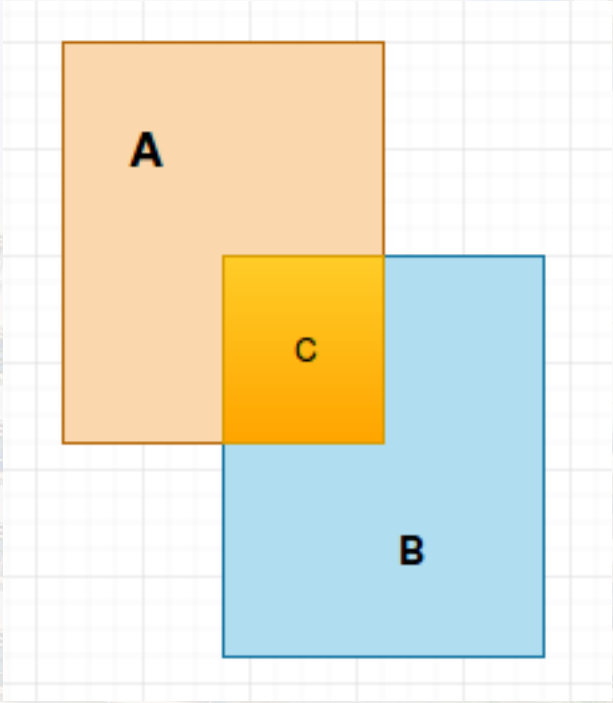
# U Net气球识别实验

# 实验内容及要求

- 实验内容：本次实验的内容为基于UNet气球识别。本次实验提供了74样本图像，其中训练集61张图像，测试集13张图像，要求使用Unet的网络结构对图像先编码后解码，生成预测的掩膜分类。计算测试集的IoU(Intersection over Union)。

IoU (Intersection over Union) 是一种常用的评估指标，用于衡量两个区域（通常是预测的区域和真实的区域）之间的重叠程度。它常用于图像分割、目标检测等计算机视觉任务中，用来评估模型预测的准确性。

