

**中国矿业大学计算机学院**

**2017级本科生课程报告**

视频全量目标分析与建模

|  |  |
| --- | --- |
| 课程名称 | 《应用软件开发实践》 |
| 报告时间 | 2020.06.15 |
| 学生姓名 | 李治远 |
| 学 号 | 07172757 |
| 专 业 | 计算机科学与技术 |
| 任课教师 | 薛猛老师 |

评 分 表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **课程教学目标** | **考查方式与考查点** | **占比** | **得分** |
| 1 | **目标1：**能够采用结构化方法或面向对象方法分析系统需求。 | 通过学生答辩及软件验收情况，考察其知识熟练应用程度。考察撰写的报告和设计文稿与原有业务要求的贴近度，描述的清晰性、完整性、无歧义。 | 30% |  |
| 2 | **目标2：**综合考虑设计、测试、维护，对设计方案进行优化，开发满足系统需求和约束条件的软件系统、模块或算法流程。 | 通过学生答辩及软件验收和设计文档，考察学生是否开发完成了满足系统需求和约束条件的软件系统、模块或算法流程。 | 30% |  |
| 3 | **目标3**：熟悉软件开发过程，具有系统的工程研究与实践经历。 | 通过答辩，考察学生需求分析、方案设计、详细设计、编码、测试等各环节中对于软件开发和管理技术的综合应用情况。 | 15% |  |
| 4 | **目标4：**掌握软件需求分析、设计、编码、测试等环节的常用技术和工程开发工具。 | 通过答辩，考察学生在分析、设计、编码和测试过程中，对需求分析、软件设计、源代码版本管理、软件测试等计算机辅助软件工程工具的使用情况。 | 15% |  |
| 5 | 目**标5：**理解并遵守计算机职业道德和规范，具有良好的法律意识、社会公德和社会责任感。 | 通过应用软件开发综合实训环节的选题和设计文档，考察学生是否具有良好的法律意识、社会公德和社会责任感，是否理解并遵守计算机职业道德和规范。 | 10% |  |
| 总分 | | | |  |

**摘 要**

数字图像已经称为信息传递的有利载体，广泛应用于社会生活的各个领域，对图像中的街景进行分析和理解是计算机视觉的目标之一。街景与人们的日常生活紧密相关，对这类图像进行语义分割和理解具有广阔的前景，但也充满挑战。

目前的图像分割算法大多是针对某一类图像进行的，且图像场景比较简单，包含的对象类别比较少；同时，一般的图像分割算法对图像的旋转、缩放和亮度信息比较敏感，势必会造成因上述信息的变化而产生不同的分割结果，从而产生误分割。本文利用Pytorch深度学习框架对图像进行语义分割，针对于两种不同的数据集进行训练，对图片和视频分别进行预测。然后使用Flask框架和小程序对项目进行展示，具体内容与成果如下：

1. 介绍了语义分割的概念，分析了项目的需求，详细介绍了卷积神经网络中卷积、池化的原理，激活与损失函数的计算方法。
2. 在街景图像的处理过程中，使用归一化对图片中的每个像素进行处理。训练结束，使用tensorboard对训练过程中的损失和准确率变化进行记录，最后得到了使用Adam优化器对街景图像语义分割准确率达到了95%，损失为0.05。
3. 在客户端可视化中，使用了小程序和网页对项目结果进行展示，详细介绍了客户端实现和服务器部署的过程。

关键字：卷积神经网络；Pytorch；语义分割；U-Net；Flask

**ABSTRACT**

Digital images have become known as beneficial carriers of information transmission and are widely used in various fields of social life. Analyzing and understanding the street scenes in images is one of the goals of computer vision. Street view is closely related to people's daily lives. Semantic segmentation and understanding of such images have broad prospects, but they are also full of challenges.

Most of the current image segmentation algorithms are for a certain type of image, and the image scene is relatively simple and contains fewer object categories. At the same time, the general image segmentation algorithm is more sensitive to the image rotation, scaling and brightness information, which is bound to cause Due to the change of the above information, different segmentation results are generated, resulting in erroneous segmentation. This article uses the Pytorch deep learning framework to semantically segment images, trains on two different data sets, and predicts pictures and videos separately. Then use the Flask framework and small program to display the project, the specific content and results are as follows:

1. Introduced the concept of semantic segmentation, analyzed the needs of the project, introduced in detail the principles of convolution and pooling in convolutional neural networks, and the calculation methods of activation and loss functions.
2. In the processing of street view images, normalization is used to process each pixel in the picture. At the end of the training, tensorboard was used to record the loss and accuracy change during the training process. Finally, the Adam optimizer was used to achieve the accuracy rate of 95% for the semantic segmentation of street view images, and the loss was 0.05.
3. In client visualization, applets and web pages are used to display the project results, and the process of client implementation and server deployment is described in detail.

Key Words: CNN; Pytorch; Semantic segmentation; U-Net; Flask

目 录

[1 背景 1](#_Toc44229213)

[1.1 软件杯描述 1](#_Toc44229214)

[1.2 语义分割概念 1](#_Toc44229215)

[1.3 论文结构 1](#_Toc44229216)

[2 需求分析 3](#_Toc44229217)

[2.1 功能性需求 3](#_Toc44229218)

[2.1.1 基本需求 3](#_Toc44229219)

[2.1.2 提升需求 3](#_Toc44229220)

[2.2 非功能性需求 3](#_Toc44229221)

[3 卷积网络算法相关技术 4](#_Toc44229222)

[3.1 与传统多层神经网络对比 4](#_Toc44229223)

[3.2 卷积结构 4](#_Toc44229224)

[3.2.1 卷积层 5](#_Toc44229225)

[3.2.2 池化层 6](#_Toc44229226)

[3.3 激活函数 7](#_Toc44229227)

[3.3.1 Sigmoid 7](#_Toc44229228)

[3.3.2 ReLU 7](#_Toc44229229)

[3.3.3 Tanh 8](#_Toc44229230)

[3.4 损失函数 8](#_Toc44229231)

[3.4.1 均方差函数 8](#_Toc44229232)

[3.4.2 交叉熵函数 9](#_Toc44229233)

[4 U-Net网络 9](#_Toc44229234)

[4.1 网络结构 9](#_Toc44229235)

[4.2 Overlap-tile策略 10](#_Toc44229236)

[5 图像语义分割 11](#_Toc44229237)

[5.1 运行环境 11](#_Toc44229238)

[5.2 数据集处理 11](#_Toc44229239)

[5.2.1 CamVid数据集处理 12](#_Toc44229240)

[5.2.2 CityScapes数据集处理 13](#_Toc44229241)

[5.3 建模与分析 15](#_Toc44229242)

[5.3.1 构建Dataset 15](#_Toc44229243)

[5.3.2 网络分析 16](#_Toc44229244)

[5.3.3 图片、视频预测方法 17](#_Toc44229245)

[5.4 实验结果 18](#_Toc44229246)

[5.4.1 训练参数 18](#_Toc44229247)

[5.4.2 训练Loss与Acc 19](#_Toc44229248)

[5.4.3 图片预测结果 20](#_Toc44229249)

[5.4.4 视频预测结果 20](#_Toc44229250)

[6 客户端可视化 21](#_Toc44229251)

[6.1 FLask服务 21](#_Toc44229252)

[6.1.1 基本配置 21](#_Toc44229253)

[6.1.2 网页 22](#_Toc44229254)

[6.1.3 后端服务 23](#_Toc44229255)

[6.2 小程序 24](#_Toc44229256)

[6.2.1 小程序准备 24](#_Toc44229257)

[6.2.2 新建小程序 26](#_Toc44229258)

[6.2.3 小程序界面设计 27](#_Toc44229259)

[6.2.4 主要实现代码 28](#_Toc44229260)

[6.3 服务部署 29](#_Toc44229261)

[6.3.1 配置远程ssh登录 29](#_Toc44229262)

[6.3.2 添加域名云解析 30](#_Toc44229263)

[6.3.3 基础环境安装 31](#_Toc44229264)

[6.3.4 uwsgi启动服务 31](#_Toc44229265)

[6.3.5 nginx转发 32](#_Toc44229266)

[7 项目评估 34](#_Toc44229267)

[7.1 项目优缺点 34](#_Toc44229268)

[7.1.1 项目优点 34](#_Toc44229269)

[7.1.2 项目缺点 34](#_Toc44229270)

[7.2 改进方法 34](#_Toc44229271)

[7.2.1 迁移学习训练模型 34](#_Toc44229272)

[7.2.2 DeepLabV3+模型 34](#_Toc44229273)

[8 总结 35](#_Toc44229274)

[参考文献 36](#_Toc44229275)

# 背景

## 软件杯描述

人工智能结合视觉分析，极大推动各行业视觉应用。人脸识别、车辆测别、车辆智能驾驶等。结合当前高清1080P、2K、4K等视频，利用人工智能进行视觉分析技术，具有广泛得应用场景。

针对2分钟的1080P视频，视频内容街景（行车记录仪、电影等拍摄）内容，需要利用视觉分析技术，对高分辨率视频进行视频图像分析。赛题要求是为了对街景或高楼的该请视频进行语义分割，对识别出的不同种类对象进行标识为不同颜色。

## 语义分割概念

语义分割是一种典型的计算机视觉问题，其涉及将一些原始数据（例如，平面图像）作为输入并将它们转换为具有突出显示的感兴趣区域的掩模。许多人使用术语全像素语义分割（full-pixel semantic segmentation），其中图像中的每个像素根据其所属的感兴趣对象被分配类别ID，如图 1‑1所示。

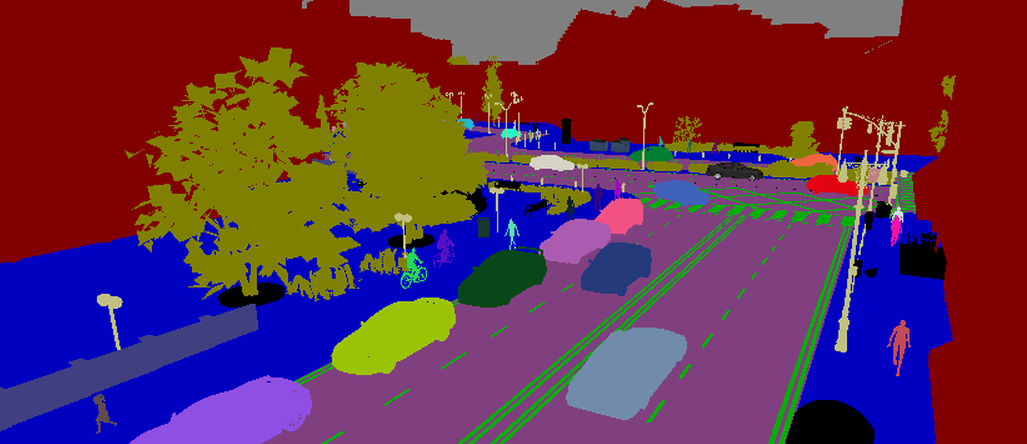


图 1‑1示例图片

早期的计算机视觉问题只发现边缘（线条和曲线）或渐变等元素，但它们从未完全按照人类感知的方式提供像素级别的图像理解。语义分割将属于同一目标的图像部分聚集在一起来解决这个问题，从而扩展了其应用领域。

与其他基于图像的任务相比，语义分割是完全不同的且先进的，例如：

1. 图像分类，识别图像中存在的内容；
2. 物体识别和检测，识别图像中的内容和位置（通过边界框）；
3. 语义分割，识别图像中存在的内容以及位置（通过查找属于它的所有像素）。

## 论文结构

本文主要对视频语义分割方面进行了研究，全文共8章内容，第2章到第8章的研究内容大致如下：

第2章主要介绍项目的一些需求分析，包括功能性需求和非功能需求。

第3章为项目所需的相关技术，详细介绍了所学习到的深度学习中卷积神经网络相关知识。

第4章介绍了实验中使用的网络架构U-Net，分析了其中使用的一些方法及模型。

第5章为语义分割的详细过程，介绍了语义分割常用数据集以及本项目选择的CamVid和CityScapes数据集，也详细介绍了U-Net实验过程及实验结果。

第6章为实验结果可视化，介绍了服务、网页和小程序构建过程，并且包括服务器部署，主要有开发环境、界面设计、接口设计以及测试等。

第7章分析了U-Net网络的优缺点，并提出了一些改进方法。

第8章主要为个人总结课程学习到的知识以及体会。

# 需求分析

## 功能性需求

### 基本需求

1. 视频内移动目标识别，包括人、机动车、非机动车、建筑、植物等；
2. 实现对高清视频的图像语义分割，对每帧的每个点的类别预测；
3. 重点实现对建筑物等固定目标的识别和分割。

### 提升需求

1. 实现移动目标的识别，10种以上；
2. 能够实现实例分割，即同一类别的不同个体的识别；
3. 不仅能够处理图片，还能够处理视频，并输出合成结果；
4. 拥有多个客户端可以展示图片或视频的处理结果，如，小程序、网页等；
5. 能够供其他用户使用，扩展模型上传功能。

## 非功能性需求

1. 响应时间，输入一个图片进行处理，响应时间应≤5s，页面跳转时间应≤0.1s；
2. 资源利用率，投入的服务器这类资源，利用率应达到80%；
3. 数据加密保护，保证数据在采集、传输、处理过程中不被窃取、篡改；
4. 能根据用户权限控制访问数据进行相关的操作；
5. 模块化，当业务流程变动较多，此时将系统功能模块化，支持灵活配置；
6. 类似的组件应该统一设计，在需要的地方可以进行微调；
7. 容错性，在系统出错时，不影响用户的行为操作与数据。

# 卷积网络算法相关技术

卷积神经网络(Convolutional Neural Networks，CNN)本质上是将卷积运算作为网络层之一的神经网络[1]，具有通过卷积操作进行分层抽象表示的能力。因此，CNN在计算机视觉应用领域效果显著。两个关键设计推动着CNN发展，首先，CNN利用图像二维结构以及像素在一个邻域内高度相关性，使用分组连接，避免所有像素单元之间一对一连接，实现网络稀疏性。其次，CNN权值共享，每个通道都是根据相同的滤波器进行卷积生成的。CNN的这些特征使得其与普通神经网络相比，参数更少[2]。

## 与传统多层神经网络对比

* 传统意义上的多层神经网络是只有输入层、隐藏层与输出层。其中隐藏层的层数根据需要而定，没有明确的理论推导来说明到底多少层合适。
* 卷积神经网络CNN，在原来多层神经网络的基础上，加入了更加有效的特征学习部分，具体操作就是在原来的全连接层前面加入了卷积层与池化层。卷积神经网络出现，使得神经网络层数得以加深，“深度”学习由此而来。

## 卷积结构

* 卷积层——通过在原始图像上平移来提取特征。
* 激活层——增加非线性分割能力。
* 池化层——减少学习的参数，降低网络的复杂度（最大池化和平均池化）。
* 全连接层——为了能够达到分类效果。

卷积神经网络的输入数据为没有经过预处理的原始数据，比如验证码原图、原始动画以及音频等数据。输入数据通过卷积、池化、和非线性激活函数映射等一系列操作，将数据特征从输入层中抽象提取出来，这个过程称为前馈运算。CNN的最后一层将图像处理任务转化为计算预测值与真实值之间误差的目标函数，反向传播算法将前馈运算预测出的结果与真实结果之间的误差再由后一层向前传递，通过误差传递调整每层参数，并在更新参数后再次前反馈，前向传播与反向传播往复进行，当预测结果收敛时，完成模型训练[3]，图 3‑1展示了简单的CNN结构图。



