Uformer: A General U-Shaped Transformer for Image Restoration

黄嘉诚

摘要

在本文中，我们提出了 Uformer，一种有效且高效的基于 Transformer 的图像恢复架构，其中我们使用 Transformer 块构建了分层编码器-解码器网络。在 Uformer 中，有两个核心设计。首先，我们引入了一种新颖的局部增强窗口（LeWin）Transformer 块，它基于非重叠窗口的自注意力，而不是全局自注意力。它在捕获局部上下文的信息的同时显著降低了高分辨率特征图的计算复杂度。其次，我们提出了一种可学习的多尺度恢复调制器，以多尺度空间偏差的形式来调整 Uformer 解码器的多个层中的特征。我们的调制器展示了在各种图像恢复任务中恢复细节的卓越能力，然而只引入了少量的额外参数和计算成本。在这两种设计的支持下，Uformer 拥有强大的能力来捕获局部和全局的特征去进行图像恢复。此外，我们在多个图像恢复任务上进行了广泛的实验来进行评估，包括图像去噪、运动去模糊、散焦去模糊和去雨。与最先进的算法相比，我们的 Uformer 只需简单的架构即可实现最卓越的性能。

关键词：图像恢复；注意力机制；多尺度恢复调制器；局部增强前馈神经网络

1 引言

随着消费和工业相机以及智能手机的快速发展，消除图像中不需要的退化（例如噪声、模糊、雨水等）的要求不断增长。把不清晰的原始图像恢复成真实图像，即图像恢复，是计算机视觉中的一项经典任务。近年来最先进的方法 [2] 大多是基于卷积神经网络的，虽然这些方法的结果十分出众，但是在捕获长期依赖方面依然有很大的限制。为了解决这个问题。我们提出了一种有效且高效的基于 Transformer 的图像恢复结构 Uformer。Uformer 基于 UNet ，将卷积层修改为 Transformer 块，同时保持相同的整体分层编码器-解码器结构和跳跃连接。为了使 Uformer 适合图像恢复任务，我们提出了两个核心设计。首先，我们提出了局部增强窗口（LeWin）Transformer 块，这是一个高效且有效的基本组件。并且在 Transformer 块中前馈网络的两个全连接层之间我们引入深度卷积层 [3]，以更好地捕获局部上下文。

其次，我们提出了一种可学习的多尺度恢复调制器来处理各种图像退化问题。调制器被表述为多尺度空间偏置，以调整 Uformer 解码器多层的特征。具体来说，就是将可学习的基于窗口的张量添加到每个 LeWin Transformer 块中的特征中，以适应特征恢复更多细节。得益于简单的操作和基于窗口的机制，它可以灵活地应用于不同框架中的各种图像恢复任务。

基于上述两种设计，我们简单的 Uformer 结构在多个图像恢复任务上取得了最好的效果。

2 相关工作

2.1 图像恢复架构

图像恢复的目的是从退化的图像中恢复干净的图像。一种流行的解决方案是使用具有跳跃连接的 U 形结构模型，以分层捕获多尺度信息执行各种图像恢复任务，包括图像去噪 **4**，去模糊。一些图像恢复方法的灵感来自图像分类快速发展，例如：基于 ResNet 的结构已广泛用于一般图像恢复以及图像恢复中的特定任务，例如超分辨率和图像去噪。

直到最近，一些工作 [4–6] 开始探索注意力机制来提高性能。例如，挤压和激励网络和非局部神经网络激发了用于不同图像恢复任务的方法分支，例如超分辨率、去雨、去噪、去模糊、去阴影等等。我们的 Uformer 还应用分层结构来构建多尺度特征，同时使用新推出的 LeWin Transformer 块作为基本构建块。

2.2 视觉 Transformers

Transformer 在自然语言处理方面表现出了显著的性能。与 CNN 的设计不同，基于 Transformer 的网络结构擅长通过全局自注意力捕获数据中的长程依赖关系。计算机视觉研究人员受到了 Transformer 的启发 [7]，视觉 Transformers 的开创性工作直接在中等尺寸（16×16）扁平补丁上训练纯基于 Transformer 架构的模型。通过大规模数据预训练，与最先进的 CNN 相比 [8]，视觉 Transformers 在图像分类方面获得了优异的结果。

