CVDL大作业
期末报告

葛博文 1500019707
周清逸 1500012930

2018年6月13日

1 任务动机

在一次寒假调研时，我们意外地发现，尽管地方政府对农业用地早已有了诸多保护政策，农业用地违法挪用建造商品房以谋取私利的情况依然广大范围内存在。这种行为对我国的农业生产工作产生了极大的危害，必须严加监管。而这现象屡禁不止的原因之一，是高层政府对于农业用地的实际监管困难。因此我们希望做一个程序来试图识别农业用地的实际使用情况。

2 任务分析

如何从遥感卫星图片判断一块土地是否为农业用地是一个比较泛泛的概念。考虑到很多土地挪用是为了商品房建造，同时房屋的数据集也5比较容易得到，因此我们可以将这个土地使用方式识别的任务转化为在一片地图上寻找房屋的任务。我们从“一个网站”上得到了有关房屋的数据集 [**?**]。初步分析数据集我们可以猜想这个任务的主要难点在于:

- 房子数量多而分布密集，单个房子却很小，会对网络产生压力。
- 由于光照问题，房屋和树木遮挡，会相应地产生阴影投射到房子上，对于网络的识别增加了困难。
- 不同数据集之间差别很大，而且不均匀。例如有些城市沿海，就会有少量的水面和船等景象，然而这些数量很小，不能在网络中充分训练。
- 房屋和船、大卡车等物品天生很像。
- 在这些数据集上有关绿地的部分主要为树木和草地，而作为我们最终目标的中国地区作为测试的图片中，却主要为农田，这样的数据集变化可能会导致很糟糕的效果。

3 图片预处理

我们使用的数据集的特点如下：

- 包含覆盖了$810km^2$（其中$405km^2$用于训练集，$405km^2$用作测试集）。
- 是空间分辨率为0.3m的正交矫正彩色图像。
- 标签是二分类分割：是房子和不是房子。
- 这些图像涵盖了不同的城市住区，从人口稠密地区(例如旧金山金融街)到高山城镇(例如奥地利蒂罗尔的利恩茨)。

我们先将原先大小为5000×5000的图片补成5120×5120（$padding = 60$）的大小。然后将按照步长160（$stride = 160$）其顺序切成961张320×320的小图。
我们随后对图片做的数据增强如下：

- 将图片旋转一个随机的角度
- 将图片随机水平和数值翻转
- 将图片加入高斯噪声
- 随机调整图片的亮度、对比度、饱和度和色调

同时也对标签做相同的旋转和翻转的变换。
我们这样做的原因是主要考虑到对于美国城市而言，房子的方向大多是极其统一的，而卫星的拍照方向固定，将造成训练集中大量的房屋方向相同，这样将使得网络的训练数据过于局限。在此考虑下，我们决定引入图片的旋转和翻转，这样可以引入训练集的随机性，用于增强网络的范性。同时，由于不同城市不同国家的房屋整体色调也有所不同，于是我们希望淡化网络对特定颜色的认知，而主要提取形状特征，这样调整图片的色调就是必要的。同时为了增强网络的鲁棒性，我们也对输入训练数据加入了高斯噪声。

4 网络结构

任务本身属于语义分割。用深度学习进行语义分割的方法多种多样，在本次任务中我们决定采用U-Net，如图所示。 采用U-Net的原因主要是由于结构清晰：属于典型的Encoder-Decoder架构，专门为分割任务设计，考虑到单纯的Encoder-Decoder型网络会丢失图片细节，加入了short-cut，通过将不同尺度的信息进行拼接的方式，缓解了细节的丢失。

最初使用一个U-Net进行测试的效果并不是很理想，于是我们尝试将两个U-Net拼接在一起，希望通过增加模型的深度，取得更好的效果。然而在训练时，拼接的模型很快就出现了过拟合现象，在验证集上的loss无法很好地收敛，泛化性能很差，因此最终我们没有采用这一模型。

