**长　沙　学　院**

智慧医疗

学 院： 计算机科学与工程学院

班 级： 21智能01班

学 号： B20210307118

姓 名： 谢文杰

2024年 1 月 5 日

# 选题与数据集选择

**背景与意义**

青光眼是全球导致不可逆失明的第二大原因，仅次于白内障，威胁着人们的视觉健康。根据世界卫生组织的统计，全球青光眼患者已超过7000万，预计到2040年将增至1.1亿人。青光眼由眼压升高引起的视神经损害，早期症状往往不明显，许多患者直到视力受损才意识到问题。然而，早期发现并干预可以减缓病情进展。因此，青光眼的早期诊断和筛查至关重要。

传统诊断方法依赖医生经验，如眼压测量、视野检查和眼底成像，但这些方法主观性较强，且受限于医生的技术水平。在此背景下，深度学习特别是卷积神经网络（CNN）在医学影像分析中的应用为青光眼的自动化诊断提供了新的解决方案。通过深度学习，模型能从大量数据中学习到复杂特征，减少人为误差，实现疾病的自动诊断。

本次实验采用深度学习技术，通过Fundus Glaucoma Detection Data数据集，结合PyTorch框架和EfficientNet V2-S模型，对青光眼进行分类研究。该数据集包含大量眼底图像，适用于训练深度学习模型，以提高青光眼的自动化筛查效率，减少诊断误差，并为临床提供准确、快速的诊断工具。

通过自动化分析眼底图像，旨在提高青光眼筛查的诊断效率。利用深度学习技术，能够更早发现青光眼，及时进行干预，防止病情恶化。同时，本研究为医学AI应用提供了实践经验，推动了医学影像分析领域的技术发展。随着深度学习技术的不断进步，AI将在医学影像分析中发挥越来越重要的作用，进一步提升医疗服务的质量和效率，可以成为临床医生的重要辅助工具。

**数据集选择**

Fundus Glaucoma Detection Data（眼底青光眼检测数据）数据集是SMDG-19 for Glaucoma 数据集PyTorch 格式的子集，是一个专门用于青光眼检测的眼底图像数据集，由多个医疗机构提供，数据集包含了大量标注为0和1的眼底图像，用于青光眼的分类任务。数据集包括以下部分：

训练集（Train）：包含大量已标注的训练图像，用于模型训练。

验证集（Val）：用于模型训练过程中超参数的调优和模型性能的验证。

测试集（Test）：用于评估模型的最终性能。

该数据集具有较高的质量和多样性，包含的眼底图像多来自不同的患者，覆盖了不同年龄段、性别和不同的眼底病变情况，这使得该数据集成为青光眼分类研究的理想数据源。数据集中的标签为0和1，其中1表示存在青光眼，0表示不存在青光眼。

数据集下载地址：[眼底青光眼检测数据](https://www.kaggle.com/datasets/sabari50312/fundus-pytorch/data)

# 数据预处理

在训练卷积神经网络（CNN）模型时，预处理不仅帮助模型更好地理解输入数据，还能提高模型的训练效率和泛化能力。以下是我选择的针对训练集和测试集的预处理方法，包括图像归一化、裁剪、增强、旋转、翻转等操作。

**训练集预处理（train\_transforms）**

train\_transforms = v2.Compose([

    v2.Resize(256),

    v2.RandomResizedCrop(size=(224, 224), antialias=True),

    v2.RandomHorizontalFlip(p=0.5),

    v2.RandomVerticalFlip(p=0.5),

    v2.RandomAffine(degrees=(-10, 10), translate=(0.1, 0.1), scale=(0.9, 1.1)),

    v2.RandomErasing(p=0.5, scale=(0.1, 0.15)),

    v2.PILToTensor(),

    v2.ToDtype(torch.float32),

    v2.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),

])

**测试集预处理（test\_transforms）**

test\_transforms = v2.Compose([

    v2.Resize((224, 224)),

    v2.PILToTensor(),

    v2.ToDtype(torch.float32),

    v2.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

])

**具体操作步骤**

1、图像缩放和裁剪：

Resize(256)：将输入图像缩放到256x256，确保图像的大小不会影响后续裁剪的质量和多样性。

RandomResizedCrop(size=(224, 224))：随机裁剪224x224的区域，这有助于模型更好地学习从不同位置和尺度上提取特征，增加数据的多样性，避免模型过度拟合特定区域或尺度的特征。

