成都信息工程大學

Chengdu University of Information Technology

多尺度遥感影像上建筑物的智能提取研究

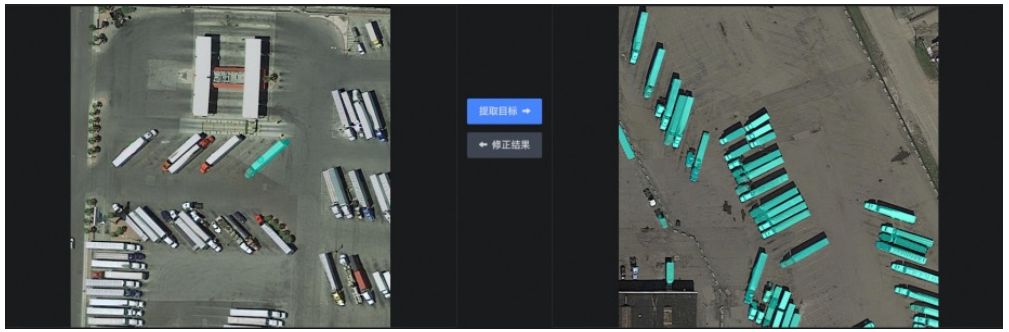
小组成员：李林峰、邓敏、朱浩玮

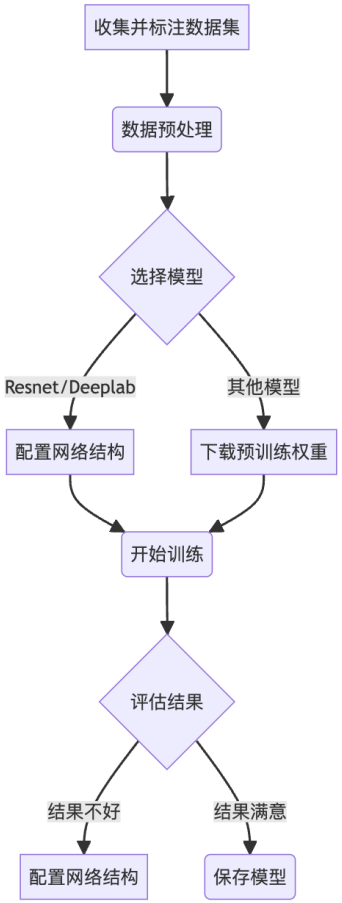
- 从不同尺度的遥感影像中提取出建筑物对城市建设和发展进行分析和规划具有重点意义。如可以获得城市建筑物的分布、密度、类型等信息，进而对城市规划和土地利用进行科学决策

- 通用人工智能AIG的快速发展，AI+遥感
  - Meta AI ——Segment Anything
  - 阿里达摩院——AI Earth

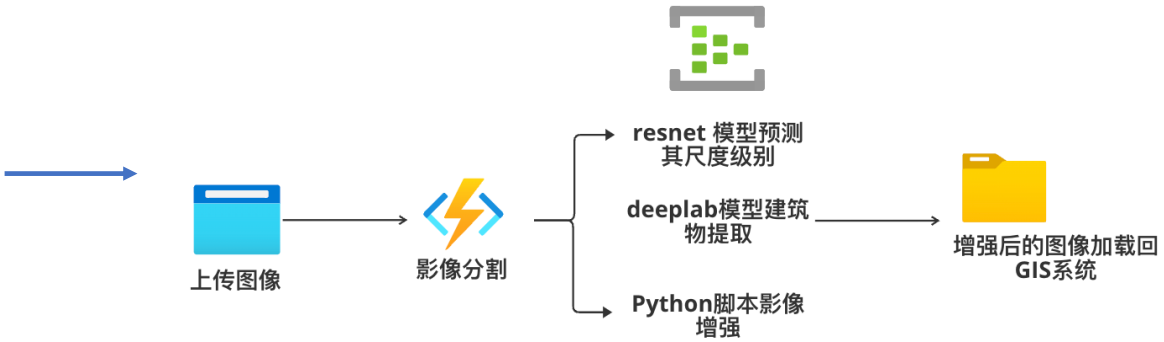
- AI+传统办公软件
  - Microsoft Office Copilot

