[WCFun 遥感解译平台]

详细设计文档

[V1.0(版本号)]

拟 制 人__汪成飞 海江涵 李满园 郑沧平__

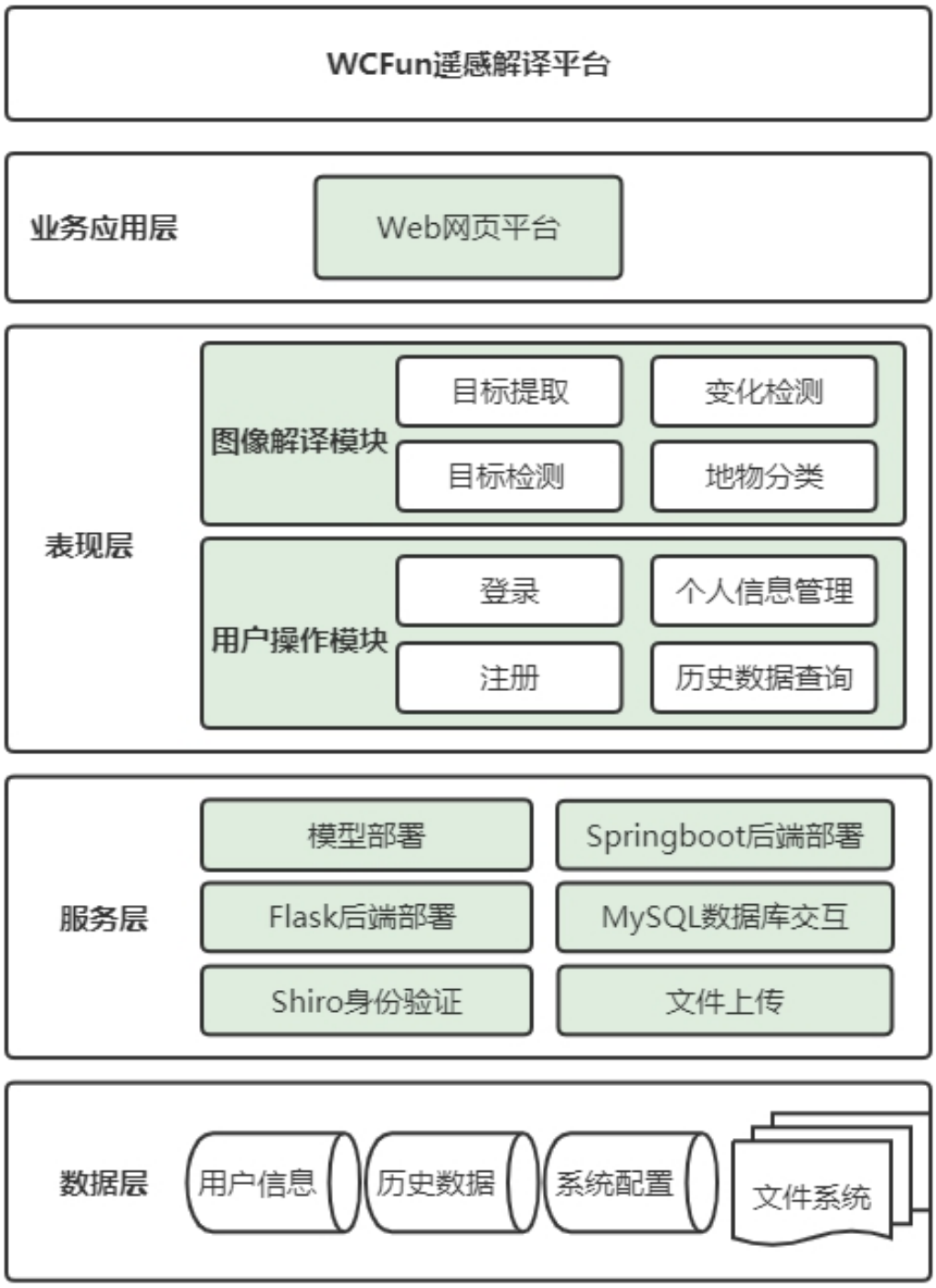
[二零二二年八月十二日]

