六、模型构建与实验

**（一）搭建实验环境**

**本实验代码使用Python语言编写，调用了当下主流的深度学习库PyTorch以搭建卷积神经网络模型，并使用Pandas、OpenCV等库对图像数据进行读取、变换，使用Ultralytics库中的YOLOv8-n模型进行目标检测，使用Sklearn中的相关方法计算Kappa系数等模型评估数据，调用Optuna中的TPE算法框架对深度学习模型的超参数进行调优。另外，为加速模型训练过程，本实验需要使用GPU进行计算，因此依赖NVIDIA GPU、CUDA框架以及Cudnn神经网络训练加速模块。**

**（二）数据的读取与处理**

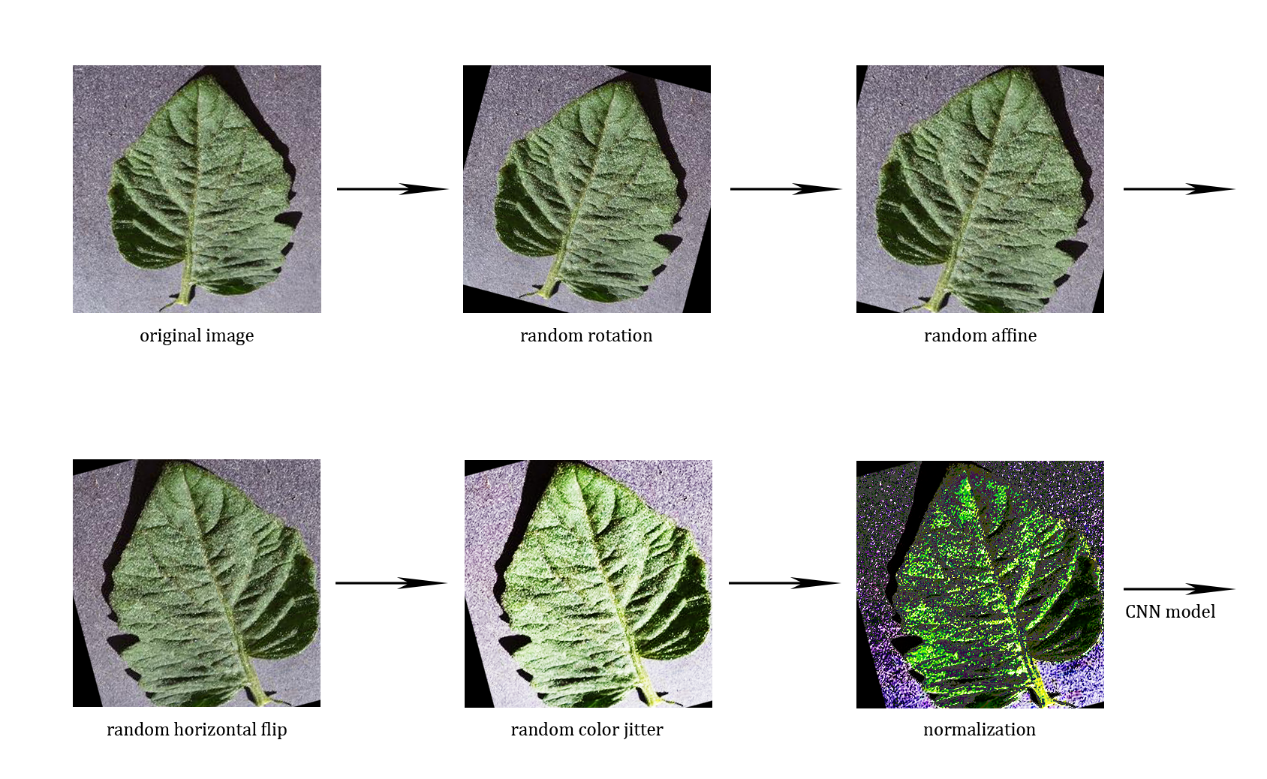
**使用Torchvision中的ImageFolder类，可以方便地读取已经完成分类且存放在树状文件夹中的图片数据。加载之后的数据集已经分好训练集与验证集，我们随机抽出了验证集中三分之一的数据，用于最终模型效果的测试。以下是训练集中各个类别图片的数量统计：**

