華中科技大學

图像处理与计算机视觉课程设计

基于深度学习的建筑物变化检测系统

院　　系　人工智能与自动化学院

专业班级　人工智能本硕博 2001 班

姓　　名　[redacted]

学　　号　[redacted]

指导教师　高常鑫

2023 年 12 月 5 日

摘　　要

基于遥感数据的变化检测是检测地球表面的重要方法，其在城市规划、环境监测、地形变化探测、农业调查等方面均具有广泛的应用。目前，随着经济水平和人口的增长，城市地区在地球表面的分布在进行剧烈的变化、能否准确的检测并识别出地形的变化与城市的管理者以及居民的生活息息相关。

变化检测任务的主要难点在于光照角度，阴影，地面物体的反射，植被生长，泥土冲刷等多种随机因素对最终检测效果的影响，提出一个高准确度的、具有鲁棒性、高泛化性的模型，是目前该方向的主要目标。

在本次课程设计中，我们首先尝试了 STANet[4] 模型，并将其作为本次课程设计中的 baseline。鉴于 Swin-transformer 在视觉任务的多个领域都具有很好的效果，我们还尝试了 SwinSUNet[2]。考虑到现有的多尺度特征融合方法往往采用冗余特征提取和融合策略，这导致了变化检测模型需要较高的计算成本和内存占用且变化检测中的常规的注意力机制难以同时对空间和谱间特征进行建模并生成三维注意力权重，忽略了空间特征和谱间特征之间的协同作用，我们采用了 USSFC-Net[10]。最后，我们尝试了基于 Transformer 的孪生网络结构 ChangeFormer[1]。通过四个模型在不同数据集上的训练结果，我们最终选取了在 LEVIR-CD 测试集上取得 F1score =0.9137 IOU=0.8410 的测试结果，且在其余两个 SYSU-CD、WHU-CD 测试集上更加鲁棒的算法 USSFC−Net 作为本次课设的最终结果提交。

在 Zero-shot learning 中，我们的模型在处理新领域或未见过的情境时遇到了性能下降的问题。值得注意的是，目前学术界的在变化检测领域的研究主要集中在传统的迁移学习和有监督学习，而零样本迁移并不是主流关注的方向。然而，由于我们的课程设计最终的测试集可能包含未公开的、以不同时间和方式收集的样本，我们选择进行了零热启动测试，以评估模型在这种零样本迁移情况下的表现。

值得一提的是，我们的模型在零热启动情况下的表现并不能使人满意，这一问题的挑战性在于模型需要在没有充分信息的情况下进行推断和泛化，这可能导致性能下降。因此，我们需要进一步研究和改进模型，以提高其在零热启动任务中的性能，从而更好地适应不同领域和未知情境。

关键词：变化检测；遥感数据；注意力机制；零样本迁移学习；

Abstract

Change detection using remote sensing data is a crucial method for monitoring changes on the Earth's surface. It finds wide-ranging applications in urban planning, environmental monitoring, terrain change detection, agricultural surveys, and more. In recent times, with the growth in economic prosperity and population, urban areas have been undergoing rapid transformations. The accurate detection and identification of terrain changes have become vital for urban management and the well-being of the general population.

The primary challenges in change detection tasks stem from various random factors like lighting angles, shadows, surface object reflectance, vegetation growth, soil erosion, and more. These factors can significantly affect the final detection results. Thus, the central goal in this field is to develop a high-precision, robust, and highly adaptable model.

In this course project, we initially explored the STANet model, taken as our baseline, as introduced in [4]. Given the exceptional performance of the Swin-Transformer in various visual tasks, we also ventured into SwinSUNet as an alternative [2]. Considering that existing multi-scale feature fusion methods often involve redundant feature extraction and fusion strategies, resulting in higher computational costs and memory usage in change detection models, and conventional attention mechanisms struggle to effectively model both spatial and spectral features while generating three-dimensional attention weights, overlooking the synergy between spatial and spectral features, we adopted the USSFC-Net [10].

Lastly, we delved into the ChangeFormer model, which is a Transformer-based Siamese network structure [1]. After training the four models on different datasets, we ultimately selected the algorithm that achieved F1score =0.9137 IOU=0.8410 on the LEVIR-CD test set and demonstrated greater robustness on the SYSU-CD and WHU-CD test sets with USSFC−Net. This algorithm serves as the final result for our course project submission."

In the context of zero-shot learning, our model encounters performance degradation when dealing with new domains or previously unseen scenarios. It is noteworthy that current research in the field of change detection primarily focuses on traditional transfer learning and supervised learning, with zero-shot transfer not being the mainstream area of interest. However, due to the possibility that our course project's final test set may include

undisclosed samples collected at different times and in various ways, we conducted zero-shot testing to evaluate the model's performance in such a zero-sample transfer scenario.

It is worth mentioning that our model's performance in zero-shot scenarios is not satisfactory. The challenge lies in the fact that the model needs to make inferences and generalize without having sufficient information, which can lead to a decrease in performance. Therefore, further research and model improvements are needed to enhance its performance in zero-shot learning tasks, enabling it to adapt better to different domains and unknown scenarios.

Key Words: Change Detection; Remote Sensing data; Attention mechanism;Zero-shot learning

目　　录

1 选题介绍

1.1 研究背景及现状

研究背景: 基于遥感数据的变化检测是检测地球表面变化的重要方法，并且在城市规划，环境监测，农业调查，灾害评估和地图修订中具有广泛的应用。变化检测是通过在不同时间观察来识别物体或现象状态的差异的过程。它是地球观测中的主要问题之一，并在近几十年中被广泛研究。

经济发展、人口增长和城市化进程的加快导致了城市地区的急剧变化，准确及时地识别这些变化并分析其趋势已经成为城市管理者的重要课题，是地理国情普查与监测的重点关注内容，成为目前亟待解决的问题。其中建筑的拆除、建设和扩建是城市蓝图规划关注的重要组成部分，与人类的生存活动密切相关。

遥感监测任务是在不同的时刻图像对之间探测地形的变化，由于图像的光照变化以及不同时刻图像之间的对齐困难使得不同时刻之间的地形变化难以被探测到。

研究现状: 遥感图像变化检测任务旨在多时刻的遥感图像之间进行地形变化的检测 [13], 在城市化进程监测、环境监测、灾难评估等方面具有极高的应用价值。

在过去的十几年中，有许多面向遥感图像变化任务所提出的模型，大多数模型分为两个部分：组成单元分析和变化探测。组成单元分析旨在从图像单元的原始数据之中提取出信息含量高的特征，所分析的组成单元有像素和对象，大量的工作针对组成单元提取光谱特征和空间位置特征，从而进行进一步的探测分析。

针对分析的单元，可以利用手工提取的特征或者借助深度学习网络对特征进行提取，在提取完特征之后，有一种简单的方式是通过不同时刻之间像素特征的差异和阈值进行比较，超过阈值则该像素被判断为变化的像素 [13]。尽管简单，该方法对阈值依赖严重，泛化能力较差，手工提取的特征往往很难在其余场合下具有高信息含量。向量变化分析 (CVA)[9] 同时分析变化向量的模长和方向来判断像素的类型，一些常用的分类器如支持向量机 (SVM)、决策树 (DTs)、图模型、马尔可夫随机场都可以用于后续的像素分类。

除了仅仅利用 2D 信息来对变化像素进行检测之外，高度信息也对遥感变化探测任务有宜，由于不同的高度位置会有不同光照信息和投影变化，3D 的信息也可以提升变化检测的准确性 [12]。3D 的信息可以借助激光雷达或者稠密像素匹配的

方式取得，然而，激光雷达通常十分昂贵，借助计算机视觉方式对立体图对进行推断从而获得高度信息的方式通常经济代价较低，但是所得到的三维信息质量较差，对后续的变化探测可能带来负面效益。

随着深度学习的兴起，有大量的基于深度学习的变化探测模型被提出，相较于传统方式，深度学习在遥感变化检测任务中也显现了优异的性能，主要可以分为两类：基于指标的方法和基于分类的方法。

基于指标的方法通过比较参数化的不同时刻的数据之间的距离来判别是否发生变化。基于指标的方法通常需要学习一个参数化的嵌入空间，在特征空间中，没有发生变化的像素的特征向量距离近 (类间差异小)，而发生了变化的像素的特征向量的距离远 (类间差异大)，嵌入空间的学习可以借助稠密预测网络 FCN[17],FCN 架构包括两个共享权重的卷积网络，分别对不同时刻的图像输入进行特征提取，从而进行进一步的判断。

基于分类的方法则将变化检测问题视作二分类问题，普遍使用的方法是给图像中的每一个位置一个变化分数，有变化的像素的分数高于没有变化的像素。卷积神经网络和循环神经网络被广泛用于提取不同时刻图像之间的特征，这些网络架构输入非常小的图像 patch，从而进一步进行特征提取等处理，然而无论是基于指标的方式还是基于分类的方法，都没有很好的利用图像的时间特性。

1.2 LEVIR-CD 数据集介绍

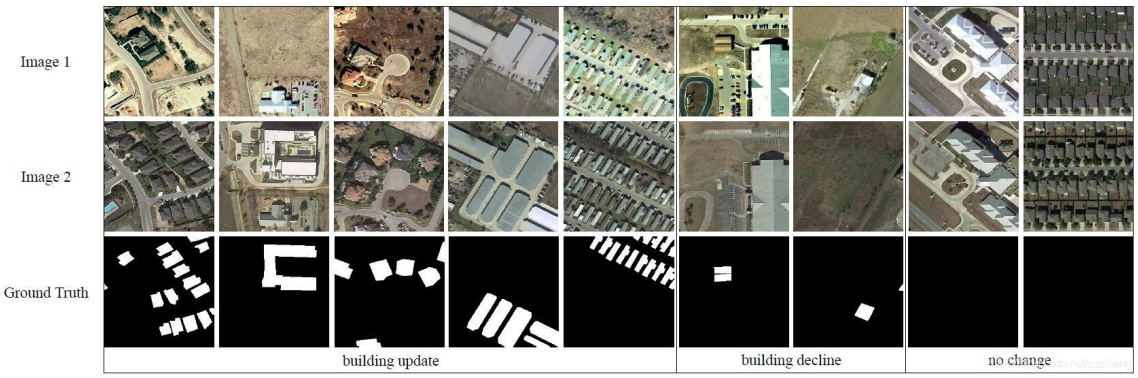

图 1-1 LEVIR-CD 数据集展示

LEVIR-CD 由 637 个高分辨率（VHR，0.5 米/像素）的谷歌地球（GE）图像补丁对组成，大小为 1024×1024 像素。这些时间跨度为 5-14 年的双时态图像具有显著的土地利用变化，尤其是建筑业的增长。LEVIR-CD 涵盖各种类型的建筑，如别

墅住宅、高层公寓、小车库和大型仓库。在这里，我们关注与建筑相关的变化，包括建筑的增长（从土壤/草地/硬化地面或在旧建筑到新的建筑区域的变化）和建筑的衰落。遥感图像解释专家使用二进制标签（1 表示变化，0 表示不变）对这些双时态图像进行注释。我们数据集中的每个样本都由一个注释器进行注释，然后由另一个进行双重检查，以生成高质量的注释。完整注释的 LEVIR-CD 总共包含 31333 个单独的变更构建实例。

LEVIR-CD 的地理空间分布：LEVIR-CD 中的双时态图像来自美国得克萨斯州几个城市的 20 个不同地区，包括奥斯汀、Lakeway、Bee Cave、Buda、Kyle、Manor、Pflugerville tx、Dripping Springs 等。LEVIR-CD 数据集图像数据的拍摄时间从 2002 年到 2018 年不等。可以在不同的时间拍摄不同区域中的图像，从而引入了季节变化和光照变化而引起的变化。

1.3 项目难点

遥感变化检测可定义为利用多时相遥感数据，采用多种图像处理和模式识别方法提取变化信息，并定量分析和确定地表变化的特征与过程。

影响监测结果的因素主要可以分为两方面：一方面是遥感系统的影响，另一方面是环境的影响。遥感系统的影响可以从遥感数据特征角度分析，即四个分辨率。时间分辨率是变化检测最基本的考量因素，只有图像之间跨越调查区域的变化时期，我们才可以从多时相图像中获取变化信息；不同的空间分辨率获取到的同一地区数据特征可能是不同的，需要根据实况进行合理选择；另外，我们所选择的系统的光谱分辨率应当足以记录光谱区域内反射的辐射通量，从而有效描述有关对象的光谱属性；不同辐射分辨率之间的图像数据应当注意及时转换。环境因素的影响因子是很多的。最常见的就是大气影响，另外还有土壤湿度等。数据集中的每一对数据一般是不同时间拍摄的两幅配准图像。在不同的时间，由太阳照射角度的变化会引发图像采集过程的照明变化。由于太阳位置的变化，在两幅遥感图像中，建筑物周围可能会呈现出不同的阴影。另外，由图像配准所带来的在建筑物边缘处的误差在遥感图像中非常显著，以上两个因素会影响建筑物变化检测的结果。例如，co-registered images 虽然都是对同一地区影像内容的呈现，但可能前一张在拍摄时有云雾的遮盖而后一张没有，但算法却将云雾识别为地物内容的变化；另外，光照角度，阴影，地面物体的反射，植被生长，泥土冲刷等因素都可能影响到变化