详细设计文档

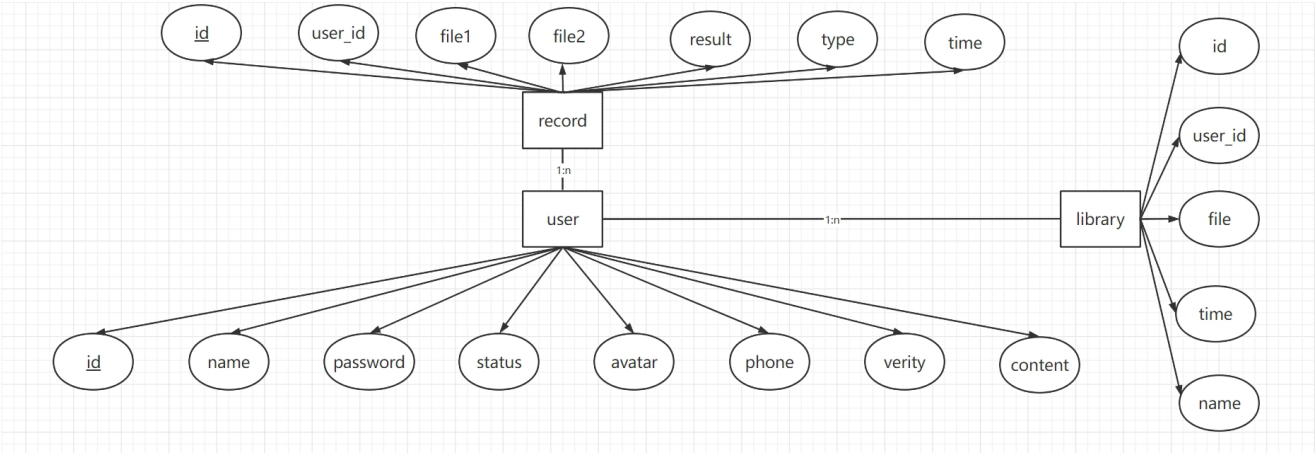
目录

一、系统设计

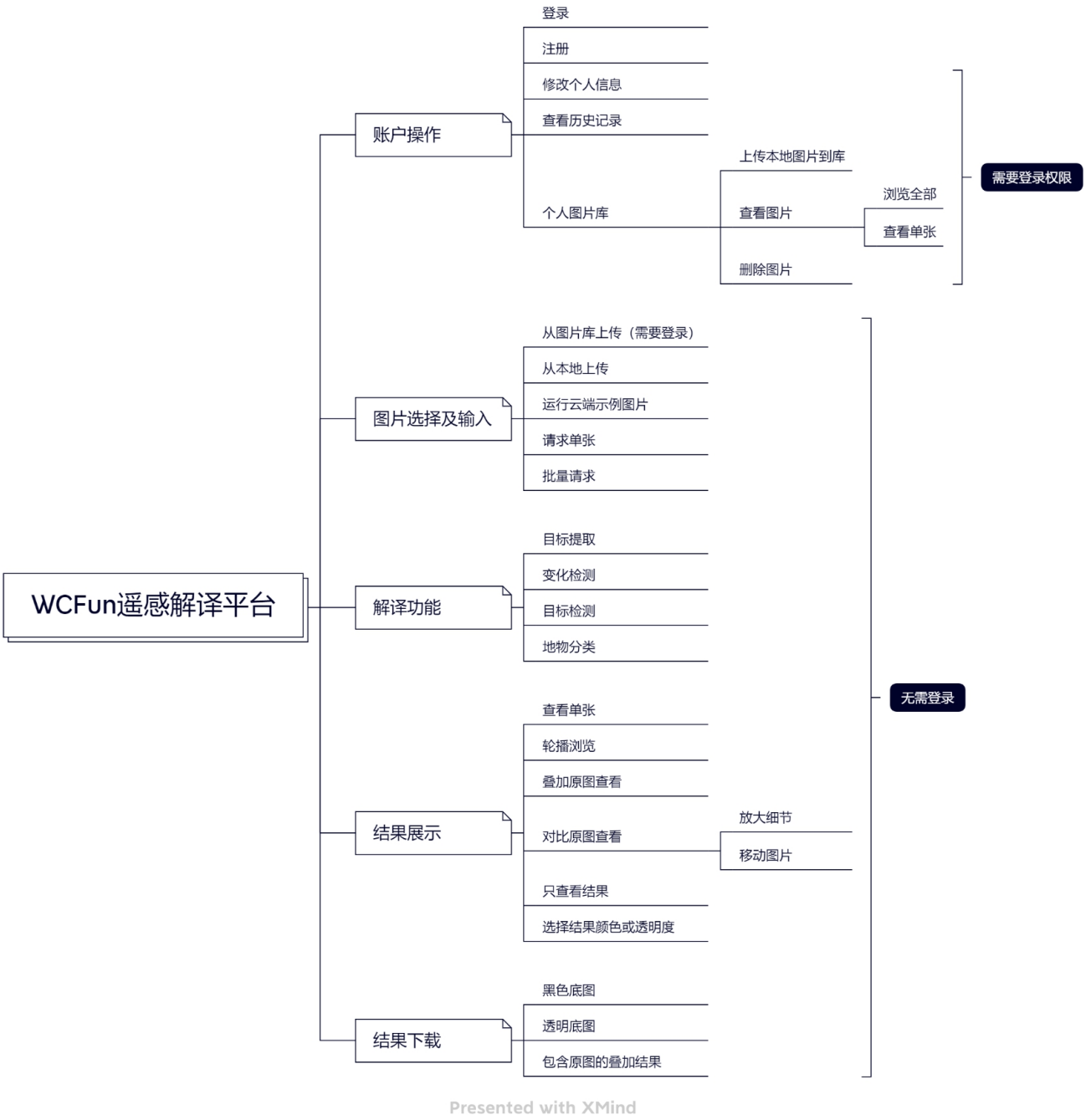
1.1 系统架构图



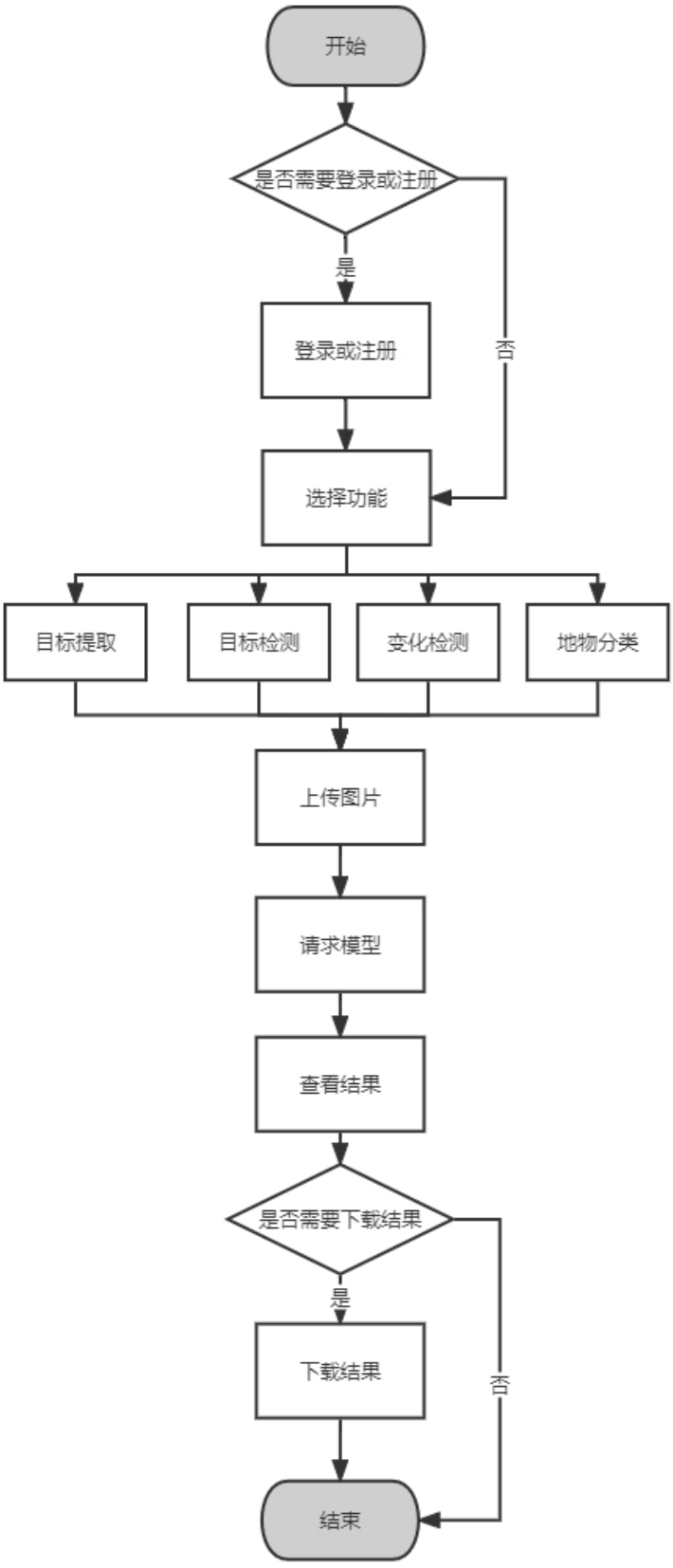
1.2 数据库 ER 图


id
user_id
file1
file2
result
type
time
record
1:n
user
1:n
library
id
user_id
file
time
name
id
name
password
status
avatar
phone
verity
content

1.3 功能模块图


WCFun遥感解译平台
账户操作
登录
注册
修改个人信息
查看历史记录
个人图片库
上传本地图片到库
查看图片
浏览全部
查看单张
删除图片
需要登录权限
图片选择及输入
从图片库上传（需要登录）
从本地上传
运行云端示例图片
请求单张
批量请求
解译功能
目标提取
变化检测
目标检测
地物分类
结果展示
查看单张
轮播浏览
叠加原图查看
对比原图查看
放大细节
移动图片
只查看结果
选择结果颜色或透明度
结果下载
黑色底图
透明底图
包含原图的叠加结果
无需登录
Presented with XMind

1.4 整体流程图


开始
是否需要登录或注册
是
否
登录或注册
选择功能
目标提取
目标检测
变化检测
地物分类
上传图片
请求模型
查看结果
是否需要下载结果
是
否
下载结果
结束

二、关键技术

1.前端网页

本项目使用 Vue 框架和 Element Plus 组件库构建网页 Web 端。利用基于 Vue 的 Doob 模板快速搭建响应式页面，并实现网页明暗两种风格的任意切换。

在图像上传方面，使用 Element Plus 组件实现图片上传和图片预览，运用 Axios 向后端服务器发送请求，实现与服务器的交互。对于结果的图片的展示，采用绝对定位的方式将结果图片与原图进行叠加，并使用颜色选择器或透明度选择器改变 mask 背景图片样式，以实现结果图片颜色和透明度的实时改变；通过自己编写子组件，监听鼠标移动和滚轮等相关事件来实现结果图与原图的同步对比缩放移动查看；特别的，对于目标检测任务，使用 canvas 画布将接收到的检测目标信息绘制到原图上。