2、随机翻转：

RandomHorizontalFlip(p=0.5) 和 RandomVerticalFlip(p=0.5)：随机翻转图像，增加模型对物体在水平或垂直方向上的不变性的适应能力。例如，物体的方向变化不会影响模型的识别能力，提升了模型的鲁棒性。

3、几何变换（旋转、平移、缩放）：

RandomAffine(degrees=(-10, 10), translate=(0.1, 0.1), scale=(0.9, 1.1))：通过随机旋转、平移和缩放，模型可以学习到在不同角度、位置和尺寸下的特征，从而更好地适应现实世界中的变化。

4、随机擦除：

RandomErasing(p=0.5, scale=(0.1, 0.15))：随机擦除图像的一部分区域，迫使模型不依赖于某个局部特征进行分类。这个方法增强了模型的鲁棒性，尤其在图像中部分遮挡的情况下有助于模型继续正确识别图像内容。

5、图像归一化：

Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])：归一化操作是为了将图像的像素值调整为标准范围。我使用了ImageNet的均值和标准差，使得训练过程中数据的分布与ImageNet预训练模型兼容，能够加速收敛并提升性能。

6、Tensor 转换和数据类型设置：

PILToTensor() 和 ToDtype(torch.float32)：将图像转换为Tensor类型并设置为float32，这是PyTorch深度学习框架所需的数据格式。Tensor格式是进行矩阵运算和深度学习训练所必需的，float32也是标准的数值类型。

测试集的变换较为简单，仅包括调整图片尺寸至模型所需的输入尺寸、转换为Tensor格式、数据类型转换为float32以及标准化处理。这些变换没有使用任何随机变换，目的是确保测试过程的一致性和可重复性，从而使模型在评估阶段的表现能够真实反映其泛化能力，而不受额外随机因素的干扰。同时，这些变换确保了测试数据的处理与训练数据处理方式一致，以便模型在未见过的数据上能够保持良好的表现。

# 模型构建

class\_size = 2

model = models.efficientnet\_v2\_s(weights='DEFAULT')

model.classifier[1] = torch.nn.Linear(1280, class\_size)

EfficientNetV2 是一种高效的卷积神经网络架构系列，在计算效率和分类准确性之间取得了很好的平衡。EfficientNetV2-S 是该系列中的一个小型变种，设计上注重效率，能够在较低的计算资源下获得较好的性能。

**EfficientNetV2-S 模型主要由以下几个部分组成：**

1 输入层

输入图像的尺寸大小为(224, 224, 3)，并且在训练时会使用 ImageNet 数据集的均值和标准差进行归一化进行标准化处理。

2 特征提取部分：卷积层

EfficientNetV2-S 使用多个卷积层和深度可分离卷积来提取特征。用复合缩放方法来同时调整网络的宽度、深度和分辨率，获得最佳的平衡。

3 全局平均池化

在特征提取之后，模型通常会使用全局平均池化层将所有特征图的空间维度（例如高和宽）进行平均，最终输出一个固定长度的特征向量。这种池化方式既可以减少参数量，也能保留丰富的特征信息。

4 分类头（最后的全连接层）

原始的 EfficientNetV2-S 模型的分类头是一个全连接层，输出维度为 1000，用于处理 ImageNet 中的 1000 类分类任务。为了适应青光眼二分类任务，我将分类头的输出维度从 1000 改为 2。

通过这些方式，利用预训练的 EfficientNetV2-S 模型，可以在保持高效性的同时，快速适应新的任务，尤其是当数据集不大时，预训练模型的优势尤为明显。

# 模型训练与调优

**训练脚本**

def train(dataloader, model, loss\_fn, optimizer, lr\_scheduler):

    size = len(dataloader.dataset)

    num\_batches = len(dataloader)

    model.train()

    epoch\_loss = 0.0

    epoch\_correct = 0

    for (data\_, target\_) in tqdm(dataloader):

        target\_ = target\_.type(torch.LongTensor)

        data\_, target\_ = data\_.to(device), target\_.to(device)

        optimizer.zero\_grad()

        outputs = model(data\_)

        loss = loss\_fn(outputs, target\_)

        loss.backward()

        optimizer.step()

        epoch\_loss = epoch\_loss + loss.item()

        \_, pred = torch.max(outputs, dim=1)

        epoch\_correct = epoch\_correct + torch.sum(pred == target\_).item()

    lr\_scheduler.step()

    return epoch\_correct / size, epoch\_loss / num\_batches

model.to(device)