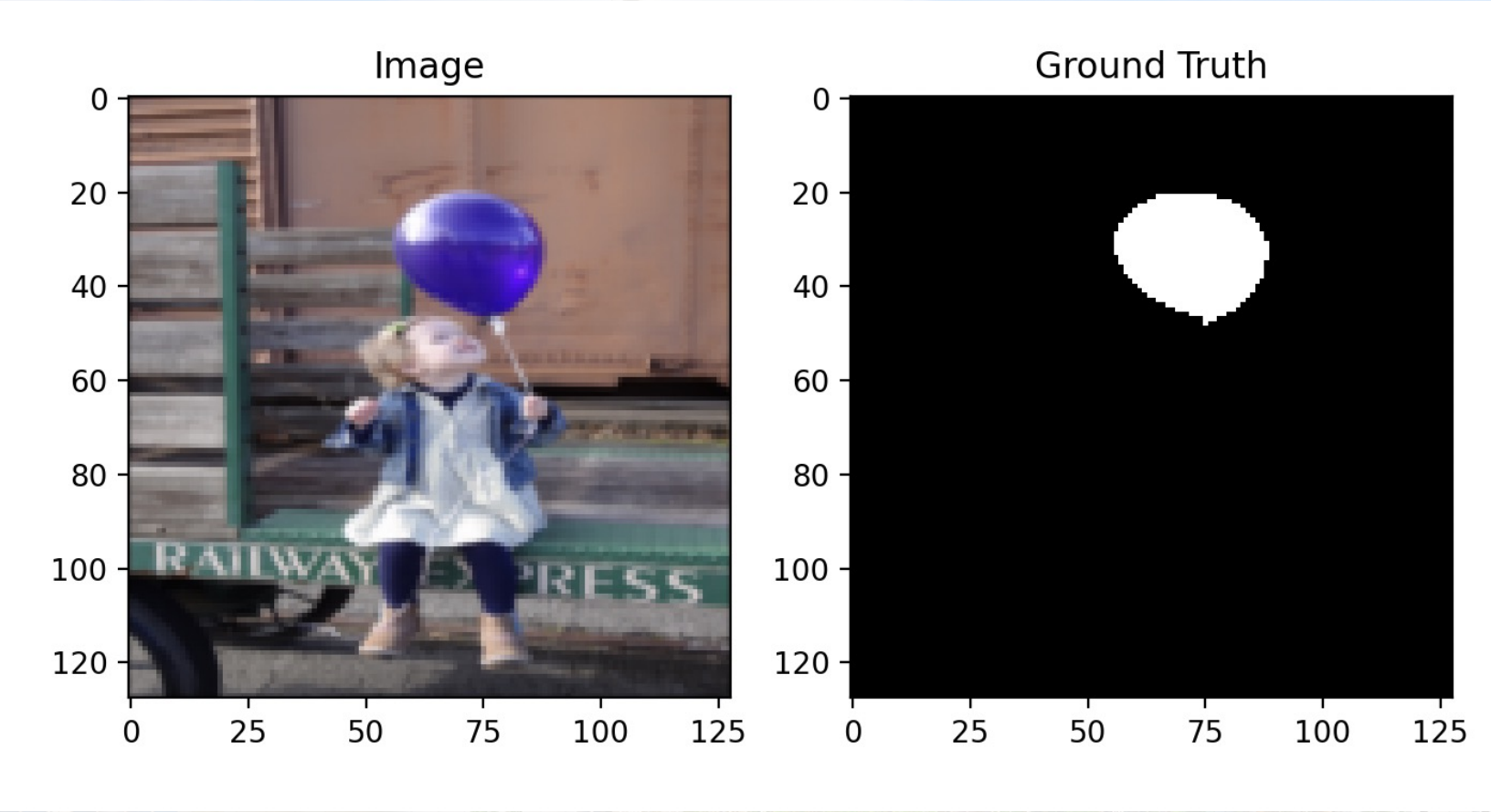



IoU 的计算公式如下：

$$\text{IoU} = = \frac{\text{Prediction} \cap \text{Ground Truth}}{\text{Prediction} \cup \text{Ground Truth}}$$

# 实验内容及要求

- 实验图片示例：



# 实验内容及要求

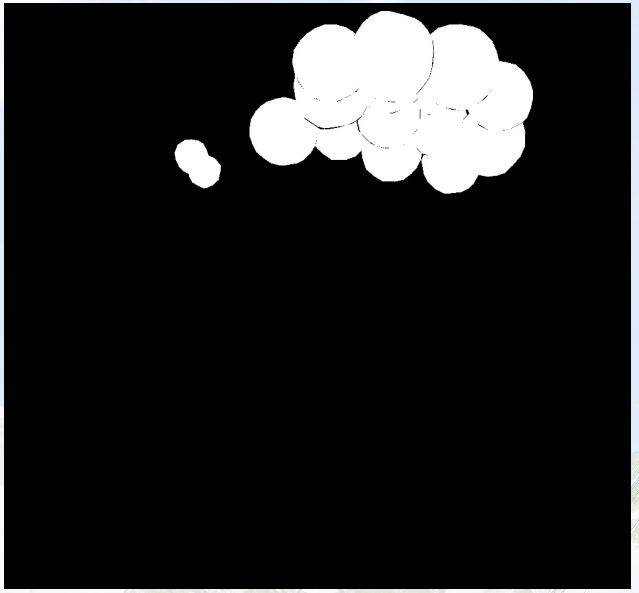
- 实验要求：
  - 1、掌握利用Unet网络的实现原理和方法
  - 2、掌握基于pytorch的的网络搭建编程
  - 3、掌握训练模型的使用

