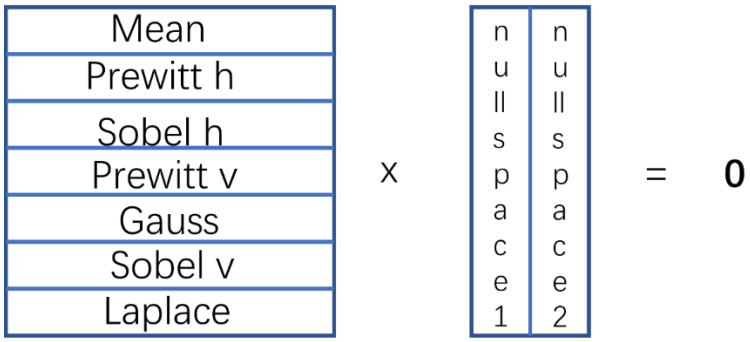


图 4

为了更方便读者理解，下面附上频域分解可逆九卷积的核生成代码，该代码在上传文件的 blind_watermark.py 中。

至此我们已经找到了如何找到 9 个能提取不同频率语义的卷积核及其逆变换的方法，这种方法与小波变换有异曲同工之妙，唯一的美中不足是这 9 个卷积核的展平形式并不是 R^9 的一组标准正交基，我们会在后面的一节提出另外一组卷积核作为备选方案，下面一节我们先描述水印嵌入的细节。

```python
#频域可逆九卷积核
def generate_kernels(self):

    from scipy.linalg import null_space
    mean=np.array([[1/9,1/9,1/9],[1/9,1/9,1/9],[1/9,1/9,1/9]])
    gauss=np.array([[1,2,1],[2,4,2],[1,2,1]])/16
    prewitt_h=np.array([[-1,-1,-1],[0,0,0],[1,1,1]])
    prewitt_v=np.array([[-1,0,1],[-1,0,1],[-1,0,1]])
    sobel_h=np.array([[-1,-2,-1],[0,0,0],[1,2,1]])
    sobel_v=np.array([[-1,0,1],[-2,0,2],[-1,0,1]])
    laplace_1=np.array([[0,1,0],[1,-4,1],[0,1,0]])
    #laplace_2=np.array([[-1,-1,-1],[-1,8,-1],[-1,-1,-1]])

    filter_mat=np.concatenate((mean.reshape(1,9),
                               gauss.reshape(1,9),
                               prewitt_h.reshape(1,9),
                               prewitt_v.reshape(1,9),
                               sobel_h.reshape(1,9),
                               sobel_v.reshape(1,9),
                               laplace_1.reshape(1,9),
                               ),axis=0)

    pseudo_filter=null_space(filter_mat).reshape(2,9)
    filters=np.concatenate((filter_mat,pseudo_filter),axis=0)
    kernels=[filter.reshape([3,3]) for filter in list(filters)]
    inv_kernels=[inv_filter.reshape([3,3]) for inv_filter in list(np.linalg.inv(filters))]

    return kernels,inv_kernels
```

2.5 盲水印嵌入总流程

本文的主要创新点是扩展已有的小波变换范式，建立使卷积可无信息损失地逆转，进而能凭借此进行鲁棒水印嵌入的方式，其余的模块则比较常规。为了避免重写一些比较基本的模块，将重点放在可逆九卷积变换，我们的代码基于下面的 github 开源仓库 https://github.com/guofei9987/blind_watermark，在其水印嵌入的流程中的小波变换替换为我们所提出的九卷积变换。

图 5 是该开源仓库进行水印嵌入的方式，首先对原图像进行小波变换，对 LL 域分块，在每个 4x4 子块上嵌入 1bit 的水印。具体来说，先对每个子块进行 DCT 变换，再进行 SVD 分解，将水印嵌入最大的奇异值（注意这里嵌入的是盲水印，具体算法简单，不再钻进这些细节），然后逆 SVD，逆 DCT，逆小波变换回去。为了方便读者阅读，我们这里的描述略去了一些不重要的细节，将重点放在可逆九卷积变换上。

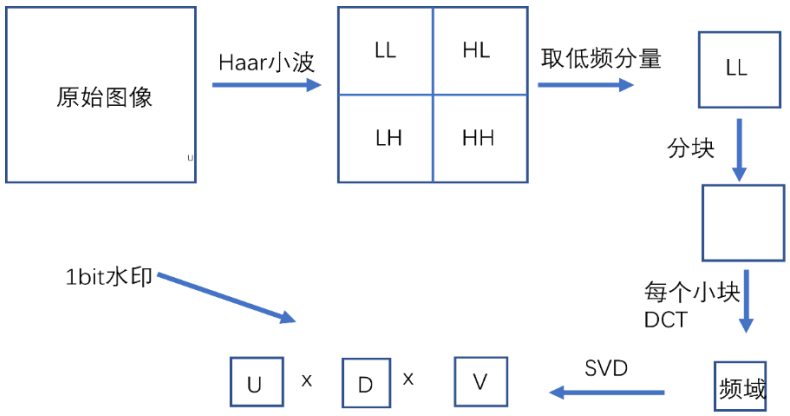


图 5

图 6 是本文提出的新型嵌入方式，我们将小波变换替换为本文提出的可逆九卷积变换，我们选取 Mean 域嵌入水印，剩下的流程与前述无异，所有的关键在于我们的九卷积变换可逆，这在前几节已有非常详尽的描述。