**插入统计柱状图**

**统计./datasets/TomatoLeavesDataset/train下各文件夹文件数**

**由于训练集中各个类别的数据量并没有出现极端不平衡的现象，因此我们没有使用相关算法均衡数据集。**

**在读取数据之后，我们选择对图片做数据增强处理。数据增强包括对图片进行平移、旋转、翻转、缩放、对比度调整等操作，这种做法可以增加数据的多样性，从而防止模型过拟合，迫使模型学习更加一般化的特征。下图展示了对一张训练集进行数据增强的过程：**

****

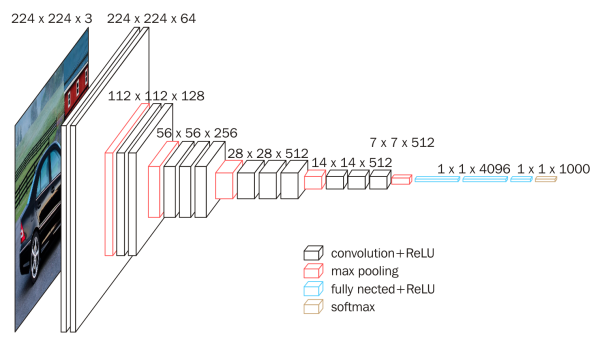
**经过数据增强的图像被进一步转化为张量形式，并按通道进行归一化处理。归一化后RGB三通道数据的分布近似是均值为**，方差为的正态分布。在上图中，经过归一化处理的图像似乎在肉眼观察上会有失真，这是图像不同通道的数据分布偏差造成的，不会影响模型训练效果核最终表现。

**（三）模型的构建与调整**

**在本研究中，我们使用了三种不同的CNN模型——VGG16模型、ResNet50模型、ResNeXt-CBAM模型进行对比试验，以深入探究不同架构的神经网络在训练过程中与最终表现上的差别。下面介绍这三种模型的基本结构和搭建方法。**

1. **VGG16模型**

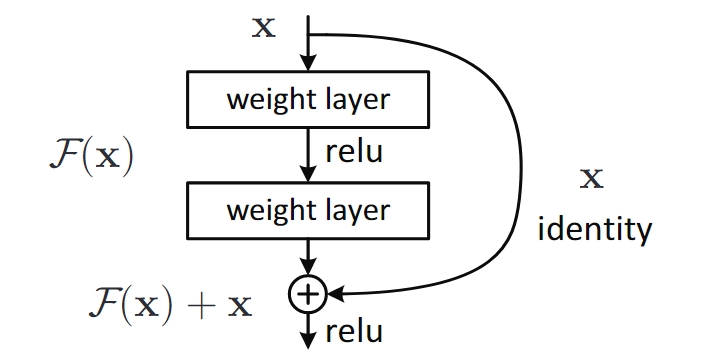
**VGG（**Visual Geometry Group Network**）是一种经典的卷积神经网络模型，由**Visual Geometry Group提出，并在ImageNet数据集上效果优秀。VGG相较之前的卷积神经网络，拥有更小的卷积核与下采样核尺寸，从而减少了模型的参数量，并增加了模型的深度，以便提取更多有效特征。VGG16模型拥有13个卷积层与3个全连接层，并在卷积层和全连接层之间安置了一个自适应平均池化层，以控制全连接层的输入张量形状。以下是VGG模型的结构示意图：



1. **ResNet50模型**

**随着卷积神经网络模型深度的增加，学者发现，模型在训练集上的准确率会趋于饱和甚至下降。这并非过拟合所为，因为过拟合现象发生时，模型在训练集上的准确率应该进一步上升。这种深模型的准确率饱和现象被称为退化现象，其原因是在模型训练时，深模型中梯度的反向传播受阻，模型参数难以得到有效优化。**