检测。所以遥感变化检测的目标是能设计出不受以上无关因素影响的 robust 算法。

2 方案介绍

2.1 Baseline 方案：STANet

在本节中，本文将介绍 LEVIR-CD 数据集的 baseline 方案 [4]，该方案基于空间-时序注意力机制对变化进行检测。Chen[4] 针对遥感图像变化检测问题提出了一个基于孪生神经网络的时空注意力变化检测模型 STANet，其中的自注意力模块可以计算任意两张拍摄于不同日期和位置的图像的注意力权重，并产生更具辨别性的特征。考虑到物体可能具有不同的大小，将图像分割成了多尺度的子区域，并在每个子区域中引入了自注意力机制。此外还创建了新的变化检测数据集 LEVIR-CD。该数据集也为本次课程设计中所使用的数据集。如图 2-1所示，(a) 是时空注意力的示意图，(b) 是图像误配准的情况。

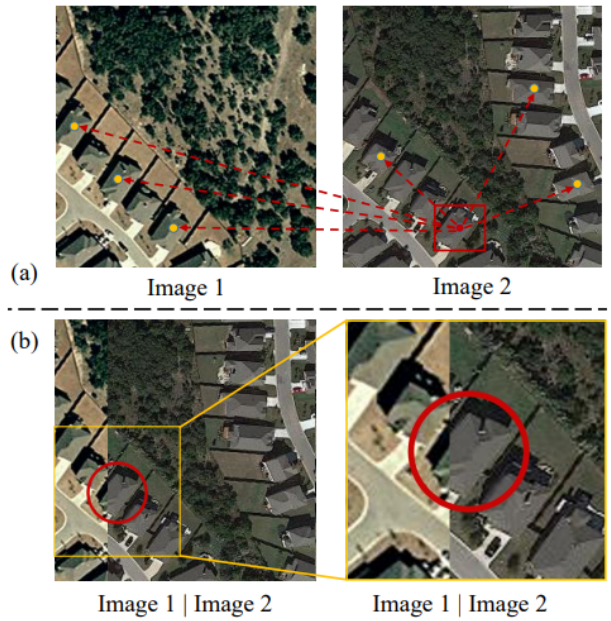

图 2-1 STANet 示意图

基于深度学习的变化检测方法主要可以分为两类：基于度量的方法和基于分类的方法。基于度量的方法通过对比图像之间参数化的距离来决定是否发生变化。每一对点之间的特征的度量表示是否发生了变化。基于分类的方法通过对提取到的图像特征进行分类，从而识别变化的类别。STANet 属于基于度量的方法。

2.1.1 STANet Motivation

Chen 提出 STANet[4] 主要有两点动机：

- 变化检测数据是有时间维度和空间维度的光谱向量组成的，开发不同时空位置之间的关系可以提升变化检测方法的效果。因此提出了时空自注意力机制。
- 由于变换物体可能具有不同的大小，从一个合适的范围内提取特征可以更好地表示一定尺度的对象。可以通过从不同大小区域提取得到的特征结合起来以获得不同尺度的特征。因此将图像分割成了多尺度的子区域，并在每个子区域中引入了自注意力机制。

2.1.2 STANet Pipeline

文章设计了两种自注意力模块，基本的时空注意力模块 BAM 和金字塔时空注意力模块 PAM。BAM 任意两个位置之间的时空独立性注意力权重，并通过时空中所有位置特征的加权和来计算每个位置的响应。PAM 则将 BAM 嵌入得到一个金字塔结构以产生多尺度的注意力表示。STANet 的架构如图 2-2所示。

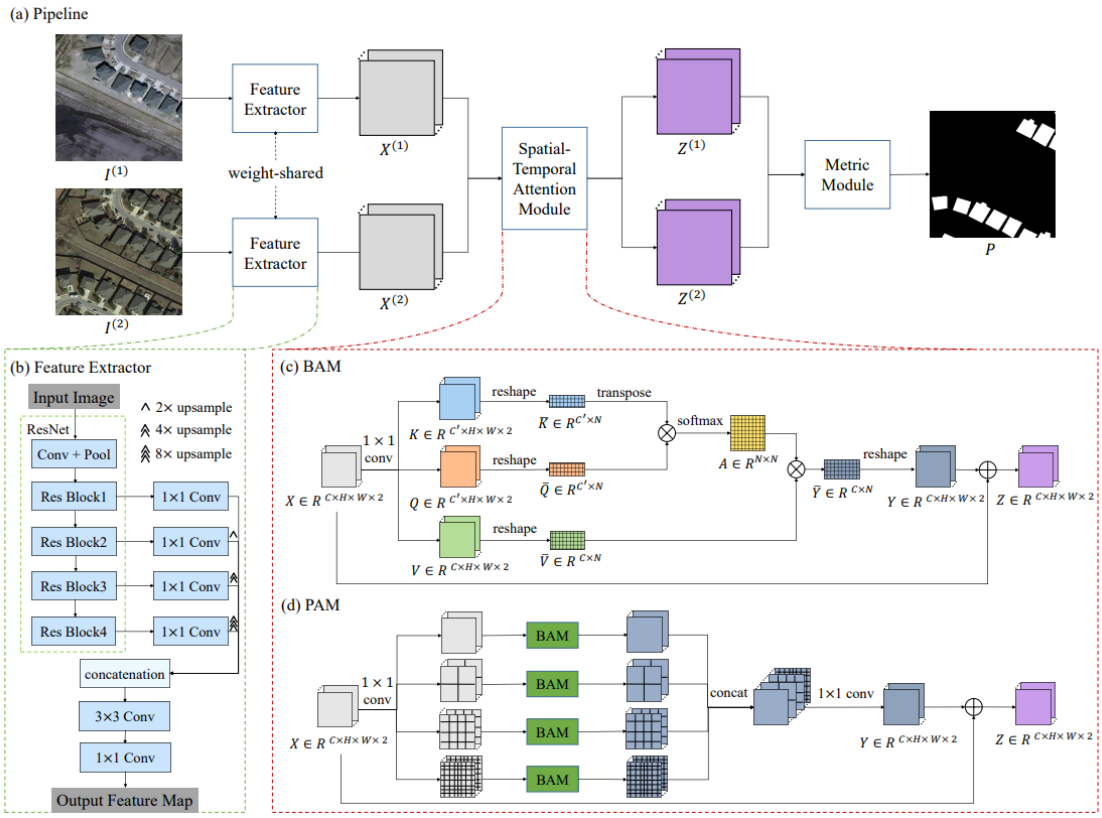

图 2-2　STANet Pipeline

STANet 包括特征提取器、注意力模块、度量模块三部分。

- **特征提取器**特征提取器中用到了 ResNet-18，由于 ResNet 是用来进行图像分类任务而变化检测是密集分类任务，所以省略了 ResNet 中的全局池化层和全连接层。
- **BAM** 在 BAM 中，特征图 X 首先通过三个不同的 1×1 的卷积层得到三个特征向量 Q、K、V 分别表示查询、键和值。然后对其 reshape 得到矩阵 $\tilde{Q}$、$\tilde{K}$、$\tilde{V}$，并使用转置后的 $\tilde{Q}$、$\tilde{K}$ 进行矩阵乘法并使用 softmax 计算一个相似矩阵 A，该相似矩阵与 $\tilde{V}$ 进行矩阵乘法得到输出矩阵 $\tilde{Y}$，对其进行 reshape 得到注意力 Y，Y 与 X 进行像素级乘法得到最终的注意力特征图 Z。
- **PAM** PAM 有 4 个分支，每个分支将特征图 X 分成了不同大小的子区域，并在每个子区域中应用 BAM，每个分支的输出拼接起来和输入大小相同，将 4 个分支的输出 concate 起来并用 1×1 的卷积层进行处理得到注意力 Y，Y 与 X 进行像素级乘法得到最终的注意力特征图 Z。

2.1.3 损失函数

模型的度量首先将特征图使用双线性插值 resize 到和输入相同的大小，然后计算两个特征图之间像素级的欧氏距离图 D，在训练阶段，用其来计算损失值，在测试阶段使用一个固定的阈值方法进行分割。该模型设计了一个批量平衡对比损失（BCL），利用批次权重对原始对比损失的类权重进行修正，损失函数的定义如式子2-1:

$$L(D^*, M^*) = \frac{1}{2}\frac{1}{n_u}\sum_{b,i,j}(1 - M^*_{b,i,j})D^*_{b,i,j} + \frac{1}{2}\frac{1}{n_c}\sum_{b,i,j} M^*_{b,i,j} Max(0, m - D^*_{b,i,j}) \quad (2\text{-}1)$$

其中，M^* 是二值标签图的一个批次,b, i, j 表示批次的下标、高度、宽度。m 是 margin,n_u,n_c 是未变化和变化了的像素对的个数，其计算公式如2-2：

$$\begin{aligned} n_u &= \sum_{b,i,j} 1 - M^*_{b,i} \\ n_c &= \sum_{b,i,j} M^*_{b,i,j} \end{aligned} \quad (2\text{-}2)$$

2.2 SwinSUNet 算法

2.2.1 SwinSUNet Motivation

卷积神经网络能过够提取语义特征，但是由于卷积运算固有的局部性导致卷积神经网络无法捕捉是空的全局信息。近年来，一种新的网络 transformer[15] 被提出：transformer 通过 multi-head self attention 机制能够提取到图像的全局信息。所以尝试将 transformer 应用于特征提取部分。

2.2.2 SwinSUNet pipeline

网络整体包括 encoder ,fusion module 和 decoder, 如图 2-3所示。SwinSUNet 网络的输入为一对分别于不同时间拍摄的同一地点的遥感图像。输出为变化图, 输出为 0，代表没有变化，输出为 1，代表发生变化。encoder 和 decoder 仿照 FC-Siam-conc[3] 的结构设计，并且 encoder 设计为连体结构，从而可以处理双时态图像。在 encoder 中，首先通过 patch partition 和 linear embedding 提取输入图像的特征，将特征转换为 tokens, 然后使用由 patch merging 和 swin-transformer block 组成的分层 swin-transformer[11] 来提取多尺度特征。对于由 encoder 提取到的双时态特征，fusion module 先拼接两个特征，然后经过线性映射和 swin-transformer block 来进一步的融合特征，decoder 使用分层的 swn-transformer 来逐步预测变化信息。与 encoder 不同的是, decoder 中的分层的 swin-transformer 由 umsampling-merging(UM) block 和 swin-transformer block 组成，随着分层的 swin-transformer 处理过程的进行，特征图的分辨率逐渐变大，逐步生成变化信息。在 decoder 中，最后使用上采样层和线性投影层来输出与输入图像具有相同分辨率的变化图。