# 实验过程介绍-思路

**图像预处理**

图像尺寸：1024*956*3

获取掩膜



图像尺寸：1024*956

resize

图像尺寸：128*128*3

图像尺寸：128*128

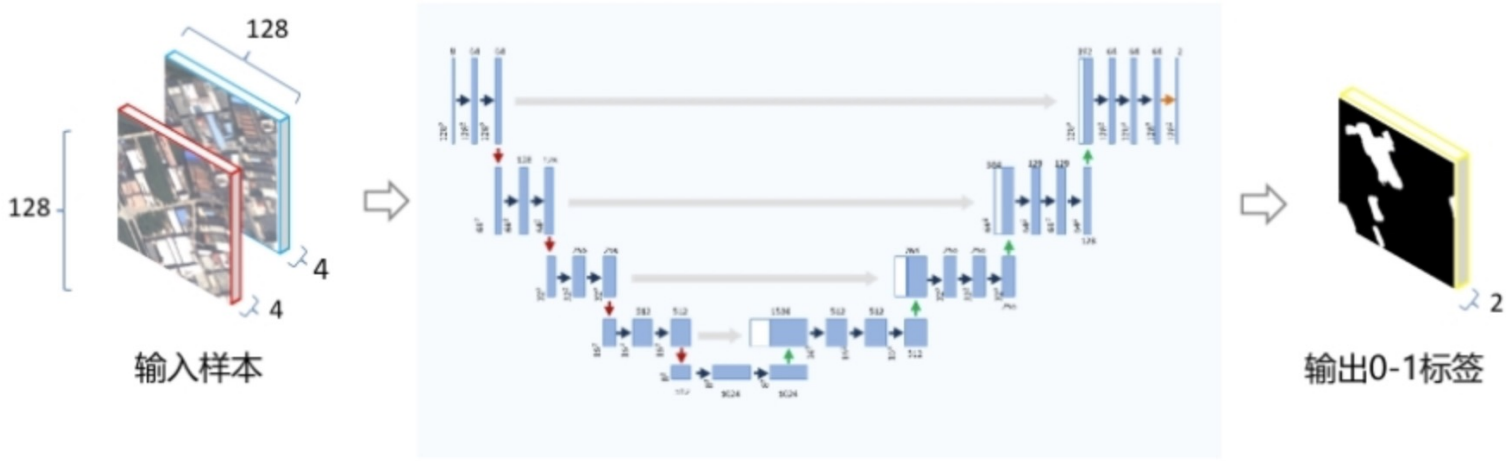
# 理论介绍-Unet的结构

UNet 的结构类似于一个对称的 U 形，包含一个编码器（Contracting Path）和一个解码器（Expanding Path）。

- 编码器逐渐将输入图像的空间分辨率降低，同时提取特征；解码器逐步恢复空间分辨率并重新构建细节。
- 跳跃连接（Skip Connections）在编码器和解码器之间传递特征，从而保持高分辨率的细节信息，提升分割效果。