在多图片批处理预测时，采用函数递归调用的方式向后端发送请求，实现了请求同步，保证结果图片返回的顺序和正确性。对于图片历史记录，使用懒加载和分页的方式方便用户查阅。

对于进度条，是调用了 js 的计时器功能来实现的。因为我们有单张图片处理和批处理，所以必须要有一个单张进度 p1 和总体进度 p2.对于每个进度，在进度到达 100 之后，使用 setInterval 来设置 1s 的时间间隔，使得进度条动画得以显示。完成之后再退出，最后系统返回检测结果。

2.后端服务器

本项目使用 Flask，Spring Boot，Mybatis Plus 等框架在搭建并部署了 Python 服务器与 Java 服务器，之后采用 natapp 用内网穿透的方式将其部署在本地。其中 Python 服务器负责处理四大检测相关功能的请求，以及神经网络模型的调用；Java 服务器负责基础功能的实现，数据库的交互等，二者相互结合，共同实现了强大的云端服务器功能。

2.1 使用 shiro+jwt+redis 实现登录功能

shiro 是 Java 的一个安全框架，相对于 Spring Security，shiro 更为简单。本项目涉及到登录功能，用户和角色很多，我们使用 shiro 来实现用户的身份验

证和鉴权。在其中涉及到了 profile 和 realm 进行用户-角色-权限的对应。具体流程如下：

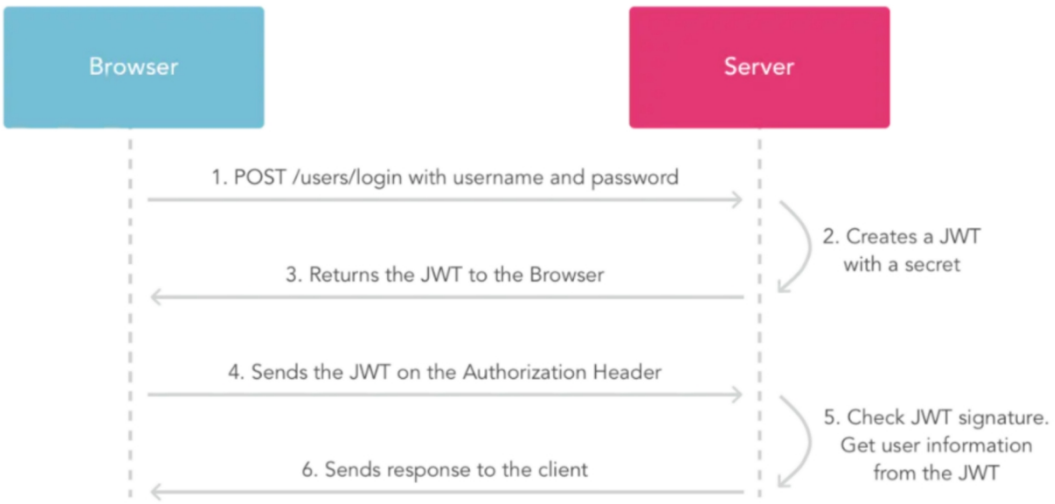
1.前端将用户名、密码等信息请求到服务端。

2.服务端接到该请求头，获取用户的 profile，并加密为 JWT(JSON WEB TOKEN)，并将 token 返回给前端，并且将该 token 存到 redis 中，为期一个月，用于令牌刷新

3.前端拿到 token 后存到浏览器。在之后的业务请求时，请求头带上 token，访问网关服务。

4.网关服务接到请求后，jwt 解析请求头令牌，并将令牌发送给认证中心。认证中心以 shiro 为例，shiro 不能直接使用该 token 字符串，需要对 token 进行封装。