- 本次实践也是一次对AI+传统GIS平台的一次尝试





Deeplab 和 Resnet 模型训练流程图

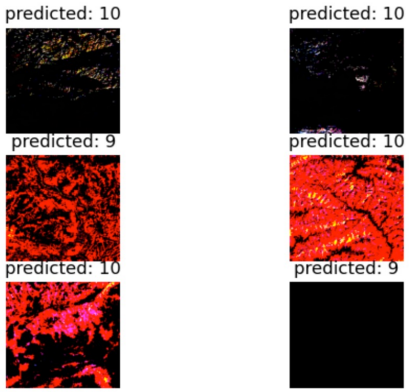


C#+ArcGIS Engine组件式GIS流程图

- **工程实践1**得到的遥感影像分级模型grad.net（能对2-18级的影像做出70%正确率的分级）

- **工程实践2**手工采集的16、17、18三级各200张遥感影像及其对应的建筑标记xml文件

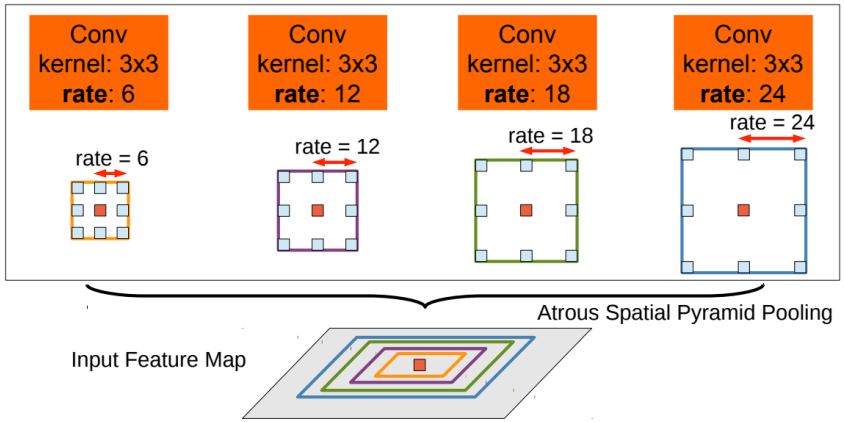
- **工程实践3**得到的XML转图片后深度学习模型可用的数据集，图片是二值的，值为0为背景，值为1为建筑物



一、模型的训练与部署

1. 训练平台：Google Colab提供的免费算力资源

2. DeepLab 是一个基于深度学习的语义分割模型，其目标是对图片进行像素级别的分类和分割。这个模型使用了各种卷积神经网络技术（如空洞卷积、金字塔池化等）以及多尺度图像处理方法来提高准确性。



3. 数据集的结构（以17级为例）

-17
——train
————images (180 张 512x512 遥感影像)
————labels (180 张 512x512 遥感影像)
——test
————images (20 张 512x512 遥感影像)
————labels (20 张 512x512 遥感影像)

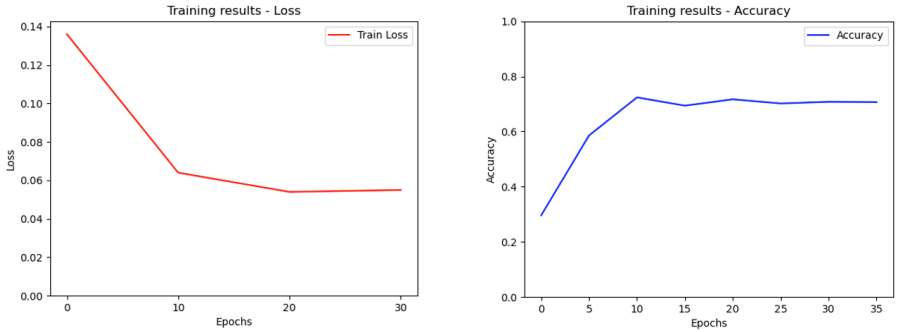
一、模型的训练与部署-核心代码

- 通过deeplab语义分割模型对不同级别的遥感影像进行训练，得到最适合各自尺度下的建筑物提取模型
- 像素点分为两类
  - Backgroud：背景
  - Built-up：建筑
- Batch Size设置为2
- Epochs设置为40轮时准确度已经能收敛至较佳的水平

