《人工智能在医学影像技术中的应用》实验报告

姓名：杨浩然	学号：221180053

一、 实验目的

```
写在前面：

可惜这门课程的课时太少，很难在这么短的时间里面深入地了解和学习机器学习和深度学习在图像处理中的应用；还有一些关于一些机器学习算法和神经网络模型数学上的表达的理解，则更难以做到。但是，作为一门选修课来说，我了解到了机器学习和深度学习的一些概念，并且可以自己动手构建一个机器学习算法亦或是深度学习的简单网络，从通识的角度来说又是非常有意义的。

在AI潮流席卷的今天，接触到AI算法，并在算法解决回归，分类这些问题中思考算法构建者的思想，也是一种乐趣。有时候，也许过程远比结果更重要。
```

学习并了解深度学习的相关概念，并通过yolov8模型尝试完成对于医学影像数据集的目标检测任务。最后，评估模型性能并探索超参数对于网络检测能力的影响。

二、 实验原理与方法

2.1 数据集

鼓楼医院采集的乳腺癌影像集：
训练集：验证集：测试集=8：1：1的比例随机划分数据集，验证集和测试集

- train: 训练集480张
- val: 验证集61张
- test: 测试集60张

每一个数据集下面的构成：

- images：训练的图片
- labels：用于存放yolo格式的标注txt文件

2.2 深度学习与迁移学习原理特点

深度学习基于神经网络，通过多层隐藏层逐步提取数据中的复杂特征，从而实现对复杂任务的建模。深度学习中的神经网络具有万有逼近性，它的每一层网络通过非线性激活函数对输入进行变换，使得整个网络能够逼近任何复杂的非线性函数。通过多层网络的逐层抽象，深度学习能够从原始数据中自动学习高层次的特征表示，避免了传统方法中需要人工设计特征的困难。这种结构在图像识别、语音处理等任务中具有巨大的优势和潜力。

2.2.1深度学习的基本网络架构

这里用最简单的深度学习网络多层感知机（MLP）为例来介绍深度学习的基本网络架构：

（1）输入层：接收数据输入，通常是原始数据，如图像的像素值、文本的单词向量等。

（2）隐藏层：通过多个神经元对输入进行处理，层数较多的神经网络被称为“深度神经网络”。

（3）输出层：根据网络学习的结果，给出最终的预测或分类结果。

```
class MLP(nn.Module):
    def __init__(self):
            super(MLP, self).__init__()
            self.network = nn.Sequential(
                nn.Linear(5, 64),
                nn.ReLU(),
                nn.Linear(64, 256),
                nn.ReLU(),
                nn.Linear(256, 1024),
                nn.ReLU(),
                nn.Linear(1024,512),
                nn.ReLU(),
                nn.Linear(512, 300)
        )
```

前向传播：数据从输入层传递到输出层，经过各个隐藏层的处理。每一层通过加权和激活函数计算出下一层的输出。

```
def forward(self, x):
        return self.network(x)
```

反向传播：深度学习的核心训练过程，利用误差反向传播算法（Backpropagation），将输出的误差通过网络传回各个层，计算每个权重的梯度，并根据这些梯度调整权重，以减小误差。通常使用梯度下降法来更新权重。

```
#梯度清零
optimizer.zero_grad()
outputs = model(inputs)

# 计算总的损失
total_loss = criterion(outputs[:, 0:300], targets[:, 0:300])  # 总损失

# 反向传播
total_loss.backward()
optimizer.step()
```