5.shiro 拿到封装后的 token 进行判断是否有效。如果有效，那么请求继续；如果无效，请求终止。



对于短信验证码的登录，这里调用了腾讯云的短信验证码服务接口用以发送短信，后端并不直接将验证码返回给前端，而是将其保存在 redis 里，为期 5 分钟，用户进行登录，注册等操作时后端都要从 redis 里获取正确的验证码与用户输入的进行比较，这样就保证了数据的安全性。

2.2 使用 flask+springboot 双后端加快访问速度

在进行变化检测，地物分类等涉及到 python 模型的加载过程中，若只用 flask，由于 python 的特性会降低后端的运行速度与并发性能，且无法完成

springboot 可以完成的复杂业务逻辑；若只用 springboot，则要使用 process 类调用 python 代码，每次运行都要重新加载模型，效率很低。为了提高效率，我们使用 springboot+flask 双后端的形式，先在 flask 中加载好模型，用 springboot 直接调用 flask 的接口进行结果输出，这样就只需要在开始的时候加载一次模型，之后再使用无需加载。这样处理后每个批次的平均执行时间在 1 秒左右，也保证了业务逻辑的完整性。

2.3 使用 natapp 进行内网穿透

natapp 是基于 ngrok 的反向代理软件，通过在公网和本地运行的 Web 服务器之间建立一个安全的通道。natapp 可捕获和分析所有通道上的流量，便于后期分析和重放。本项目使用了 natapp 做内网穿透，直接在自己的电脑上运行后端，加快了模型的加载速度，减少了成本。

2.4 优化图片加载速度

对于图片加载，由于本项目所用的输入输出图片均为卫星遥感图片，普遍比较大，如果不加处理的话会很慢，用户体验会很差。为此，我们采用了三种方案来解决：

1） 使用内容分发网络 cdn。我们使用七牛云的 cdn，七牛云不仅提供了 cdn，还提供了强大的图片压缩功能使得图片更加容易加载。每次上传，将运行结果和用户输入同时上传到七牛云对象存储中，同时更新数据库 record 记录，这样前端只需要访问七牛云的 url 就可以很快地加载图片。

2） 前端使用图片懒加载。Vue 提供了图片懒加载插件 lazy-load，前端个人中心页面就使用此插件实现了懒加载，优化了用户体验。

3） 图片分页。对于历史记录这种有大量图片的页面，我们使用了图片分页来进一步优化页面加载速度。

3. 变化检测算法

3.1 变化检测模型——BIT-CD

BIT-CD 模型使用一个双时空图像转换器(BIT)，以高效和有效地建模上下文。为了将变化用语义标记表示，将双时态图像表示为几个标记，并使用变压器编码

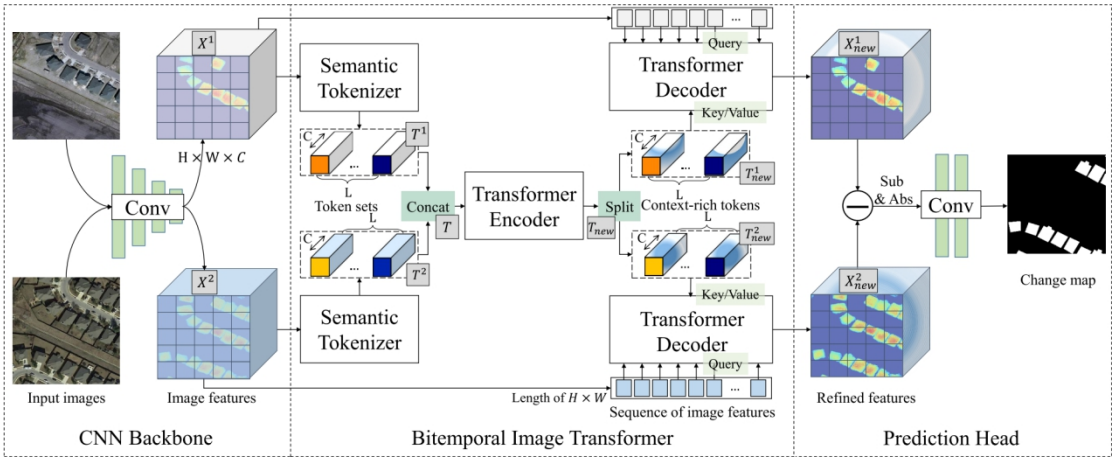
器在紧凑的基于标记的时空中建模上下文。然后将学习到的丰富上下文的令牌反馈到像素空间，通过转换器解码器细化原始特征。将 BIT 融合到一个基于深度特征差异的 CD 框架中。在三组光盘数据集上的大量实验证明了该方法的有效性和高效性。模型仅使用三倍低的计算成本和模型参数，就显著优于纯卷积基线。基于一个朴素的骨干(ResNet18)，没有复杂的结构(例如，特征金字塔网络(FPN)和 UNet)，超越了几种最先进的 CD 方法，包括在效率和准确性方面优于四种最近的基于注意力的方法。

模型的总体结构如图所示。将 BIT 合并到一个普通的 CD 管道中，以利用卷积和转化器的优势。模型首先使用几个卷积块来获得每个输入图像的特征图，然后将它们输入 BIT 来生成增强的双时特征。最后，将得到的特征映射输入到预测头，产生像素级的预测。关键在于 BIT 学习并联系高级语义概念的全局上下文和反馈，以受益于原始的双时态特征。模型有三个主要组件:

1)连体语义标记器，它将像素分组成概念，为每个时间输入生成一组紧凑的语义标记;