自从视觉 Transformers 出现以来，人们做出了许多努力来降低全局自注意力的二次计算成本 [9]，为了使 Transformer 更适合视觉任务。一些工作 [10,11] 专注于建立类似于卷积结构的层次 Transformer。为了克服原始自注意力的二次复杂度，通过窗口移位在局部窗口上进行自注意力以帮助不同窗口进行交互，并获得良好的结果。

除了高级的分类任务之外，还有一些工作 [12,13] 基于 Transformer 来用于生成任务，虽然 Transformer 在视觉领域已经进行了大量的探索，但将 Transformer 引入低层视觉的工作 [14–16] 依然十分少。早期工作利用自注意力机制来学习纹理以实现超分辨率 [17,18]，对于图像恢复任务，IPT [19] 首先在多任务学习框架内应用标准 Transformer 块。然而，IPT 依赖于大规模合成数据集的预训练和多任务学习。相比之下，我们设计了一种基于通用 U 形 Transformer 的结构，事实证明该结构对于图像恢复来说是高效且有效的。

3 本文方法

在本节中，我们首先描述 Uformer 用于图像恢复的整体流程和层次结构。然后，我们提供了 Uformer 的基本组件 LeWin Transformer 块的详细信息。之后，我们提出了多尺度恢复调制器。

3.1 本文方法概述

如图 1所示，所提出的 Uformer 的整体结构是一个 U 形分层网络，编码器和解码器之间具有跳跃连接

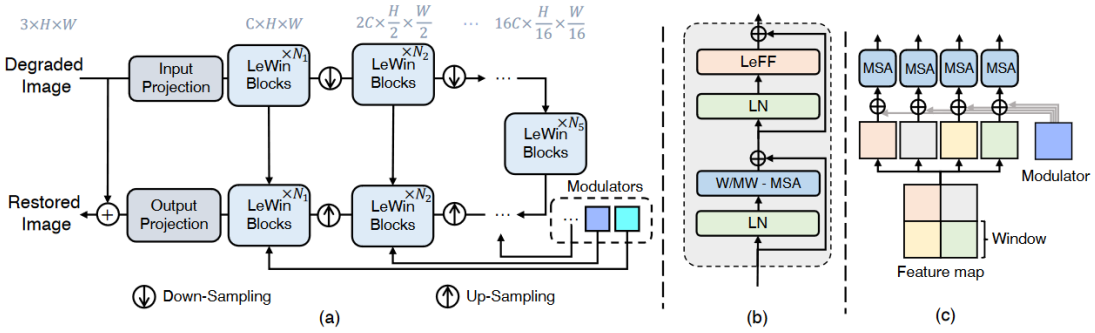


图 1. (a) Uformer 模型结构概述。(b) LeWin Transformer 块。(c) 说明调制器如何调制每个 LeWin Transformer 块中的 W-MSA(在 (b) 中称为 MW-MSA)

具体来说，对于一个给定的退化图像 $\mathbf{I} \in \mathbb{R}^{3\times H\times W}$ ，Uformer 首先应用带有 LeakyReLU 的 3×3 卷积层来提取低级特征 $\mathbf{X}_0 \in \mathbb{R}^{C\times H\times W}$ 。接下来，根据 U 形结构的设计，特征图 $\mathbf{X}_0$ 总共经过 K 个编码阶段。每个阶段都包含一系列我们提出的 LeWin Transformer 块和一个下采样层。LeWin Transformer 块利用自注意力机制来捕获长期依赖关系，并且还通过特征图上的非重叠窗口使用自注意力来降低计算成本。在下采样层中，我们首先将展平的特征重塑为 2D 空间特征图，然后对图进行下采样，使用步幅为 2 的 4×4 卷积将通道加倍。例如对于给定的输入特征图 $\mathbf{X}_0 \in \mathbb{R}^{C\times H\times W}$，经过 l 次编码器编码后产生的特征图为 $\mathbf{X}_l \in \mathbb{R}^{2^l C\times \frac{H}{2^l}\times \frac{W}{2^l}}$ 。

