一．模型架构

最开始我们使用了 CNN 对图片进行分类，但是 CNN 模型对局部区域的噪声信号过于敏感。对于乳腺图像，由于其多纹理的特性，CNN 架构容易学习到大量噪声，而无法有效提取用于分类的正确特征信息，如图 1 所示。

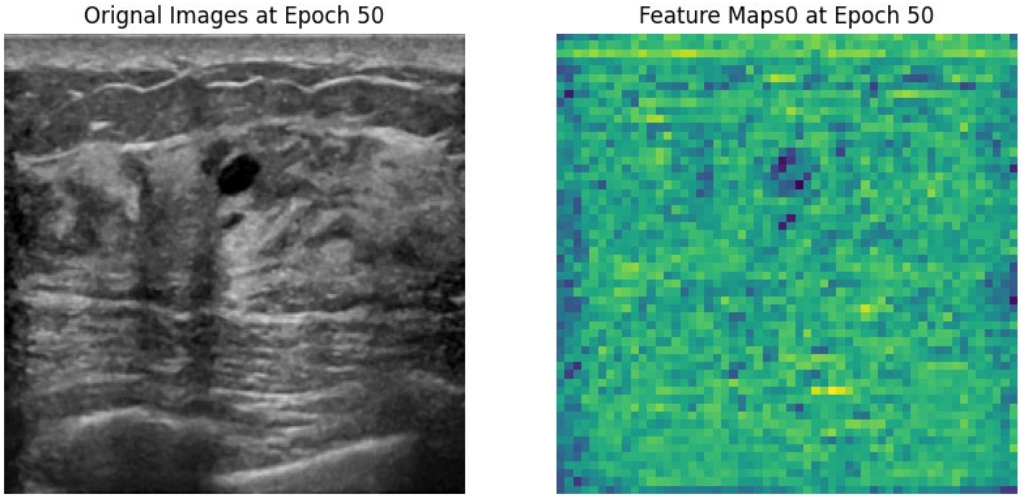

图 1 学习到背景噪声的 cnn

作为一种在计算机视觉任务中表现出色的 Transformer 变种，Swin Transformer 使用局部注意力机制，相对于 CNN 架构而言，对乳腺图像的多纹理模式具有更强的抗噪能力，同时保持了 CNN 架构低参数量的优点。因此，本模型以 Swin Transformer 作为 backbone。分类模型的全连接层输出维度为 6，对应 6 个 BIRADS 类型概率的相对大小；特征识别模型的全连接层输出维度为 4，对应 4 个特征为阳性的概率。

具体来说，Swin Transformer 具有以下特点：

1. 窗口划分（Window Partition）

局部自注意力计算: 将输入图像划分为多个非重叠的窗口，每个窗口内的 token 进行局部自注意力计算，显著减少了计算复杂度。

动态窗口大小: 可以根据输入分辨率动态调整窗口大小，适应不同尺度的输入。

2. 移位窗口（Shifted Window）

跨窗口交互: 在相邻的层中，窗口的位置进行移位，使得每个 token 能够与相邻窗口的 token 进行交互，增强了模型的建模能力。
注意力掩码: 使用注意力掩码处理移位窗口带来的边界问题，确保注意力计算的正确性。

3. 多层感知机（MLP）和前馈网络

高效的 MLP 设计: 使用 GELU 激活函数和 Dropout 层，提高了 MLP 的表达能力和泛化能力。
残差连接: 在 MLP 和自注意力模块之间添加残差连接，增强了模型的训练稳定性。

4. 相对位置编码（Relative Position Bias）

相对位置编码: 使用相对位置编码捕获 token 之间的相对位置信息，增强了模型的空间感知能力。
参数共享: 相对位置编码参数在所有窗口中共享，减少了模型的参数量。

5. 层规范化（Layer Normalization）

层规范化: 在每个 Swin Transformer 块中使用层规范化，提高了模型的训练稳定性和泛化能力。
位置规范化: 在位置嵌入后使用层规范化，确保位置信息的有效传递。

二．数据预处理步骤

1. 动态图像处理和数据增强

为了提高模型的泛化能力和训练效果，我们在数据预处理和增强方面采用了以下方法：
（1）颜色抖动。随机调整图像的颜色属性，包括亮度、对比度、饱和度和色调，增加模型对不同光照条件的鲁棒性。
（2）自动增强: 自动选择一系列数据增强策略，这些策略已经在多个数据集上