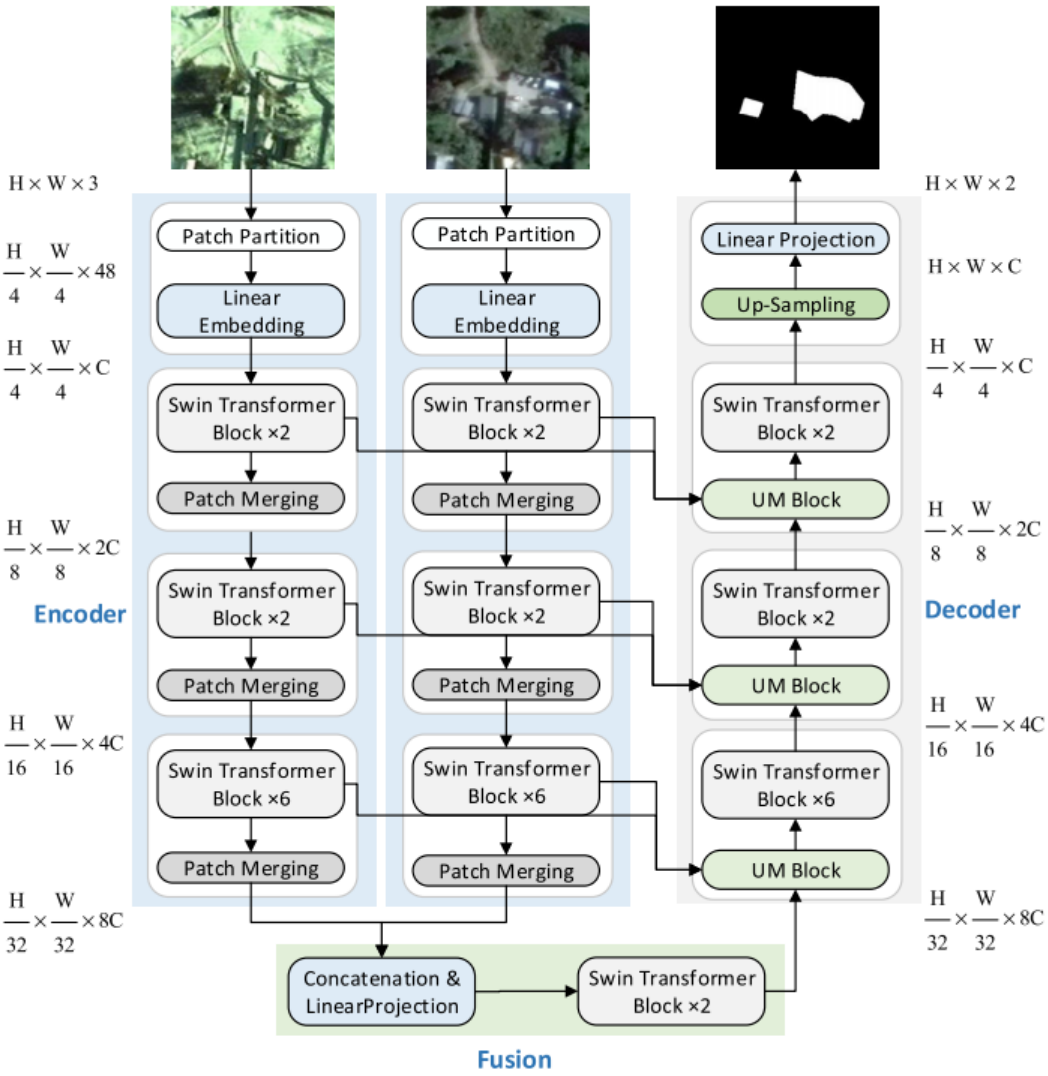

图 2-3　overall framework

- **Swin-transformer**　为了降低全局自注意的计算复杂度, Swin transformer 设计了一个 Swin transformer block。在 Swin transformer block 中，标准的 multi-head attention（MSA）模块被 (shifted) window based multi-head attention(SW-MSA）模块取代。Swin transformer block 中一共包含 shifted window based multi-head attention(SW-MSA）模块，如图 2-4所示, 多层感知器（MLP）模块、LayerNorm（LN）层和 residual connections.

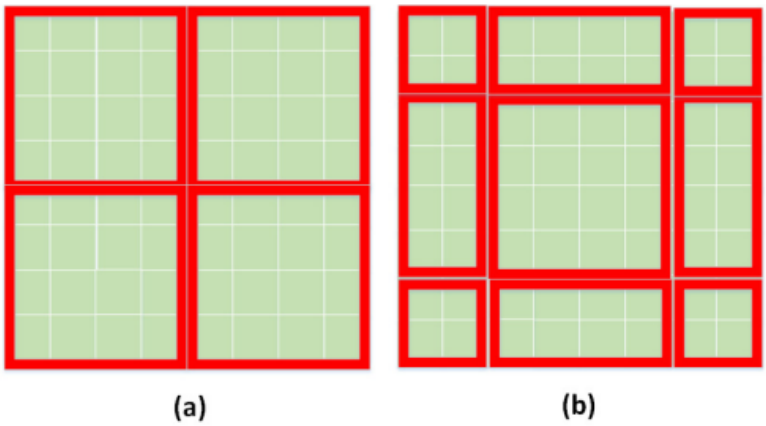

图 2-4 (a)Window partition strategy of W-MSA module(b) Window partition strategy of SW-MSA module

– **W-SMA module** ViT[7] 和 DeiT[14] 采用标准的 MSA 模块，在整个特征图上进行全局自注意力计算。这一部分的计算复杂度与输入的图像尺寸大小呈平方关系，计算开销比较大。为了降低计算开销，Swin transformer 提出了 W-MSA 模块。W-MSA 模块只在每个窗口内进行局部自注意力计算，使计算复杂度与输入的图像尺寸大小呈线性关系。W-MSA 模块将特征映射划分为多个大小为 M×M 的窗口。所示这个模块在每个窗口中分别计算自我注意。W-MSA 模块中自注意的计算公式与标准 MSA 模块中相同。

– **SW-SMA module** 由于 W-MSA 模块只在局部窗口中进行自注意，因此网络的感受野受到了限制，这会影响网络的表征能力。为了扩大感受野，Liu 等人 [32] 提出了 SW-MSA 模块。SW-MSA 模块的窗口划分策略如 b 所示，SW-MSA 模块的窗口相对于 W-MSA 模块窗口偏移了 (M/2) 和 (M/2) 偏移。在 Swin transformer 中，W-MSA 模块与 SW-MSA 模块交替使用，以增加网络的感受野。

– **self attention** 与之前的方法不同，Swin transformer 只在局部窗口中计算自注意力，但是计算注意力的方法没有发生改变。

$$Q = TW_q \tag{2-3}$$

$$K = TW_k \tag{2-4}$$

$$V = TW_v \tag{2-5}$$

$$Attention(Q, K, V) = SoftMax(\frac{QK^T}{\sqrt{d}} + B)V \tag{2-6}$$

其中 $T \subset R^{(M^2)\times(d)}$ 代表自注意力的输入，$W^q, W^k, W^v \subset R^{(d)\times(d)}$ 代表的是三个线性映射层的权重，在本文中，线性映射层使用的是全连接层表示的，$Q, K, V \subset R^{(M^2)\times(d)}$ 分别代表 query,key,value 矩阵，M^2 代表的是窗口的分辨率，V 和 d 代表的是通道维度，在 Swin transformer，偏置矩阵 $\hat{B} \subset R^{(2M-1)\times(2M-1)}$ 是可学习的参数.

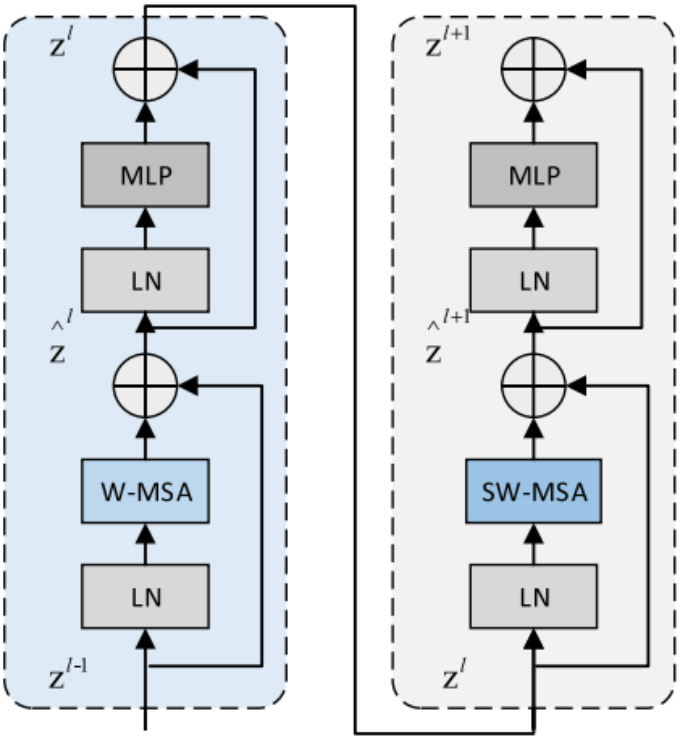

图 2-5　两个连续的 swin transformer block

– **Swin Transformer Block** 如图 2-5所示显示了两个连续的 Swin transformer block：第一个使用 W-MSA 模块，另一个使用 SW-MSA 模块。正如我们前面所描述的，W-MSA 和 SW-MSA 模块使用不同的窗口划分策略来减少窗口局部性的影响。两个连续的 swin transformer block 的计算公式如下所示

$$\hat{z}^l = WSMA(LN(z^{l-1})) + z^{l-1} \tag{2-7}$$

$$z^l = MLP(LN(\hat{z}^l) + \hat{z}^l) \tag{2-8}$$

$$\hat{z}^{l+1} = SWSMA(LN(z^l)) + z^l \tag{2-9}$$

$$z^{l+1} = MLP(LN(\hat{z}^{l+1}) + \hat{z}^{l+1}) \tag{2-10}$$

其中 $\hat{z}^l$ 代表的是 SW-MSA 模块的输出，z^l 代表的是 MLP 的输出

- **Encoder** 类似于 ViT[7] 和 Swin transformer,SwinSUNet 需要在提取语义特征之前，先将输入的维度为 H*W*C 的图像转换成 tokens。这个过程 SwinSUNet 和 Swin transformer 完全一样，包括一个 patch partition module 和一个 linear embedding module。

  patch partition module 的作用是将图像转换成多个 tokens, 而 linear embedding module 则是将 tokens 的维度映射到一个指定的维度，在论文中为 C。首先，patch partition module 将图像分割成很多个 small patches, 每个 patch 的大小为 4*4，通道数为 3，通过分割得到的 patch 总数为 (H/4)*(W/4), 每一个 patch 被看作一个 token, 然后 patch partition module 将每一个 patch 从三维展平为一维，每个 token 的尺寸从 4*4*3 变为 48，然后 linear embedding module 将 tokens 的尺寸从 48 映射为一个固定值 C，这样，feature map 的尺寸变为 (H/4)*(W/4)*C。特征图上每一个空间中的点都是一个 token。论文仿照 swin transformer,patch partition module 和 linear embedding module 都由卷积操作实现，卷积操作输入的通道数为 3，输出的通道数为 C，每个卷积核的大小为 4，步长为 4.

  在 encoder 中，得到的 tokens 送入一个由 patch merging 和 swin-transformer block 组成的分层 swin-transformer。分层 swin-transformer 一共有三个阶段，每一个阶段由几个 swin transformer block 和一个 patch merging layer 组成。swin transformer block 用来提取有效的特征，patch merging layer 的作用是对特征进行下采样。类似于 swin transformer,SwinSUNet 也使用 patch merging layer 来对特征进行下采样，如图 2-6所示。首先，patch merging layer 先对输入的特征进行间隔采样，输入的特征 $X_{E_i}^p \subset R^{(H/2^{i+1})\times(W/2^{i+1})\times(2^{i-1}C)}$ 得到四个特征图 (其中 i 代表第 i 个阶段)，特征图的维度为 $H/2^{i+2} \times (W/2^{i+2}) \times (2^{i-1}C)$，然后在通道维度上对四个特征图进行拼接，从而生成尺寸为 $H/2^{i+2} \times (W/2^{i+2})$，通道维度为 $2^{i-1}C$ 的特征图，然后使用 linear projection layer 将特征的通道维度从 $2^{i-1}C$ 变为 2^iC 。最终得到的特征尺寸和通道维度为 $H/2^{i+2} \times (W/2^{i+2}) \times (2^iC)$

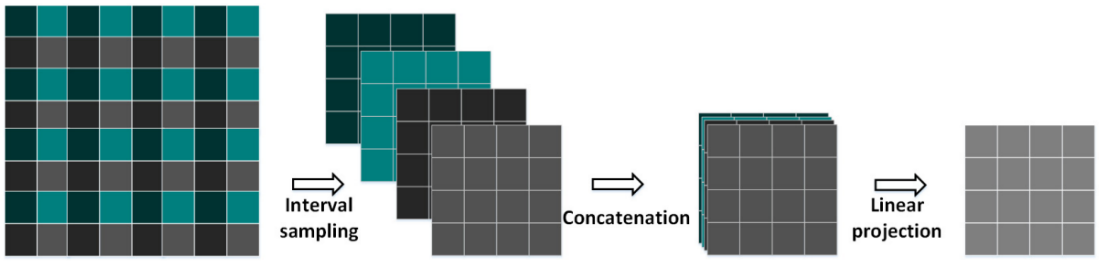

图 2-6　patch merging layer

在每一个阶段后，特征图的尺寸变为原来的一半，而维度变为原来的两倍，经过三个阶段，特征图的尺寸由 (H/4)*(W/4) 变为 (H/32)*(W/32), 而特征图的维度由 C 变为 8C。

在 encoder 的每个阶段，Swin transformer block 的输出特征映射被保存并表示为 $X_{E_i}^1$ 和 $X_{E_i}^2$，其中 i 代表第 i 个阶段。$X_{E_i}^1$ 和 $X_{E_i}^2$ 将被输入到相应的 decoder 阶段。它们将与解码器特性一起用于恢复更改信息。此外，encoder 最终输出双时态特征 X_E^1 和 X_E^2，并将这两个 feature map 输入到 fusion module 中。明显可知，encoder 可以生成多尺度特征，其中高层特征包含丰富的语义信息，而低层特征保留了更多的细节信息。

- **Fusion module**　Fusion module 的作用是对 encoder 生成的双时态特征 X_E^1 和 X_E^2 进行合并，从而便于 decoder 生成 change map.Fusion module 由一个 concatenation layer, 一个 linear projection layer 和 2 个 swin-tranformer block 组成。concatenation layer 负责对 X_E^1 和 X_E^2 进行连接，linear projection layer 主要负责对特征进行降维，而 swin-transformer block 则是对特征进行深度融合。特征图在输入 fusion module 前和从 fusion module 中输出分辨率保持不变。
- **Decoder**　和 encoder 对应，decoder 也使用分层 swin-transformer。decoder 的 swin-transformer 也包含三个阶段，每一个阶段包括一个 unsampling-merging(UM) block 和几个 swin transformer block,UM block 包含一个 unsampling block 和一个 merging block，如图 2-7所示. 首先，UM block 使用一个 patch reshaping block 来实现上采样操作，然后 Merging Block 用来对上采样得到的特征图 $X_{D_i}^U$ 以及从 encoder 生成的特征 $X_{E_{3-i+1}}^1$ 和 $X_{E_{3-i+1}}^2$ 进行融合.