然后，在编码器的末尾阶段添加带有 LeWin Transformer 块栈的瓶颈阶段。在这个阶段，由于分层结构，Transformer 块能捕获更长的依赖关系（当窗口大小等于特征图大小时甚至是全局的）依赖关系。

对于特征重建，所提出的解码器也同样包含 K 个阶段。每个层都包含一个上采样层和一堆类似于编码器的 LeWin Transformer 块。我们使用步长为 2 的 2×2 转置卷积进行上采样。该层减少了一半的特征通道，并使特征图的大小加倍。之后，输入到 LeWin Transformer 块的特征是上采样特征和来自编码器的相应特征通过跳跃连接的串联。接下来，利用 LeWin Transformer 块来学习恢复图像。在 K 个解码器阶段之后，我们将展平的特征重塑为 2D 特征图，并应用 3×3 卷积层来获得残差图像 $\mathbf{R} \in \mathbb{R}^{3\times H\times W}$。最后，通过 $\mathbf{I}^{'} = \mathbf{I} + \mathbf{R}$ 获得恢复图像。我们使用 Charbonnier 损失来训练 Uformer ：

$$(\mathbf{I}^{'}, \mathbf{I}) = \sqrt{\left\|\mathbf{I}^{'} - \mathbf{I}\right\|^2 + \varepsilon^2} \tag{1}$$

其中 I 是真实图像，$\varepsilon = 10^{-3}$ 是所有实验中的常数。

3.2 局部增强窗口 Transformer 块

应用 Transformer 进行图像恢复有两个主要挑战。首先，标准 Transformer 架构在所有令牌之间全局计算自注意力，对应的计算复杂度是令牌数量的平方。不适合在高分辨率特征图上应用全局自注意力。其次，局部上下文信息对于图像恢复任务至关重要，因为可以利用退化像素的邻域来恢复其干净版本，但之前的工作表明 Transformer 在捕获局部依赖性方面表现出局限性。为了解决上述两个问题，我们提出了一个局部增强的窗口 (LeWin) Transformer 块，如图 1中 (b) 所示，它受益于 Transformer 中的自注意力来捕获远程依赖关系，并且还涉及将卷积运算符输入到 Transformer 中以捕获有用的局部上下文。具体来说，使用了两个核心

设计构建该块：(1) 非重叠的基于窗口的多头自注意力 (W-MSA) 和 (2) 局部增强的 Feed-前向网络（LeFF），局部增强窗口的 Transformer 块的计算公式如下:

$$\begin{aligned} X_l^{'} &= W - MSA(LN(X_{l-1})) + X_{l-1} \\ X_l &= LeFF(LN(X_l^{'})) + X_l^{'} \end{aligned} \tag{2}$$

其中 $X^{'}, X_l$ 分别是 W-MSA 模块和 LeFF 模块的输出，LN 表示层归一化。

基于窗口的多头自注意力 (W-MSA) 即在不重叠的局部窗口内执行自注意力计算，通过这个方法可以显著降低计算成本，对于给定的二维特征图 $X\epsilon\mathbb{R}^{C\times H\times W}$，其中的 H 和 W 为图的高度和宽度，我们将 X 分割为不重叠的窗口，窗口的大小为 M×M，然后从每个窗口 i 得到展平和转置之后的特征 $X^i\epsilon\mathbb{R}^{M^2\times C}$。然后就可以对每个窗口中扁平化特征进行自注意力。若注意力头的数量为 K，则每个头的维度是 $d_k = C/k$，第 K 个头的自注意力的计算表示如下

$$\begin{aligned} &X = \{X^1, X^2, \cdots, X^N\}, N = HW/M^2 \\ &Y_k^i = Attention(X^iW_k^Q, X^iW_k^K, X^iW_k^V), i = 1\cdots, N \\ &\hat{X}_k = \{Y_k^1, Y_k^2, \cdots, Y_k^N\} \end{aligned} \tag{3}$$