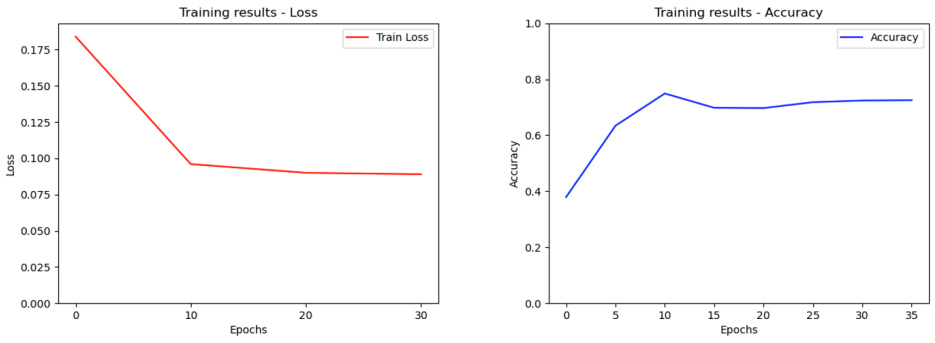
```
def train_model(model, device,dataloaders,criterion, optimizer, scheduler, num_epochs=25):
    since = time.time()
    class_names = ['background', 'built-up']
    t_zero = torch.tensor(0).to(device)
    for epoch in range(num_epochs):
        print('Epoch {}/{}'.format(epoch, num_epochs - 1))
        print('-' * 10)
        for phase in ['train', 'val']:
            if phase == 'train':
                model.train()  # Set model to training mode
            else:
                model.eval()  # Set model to evaluate mode
            running_loss= 0.0
            avg_accu=0
            cur_accu=0
            total_accu=0
            i=0
            for inputs, labels in dataloaders[phase]:
                inputs = inputs.to(device)
                target = labels.to(device)
                optimizer.zero_grad()
                output = model(inputs)
                _, preds = torch.max(output, 1)
                target = target[..., 0]  # 只保留第一个通道
                total = torch.sum(target != t_zero)
                prid = torch.sum((preds == target) * (target != t_zero))
                total=total.float().item()
```

模型训练的关键代码

一、模型的训练与部署-训练过程



17级训练损失值Loss以及精度Accuracy变化图



18级训练损失值Loss以及精度Accuracy变化图

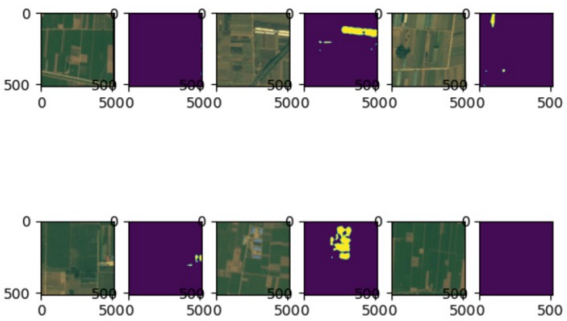
以17级和18级训练过程的损失值Loss和精度Accuracy的变化图为例。

可以观察出两个级别下的训练的损失值**Loss随着训练的轮数增加而减小**，最后趋于一个稳定值；而**精度Accuracy随着训练轮数的增加而增加**，最后趋于一个稳定值。