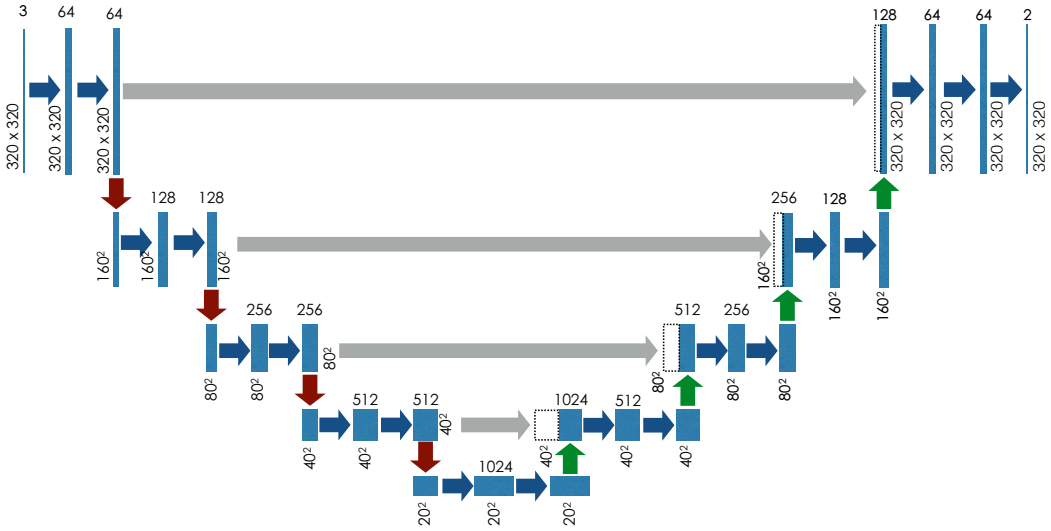

Figure 1: A schematic illustration of U-Net's structure.

4.1 Loss函数

如果简单地把分割任务看作一个二分类，在网络的最后一层使用Softmax激励函数，得到每个像素分属于两类（建筑/非建筑）的概率，那么可以直接用交叉熵作为Loss函数。从代码train.py中可以看到，pyTorch中对于分割问题的Loss，使用的是Nllloss()函数，此时网络最后一层是logsoftmax，但本质是相同的。

对于二分类问题，交叉熵的表达式可以写作：

$$L = -\frac{1}{N}\sum[y_i \ln(p_i) + (1-y_i)\ln(1-p_i)]. \tag{1}$$

在模型V1.0中，我们直接使用交叉熵作为Loss函数，效果比较差，存在很多大量False negative，很多建筑物无法判断出来。False negative较多的主要原因是正负样本不均衡，因此在此后的模型中，我们决定调整交叉熵表达式中两项的权重。根据参考文献 []，最直接的解决方法是计算正负两类所占的比例α_0和α_1（自然满足$\alpha_0+\alpha_1=1$），然后将两项分别乘上对应的系数：

$$L = -\frac{1}{N}\sum[\alpha_0 y_i \ln(p_i) + \alpha_1(1-y_i)\ln(1-p_i)]. \tag{2}$$

经过统计，在数据集中，建筑物对应的像素所占比例约为$15\% - 20\%$，因此在模型V1.2中，采用比例$\alpha_1 = 0.2$。这样，如果对于某个像素满足$y_i = 1$，但预测p_i明显小于1，乘以α_0相当于对此类情形施加惩罚，从而一定程度上解决正负样本不均的问题。

一般而言，针对语义分割任务，一个重要的评价标准是IoU。实际上IoU比单纯计算准确率更加合理（对于建筑物只占15%的情形，预测全0便可以达到85%的准确率）。

对于一个类别而言，记预测的mask为X，实际的mask为Y，IoU的定义为：

$$\mathrm{IoU} = \frac{|\mathrm{X} \bigcap \mathrm{Y}|}{|\mathrm{X} \bigcup \mathrm{Y}|}. \tag{3}$$

经过查阅资料 []，针对IoU可以设计更为合理的Loss。根据Y和预测的X定义Dice系数：

$$\mathrm{DSC} = \frac{2\,|\mathrm{X} \bigcap \mathrm{Y}|}{|\mathrm{X}| + |\mathrm{Y}|}. \tag{4}$$