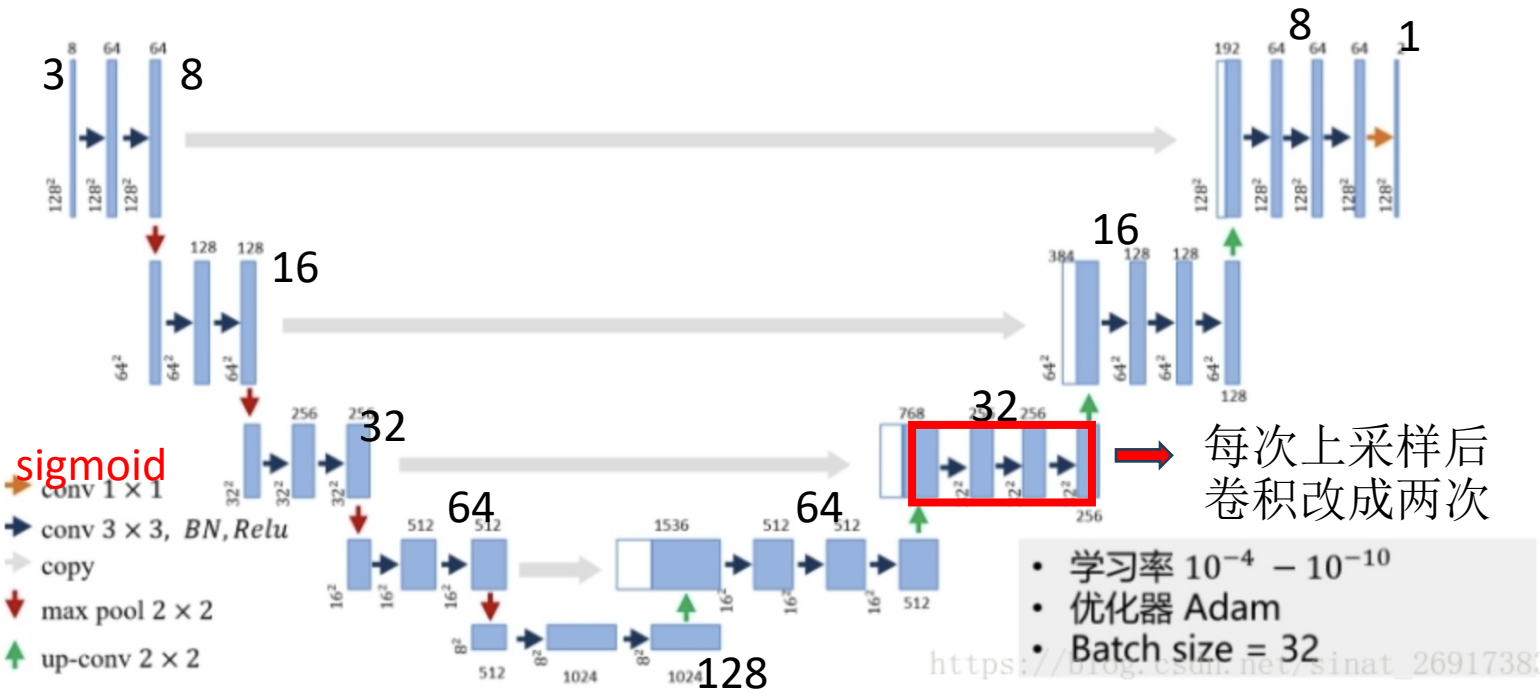
## 改进U-Net

使用U-Net检测新增建筑的整体流程如下：




U-Net网络

U-Net的整体架构如下：




- 学习率 $10^{-4}$ – $10^{-10}$
- 优化器 Adam
- Batch size = 32

# 理论介绍-Unet特征提取（编码器）

- 编码器类似于常见的卷积神经网络结构，通常包含多个卷积层和最大池化层。每个卷积块包含两层卷积操作和一次 ReLU 激活。
- 卷积层 (Conv Layer)：使用小卷积核（通常为 3 * 3），提取输入图像中的特征。
- 池化层 (Pooling Layer)：通常是最大池化（Max Pooling），每经过一次池化，图像的尺寸减半，而特征图的深度增加。
- 每次池化操作后，特征图的尺寸会减少，信息密度增加。编码器的输出是压缩的高级特征图。

# 理论介绍-Unet恢复分辨率（解码器）

- 解码器部分包含上采样（上采样卷积）和卷积层，逐步恢复特征图的空间分辨率。
- **上采样 (UpSampling or Transposed Convolution)**：使用反卷积或插值上采样，使图像尺寸逐步恢复到原始分辨率。
- 每次上采样后，解码器从跳跃连接中接收来自编码器的特征图，并在通道维度上拼接，形成丰富的特征图输入。
- **卷积层 (Conv Layer)**：恢复后的特征图再经过卷积和激活层，从而生成精细的输出，细化边缘和边界信息。