激活函数：激活函数在神经网络中起到重要作用，它决定了神经元是否被激活，帮助网络捕捉非线性特征。除了上文所列出的ReLU函数之外，常用的激活函数有：

（1）Sigmoid：输出值在0到1之间，适用于二分类问题，但容易出现梯度消失问题。

（2）Tanh：输出值在-1到1之间，适用于处理具有对称性的数据。

损失函数：损失函数（或目标函数）衡量模型预测值与真实值之间的差异。通过最小化损失函数来训练模型。

```
#定义损失、优化器和学习率
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

除了代码中所使用的MSE之外，还有一下常用的损失函数：

（1）交叉熵：常用于分类问题，特别是二分类和多分类任务。

（2）优化算法： 优化算法通过计算梯度并调整网络权重来优化模型，最常见的优化算法包括：

（3）梯度下降法（Gradient Descent）：通过计算损失函数的梯度来更新权重。

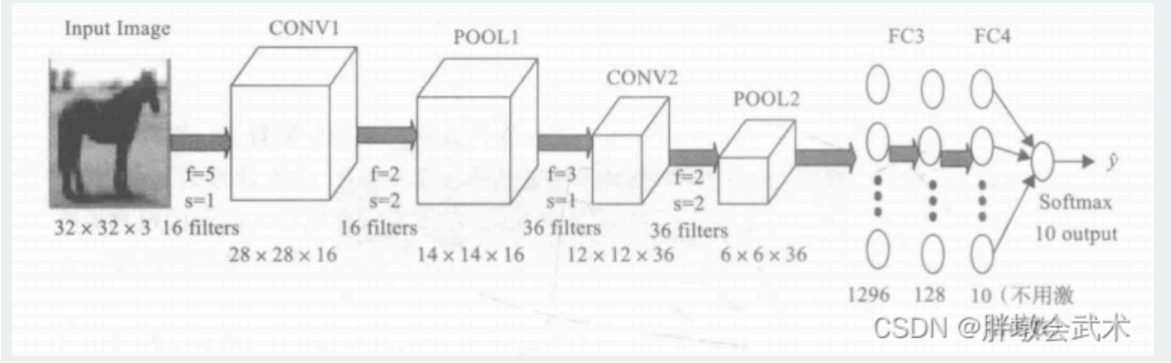
（4）随机梯度下降（SGD）：每次使用一个小批量数据来估算梯度，能够减少计算开销。

（5）Adam：结合了梯度下降和动量的方法，广泛应用于深度学习中，能更快地收敛。

2.2.2 常见的深度学习网络介绍

以下是对各类深度学习网络的简要介绍：

1. 卷积神经网络 (CNN)



卷积神经网络（CNN）主要用于处理图像和视觉数据。其结构通常包括：

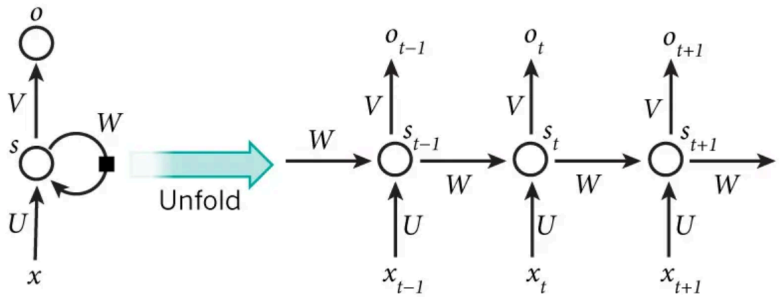
（1）输入层：接受原始数据（如图像）输入。

（2）卷积层：通过卷积操作提取局部特征，如边缘、纹理等。

（3）池化层：通过下采样（如最大池化）减少特征图的维度，保留最重要的信息。

（4）全连接层：将提取的特征进行分类或回归等任务，生成最终输出。 CNN能够自动学习数据中的空间层次特征，在图像识别、目标检测等任务中表现卓越。

2. 循环神经网络 (RNN)

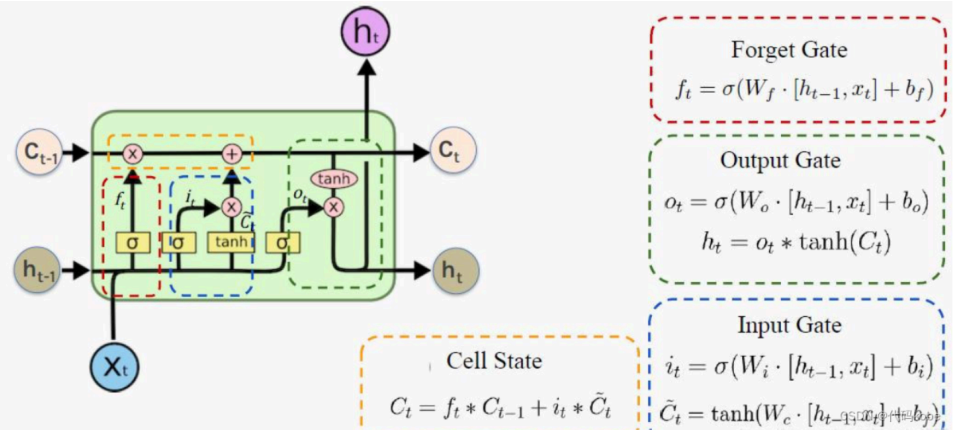


循环神经网络（RNN）主要用于处理序列数据，如文本、时间序列等。其核心特点是通过循环连接使网络在时间维度上具有记忆能力，能够处理序列中元素之间的依赖关系。但标准RNN存在梯度消失或梯度爆炸问题，难以处理长序列数据。

3. 生成对抗网络 (GAN)