直观来说，Dice系数形式上比较接近IoU。为了达到最大化IoU的目的，可以尝试将Dice系数加入Loss函数。包含X的Dice系数是不可导的，因此记P为预测的概率（不转化为0/1标签），定义非负的$\mathrm{DSC}_{\mathrm{diff}}$为：

$$\mathrm{DSC}_{\mathrm{diff}} = \frac{2\sum \mathrm{y_i p_i}}{\sum \mathrm{p_i} + \sum \mathrm{y_i}}. \tag{5}$$

当完美预测时（所有标签为1的像素，预测为建筑的概率均为1），$\mathrm{DSC}_{\mathrm{diff}}$达到1。在Kaggle比赛中，有人采用$1 - \mathrm{DSC}_{\mathrm{diff}}$作为Loss函数中的一项。在本次实验中，为了对IoU较小的预测施加更大的惩罚力度，我们将$-\ln(\mathrm{DSC}_{\mathrm{diff}})$加入Loss函数中（在$\mathrm{DSC}_{\mathrm{diff}}$接近1的时候二者相近）。对于模型V1.4，我们采用如下形式的Loss函数进行训练：

$$L = -\frac{1}{N}\sum[(1-\alpha_1)y_i \ln(p_i) + \alpha_1(1-y_i)\ln(1-p_i)] - \beta \ln(\mathrm{DSC}_{\mathrm{diff}}). \tag{6}$$

我们在报告后面会对各个不同模型进行测试，测试条件和结果汇总在表格当中。

5 条件随机场（CRF）

条件随机场（Conditional Random Field）是一种判别式概率模型，具有无向的图模型 []。实际上，对于图像分割的任务而言，CRF属于传统算法。在谷歌的DeepLab中[]，CRF被用于与深度神经网络结合，作为后处理的手段。迄今为止，DeepLab在诸多图像分割数据集（VOC, Cityscapes）上都达到了相当不错的效果，展现了深度学习方法与CRF结合的潜力。

在DeepLab的论文中，神经网络的输出被作为先验概率，如果直接使用这一概率进行分割，结果是很粗糙的（特别是在两类的边缘附近），即便加入short-cut也无法完全解决细节丢失的问题。CRF则根据神经网络的输出，借助像素的位置和颜色信息，计算后验概率。在区域中间部分，先验概率远离阈值，基本不会受到CRF的影响；在两区域边缘处，CRF进行后处理的主要依据则是像素颜色，处理之后可以达到比较精细的分割结果。

参考论文 []，采用全连接CRF进行图像分割，吉布斯自由能为：

$$E(\mathbf{x}) = \sum_i \phi_u(x_i) + \sum_{i,j} \phi_p(x_i, x_j). \tag{7}$$

第二项为成对的像素对应的能量，表达式为：

$$\phi_p(x_i, x_j) = \mu(x_i, x_j) k(\mathbf{f}_i, \mathbf{f}_j), \tag{8}$$

其中，核$k(\mathbf{f}_i, \mathbf{f}_j)$的表达式采用如下形式：

$$k(\mathbf{f}_i, \mathbf{f}_j) = w^{(1)} \exp[-\frac{\|p_i - p_j\|^2}{2\theta_\alpha^2} - \frac{\|I_i - I_j\|^2}{2\theta_\beta^2}] + w^{(2)} \exp[-\frac{\|p_i - p_j\|^2}{2\theta_\gamma^2}]. \tag{9}$$

在CRF算法中，涉及到诸多参数。在本次实验当中，用CRF进行后处理并不是任务最主要的部分，只是作为一种尝试。因此我们采用了一个轻量级的名为pydensecrf的库（github网址），然后人为设定了相应的参数，并根据部分图片的处理情况进行了一定的调整。

经过尝试，我们发现，尽管人为调整CRF的参数θ_α、θ_β、θ_γ可以在一个小区域内达到比较好的结果，但是这一组参数用于其他区域的效果可能不尽人意（泛化性能较差）。这一现象并不难理解：首先，不同城市的建筑/非建筑色调差异很大；其次，即便是同一城市，若照片的拍摄时间、地点不同，也会存在较明显的光照差异，影响到像素点的颜色。本次实验采取的CRF算法比较简单，只考虑到一对像素点之间的能量，特征选取也很简单，仅包括位置和颜色，本身鲁棒性不够强。