**为解决退化现象对模型深度的问题，Kaiming He等人于2017年提出了残差神经网络ResNet。该网络在瓶颈卷积模块（Bottleneck）中引入了桥接（Shortcut）的概念，这个结构直接将卷积模块的输入与输出相加，为提供了梯度绕开卷积层继续反向传播的途径，从而有效缓解了深层网络难以优化的问题：**

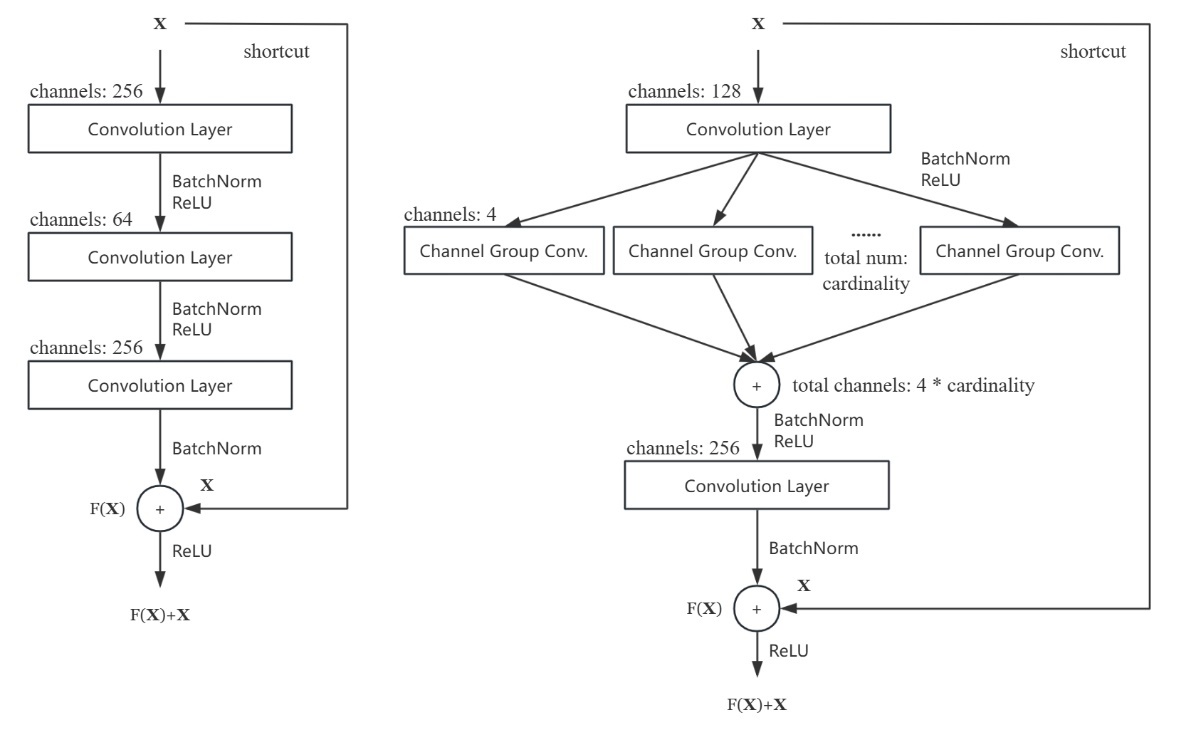
****

**ResNet的出现为卷积神经网络深度的增加扫清了障碍，使得现代卷积神经网络得以学习到更多特征。ResNet152模型在ImageNet数据集上的错误率仅为3.6%，这一比例甚至低于人类。ResNet50模型作为一种常用的残差模型，拥有4个残差块共计50个卷积层，其使用卷积核边长为7的卷积层作为感知层，并简化了全连接层与分类器的结构。**

**在搭建ResNet50模型时，我们注意到，该模型的感知层卷积核边长为7，移动步长为2，输出通道数直接从3膨胀至64；经过分析我们认为，感受野过大可能会造成感知层无法有效提取图像特征，通道膨胀过陡会导致特征丢失，因此我们将感知层替换为3层卷积核尺寸为3的卷积层，通过多级卷积缓解通道膨胀陡度，从而有效提取图像特征。**