生成对抗网络（GAN）由两个网络组成：生成器（Generator）和判别器（Discriminator）。生成器尝试生成逼真的数据，判别器则尝试判断数据是否来自真实数据集。通过对抗训练，生成器不断改进，以产生更加真实的数据。GAN广泛应用于图像生成、图像超分辨率、风格迁移等任务。

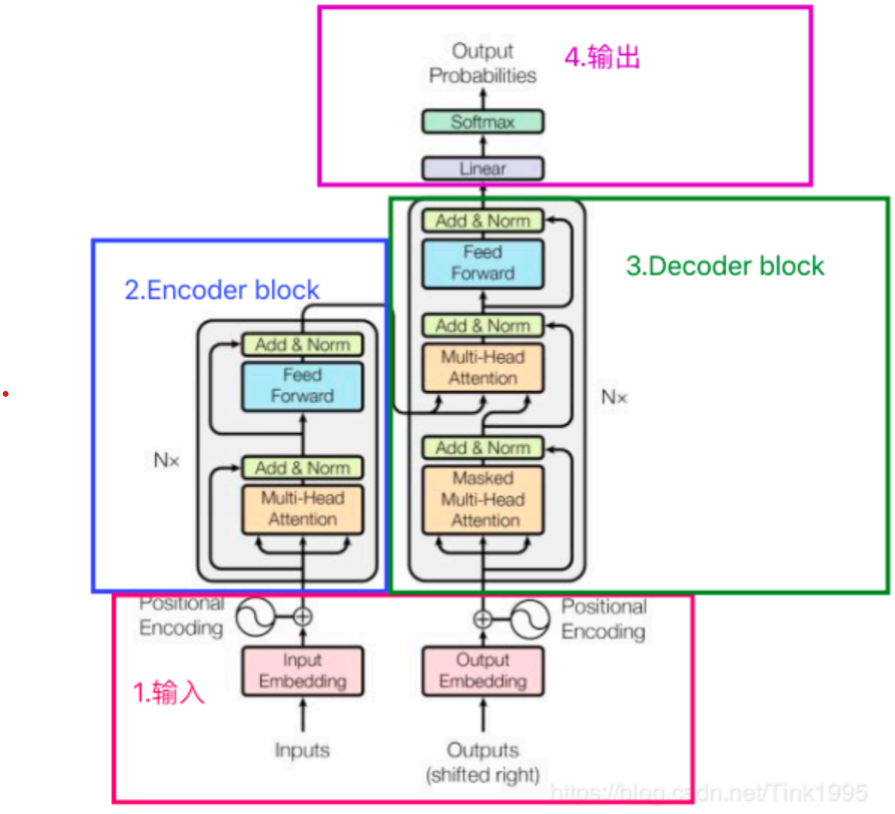
4. 长短期记忆网络 (LSTM)



Ct是细胞状态（记忆状态），X_t是输入的信息，h_{t-1}是隐藏状态（基于C_t得到的）

长短期记忆网络（LSTM）是RNN的一种改进，旨在解决传统RNN在处理长序列时出现的梯度消失或梯度爆炸问题。LSTM通过引入遗忘门、输入门、输出门和记忆门来控制信息的传递与记忆，从而有效捕捉长时间依赖关系，广泛应用于自然语言处理、语音识别等领域。

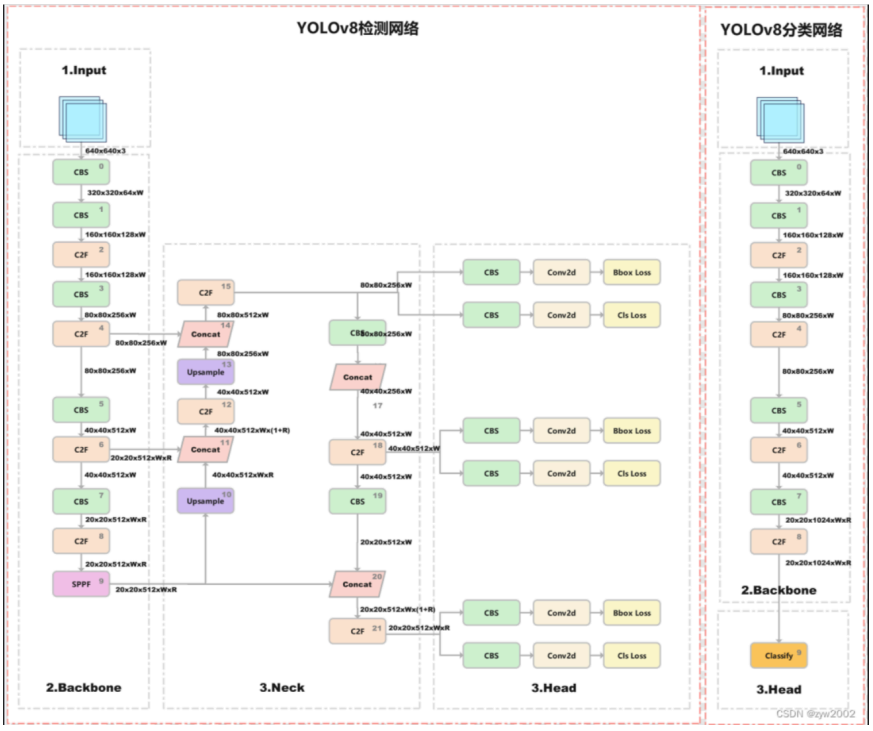
5. Transformer (转换器)



Transformer是一种基于自注意力机制（Self-Attention）的网络架构，主要用于处理序列数据，尤其在自然语言处理（NLP）任务中表现突出。其结构包括Encoder（编码器）和Decoder（解码器）两个部分，Encoder负责输入数据的特征提取，Decoder负责生成输出。自注意力机制使得模型能够捕捉序列中不同位置的依赖关系，并且具有并行计算的优势。Transformer是许多先进NLP模型（如BERT、GPT）

的基础。

6. YOLO



YOLO是一种实时物体检测模型，其核心思想是在单一网络中同时进行物体定位和分类。YOLO模型将图像划分为多个网格，并在每个网格上预测边界框和类别，最终输出物体的位置和类别。其高效性使得YOLO在实时物体检测任务中得到广泛应用，如视频监控、自动驾驶等。

2.3.yolo模型的基本原理

YOLO模型将物体检测任务转化为一个**回归问题**，而非像传统的检测方法那样分阶段处理（例如，先生成候选框，再分类）。其主要原理可以分为以下几个步骤：

1. **输入图像的划分**： YOLO模型首先将输入图像划分为一个 $S \times S$ 的网格，每个网格负责检测图像中物体的某一部分。每个网格单元预测多个边界框（bounding boxes）及其对应的类别概率。
2. **每个网格预测多个信息**： 对于每个网格单元，YOLO模型会输出以下信息：

   B个边界框（通常B=2或3），每个边界框包括：

   x，y：框的中心点坐标，相对于该网格的位置。

