上海对外经贸大学

《数字图像处理》课程报告

报告题目： NASA 星云图像理解和分类

小组成员： 19024075 陶盛皿

完成日期： 2022 年 05 月 12 日

基本情况			
项目来源	Kaggle	完成时长	5.1 - 5.14
数据来源	Kaggle	数据集链接	https://worldview.earthdata.nasa.gov/ https://www.kaggle.com/competitions/understanding_cloud_organization/data
是否借鉴 开源代码	是	代码链接	https://github.com/milesial/Pytorch-UNet https://github.com/albumentations-team/albumentations
是否用于 参加相关竞赛	否	获奖情况	不适用

主要研究内容及创新点

【主要研究内容】

本实验主要研究内容包括：数字图像处理方法对于 Opencv 包和 albumentations 包应用和探索以及从传统神经网络跨越到用于图像分割深度网络模型的个人学习。本实验关注的核心是视觉领域图像分割问题的 U-Net 架构，从 U-Net 论文理论性内容迁移到实际实验过程中，完成论文学习和理解的闭环学习。此外，对深度学习领域中，广泛使用的 EfficientNet-b4 预训练模型的迁移学习和参数微调的方法进行实验。

【创新点】

本实验没有实质性的创新点，主要是用于个人学习，实验过程包含四个阶段（其中两次为不成功尝试）。传统卷积神经网络到 YoLov4 预训练模型，最后使用了基于 EfficientNet 预训练模型的 Encoder 的 U-Net 架构。并且通过实验了解了数字图像预处理的 Opencv 包和 albumentations 包。

阶段一：直接使用卷积神经网络进行图像特征提取并进行像素位置预测；

阶段二：使用 YoLov4 预训练模型用目标检测的算法进行图像分类检测；

阶段三：使用 U-Net 架构通过使用 EfficientNet-b4 的预训练模型进行迁移学习和参数微调；

阶段四：进行图像处理（包括：图像翻转、放射变换、线性插值、非刚体转换、亮度调整、尺寸裁剪、颜色空间转换等）进行数据增广，使用上阶段 baseline 进行图像分割。

Dice 系数从阶段一：0.22 到阶段四：0.52，总体提升：136%。

一、 研究背景及意义

a) 研究背景

气候变化问题是人类在全球范围内最关心的问题之一，在分析全球气候变化时，人们重点关注大气研究，因此云层的分析是气象分析的核心，浅层云距离地球最近，地球气候（降水情况）往往由浅层云决定，因此浅层云的图像理解和模式识别与气候预测紧密相关。

目前，云检测技术主要有两类：地表基站监测和遥感卫星观测。两种方法均利用模式识别与计算机视觉技术对星云图像理解，通过云分类和云识别精准识别不同天气系统。该技术能够为气候分析提供重要依据，并且指导人类气候预报，提高降水等天气预测的质量和预测台风等灾害的能力。

随着人工智能中计算机视觉领域的研究和技术的成熟，气象研究中模式识别从 04 年 BP 神经网络进行云层进行光谱亮度特征提取[1]进行云层分类转向使用深度网络进行云层图像分类。目前，对于云分类和云识别主要依赖图像分割技术，包括：阈值分割、边缘检测、区域分割、基于神经网络的图像分割、基于深度网络的图像分割。[2]由此，云识别和云检测准确度获得大幅度提升。

虽然检测准确度得到大幅度提升，但是由于深度网络模型依赖于大规模数据，而实际工业界卫星云图数据量相比之下规模较小、收集数据的成本高，加之云层形状具有复杂的、变化的多样性，深度网络的参数量会随着网络深度增加而使得参数量指数型增长，增加了训练时间、训练成本，因此，云识别的图像分割目前依旧是个研究重点和难点。

b) 研究意义

云识别在气象领域通常用于指导天气预报和气候变化监测，提高云识别的准确性意味着天气预报和气候监测的准确度，对于人类长期分析和监测气候变化以及短期的天气预报和气象灾害预报预警具有重要意义。而目前云识别技术准确度的提升依赖于使用的模型，云识别依旧是气象领域的难点和技术发展重点，因为因此具有极大研究价值。

二、 数据集介绍

本次实验中，原始数据图像来自于 NASA 网站，Kaggle 网站将数据进行处理分为包含人工标注的训练集和未标注的测试集。原始数据总量：5546，通过透视表进行数据预处理后发现，数据分为四类：Sugar、Gravel、Fish 和 Flower。“Sugar” 字面意思是云层形状像棉花糖，呈现出松散絮状布局；“Flower”字面意思是云层形状像繁花，呈现出聚集性的不同簇；“Fish” 字面意思是云层形状像鱼，呈现出条形的大面积覆盖；“Gravel”字面意思是云层形状像碎石，呈现出具有明显界限的块状的布局。