图 3‑1简单的CNN结构图

在图像处理应用领域中，一般将RGB彩色图像作为卷积网络的输入，有H行、W列、RGB通道，将输入与参数做一次卷积运算输出，然后再将与做卷积运算，输出，…，一直进行次，得到数据，通过公式3-1计算输入数据y与真实标记之间的误差，结束整个网络。损失函数表示为[4]：

公式中，是的参数。在实际应用中，对于不同的任务，损失函数的形式不同。交叉熵损失函数被用作分类任务中CNN目标函数，其中，c为分类任务的类别[2]。损失函数通常作为回归问题中CNN的目标函数，则有。无论回归任务还是分类任务，计算z之前，都需要一系列运算得到与y同维度的。

### 卷积层

卷积层是CNN的核心构件，它承担了大部分计算量。卷积具有平移不变性，主要负责特征提取。卷积核的参数可以通过学习得到，每个卷积核在空间上较小，对输入数据执行局部加权组合，根据所选择的权重集合，揭示了输入数据的不同特性。网络每一层的输出特征与当前层卷积核相乘后累加，加上偏置项，并通过激活函数非线性映射得到当前层输出。输入数据中包含很多重要信息，根据这些信息可以对输入内容做出强有力的推断，因此选择正确的卷积核来获取最显著特征非常重要。如公式3-2所示：

为该层输入，为卷积核，表示偏置项。

讨论单通道输入，如灰度图片只有灰度值一个通道，单个卷积核情况。输入为，卷积核为，如图 3‑2。与卷积核同大小的感受野(输入上方的方框)首先移动至输入最左上方，选中输入上的感受野元素，与卷积核(图片中间方框)对应元素相乘：

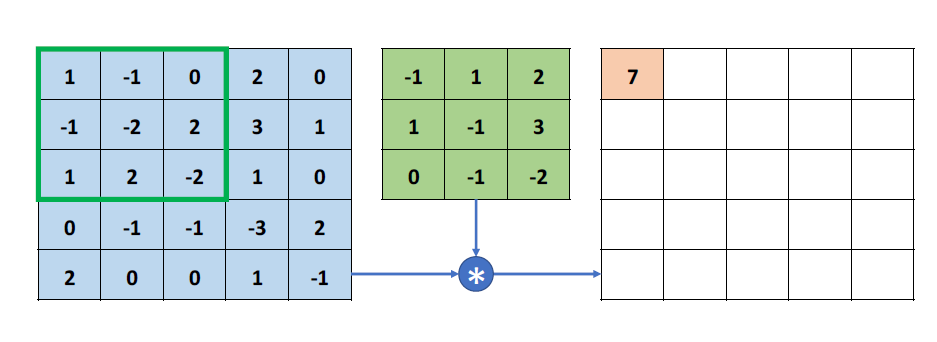


图 3‑2卷积运算-1

运算后得到的矩阵，这9个数值全部相加：，得到标量7，写入输出矩阵第一行、第一列的位置如图 3‑2所示。

完成第一个感受野区域的特征提取后，感受野窗口向右移动一个步长单位(Strides，记为s，默认为1)，选中图 3‑3中方框中的9个感受野元素，按照同样的计算方法，与卷积核对应元素相乘累加，得到输出10，写入第一行、第二列位置。

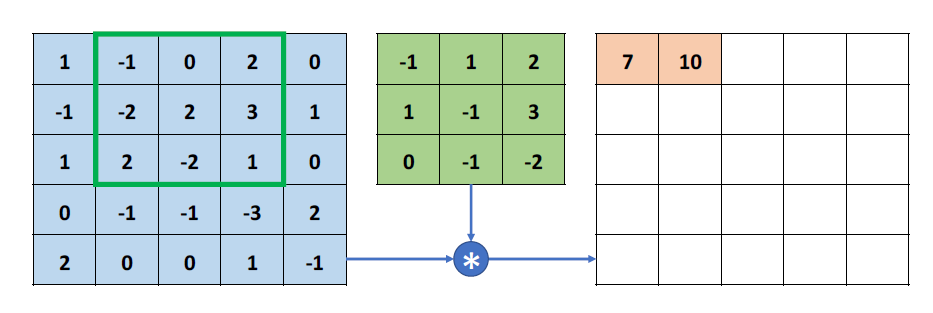


图 3‑3卷积运算-2

按照上述方法，每次感受野向右移动步长单位，若超出输入边界，则向下移动步长单位，并回到行首，直到感受野移动至最右边、最下方位置，如下图 3‑4所示。

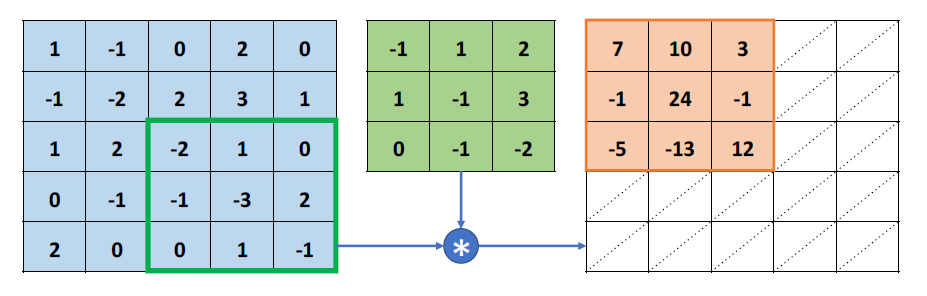


图 3‑4卷积运算-3

### 池化层

任何CNN模型，无论是生物学启发或者纯粹基于学习，都包含一个合并步骤称为池化操作。池化的作用是保持位置和规模在一定程度上不变的情况下对卷积层提取出的特征进行二次提取，从而减少网络原始输入数据抽取出的特征，降低网络后续的计算量，有效避免了过拟合情况的发生。池化操作是对某一位置的邻域进行特征统计，然后用统计值取代原来的特征值[5]。

一些早期生物学启发的CNN依赖于均值池化(ave- pooling)[6]。均值池化将相邻区域所有值的均值作为池化结果，用均值池化进行二次特征抽取主要是为了降低网络对位置变化的敏感性。另一方面，HMAX[5]类网络依赖于最大值池化(max-pooling)。最大池化将相邻区域所有值的最大值作为池化结果，当池化特征非常稀疏时，选择最大池化更合适。均值池化与最大值池化表示如下[6]：

其中。区别于卷积操作，池化层只需要指出池化函数类型、池化操作的窗口大小与步长等超参数。最常见的池化形式是池化层带有大小为的过滤器，沿宽度和高度对输入进行下采样，丢弃36%的激活。图 3‑5用最大值池化以的过滤器对的特征数据，进行二次特征抽取。

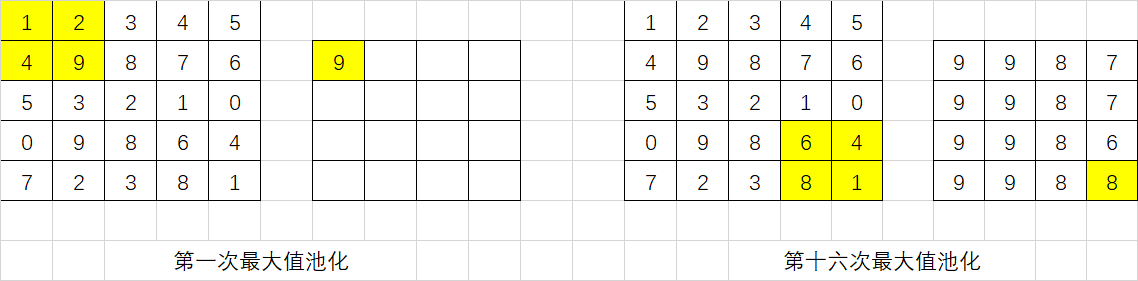


图 3‑5最大值池化过程

池化步骤的本质是模仿生物观察物体时对观测到的物体抽象表示的过程。从图 3‑5可以发现，池化后的结果相比输入特征减少，起到降维作用。

## 激活函数

神经网络中的常见激活函数，与阶跃函数和符号函数不同，这些函数都是平滑可导的，适合于梯度下降算法。

### Sigmoid

Sigmoid函数也叫Logistic函数，定义为：

1. 概率分布——(0,1)区间的输出和概率的分布范围[0,1]契合，可以通过Sigmoid函数将输出转译为概率输出。
2. 信号强度——一般可以将0、1理解为某种信号的强度，如像素的颜色强度，1代表当前通道颜色最强，0代表当前通道无颜色；抑或代表门控值（Gate）的强度，1代表当前门控全部开放，0代表门控关闭。

Sigmoid函数连续可导，如图 3‑6所示，可以直接利用梯度下降算法优化网络参数，应用的非常广泛。

### ReLU

在ReLU(REctified Linear Unit，修正线性单元)激活函数提出之前，Sigmoid函数通常是神经网络的激活函数首选。但是Sigmoid函数在输入值较大或较小时容易出现梯度值接近于0的现象，称为梯度弥散现象。出现梯度弥散现象时，网络参数长时间得不到更新，导致训练不收敛或停滞不动的现象发生，较深层次的网络模型中更容易出现梯度弥散现象。2012年提出的8层AlexNet模型采用了一种名叫ReLU的激活函数，使得网络层数达到了8层，自此ReLU函数应用的越来越广泛。ReLU函数定义为：

函数曲线如图 3‑7所示。可以看到，ReLU对小于0的值全部抑制为0；对于正数则直接输出，这种单边抑制特性来源于生物学。

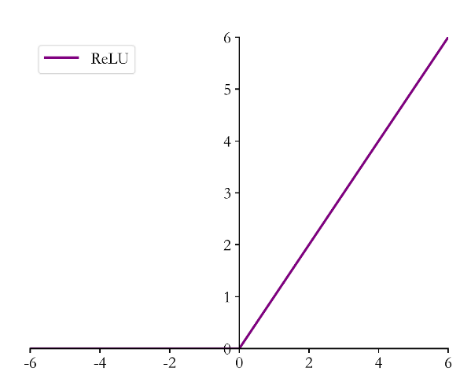


图 3‑6 Sigmoid函数曲线

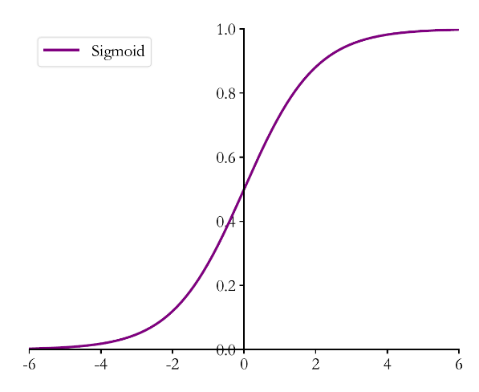


图 3‑7 ReLU激活函数

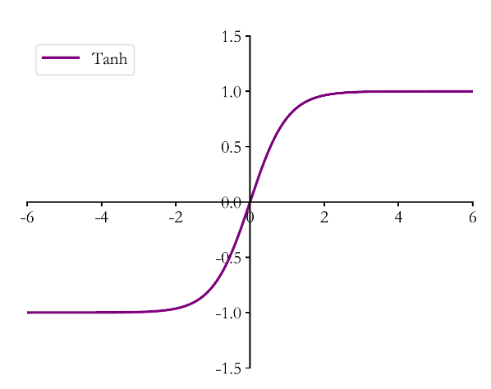


图 3‑8 tanh函数曲线

### Tanh

Tanh函数能够将的输入“压缩”到区间，定义为：

可以看到tanh激活函数可通过Sigmoid函数缩放平移后实现，函数曲线如图 3‑8所示

## 损失函数

### 均方差函数

均方差（Mean Squared Error，简称MSE）函数把输出向量和真实向量映射到笛卡尔坐标系的两个点上，通过计算这两个点之间的欧式距离（欧式距离的平方）来衡量两个向量之间的差距：

其中，为第i个样本的真实值，为第i个样本的输出值，即预测值，为样本数量。MSE误差函数的值总是大于等于0，当MSE函数达到最小值0时，输出等于真实标签，此时神经网络的参数达到最优状态。

### 交叉熵函数

交叉熵（cross entropy）描述的是两个概率分布之间的距离，距离越小表示这两个概率越相近，越大表示两个概率差异越大。

其中p(i)是真实值，q(i)为预测值。交叉熵可以很好地衡量2个分布之间的“距离”，当使用sigmoid作为激活函数的时候，常用交叉熵损失函数而不用均方误差损失函数，因为它可以完美解决平方损失函数权重更新过慢的问题，具有“误差大的时候，权重更新快；误差小的时候，权重更新慢”的良好性质。

# U-Net网络

## 网络结构

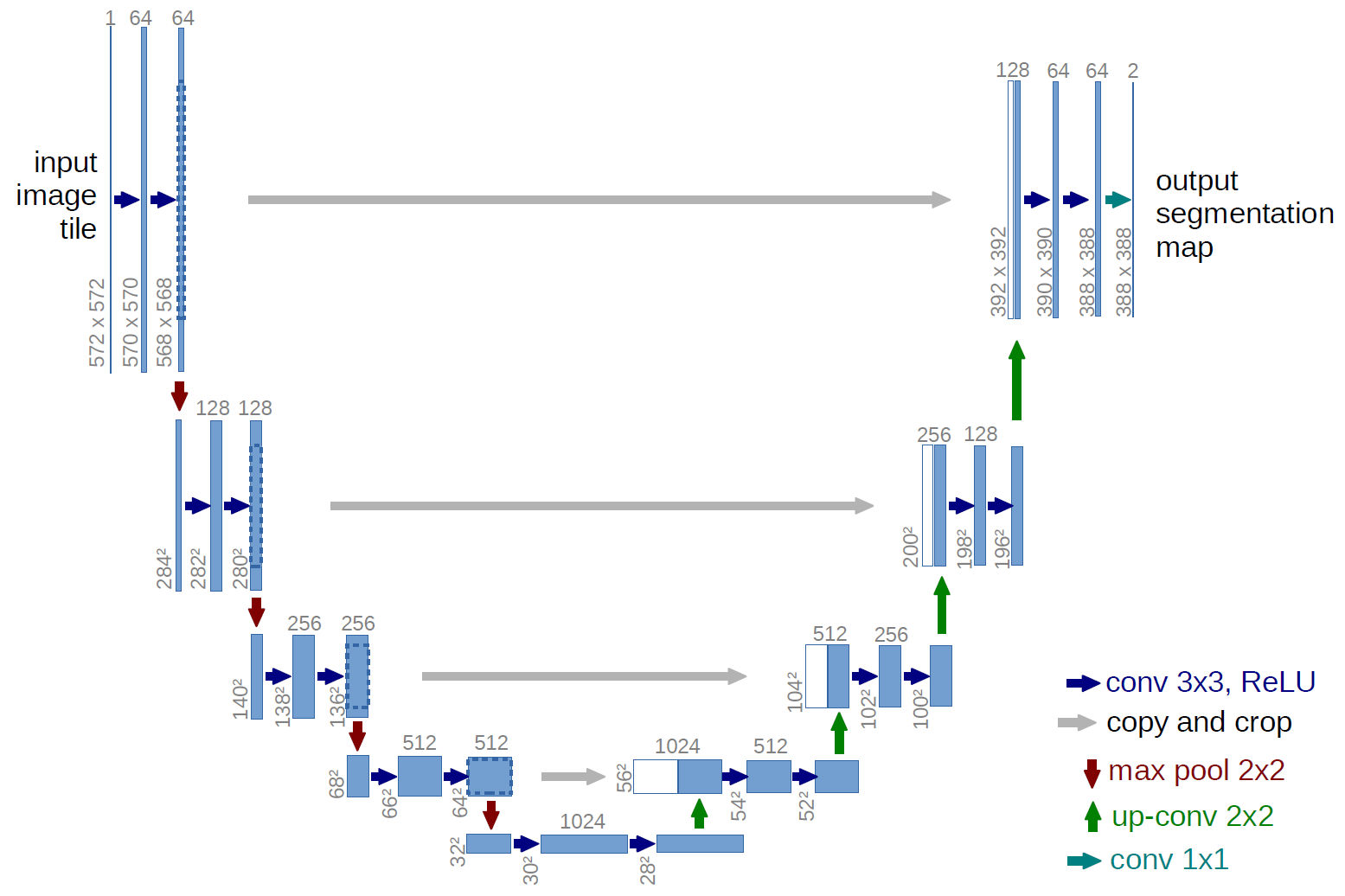


图 4‑1 U-Net网络结构

如图 4‑1所示，U-Net整体结构呈现一个“U”型的状态，先编码（下采样，左侧部分），再解码（上采样，右侧部分），回归到跟原始图像一样大小的像素点分类[7]。

使用全卷积神经网络取代全连接层，全连接层必须固定图像大小而卷积不用，所以这个策略使得，可以输入任意尺寸的图片，而且输出也是图片，所以这是一个端到端的网络。

左边的网络contracting path：首先进行两次卷积核为3的卷积，激活函数选择ReLU，然后再使用池化窗口为2的最大池化完成一次下采样，得到的结果作为下一层的输入，连续做4次下采样就完成了整体的下采样过程。