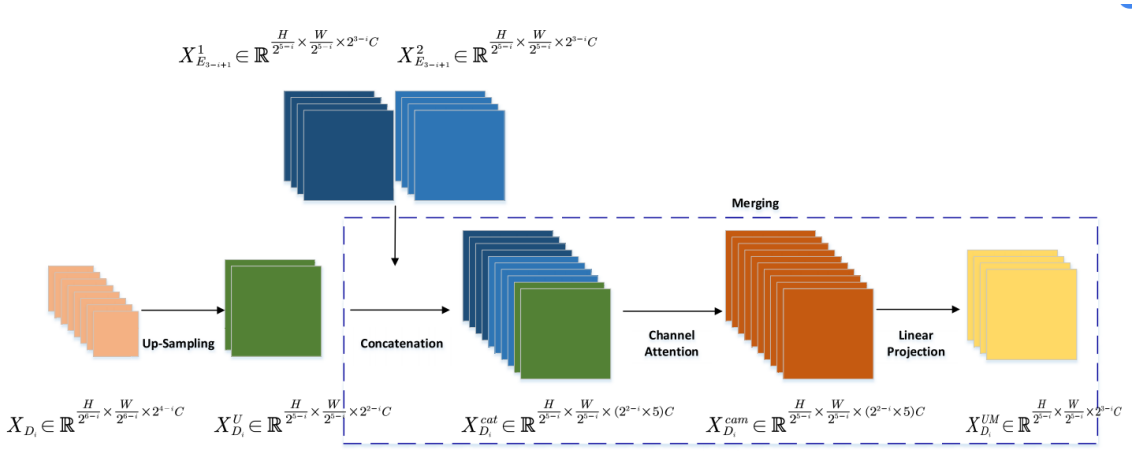

图 2-7　UM Block

decoder 第 i 个阶段之前输入的特征图定义为 X_{D_i}，其中 i 代表第 i 个阶段，第 i 阶段的输出特征图 X_{D_i} 的尺寸是输入特征图 $X_{D_{(i-1)}}$ 的尺寸的两倍，输出的特征图通道维度是输入特征图的通道维度的一半，经过三个阶段，change information 逐渐生成，特征图的尺寸由 (H/32)×(W/32) 变为 (H/4)×(W/4)，通道维度由 8C 变为 C。经过分层 Swin transformer 后，decoder 首先使用 unsampling layer 将特征映射扩展到与原始输入图像相同的分辨率。然后 decoder 利用一个 linear projection layer 将特征维数从 C 转换为 2；这样就得到了 change map.

– **Patch Reshaping**　在之前的算法中，常用的上采样方法包括双线性插值和转置卷积。在论文 Swin-Unet[2] 中，提出了一种新的上采样方法 patch expanding。patch expanding 包括一个 linear projection layer 和一个 rearrange layer，linear projection layer 的作用是将 $X_{D_i} \subset R^{(H/2^{6-i})\times(W/2^{6-i})\times(2^{4-i}C)}$ 的特征维度由 $2^{4-i}C$ 变为 $2^{5-i}C$ 。rearrange layer 的作用是增大 feature map 的尺寸，由 $H/2^{6-i}\times W/2^{6-i}$ 变为 $H/2^{5-i}\times W/2^{5-i}$，并且将通道维度由 $2^{5-i}C$ 变为 $2^{3-i}C$。patch expanding 输出的 feature map $X_{D_i}^{U} \subset R^{(H/2^{5-i})\times(W/2^{5-i})\times(2^{3-i}C)}$。论文对 patch expanding 做了一些小小的改变，并将新的上采样方法命名为 patch reshaping, 如图 2-8所示。patch reshaping 也包括一个 linear projection layer 和一个 rearrange layer。在 linear projection layer 中，与 patch expanding 不同的是，feature map 的通道维度并没有发生改变。在 rearrange layer 中，feature map 的分辨率从 $H/2^{6-i}\times W/2^{6-i}$ 变为 $H/2^{5-i}\times W/2^{5-i}$ ，通道维度从 $2^{4-i}C$ 变为 $2^{2-i}C$ 。patch reshaping 输出的 feature map $X_{D_i}^{U} \subset R^{(H/2^{5-i})\times(W/2^{5-i})\times(2^{2-i}C)}$ 。

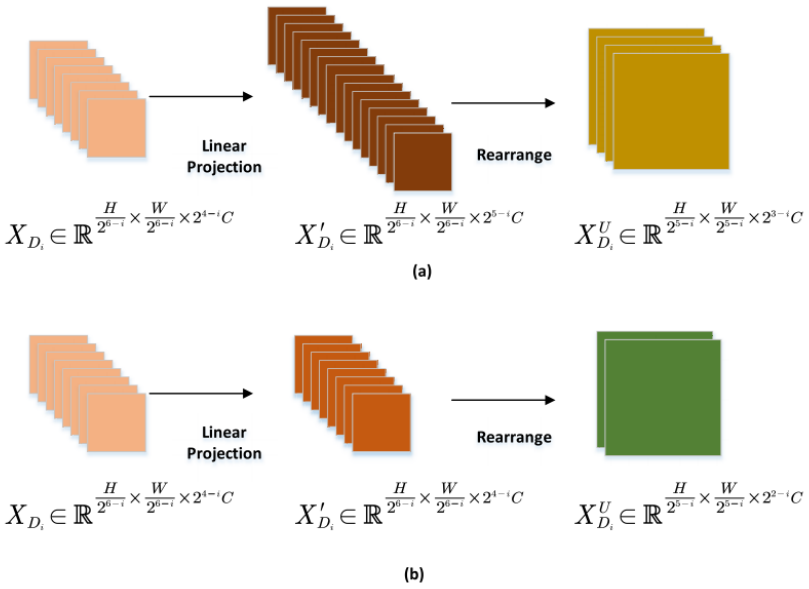

图 2-8　(a) patch expanding (b)patch reshaping

这一部分的改动一方面减少了计算复杂度，因为在 patch reshaping 中的 linearprojection layer 并没有改变特征的维度，另一方面，在消融实验中发现 patch reshaping 的效果比 patch expanding 的效果更好，因为 patch reshaping 输出的冗余信息比 patch expanding 更少，更有利于 change information 的提取。

– **Merging Block**　输入到 decoder 中的特征进行几次下采样操作后，部分空间信息和细节信息丢失，之前的方法比如说，FC-EF,FC-Siam-conc 和 FC-Siam-diff[3] 通过引入由 encoder 生成 $X^*_{E_{3-i+1}}$ 的到 decoder 中，来恢复丢失的信息。SwinSUNet 引入由 encoder 生成的 $X^1_{E_{3-i+1}}$ 和 $X^2_{E_{3-i+1}}$ 和 $X^U_{D_i}$ 进行融合，并且在 merging block 中加入了通道注意力模块 [8]。首先，merging block 使用一个 concatenation layer 对由 encoder 生成的 $X^1_{E_{3-i+1}}$ 和 $X^2_{E_{3-i+1}}$ 和 $X^U_{D_i}$ 进行融合，得到的 feature map 定义为 $X^cat_{D_i} \subset R^{(H/2^{5-i}) \times (W/2^{5-i}) \times (2^{2-i} \times C)}$。因为 $X^cat_{D_i}$ 是由 $X^1_{E_{3-i+1}}$ 和 $X^2_{E_{3-i+1}}$ 和 $X^U_{D_i}$ 在 channel-wise 拼接得到的，所以引入了通道注意力机制来更好的在 channel-wise 对特征进行深度融合。最后，merging block 使用一个 linear projection layer 来对 feature map 进行降维，通道维度从 $2^{2-i} \times 5C$ 变为 $2^{3-i}C$

2.3 USSFC-Net 算法

2.3.1 USSFC-Net motivation

Lei et al. 在论文 [10] 中提出了 USSFC-Net。用于解决现有的深度卷积神经网络在遥感变化检测任务中的两个问题：首先，现有的多尺度特征融合方法往往采用冗余特征提取和融合策略，这导致了变化检测模型需要较高的计算成本和内存占用。其次，变化检测中的常规的注意力机制难以同时对空间和谱间特征进行建模并生成三维注意力权重，这忽略了空间特征和谱间特征之间的协同作用。提出的 USSFC-Net 主要有两个优势。首先，设计了一种多尺度解耦卷积 (MSDConv)，它与目前流行的空洞空间金字塔池化（ASPP）[5] 模块及其变体明显不同，它可以通过循环多尺度的卷积灵活地捕捉变化对象的多尺度特征。同时，MSDConv 的解耦设计可以大大减少参数的数量和计算冗余。其次，引入了一种高效的空谱特征协同策略（SSFC），从而获得更加丰富的特征。SSFC 不同于现有的注意力机制，因为它不需要添加任何参数就能够推理出三维空谱注意力权重。

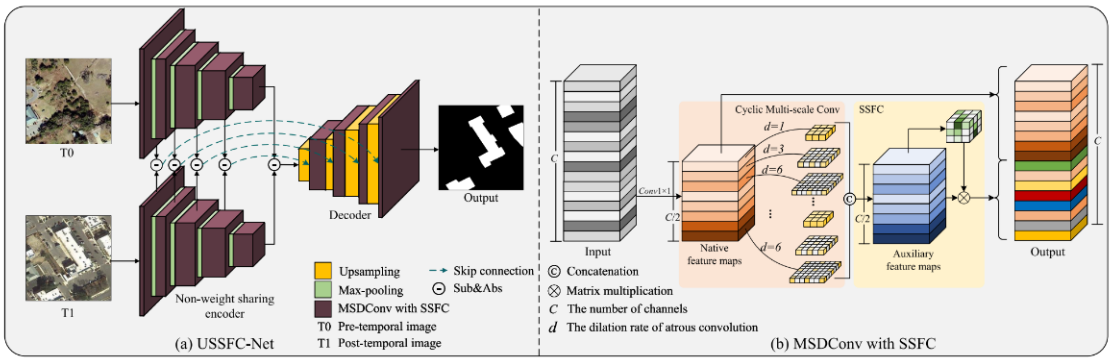

图 2-9 (a) 总体架构：将一对双时相图像输入到非权重共享的编码器中，并将每个阶段的差分图像融合到解码器。基于 SSFC 策略的 MSDConv 作为特征提取器的基本组成部分。(b) 基于 SSFC 策略的 MSDConv。MSDConv 可以捕获变化对象的多尺度特征表示。SSFC 为 MSDConv 生成了更丰富的特性。两者可以协同提高变化检测精度。

2.3.2 USSFC-Net pipeline

如图 2-9所示，USSFC-Net 由一个双分支非权重共享编码器和解码器组成。首先，我们将一组双时相图像分别输入由 MSDConv 和 SSFC 组成的双分支编码器中进行特征提取。在此阶段，每个 MSDConv 块可以有效地捕获双时相图像的多尺度特征。为了丰富 MSDConv 生成的特征，我们在空间相关性扩展阶段使用 SSFC 策

略进行特征增强。接下来是解码器，它由反卷积上采样层和使用提出的 MSDConv 的特征恢复层组成。在编码器的每个阶段，我们获取一个差分图像，并将其连接到解码器的相应位置，以获得更丰富的变化对象的特征映射。最后，网络通过点卷积进行降维和归一化操作，输出最终的变化检测结果。下面对 USSFC-Net 的两个模块进行分析:

- **多尺度解耦卷积（MSDConv）** MSDConv 的结构如图 2-9(b) 所示。MSDConv 的灵感来自于 Xception[6]，但又与之完全不同。MSDConv 不仅遵循 Xception 中提出的空间相关性和通道相关性可以充分解耦的结论，而且还实现了一种有效的空间相关性扩展策略。具体来说，设 X 为输入特征映射，$X \subset R^{C \times H \times W}$，其中 C、H、W 分别表示特征映射的通道数、特征映射的高度、特征映射的宽度。使用普通卷积生成特征图的过程可以表示为:

$$Y_{h,w,c'} = \sum_{i,j,c} W_{i,j,c,c'} \times \tilde{X_{h+i-1,w+j-1,c}} \tag{2-11}$$

由公式2-11可知，利用普通卷积学习特征时，参数量 P 和计算量 Q 为:

$$P = K \times K \times C \times C' \tag{2-12}$$

$$Q = K \times K \times C \times C' \times H \times W \tag{2-13}$$

为了减少普通卷积运算的参数冗余，USSFC-Net 提出了一种新的具有解耦空间和通道相关性的卷积运算。具体来说，为了获得 C_0 特征映射，我们首先通过点卷积生成 $C_0/2$ 原生特征映射。因此，生成的原生特征图不需要任何空间相关性映射，仅通过降维获得紧密的信道相关性。在第二阶段，我们使用循环多尺度卷积来扩展原生特征图的空间相关性，从而得到辅助特征图。如图 2-9(b) 所示，循环多尺度卷积由 (1,3,6) 等膨胀率组合的空洞卷积实现。值得注意的是，循环多尺度卷积将不同的膨胀率扩展到同一卷积层对应的卷积核上。这保证了 MSDConv 可以仅通过一个卷积层捕获变化对象的多尺度特征，并通过层的迭代融合多尺度特征。与公式2-11类似，膨胀率为 d 的循环多尺度卷积可表示为:

$$Y'_{h,w,c'} = \sum_{i,j,c} W'_{i,j,c,c'} \times \tilde{X_{h+id-1,w+jd-1,c}} \tag{2-14}$$

其参数量 P_m 和计算量 Q_m 为:

$$P_m = C \times C'/2 + K \times K \times C'/2 \tag{2-15}$$

$$Q_m = C \times C'/2 \times H \times W + K \times K \times C'/2 \times H \times W \tag{2-16}$$

- **空谱特征协同策略（SSFC）**为了得到注意力机制中查询（Query）和键（Key）之间的关系，论文受到 Nadaraya-Watson 核回归的启发，设计了一种基于高斯核的 SSFC 策略，它的推导过程为:

$$\tilde{Y} = \sum_{i=1}^{n} \frac{F(Q - K_i)}{\sum_{j=1}^{n} F(Q - K_j)} \times V_i \tag{2-17}$$

$$F(x) = e^{-\frac{x^2}{2}} \tag{2-18}$$

$$\tilde{Y} = \sum_{i=1}^{n} Softmax(-\frac{(Q - K_i)^2}{2}) \times V_i \tag{2-19}$$

$$\tilde{Y} = N(\omega(Q, K)) \times V_i \tag{2-20}$$

$$\tilde{Y} = Sigmoid(\frac{(Q - K)^2}{2\sigma^2} + \frac{1}{2}) \times V \tag{2-21}$$

在实际应用中，K 和 V 都是输入特征 $X \subset R^{C \times H \times W}$，其中 C、H 和 W 分别表示特征图的通道数、特征图的高度和特征图的宽度。Q 为通道维度 $X \subset R^{C \times 1 \times 1}$ 的均值，σ^2 为通道维度方差。σ^2 的值影响特征图的丰富度，σ^2 值越大，特征图的方差越大，对应的特征图中的上下文信息越丰富。为了使注意力权重为正易化，我们在原始注意力分数的基础上加上 1/2，用 Sigmoid 函数进行归一化，得到注意力权重。最后，将得到的注意力权重与输入特征图 X(Value) 相乘，得到输出特征图 Y。SSFC 策略如图 2-10所示。

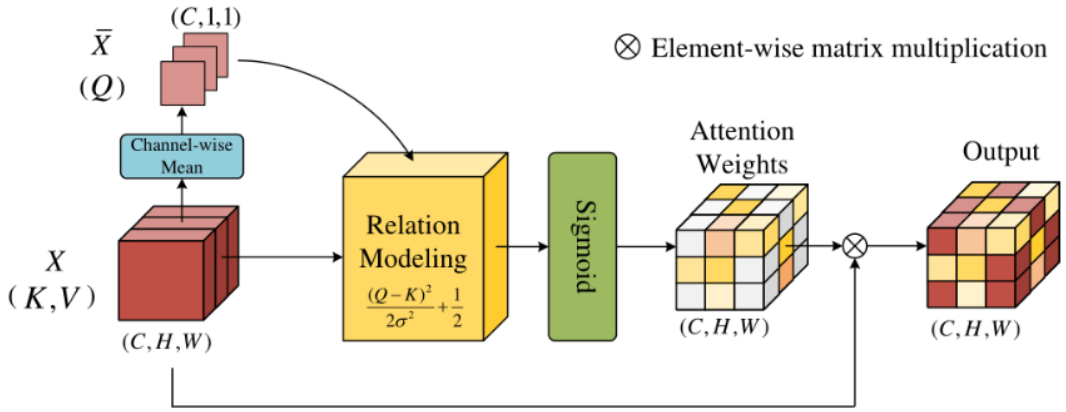

图 2-10　SSFC 策略流程图。该策略与目前流行的 2D 注意力机制明显不同，它不需要任何可学习参数，仅通过启发式计算在特征图上生成 3D 注意力权重。

本文提出的 SSFC 策略可以利用空谱特征协同的思想生成三维注意力权重。通过建立空间-谱间依赖关系，SSFC 可以增强遥感图像中变化目标的边缘和内部细节。与现有的注意力机制如 SE[8]、CBAM[16] 相比，SSFC 没有添加任何可学习参数，更加简单高效。最后，我们将 SSFC 嵌入到 MSDConv 中，如图 2-9(b) 所示。

2.4　ChangeFormer 算法

2.4.1　ChangeFormer Motivation

Bandara et al. 在论文 [1] 中提出了一种基于 Transformer 的孪生网络结构（ChangeFormer），用于从一对已配准的遥感图像中进行变化检测（change detection，CD）。目前基于全卷积网络的 CD 框架不同，该方法将分层结构的 Transformer 编码器与多层感知（MLP）解码器结合在一个孪生网络结构中，有效地呈现精确 CD 所需要的多尺度远程细节。

变化检测（CD）目的是检测不同时间获得的一对已配准图像的相关变化。目前的 CD 方法主要基于深度卷积网络 ConvNet，因为其具有强大的特征提取能力。注意力机制能捕获全局信息，但难以关联时空上的长距离信息。Transformer 网络具有更大的有效接受域（ERF）—在图像中任何一对像素之间提供比卷积神经网络强得多的上下文建模能力。近来，Tranformer 在 CV 方面（图像分类、分割）表现出其强大的性能，如 ViT、SETR、Swin、Twins 和 SegFormer 等。Transformer 结构与 ConvNet 编码器（ResNet18）结合使用，能增强特征表示，同时保持基于 ConvNet

的整体特征提取过程。本文表明，这种对 ConvNet 的依赖是非必须的，可以仅使用一个 Transformer 编码器和一个轻量 MLP 解码器完成变化检测任务。

2.4.2 ChangeFormer Pipeline

Changeformer 的网络 pipeline 如图 2-11所示，网络使用构成孪生网络的分层 Transformer 编码器（hierarchical transformer encoder）来提取双时间图像的粗粒度和细粒度特征；在双流网络的中间使用四个差异模块（difference modules）计算多尺度特征差异；最终使用一个轻量 MLP 解码器（lightweight MLP decoder）融合这些多尺度特征差异并预测出最终的 CD mask。

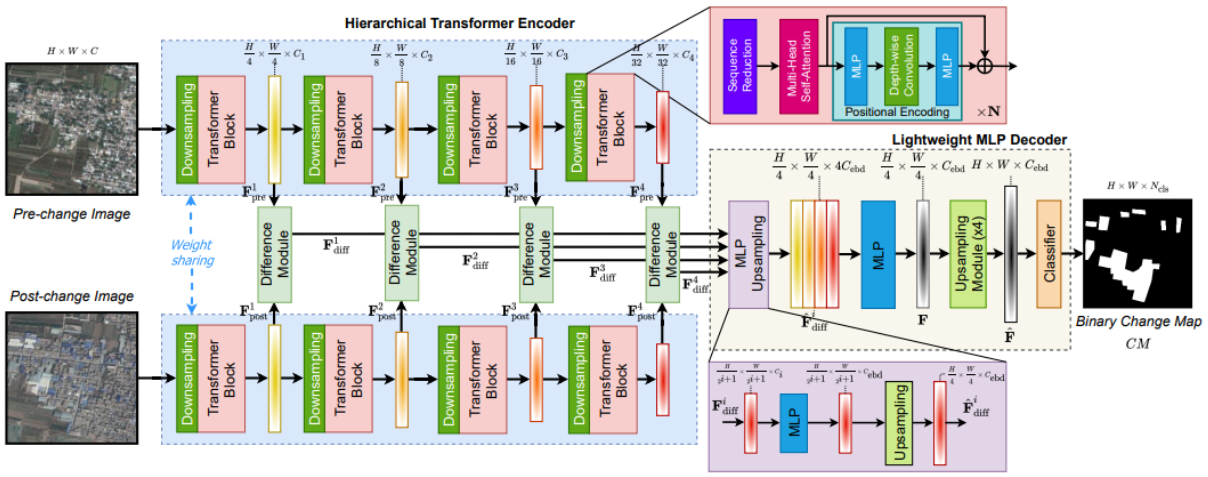

图 2-11　prediction head and loss function

下面逐模块对网络的架构进行分析：

- **Hierarchical Transformer Encoder** 多级 Transformer 编码器的输入为给双时间图像，多分辨率尺度提取多级特征，多级特征具有变化检测所需要的高分辨率和低分辨率特征。将提取的多级特征用 Difference Module 处理后传入 MLP 解码器获得变化特征。
- **Transformer Block** 为减少网络的计算量，首先使用 Sequence Reduction 对图像进行处理，缩小图像的分辨率。先利用序列约简比率 R 对输入 patch 进行 reshape，减小尺寸，扩展通道数，再对通道数进行线性映射到原始通道数。为了在特征中增加位置信息，使用两个 MLP 层和一个深度可分离卷积。不同于 ViT 的绝对位置编码，Changeformer 使用的是相对位置编码，可以在测试时使

用不同于训练时的分辨率的图像。

- **Downsampling** 每个 Transformer Block 之前都接着一个 Downsampling Block，对输入的 patch 进行下二分之一采样，减小尺寸，再经过 Transformer 提取多尺度特征。
- **Difference Module** 共有 4 个差异模块，接收来自孪生网络两条分支的 4 种不同尺寸的特征，concat 后进行卷积。该模块没有使用绝对差值，而是在训练过中学习每个尺度上的最优举例度量。
- **MLP Decoder** MLP 解码器主要由三个模块构成，首先利用 MLP 对多尺度差异特征进行处理，上采样到特定尺寸 $H/4 * W/4$。再将四个差异特征在通道维度进行拼接，再利用 MLP 层融合这些特征。随后借助转置卷积将融合的特征进行上采样到 H*W 并通过另一个 MLP 层处理 H*W*Ncls 的特征图，实现最终的分类。

3 实验结果

3.1 测试指标介绍

测试中，我们主要关注四个指标，分别为识别精度 Precision、召回率 Recall、F1 分数和 IoU。

- **精确率（Precision）**又叫查准率，表示预测结果为正例的样本中实际为正样本的比例。计算公式为:
  $Precision = \frac{TP}{TP+FP}$。
- **召回率（Recall）**又被称为查全率，表示预测结果为正样本中实际正样本数量占全样本中正样本的比例。计算公式为:
  $Recall = \frac{TP}{TP+FN}$。
- **F1 分数 (F1 Score)** 是精确率和召回率的一个加权平均。计算公式为:
  $F_1 = 2 * \frac{Precision*Recall}{Precision+Recall}$。
- **交并比 (Iou)** IoU 表示算法输出框与原标记框（ground truth bound）之间的交并比，定义为两个图形面积的交集和并集的比值。计算公式为:
  $IoU = \frac{|A \bigcap B|}{|A \bigcup B|}$

其中部分缩写含义如下：**TP**（True Positive）：被正确预测的正例。即该数据的真实值为正例，预测值也为正例的情况；**TN**（True Negative）：被正确预测的反例。即该数据的真实值为反例，预测值也为反例的情况;**FP**（False Positive）：被错误预测的正例。即该数据的真实值为反例，但被错误预测成了正例的情况；**FN**（False Negative）：被错误预测的反例。即该数据的真实值为正例，但被错误预测成了反例的情况。