图表 1 花形

图表 2 鱼形

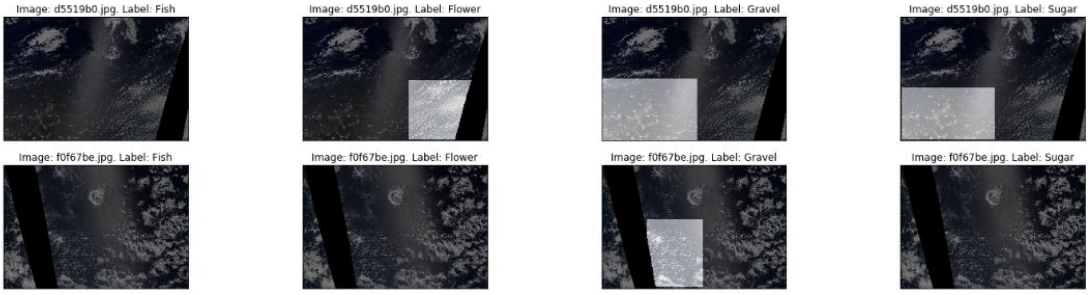
图表 3 糖形

图表 4 碎石形

而上述定义是通过人工观察提出的理想化的模式，通过数据透视表分析，发现四种类型分布不均匀，即在实际工业中一张卫星云层图片中，可能会出现多种类型的云，如下图所示。

图表 5 数据预处理：透视表

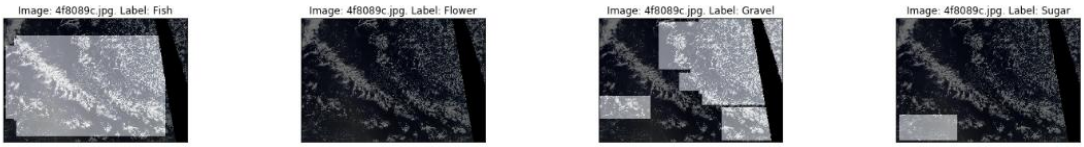
该数据集要解决的问题是通过建立模型进行多分类预测，将是识别（即预测）不同类型的云并进行图像分割，结果输出标注的像素位置信息。人工标注的数据可视化结果，如下图所示。



图表 5 人工标注的训练集

数据中存在的两个问题：

1) 由于是两颗不同极地轨道卫星捕获图像拼接而成，因此拼接部分图像损失以黑色掩蔽框进行标注。
2) 虽然训练集的数据标注区域在合理范围，但是像素位置信息基本为带有棱角的多边形形状并且具有重叠部分，如下图所示。



图表 6 人工标注中的重叠案例

三、 领域相关研究（800 字及以上）

a) 气象卫星云模式识别技术

气象卫星云模式识别的任务通常服务于气象预报领域，包括降水预测、气候灾害监测和预报。气象领域云识别技术随着计算机技术中模式识别技术的发展，特别是计算机视觉技术的兴起和成熟，从最初通过卫星云图或雷达图像的人工识别判断，转向气象卫星云模式识别当中。[3]通过对卫星云图进行图像处理和深度学习模型提取图像特征训练参数，大幅提升了云模式识别的准确度。就目前而言，云模式识别使用数据驱动的深度网络进行图像分割，训练过的深度学习模型往往能实现预测的实时性，适用于短中期的气象预报。目前瓶颈在于云图像的复杂性和云图像数据相较于其他领域的数据规模较小，云检测和云模式识别依旧是难点。

b) 图像分割技术

图像分割是计算机视觉研究中的重点之一，并且是图像理解领域热点。图像分割是现实业务中进行图像分析的第一步。过去传统图像分割利用数字图像处理技术、拓扑学、数学等方法进行计算，包括：基于阈值分割、区域分割、边缘检测等数字图像处理技术以及聚类、分类等机器学习算法为主。随着算力和计算机技术的发展，自 2010 年深度学习的崛起和神经网络模型的提出，图像分割进入深度学习阶段。

目前图像分割领域有全卷积神经网络（Fully Convolutional Network, FCN)、U-Net、空洞卷积神经网络（Dilated Convolutional Network)、基于特征增强和多尺度特征提取的卷积神经网络等，其中 FCN 该网络实现了端到端的逐像素分类（即语义分割)，是图像分割领域第一个深度学习模型，为后续网络模型的提出提到奠基作用。随后提出的 Efficient-Net 和 U-Net 显著提高了模型准确度和性能，目前工业界图像分割领域使用 U-Net 架构较多。

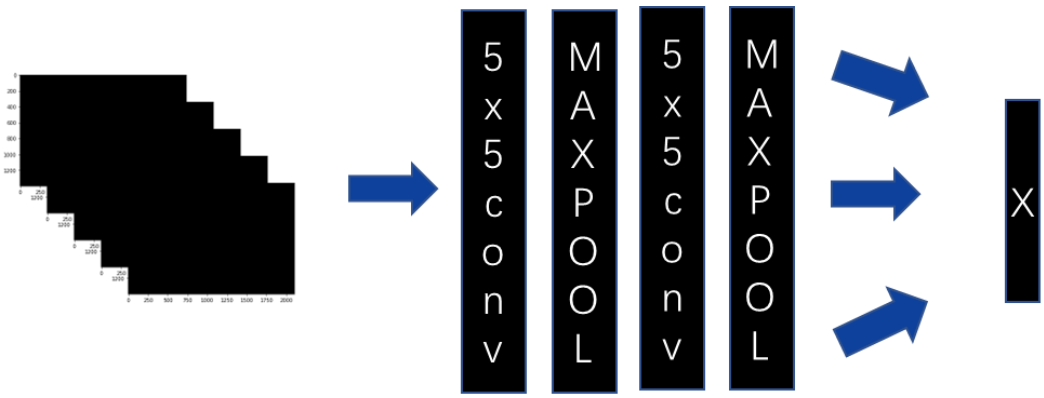
U-Net 是在生物医学图像分割应用广泛卷积神经网络，如脑图像分割和肝脏图像分割等[4]。该网络基于全卷积网络（FCN)，是其架构进行了修改和扩展，以使用更少的训练图像并产生更精确的分割。[5]