其中 $W_k^Q, W_k^K, W_k^V\epsilon\mathbb{R}^{C\times d_k}$ 分别代表第 k 个头的查询，键和值的投影矩阵，$\hat{X}_k$ 是第 k 个头的输出，最后将所有头的输出连接起来，线性投影之后就能获得最终的输出。我们的 W-MSA 与全局注意力相比，可以显著的降低计算成本，对于给定的特征图 $X\epsilon\mathbb{R}^{C\times H\times W}$，计算复杂度由原来的 $O(H^2W^2C)$ 降低到 $O(M^2HWC)$。

局部增强前馈网络 (LeFF) 正如之前的作品所指出的，标准 Transformer 中的前馈网络（FFN）利用本地上下文的能力有限。实际上，相邻像素是图像恢复的重要参考。为了克服这个问题，在基于 Transformer 的结构中的 FFN 中添加了一个深度卷积块。如图 2 所示，我们首先对每个 token 应用线性投影层以增加其特征维度。接下来，我们将标记重塑为 2D 特征图，并使用 3 × 3 深度卷积来捕获局部信息。然后，我们将特征展平为标记，并通过另一个线性层缩小通道以匹配输入通道的尺寸。我们使用 GELU 作为每个线性/卷积层之后的激活函数。

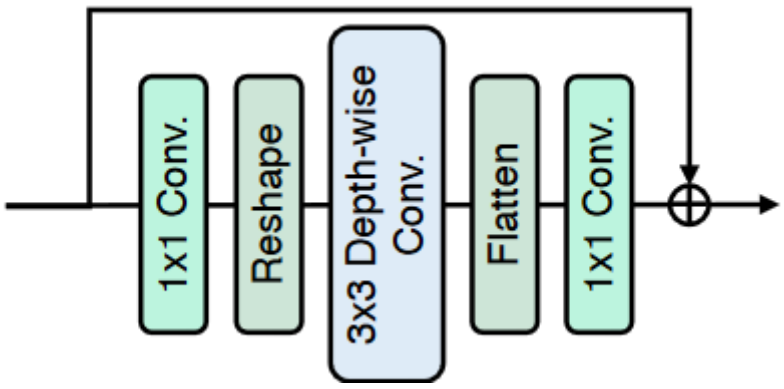


图 2. Caption

3.3 多尺度恢复调制器

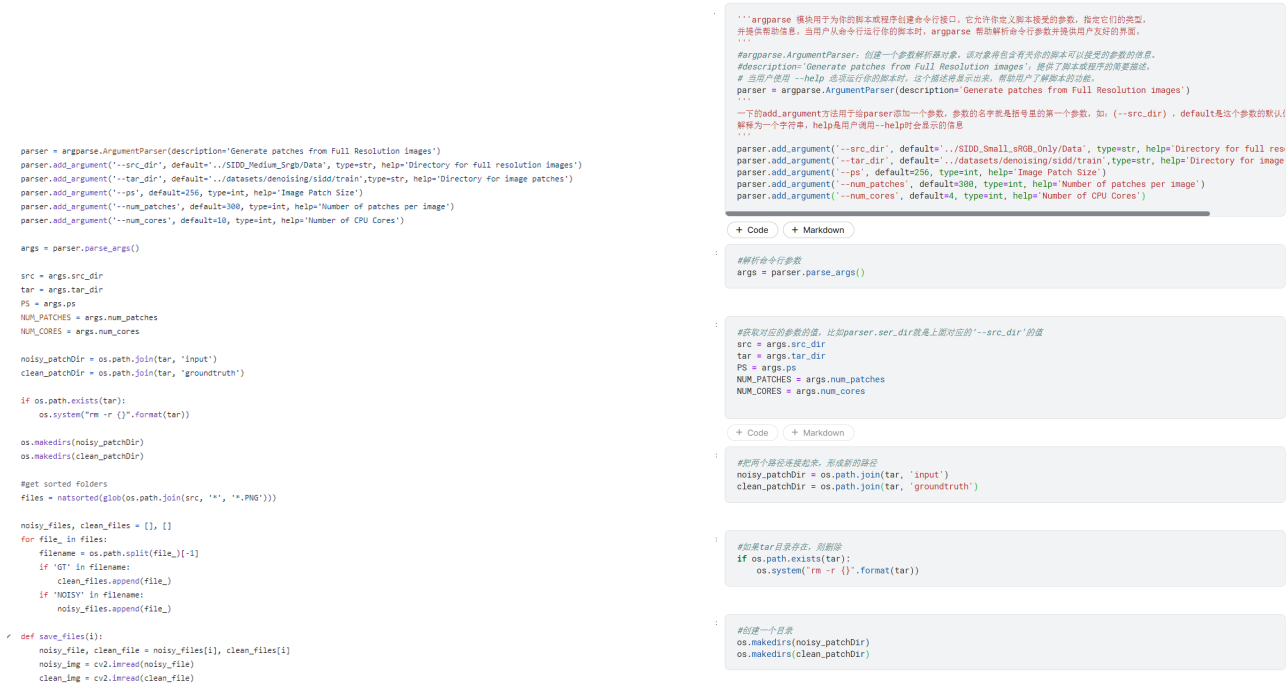
不同类型的图像退化（例如模糊、噪声、雨等）都有自己独特的扰动模式需要处理或恢复。为了进一步提高 Uformer 处理各种扰动的能力，提出了一种轻量级多尺度恢复调制器来