3.2 实验结果与可视化

我们首先比较了模型的参数量和计算量。其中 STANet、SwinSUNet 和 ChangeFormer 在训练的时候输入的图像都是尺寸为 256*256 的 patch, 而 USSFCNet 则是直接以 LEVIR 未经处理的图像大小（1024*1024）输入。由于 USSFCNet 在网络中引入了可膨胀的空洞卷积，所以极大的减少了参数量，从而降低了高分辨率图像输入对算力的要求。

表 3-1　模型 flops 及 param

model	Params(M)	GFLOPs
SwinSUNet	40.96	11.20
ChangeFormer	39.17	144.60
USSFCNet	15.19	77.76

我们在 LEVIR 的测试集上对四个模型的性能进行了评估。 图 3-1是三个模型

表 3-2　LEVIR-CD 测试集测试结果

model	precision	recall	F1-score	Iou
SATNet	0.8385	0.9132	0.8753	0.7887
SwinSUNet	0.8807	0.9051	0.8927	0.8062
ChangeFormer	0.9205	0.8880	0.9040	0.8248
USSFCNet	0.9053	0.9222	0.9137	0.8410

表 3-3　WHU 测试集测试结果

model	precision	recall	F1-score	Iou
SwinSUNet	0.5466	0.6121	0.5775	0.4060
ChangeFormer	0.6957	0.7775	0.7343	0.5802
USSFCNet	0.4171	0.7968	0.5476	0.3770

在 LEVIR 数据集上测试图像的可视化结果。在 (a) 行图像 A,B 中，两张图像的光照强度变化比较明显. 在 B 中变化的建筑物处存在阴影的干扰，建筑物占据的面积越大，在建筑物附近产生的阴影面积越大，存在的干扰越明显。可以观察到三个网络均能够检测到尺寸较小的建筑，而对于尺寸比较大的建筑，因为阴影的干扰，导致 changeformer 和 ussfcnet 部分轮廓处检测失误，而 swinsunet 因为自身提取特征的强大能力，能够很好的识别出阴影干扰处的轮廓。

在 (b) 行图像 A,B 中，可以观察到道路上因为季节的变化，树木茂盛和凋零，导致两张图像中道路的宽窄并不一致，凭肉眼很容易被辨别为变化区域。而三个网络得到的结果和 GT 基本一致，说明了三个网络对于季节更替引起的树木茂盛程度的干扰因素均具有一定的鲁棒性。

在 (c) 行图像 A,B 中，变化目标在整张图片中占据的面积比例较小，而且分布的比较散乱，难以提供有效的结构信息。很明显的看出，SwinSUNet 和 ChangeFormer

表 3-4　SYSU-CD 测试集测试结果

model	precision	recall	F1-score	Iou
SwinSUNet	**0.7939**	**0.0054**	**0.0107**	**0.0054**
ChangeFormer	**0.7569**	**0.0076**	**0.015**	**0.0076**
USSFCNet	**0.8012**	**0.0016**	**0.0033**	**0.0016**

表 3-5　DSIFN-CD 测试集测试结果

model	precision	recall	F1-score	Iou
SwinSUNet	**0.7868**	**0.1073**	**0.1889**	**0.1043**
ChangeFormer	**0.8977**	**0.0330**	**0.0636**	**0.0328**
USSFCNet	**0.9251**	**0.0383**	**0.0736**	**0.0382**

对结构比较散乱且形状较小的目标的检测识别能力并不理想，而 ussfcnet 因为在网络中引入了多尺度的不同膨胀率的卷积核，所以能够有效的提取小尺度目标的特征。

我们分别评估了 SwinSUNet、ChangeFormer 和 USSFCNet 三个模型在 WHU、SYSU-CD 以及 DSIFN-CD 测试集上的四个指标，三个模型均在 LEVIR 数据集上进行训练，从而评估三个网络的泛化性能。

在 WHU 数据集上，因为原数据集为高分辨率图像，考虑到算力限制，我们将图像裁剪为 512*512 的图像块，并在三个模型上进行测试。由表3-3可知，三个模型中，ChangeFormer 的 precision 指标最高，SwinSUNet 次之，USSFCNet 最低。这与 ChangeFormer 和 SwinSUNet 使用了大量的 transformer block 有关，所以提取特征的能力比 USSFCNet 采用空洞卷积的方式更强。而因为 WHU 数据集和 LEVIR 数据分布差异较大，所以三个模型对于目标的召回率都不太理想。

在 SYSU-CD 数据集上，数据集作者已经将数据集划分为 256*256 的大小。由表3-4可知，三个模型在 SYSU-CD 测试集上的性能都不太理想，原因可能是 SYSU-CD 数据集中变化方式比较简单，而 LEVIR 数据集中变化方式种类多样，并且有光照，建筑阴影，校准误差等多因素影响，两个数据集的数据分布的 domain gap 过大，导致模型不能取得很好的测试结果。

由表3-5可知，在 DSIFN-CD 测试集上，USSFCNet 网络的 precision 性能最高，ChangeFormer 次之，SwinSUNet 最小。可能的一个原因是，DSIFN-CD 测试集中目标的尺寸变化比较大，而 USSFCNet 和 ChangeFormer 都引入了多尺度特征提取模

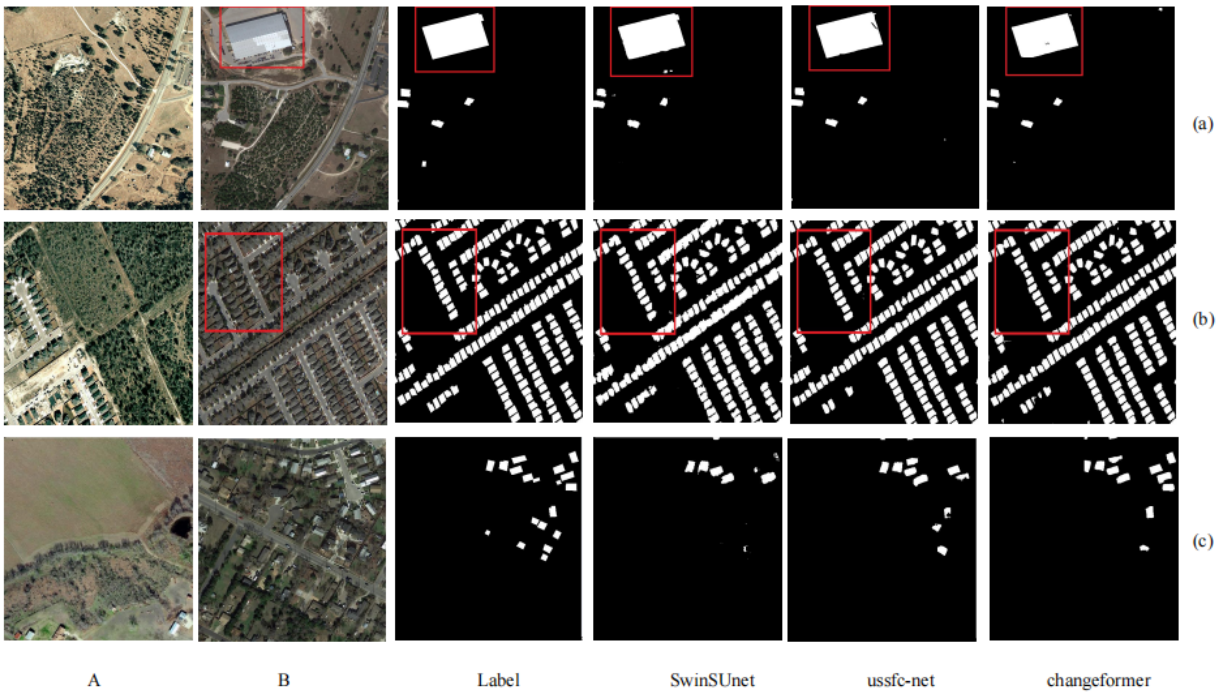

图 3-1　LEVIR-CD 测试集结果

块，其中 USSFCNet 在每一个卷积层中都引入了膨胀率为 1，3，6 的空洞卷积，感受野的范围比较大，所以 USSFCNet 提取小尺度目标特征的能力比较强。而因为两个数据集的数据分布的 domain gap 过大，导致三个模型均不能取得较好的召回率。

图 3-2是 SwinSUNet 和 ChangeFormer 两个模型在 WHU 数据集上的测试结果，可以很明显的发现，SwinSUNet 预测结果中目标出现了大量的空洞，而 ChangeFormer 对于一些比较简单的变化能够很好的拟合，而对于变化比较复杂，轮廓变化较大的目标并不能预测出来。

图图 3-3是 SwinSUNet 和 ChangeFormer 两个模型在 DSIFN-CD 数据集上的测试结果，因为两个数据集的数据分布的 domain gap 过大，导致模型不能取得很好的测试结果。

