我们将使用Python语言，并依托OpenCV和PyDWT（小波变换）库来实现数字水印的嵌入和提取。这里选择基于离散小波变换（DWT）的数字水印技术，因为它在鲁棒性方面表现较好，能够抵抗一些常见的攻击。

步骤：

1. 水印嵌入：

- 读取原始图像和水印图像（这里水印图像为二值图像，如黑白Logo）。

- 对原始图像进行小波变换（通常使用一级分解），得到低频分量（LL）和高频分量（HL, LH, HH）。

- 将水印图像嵌入到低频分量中（因为低频分量包含图像的主要信息，嵌入水印后鲁棒性更强，但可能会影响图像质量，所以需要权衡）。

- 具体嵌入方法：将低频分量的部分系数与水印图像进行融合（例如，通过修改低频系数：LL' = LL + alpha \* W，其中alpha是嵌入强度，W是水印）。

- 然后进行小波逆变换得到嵌入水印的图像。

2. 水印提取：

- 对嵌入水印的图像进行小波变换，得到低频分量。

- 根据嵌入的逆过程提取水印

- 然而，在实际泄露检测中，我们可能无法获得原始图像，因此需要设计一个不需要原始图像的提取方法（非盲提取通常需要原始图像）。这里为了简化，我们先实现非盲提取，然后尝试实现盲提取

3. 鲁棒性测试：

- 对嵌入水印的图像进行各种攻击（翻转、平移、截取、调整对比度等）。

- 然后尝试从攻击后的图像中提取水印，并与原始水印比较相似度 注意： 由于鲁棒性测试需要，我们将实现以下攻击：

- 水平翻转

- 垂直翻转

- 平移（例如向右下角平移10个像素，并用黑色填充）

- 截取（裁剪部分图像）

- 调整对比度（增加或减少对比度）

- 高斯噪声

- JPEG压缩

我们将使用OpenCV进行图像处理，PyDWT进行小波变换。

安装依赖：

pip install opencv-python-headless numpy matplotlib pywt scikit-image

实现步骤：

1. 水印嵌入函数：embed\_watermark(original\_img, watermark\_img, alpha=0.1)

- 将原始图像和水印图像转换为合适的类型（浮点型）。

- 对原始图像进行小波分解（一级分解）。

- 将水印图像缩放到与低频分量相同的大小（因为低频分量是原图的1/4）。

- 将水印嵌入到低频分量：LL2 = LL1 + alpha \* watermark\_resized

- 进行小波逆变换，得到嵌入水印的图像。

2. 水印提取函数（非盲）：extract\_watermark(watermarked\_img, original\_img, alpha=0.1)

- 对原始图像和嵌入水印的图像分别进行小波分解。

- 提取水印：watermark\_extracted = (LL2\_w - LL1) / alpha

- 将提取的水印二值化（因为原始水印是二值的）。

3. 水印提取函数（盲提取）：这里我们尝试一种方法：在嵌入时，我们可能不需要原始图像，但提取时通常需要。然而，为了盲提取，我们可以将水印嵌入到高频分量？但是鲁棒性差。或者我们可以采用量化方法。但为了简单，我们这里先实现非盲提取，因为鲁棒性测试中，我们可能需要原始图像来提取水印。

4. 鲁棒性测试函数：对嵌入水印的图像进行攻击，然后提取水印，计算NC（归一化相关系数）或BER（误码率）来评估提取水印的质量。

对于鲁棒性测试，我们使用非盲提取。

步骤：

1. 读取原始图像和水印图像（水印为二值图像）。

2. 嵌入水印。

3. 对嵌入水印的图像进行攻击（攻击函数）。

4. 从攻击后的图像中提取水印（需要原始图像的低频分量）。

5. 计算原始水印和提取水印的相似度（NC）。

归一化相关系数（NC）计算公式：

NC = (Σ\_i (W\_i \* W'\_i)) / (sqrt(Σ\_i W\_i^2) \* sqrt(Σ\_i (W'\_i)^2))

其中W\_i是原始水印的像素值（二值0或1），W'\_i是提取水印的像素值（可能是连续的，所以我们需要将其二值化后再计算？或者直接计算？）。由于提取的水印是连续的，我们可以不二值化直接计算，但为了公平，我们通常将提取的水印二值化（例如，大于0.5则为1，否则为0）后再与原始水印比较。

但是，由于攻击的影响，提取的水印可能已经变形，所以我们可以用NC来评估连续水印的相似性，也可以使用二值化后的水印计算NC。

我们这里采用：先对提取的水印进行归一化（使其均值为0，方差为1）？或者直接计算原始二值水印和提取水印（连续）的NC？这里我们按照通常做法：将提取的水印缩放到[0,1]区间，然后计算NC（不二值化），因为二值化可能会丢失信息。

然而，在嵌入时水印是二值的，提取的水印理论上也是二值的（但攻击后可能不是）。所以，我们可以将提取的水印进行二值化（阈值0.5），然后计算二值图像之间的NC。但这样可能不够鲁棒，因为攻击可能导致二值化阈值变化。

因此，我们采用不二值化，直接计算原始水印（0和1）和提取水印（连续值）的NC。注意：原始水印是0和1，提取水印是连续的，但是NC公式可以处理这种情况。

但是，为了简单，我们也可以使用另一种指标：比特错误率（BER）。但我们这里先计算NC。

实现：

1. 嵌入函数

2. 提取函数（非盲）

3. 攻击函数

4. 测试函数

为了简化，我们只处理灰度图像。如果是彩色图像，我们将其转换为灰度图。

因此，我们修改：

- 对于彩色图像，转换为YUV，然后对Y通道进行嵌入，UV保持不变。

- 对于水印提取，同样处理Y通道。

开始编码：

步骤：

1. 读取原始图像和水印图像。

2. 如果原始图像是彩色，则转换为YUV，取Y通道；否则直接使用。

3. 水印图像需要是二值图像，如果不是，则二值化（例如，阈值127）。

4. 嵌入水印到Y通道（或灰度图像）。

5. 如果是彩色，将嵌入后的Y通道与UV合并，再转回RGB；如果是灰度，则直接输出。

提取时：

1. 将原始图像（未加水印）同样处理（得到Y通道或灰度图像）。

2. 将含水印图像（攻击后）同样处理（得到Y通道或灰度图像）。

3. 从Y通道中提取水印（非盲）。

4. 将提取的水印二值化（可选，用于显示）或者直接评估。

我们首先实现灰度图像的情况，然后扩展到彩色图像。

输出结果：