   w，h：框的宽度和高度，相对于整个图像的比例。

   置信度（Confidence）：预测框内包含物体的概率以及该框与真实框的重叠程度（通过IoU（Intersection over Union）计算）。

   C个类别概率（C是类别数）：每个网格单元预测物体属于某个类别的概率。

   因此，每个网格单元的输出通常是一个包含上述信息的向量，YOLO最终将每个网格的预测结果拼接起来，得到整个图像的预测信息。
3. **输出的结构**： 假设图像大小为 416×416，YOLO将其划分为 $S \times S$ 的网格，每个网格单元预测B个边界框和C个类别的概率。因此，YOLO网络的最终输出是一个 $S \times S \times (B \times 5 + C)$ 的张量。其中：

   $B \times 5$：每个边界框的5个值：x，y，w，h 和 置信度。

   C：类别概率。

4. **边界框过滤和NMS（非最大抑制）**： YOLO模型输出多个边界框，而这些边界框往往有很大的重叠，因此需要通过**非最大抑制（NMS）**来去除冗余框。NMS会根据置信度分数选择最佳的边界框，保留具有最高置信度的框，去除与其重叠度（IoU）超过一定阈值的其他框。
5. **回归问题的转化**： YOLO将物体检测视为一个回归问题，即直接从图像像素回归出每个物体的位置和类别，而不是通过滑动窗口或区域提议生成候选框后再进行分类。这样可以显著提高检测速度，并减少计算复杂度。

2.4 迁移学习特点

迁移学习是一种通过借用在一个任务上获得的知识来提高在另一个相关任务上学习效率的机器学习方法。它的核心思想是将源任务（通常是数据较多的任务）中学到的特征、模式或模型参数迁移到目标任务中，从而减少目标任务对大量标注数据的依赖，提升其学习效果。在深度学习中，迁移学习通常通过使用在大规模数据集上预训练的模型，并对其进行微调，来适应目标任务。这种方法尤其在数据匮乏或目标任务与源任务具有一定相似性时表现出显著优势，能够有效提升模型性能，同时减少训练时间和计算成本。

迁移学习方法：冻结模型的部分参数与权重，只对部分权重和参数进行优化。可以使得训练时间大大减少并且摆脱对于计算资源的依赖。

三、 实验步骤与结果

运行mytrain.py文件，观察训练结果，并且调整超参数观察是否可以使得训练效果进一步得到提升；

初始优化器为Auto（在迭代次数大时使用“SGD”,在数据量小时使用“Adam”）