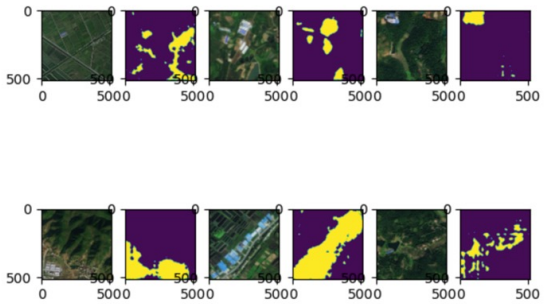
由此可分析得，随着训练的次数不断增加，建筑物识别得精度不断提高最后趋于稳定。

一、模型的训练与部署-训练结果

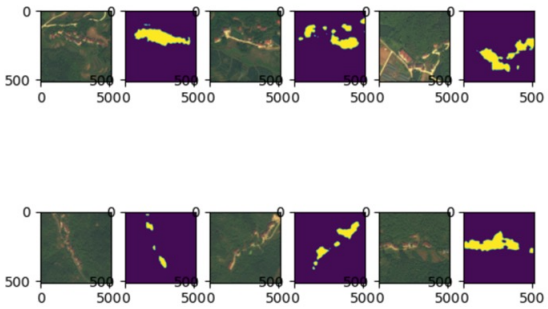
分别从不同尺度下的测试集中选择6张图片作为测试样本，测试结果如下：



16 级建筑物提取结果



17 级建筑物提取结果



18 级建筑物提取结果

由上图3个级别下的测试结果可以看出：通过目视解译所观察到的建筑物大体都能够被模型提取出来，虽有些许误差，但在目前有限的条件下已达到了一个较符合要求的标准，故而各级提取模型基本满足本项目对建筑物提取的业务需求，可以进行后续识别使用。

二、AI+组件式GIS系统的搭建-窗体框架

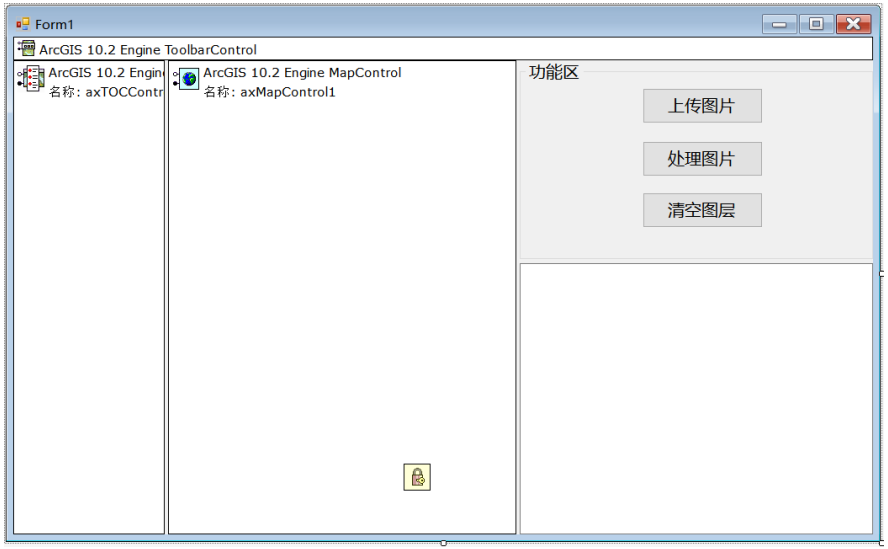
（1）界面上方利用ToolbarControl控件构成工具栏，其中包含着地图操作的基本功能，如：放大、缩小、平移、全图显示、刷新地图、返回上一视图等。

（2）左边区域利用axTOCControl控件显示地图的图层。

（3）中间利用axMapControl控件来显示地图。

（4）右上方由groupBox控件构成的功能区，主要功能为：上传所要识别的遥感图像、识别并提取遥感影像中的建筑物以及清空图层。

（5）右下方由textBox控件构成的处理进程显示的功能区。



二、AI+组件式GIS系统的搭建-上传图片与裁剪图片

1. 对象将图片作为**栅格数据**加载到地图控件中。
   IWorkspcaeFactory ->IRasterWorkspace ->IRasterDataset ->IRasterLayer