校准特征并鼓励恢复更多细节。如图 1(a) 和 1(c) 所示，多尺度恢复调制器在 Uformer 解码器中应用多个调制器。特别是在每个 LeWin Transformer 块中，调制器被公式化为形状为 M × M × C 的可学习张量，其中 M 是窗口大小，C 是当前特征图的通道维度。每个调制器作为一个共享偏置项，在自注意力模块之前添加到所有非重叠窗口中。由于这种轻量级加法运算和窗口大小的形状，多尺度恢复调制器引入了边际额外参数和计算成本，多尺度恢复调制器有利于图像去模糊和图像去噪，并且可以消除更多的运动模糊和噪声模式，并产生更清晰的图像。一种可能的原因是，在解码器的每个阶段添加调制器可以灵活调整特征图，从而提高恢复细节的性能。这与之前的工作 StyleGAN [26] 一致，使用多尺度噪声项添加到卷积特征中，实现随机变化以生成逼真的图像。

4 复现细节

4.1 与已有开源代码对比

复现过程参考了官方开源代码（https://github.com/ZhendongWang6/Uformer），本次复现将 Script 模式的代码在 Kaggle 平台上改成了 Notebook 格式，并对每一处代码进行了详细的注释，学习者可以 Kaggle 上访问，逐步运行代码，观察每一步的输入输出效果。



```
parser = argparse.ArgumentParser(description='Generate patches from Full Resolution images')
parser.add_argument('--src_dir', default='../SIDD_Medium_Srgb/Data', type=str, help='Directory for full resolution images')
parser.add_argument('--tar_dir', default='../datasets/denoising/sidd/train',type=str, help='Directory for image patches')
parser.add_argument('--ps', default=256, type=int, help='Image Patch Size')
parser.add_argument('--num_patches', default=300, type=int, help='Number of patches per image')
parser.add_argument('--num_cores', default=10, type=int, help='Number of CPU Cores')

args = parser.parse_args()

src = args.src_dir
tar = args.tar_dir
PS = args.ps
NUM_PATCHES = args.num_patches
NUM_CORES = args.num_cores

noisy_patchDir = os.path.join(tar, 'input')
clean_patchDir = os.path.join(tar, 'groundtruth')

if os.path.exists(tar):
    os.system("rm -r {}".format(tar))

os.makedirs(noisy_patchDir)
os.makedirs(clean_patchDir)

#get sorted folders
files = natsorted(glob(os.path.join(src, '*', '*.PNG')))

noisy_files, clean_files = [], []
for file_ in files:
    filename = os.path.split(file_)[-1]
    if 'GT' in filename:
        clean_files.append(file_)
    if 'NOISY' in filename:
        noisy_files.append(file_)

def save_files(i):
    noisy_file, clean_file = noisy_files[i], clean_files[i]
    noisy_img = cv2.imread(noisy_file)
    clean_img = cv2.imread(clean_file)
```