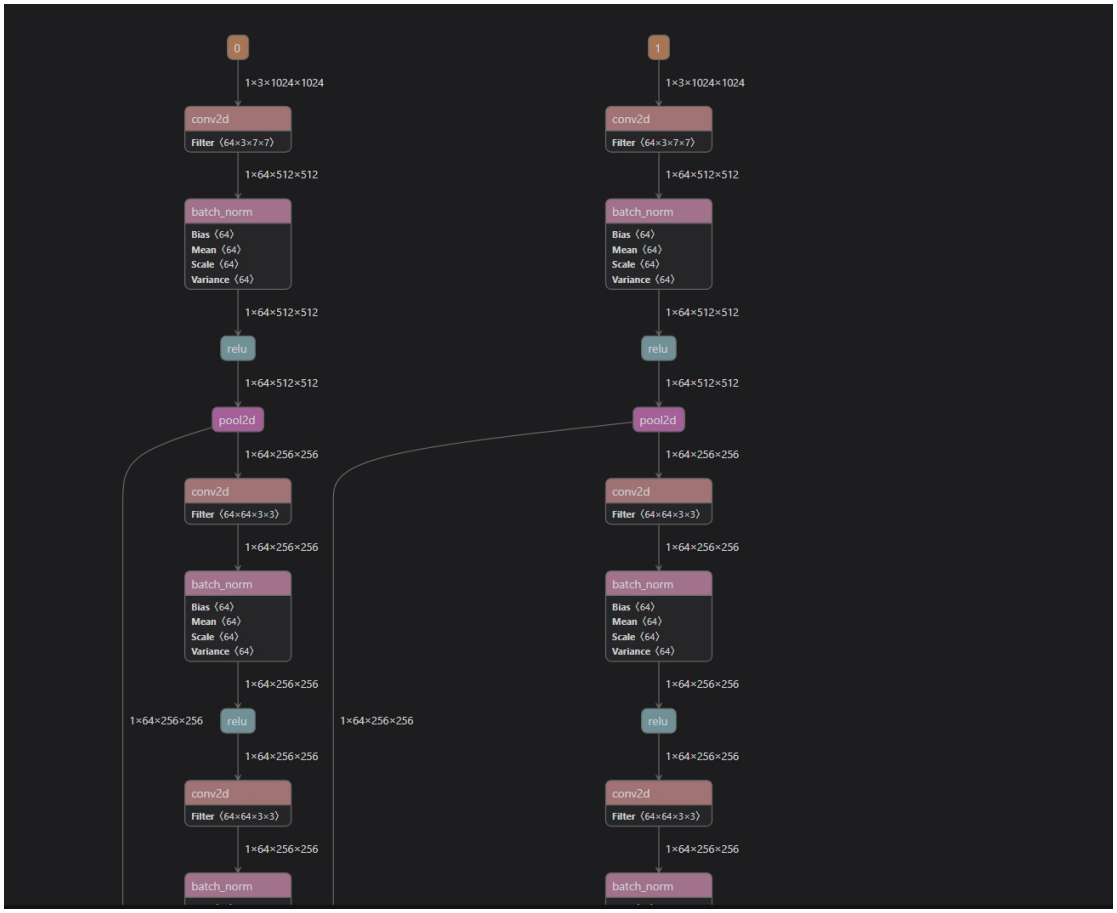
2)一个转化编码器(TE)，在基于符号的时空中建模语义概念的上下文;

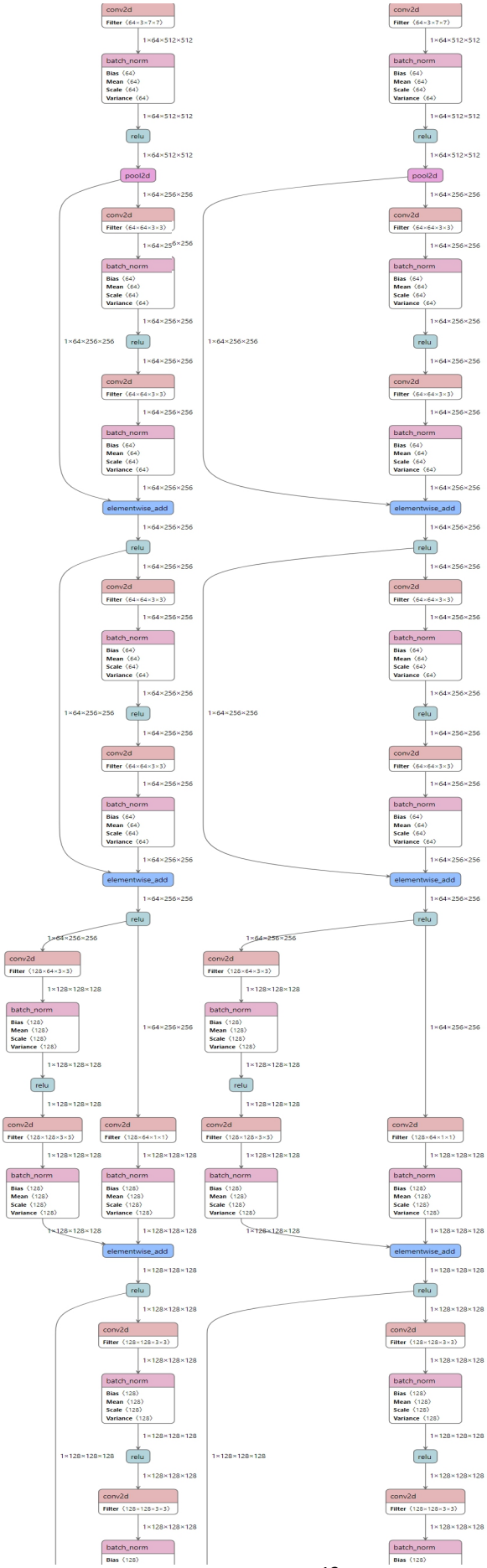
3) Siamese TD 转化解码器，将对应的语义标记投影回像素空间，获得每个时态的精细化特征图。



由图中可见，模型首先用一个 CNN 主干提取输入图片中的高级特征，之后使用 BIT 来细化双时态图的特征,计算每个特征的令牌集,然后又使用 transformer 编码器在基于符号的时空中建模语义概念的上下文，使用解码器细化原始特征，最后通过预测头得到结果。