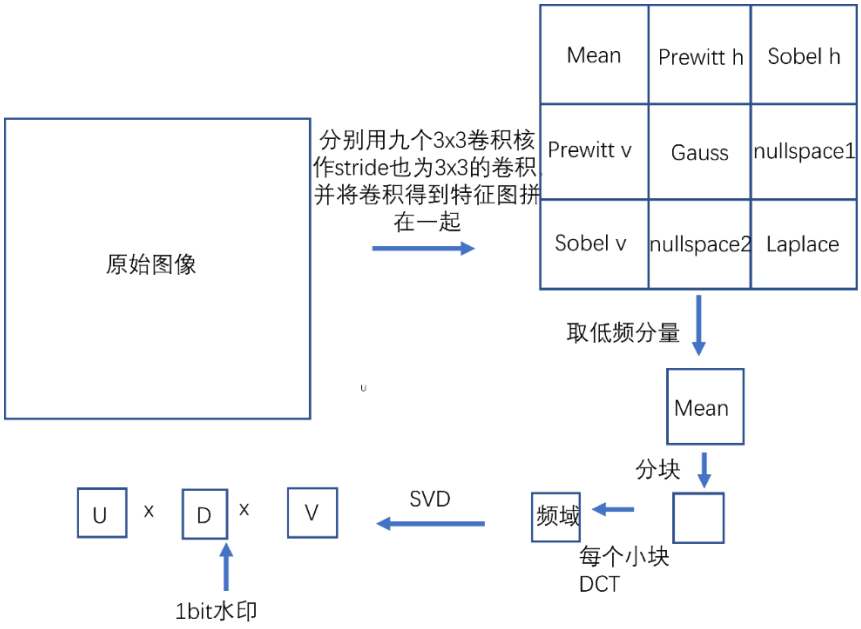


图 6

为了方便读者理解，下面附上九卷积变换与可逆九卷积变换的代码。这里函数名称是 generalize_dwt 和 generalize_idwt，主要目的是强调可逆九卷积变换是小波变换思想的一种延伸。

```
def generalize_dwt(self,img):
    h=img.shape[0]
    w=img.shape[1]
    feature=list(np.zeros((9,h//3,w//3)))
    for i in range(h//3):
        for j in range(w//3):
            block=img[i*3:(i+1)*3,j*3:(j+1)*3]
            for k,kernel in enumerate(self.kernels):
                feature[k][i][j]=(block*kernel).sum()
    return feature

def generalize_idwt(self,feature):
    h=feature[0].shape[0]
    w=feature[0].shape[1]
    img=np.zeros((3*h,3*w))
    for i in range(h):
        for j in range(w):
            feature_block=np.array([[feature[0][i][j],feature[1][i][j],feature[2][i][j]],
                                    [feature[3][i][j],feature[4][i][j],feature[5][i][j]],
                                    [feature[6][i][j],feature[7][i][j],feature[8][i][j]]])
            img[3*i:3*(i+1),3*j:3*(j+1)]=np.array([(feature_block*inv_kernel).sum() for inv_kernel in self.inv_kernels]).reshape(3,3)
    return img
```

2.6 另一种可逆九卷积变换

我们之前提到的九卷积变换对应的展平向量并没有进行单位化，同时他们也不是两两正交的，这是我们所不满意的地方，也是其稍微逊色于小波变换的小波基的地方，但为了模仿小波基的低频高频语义，我们不得不 prewitt,sobel

和 laplace 核考虑进来。

现在我们放弃考虑这种低频高频的语义，我们直接找九个 R^9 的标准正交基，当然也不能太随意，我们要求其中的一个向量为[1,1,1,1,1,1,1,1,1]/3,我们希望在这个向量对应的卷积域上嵌入水印，这是希望模仿 Mean 核的行为，剩下的向量如图 7 所示用 GramSchmit 正交化方法找到。