图像分割技术是计算机视觉发展最快的方向之一，目前以图像分割技术中语义分割的应用为研究热点，该技术未来展望将在机器人技术和自动驾驶技术等中广泛应用。

四、 本文方法介绍

本文根据气象学中模式识别技术的发展脉络进行实验，主要学习和体会技术发展深度学习模型的发展对于该领域应用的性能的提升程度，实验总共分为四个阶段（论述以 U-Net 架构方法为主)。

阶段一：使用卷积神经网络模型，通过使用 1 层全连接层和 2 层卷积层和最大池化层进行前向传播、图像特征提取，模型架构小金字塔型架构，激活函数为：sigmoid 函数。



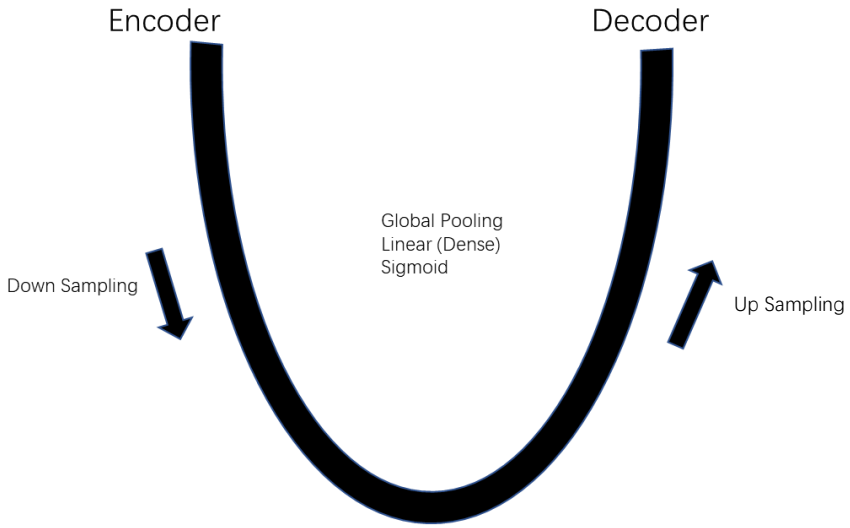
图表 7 阶段一：极简模型架构示意图

阶段二：通过使用 GitHub 网站 Git 克隆了 AlexeyAB 的 darknet 仓库中 YoLov4-tiny 进行部署和训练。模型总共 161 层，模型架构如图[6]。

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

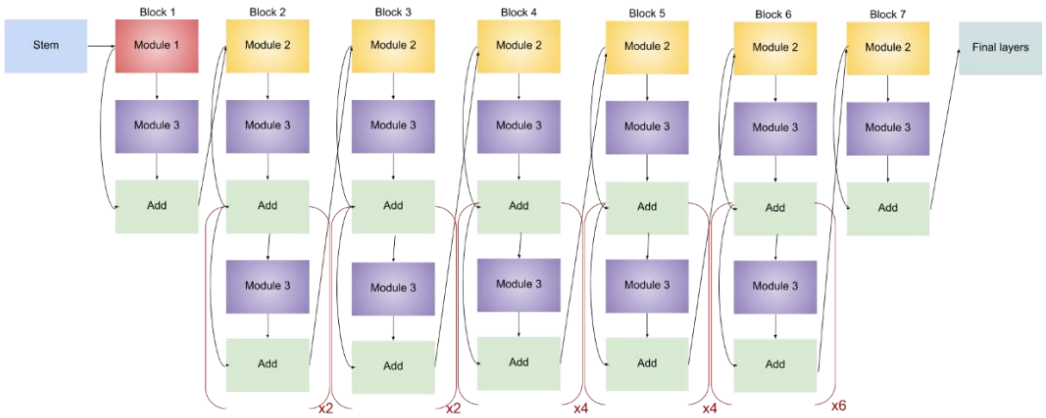
图表 8 YoLoV4 简易架构图

阶段三：使用 U-Net 架构，解码器部分使用 EfficientNet 系列模型进行迁移学习。总体模型架构如图。



图表 9 U-Net **架构简易图**

EfficientNet 系列模型是图片分类领域精度最高的模型，它使用了深度、宽度、输入图片分辨率共同调节模型。本实验使用的 EfficientNet-b4 作为编码器对输入图像进行特征提取。由于一张图像中可能包含多种云类型，因此需要根据提取的特征对每种云类型进行判断，再输入 U-Net 解码器进行图像分割。EfficientNet-b4 架构[7]如下图所示[1]。