右边的网络expansive path：使用上采样与左侧pooling层的feature map相结合，然后逐层采用卷积和反卷积上采样到大小heatmap。

最后再经过两次卷积，达到最后的heatmap，再用一个的卷积做分类，这里是分成两类，所以用两个神经元做卷积，得到最后的两张heatmap，然后作为softmax函数的输入，算出概率比较大的softmax类，选择它作为输入给交叉熵进行反向传播训练。

pooling层会丢失图像信息和降低图像分辨率且是不可逆的操作，对图像分割任务有一些影响，对图像分类任务的影响不大，但是上采样可以补足一些图片的信息，但是信息补充的不完全，所以还需要与左边的分辨率比较高的图片相连接起来（直接拷贝过来再裁剪到与上采样图片一样大小），这就相当于在高分辨率和更抽象特征当中做一个折衷，因为随着卷积次数增多，提取的特征也更加有效，更加抽象，上采样的图片是经历多次卷积后的图片，是比较高效和抽象的图片，然后把它与左边不怎么抽象但更高分辨率的特征图片进行连接。

## Overlap-tile策略

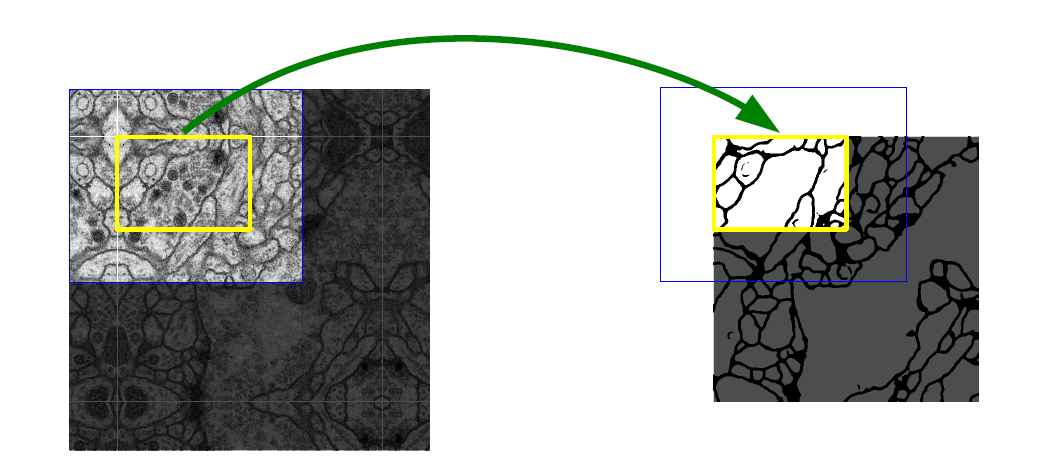


图 4‑2 overlap-tile策略

在对像素点标签值进行预测时，使用的overlap-tile策略，如图 4‑2所示。对图像的某一块像素点（较小框内部分）进行预测时，需要该图像块周围的像素点（较大框内）提供上下文信息（context），以获得更准确的预测。这样的策略会带来一个问题，图像边界的图像块没有周围像素，因此对周围像素采用了镜像扩充。正方形框部分为原始图片，其周围扩充的像素点均由原图沿白线对称得到。这样，边界图像块也能得到准确的预测。

# 图像语义分割

## 运行环境

由于图像语义分割需要较高配置的显卡，这里使用了谷歌的colab平台，显卡具体参数和其他环境如下所示。

* 显卡：Tesla P100 16GB，CUDA 10.1；
* 编程环境：python 3.6.9、torch 1.5.0、notebook 5.2.2和其他科学计算库。

## 数据集处理

常见的街景语义分割数据集如表 5‑1常见街景语义分割数据集所示，本文选择CamVid和CityScapes进行实验。

表 5‑1常见街景语义分割数据集

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名字 | 时间 | 类别 | 图片数量 | 地点 |
| CamVid | 2007 | 32 | 700 | 欧洲 |
| CityScapes | 2016 | 30 | 5000 | 德国、瑞士、法国 |
| Mapillary | 2017 | 66 | 25000 | 美国、欧洲、非洲 |
| Apollo Scape | 2018 | 28 | 143906 | 中国 |

CamVid数据集如图 5‑1-图 5‑2所示。



图 5‑1 CamVid数据集样本



图 5‑2 CamVid数据集标签

CityScapes数据集如图 5‑3-图 5‑4所示。



图 5‑3 CityScapes数据集样本

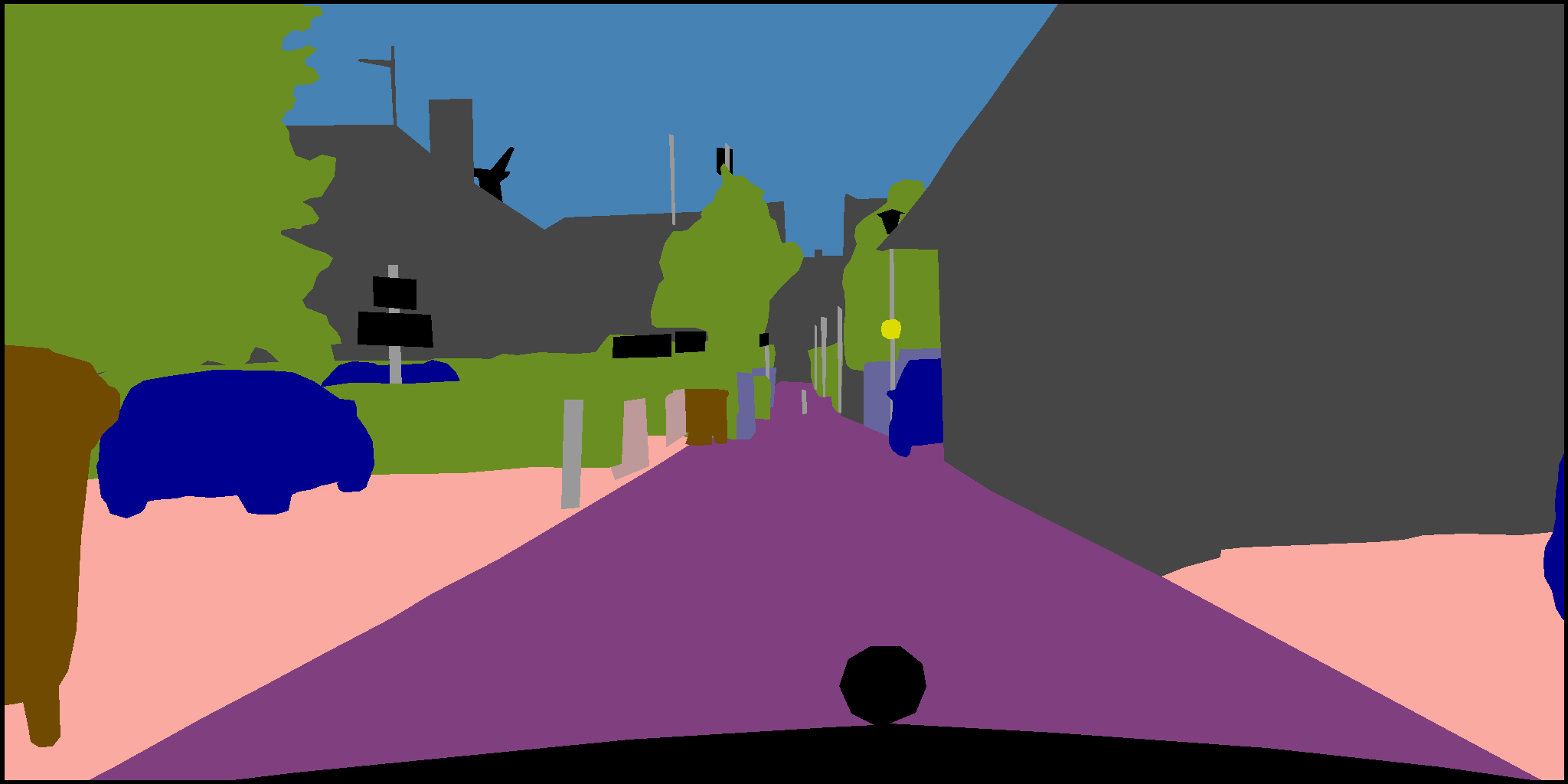


图 5‑4 CityScapes数据集标签

### CamVid数据集处理

**归一化：**归一化是图像预处理中必不可少的一步，其原理就是通过一系列的变化使图像转化为某一种标准的形式给。图像归一化使得图像可以抵抗几何变换的攻击，它能够找出图像中的那些不变量，好的归一化算法可以提高后续特征提取在同一类内的一致性。在神经网络模型中归一化可以加快训练网络的收敛性。下述代码展示此项目的归一化过程，首先将Pillow读取的图片对象转为np.array类型，然后再判断图片的维度。如果维度为2，说明是标签，需要增加一个维度以供归类编码处理，否则为样本，需要对每个像素值标准化为0--1之间的值。最后需要转置矩阵，以为pytorch接受的tensor为[通道数,高度,宽度] ，而np.array为[高度,宽度,通道数]。

*@classmethod*

*def preprocess(cls, pil\_img):*

*img\_nd = np.array(pil\_img)*

*if len(img\_nd.shape) == 2:*

*img\_nd = np.expand\_dims(img\_nd, axis=2)*

*else:*

*img\_nd = img\_nd / 255*

*img\_trans = img\_nd.transpose((2, 0, 1))*

*return img\_trans.astype(float)*

**归类编码：**在机器学习中，常常使用one-hot作为分类问题的编码。在手写数字识别问题中，使用一个长度为10的向量来表示标签值，如：[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]来表示数字1，[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]来表示数字2，等等，也就是说这个长度为10的向量不同位置为1代表相应的数字。

而这里需要对标签图片进行通道数的扩展，在归一化的时候增加了通道数，每个通道是一个矩阵，值为0或者1。初始标签的矩阵每个像素的值都代表所对应的分类，如，数字0代表天空，数字1代表汽车。然后使用第一个通道来表示天空，原标签位置为0的位置在此通道均表示为1，其他位置均为0。用第二个通道来表示汽车，原标签为1的位置在此通道表示为1，其他位置为0，对原标签所有的分类号都建立一个维度后，编码阶段也就完成了。

### CityScapes数据集处理

**数据集获取：**由于CityScapes数据集很大（11GB），并且在国外服务器，国内下载很慢，所以这里使用colab和谷歌云盘对数据集进行储存。

新建一个notebook环境，进入相应的实验目录，首先使用wget命令登陆CityScapes网站，将登陆状态cookie保存在cookies.txt文件中，然后再使用wget的cookies加载命令进行相应的数据集下载，代码如下所示，这里下载了如图 5‑5所示的两个数据集。

*!wget --keep-session-cookies --save-cookies=cookies.txt --post-data*

*'username=user&password=pwd&submit=Login'*

*https://www.cityscapes-dataset.com/login/*

*# user为CityScapes网站的用户名，pwd为密码*

*!wget --load-cookies cookies.txt --content-disposition*

*https://www.cityscapes-dataset.com/file-handling/?packageID=1*

*!wget --load-cookies cookies.txt --content-disposition*

*https://www.cityscapes-dataset.com/file-handling/?packageID=3*

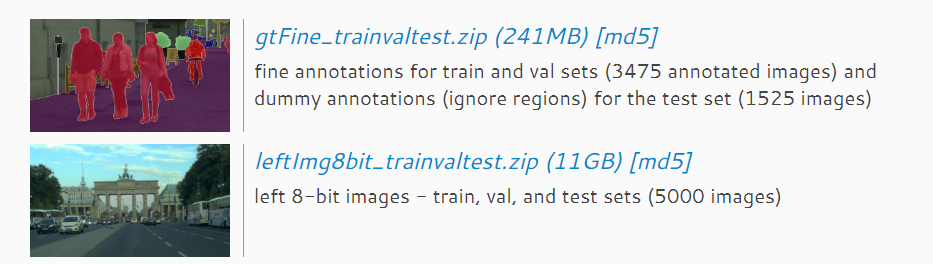


图 5‑5 CityScapes数据集

然后使用uzip命令对所下载的数据集进行解压得到leftImg8bit和gtFine两个文件夹，均有训练集、验证集和测试集三个目录。

**处理得到标签：**CityScapes数据集并不是将所有的标签都分类完全，需要在归一化和归类编码之前进行处理得到标签，使用的工具集为cityscapesScripts，如图 5‑6所示。

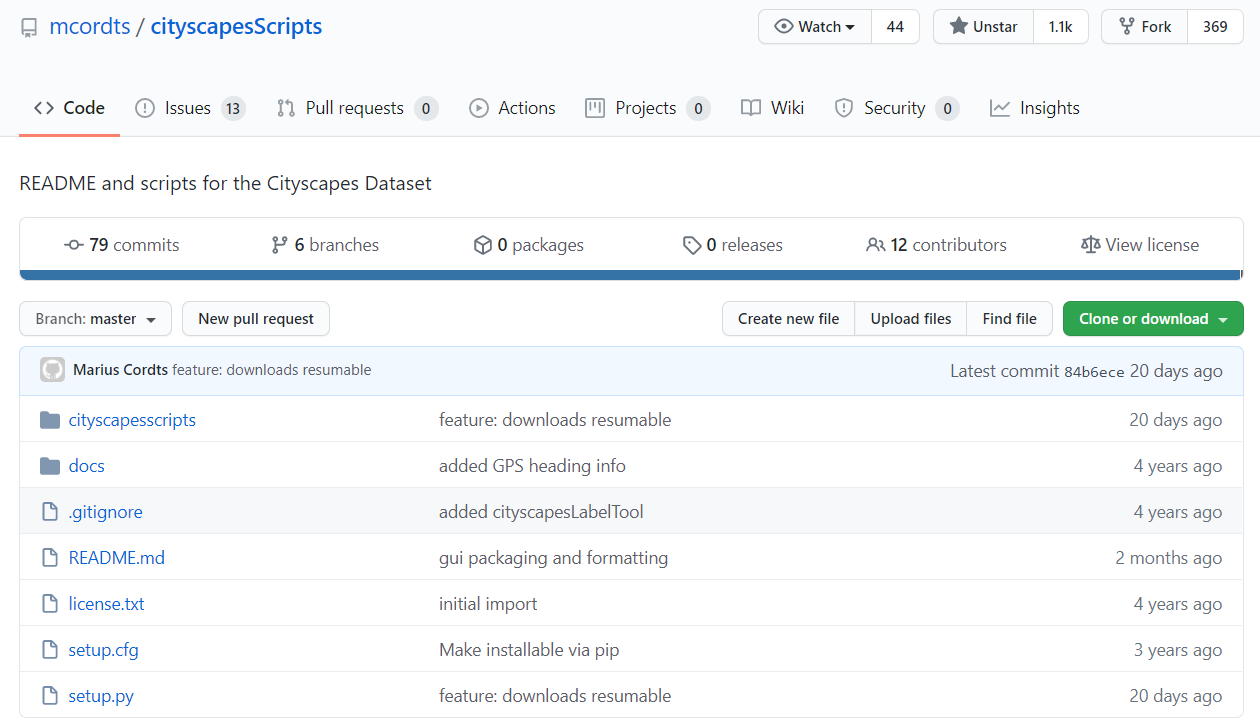


图 5‑6 cityscapesScripts

脚本目录结构如下所示。

* helpers：其他脚本包含的帮助文件
* viewer：查看图像和注释
* preparation：将地面真相注释转换为适合方法的格式
* evaluation：验证方法
* annotation：用于标记数据集的注释工具
* download：Cityscapes软件包的下载器