本项目所用的 BITS-CD 模型结构较为复杂，我们选取了部分结构如图所示：


0
1×3×1024×1024
conv2d
Filter 〈64×3×7×7〉
1×64×512×512
batch_norm
Bias 〈64〉
Mean 〈64〉
Scale 〈64〉
Variance 〈64〉
1×64×512×512
relu
1×64×512×512
pool2d
1×64×256×256
conv2d
Filter 〈64×64×3×3〉
1×64×256×256
batch_norm
Bias 〈64〉
Mean 〈64〉
Scale 〈64〉
Variance 〈64〉
1×64×256×256
1×64×256×256
relu
1×64×256×256
conv2d
Filter 〈64×64×3×3〉
1×64×256×256
batch_norm
1
1×3×1024×1024
conv2d
Filter 〈64×3×7×7〉
1×64×512×512
batch_norm
Bias 〈64〉
Mean 〈64〉
Scale 〈64〉
Variance 〈64〉
1×64×512×512
relu
1×64×512×512
pool2d
1×64×256×256
conv2d
Filter 〈64×64×3×3〉
1×64×256×256
batch_norm
Bias 〈64〉
Mean 〈64〉
Scale 〈64〉
Variance 〈64〉
1×64×256×256
relu
1×64×256×256
conv2d
Filter 〈64×64×3×3〉
1×64×256×256
batch_norm


conv2d
Filter {64×3×7×7}
batch_norm
Bias {64}
Mean {64}
Scale {64}
Variance {64}
relu
pool2d
elementwise_add
Filter {128×64×3×3}
Filter {128×128×3×3}
Filter {128×64×1×1}
Bias {128}
Mean {128}
Scale {128}
Variance {128}

上图即 BIT-CD 的网络结构图。

3.2 输入端

3.2.1 数据增强

对于数据增强方面，由于本项目采用数据集的数据量较小，所以数据增强显然是必不可少的。本项目采用了 Compose 组合多种变换方式，将包含的变换按顺序串执行。对于训练集，首先进行随机色彩变换、随机模糊、随机翻转和随机旋转，之后以 50%的概率交换输入的两个图像，并将数据归一化到[0.5，0.5，0.5]；对于验证集，输入原始尺寸影像，对输入影像仅进行与训练集相同的归一化处理。

3.2.2 数据集介绍

本项目采用赛事官方给定的数据集。一共包含 1000 个样本，数据主要来源于北航 LEVIR 团队的公开论文 。

本数据集包含赛事的变化检测任务所需训练（training）和验证（evaluation）数据，以及测试（testing）影像对。train_data.zip 对应训练/验证集，其中共含有 637 个遥感影像对以及对应的二值变化标签，每个时相的影像均包含 RGB 三个波段（亮度值已量化到 0-255），长、宽均为 1024。test_data.zip 对应测试集，其中仅包含双时相影像对，而不提供变化标签。

3.3 BackBone

BIT-CD 模型使用改进的 ResNet18 来提取双时态图像特征图。原始的 ResNet18 有 5 个阶段，每个阶段下采样 2 个。将最后两个阶段的步幅替换为 1 并添加对 ResNet 进行点向卷积(输出通道 C = 32)，降低特征维数，再进行双线性插值层，得到降采样因子为 4 的输出特征图，减少空间细节的损失。BIT-CD 将这个骨干命名为 ResNet18_S5。为了验证所提方法的有效性，还使用了两个较轻的骨干 ResNet18_S4/ResNet18_S3，它们只使用了 ResNet18 的第一个 4/3 阶段。

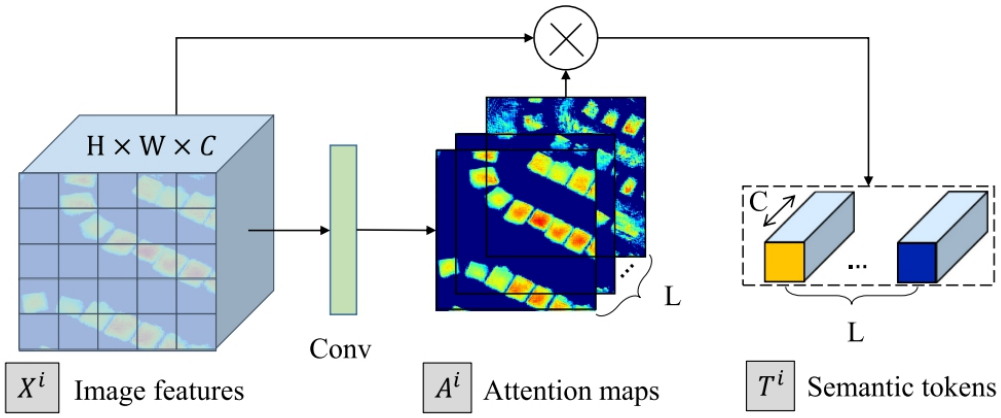
3.3.1 RS 图像的处理

对于 RS 图像的处理，一般可分为两大主流。一种是两阶段解决方案，训练一个 CNN/完全卷积网络(FCN)对双时态图像进行单独分类，然后比较它们的分类

结果进行更改决策。这种方法只有当变更标签和双时态语义标签都可用时才实用。另一种是单阶段解决方案，直接产生双时态图像的变化结果。相似度检测过程是将双时态图像分组成一对对 patch，对每一对 patch 使用 CNN 来获得其中心预测。之后使用 FCNs 从两个输入直接生成 HR 变化图，通常是更多的比补丁级的方法更高效有效。与现有的基于注意力的 CD 方法直接建模基于像素的空间中任何一对元素之间的密集关系不同，BIT-CD 模型提取了一些语义来自图像的符号，并在基于符号的时空中建模上下文。然后使用由此产生的上下文丰富的标记来增强像素空间中的原始特征。场景中的兴趣变化可以用几个视觉词汇(标记)来描述，而每个像素的高级特征可以用这些语义标记的组合来表示。结果表明，该方法具有较高的效率和性能。