# 理论介绍-损失函数

协方差矩阵：

- **损失函数**：对于二分类分割任务，通常使用二元交叉熵损失（Binary Cross Entropy Loss）

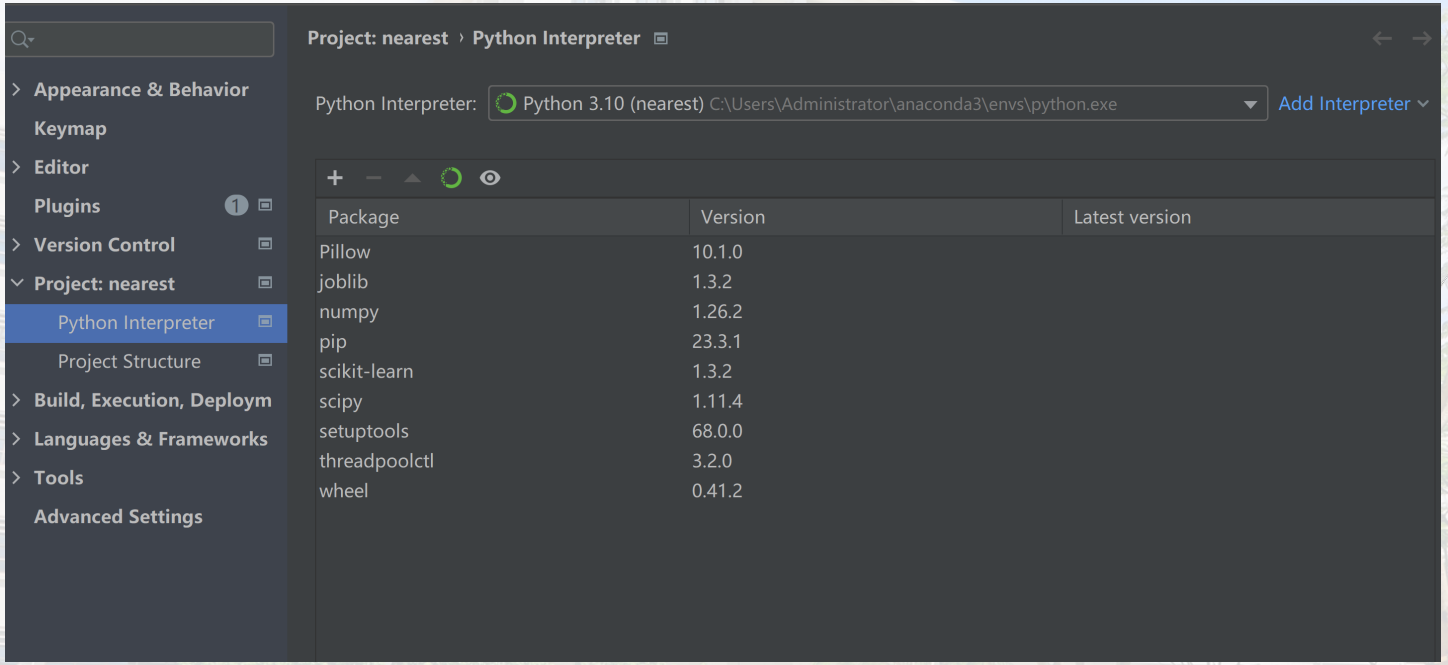
# 实验过程介绍-环境配置及测试

- 1、（实验室电脑已安装,忽略此步骤）

- 助教演示安装流程，可以带自己笔记本来安装

# 实验过程介绍-环境配置及测试

- 2、环境配置（已完成,忽略此步骤）
  安装最新版本pytorch
  安装pip依赖项
  Pip install –r requirements.txt



# 实验过程介绍-代码解读：Unet

程序流程图

# 实验过程介绍-代码解读： dataset

构造数据集类：

```python
class BalloonDataset(Dataset):  4 用法
    def __init__(self, annotations, dataset_dir, img_size=(128, 128), transform=None):
        self.annotations = annotations
        self.dataset_dir = dataset_dir
        self.img_size = img_size
        self.transform = transform

    def __len__(self):
        return len(self.annotations)

    def __getitem__(self, idx):
        tid = list(self.annotations.keys())[idx]
        a = self.annotations[tid]
        mask, image, _, _, _, _ = get_mask(a, self.dataset_dir)
        # print(mask.shape)
        # print(image.shape)
        # cv2.imshow("image", image)
        # print("mask:",np.max(mask))
        # Resize and normalize
        mask = resize(mask, self.img_size, mode='constant', preserve_range=True).astype(np.float32)
        image = resize(image, self.img_size, mode='constant', preserve_range=True).astype(np.float32) / 255.0
        if self.transform:
            image = self.transform(image)

        mask = torch.tensor(mask, dtype=torch.float32).unsqueeze(0)  # Single channel mask
        image = torch.tensor(image, dtype=torch.float32)  # Channels-first for PyTorch
        # print(image.shape)
        return image, mask
```