最重要的文件为：

* helpers/labels.py：定义所有语义类的ID并提供各种类属性之间映射的中央文件。
* helpers/labels\_cityPersons.py：定义所有CityPersons行人等级ID并提供各种等级属性之间映射的文件。

在notebook新建一个代码段，输入*!git clone https://github.com/mcordts/cityscapesScripts. git*下载此脚本。

打开cityscapesscripts/helpers/labels.py文件，如图 5‑7所示。可以看出这是定义物体类别的脚本，name指的是物体名称，id是物体序号，trainId是物体标签值，category是分类，hasInstances表示是否有实例，ignoreInEval表示是否在评估中忽略，color表示标签中的颜色。

对于不想识别的物体，将其trainId设置为255，ignoreInEval设置为True即可，本文共设置19类物体。

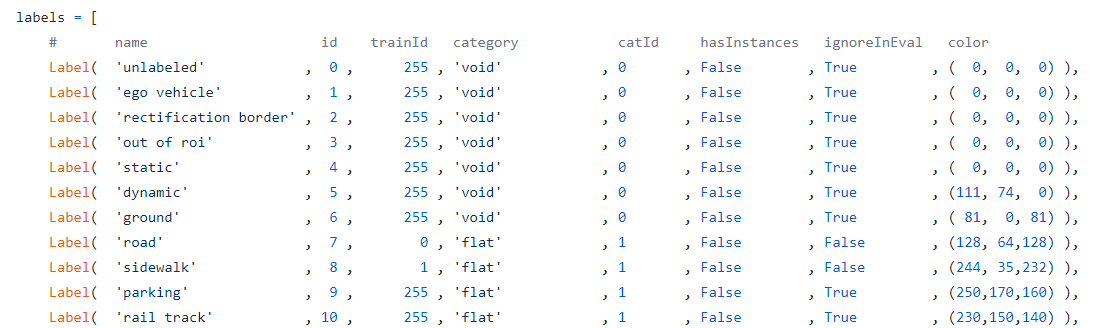


图 5‑7 定义物体类别

继续在notebook输入*!mv cityscapesScripts/cityscapesscripts ./*，用于将cityscapesscripts脚本移至CityPersons数据集的同级目录下。然后需要输入*!pip install cityscapesscripts*来安装脚本环境，不然环境找不到cityscapesscripts脚本包，会报model not found错误。最后需要运行preparation目录下的createTrainIdLabelImgs.py的脚本读取之前设置的label标签来生成训练的标签，在notebook输入*!python cityscapesscripts/preparation/createTrainIdLa belImgs.py*。运行结果如图 5‑8所示，可以看出最后生成了5000个annotation文件。

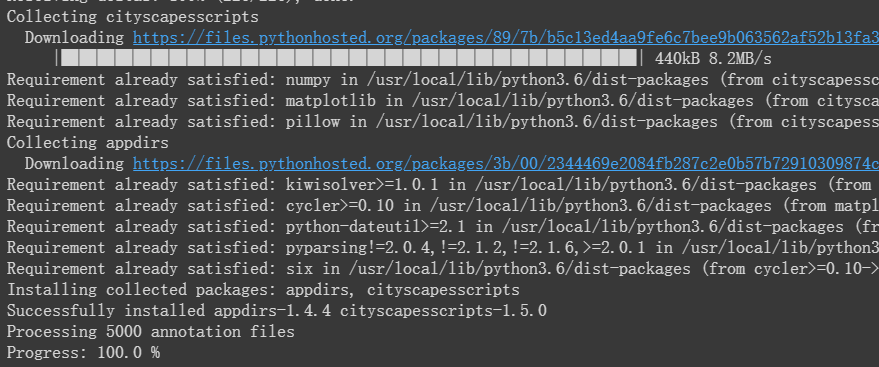


图 5‑8生成标签

## 建模与分析

### 构建Dataset

Pytorch用*torch.utils.data.Dataset*构建数据集，想要构建自己的数据集，则需继承Dataset类，并重写两个方法，\_\_len\_\_：定义整个数据集的长度，使用len(dataset)时会被调用。\_\_getitem\_\_：用于索引数据集中的数据，比如dataset[i]。接着创建Dataset子类，如下代码所示：

*class BasicDataset(Dataset):*

*def \_\_init\_\_(self, imgs\_dir, masks\_dir): # 初始化子类*

*self.imgs\_dir = imgs\_dir*

*self.masks\_dir = masks\_dir*

*self.ids = [splitext(file)[0] for file in listdir(imgs\_dir)]*

*def \_\_len\_\_(self): # 获取数据集长度*

*return len(self.ids)*

*def \_\_getitem\_\_(self, i): # 获取一个样本和标签*

*pass*

重写\_\_getitem\_\_方法，用来打开样本和标签，调用图片处理和返回一个样本和标签numpy对象，最后转为pytorch的tensor张量类型，过程如图 5‑9所示：

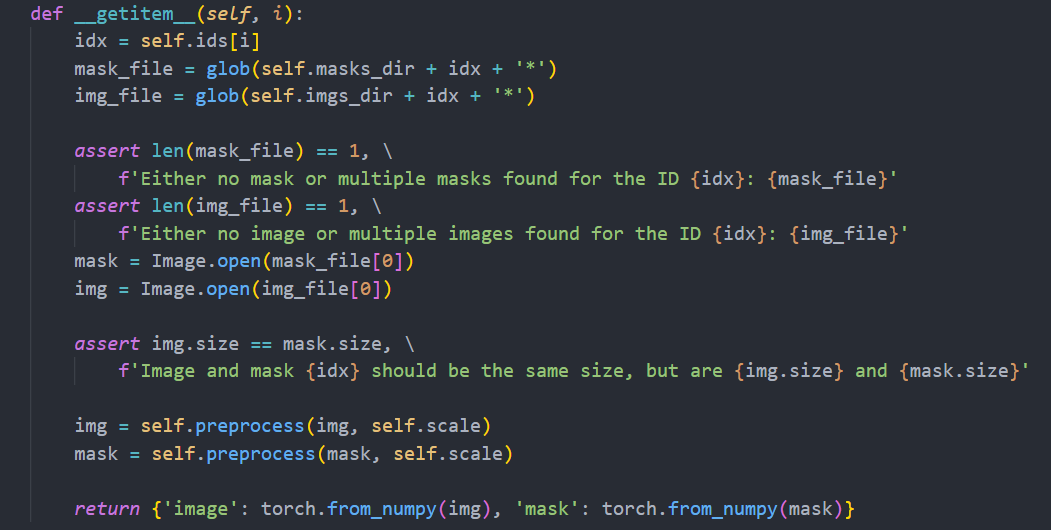


图 5‑9 getitem方法

### 网络分析

**双卷积层：**如图图 5‑10所示，在双卷积层，由两个卷积核大小为3，填充大小为1，步长为1的卷积层，两个数据归一化层和两个ReLU激活层组成。

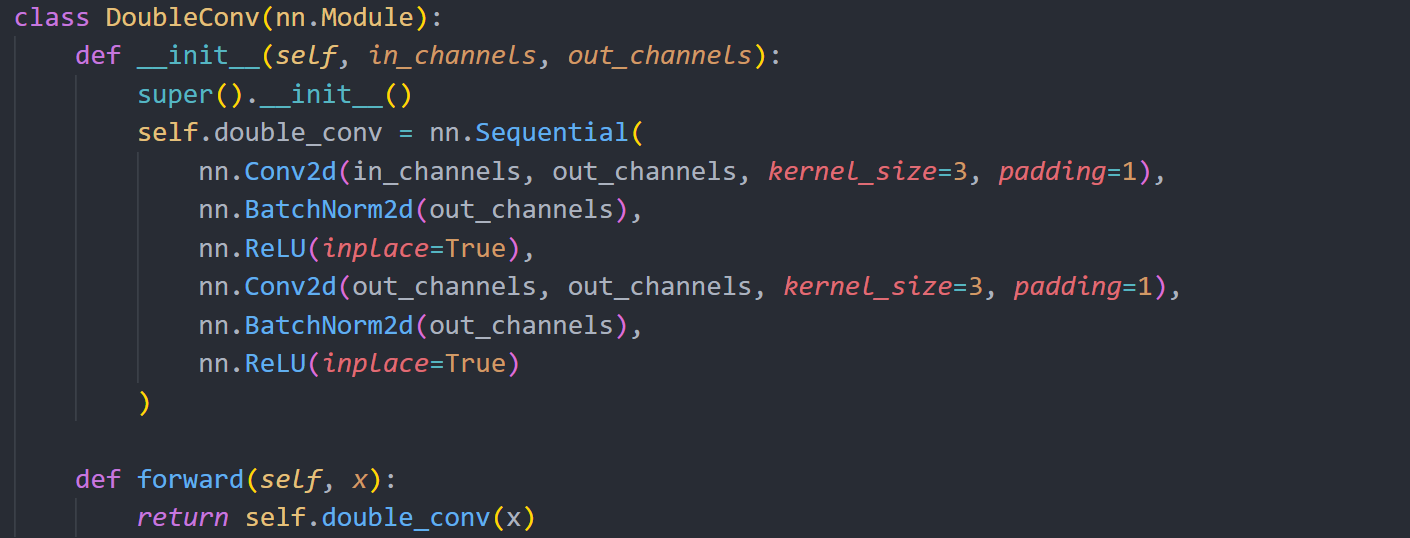


图 5‑10输入双卷积

在这里，输入图片通道数in\_channels的值为3，输出图片通道数out\_channels的值为64，输入图片的shape为[-1, 3, 360, 480]，含义依次是批量数（可调节，这里设置为-1，具体值由封装的数据集给出）、图片通道数、图片的高度和图片的宽度。

卷积层参数计算：根据公式，I为输入向量大小，K为卷积核大小，P为填充大小，S为步长大小，输出。

卷积层的输出参数个数为，即卷积核大小的平方与输入通道数的乘积，然后根据公式（权重的参数个数与输入的参数个数乘积，加上偏置的个数），来计算总的参数个数，这里权重与偏置的参数个数均为卷积核个数64，所以这个卷积层参数为：。

数据归一化参数计算：在上一层卷积中，输出shape=[-1, 64, 360, 480]，输入参数个数为，由于归一化的偏置个数为1，所以数据归一化中的参数个数为：，输出的shape为[-1, 64, 360, 480]。

**上采样：**如图 5‑11所示，一个上采样层由一个池化窗口大小为1的最大池化层和一个双卷积层组成。

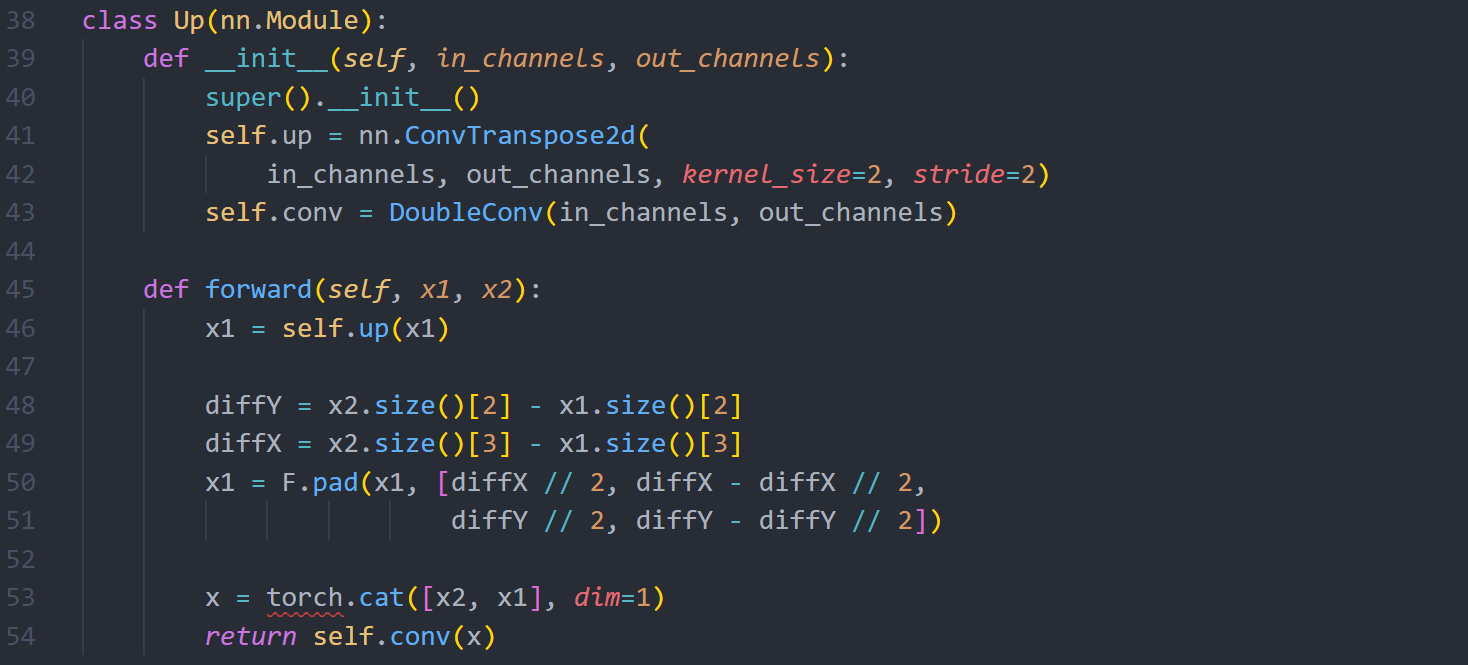


图 5‑11上采样

**下采样：**如图 5‑12所示，一个下采样由一个卷积核大小为2，步长为2的反卷积和一个双卷积层组成。

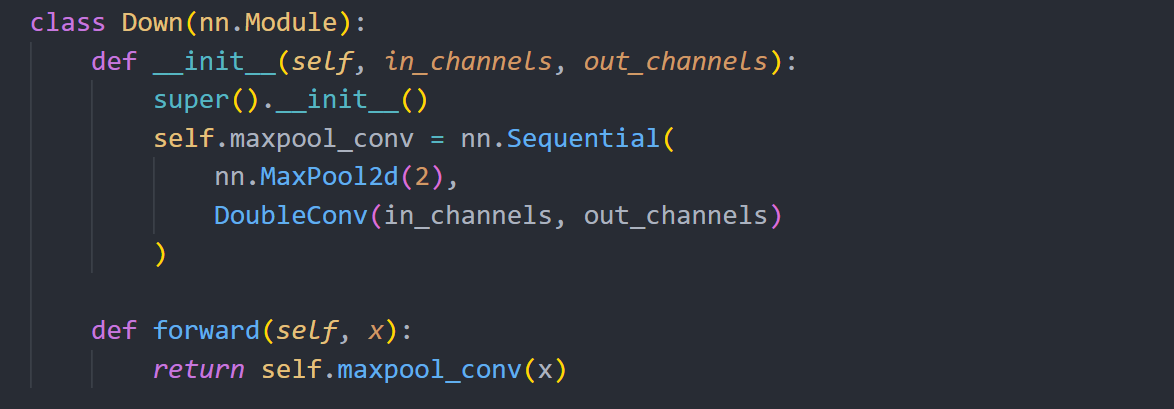


图 5‑12下采样

有所不同的是，在forward进行网络数据传输中进行了pad补充，目的是将输出的shape宽高大小与输出的shape宽高大小一致。

### 图片、视频预测方法

**图片预测：**相关代码如图 5‑13所示，首先使用BasicDataset中的图片处理函数处理为网络可用的图片。然后使用unsqueeze(0)，在0维插入一个维度，此时tensor从torch.Size([3, 360, 480])变为Size([1, 3, 360, 480])，即batch批量数为1。