EPOCHS = 10

criterion = nn.CrossEntropyLoss()

for epoch in range(EPOCHS):

    train\_acc, train\_loss = train(train\_loader, model, criterion, optimizer, multi\_step\_lr\_scheduler)

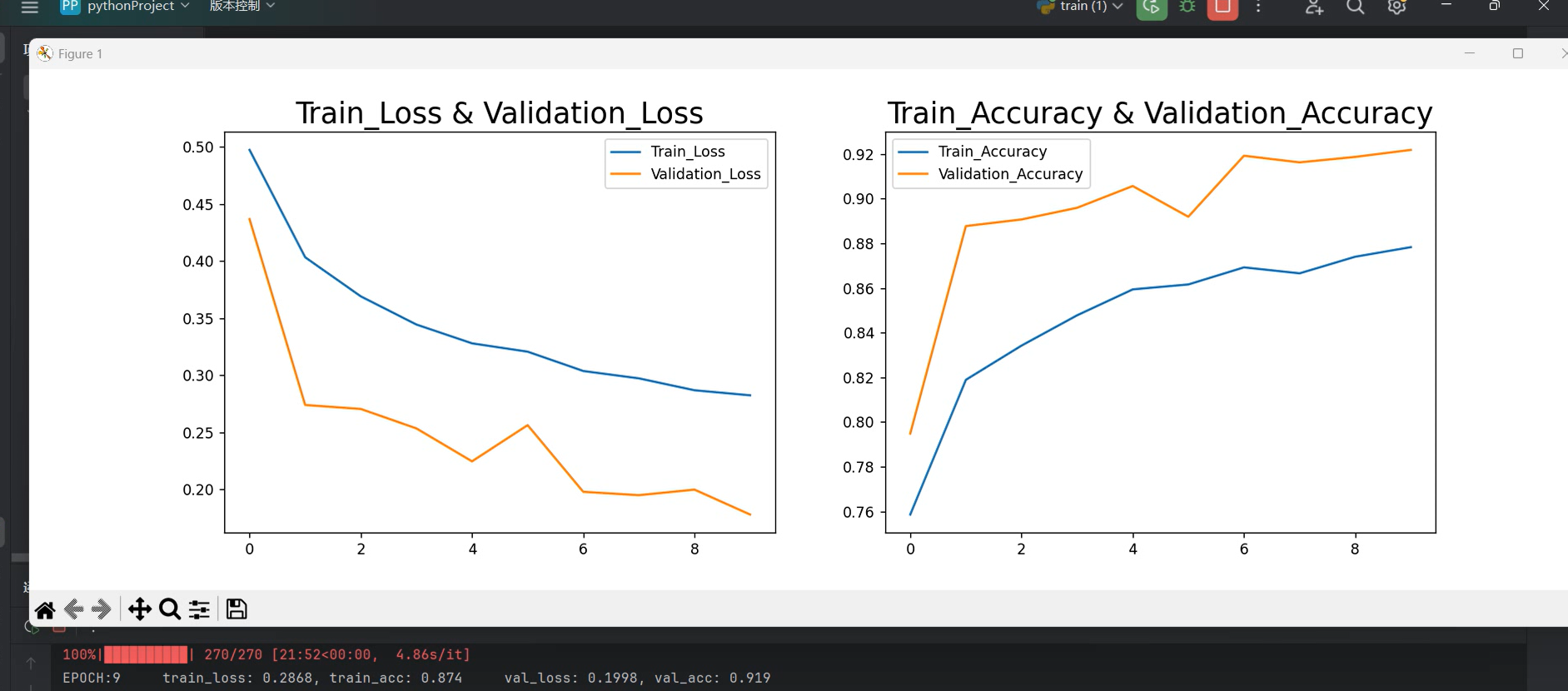
    val\_acc, val\_loss = test(val\_loader, model, criterion)

    print(f'EPOCH:{epoch + 1} \

    train\_loss: {train\_loss:.4f}, train\_acc: {train\_acc:.3f} \

    val\_loss: {val\_loss:.4f}, val\_acc: {val\_acc:.3f} ')

**结果图表**



**调优问题与解决**

**问题1：训练损失不下降或过拟合**

**问题描述：**模型在训练集上的表现不佳，损失没有显著下降，或者训练集与验证集的性能差距过大，表明模型可能过拟合。

**解决方案:**

1、数据增强：通过增加样本的多样性，可以有效减少过拟合。使用数据增强（如 train\_transforms）能够让模型见到更多不同的训练样本。

2、权重衰减（L2 正则化）：通过在优化器中加入权重衰减，可以防止模型参数过大，减轻过拟合。代码中已使用 torch.optim.AdamW 优化器，该优化器内建了 L2 正则化功能。