# 实验过程介绍-代码解读：

获取掩膜：

```
76    def get_mask(a, dataset_dir):  1个用法
77        image_path = os.path.join(dataset_dir, a['filename'])
78        image = io.imread(image_path)
79        height, width = image.shape[:2]
80        polygons = [r['shape_attributes'] for r in a['regions'].values()]
81        mask = np.zeros( shape: [height, width, len(polygons)], dtype=np.uint8)
82
83        for i, p in enumerate(polygons):
84            # Get indexes of pixels inside the polygon and set them to 1
85            rr, cc = skimage.draw.polygon(p['all_points_y'], p['all_points_x'])
86            # print(max(cc))
87            rr = list(map(lambda x: height-1 if x > height-1 else x, rr))
88            cc = list(map(lambda x: width-1 if x > width-1 else x, cc))
89            # print("i:",i)
90            mask[rr, cc, i] = 1
91
92        mask, class_ids = mask.astype(bool), np.ones( shape: [mask.shape[-1]], dtype=np.int32)
93
94        # boxes = extract_bboxes(mask)
95        boxes = extract_bboxes(resize(mask,  output_shape: (128, 128), mode='constant', preserve_range=True))
96
97        unique_class_ids = np.unique(class_ids)
98        mask_area = [np.sum(mask[:, :, np.where(class_ids == i)[0]])
99                     for i in unique_class_ids]
100       top_ids = [v[0] for v in sorted(zip(unique_class_ids, mask_area),
101                                       key=lambda r: r[1], reverse=True) if v[1] > 0]
102       class_id = top_ids[0]
103       # Pull masks of instances belonging to the same class.
104       m = mask[:, :, np.where(class_ids == class_id)[0]]
105       m = np.sum(m * np.arange(1, m.shape[-1] + 1), -1)
106       return m, image, height, width, class_ids, boxes
107
```

# 实验过程介绍-代码解读：

加载与验证数据集：

```
### 加载数据集
annotations_path = "dataset/balloon/train/via_region_data.json"
dataset_dir = 'dataset/balloon/train'
annotations = json.load(open(annotations_path))
train_transform = transforms.Compose([transforms.ToTensor()])
train_dataset = BalloonDataset(annotations, dataset_dir, transform=train_transform)
train_loader = DataLoader(train_dataset, batch_size=1, shuffle=True)

# 验证数据集
annotations_test_path = "dataset/balloon/val/via_region_data.json"
testset_dir = 'dataset/balloon/val'
annotations_test = json.load(open(annotations_test_path))
test_dataset = BalloonDataset(annotations_test, testset_dir, transform=train_transform)
test_loader = DataLoader(test_dataset, batch_size=13, shuffle=False)
```

# 实验过程介绍-代码解读：

模型训练：

```python
def train_model(model, criterion, optimizer, train_loader, val_loader, num_epochs=30):  1个用法
    best_model_wts = model.state_dict()
    best_iou = 0.0
    iou_metric = JaccardIndex(task='binary', num_classes=2).to(device)

    for epoch in tqdm(range(num_epochs)):
        model.train()
        train_loss = 0.0
        for images, masks in train_loader:
            # print(images.shape)
            images, masks = images.to(device), masks.to(device)
            masks = masks.bool()
            masks =
            optimize
            outputs = model(images)
            # print("masks:",masks.shape)
            # print(torch.max(outputs))
            # print(torch.max(masks))
            loss = criterion(outputs, masks)
            loss.backward()
            optimizer.step()
            train_loss += loss.item() * images.size(0)

        model.eval()
```

masks: Any

# 实验过程介绍-代码解读：

模型加载：

```
21    model = UNetModel( IMG_HEIGHT: 128, IMG_WIDTH: 128, IMG_CHANNELS: 3).to(device)
22    model.load_state_dict(torch.load("./best_model.pth"))
```