3.3.2 Semantic Tokenizer 语义赋予器

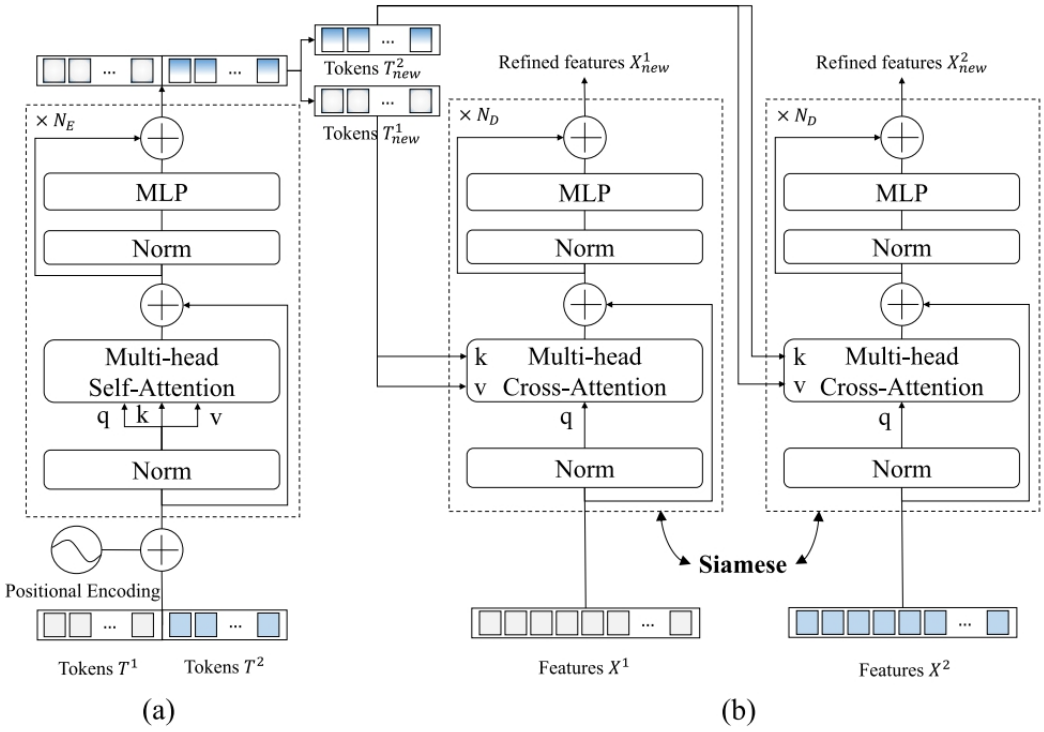
语义赋予器的主要作用是获取语义标记。语义概念可以通过双时态图像共享。为此，、使用 Siamese 标记器从每个时态的特征图中提取紧凑的语义标记。与 NLP 中的标记赋予器(将输入句子分割成几个元素(即单词或短语)，并用一个标记向量表示每个元素)类似，我们的语义标记赋予器将整个图像分割成几个视觉单词，每个单词对应一个标记向量。如图所示，为了获得紧凑的令牌，令牌赋予器学习了一组空间注意映射，将特征映射在空间上池化为一组特征，即令牌集。



3.3.3 转化编码器和解码器

对于转化编码器（Transformer Encoder），在为输入的双时态图像获得两个语义标记集 T1，T2 之后，用 TE 对这些标记之间的上下文进行建模。这样做是

为了基于标记的时空中的全局语义关系可以被转换器充分利用，从而为每个时间产生上下文丰富的标记表示。如图 (a)所示，首先将两组令牌连接到一个令牌集 TE R2LxC，并将其输入到 TE 以获得一个新的令牌集 Tnew。最后，将 token 分成两个集合T_{new}^{i} (i = 1,2)。新 TE 由多头自我意识(MSA)和多层感知器(MLP)块 Ne 层组成[图 (a)]。与原变压器使用后范数残差单元不同，遵循 ViT 采用前范数残差单元(prenorm)，即层归一化发生在 MSA/MLP 之前。



对于转化解码器（Transformer Decoder），转化编码器已经为每个时间图像获得了两组上下文丰富的令牌 Tnew (i = 1,2)。这些背景丰富新令牌包含紧凑的高级语义信息，这些信息很好地揭示了兴趣点的变化。现在，我们需要将概念的表示投影回像素空间，以获得像素级特征。为此，BIT-CD 使用改进的 Siamese TD 来细化每个时间点的图像特征。如图 (b)所示，给定一个特征序列 X'， TD 利用每个像素与令牌集 T 之间的关系获得精细特征 X。将 X'中的像素视为查询和标记作为键。每个像素都可以用紧凑的语义标记的组合表示。

3.4 输出端

3.4.1 损失函数

在训练阶段，模型采用最小化交叉熵损失来优化网络参数。形式上，损失函数定义为

$$L = \frac{1}{H_0 \times W_0} \sum_{h=1,w=1}^{H,W} l(P_{hw,} Y_{hw})$$

其中 $l(P_{hw},y) = -\log(P_{hwy})$为交叉熵损失，$Y_{hw}$为位置(h, w)像素的标签。

3.4.2 dropout 的增加

在训练阶段，在 Transformer 解码器和译码器都增加了 0.1 的 dropout，以提高模型的鲁棒性。在模型推理阶段，取消 dropout，提高推理准确性。

3.5 训练及参数调整

具体训练的时候迭代了 200 个 epoch，每 10 个 step 记录一次日志，每训练 1 个 epoch 保存一次模型参数，训练集和测试集按照 95：5 的结构划分。