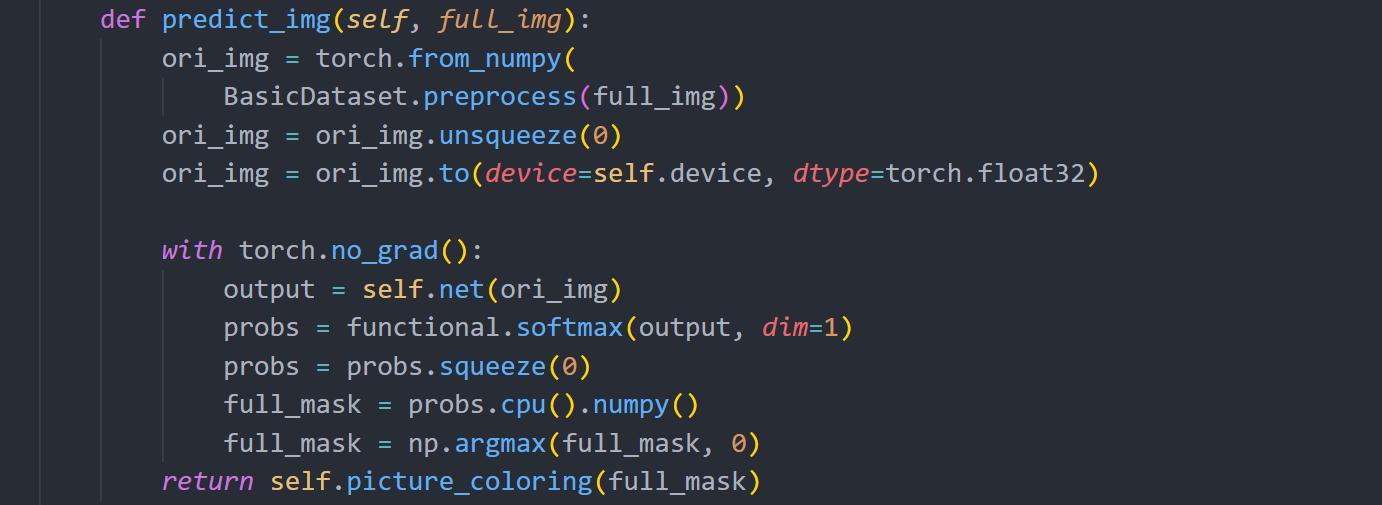


图 5‑13图片预测

之后将图片tensor转为torch.float32类型，并加载到cpu或者gpu设备。然后使用*with torch.no\_grad()*来取消计算导数，因为是预测而不是进行训练。然后将图片tensor张量喂入网络，得到预测输出output，对第1维度进行softmax归一化后，使用squeeze来删除批量数0维。再将变量或者数据移到CPU，并转换为numpy array，使用np.argmax()来求出0维最大值所在的编号，即预测的one-hot值，最后使用自行编写的涂色函数对图片进行上色，就会得到最终结果。

**视频预测：**视频预测相关代码如图 5‑14所示，首先使用opencv读取所要识别的视频，每10帧读取一个图像进行预测，然后再创建一个视频，将预测的图像从numpy对象转为opencv格式，再转为Pillow对象，最后写入视频，进行释放。

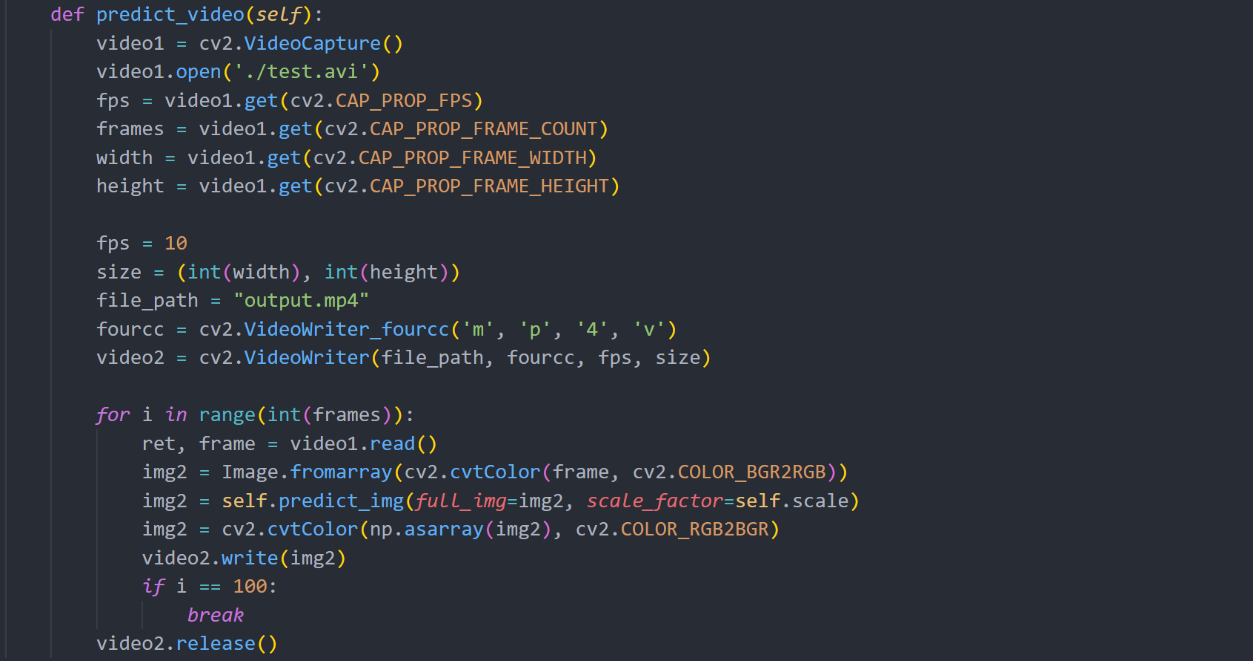


图 5‑14视频预测

## 实验结果

### 训练参数

**样本参数：**如图 5‑15所示，train\_images\_path设置训练样本文件夹，train\_segs\_path设置标签文件夹。n\_classes为识别分割的物体数，可以根据数据集进行调整。

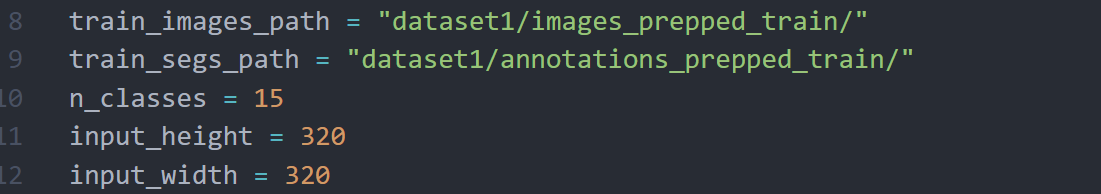


图 5‑15样本参数

**网络参数：**如图 5‑16所示，train\_batch\_size为训练批次数，可以根据GPU和内存进行动态调整，以获取最大利用率，epochs为训练的轮数，本文设置为500。网络优化器使用Adam，学习率为1e-4，损失函数使用categorical\_crossentropy，准确率函数使用acc。

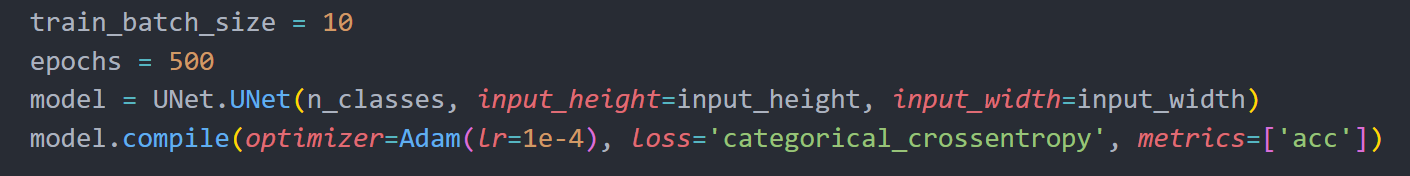


图 5‑16网络参数

**模型保存：**如图 5‑17所示，filepath设置模型保存路径和文件，monitor为被监测的数据，这里为准确率acc，mode选择auto模式，方向会自动从被监测的数据的名字中判断出来，save\_weights\_only为True，表示只有模型的权重会被保存。

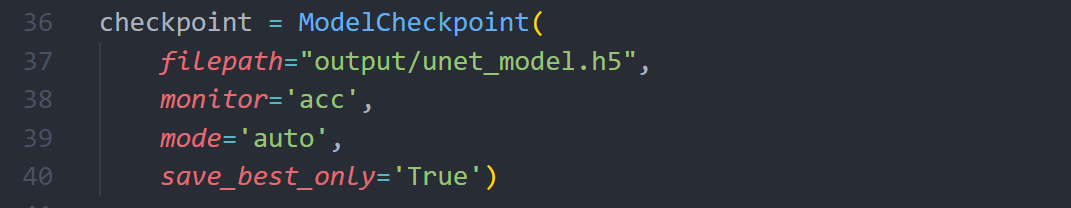


图 5‑17模型保存参数

### 训练Loss与Acc

训练500个epoch，结束后将模型保存至项目目录，每次epoch也增加计算准确率和损失，并保存至tensorboard。将评价指标acc和loss通过tensorboard可视化，相应的曲线如图 5‑18-图 5‑19所示。通过曲线可以看出训练了大概500轮，损失值大概已经降低到0.05，准确率大约在95%。

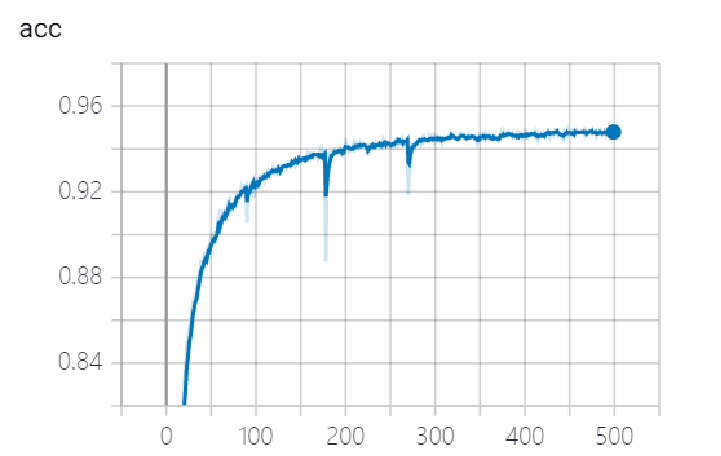


图 5‑18 acc

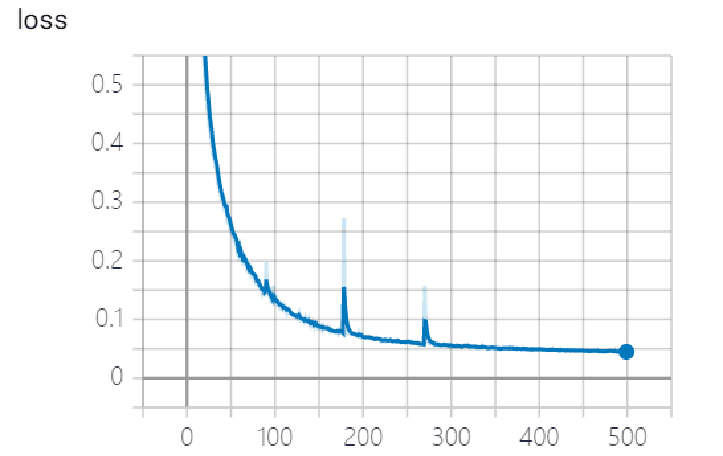


图 5‑19 loss

### 图片预测结果

图片预测结果如图 5‑20-图 5‑21所示，可以看出图像物体大都被分割了出来。



图 5‑20原图

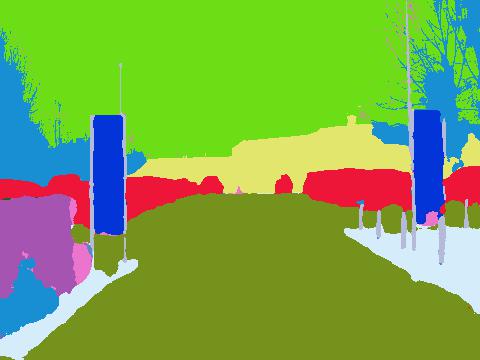


图 5‑21预测图

### 视频预测结果

由于视频无法展示，所以截取了一张图来展示结果，如图 5‑22所示。



图 5‑22视频处理结果

# 客户端可视化

为了便于展示实验结果，实验采用了网站和小程序进行结果的可视化展示，可以进行图片的上传进行预测，并且通过服务的部署达到了公有网络的访问

## FLask服务

Flask是一个使用Python编写的轻量级Web应用框架，基于Werkzeug WSGI工具箱和Jinja2 模板引擎，使用BSD授权。本项目使用工厂模式进行开发，相应的目录与文件结构如下所示。



### 基本配置

**根目录配置：**打开config.py文件，输入*import os*和*basedir=os.path.abspath(os.path. dirname(\_\_file\_\_))*，目的是将项目的根目录绑定在basedir变量，便于其他文件对项目中的其他目录进行访问。

**创建app应用：**在apps内新建一个Python Package，名称为predict，并增加一个views.py文件以便后续增加代码，修改\_\_init\_\_.py文件，加入蓝图Blueprint实例的创建，并导入新建的views.py，如图 6‑1所示。

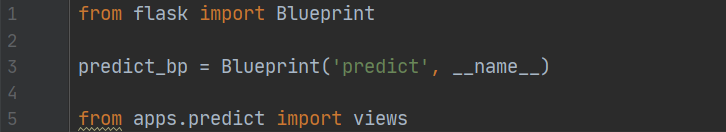


图 6‑1创建app应用

**创建app方法：**打开apps/\_\_init\_\_.py，加入如图 6‑2所示的代码，前3行导入相关的支持库，第7-8行template指明项目存放html模版的目录，static指明静态文件存放的目录。接下来一行是创建Flask应用实例，然后加载配置对象。第12-13行是导入predict的蓝图，然后注册蓝图，默认路由为“/”。