(a) 部分官方开源代码示例

```
'''argparse 模块用于为你的脚本或程序创建命令行接口，它允许你定义脚本接受的参数，指定它们的类型，
并提供帮助信息。当用户从命令行运行你的脚本时，argparse 帮助解析命令行参数并提供用户友好的界面。
'''
#argparse.ArgumentParser：创建一个参数解析器对象，该对象将包含有关你的脚本可以接受的参数的信息。
#description='Generate patches from Full Resolution images'：提供了脚本或程序的简要描述，
# 当用户使用 --help 选项运行你的脚本时，这个描述将显示出来，帮助用户了解脚本的功能。
parser = argparse.ArgumentParser(description='Generate patches from Full Resolution images')
'''
一下的add_argument方法用于给parser添加一个参数，参数的名字就是括号里的第一个参数，如：('--src_dir') ，default是这个参数的默认值
解释为一个字符串，help是用户调用--help时会显示的信息
'''
parser.add_argument('--src_dir', default='../SIDD_Small_sRGB_Only/Data', type=str, help='Directory for full res
parser.add_argument('--tar_dir', default='../datasets/denoising/sidd/train',type=str, help='Directory for image
parser.add_argument('--ps', default=256, type=int, help='Image Patch Size')
parser.add_argument('--num_patches', default=300, type=int, help='Number of patches per image')
parser.add_argument('--num_cores', default=4, type=int, help='Number of CPU Cores')

+ Code + Markdown

#解析命令行参数
args = parser.parse_args()

#获取对应的参数的值，比如parser.ser_dir就是上面对应的'--src_dir'的值
src = args.src_dir
tar = args.tar_dir
PS = args.ps
NUM_PATCHES = args.num_patches
NUM_CORES = args.num_cores

+ Code + Markdown

#把两个路径连接起来，形成新的路径
noisy_patchDir = os.path.join(tar, 'input')
clean_patchDir = os.path.join(tar, 'groundtruth')

#如果tar目录存在，则删除
if os.path.exists(tar):
    os.system("rm -r {}".format(tar))

#创建一个目录
os.makedirs(noisy_patchDir)
os.makedirs(clean_patchDir)
```

(b) 我的工作代码部分示例

图 3. 与开源代码对比

4.2 输入映射方式改进代码

原文的 Input Projection 和 Output Projection 使用了一个 3*3 的卷积输出后的特征图进行输入数据的格式化，这里我们改成不采用卷积，直接将原图信息拓展输入和输出。对比如下：

Listing 1: 改进后的代码

```
def forward(self,x):
    # 获取特征图的形状
    B, C, H, W = x.shape
    # 将特征图重塑为 B x (H * W) x C
    reshaped_feature_map = x.transpose((0, 2, 3, 1)).reshape((B, H
        * W, C))
     if self.norm is not None:
            x = self.norm(x)
    return reshaped_feature_map
```

Listing 2: 原始代码

```
    def forward(self, x):
        B, C, H, W = x.shape    #B是批量大小
        #原本是(B,C,H,W)经过flatten(2)后变为(B,C,H*W)
        x = self.proj(x).flatten(2).transpose(1, 2).contiguous()  #
            B H*W C
        if self.norm is not None:
            x = self.norm(x)
        return x
```

4.3 位置编码方式改进代码

原文的采用了相对位置进行编码，这里我们改成采用 2D 位置进行编码，对比如下：

Listing 3: 改进后代码

```
# 2D位置编码 layer
        self.positional_encoding = nn.Parameter(torch.randn(1,
            image_size, image_size, head_dim))

    def forward(self, x, attn_kv=None, mask=None):
        B_, N, C = x.shape
        q, k, v = self.qkv(x,attn_kv)
        q = q * self.scale
        attn = (q @ k.transpose(-2, -1))    #计算注意力分数矩阵
        #增加2D位置编码
        attn = attn + self.positional_encoding[:, :H, :W, :].to(x.
            device)
```