图表 10 EfficientNet-b4

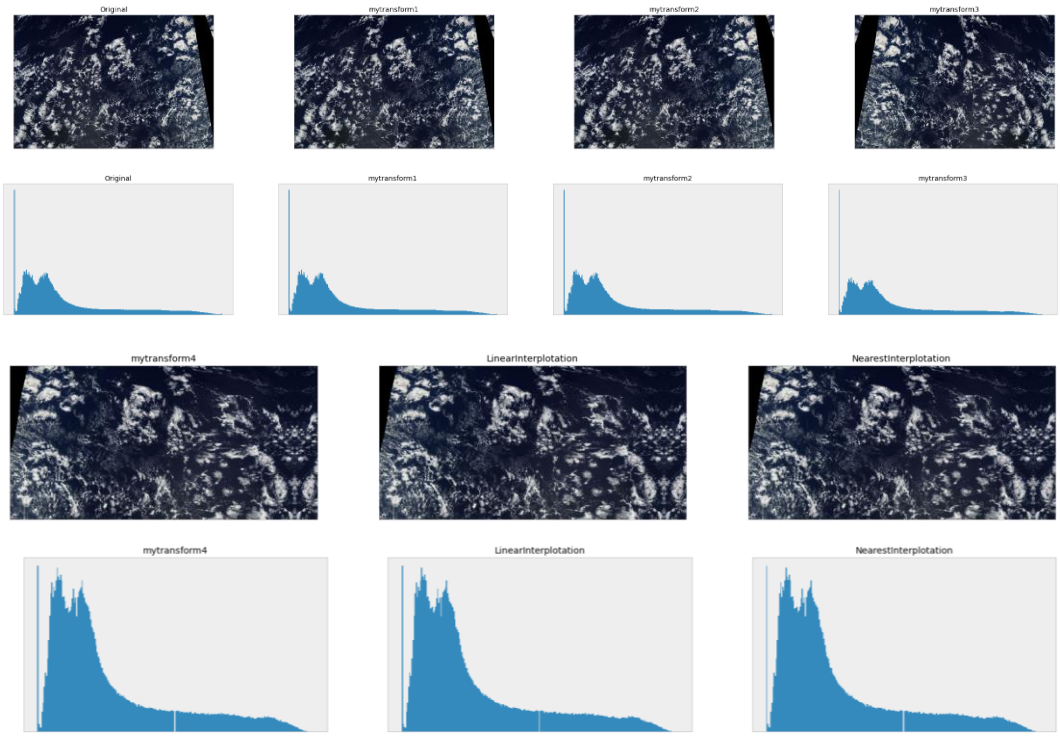
阶段四：由于在研究背景中提及的卫星云层图像往往数据规模不够大，本

[1] EfficientNet-b4 架构图源于网络，链接：https://blog.csdn.net/u011984148/article/details/108570902.

实验数据集的数据规模亦是较小，因此，本实验利用多种图像增强的方式，降低数据量这一因素的负面影响。此外，在数据集介绍中提及一张图像往往存在多种类型的云，并且这四类的判定界限不足够清晰，因此本实验对图像增加了对比度。此外，图像插值就是利用已知邻近像素点的灰度值或图像中的三色值来补充未知像素点的灰度值，以提高分辨率的图像。

在数字图像处理中，灰度直方图是一种计算代价非常小但却能有效统计出概括了一幅图像的灰度级信息。即每个灰度级在图像矩阵中的占有率的描述性统计。因为灰度级范围越大表明图像对比度越高，因此通过上述图像变换的算法调整图像的对比度。在第四种变换方式的基础上，使用双线性插值和最近邻插值方式效果较好，因此，实验采取最近邻插值方式。

其余实验中使用数字图像处理方法包括：图像翻转、放射变换、图像几何变换线性插值、亮度调整、尺寸裁剪、颜色空间转换等）进行数据增广。具体图像变换过程如下图所示。在进行一系列数字图像处理变换之后，使用上阶段 U-Net baseline 进行模型训练。



图表 11 图像变换处理过程可视化

图像分割的评分标准主要有以下四种：像素准确性（Pixel

accurancy)、交并比 (Intersection over Union, IoU)、平均交并比（Mean IoU）和 Dice 系数。本实验采用 Dice 系数来评估模型性能。Dice 系数取值在 0 至 1 之间，且数值越大，证明预测的像素集合与人工标记的像素集合重合度越高，预测模型越准确。

五、 实验环境及实验结果

a) 实验结果总结

Dice 系数从阶段一 0.22 到阶段四的 0.52，总体模型性能提升 136%。

b) 实验环境

平台	模型 / 方法	GPU 类型
Jupyter-lab	图像变换和可视化 训练集预处理	本地 CPU
Colab	卷积神经网络	Tesla T4 NVIDIA-SIM 460.32.03 CUDA Version：11.2
Colab	U-Net (编码器：EfficientNet-b4)	Tesla T4 NVIDIA-SIM 460.32.03 CUDA Version：11.2
Kaggle	YoLoV4	Tesla P100 NVIDIA-SIM 470.80.01 CUDA Version：11.4
Pycharm	数据增广 U-Net (编码器：EfficientNet-b4)	GeForce GTX 1080 Ti CUDA Version 11.0