3、早停（Early Stopping）：在验证损失不再持续降低时，终止训练以避免过拟合。代码中已实现该功能，通过监控验证损失来实现早停。

**问题2：**模型收敛慢

**问题描述：**模型在训练过程中，损失和准确率的提升非常缓慢。

**解决方案:**

1、调整模型架构：尝试更复杂的模型，或者针对特定任务设计的模型结构，可以加速收敛。需要根据具体任务来调整。

2、增大批次大小：例如，将批次大小 BATCH\_SIZE 设置为 32 或更大。可以加快训练速度，但要注意硬件内存的限制。

3、数据平衡：确保训练集中的类别分布均衡，避免由于类别不平衡导致的偏斜学习。可以使用过采样、欠采样或者类别加权等方法来平衡数据集。

**问题3：结果不稳定**

**问题描述：**测试时，模型预测结果波动较大，结果不稳定。

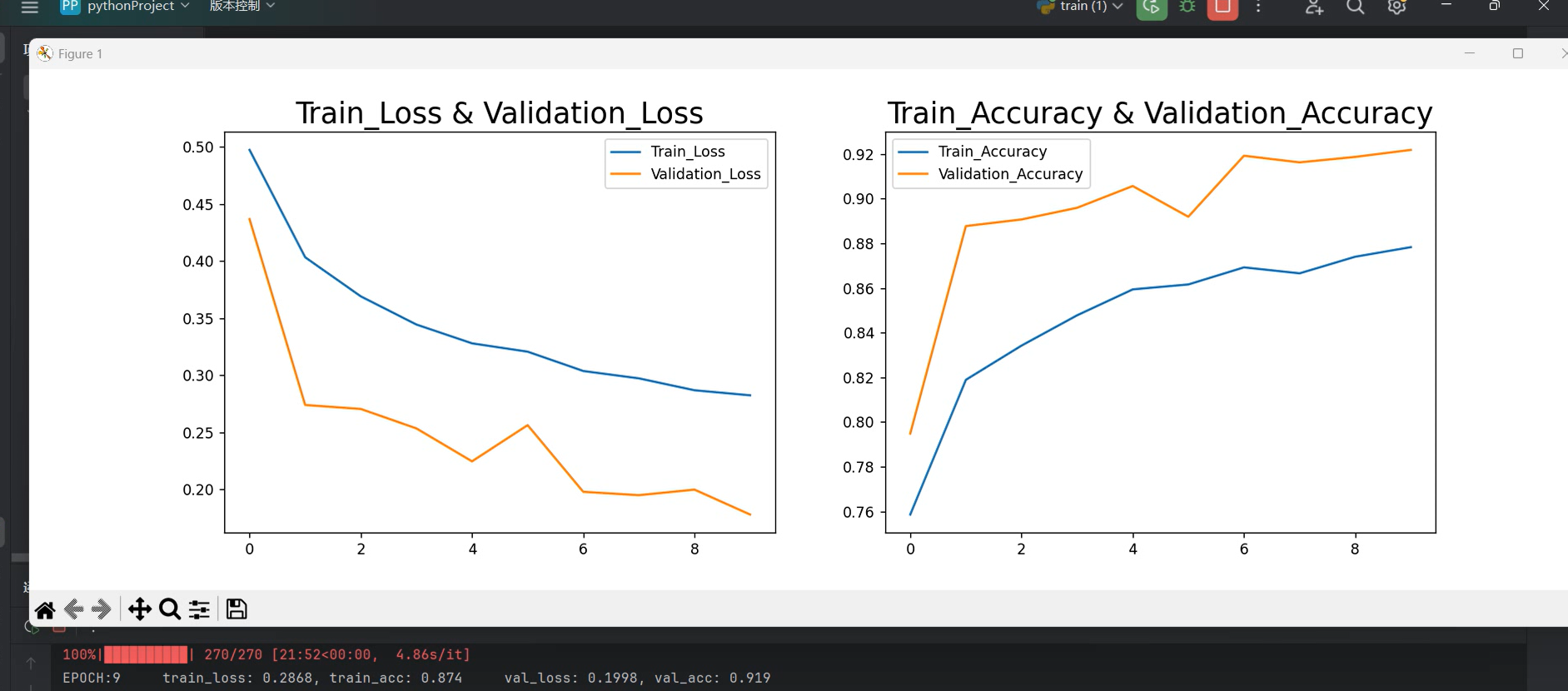
**解决方案:**

1、重复实验：多次运行训练和测试过程，检查模型结果的一致性，避免由于随机初始化或数据顺序问题导致的波动。

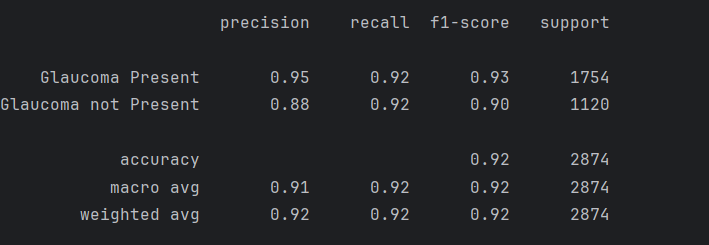
2、保存和恢复最佳模型：通过保存验证集上损失最低时的模型状态，确保最终测试使用的是最优模型。代码中已实现，通过检查验证损失来保存最佳模型。

# 模型评估

**结果图表**

****

****

****

该模型表现出色，具有较高的精确率、召回率和F1得分，尤其在青光眼存在的预测上表现突出。准确率高达92%，证明模型能够较好地分类青光眼与非青光眼样本，适用于青光眼筛查。然而，精确率和召回率之间的差异表明，青光眼不存在的样本有时会被误分类为青光眼。因此，在临床应用中，可能需要结合其他辅助检查方法，以确保诊断的准确性和可靠性。

**优势：**

1、模型选择：EfficientNet V2系列模型在保持较高准确率的同时，具有更小的模型体积和更高的计算效率。选择EfficientNet V2-S作为基础模型，有利于在资源有限的情况下快速训练和部署。

2、数据增强：通过v2.Compose应用了一系列的数据增强技术，如随机裁剪、水平和垂直翻转、随机仿射变换和随机擦除等，这能有效增加训练集的多样性，减少模型过拟合的风险，并提升模型的泛化能力。