# 实验过程介绍-代码解读：

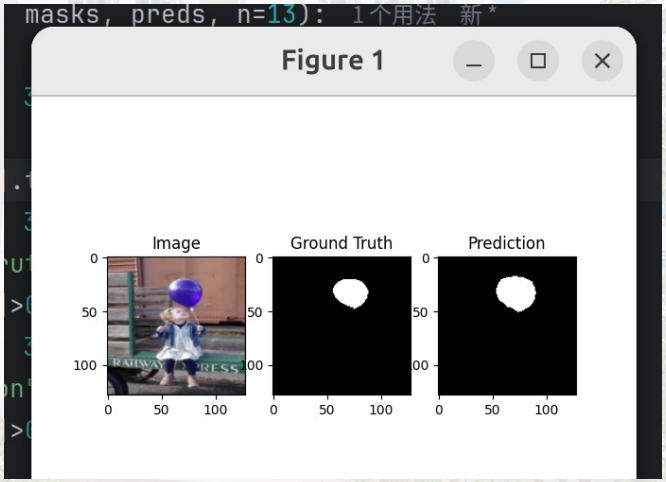
模型：

```python
def predict(model, dataloader, device):  7 个用法 (6 个动态)
    model.eval()
    preds = []
    with torch.no_grad():
        for images, _ in dataloader:
            images = images.to(device)
            outputs = model(images)
            preds.append(outputs.cpu().numpy())
    return np.concatenate(preds, axis=0)
```

# 实验任务：完成Unet模型的创建，并保存模型预测结果等（代码中need to be done部分）

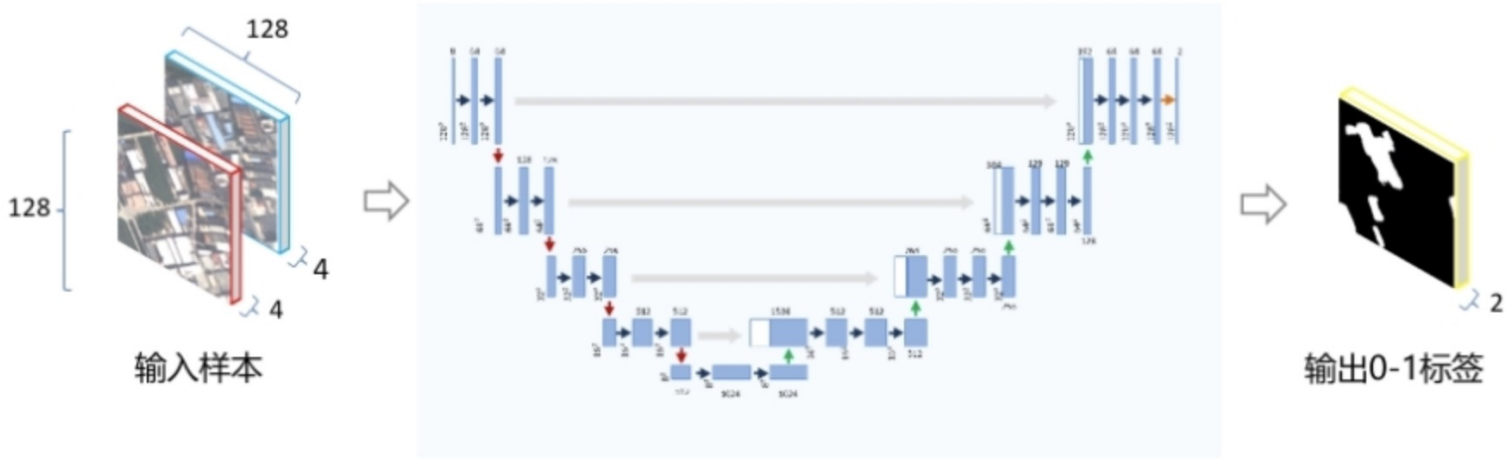
```
313  class UNetModel(nn.Module):  4 用法
314      def __init__(self, IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS):
315          super(UNetModel, self).__init__()
316      def forward(self, x):
317          .....
```