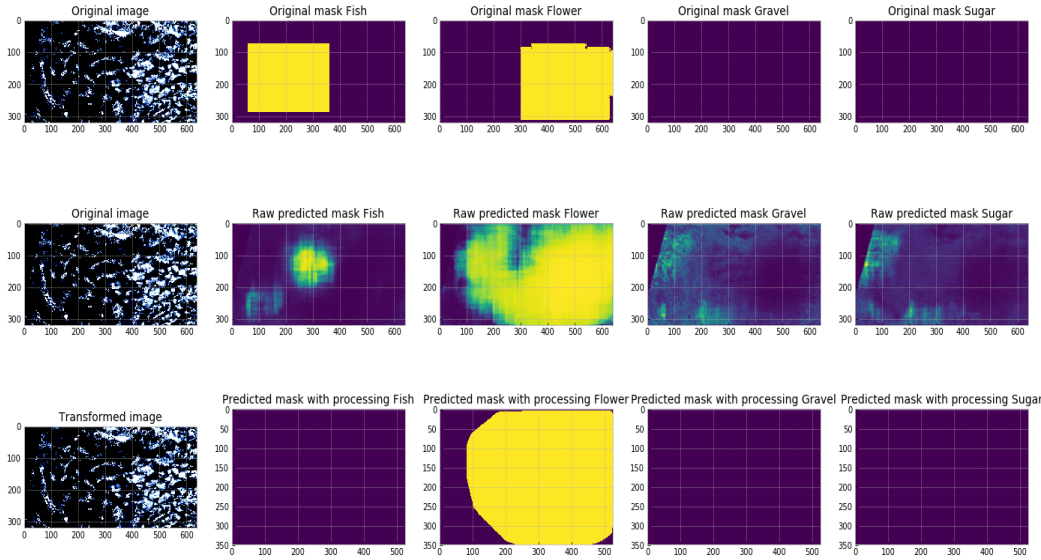
图表 12 **实验环境**

c) 实验结果可视化

i. 阶段一

使用卷积神经网络，Dice 系数仅 0.22，模型性能十分不理想，因此不展开详细赘述。

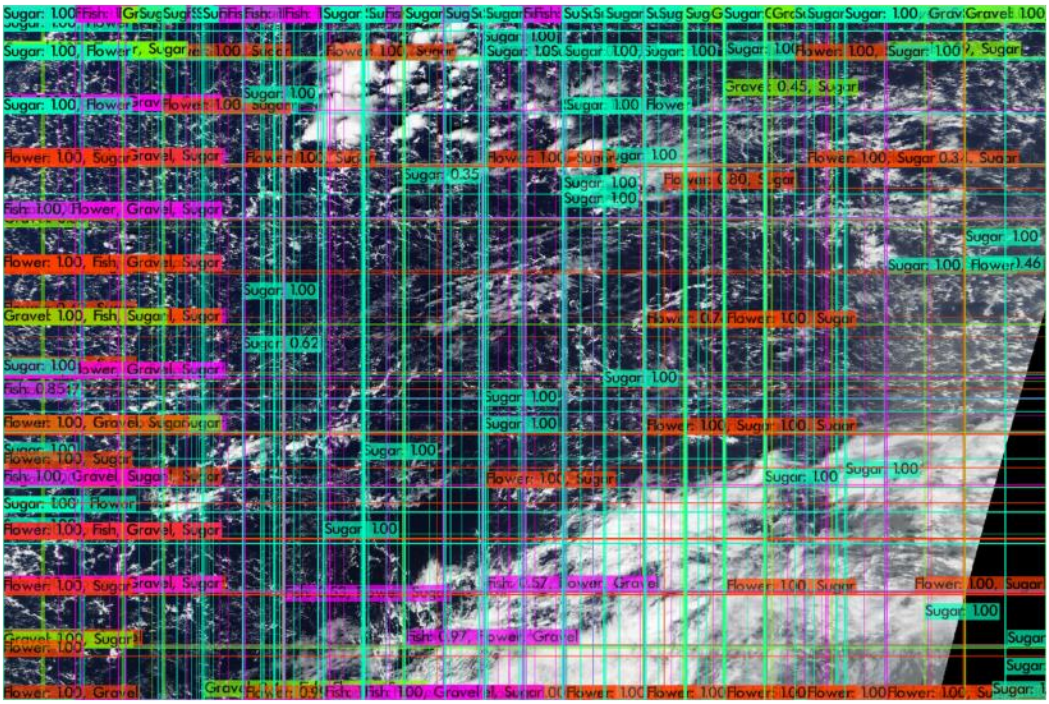
模型预测结果如下图所示，第三行经过图像分割掩蔽的图像与第一行图像为人工标注的原始数据的掩蔽差异甚大，足见模型效果不好。因此，本实验阶段一间接验证了，随着深度网络的提出，气象云图模式识别原始使用方法（包括 2010 年以前使用机器学习分类和聚类算法）逐渐被淘汰。



图表 13 卷积神经网络预测可视化

ii. 阶段二

尝试使用 YoLov4 目标检测算法，Dice 系数虽然得到提升，但是模型结果离谱，因为模型无法理解和区别四种类型云的区别，只能进行“盲猜”，因此本阶段的 Dice 系数不具有参考性。数据集描述部分提及四种类型界限并不明确，需要模型自己学习，因此，YoLov4 模型的尝试失败正是在于此。



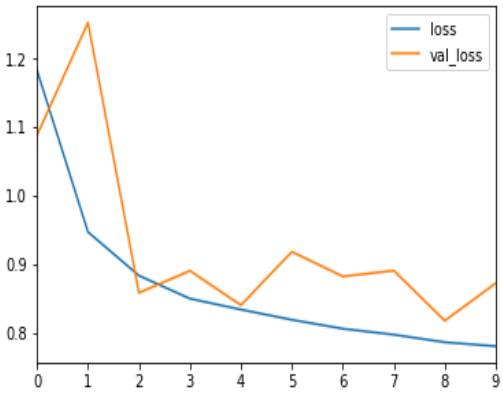
图表 14 YoLoV4 预测结果

iii. 阶段三

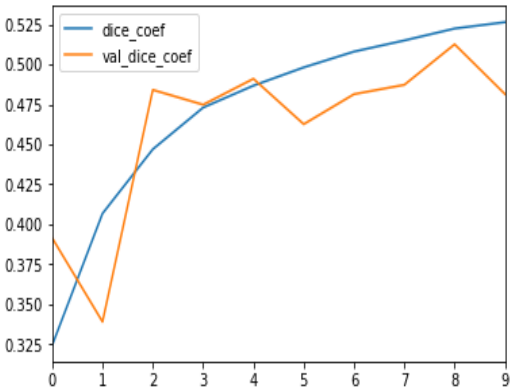
阶段三将数据直接输入使用 Efficient-Net b4 作为解码器的 U-Net 模型，最终 Dice 系数 0.44.在小规模训练（总 epochs 为 9）的可视化效果中可以看到，虽然训练集和验证集损失随着训练轮数下降，并且 Dice 系数随着训练轮数得到明显提升，但是从折线可以看出验证集的损失和 Dice 系数波动性较大，因此认为模型出现过拟合现象，终止本阶段实验。