1. **ResNeXt-CBAM模型**

**ResNeXt神经网络模型是ResNet模型的重要改进，其参数量与普通ResNet模型相当，但模型效果却优于ResNet，并在训练时更为高效。ResNeXt相较原型的最大改变，在于瓶颈卷积模块（Bottleneck）的中间卷积层。受Inception模型中多路径处理的启发，其引入基数（Cardinality）的概念，将该卷积层按基数分解为多个通道组，每个组独立进行卷积操作，且各组的结构完全一致：**

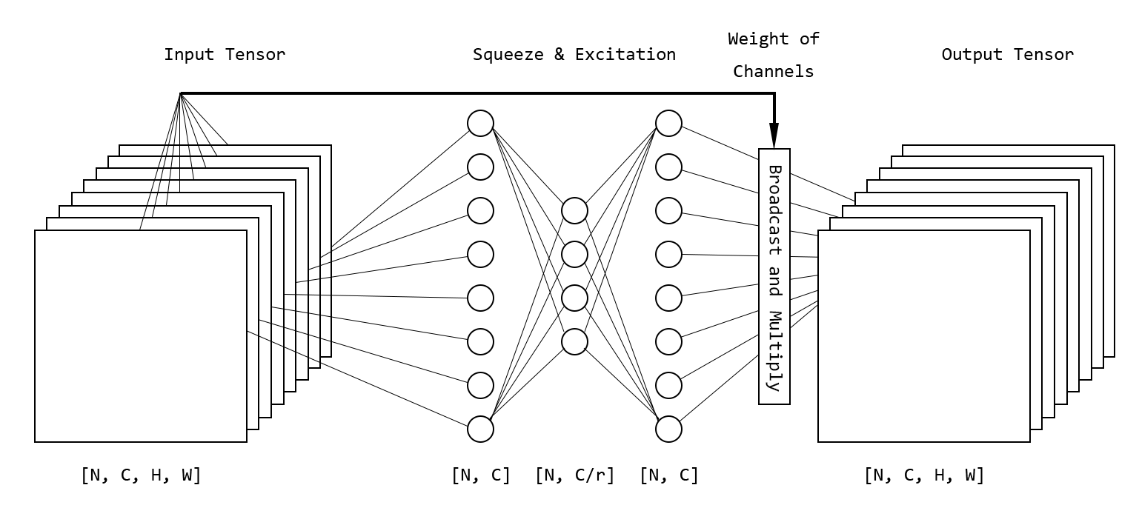
****

**在ResNeXt50的基础上，我们创新性地引入了自注意力机制（Self-Attention），即CBAM（Convolutional Block Attention Module）模块。自注意力机制旨在探寻输入信息内部各部分之间的相关性，并为各部分数据赋予对应的权重，并对信息进行加权求和之后送至下一层进行处理。自注意力机制使得模型可以为数据的各部分学习权重，从而使得模型将注意力放在主要信息上，忽略次要特征和噪声。**

**2017年Google Brain提出的Transformer架构，使得自注意力机制被广泛应用于自然语言处理（NLP）相关领域，而自注意力机制在计算机视觉任务中也有非凡的效果。在图像识别上，自注意力机制主要用于明确图像不同通道之间的权重关系，以及不同空间位置之间的权重关系；在CBAM模块中，上述功能由SEBlock（Squeeze & Excitation Block）和SABlock（Spatial Attention Block）两个子模块实现。**