3、早停机制：通过设置耐心值（patience）来实现早停策略，可以避免在验证集性能不再提升时浪费计算资源，是一种有效的防止过拟合的方法。

4、模型评估与可视化：代码不仅计算了训练和验证的损失及准确率，还绘制了训练过程中的指标变化图以及混淆矩阵，同时输出了分类报告，这些都有助于深入理解模型的表现并进行调试优化。

**不足之处：**

1、训练时间长：模型训练到最后得出各类结果时间较长，训练轮数10轮花费将近4个半小时，长时间占用GPU或CPU资源，影响其他任务的执行。

2、模型复杂度与调整：虽然EfficientNet模型较为高效，但直接使用预训练模型并仅替换最后一层可能不是所有情况下最优的选择。针对特定任务，微调更多层或者对模型结构进行适当调整可能会带来更好的性能。

3、超参数调优：学习率、权重衰减等超参数的选择相对固定，没有进行细致的调优。使用网格搜索、随机搜索或贝叶斯优化等方法来寻找最佳超参数组合可能会进一步提升模型性能。

# 代码链接

网址：https://github.com/xxx-wj/ZHYL

Url：[xxx-wj/ZHYL: 智慧医疗大作业](https://github.com/xxx-wj/ZHYL)

# 总结

本次实验通过应用深度学习技术，结合EfficientNet V2-S模型和Fundus Glaucoma Detection Data数据集，成功实现了青光眼的自动化分类。实验结果表明，该模型在青光眼筛查任务中表现出较高的准确性，这些结果验证了深度学习方法在医学影像分析中的潜力，尤其是在青光眼的早期筛查和诊断中。

深度学习技术，特别是EfficientNet V2-S模型的应用，能够有效减少传统诊断方法中的人为误差，提高筛查效率与准确性。此外，模型能够在有限的计算资源下提供较为高效的解决方案，适应不同规模的医疗机构。然而，实验也暴露出一些局限性，如数据集的多样性和图像质量对模型性能的影响，这需要在未来进一步优化和补充数据。

总的来说，本研究为青光眼自动化诊断提供了有效的技术支持，具有广泛的应用前景。未来可以通过结合多模态数据、优化模型架构以及拓展数据集来进一步提升模型的泛化能力，推动其在实际临床中的应用。随着技术的不断进步，深度学习有望成为青光眼及其他眼科疾病诊断中的重要工具，为全球眼科领域的发展做出贡献。