考虑通过改进输入数据的方式改进实验。

改进一：通过数字图像变换进行数据增广，目的是为了减轻模型过拟合现象；改进二：通过使用数字图像处理 albumentations 包提高图像对比度突出图像信息，以将云层部分区别于卫星图像中的背景；改进三：通过最近邻插值方式改善图像分辨率、提高图片质量。



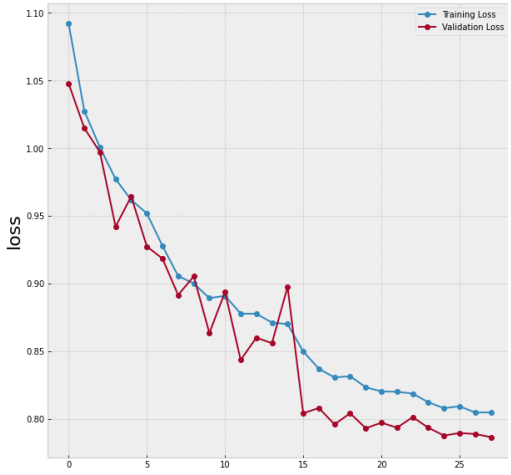
图表 15 U-Net (Encoder:EN-b4) 损失图



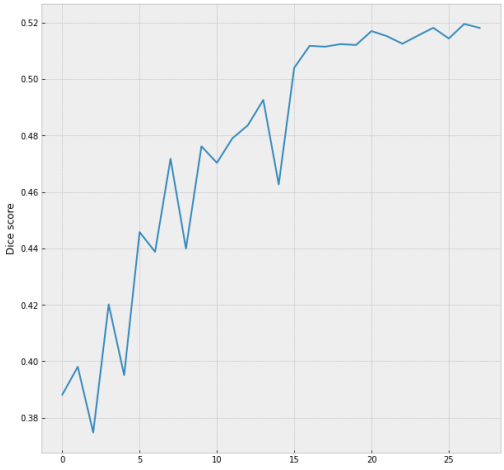
图表 16 U-Net (Encoder: EN-b4) Dice

iv. 阶段四

由于本阶段总轮数 29，开始随着训练轮数增加，训练集和验证集损失同比下降，并且验证集损失变化总体相较于阶段三平稳较多，从可视化结果可知，从第 15 轮开始，模型已开始收敛。同时，验证集 Dice 系数从第 15 轮开始变化平稳趋于稳定，最终结果 Dice 系数 0.52。