```
val: Scanning D:\ultralytics-main\images\val\labels.cache... 61 images, 0 backgrounds, 0 corrupt: 100%|          | 61/61 [00:00
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|          | 16/16 [00:01<00:00,
                   all         61         68      0.983      0.926      0.967      0.721
Speed: 0.3ms preprocess, 19.3ms inference, 0.0ms loss, 0.3ms postprocess per image
```

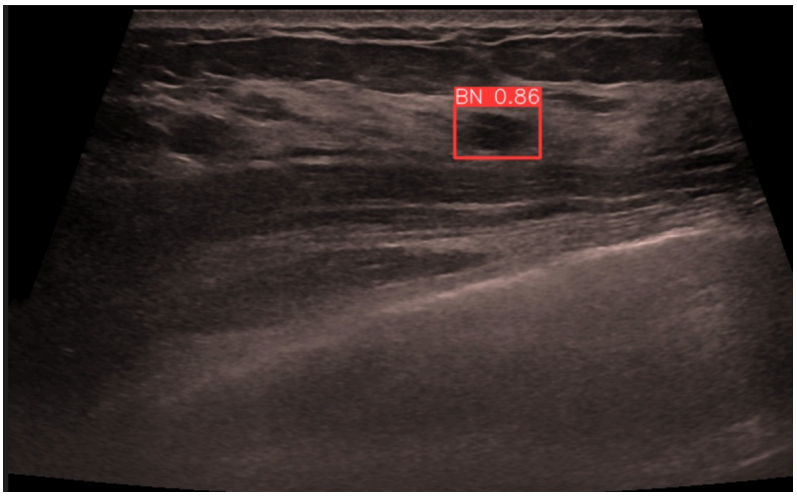
由于这里迭代次数没有达到阈值，默认使用“Adam”优化器；而将优化器调整为“SGD”之后：

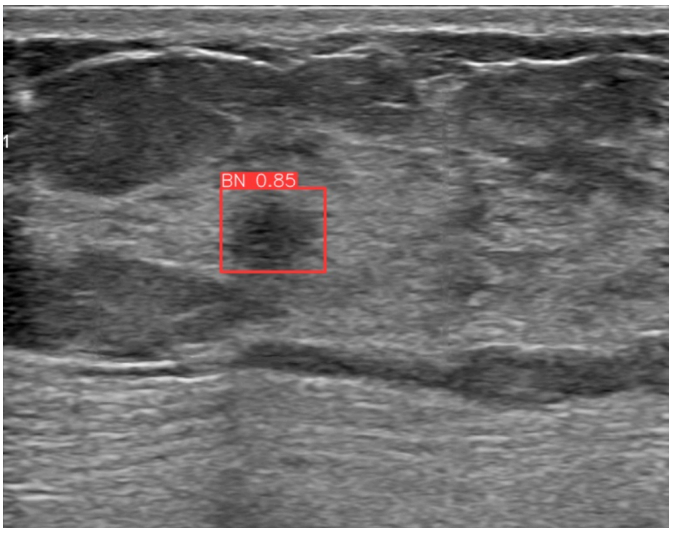
```
Model summary (fused): 168 layers, 3005843 parameters, 0 gradients, 8.1 GFLOPs
val: Scanning D:\ultralytics-main\images\val\labels.cache... 61 images, 0 backgrounds, 0 corrupt: 100%|          | 61/61 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|          | 16/16 [00:01<00:00, 10.31it/s]
                   all         61         68      0.982      0.926      0.964      0.716
Speed: 0.3ms preprocess, 18.2ms inference, 0.0ms loss, 0.3ms postprocess per image
```

整体上差别不大，效果上”SGD“相较于“Adam”要稍差一些。

运行mypredict.py文件，观察预测结果：