进行了验证，能够显著提升模型性能。

（3）随机擦除。随机删除图像的一部分区域，模拟遮挡情况，增强模型的泛化能力。

（4）高级插值方法。 使用 InterpolationMode.BICUBIC 进行图像缩放，这种方法在保持图像质量的同时，减少了失真，提高了模型的性能。

2. 标准化的图像预处理

为了提高模型的训练效果和泛化能力，我们在图像预处理方面采用了标准化处理和中心裁剪的方法。

（1）图像标准化。使用 ImageNet 数据集的均值 [0.485, 0.456, 0.406] 和标准差 [0.229, 0.224, 0.225] 对图像进行标准化处理，这有助于模型更快地收敛，并使图像数据的分布更加均匀。

（2）中心裁剪。对验证集图像进行中心裁剪，确保验证集的数据处理方式与训练集一致，避免过拟合。中心裁剪保留了图像的核心部分，减少了边缘信息的影响。通过这些方法，我们的图像预处理流程不仅提高了模型的训练效果，还确保了验证集的数据处理方式与训练集一致，避免了过拟合。

3. 数据加载器的优化

（1）多线程数据加载。使用 torch.utils.data.DataLoader 的 num_workers 参数，设置多个工作进程并行加载数据，提高数据加载速度，减少训练时间。

（2）内存固定。 使用 pin_memory 参数，将数据加载到固定内存中，加速数据从 CPU 到 GPU 的传输速度。

三．模型训练

1. 学习率调度器

（1）余弦退火（Cosine Annealing）学习率调度器。余弦退火通过动态调整学习率，使其从初始值 ηmaxηmax 逐渐降低到最小值 ηminηmin，遵循余弦函数的形式。这种平滑的退火过程有助于模型在训练后期更稳定地收敛，提高最终性能。

余弦退火策略通过平滑地调整学习率，避免了突变，有助于模型更好地适应训练数据的变化。

（2）预热阶段。在训练初期，学习率从一个较小的初始值 ηwarmupηwarmup 逐渐增加到设定的学习率 ηmaxηmax。这有助于模型更快地进入稳定状态，避免梯度爆炸。预热阶段通过平滑地增加学习率，帮助模型在训练初期更快地找到合适的参数更新方向。

2. 优化器

使用 AdamW 优化器，这是一种 Adam 优化器的改进版，结合了 L2 正则化，适用于复杂的模型和小批量训练。AdamW 优化器通过动态调整学习率和动量，提高了模型的收敛速度和最终性能。此外，我们还对不同类型的参数应用了不同的权重衰减策略，避免不必要的正则化，进一步提高了模型性能。具体来说，对于形状为一维的参数（如偏置项）和特定关键字的参数（如 norm 层），不应用权重衰减，从而避免不必要的正则化。

3. 正交化提高模型的稳定性和泛化能力

在深度学习模型训练中，权重矩阵的正交性对于提高模型的稳定性和泛化能力至关重要。我们使用了一种基于双软正交化（DSO）的权重正则化方法。在训练过程中对模型的权重矩阵施加正交性约束：遍历模型的所有非偏置参数，将参数展平为二维矩阵，并计算其对称矩阵 SS。通过对称矩阵减去单位矩阵再计算其绝对值之和作为正交化损失。最终，将正交化损失添加到标准损失中，形成总损失。这种方法通过软性约束实现了权重矩阵的近似正交性，有效减少了梯度消失和梯度爆炸的问题，提高了模型的训练稳定性和泛化能力。

4. 梯度缩放和混合精度训练

为了提高模型的训练稳定性和效率，我们采用了梯度缩放和混合精度训练的方法。我们使用了 NativeScalerWithGradNormCount 类对梯度进行缩放，防止梯度爆炸，提高训练稳定性。在反向传播过程中对梯度进行缩放，确保训练过程的稳定性。同时，我们结合了 torch.cuda.amp 模块，使用混合精度训练，将部分

计算从 FP32 转换为 FP16，减少内存占用，加快训练速度。通过这些方法，我们不仅提高了训练的稳定性，还显著减少了内存占用，加快了训练速度。

5. 特征图可视化

在训练过程中定期可视化特征图，帮助理解模型的内部工作机制，发现潜在的问题，如图 2、图 3 所示。在每个 epoch 结束后随机选择一个验证集样本，生成特征图并保存。使用 matplotlib 库绘制原始图像和特征图，帮助理解模型的内部工作机制。