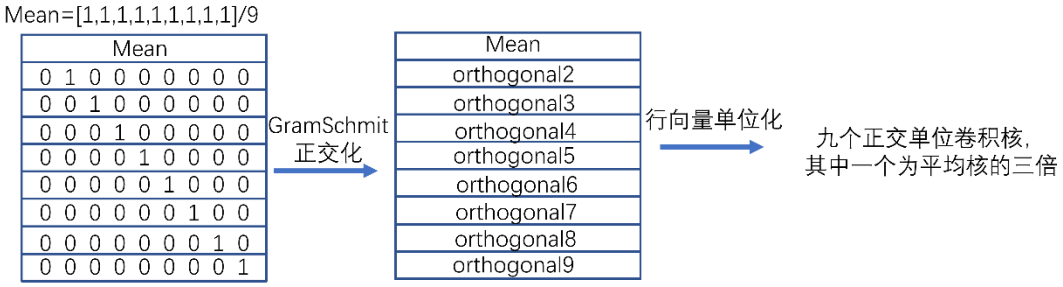


图 7

反思一下，这种方式保留了正交性，抛弃了频域的语义，但实际应用中似乎正交性已足够好用。

我们可以看到正交分解可逆九卷积中，平均核被扩大了三倍，这是因为（1/3）^{2}x9=1 为单位向量。这样的做法使得其对亮度攻击，椒盐攻击这样攻击不太鲁棒，但使得其对裁剪，遮挡等攻击更加鲁棒。这在后面的实验结果可以看到。

因此可以根据实际需要选取正交分解九卷积与频域分解九卷积，他们各有优劣，下面实验结果将更详尽地对此作出说明。

三、实验结果

3.1 原始图像和待嵌入水印以及嵌入容量分析

如图 8，原始图像为 1280x720 的灯塔图像,。如图 9，待嵌入水印为我的名字，分辨率为 68x45。我们的方法仅在 Mean 域嵌入水印，且在 4x4 的小分块

嵌入一个 bit，因此图像中每 9x4x4=144 个像素能嵌入 1 个 bit，嵌入容量可能不算太高，但由于这是鲁棒水印而不是数字隐写场景，我们认为鲁棒性的重要性比嵌入容量更高。可以看到，根据该嵌入容量分析，我们的名字水印可以嵌入到灯塔图像中。

从下面实验结果可以看出。从视觉质量上说，频域分解九卷积是稍微要差一些，你仔细看的话里面会有一些白色小点，应该是高频噪声。这是因为某些高频卷积核的矩阵范数过大或过小，不像小波变换每个频域核的范数都一样。从水印提取效果来说，正交分解九卷积效果也是更好，这是因为它将直流分量乘 3 再进行嵌入，从数值上说更容易准确提出盲水印。

图 8　原始图像

图 9 待嵌入水印

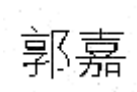
3.2 频域分解九卷积与正交分解九卷积水印嵌入与提取效果对比

频域分解九卷积嵌入后:

正交分解九卷积嵌入后:

频域分解九卷积提取水印:

正交分解九卷积提取水印:



3.3 频域分解九卷积与正交分解九卷积对不同攻击的鲁棒性对比

在下面的叙述中左边为正交分解九卷积结果，右边为频域分解九卷积结果。

1）亮度调低和亮度调高攻击

亮度调低攻击

亮度调高攻击

2）椒盐攻击

我们首先看看从 1）2）中我们能得出什么有用的结论。我们实际上在 2.6 中就已经预见了这种结果，即正交九卷积在面对亮度调低，亮度调高与椒盐攻击时是不如频域九卷积的。尽管他们都是在平均域，即直流域上嵌入唯一的区别在于正交九卷积为了保证向量为单位向量，系数从 1/9 变成了 1/3，即这是直流分量乘 3，这就使得其过分依赖于低频域的“整体”信息，从而难以抵挡这些攻击。

3）横向裁剪攻击和纵向裁剪攻击

横向裁剪攻击

纵向裁剪攻击

3）遮挡攻击

4）旋转攻击


我们再看从 3）4）能得出什么结论。这里正交分解九卷积在遮挡攻击和旋转攻击中表现得更为出色。我们认为这是因为正交分解九卷积实际上使得在每个卷积分量的分解更为平均，而频域分解九卷积核九卷积由于其卷积核的矩阵范数大小不一，因此在变换和逆变换的过程中由于与抵抗遮挡，旋转攻击的卷积核范数过小，导致这部分有用信息丢失了。

5）分辨率改变攻击


从 5）我们看出我们的两种方法都无法抵挡分辨率改变攻击，直观分析是由于分辨率攻击对每个 4x4 的分块改变太大，在分辨率变小的过程中，每个小分块的有用信息全部因为降采样丢失了。我们认为这不是我们的方法独有的问题，所有的基于分块的嵌入，我们觉得都很难抵挡这种攻击。