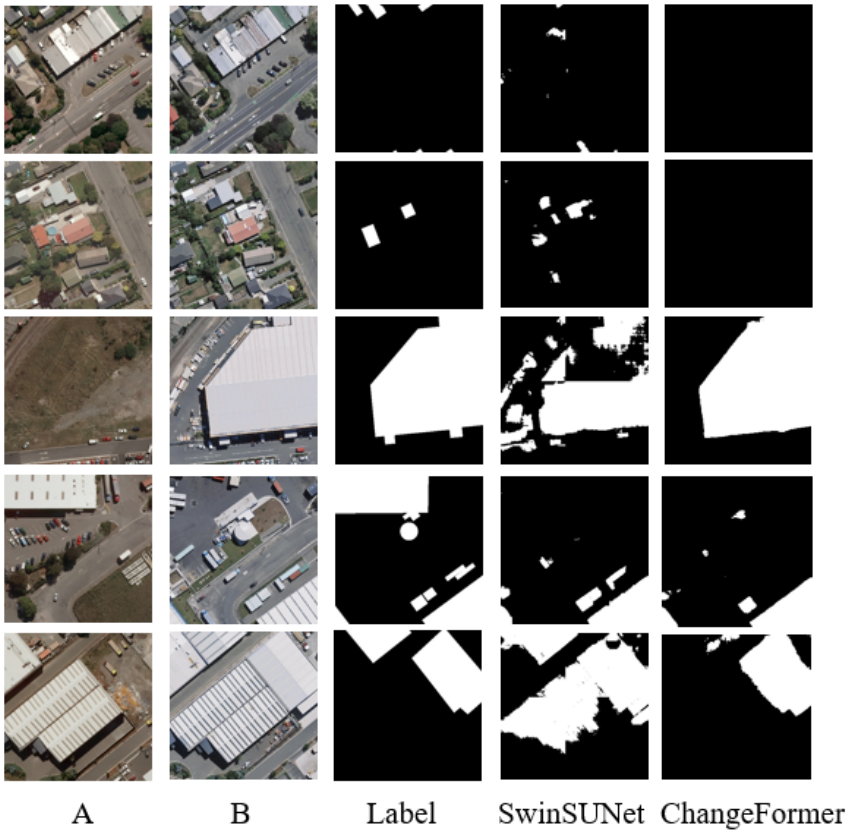

图 3-2　WHU 测试集结果

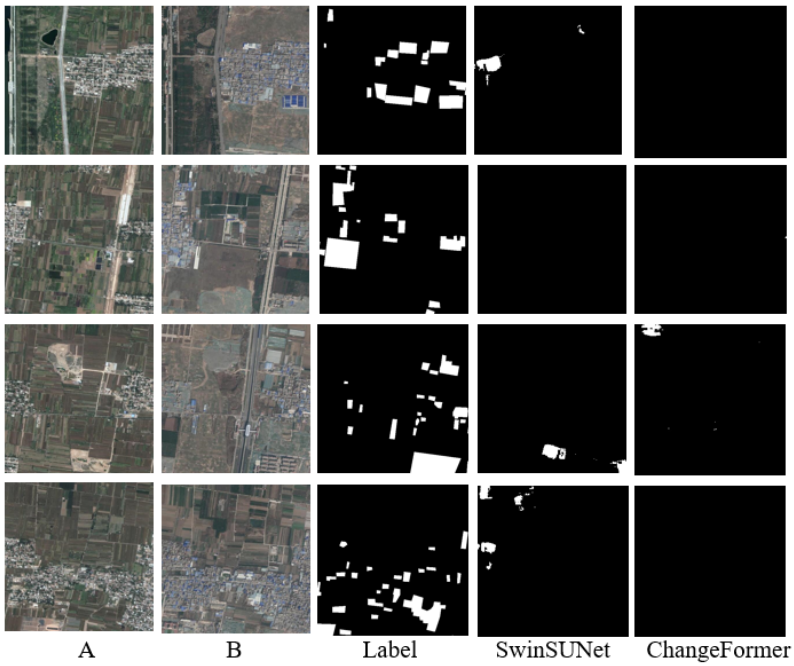

图 3-3　DSIFN-CD 测试集结果

3.3 失败案例及分析

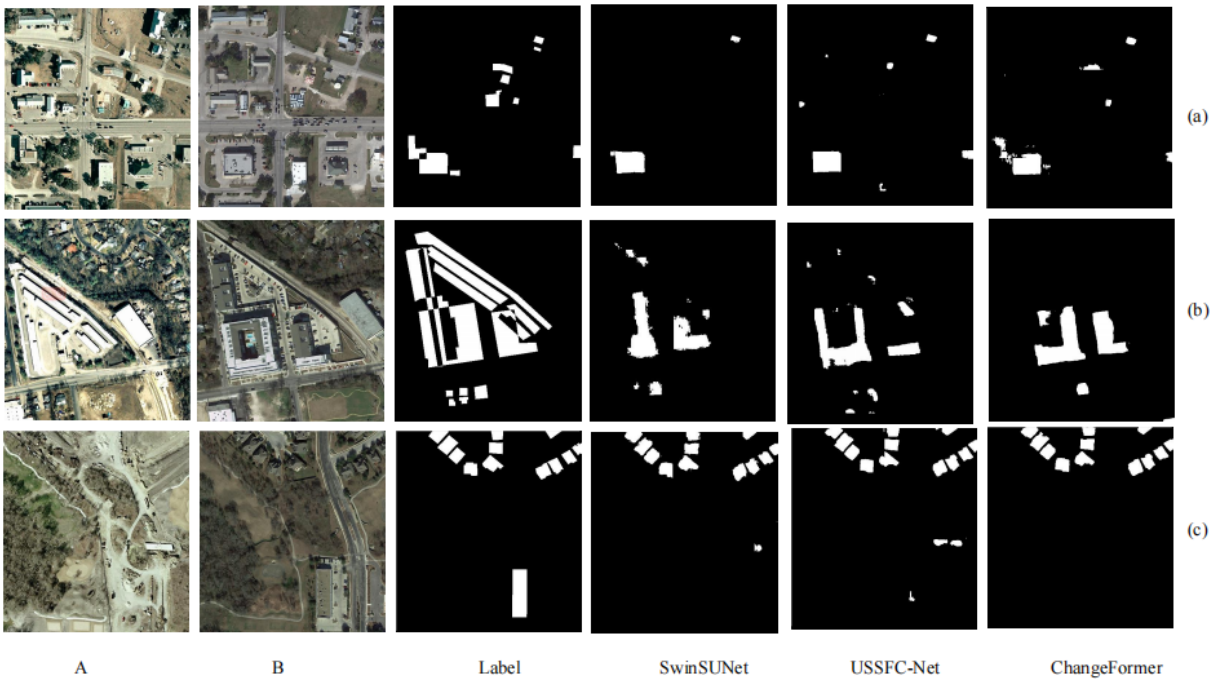

图 3-4 LEVIR-CD 失败案例

图 3-4是 SwinSUNet、USSFCNet 和 ChangeFormer 三个模型在 LEVIR 上的预测的一些失败案例。

在 (a) 行 A,B 中，同时存在因光线照射造成的阴影和微小目标，swinsunet 可以很好的避免阴影造成的干扰，但是对于微小目标的检测能力并不太理想，并且右上方还出现了误检问题。ussfcnet 因为网络结构中设置了不同膨胀率的空洞卷积，所以感受野的变化范围比较大，可以很好的检测出微小目标，但是左下方的目标受到阴影的干扰，所以无法准确划分边界。

在 (b) 行 A,B 中，规则建筑物检测结果三个网络均出现了空洞，通过用画图软件进行排查，发现规则建筑物变化区域未被检测的原因是这一部分存在过曝现象，RGB 值均达到了 254，255，这样因为位宽的限制，导致部分信息丢失，网络无法捕捉有用的特征，所以无法进行有效的检测。

在 (c) 行 A,B 中，右下方有矩形区域，而三个网络均没有检测到，通过观察 A,B 两图发现，在轮廓线的内外两侧的像素值基本没有变化，只是在轮廓处像素值发生了突变。可能三个网络可能认为这种突变是由噪声产生的无关干扰。经过观察，发现 A,B 两图以公路为参照点并未校准对齐，猜测可能是由于像素级的偏移

导致 ussfcnet 出现了误检现象。

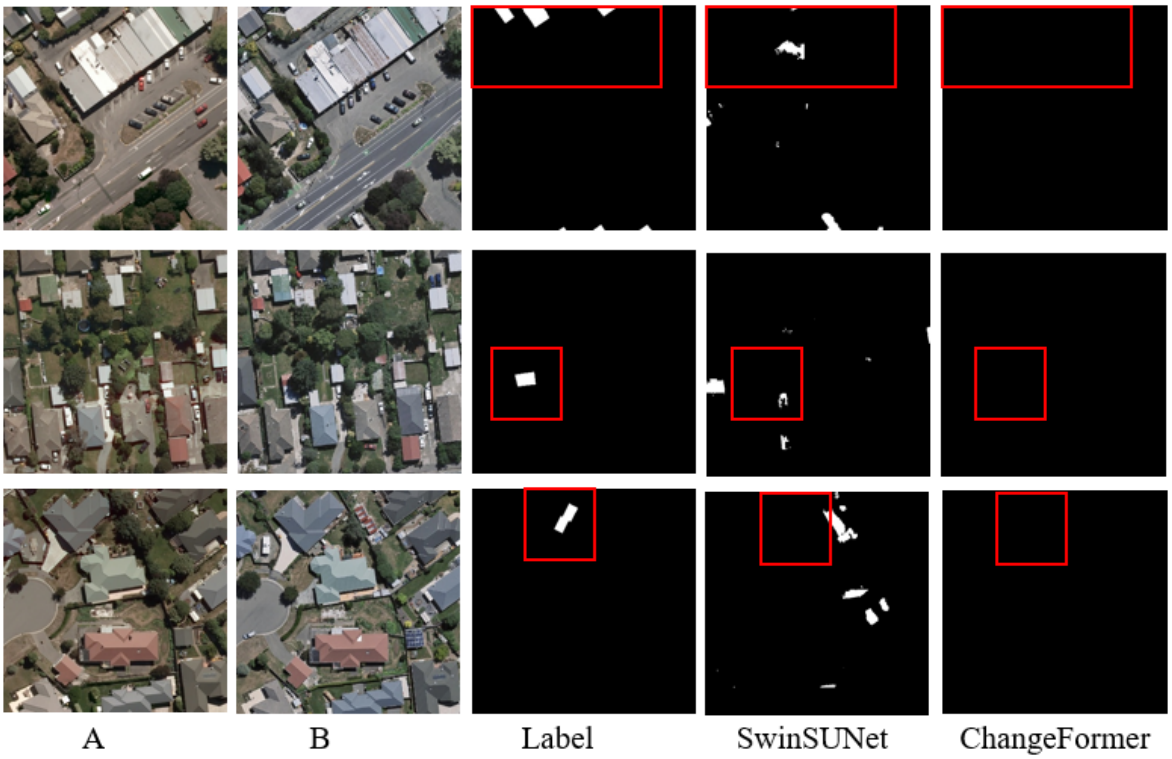

图 3-5　WHU 失败案例

图 3-5是 SwinSUNet 和 ChangeFormer 两个模型在 WHU 上的预测的一些失败案例。因为 LEVIR 与 WHU 数据集拍摄的地点不同，导致图像中变化目标的类别差异较大。并且 LEVIR 的数据拍摄时间跨度为 5-14 年，而 WHU 的数据拍摄跨度为 4 年，并且在这期间，WHU 拍摄区域还发生了地震，地震对于建筑物的变化造成了很大的影响，导致两个数据集之间的 domain gap 比较大，模型在 WHU 数据集上并不能取得很好的结果。

4 课设总结

遥感数据在地表变化检测中发挥着重要作用，尤其在城市规划、环境监测和农业调查等领域。然而，这项任务面临着挑战，如光照角度、阴影、地面物体的反射、植被生长等多种因素对最终的检测结果产生影响。

在当前研究背景下，多尺度特征融合方法在变化检测中备受关注，但存在一些限制，如高计算成本和内存占用。此外，现有的注意力机制往往难以有效同时对空间和谱间特征进行建模。因此，学者们致力于发展更具鲁棒性和适用性的模型。

在课程设计中，我们探索了多种前沿模型，从 STANet 到 SwinSUNet，再到 USSFC-Net 和 ChangeFormer。这些模型均致力于解决光照、阴影等随机因素对检测结果的干扰，并提高模型的鲁棒性和泛化能力。

目前，变化检测领域对于提高模型鲁棒性、减少计算成本以及增强模型对多种环境干扰的适应能力有着迫切需求。最终，我们在 LEVIR-CD 测试集上相较于 baseline STANet 取得了显著的改进，在测试集上达到了 F1score =0.9137 IOU=0.8410 的指标，并在其他测试集上进行了 zero−shot 测试，尽管零样本迁移测试的效果并不令人满意，但这也为我们未来的研究提供了一个值得前进的方向。

致　　谢

从大三开始学习计算机视觉课程，到现在能够独立的对应用任务进行调研，调试算法，在这个过程中，我们收获了许多了，也成长了许多。这一路上所经历的酸甜苦辣，仿佛都融入本文之中。这是我们人生中一段无比重要的时光，也是一段无比充实的时光。值此论文完成之际，谨向曾经给予我们关怀、帮助和支持的老师，同学，朋友们致以最诚挚的谢意！

衷心感谢学识渊博，风趣幽默的桑农教授！他以敏锐的眼光和独特的视角从计算机视觉发展的脉络出发，从人类的视觉感知过程出发授予我们知识，让我们深刻领悟了计算机视觉传统方法和深度学习两种方法的原理，从而激发我们对于科研的热情！在科研方面，他严谨踏实的治学态度，坚实渊博的学术素养，诲人不倦的学者风范深深地影响着我，为我一生的工作和学习树立了良好的榜样！

衷心感谢严于律己，平易近人的高常鑫教授！他对于科研的激情不断激励着我向更高的学术标准看齐；他在学术上的丰富经验不断地为我提供解决困难的方法；他提出的创新且有效的好主意不断地刷新我看待问题的角度和方式。更为重要的是，高老师严谨的学术态度、高效的工作方法深深影响着我，教会了我如何做一名合格的学术研究者。

衷心感谢严谨治学，亦师亦友的王岳环教授！他经常就课题细节和与同学们进行认真而热烈的讨论。从授课的过程中，他一点一滴地教会了我做研究的方法和态度。他无微不至的指导，赋予了我在科研领域前进的动力。

衷心感谢在这个过程中和我一起前进的队友！感谢你在我沮丧的时候不断的鼓励我，让我重拾坚持下去的决心。感谢你在我在我迷茫无措的时候，向我提出你真诚的建议。感谢你能够包容我的缺点，并陪伴我不断努力坚持下去！

参考文献

[1] Wele Gedara Chaminda Bandara and Vishal M Patel. A transformer-based siamese network for change detection. In *IGARSS 2022-2022 IEEE International Geoscience and Remote Sensing Symposium*, pages 207–210. IEEE, 2022.

[2] Hu Cao, Yueyue Wang, Joy Chen, Dongsheng Jiang, Xiaopeng Zhang, Qi Tian, and Manning Wang. Swin-unet: Unet-like pure transformer for medical image segmentation, 2021.

[3] Rodrigo Caye Daudt, Bertr Le Saux, and Alexandre Boulch. Fully convolutional siamese networks for change detection. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 4063–4067, 2018.

[4] Hao Chen and Zhenwei Shi. A spatial-temporal attention-based method and a new dataset for remote sensing image change detection. *Remote Sensing*, 12(10):1662, 2020.

[5] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *CoRR*, abs/1802.02611, 2018.

[6] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.

[8] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507, 2017.

[9] R D Johnson and ES Kasischke. Change vector analysis: A technique for the multispectral monitoring of land cover and condition. *International journal of remote sensing*, 19(3):411–426, 1998.

[10] Tao Lei, Xinzhe Geng, Hailong Ning, Zhiyong Lv, Maoguo Gong, Yaochu Jin, and

Asoke K. Nandi. Ultralightweight spatial–spectral feature cooperation network for change detection in remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–14, 2023.

[11] 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. *CoRR*, abs/2103.14030, 2021.

[12] José A Malpica, María C Alonso, Francisco Papí, Antonio Arozarena, and Alex Martinez De Agirre. Change detection of buildings from satellite imagery and lidar data. *International Journal of Remote Sensing*, 34(5):1652–1675, 2013.