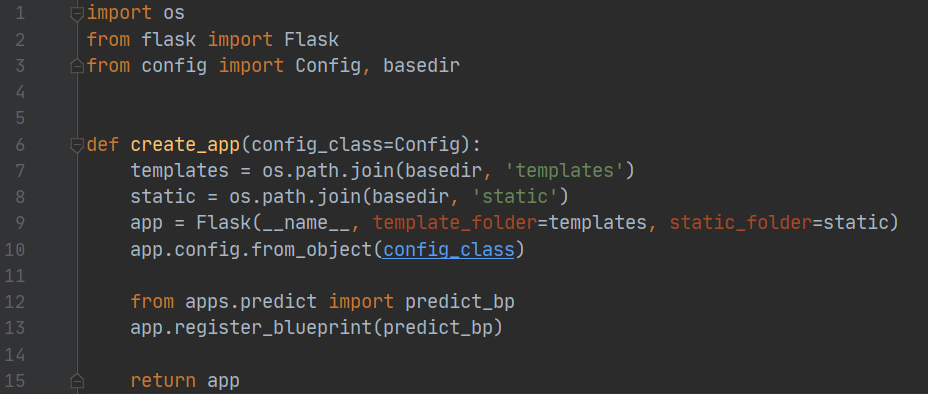


图 6‑2创建app方法

**拷贝模型文件：**将语义分割的模型pytorch\_unet.py和数据集处理脚本dataset.py文件拷贝至项目的utils目录下。从谷歌colab下载模型训练的保存结果MODEL.pth，拷贝至项目的checkpoints目录下。

### 网页

**前端框架：**由于项目功能逻辑简单，并且需要快速上线，所以使用layui前端UI框架，如图 6‑3所示。通过*git clone https://github.com/sentsin/layui.git*到本地后，复制dist目录到flask项目，并修改dist目录为layui目录。

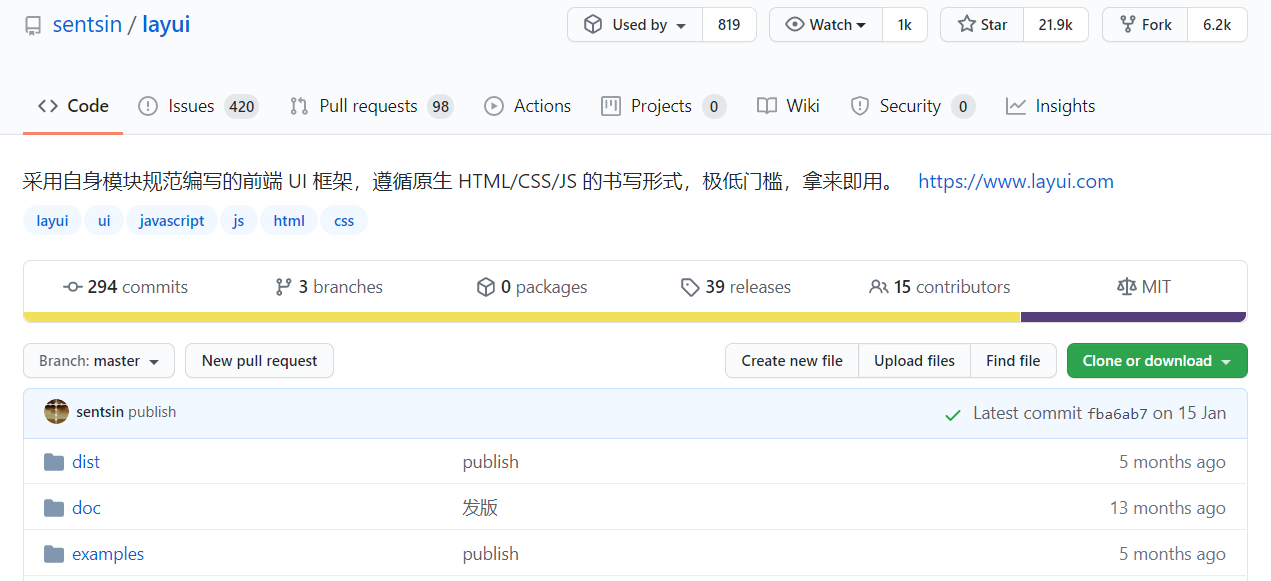


图 6‑3 layui框架

在templates目录下新建predict子目录，新建一个index.html文件，在head标签内导入layui的css样式，使用flask模版的url\_for()方法，代码为：*<link rel="stylesheet" href= "{{ url\_for('static', filename='layui/css/layui.css') }}">*。在body标签后面增加一个script标签导入layui的js脚本，代码为：*<script src="{{ url\_for('static', filename='layui/layui.js') }}"> </script>*。

同时，修改index.html文件，编写出一个后台管理样式的网页，如图 6‑4所示。



图 6‑4网页布局

### 后端服务

**构建图片上传工具集：**在项目utils目录下增加img\_deal.py文件，首先增加一个支持上传的图片格式集合['png', 'jpg', 'JPG', 'PNG', 'gif', 'GIF']，得到上传的图片后进行文件后缀的检验。之后利用datetime和random创建一个重命名的方法，格式为时间和一个两位的随机数，代码如图 6‑5所示。

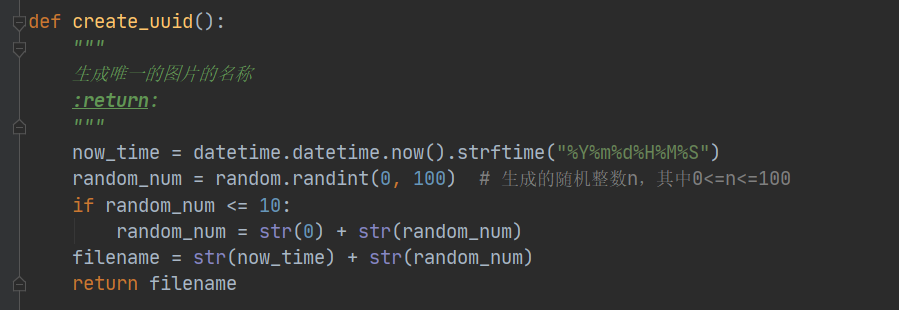


图 6‑5图片名称处理

**注册图片回显路由：**修改apps/predict/views.py文件，增加如下图 6‑6所示的代码，绑定了两个url，首先根据访问的url判断是否访问static目录下的文件，若是则继续判断是否访问子目录filepath，然后修改file\_dir变量来指向文件路径，最后通过send\_from\_directory()来返回指定目录下的文件名为filename的文件。

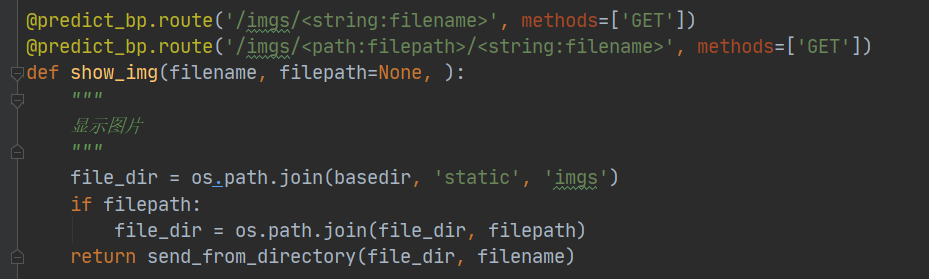


图 6‑6图片回显

**注册网页回显路由：**通过*@predict\_bp.route('/', methods=['POST', 'GET'])*来注册“/”路由，并且设置支持请求方法为post和get。

若是get请求则直接通过*render\_template('predict/index.html')*返回网页。若是post请求则是网页有图片进行了上传操作，如图 6‑7所示。首先通过*request.files.get('in\_file')*来获取上传的文件对象，然后判断文件格式是否合法，若合法则根据时间戳和随机数创建一个文件名，随后将图片保存在static/imgs目录下。

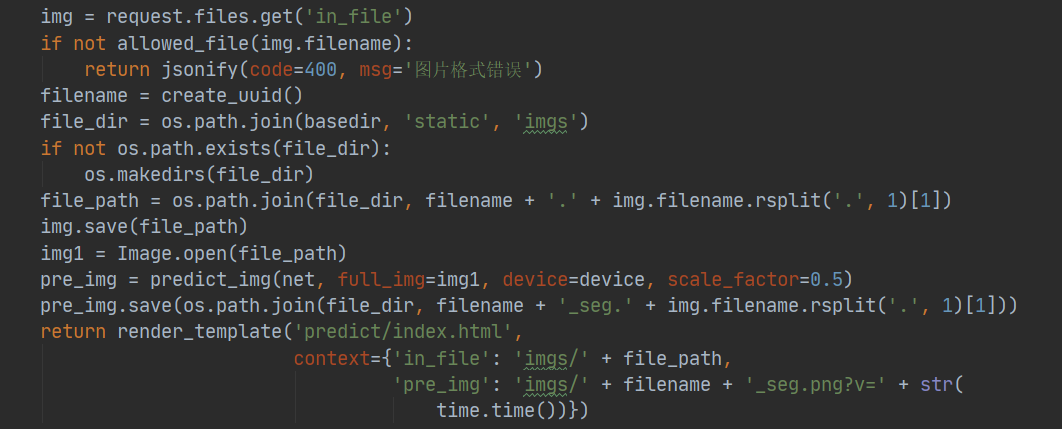


图 6‑7图片语义分割预测

之后使用Pillow读取此文件，调用predict\_img()方法来对图片进行语义分割，将分割结果保存在相同目录下，通过flask的模版进行回显。

## 小程序

### 小程序准备

**注册小程序：**打开网站https://mp.weixin.qq.com/wxopen/waregister?action=step1，如图 6‑8所示，按照要求输入邮箱和密码进行注册，主体类型选择个人。



图 6‑8小程序注册页面

注册完成会要求邮箱进行验证，点击链接会跳转到小程序信息登记页面，按照要求填写即可，本文使用的小程序信息如图 6‑9所示。



图 6‑9小程序信息填写页面

**安装微信开发者工具：**微信开发者工具是腾讯推出的一款集成公众号和小程序调试的工具。使用公众号网页调试，开发者可以调试微信网页授权和微信JS-SDK，使用小程序调试，开发者可以完成小程序的API和页面的开发调试、代码查看和编辑、小程序预览和发布等功能。

小程序注册完成后点击右上角的文档选项卡，在新打开的网页顶部导航栏点击工具，然后点击左侧下载超链，可以看到开发工具下载地址及版本，如图 6‑10所示，本文选择了Windows 64位的稳定版。



图 6‑10微信开发工具下载



图 6‑11微信开发工具登录

下载完成后，按照要求安装，安装完成打开开发工具，如图 6‑11所示，微信扫码登录。登录完成后，界面如图 6‑12所示，下面可以进行小程序新建过程。

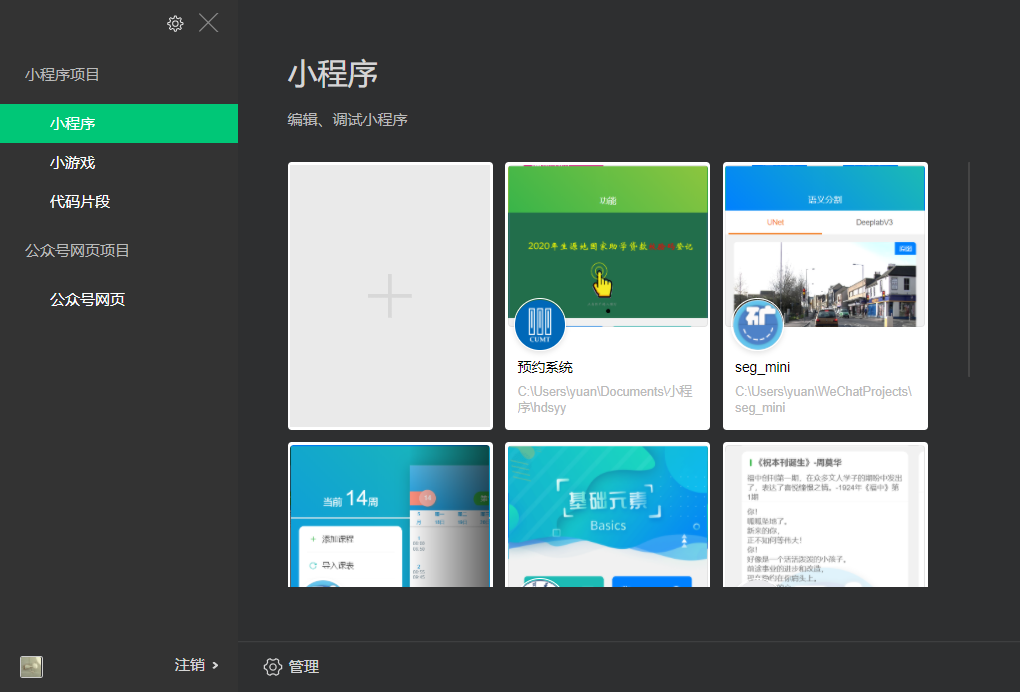


图 6‑12微信开发工具登录完成

### 新建小程序

在小程序开发工具登录完成后，点击“+”号，填写小程序的相关信息，如图 6‑13所示。



图 6‑13新建小程序页面

项目名称可以自行填写，AppID可以在小程序后台“设置->基本设置->账号信息”中找到，拷贝填入即可，开发模式选择小程序，后端服务选择不使用云服务，然后新建，等待几秒之后可以看到小程序的开发界面，如图 6‑14所示。最上面是工具栏，下面从左到右依次是：模拟器、目录结构、编辑器和终端输出。