Listing 4: 原始代码

```python
    def forward(self, x, attn_kv=None, mask=None):
        B_, N, C = x.shape
        q, k, v = self.qkv(x,attn_kv)
        q = q * self.scale
        attn = (q @ k.transpose(-2, -1))    #计算注意力分数矩阵

        relative_position_bias = self.relative_position_bias_table[
            self.relative_position_index.view(-1)].view(
            self.win_size[0] * self.win_size[1], self.win_size[0] *
                self.win_size[1], -1)  # Wh*Ww,Wh*Ww,nH
        relative_position_bias = relative_position_bias.permute(2,
            0, 1).contiguous()  # nH, Wh*Ww, Wh*Ww
        ratio = attn.size(-1)//relative_position_bias.size(-1)
        relative_position_bias = repeat(relative_position_bias, 'nH
            l c -> nH l (c d)', d = ratio)

        attn = attn + relative_position_bias.unsqueeze(0)    #在自注
            意力中引入相对位置偏移
```

4.4 实验环境搭建

Kaggle 平台默认环境
einops==0.6.1
natsort==8.4.0

4.5 创新点

本文采用了 Uformer 的架构来进行图像恢复任务，本质上来说是一个 U 型 (编码器-解码器型) 的 Swin-Transformer，原始的 Swin-Transformer 采用了相对位置信息编码，这样的编码方式加大了对应的计算量，在我的工作中，我对编码方式进行了改进，发现换成 2D 位置信息编码也能取得较为接近的效果，原因可能是当前数据集的结构信息并没有那么复杂，采用 2D 位置信息编码就能够很好地表示当前位置的语义信息，并且能降低计算量。

本文在进行输入输出维度变换时，采用了卷积与转置卷积来对特征图进行处理后进行维度转换，在我的工作中，我直接对特征维度进行填充转换，去掉了卷积操作，可以减少一定的计算量。并且实验结果不会有太大变化。

5 实验结果

本次实验在 SIDD-Small Dataset sRGB images only [1] 和 GoPro 数据集上采用 UformerB 模型分别进行训练，进行图像去噪，运动去模糊，散焦去模糊和去雨任务实验。

下图 4 是利用 Uformer 模型进行图像去噪任务结果的示意图。

图 4. 去噪实验结果示意

下图 5 是利用 Uformer 模型进行运动去模糊任务结果的示意图。

图 5. 运动去模糊实验结果示意

下图 6 是利用 Uformer 模型进行散焦去模糊任务结果的示意图。

图 6. 散焦去模糊实验结果示意

下图 7 是利用 Uformer 模型进行去雨任务结果的示意图。

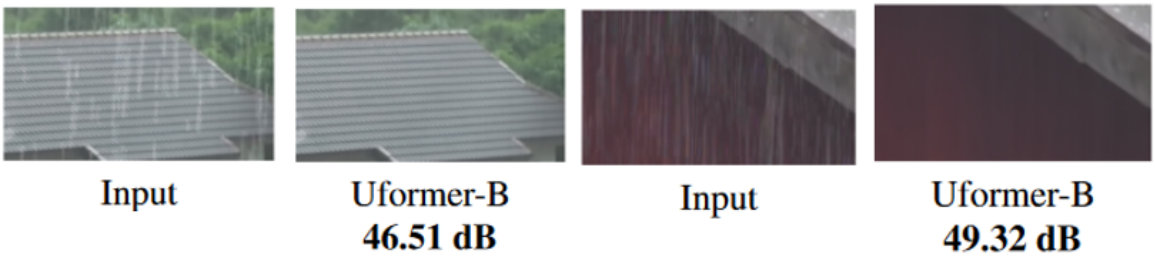


图 7. 去雨实验结果示意

除此之外我们还对不同模型在 SIDD-Small Dataset sRGB images only 数据集上的去噪效果进行了实验，实验结果如下表:

	SIDD	
Method	PSNR	SSIM
BM3D	25.65	0.685
RIDNet	38.71	0.914
VDN	39.28	0.909
DANet	39.47	0.918
CycleISP	39.52	0.957
MIRNet	39.72	0.959
MPRNet	39.71	0.958
NBNet	39.75	0.959
Uformer-B	**39.89**	**0.960**

6 总结与展望

在本次工作中，复现了一种新的进行图像恢复的技术通用的 U 型的用与图像恢复任务的 Transformer(Uformer)，该方法优于先前的算法，并且在部分图像恢复任务中达到了 SOTA，并且减少了原始的 Transformer 架构的计算复杂性。这个结果可以归功于以下几点:

1. 采用了 U 型的结构，即编码器解码器结构，这种结构被证明可以充分学习图像的特征信息。

2. 借鉴 Swin Transformer 移动窗口的思想，使得自注意力的计算不止局限在窗口内，还可以融合不同窗口的信息进行计算。

3. 提出了多尺度的恢复调制器，在解码的不同阶段加入可以训练的特征表示，可以让模型在运动去模糊任务取得更好的效果。

该模型在现实世界中的图像增强中取得了优秀的结果，但对于比现实世界更复杂的水下环境图像的增强，是否该模型仍然会表现出优异的效果仍然未知，接下来，我将进行进一步探索该模型在水下图像增强中的表现并进行分析，探索本模型在水下图形增强应用时的改进方法。

参考文献

[1] Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. A high-quality denoising dataset for smartphone cameras. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1692–1700, 2018.

[2] Abdullah Abuolaim and Michael S Brown. Defocus deblurring using dual-pixel data. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*, pages 111–126. Springer, 2020.

[3] Saeed Anwar and Nick Barnes. Real image denoising with feature attention. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3155–3164, 2019.

[4] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12299–12310, 2021.

[5] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *Advances in Neural Information Processing Systems*, 34:9355–9366, 2021.

[6] Xiaodong Cun, Chi-Man Pun, and Cheng Shi. Towards ghost-free shadow removal via dual hierarchical aggregation network and shadow matting gan. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10680–10687, 2020.

[7] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12124–12134, 2022.

[8] Shuhang Gu, Yawei Li, Luc Van Gool, and Radu Timofte. Self-guided network for fast image denoising. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2511–2520, 2019.

[9] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11936–11945, 2021.

[10] Zilong Huang, Youcheng Ben, Guozhong Luo, Pei Cheng, Gang Yu, and Bin Fu. Shuffle transformer: Rethinking spatial shuffle for vision transformer. *arXiv preprint arXiv:2106.03650*, 2021.

[11] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[12] Siyuan Li, Iago Breno Araujo, Wenqi Ren, Zhangyang Wang, Eric K Tokuda, Roberto Hirata Junior, Roberto Cesar-Junior, Jiawan Zhang, Xiaojie Guo, and Xiaochun Cao. Single image deraining: A comprehensive benchmark analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3838–3847, 2019.

[13] Yawei Li, Kai Zhang, Jiezhang Cao, Radu Timofte, and Luc Van Gool. Localvit: Bringing locality to vision transformers. *arXiv preprint arXiv:2104.05707*, 2021.

[14] Lin Liu, Xu Jia, Jianzhuang Liu, and Qi Tian. Joint demosaicing and denoising with self guidance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2240–2249, 2020.

[15] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[16] Kuldeep Purohit, Maitreya Suin, AN Rajagopalan, and Vishnu Naresh Boddeti. Spatially-adaptive image restoration using distortion-guided networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2309–2319, 2021.

[17] Kuldeep Purohit, Maitreya Suin, AN Rajagopalan, and Vishnu Naresh Boddeti. Spatially-adaptive image restoration using distortion-guided networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2309–2319, 2021.

[18] Hong Wang, Qi Xie, Qian Zhao, and Deyu Meng. A model-driven deep neural network for single image rain removal. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3103–3112, 2020.

[19] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 568–578, 2021.