[13] Ashbindu Singh. Review article digital change detection techniques using remotely-sensed data. *International journal of remote sensing*, 10(6):989–1003, 1989.

[14] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *CoRR*, abs/2012.12877, 2020.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[16] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: convolutional block attention module. *CoRR*, abs/1807.06521, 2018.

[17] Yang Zhan, Kun Fu, Menglong Yan, Xian Sun, Hongqi Wang, and Xiaosong Qiu. Change detection based on deep siamese convolutional network for optical aerial images. *IEEE Geoscience and Remote Sensing Letters*, 14(10):1845–1849, 2017.

附录

USSFC-Net.py : 定义 USSFC 网络主要模块

```python
import torch.nn as nn
import torch
from thop import profile
from torchsummary import summary

from networks.modules.MSDConv_SSFC import MSDConv_SSFC


class First_DoubleConv(nn.Module):
    def __init__(self, in_ch, out_ch):
        super(First_DoubleConv, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(in_ch, out_ch, kernel_size=3, padding=1),
            nn.BatchNorm2d(out_ch),
            nn.ReLU(inplace=True),
            nn.Conv2d(out_ch, out_ch, kernel_size=3, padding=1),
            nn.BatchNorm2d(out_ch),
            nn.ReLU(inplace=True)
        )

    def forward(self, input):
        return self.conv(input)


class DoubleConv(nn.Module):
    def __init__(self, in_ch, out_ch):
        super(DoubleConv, self).__init__()
        self.Conv = nn.Sequential(
            MSDConv_SSFC(in_ch, out_ch, dilation=3),
            nn.BatchNorm2d(out_ch),
            nn.ReLU(inplace=True),
            MSDConv_SSFC(out_ch, out_ch, dilation=3),
            nn.BatchNorm2d(out_ch),
            nn.ReLU(inplace=True)
        )

    def forward(self, input):
        return self.Conv(input)


class USSFCNet(nn.Module):
    def __init__(self, in_ch, out_ch, ratio=0.5):
        super(USSFCNet, self).__init__()

        self.Maxpool = nn.MaxPool2d(kernel_size=2, stride=2)
```

```
46
47          self.Conv1_1 = First_DoubleConv(in_ch, int(64 * ratio))
48          self.Conv1_2 = First_DoubleConv(in_ch, int(64 * ratio))
49          self.Conv2_1 = DoubleConv(int(64 * ratio), int(128 * ratio))
50          self.Conv2_2 = DoubleConv(int(64 * ratio), int(128 * ratio))
51          self.Conv3_1 = DoubleConv(int(128 * ratio), int(256 * ratio))
52          self.Conv3_2 = DoubleConv(int(128 * ratio), int(256 * ratio))
53          self.Conv4_1 = DoubleConv(int(256 * ratio), int(512 * ratio))
54          self.Conv4_2 = DoubleConv(int(256 * ratio), int(512 * ratio))
55          self.Conv5_1 = DoubleConv(int(512 * ratio), int(1024 * ratio))
56          self.Conv5_2 = DoubleConv(int(512 * ratio), int(1024 * ratio))
57
58          self.Up5 = nn.ConvTranspose2d(int(1024 * ratio), int(512 * ratio), 2,
                stride=2)
59          self.Up_conv5 = DoubleConv(int(1024 * ratio), int(512 * ratio))
60
61          self.Up4 = nn.ConvTranspose2d(int(512 * ratio), int(256 * ratio), 2,
                stride=2)
62          self.Up_conv4 = DoubleConv(int(512 * ratio), int(256 * ratio))
63
64          self.Up3 = nn.ConvTranspose2d(int(256 * ratio), int(128 * ratio), 2,
                stride=2)
65          self.Up_conv3 = DoubleConv(int(256 * ratio), int(128 * ratio))
66
67          self.Up2 = nn.ConvTranspose2d(int(128 * ratio), int(64 * ratio), 2,
                stride=2)
68          self.Up_conv2 = DoubleConv(int(128 * ratio), int(64 * ratio))
69
70          self.Conv_1x1 = nn.Conv2d(int(64 * ratio), out_ch, kernel_size=1,
                stride=1, padding=0)
71
72      def forward(self, x1, x2):
73          # encoding
74          # x1, x2 = torch.unsqueeze(x1[0], dim=0), torch.unsqueeze(x1[1], dim
                =0)
75          c1_1 = self.Conv1_1(x1)
76          c1_2 = self.Conv1_2(x2)
77          x1 = torch.abs(torch.sub(c1_1, c1_2))
78
79          c2_1 = self.Maxpool(c1_1)
80          c2_1 = self.Conv2_1(c2_1)
81          c2_2 = self.Maxpool(c1_2)
82          c2_2 = self.Conv2_2(c2_2)
83          x2 = torch.abs(torch.sub(c2_1, c2_2))
84
85          c3_1 = self.Maxpool(c2_1)
86          c3_1 = self.Conv3_1(c3_1)
87          c3_2 = self.Maxpool(c2_2)
88          c3_2 = self.Conv3_2(c3_2)
89          x3 = torch.abs(torch.sub(c3_1, c3_2))
```

```
90
91          c4_1 = self.Maxpool(c3_1)
92          c4_1 = self.Conv4_1(c4_1)
93          c4_2 = self.Maxpool(c3_2)
94          c4_2 = self.Conv4_2(c4_2)
95          x4 = torch.abs(torch.sub(c4_1, c4_2))
96
97          c5_1 = self.Maxpool(c4_1)
98          c5_1 = self.Conv5_1(c5_1)
99          c5_2 = self.Maxpool(c4_2)
100         c5_2 = self.Conv5_2(c5_2)
101         x5 = torch.abs(torch.sub(c5_1, c5_2))
102
103         # decoding
104         d5 = self.Up5(x5)
105         d5 = torch.cat((x4, d5), dim=1)
106         d5 = self.Up_conv5(d5)
107
108         d4 = self.Up4(d5)
109         d4 = torch.cat((x3, d4), dim=1)
110         d4 = self.Up_conv4(d4)
111
112         d3 = self.Up3(d4)
113         d3 = torch.cat((x2, d3), dim=1)
114         d3 = self.Up_conv3(d3)
115
116         d2 = self.Up2(d3)
117         d2 = torch.cat((x1, d2), dim=1)
118         d2 = self.Up_conv2(d2)
119
120         d1 = self.Conv_1x1(d2)
121         out = nn.Sigmoid()(d1)
122
123         return out
124
125
126 if __name__ == "__main__":
127     A2016 = torch.randn(1, 3, 256, 256)
128     A2019 = torch.randn(1, 3, 256, 256)
129     model = USSFCNet(3, 1, ratio=0.5)
130     out_result = model(A2016, A2019)
131     summary(model, [(3, 256, 256), (3, 256, 256)])
132     flops, params = profile(model, inputs=(A2016, A2019))
133     print(flops, params)
```

metrics.py：模型的性能度量函数定义

```
1 import torch
2 import numpy as np
3 from sklearn.metrics import confusion_matrix
```

```
4
5
6
7  def ConfusionMatrix(num_classes, pres, gts):
8      def __get_hist(pre, gt):
9          pre = pre.cpu().detach().numpy()
10         gt = gt.cpu().detach().numpy()
11         pre[pre >= 0.5] = 1
12         pre[pre < 0.5] = 0
13         gt[gt >= 0.5] = 1
14         gt[gt < 0.5] = 0
15         mask = (gt >= 0) & (gt < num_classes)
16         label = num_classes * gt[mask].astype(int) + pre[mask].astype(int)
17         hist = np.bincount(label, minlength=num_classes ** 2).reshape(
               num_classes, num_classes)
18         return hist
19
20     cm = np.zeros((num_classes, num_classes))
21     for lt, lp in zip(gts, pres):
22         cm += __get_hist(lt.flatten(), lp.flatten())
23     return cm
24
25
26 def get_score(confusionMatrix):
27     precision_2=confusionMatrix[1][1] / (confusionMatrix[1][1] +
           confusionMatrix[1][0] + np.finfo(np.float32).eps)
28     recall_2=confusionMatrix[1][1] / (confusionMatrix[1][1] +confusionMatrix
           [0][1] + np.finfo(np.float32).eps)
29     f1score_2 = 2 * precision_2 * recall_2 / ((precision_2 + recall_2) + np.
           finfo(np.float32).eps)
30     iou_2=confusionMatrix[1][1] / (confusionMatrix[1][1] +confusionMatrix
           [1][0]+confusionMatrix[0][1] + np.finfo(np.float32).eps)
31     precision = np.diag(confusionMatrix) / (confusionMatrix.sum(axis=0) + np.
           finfo(np.float32).eps)
32     recall = np.diag(confusionMatrix) / (confusionMatrix.sum(axis=1) + np.
           finfo(np.float32).eps)
33     f1score = 2 * precision * recall / ((precision + recall) + np.finfo(np.
           float32).eps)
34     iou = np.diag(confusionMatrix) / (
35             confusionMatrix.sum(axis=1) + confusionMatrix.sum(axis=0) - np.
                   diag(confusionMatrix) + np.finfo(
36         np.float32).eps)
37     po = np.diag(confusionMatrix).sum() / (confusionMatrix.sum() + np.finfo(np
           .float32).eps)
38     pe = (confusionMatrix[0].sum() * confusionMatrix[0:2][0].sum() +
           confusionMatrix[1].sum() * confusionMatrix[0:2][
39         1].sum()) / confusionMatrix.sum() ** 2 + np.finfo(np.float32).eps
40     kc = (po - pe) / (1 - pe + np.finfo(np.float32).eps)
41     return precision, recall, f1score, iou, kc,precision_2,recall_2,f1score_2,
           iou_2
```

```
42
43
44 def get_score_sum(confusionMatrix):
45     num_classes = confusionMatrix.shape[0]
46     precision, recall, f1score, iou, kc,precision_2,recall_2,f1score_2,iou_2 =
           get_score(confusionMatrix)
47     cls_precision = dict(zip(['precision_' + str(i) for i in range(num_classes
           )], precision))
48     cls_recall = dict(zip(['recall_' + str(i) for i in range(num_classes)],
           recall))
49     cls_f1 = dict(zip(['f1_' + str(i) for i in range(num_classes)], f1score))
50     cls_iou = dict(zip(['iou_' + str(i) for i in range(num_classes)], iou))
51     return cls_precision, cls_recall, cls_f1, cls_iou, kc,precision_2,recall_2
           ,f1score_2,iou_2
```

predict.py: 测试函数

```
1     from collections import OrderedDict
2  import numpy as np
3  import os
4  from torch.utils.data import DataLoader
5  from torchvision import transforms
6  from operation import predict
7  from path import *
8  import torch
9  from dataset import RsDataset
10 from networks.USSFCNet import USSFCNet
11 from thop import profile
12
13
14 src_transform = transforms.Compose([
15     transforms.ToTensor(),
16     # transforms.Resize(512),
17     transforms.Normalize([0.5], [0.5])
18 ])
19
20 label_transform = transforms.Compose([
21     transforms.ToTensor(),
22     # transforms.Resize(512),
23 ])
24 def count_param(model):
25     param_count = 0
26     for param in model.parameters():
27         param_count += param.view(-1).size()[0]
28     return param_count
29
30 model = USSFCNet(3, 1, ratio=0.5).to(device)
31 # model = torch.nn.DataParallel(model, device_ids=device_ids).to(device)
32 model_path = './ckps/USSFCNet_LEVIRCD/ckp_f1_9110.pth'
33 ckps = torch.load(model_path, map_location=device)
```

```
model.load_state_dict(ckps)
p_count = count_param(model)
print('params:', p_count)
input = torch.randn(1, 3, 1024, 1024).to(device)
flops, params = profile(model, inputs=(input, input,))
file_list=[]
print('flops:', flops, ' params:', params)
dataset_test = RsDataset(test_src_t1, test_src_t2, test_label, test=True,
                         t1_transform=src_transform,
                         t2_transform=src_transform,
                         label_transform=label_transform)
file_list=dataset_test.__getfile__()

dataloader_test = DataLoader(dataset_test,
                             batch_size=1,
                             shuffle=False)
pre_test, rec_test, f1_test, iou_test, kc_test,precision_2,recall_2,f1score_2,
    iou_2 = predict(model, dataloader_test,file_list)
print('test Pre:(%f,%f) test Recall:(%f,%f) test MeanF1Score:(%f,%f) test IoU
    :(%f,%f) test KC: %f' % (
    pre_test['precision_0'], pre_test['precision_1'], rec_test['recall_0'],
        rec_test['recall_1'], f1_test['f1_0'],
    f1_test['f1_1'], iou_test['iou_0'], iou_test['iou_1'], kc_test))
print("precision",precision_2,"recall",recall_2,"f1score",f1score_2,"iou",
    iou_2)
```