综合以上实验结果，给出以下结论，正交可逆九卷积变换对遮挡和旋转攻击更加鲁棒，而频域可逆九卷积变换对亮度变化攻击和椒盐攻击更加鲁棒。

四、总结

不论是频域可逆九卷积变换还是正交可逆九卷积变换都不能在所有攻击上做到比对方好，总有一些攻击是他们的弱点，这是因为他们仅在一个卷积域上进行水印的嵌入，然而我们提供了九个卷积域，我们完全可以找到一些域的组合，在这些域上都嵌入水印，使得我们的方法对许多攻击都达到鲁棒，这个留给未来工作完成。

五、致谢

感谢 github 开源仓库 https://github.com/guofei9987/blind_watermark 提供了一个扩展性高的盲水印开源仓库，使得我们可以专注于实现我们的可逆九卷积变换方法，而不费心去实现一些无关的细节和实现一些繁琐的代码。

六、关于代码以及相关文件

为方便读者在快速找到本文的创新点对应的代码，这里作出详细说明。原开源仓库在 blind_watermark/blind_watermark.py 中 read_img 函数中使用的是 dwt，本文使用的是 generalize_dwt，embed 函数中使用的 idwt 改为本文中使用的 generalize_idwt。这两个变换对应的卷积核生成利用 generate_kernels 函数予以实现，写在前面的是频域可逆九卷积，写在后面被注释掉的是正交可逆九卷积，如果要使用后面一种方法只需注释掉前面的函数，并取消后面函数的注释。本文的所有创新点体现在 blind_watermark/blind_watermark.py 中的第 208 行-第 277 行实现的四个函数。 examples/output 文件夹中的图片是正交分解九卷积的实验结果，examples/output_method2 文件夹中的图片是频域分解九卷积的实验结果。

参考文献

[1]I.J Cox, J. Kilian, F.T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia" in IEEE Transactions on Imnage Processing, vol. 6, no. 12, Dec.1997, pp:1673 - 1687

[2] Huang, J, Shi, YQ & Shi, Y 2000, 'Embedding Image Watermarks in DC Components', IEEE

Transactions on Circuits and System for Video Technology, vol. 10, no. 6, pp. 974-979.
[3] Tao, B., Dickinson, B., "Adaptive Watermarking in DCT Domain", in Proc. IEEE International Conference on Acoustics, $peech, and Signal Processing, ICASSP '97, 1997, Vol.4, pp. 1985-2988.
[4] Hsu, C-T., Wu, J-L., "Hidden Digital Watermarks in Images", in IEEE Transactions on Image Processing, vol. 8,nno. 1, pp. 56-68, 1999.
[5] Wong, P., H., W., Au, O., C., Wong, J., W., C., "Data Hiding and watermarking in JPEG Compressed Domain by DC Coefficient Modification", 2001. Available online www.ee.ust.hk/-eepeter/watermarkldchide.pdf Accessed on June 1, 2005.
[6] Zhu, W., Xiong, Z., and Zhang, Y.-Q., "Multiresolution Watermarking for Images and Video", in IEEE Trans. on circuit and Systemfor Video Technology, vol. 9, no. 4, pp. 545-550, June, 1999.
[7] Kaewkamnerd, N., Rao, K.R., "Multiresolution based image adaptive watermarking scheme", in EUSIPCO, Tampere, Finland, Sept. 2000. Available online www.ee.uta.edu/dip/paperfEUSIPCO water.pdf Accessed on June 1, 2005
[8] Kaewkamnerd, N., Rao, K.R., "Wavelet based image adaptive watermarking scheme" in IEE Electronics Letters, vol.36, pp.3 12-313, 17 Feb.2000
[9] Raval, M.S., Rege, P.P., "Discrete wavelet transforn based multiple watermarking scheme", Conference on Convergent Technologies for Asia-Pacific Region, TENCON 2003, vol. 3, pp. 935 - 938, 15-17 Oct. 2003
[10] Tao, P & Eskicioglu, AM 2004, 'A Robust Multiple Watermarking Scheme in the Discrete Wavelet Transform Domain', in Symposium on Internet Multimedia Management Systems V, Philadelphia, PA.
[11] iren Zhu, Russell Kaplan, Justin Johnson,and Li Fei-Fei. Hidden: Hiding data with deep networks. In ECCV, pages 657–672, 2018.
[12] Zhang C, Lin C, Benz P, et al. A brief survey on deep learning based data hiding, steganography and watermarking[J]. arXiv preprint arXiv:2103.01607, 2021.
[13] Ganic, E., Eskicioglu, A. M., "Robust digital watermarking: Robust DWT-SVD domain image waternarking: embedding data in all frequencies", Proceedings of the 2004 multimedia and security workshop on Mtultimedia and Security, September 2004, pp. 166 -174.