验收：保存测试集的训练结果，并给出IoU（要求平均大于0.9）
附加任务：画出折线图可视化loss下降过程，使用matplotlib绘制对比图

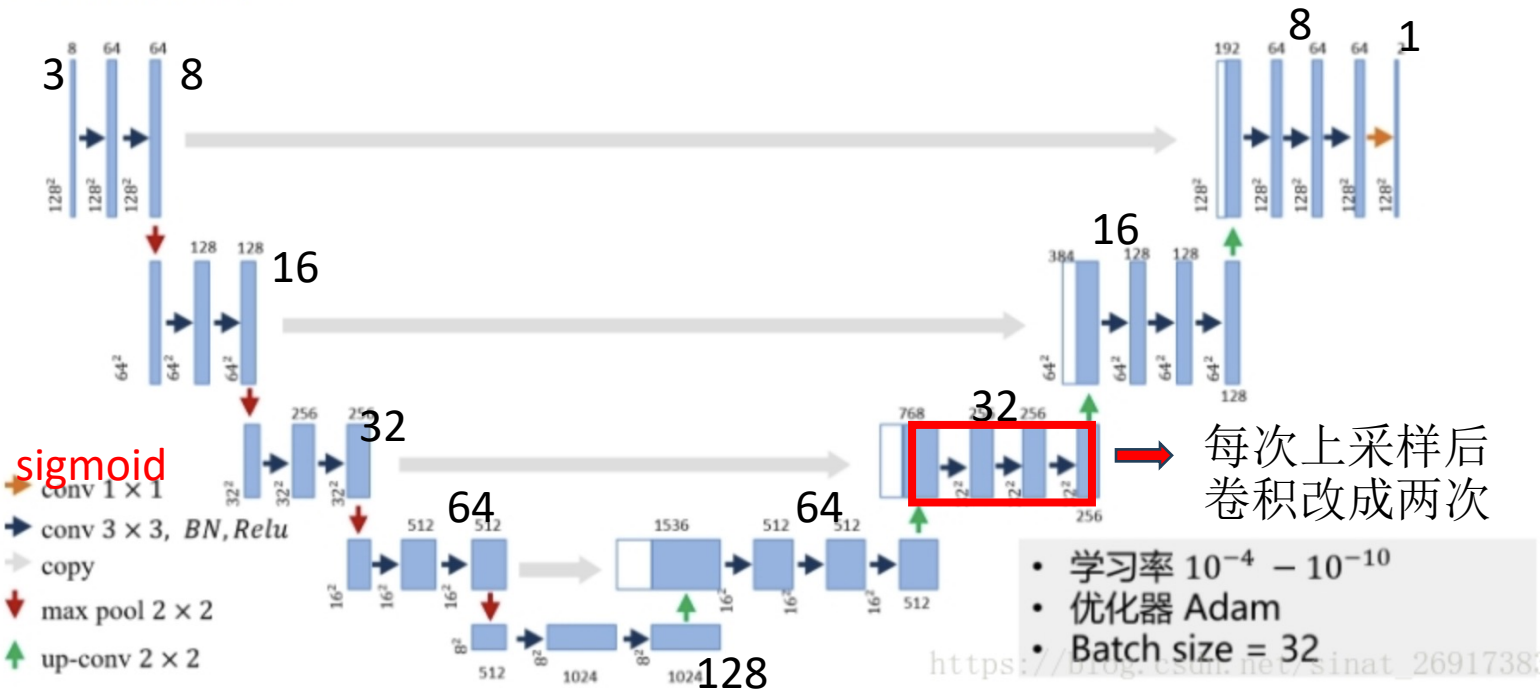


## 改进U-Net

使用U-Net检测新增建筑的整体流程如下：




U-Net的整体架构如下：



# 本章结束