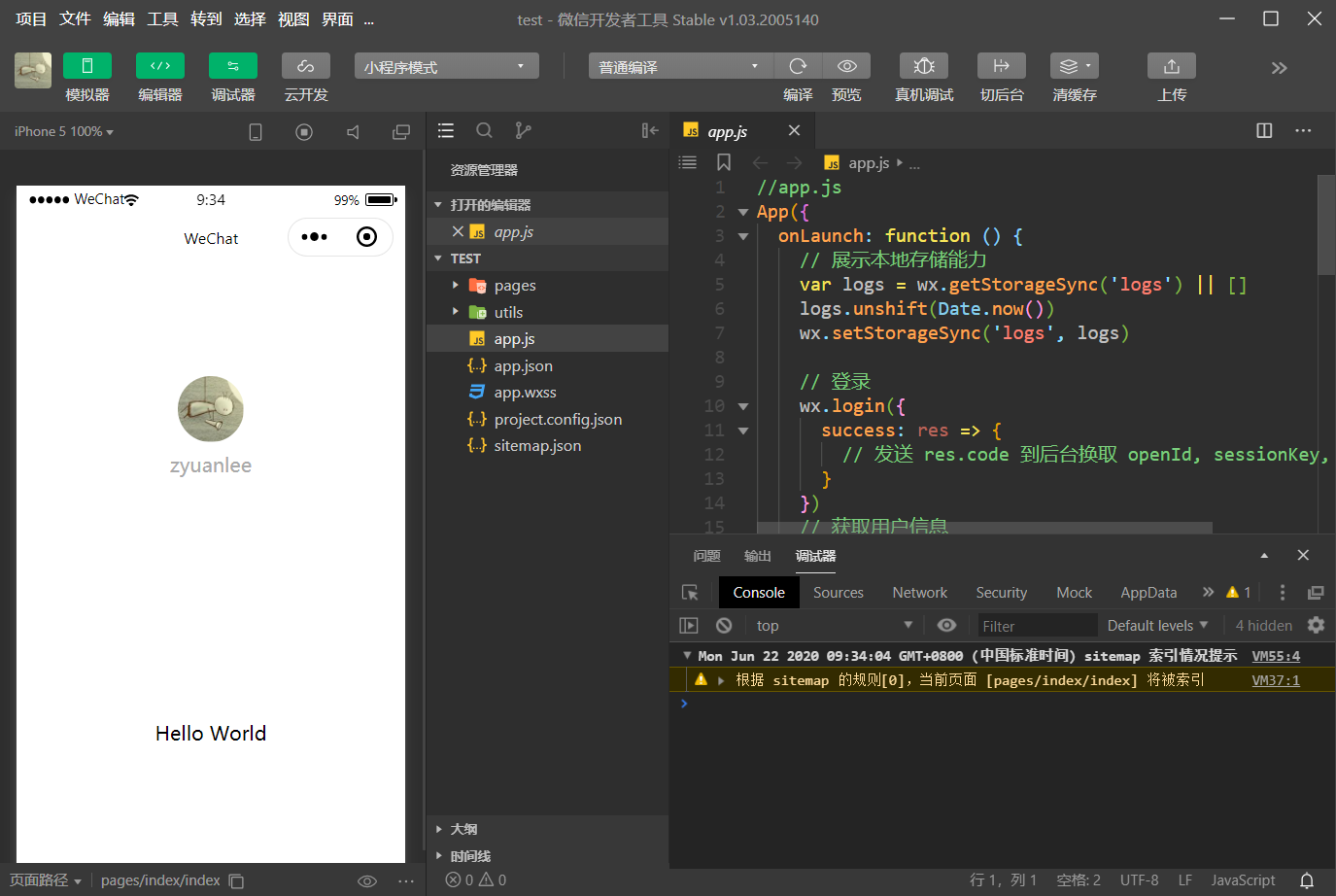


图 6‑14小程序开发页面

### 小程序界面设计

根据需求设计小程序界面，如图 6‑15-图 6‑17所示。



图 6‑15小程序界面1



图 6‑16小程序界面2



图 6‑17小程序界面3

### 主要实现代码

**登录：**登录主要代码如图 6‑18所示，首先将form表单提交的用户名和密码获取封转在post\_data对象中，再通过wx.request()方法请求api，util.rootUrl代表主域名，method指明请求方式为post，若请求成功，则将账号密码和登录成功的token缓存下来以供后续操作使用，最后跳转到首页。



图 6‑18登录主要代码

**上传图片进行预测：**预测主要代码如图 6‑19所示，使用的是wx.uploadFile()方法，filePath指明上传的图片路径，name指明图片所在的参数名字，header中的Authorization是用来登录状态的验证，formData的type来指明此次请求是进行预测，最后使用JSON.parse()将返回的数据解析成json格式，然后回显图片。



图 6‑19预测主要代码

## 服务部署

### 配置远程ssh登录

登录购买的腾讯云服务器，进入实例后台，为学生套餐：1核处理器，2GB运行内存，2Mbps带宽。在腾讯云后台登录服务器终端，选择标准登录方式，输入用户名和密码后登录成功结果如图 6‑20所示。

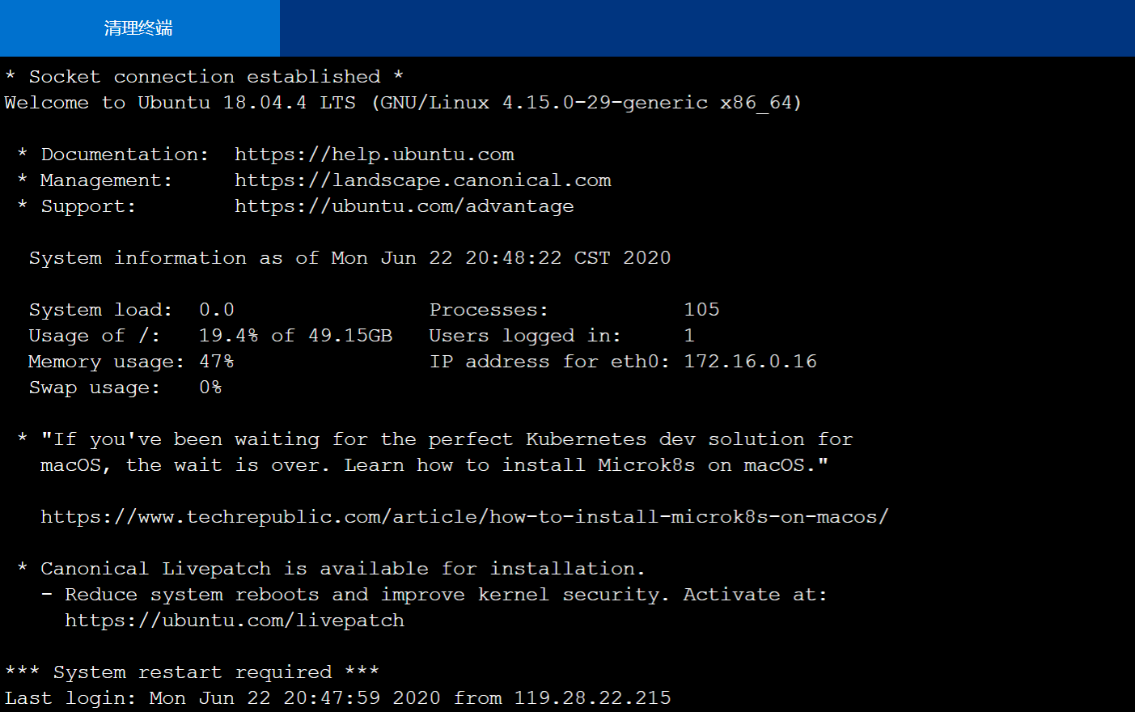


图 6‑20 ssh内网登录

在终端输入*sudo passwd root*，配置服务器的root密码，配置成功后，返回*passwd: password updated successfully*信息。然后输入*sudo vi /etc/ssh/sshd\_config*，去除PermitRoot-Login前面的“#”注释，修改其值为*yes*，然后按下Esc键，输入*:wq*保存，最后再输入*sudo service ssh restart*重启ssh服务。

下面开启安全组，点击安全组，结果如图 6‑21所示。然后点击修改规则，添加22端口开放，如图 6‑22所示。



图 6‑21安全组1



图 6‑22安全组2

然后打开Xshell，配置服务器的相关信息，登录成功后如图 6‑23所示。

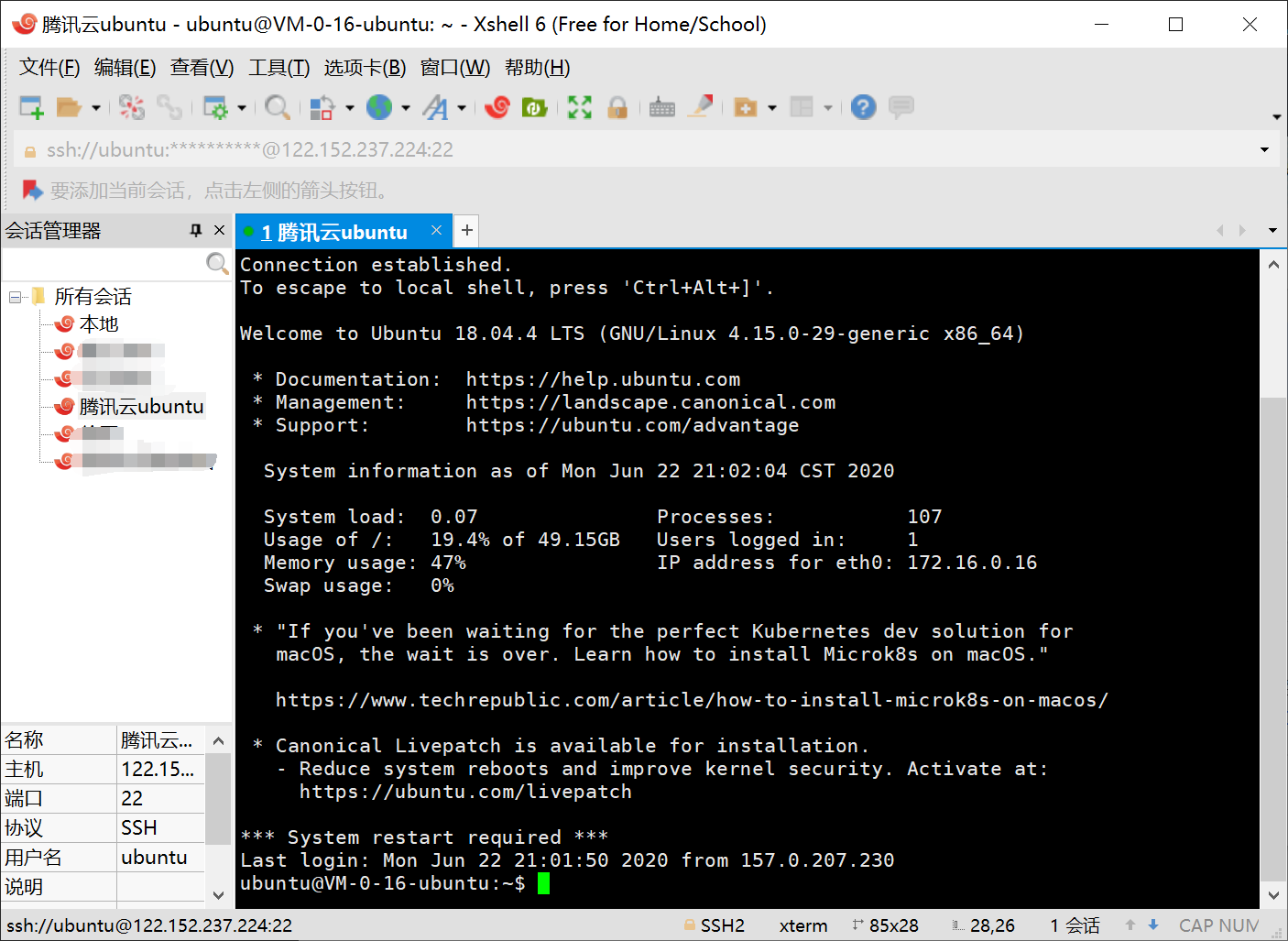


图 6‑23 Xshell远程登陆

### 添加域名云解析

在云产品中搜索“DNS 解析 DNSPod”，打开后如图 6‑24所示。



图 6‑24 DNS解析页面

依次点击解析->添加解析，添加test.cumtlee.cn到服务器的公网IP地址，如图 6‑25所示，这样就可以将后端服务转发至test.cumtlee.cn网站。



图 6‑25添加解析

### 基础环境安装

打开Xftp，进入服务器文件目录，将项目后端代码上传至服务器，如图 6‑26所示。

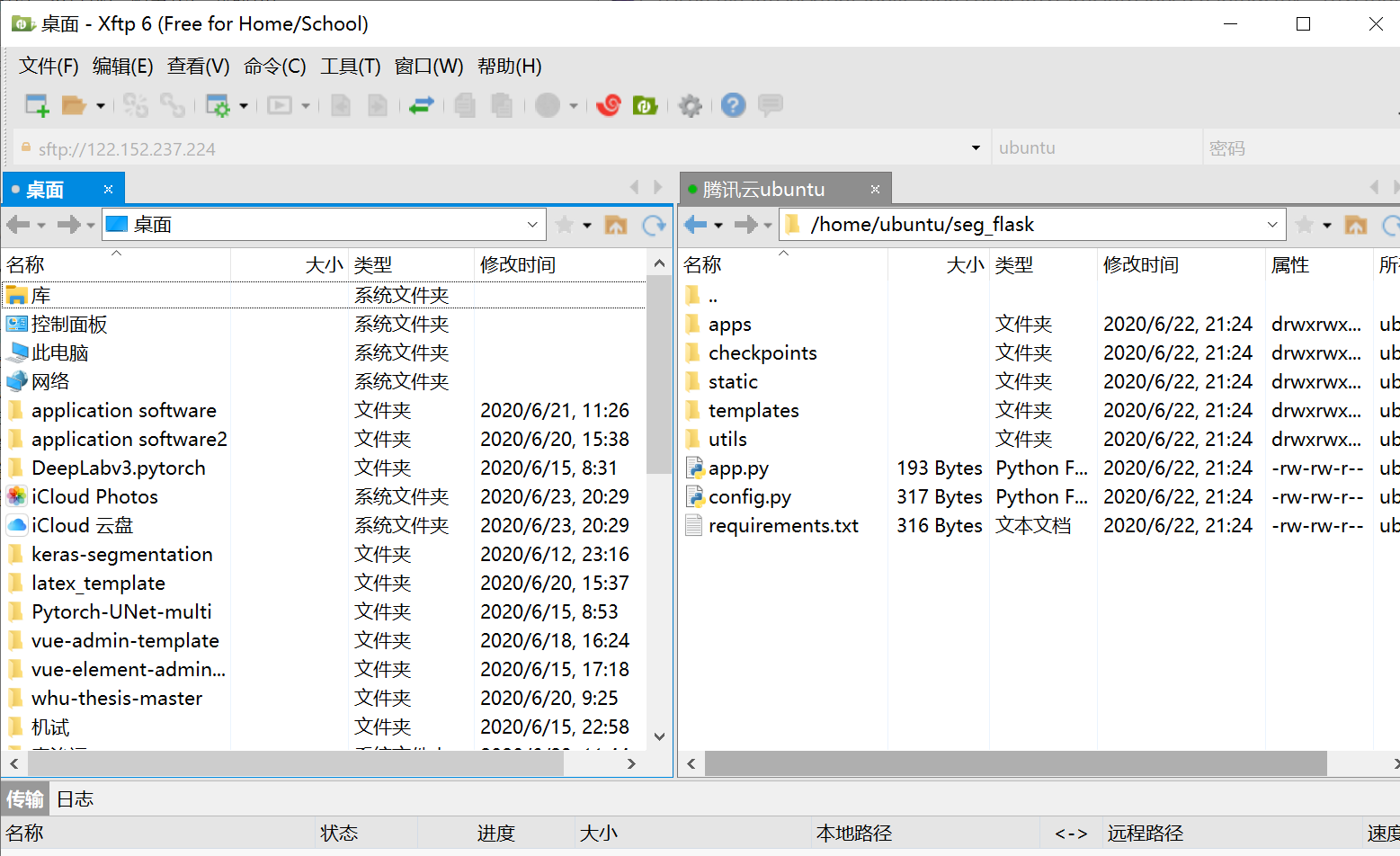


图 6‑26项目代码上传

然后输入*pip3 install virtualenv*安装建立虚拟环境的模块，安装完virtualenv后，进入项目目录，通过*virtualenv venv*建立一个虚拟目录venv，然后*source venv/bin/activate*进入虚拟环境，然后终端命令行前面多了(venv)的标识，表示已经激活虚拟环境。然后使用*pip install -r requirements.txt*来安装项目后端所需要的模块。

### uwsgi启动服务

继续在终端输入*pip install uwsgi*来安装uwsgi模块，然后在项目的根目录新建一个uwsgi.ini文件，输入下述代码，如图 6‑27所示。



图 6‑27 uwsgi配置

然后启动后端服务，在终端输入*uwsgi --ini uwsgi.ini*，然后项目目录下面就会生成log和pid文件，分别存放日志和服务进程id，如图 6‑28所示。