训练平台：飞浆 AI Studio

CPU: 4 Cores

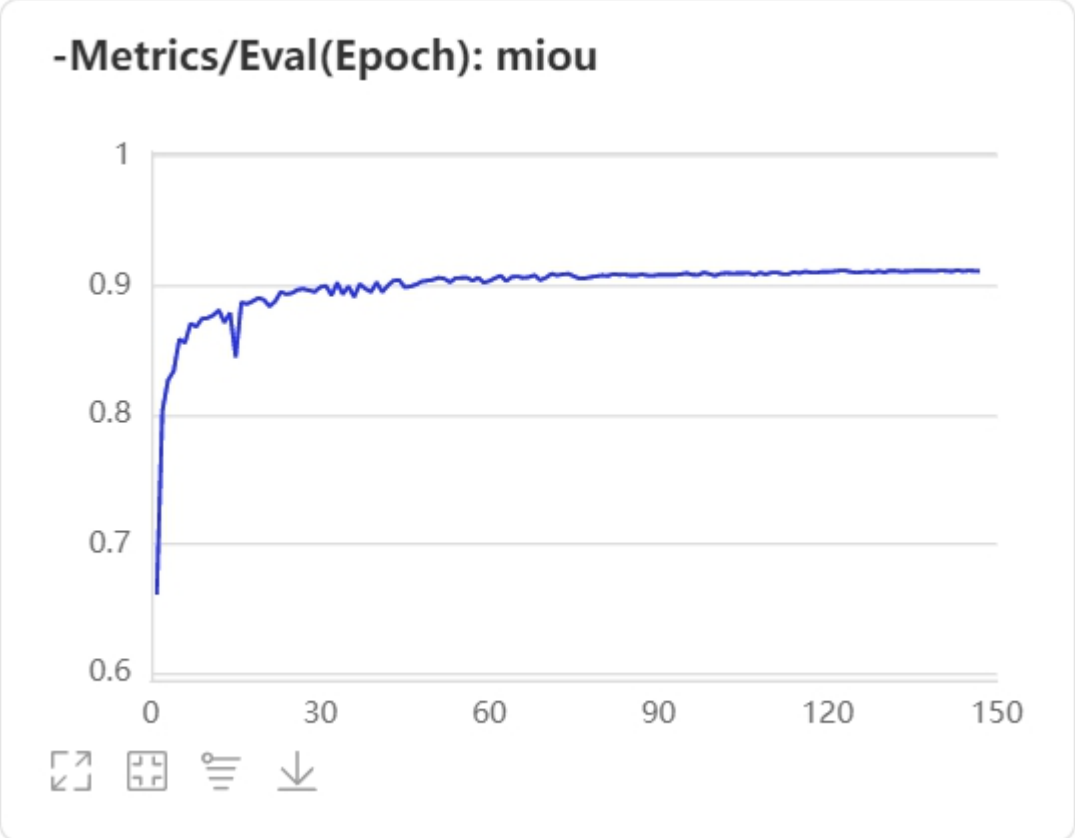
内存：32GB

RAM： 32GB

磁盘： 100GB

显卡：Tesla V100

部分训练过程中的 miou 分数变化：



超参数设置：

表 1 BIT-CD（主干网络 resnet34）超参数设置

Dataloader	**Batch_size**	**6**
	shuffle	**True**
	Epoch	**200**
	workers	**4**
Lr_scheduler	**Type**	**StepDecay**
	Step Size	**86**
	Gamma	**0.981**
	Init lr	**0.0005**
Optimizer	**Type**	**Adamax**
	ClipGradByNorm	**300**
	Weight_decay	**0.0003**

3.6 主要创新贡献

3.6.1 数据增强

由于官方提供的数据集有限，遥感影像对在 637 左右，这在深度学习任务中

是很少的，因此为了能够使得模型的鲁棒性更加优秀，我们需要做大量的数据增强。

基础的数据增强有随机翻转、随机亮度增强等，这些是对图像的领域一些基础的增强。为了增加更多的增强手段，我们还运用了其他的数据增强方式，如随机色彩变换，随机模糊，随机旋转，以 50%的概率交换两个输入图像等操作。通过这几种方式就能够大大增加数据量，从而让模型不会在既定的数据集上过拟合，大大增强了模型的鲁棒性。

在具体的实验中，使用随机色彩变换和随机模糊后，模型的 F1 成绩上涨约 2%；使用随机旋转后，模型的 F1 成绩上涨约 0.5%；使用随机交换后，模型的 F1 成绩上涨约 0.5%.

3.6.2 优化器调整

将 Adam 优化器改为 Adamax 优化器，Adamax 算法是基于无穷大范数的 Adam 算法的一个变种，使学习率更新的算法更加稳定和简单。

其参数更新的计算公式如下：

$$t = t + 1$$

$$moment_{out} = \beta_1 \times moment + (1 - \beta_1) \times grad$$

$$\inf\_norm\_out = \max(\beta_2 \times \inf\_norm + \in, |grad|)$$

$$learning_{rate} = \frac{learning\_rate}{1 - \beta_1^t}$$

$$param_{out} = param - learning\_rate \times \frac{moment\_out}{\inf\_norm\_out}$$

并且，我们在优化器中增加了 weight_decay,使用 L2 正则化策略，增强训练模型的鲁棒性；增加了 L2 范数的梯度裁剪，防止梯度爆炸；使用基于步长衰减的学习率衰减策略，使模型逐步移动到局部最优解中。

在具体的实验中，增加了正则化和梯度裁剪后，模型的 F1 成绩上涨约 1.5%；将 Adam 优化器改为 Adamax 优化后，模型的 F1 成绩上涨约 0.5%.

3.6.3 backbone 的优化与调整

原始的 BIT-CD 模型使用改进的 ResNet18 来提取双时态图像特征图。为了进一步提高模型的鲁棒性，我们在 Transformer 解码器和译码器都增加了 0.1 的

dropout。在实际训练过程中，我们发现调参以后的最佳模型的 F1 分数只有 0.857。在 ResNet18 之后提出的 ResNet34 较 ResNet18 网络结构更为复杂，效果也更好。在换用了 ResNet34 并经过一系列调参之后，最佳模型的 F1 达到了 0.867，同比增长一个百分点。但是，ResNet18 的运行速度要比 ResNet34 快得多。所以在最后的模型部署中，两种模型都有使用，其中 web 端为了保证运行速度，使用的是 ResNet18 模型。

3.6.4 集成学习

集成学习是训练多个机器学习模型并将其输出组合在一起的过程。组织以不同的模型为基础，致力构建一个最优的预测模型。组合各种不同的机器学习模型可以提高整体模型的稳定性，从而获得更准确的预测结果。集成学习模型通常比单个模型更可靠。在尝试了大量模型之后，我们最终选取了四个模型进行集成，分别是准确率为 0.867 和 0.864 的骨干网络为 ResNet34 的模型，以及准确率为 0.857，0.854 的骨干网络为 ResNet18 的模型，通过取平均值的方法进行集成，集成以后，得到了 F1 为 0.87611 的本组最佳模型。

4. 其它三种算法简介

4.1 概述

对于其它三种算法，包括目标提取、目标检测、地物分类，我们均使用官方提供的基线训练出基本的模型，可以被正常使用。

4.2 目标提取

使用 DeepLabV3p 模型，在 Massachusetts Roads 道路提取数据集上进行训练和测试，实现了遥感影像的道路提取功能，在测试集上 F1 达到了 0.75。

4.3 目标检测

使用 PPYOLO 模型，并在 RSOD 遥感影像目标检测数据集上进行训练和测试，使用了其中四种类型的遥感地物目标，分别为 playground、oiltank、overpass、aircraft，在测试集上 bbox mAp 达到了 88.56。

4.4 地物分类

使用 DeepLabV3p 模型，在遥感影像地块分割数据集上进行训练和测试，实现了对遥感影像中感兴趣的类别进行提取和分类，在测试集的五种地物类型 F1 分别为 0.62、0.78、0.63、0.46、0.94。