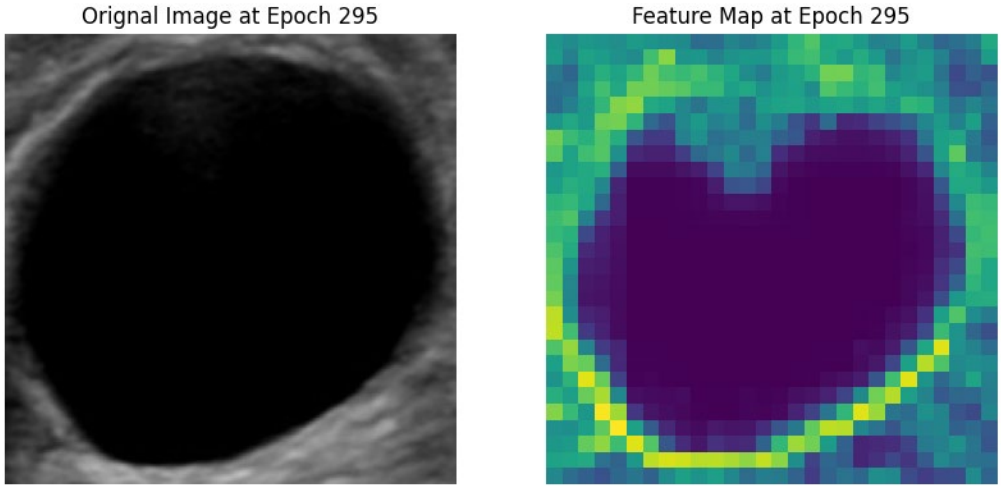

图 2 swin_transformer 特征图可视化

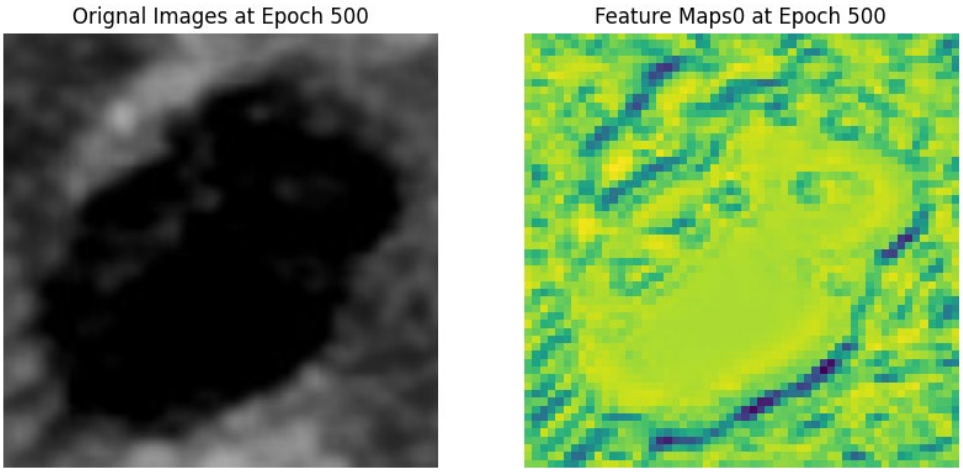

图 3　convnext 特征图可视化

四．测试流程

在本项目的测试过程中，我们首先通过自定义的数据集类加载和预处理测试图像。具体步骤包括读取彩色图像，转换为灰度图像后进行二值化处理，并通过轮廓检测提取图像中的最大边界矩形区域进行裁剪，以确保模型关注于图像的主要部分。接着，分别加载分类模型（cla_test）和特征提取模型（fea_test），并将其移动到 GPU 上。使用预定义的图像转换流程对裁剪后的图像进行标准化处理后，将图像批量输入模型进行推理。分类模型输出的类别标签和特征提取模型的多标签结果被分别收集，并与对应的图像名称一同保存到 cla_pre.csv 和 fea_pre.csv 两个 CSV 文件中。为了提高测试效率，我们优化了数据加载器的参数设置，如设置工作线程数量 num_workers=8，并启用了内存固定 pin_memory=True 以加快数据传输到 GPU 的速度。这一系列流程确保了模型能够高效、准确地对测试数据进行分类和特征提取，同时生成结构化的预测结果文件，便于后续的分析和应用。