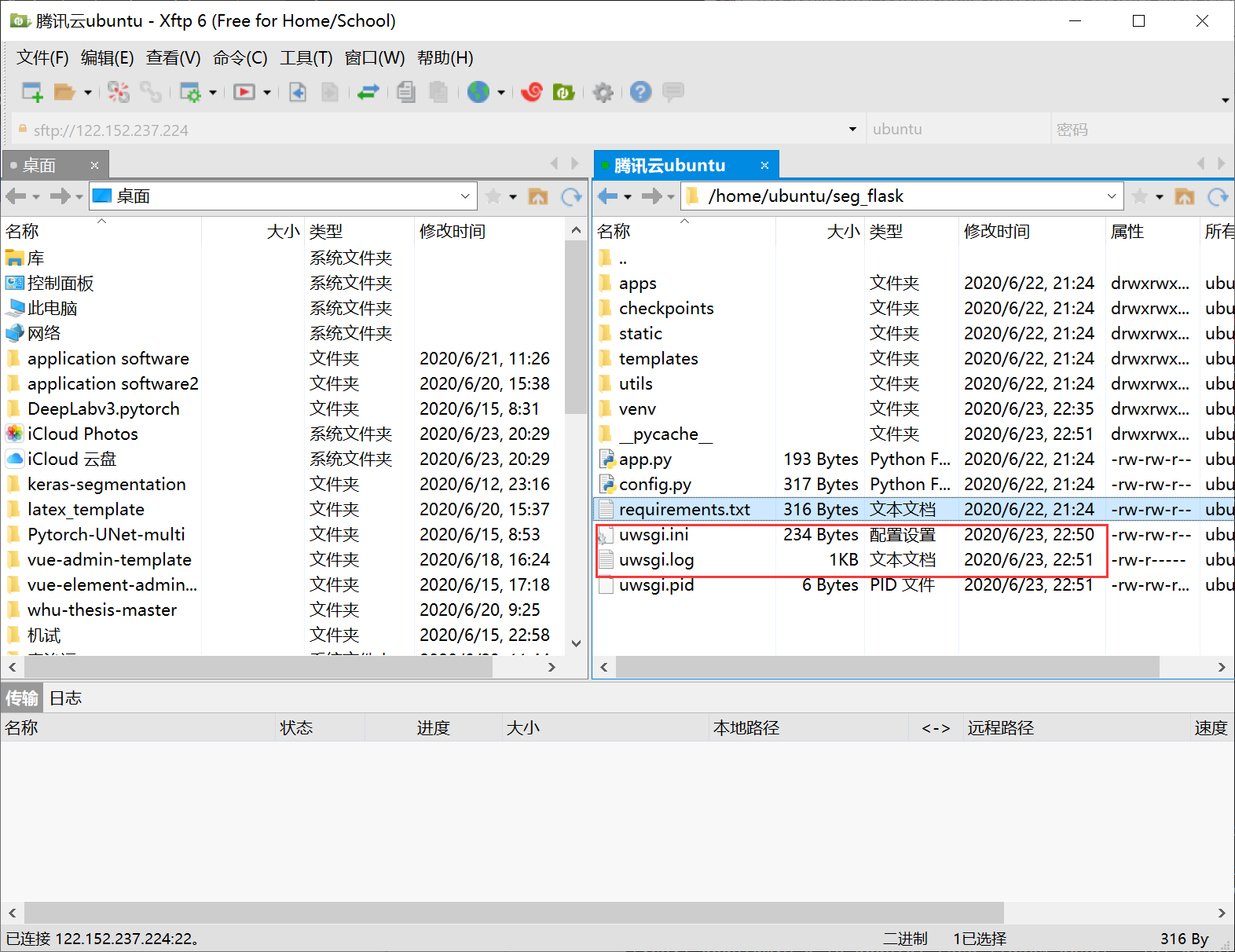


图 6‑28 uwsgi启动

### nginx转发

为了让项目可以外网访问，使用nginx来转发服务。在项目目录下新建一个nginx.conf，文件，输入如图 6‑29所示的代码。

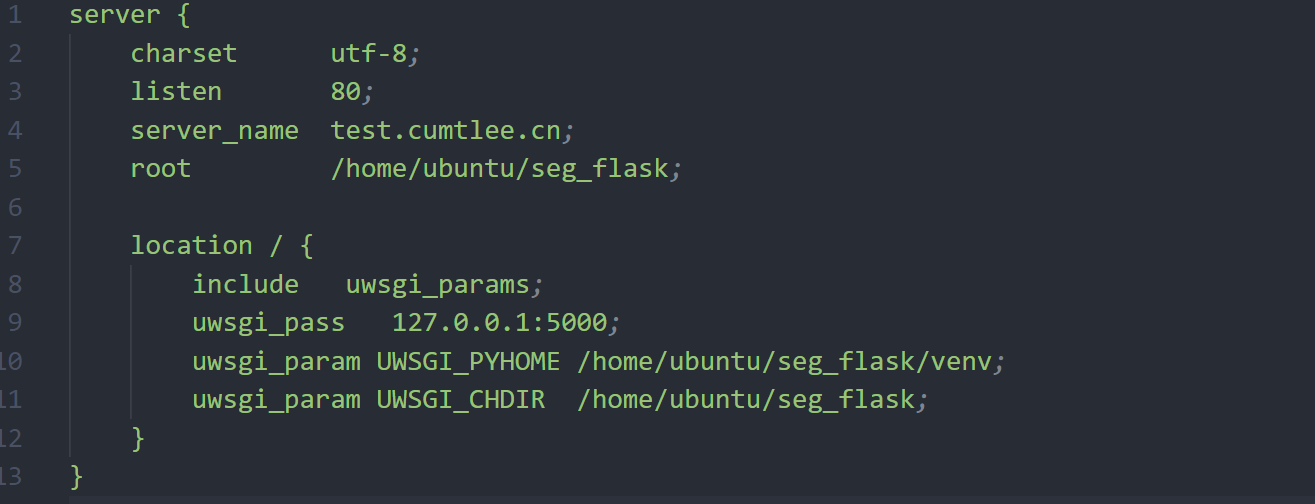


图 6‑29 nginx配置

然后输入*sudo vim /etc/nginx/nginx.conf*，在http作用域内增加*include /home/ubunt-u/seg\_flask/nginx.conf;*，之后按下Esc按键，输入:wq保存配置，最后输入*sudo service nginx restart*重启nginx服务。

浏览器输入test.cumtlee.cn，出现如图 6‑30所示的界面，说明服务启动成功。

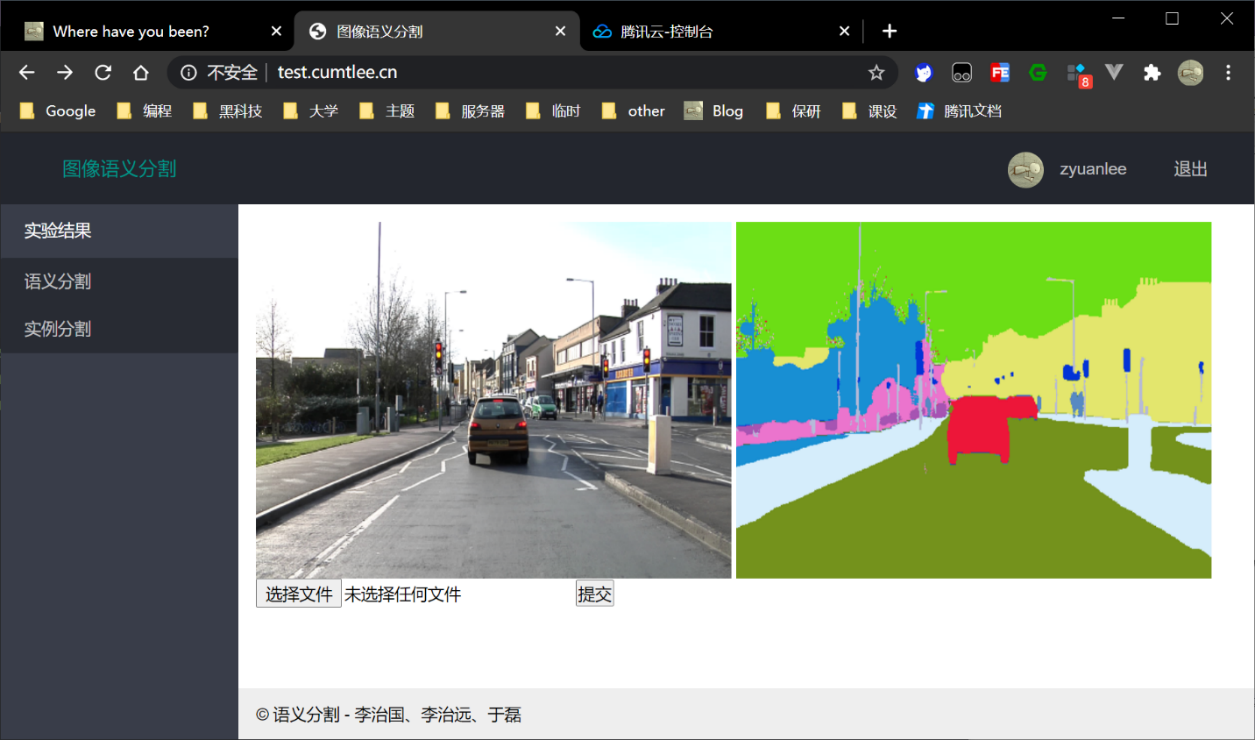


图 6‑30服务启动成功

# 项目评估

## 项目优缺点

### 项目优点

1. U-Net支持少量的数据训练模型；
2. U-Net通过对每个像素点进行分类，获得更高的分割准确率；
3. U-Net用训练好的模型分割图像，速度快；
4. 使用小程序和网页进行项目的展示，无需担心设备的不统一性；
5. 使用服务器公网部署项目后台，所有人都可以体验项目功能；
6. 后端服务增加API接口，这样对于扩展桌面端或者其他前后端分离项目很有好处。

### 项目缺点

1. 由于语义分割需要较强的GPU资源，所以需要较大的开销；
2. 服务器带宽和GPU限制，无法进行视频预测分析；
3. U-Net对物体边缘分割较差，可以使用其他网络模型，如DeepLabV3+。

## 改进方法

### 迁移学习训练模型

迁移学习（Transfer Learning）是机器学习的一个研究方向，主要研究如何将任务A上面学习到的知识迁移到任务B上，以提高在任务B上的泛化性能。

例如任务A为猫狗分类问题，需要训练一个分类器能够较好的分辨猫和狗的样本图片，任务B为牛羊分类问题。可以发现，任务A和任务B存在大量的共享知识，比如这些动物都可以从毛发、体型、形态、发色等方面进行辨别。因此在任务A训练获得的分类器已经掌握了这部份知识，在训练任务B的分类器时，可以不从零开始训练，而是在任务A上获得的知识的基础上面进行训练或微调（Fine-tuning）。通过迁移任务A上学习的知识，在任务B上训练分类器可以使用更少的样本和更少的训练代价，并且获得不错的泛化能力。如果使用迁移学习的方法，在街景图样本较复杂或者很多的时候，可以使用此方法快速训练，并且也可以得到很好的效果。

### DeepLabV3+模型

编码器在DeepLabV3的基础上，主要是采用了ASPP结构，采用四个并行的不同atrous rates的卷积对feature map进行处理，能够有效的捕捉多尺度的信息[8]。解码器部分先从低层级选一个feature，将低层级的feature用的卷积进行通道压缩（原来为256通道，或者512通道），目的在于减少低层级的比重。编码器得到的feature具有更丰富的信息，所以编码器的feature应该有更高的比重，有利于训练。再将编码器的输出上采样，使其分辨率与低层级feature一致，如果采用resnet conv2输出的feature，则这里要上采样。将两种feature连接后，再进行一次的卷积（细化作用），然后再次上采样就得到了像素级的预测。

# 总结

随着项目的完成，应用软件开发实践的课程也结束了，在项目的研究过程中，学习到了很多新的知识。由于本项目是基于机器学习和深度学习的相关研究，所以掌握其基本原理知识很有必要。所以在这个过程中，我对模糊系统、神经网络与进化计算有了大致的理解，对其中所用的算法也有很好的掌握。

作为一个大学生，如何查找文献和阅读文献的能力是应该必备的，在对项目相关文献的检索过程中，又锻炼了我的这个能力。首先我先检索与项目有关的技术及其对比，选取了U-Net和DeepLabV3+这两个网络作为主要研究对象。然后又通过检索相关技术，很快就查找到了网络模型作者的相关论文，接下来就是逐字逐句理解的过程。

深度学习有着很多的框架，比如TensorFlow，Pytorch等，掌握其中一个，能在项目的开发过程中大大的减少时间的花费，这样也能写出规范，可读性较强的代码。在确认研究主题和使用工具，以及通过阅读文献总结的方法后，我使用Pytorch框架完成了U-Net的网络编写。对于数据集的处理也掌握很多，比如图片的灰度化、二值化、缩放与像素归一化等都有着实际的操作，这对于我之后进行计算机视觉相关项目的研究有着很好的导向作用。在图片预测的过程中，深入学习率Pillow读取图片的一些操作，比如图片分割结果的涂色问题，也学习了opencv读取视频和写视频的过程，这些对于很好的展示实验结果很有帮助。

由于项目开发之初有设计客户端展示的想法，所以学习了前端网页的一些知识，对于前端的发展状况也有了一些认识。可以使用Vue.js来开发一个健壮的网页，但是需要node.js基础，为了节省时间以及较为快速的开发项目，所以选择了layui这个前端框架进行项目的开发。仅仅有前端的基础也无法将项目很好的展示出来，所以就需要一个后端来支持，在这个过程中，我学习了基于Python的Web框架——Flask，它简洁、快速，通过对其的学习，加深了对路由和软件开发模式中工厂模式等的理解。随着项目的深入，有想到现在移动设备的普及，开发一个小程序对于用户的体验有较大的提升。通过学习其官方文档，并且结合前端知识，体会到了小程序用完即走的理念，也很好的完成了小程序展示项目的功能。

linux应该是每个计算机专业学生掌握的一个系统，通过Flask后端项目的部署，学习了腾讯云服务器的使用方法。对于linux中的防火墙、端口、用户管理、进程通信和文件权限等，结合所掌握的一些操作系统知识和实际的操作过程，加深了对系统的整体理解。

当然，也不能仅仅局限于学习完这门课程与参与软件杯比赛，我也思考了如何运用所学的知识到实际的应用之中。对于这次的项目研究，可以用于自动驾驶，也可以运用在医疗图像CT等领域。前端与后端的经验可以运用在设计一个软件架构之中，比如博客系统。

总的来说，这次的课程学习对于我的综合能力和创新水平有者很大的提高，我也将会在学习的路上一直走下去。

# 参考文献

[1] LECUN Y, BENGIO Y, HINTON G. Deep learning[J]. nature, 2015, 521(7553): 436–444

[2] GANG L. Recognition of multi-fontstyle characters based on convolutional neural network[J]. Journal of Zhejiang Normal University (Natural Sciences), 2011, 4

[3] LEKE C A, MARWALA T. Introduction to Deep Learning[M]. Deep Learning and Missing Data in Engineering Systems.[S.l.]: Springer, 2019: 21–40

[4] MONTAVON G, MÜLLER K R. Big learning and deep neural networks[M]. Neural Networks: Tricks of the Trade.[S.l.]: Springer, 2012: 419–420

[5] HAMKER F H. Predictions of a model of spatial attention using sum-and max-pooling functions[J]. Neurocomputing, 2004, 56: 329–343

[6] 李彦冬, 郝宗波, 雷航. 卷积神经网络研究综述[J]. 计算机应用, 2016, 36(9): 2508–2515

[7] RONNEBERGER O, FISCHER P, BROX T. U-Net: Convolutional Networks for Biomedical Image Segmentation[J]. CoRR, 2015, abs/1505.04597

[8] CHEN L C, ZHU Y, PAPANDREOU G, et al. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation[J]. arXiv preprint arXiv:1802.02611, 2018