2. 裁剪图片：
   1. 输入模型的图片要求为512x512尺寸
   2. 计算需要将图片分割成几块标准尺寸
   3. 为每个图创建一个新的512x512尺寸的新位图bitmap并将原始图像上对应位置的像素值复制到新位图上。

```
// 定义每个裁剪区域的矩形范围
Rectangle cropRect =
    new Rectangle(x * blockSize, y * blockSize, blockSize, blockSize);

// 创建新位图对象，并从原始位图中复制指定区域到新位图中。
Bitmap croppedBitmap =
    new Bitmap(cropRect.Width, cropRect.Height);

using (Graphics g = Graphics.FromImage(croppedBitmap))
{
  g.DrawImage(originalImage,
              new Rectangle(0, 0, croppedBitmap.Width,croppedBitmap.Height),
              cropRect,
              GraphicsUnit.Pixel);
 }
```

二、AI+组件式GIS系统的搭建-调用模型

- C# Process类
  类似于C的fork，Javascript的child_process.spawn

- ProcessStartInfo
  - FileName：运行程序-python.exe（必选）
  - Arguments：参数-执行的python文件与Args（必选）
  - WindowStyle/CreateNoWindows：隐藏命令窗口?
  - RedirectStandardErr/RedirectStandardOutput：重定向输出？

- Process.Start(ProcessStartInfo)运行

- 以调用resnet模型为例，deeplab形式一样

- resnet返回当前图片的预测级别作为参数传递给deeplab

```
private string GetResnetOutput(string pngPath)
{
    textBox1.AppendText("准备调用resnet模型对图像进行级别鉴定\r\n");
    // 设置 Python 解释器路径和脚本文件路径
    string python = @"C:\python programdata\Anaconda3\python.exe";
    string resnetpy = @"C:\Users\41181\Desktop\ai\resnet\predict.py";
    string script = resnetpy + " " + pngPath;

    // 创建进程对象并设置参数
    ProcessStartInfo startInfo = new ProcessStartInfo();
    startInfo.FileName = python;
    startInfo.Arguments = script;

    // 配置进程选项以隐藏命令窗口、重定向输出等（可选）
    startInfo.WindowStyle = ProcessWindowStyle.Hidden;
    startInfo.CreateNoWindow = true;
    startInfo.RedirectStandardError = true;
    startInfo.RedirectStandardOutput = true;
    startInfo.UseShellExecute = false;

    // 运行新进程并等待其完成执行
    using (Process process = Process.Start(startInfo))
    {
        using (StreamReader reader = process.StandardOutput)
        {
            string stderr = process.StandardError.ReadToEnd();
            string result = reader.ReadToEnd().TrimEnd('\r', '\n');
            textBox1.AppendText("resnet模型运行成功! \r\n");
            return result;
        }
    }
}
```