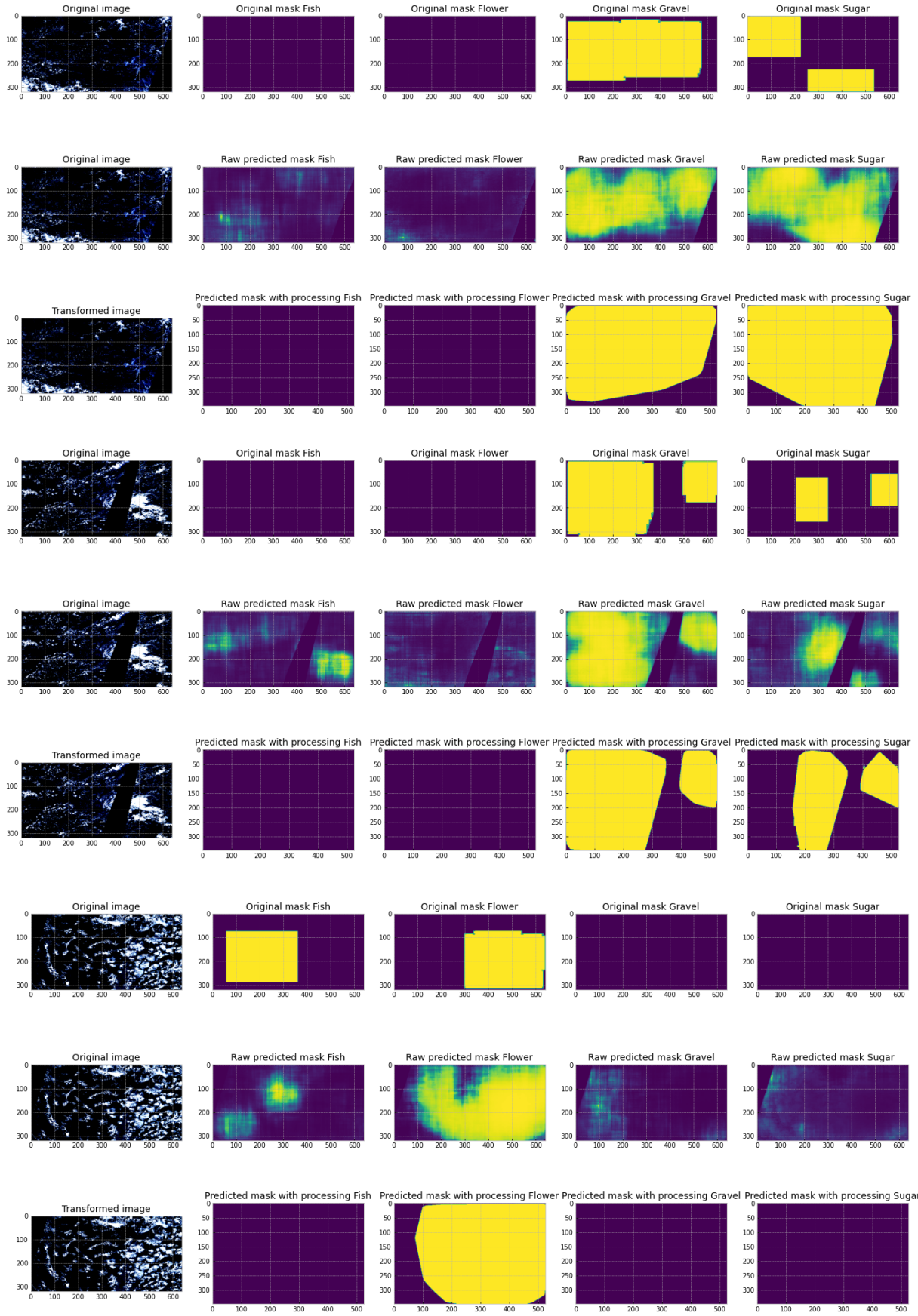


图表 17 U-Net 损失图



图表 18 U-Net 验证集 Dice

下图为模型预测可视化部分。可以看到模型预测大体上和人工标注的位置相近，模型性能确实得到了提升，认为 Dice 系数有效。



图表 19 阶段四预测可视化

六、 总结与展望

a) 实验总结

这次实验是计算机视觉领域的初次尝试，虽然实验过程比较难，但是总体收获很大。本实验总体分为四阶段，实验前两个阶段实验结果均失败，第四阶段基于第三阶段进行了实验改进，从最初 Dice 系数 0.22 提升到 0.52，总体提升 136%。姑且认为第四阶段实验合格。

本实验过程中学习到了数字图像处理技术，深刻体会到图像处理和图像变换能够减轻模型过拟合并提升模型性能。同时，了解了图像分割领域模型提出历史发展，尤其是 U-Net 和 Efficient-b4 模型。此外，学习了 Git 克隆仓库命令语句学习使用预训练模型进行迁移学习和参数微调。

b) 展望

模式识别与计算机视觉技术在工业界应用广泛，尤其是机器人和自动驾驶领域尤为重要。因此，未来需要继续深入学习计算机视觉领域的技术。本实验中使用图像分割技术应用于图像分类，未来有机会要学习图像分割技术中边缘检测和语义分割的相关应用场景的实验。

七、 附重要代码

```
def _get_pretrained_settings(encoder):
    pretrained_settings = {
        'imagenet': {
            'mean': [0.485, 0.456, 0.406],
            'std': [0.229, 0.224, 0.225],
            'url': url_map[encoder],
            'input_space': 'RGB',
            'input_range': [0, 1]
        }
    }
    return pretrained_settings
  'efficientnet-b4': {
      'encoder': EfficientNetEncoder,
      'out_shapes': (448, 160, 56, 32, 48),
      'pretrained_settings': _get_pretrained_settings('efficientnet-b4'),
      'params': {
          'skip_connections': [6, 10, 22],
          'model_name': 'efficientnet-b4'
      }
  },
```

图表 20 U-Net 预训练解码器

```python
class DecoderBlock(nn.Module):
    def __init__(self, in_channels, out_channels, up_sample=True, **_):
        super().__init__()
        self.up_sample = up_sample

        self.block = nn.Sequential(
                Conv3x3GNReLU(in_channels, out_channels),
                Conv3x3GNReLU(out_channels, out_channels),
        )

    def forward(self, x):
        if isinstance(x, list) or isinstance(x, tuple):
            x, skip = x
        else:
            skip = None

        if self.up_sample:
            x = F.interpolate(x, scale_factor=2, mode='nearest')

        if skip is not None:
            x = torch.cat([x, skip], dim=1)

        return self.block(x)
class Decoder(nn.Module):
    def __init__(self,
                 num_classes,
                 encoder_channels,
                 dropout=0.2,
                 out_channels=[256, 128, 64, 32, 16],
                 **_):
        super().__init__()
        in_channels = encoder_channels
        self.out_channels = out_channels
        self.in_channels = in_channels

        self.center = DecoderBlock(in_channels=in_channels[0], out_channels=in_channels[0],up_sample=False)
        self.layer1 = DecoderBlock(in_channels[0]+in_channels[1], out_channels[0])
        self.layer2 = DecoderBlock(in_channels[2]+out_channels[0], out_channels[1])
        self.layer3 = DecoderBlock(in_channels[3]+out_channels[1], out_channels[2])
        self.layer4 = DecoderBlock(in_channels[4]+out_channels[2], out_channels[3])
        self.layer5 = DecoderBlock(out_channels[3], out_channels[4])
        self.final = nn.Sequential(nn.Dropout(dropout),
                                   nn.Conv2d(out_channels[-1], num_classes, kernel_size=1))

    def forward(self, x, encodes):
        skips = encodes[1:]

        center = self.center(encodes[0])
        decodes = self.layer1([center, skips[0]])
        decodes = self.layer2([decodes, skips[1]])
        decodes = self.layer3([decodes, skips[2]])
        decodes = self.layer4([decodes, skips[3]])
        decodes = self.layer5([decodes, None])
        decodes4 = F.interpolate(decodes, x.size()[2:], mode='bilinear')
        outputs = self.final(decodes4)
        return outputs
class Conv3x3GNReLU(nn.Module):
    def __init__(self, in_channels, out_channels):
        super().__init__()
        num_groups = 32
        if out_channels % num_groups != 0 or out_channels == num_groups:
            num_groups = out_channels // 2

        self.blocks = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, (3, 3),stride=1, padding=1, bias=False),
            nn.GroupNorm(num_groups, out_channels),
            nn.ReLU(inplace=True),
        )
```