```
image 1/60 D:\ultralytics-main\images\test\images\case0166_00002.jpg: 512x640 1 BN, 43.8ms
image 2/60 D:\ultralytics-main\images\test\images\case0181_00002.jpg: 512x640 1 BN, 25.1ms
image 3/60 D:\ultralytics-main\images\test\images\case0235_00011.jpg: 512x640 1 BN, 24.9ms
image 4/60 D:\ultralytics-main\images\test\images\case0351_00008.jpg: 480x640 1 BN, 34.1ms
image 5/60 D:\ultralytics-main\images\test\images\case0387_00005.jpg: 512x640 1 BN, 24.1ms
image 6/60 D:\ultralytics-main\images\test\images\case0404_00006.jpg: 512x640 1 BN, 24.0ms
image 7/60 D:\ultralytics-main\images\test\images\case041_00010.jpg: 416x640 1 BN, 31.1ms
image 8/60 D:\ultralytics-main\images\test\images\case0457_00006.jpg: 416x640 1 BN, 21.6ms
image 9/60 D:\ultralytics-main\images\test\images\case0475_00014.jpg: 480x640 1 BN, 24.7ms
image 10/60 D:\ultralytics-main\images\test\images\case0481_00003.jpg: 480x640 1 BN, 24.1ms
image 11/60 D:\ultralytics-main\images\test\images\case0503_00008.jpg: 544x640 1 BN, 30.7ms
image 12/60 D:\ultralytics-main\images\test\images\case0687_00005.jpg: 544x640 1 BN, 25.6ms
```





上面截取部分预测结果。

四、 实验心得与体会

实验心得收获：

通过对于yolo模型的训练，了解了深度学习的相关知识和迁移学习在训练中的独特意义。除此以外，本次实验让我对于目标检测有了进一步的认识。

yolo算法则让我受益良多。yolo算法创造性地使用bounding box来将目标检测的问题成功转化成一个回归问题。从一般的函数的回归广播成了对于空间中bounding box的拟合。并且，yolo算法在整体上相较于使用卷积层和池化层来进行分批的查看的CNN网络来说显得更加的高效。正如其名，you only look once：只需要一次拟合就可以得到目标检测结果，大大减少了检测次数，提高了检测效率。