# 基于 SAM 大模型的肝脏肿瘤分割软件开发

答辩人：王恺　　指导老师：刘琨

质量技术监督学院　2020 级测控普通班　2024-04-19

# 目录

# 项目介绍

## 背景

### 准确分割肝脏肿瘤在医学图像领域的重要性

肝脏肿瘤作为常见的恶性肿瘤之一，其早期发现和治疗对提高患者生存率至关重要。优秀的肝脏肿瘤分割软件能够帮助医生提供更精确的分割结果，减少工作量，提高患者的存活率。

### 工程背景

深度学习技术在图像分割领域取得了显著进展，特别是 U-Net、V-Net 等模型目前在肝脏肿瘤分割上的应用较为广泛。然而，这些模型通常需要大量标注数据进行训练，且迁移能力有限。

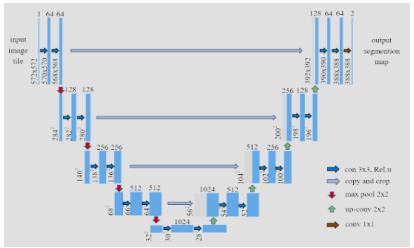


Figure 1: U-net

# 研究方案

## 使用 **SAM** 模型进行肝脏肿瘤图像分割

SAM 是一种基于 Vision Transformer 架构的图像分割模型。它通过大规模预训练学会了理解和处理各种图像特征。SAM 模型的一个关键特点是它的零样本（zero-shot）学习能力，即模型能够在没有直接在特定任务上训练的情况下，通过理解用户的提示（如文本描述、点击或框选）来执行分割任务。

## SAM 的优势

- 强大的特征学习：SAM 在大量自然图像上进行预训练，学习到了丰富的视觉特征，这些特征对于理解医学图像中的肿瘤区域非常有用。
- 泛化能力：SAM 能够泛化到新的数据集和任务上，这意味着它可以适应不同的医学图像和肿瘤类型。
- 端到端分割：SAM 可以直接从原始图像输出分割掩码，无需复杂的预处理或后处理步骤。
- 易于集成：SAM 的 API 和工具设计使得它容易集成到现有的医疗影像分析流程中，为研究调试和软件开发提供了便利。

# 项目进展

数据预处理

图像归一化

```
lower_bound = -500
upper_bound = 1000
image_data_pre = np.clip(image_data, lower_bound, upper_bound)
image_data_pre = (image_data_pre - np.min(image_data_pre))/(np.max(image_data_pre)-
np.min(image_data_pre))*255.0
image_data_pre[image_data==0] = 0
image_data_pre = np.uint8(image_data_pre)
```

将医学图像常用 **.nii** 格式转化为 **NPY** 图像格式

```
import os
import SimpleITK as sitk
import numpy as np
```

# 项目进展

```
# 输入和输出文件夹路径
input_folder = 'path_to_input_folder'
output_folder = 'path_to_output_folder'

# 获取输入文件夹中所有的.nii.gz 文件
input_files = [f for f in os.listdir(input_folder) if f.endswith('.nii.gz')]

# 遍历每个.nii.gz 文件并转换为.npz 格式
for file_name in input_files:
    # 读取.nii.gz 文件
    image = sitk.ReadImage(os.path.join(input_folder, file_name))
    image_array = sitk.GetArrayFromImage(image)

    # 保存为.npz 格式
    np.savez_compressed(os.path.join(output_folder, file_name.replace('.nii.gz',
'.npz')), image=image_array)
```

```
print("Conversion completed.")
```

将数据集分割为训练集与测试集

```
import numpy as np

# 假设 data 是你的数据集，包括特征和标签
data = np.random.randn(100, 10)
labels = np.random.randint(0, 2, size=100)

# 定义训练集和测试集的比例
train_ratio = 0.8
test_ratio = 1 - train_ratio

# 随机打乱数据集的索引
indices = np.random.permutation(data.shape[0])

# 计算训练集和测试集的样本数量
```

```
train_samples = int(data.shape[0] * train_ratio)
test_samples = data.shape[0] - train_samples

# 根据索引分割数据集
train_data = data[indices[:train_samples]]
train_labels = labels[indices[:train_samples]]
test_data = data[indices[train_samples:]]
test_labels = labels[indices[train_samples:]]

print("训练集样本数量:", train_data.shape[0])
print("测试集样本数量:", test_data.shape[0])
```

## 模型训练

### 设置优化损失函数

使用 Adam 优化器优化掩码器部分，设置 Dice+Cross Entropy Loss 作为损失函数。

```
optimizer = torch.optim.Adam(sam_model.mask_decoder.parameters(), lr=args.lr,
weight_decay=args.weight_decay)
seg_loss = monai.losses.DiceCELoss(sigmoid=True, squared_pred=True, reduction='mean')
```

### 训练模型

循环训练，通过 SAM 模型 的 掩码解码器获取预测结果，计算损失，使用优化器更新参数，最小化损失函数。

```
for epoch in range(num_epochs):
    for step, (image_embedding, gt2D, boxes) in enumerate(tqdm(train_dataloader)):
        # 获取模型输出
```

```
low_res_masks, iou_predictions = sam_model.mask_decoder(
    image_embeddings=image_embedding.to(device),
    image_pe=sam_model.prompt_encoder.get_dense_pe(),
    sparse_prompt_embeddings=sparse_embeddings,
    dense_prompt_embeddings=dense_embeddings,
    multimask_output=False,
  )

# 计算损失并优化模型
loss = seg_loss(low_res_masks, gt2D.to(device))
optimizer.zero_grad()
loss.backward()
optimizer.step()
```

# 后续工作

# 后续工作

- 对模型训练结果评估
  - 肿瘤分割的精确度、召回率和 Dice 系数等性能指标。
  - SAM 模型结果与其他分割模型的视觉比较。
  - 展示 SAM 大型模型的准确性和有效性的案例研究或实例。
- 前端接口
  - 为软件开发的前端界面。
  - 界面功能，包括结果显示、交互工具和其他功能。
  - 实现便捷的图像输入，结果显示操作。
  - 用户友好的设计和医疗专业人员的易用性。
- 论文撰写

感谢老师指导!