对于我们的模型而言，CRF比较重大的意义在于，通过后处理，可以将小块的false positive去掉，改善分割结果。然而，如果参数设置不合理，可能会出现将true positive的一部分去除的情况（例如，由于屋顶有角度，向光与背光两面的颜色差异比较大，CRF后处理之后只保留了向光的部分）。因此，在使用条件随机场进行后处理时必须谨慎。

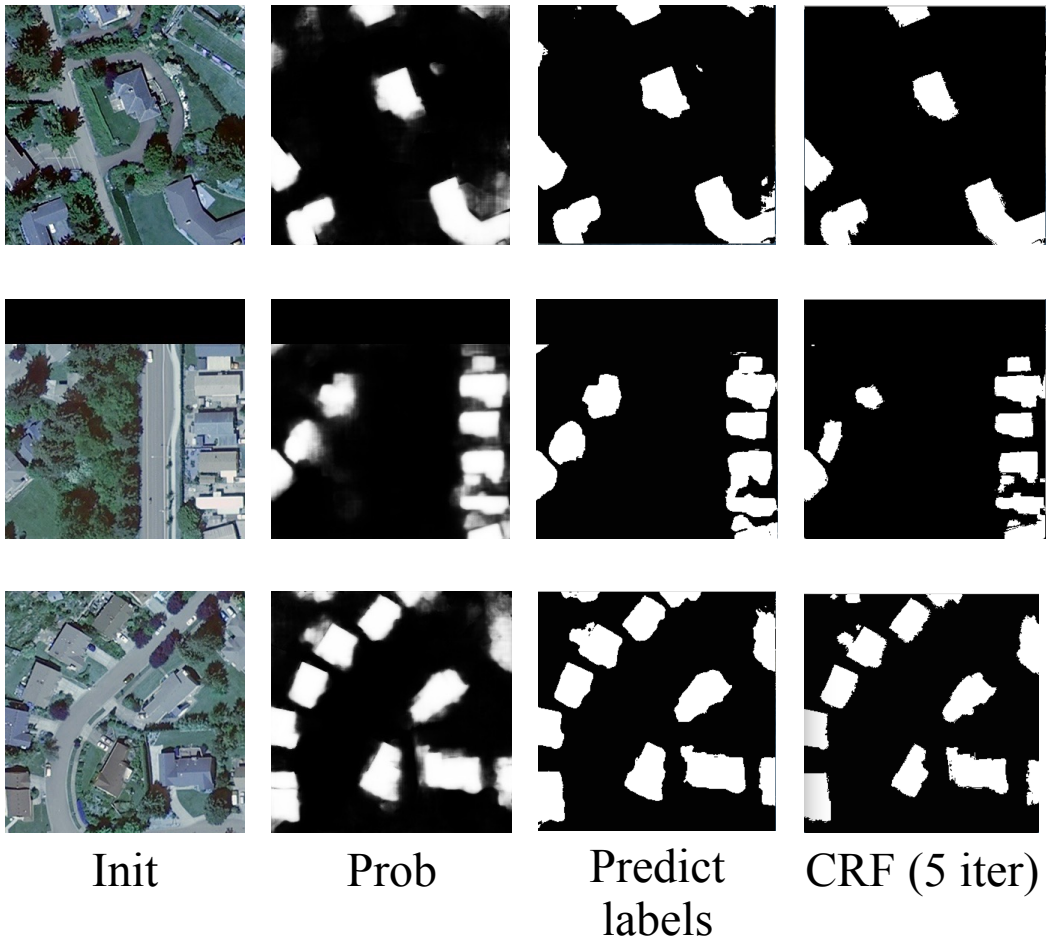

Figure 2: 借助CRF进行后处理的一些例子。不难看到，几次迭代之后，预测mask的边缘变得比较锐利，小块的false positive被消除，IoU得到了改善。