二、AI+组件式GIS系统的搭建-图像增强

- 图像增强的原因？

因为提取后建筑二值图像中，非建筑物像素值为0，建筑物像素值为1，两者区分度太小，在许多图片浏览软件中不能很好的区分。

- 图像增强的原理？

故将images/predict路径下图像中，所有像素值为1的像素点修改为255，以达到增强图像显示效果。

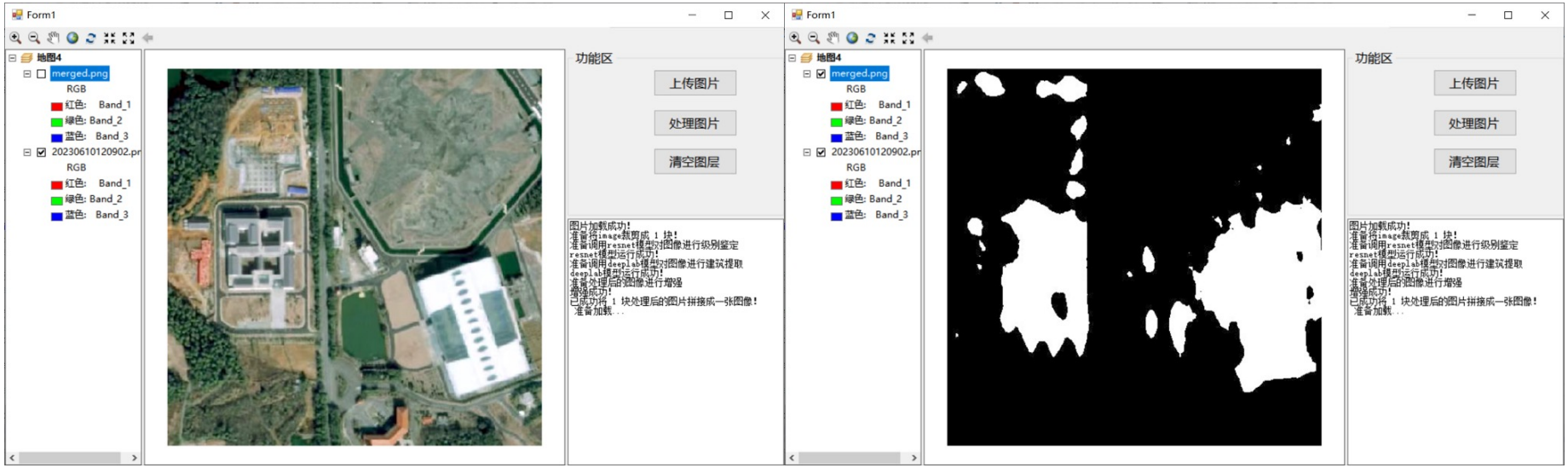
```
image_path = ""
if len(sys.argv) > 1:
image_path = sys.argv[1]
else:
exit(1)
image = cv2.imread(image_path)
image[np.where(image != 0)] = 255

filename = os.path.basename(image_path)
cv2.imwrite(DirectionName+filename,
image)
```

二、AI+组件式GIS系统的搭建-图片拼接

- 图像拼接
  - 创建一个与**原始输入图像**尺寸一样的新bitmap，将增强后的图像按照对应的位置把自身像素点的值赋值到新位图上
  - 假设原始图像被分为2行3列，2号分片的位置计算如下：
    - ✓ 行：2 / 3 +1 = 1，第1行
    - ✓ 列：2 % 3 = 2，第2列
  - 对于边缘部分进行忽略，统一赋值为0

```
// 计算总共需要多少列和行
int totalColumns = (int)Math.Ceiling(Math.Sqrt(imagePaths.Count));
int totalRows = (int)Math.Ceiling((double)imagePaths.Count / totalColumns);

// 创建新位图对象并初始化为白色背景
Bitmap resultBitmap =
    new Bitmap(imageWidth,
               imageHeight);

using (Graphics g = Graphics.FromImage(resultBitmap))
{
    g.Clear(Color.Black);

    for (int i = 0; i < imagePaths.Count; i++)
    {
        string cropPath = imagePaths[i];

        // 计算当前块应该被绘制到哪个位置上
        int columnIdx = i % totalColumns;
        int rowIdx = i / totalColumns;

        Image currentImage = Image.FromFile(cropPath);

        // 将当前块绘制到目标位图中指定的位置上。
        g.DrawImage(currentImage,
                    new Rectangle(columnIdx * 512,
                                  rowIdx * 512,
                                  currentImage.Width, currentImage.Height));

         currentImage.Dispose();
     }
 }
```

在最后搭建的组件式GIS系统中选择含有建筑物的遥感影像加载到程序中，再点击处理图片，即可得到经过调用各自尺度下对应的建筑物提取模型识别再经增强处理后的二值图（结果见下图）。对比观察可知，本系统能够大体地将建筑物提取出来，但与目视仍有一定出入，精度需后续工作持续提升。



本阶段虽然完成了基本目标的实现，但仍有很多不足之处需要完善。

- 首先，通过系统最后提取的结果可见，遥像影像中的建筑物提取的效果仍有待提高，模型目前只能提取出一个大体的形态，尤其是建筑物的边界处的识别较差。
- 另外，由于时间原因本阶段只实现了16级到18级的样本级的采集以及对应分级的提取模型，识别的尺度达仍需扩充。
- 最后，系统的功能目前是属于一些基本功能的实现，后续仍需继续完善功能，使之达到业务需求。
- 后续我们将完善数据集，增加其广度与深度，使得系统能够高精度地识别更多尺度下的遥感影像中的建筑物。此外我们也将继续完善建筑物提取系统，使其能有较为完善和复杂的功能，满足业务逻辑功能需求。

THANKS