图表 21 U-Net Decoder

```python
#encoding:utf-8
import cv2
import numpy as np
import albumentations as albu
import matplotlib.pyplot as plt
plt.style.use('bmh')

# 获取已读取图像
image = io.imread(base_path_train+train_df['image'].iloc[0])
# img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

#处理

mytransform1 = albu.Compose([ albu.HorizontalFlip(p=0.5)])(image = image)['image']
mytransform2 = albu.Compose([ albu.HorizontalFlip(p=0.5), albu.ShiftScaleRotate(scale_limit=0.5, rotate_limit=0, shift_limit=0.1, p=0.5, border_mode=0),])(image = image)['image']
mytransform3 = albu.Compose([ albu.HorizontalFlip(p=0.5), albu.ShiftScaleRotate(scale_limit=0.5, rotate_limit=0, shift_limit=0.1, p=0.5, border_mode=0), albu.GridDistortion(p=0.7),])(image = image)['image']
mytransform4 = albu.Compose([ albu.HorizontalFlip(p=0.5), albu.ShiftScaleRotate(scale_limit=0.5, rotate_limit=0, shift_limit=0.1, p=0.5, border_mode=0), albu.GridDistortion(p=0.7), albu.Resize(320, 640),])(image = image)['image']
mytransform5 = albu.Compose([ albu.HorizontalFlip(p=0.5), albu.ShiftScaleRotate(scale_limit=0.5, rotate_limit=0, shift_limit=0.1, p=0.5, border_mode=0), albu.GridDistortion(p=0.7),
                             albu.Resize(320, 640), albu.Normalize(mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225)),])(image = image)['image']
mytransform6 = albu.Compose([ albu.HorizontalFlip(p=0.5), albu.ShiftScaleRotate(scale_limit=0.5, rotate_limit=0, shift_limit=0.1, p=0.5, border_mode=0), albu.GridDistortion(p=0.7),
                             albu.Resize(320, 640), albu.Normalize(mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225)), albu.RandomBrightnessContrast(brightness_limit = 0.0,contrast_limit = 0.2,p = 0.05)])(image = image)['image']
#显示结果
LinearInterplotation = cv2.resize(mytransform4, (640, 320),interpolation=cv2.INTER_LINEAR) # 双线性插值法
NearestInterplotation = cv2.resize(mytransform4, (640, 320),interpolation=cv2.INTER_NEAREST) # 最近邻插值
# AreaInterplotation = cv2.resize(mytransform4,(640, 320), interpolation=cv2.INTER_AREA)  # 区域插值法
# CUbicInterplotation = cv2.resize(mytransform4, (640, 320),interpolation=cv2.INTER_CUBIC) # 双三次插值

titles = ['Original', 'mytransform1','mytransform2','mytransform3',
          'mytransform4',
          # 'mytransform5',
          # 'mytransform6',
          'LinearInterplotation',
          'NearestInterplotation',
          # 'AreaInterplotation',
          # 'CUbicInterplotation'
          ]
images = [image, mytransform1, mytransform2, mytransform3, mytransform4,
          # mytransform5,
          # mytransform6,
          LinearInterplotation,
          NearestInterplotation,
          AreaInterplotation,
          CUbicInterplotation]
plt.figure(figsize=(70,5))
for i in range(7):
   # plt.figure(figsize=(5,5))
   plt.subplot(1,7,i+1),plt.imshow(images[i])
   plt.title(titles[i])
   plt.xticks([]),plt.yticks([])
plt.show()
```

图表 22 图像变换

八、 主要参考文献

[1] 寿亦萱. 模式识别技术及其在气象研究中的应用[D].南京气象学院,2004.

[2] 丁叶文. 基于深度学习的卫星云图分割和识别研究[D].南京信息工程大学,2020.DOI:10.27248/d.cnki.gnjqc.2020.000332.

[3] 黄磊. 基于图像匹配和模式识别技术的卫星资料应用[D].北京大学,2007.

[4] 维基百科贡献者.U-Net[D].维基百科,2015.https://en.wikipedia.org/w/index.php?title=U-Net&oldid=1073187826.

[5] Ronneberger, O., Fischer, P., & Brox, T.U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241)[J].Springer, Cham,2015.

[6] Bochkovskiy A, Wang C Y, Liao H Y M. Yolov4: Optimal speed and accuracy of object detection[J]. arXiv preprint arXiv:2004.10934, 2020.

[7] Tan M, Le Q. Efficientnet: Rethinking model scaling for convolutional neural networks[C]//International conference on machine learning. PMLR, 2019: 6105-6114.