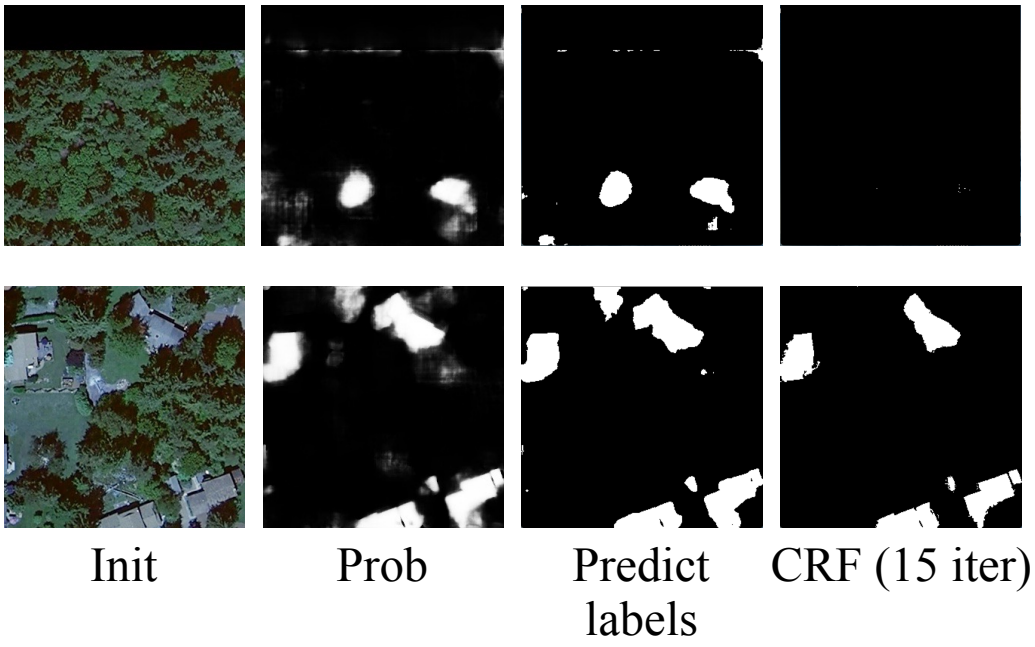

Figure 3: CRF可以去除掉false positive，改善分割结果。

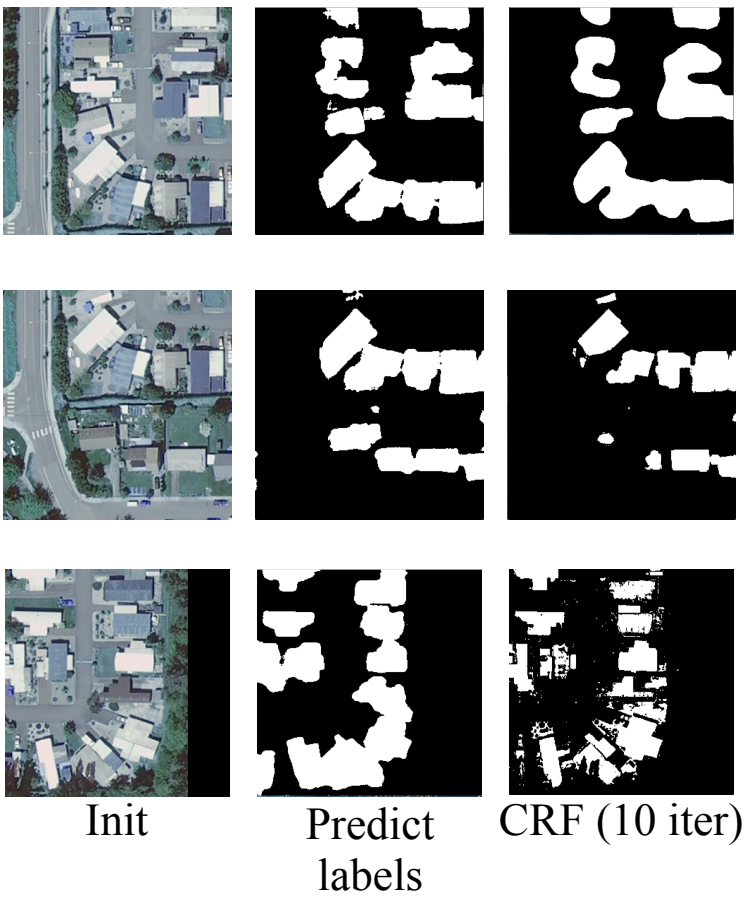

Figure 4: 如果参数设置不合理，可能导致后处理之后效果反而变差，几种典型的情况如图。