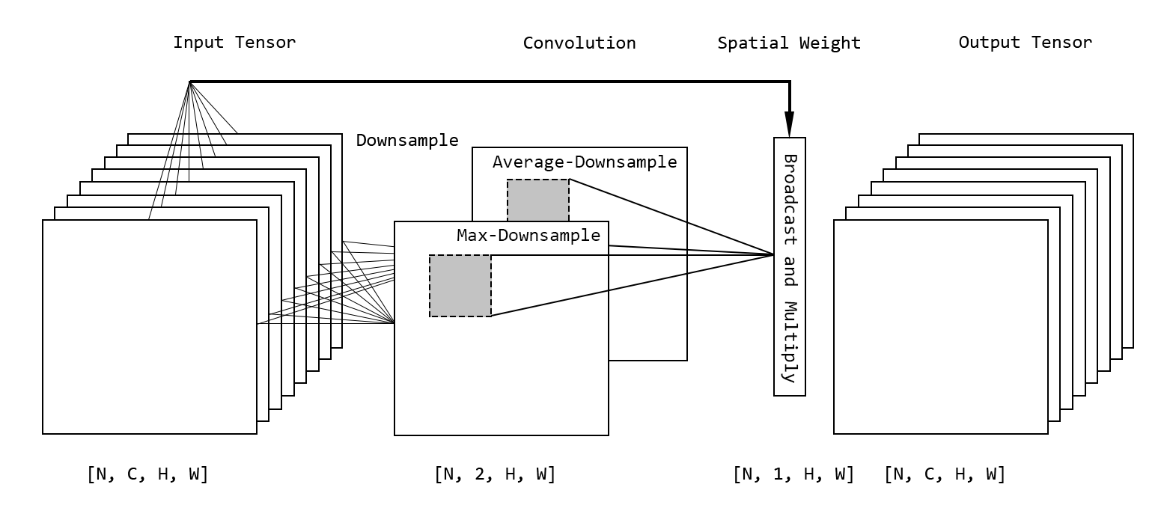
1. **SEBlock**

**SEBlock的主要目的是为每一个通道赋予一个注意力权值，并将通道内每个数据与对应权值的积作为输出结果。注意力权值通过一个线性全连接的挤压-激励（Squeeze & Excitation）单元进行学习，缩减比例（reduction ratio）决定了这个单元对信息的挤压程度。**

****

1. **SABlock**

**相比于SEBlock对通道注意力权重的学习，SABlock偏向于学习图像的空间注意力权重。这个模块首先将输入图像的每个像素分别进行跨通道最大下采样、平均下采样，从而将通道数降为2。对于这个张量再进行卷积操作，生成的单通道张量便是每个像素的注意力权值。将该权值与输入张量的每个通道按元素相乘即可得到输出。**

****

**以上是CBAM模块的基本构造。在CBAM-ResNeXt模型中，CBAM模块位于每一个瓶颈卷积模块的卷积层之后，整个模型一共有16个CBAM模块，有效提升了模型对主要特征的注意力，引导模型忽略次要特征和背景噪声。**

**（四）模型的训练与优化**

**本研究对CNN模型的实验分为三组。第一组为模型探究实验，用于测试VGG16、ResNet50、ResNeXt-CBAM三种模型的性能，并选取其中表现最好的模型做进一步优化；探究实验的训练轮数为25。第二组为模型超参数搜索实验，用于搜索能使模型损失快速收敛的超参数组合，每一次搜寻尝试会对模型进行10轮训练。第三组实验为模型优化实验，该实验用于在确定最优超参数组合后，对模型进行深度训练，该实验的训练轮数为50。灵活调整不同实验中的训练轮数，可以保证高效获得实验结果。**

1. **训练时学习率动态调整策略**

**在深度学习模型的训练过程中，若学习率始终保持不变，模型损失函数值会在一定轮数之后区域收敛，此时模型效果难以进一步提高。该问题的解决方案是动态调整学习率，使得模型参数趋于收敛后仍旧能进一步做局部优化。在本实验中，我们使用了学习率阶梯式衰减策略，即每训练10轮，学习率衰减为原来的0.1倍。**

1. **模型超参数初始值**

|  |  |  |
| --- | --- | --- |
| 超参数名称 | 变量名 | 初始值 |
| 初始学习率 | learning\_rate | 0.0001 |
| 每批次样本数 | batch\_size | 16 |
| Adam动量衰减系数 | beta1 | 0.9 |
| Adam二阶矩衰减系数 | beta2 | 0.999 |
| L2正则化系数 | l2\_lambda | 0.00001 |

1. **不同模型在探究实验中的表现**

**在对不同模型的训练中，我们发现，对模型的参数按一定方式进行初始化，可以有效加速模型收敛，并避免梯度消失或梯度爆炸现象的出现。研究表明，Kaiming初始化方案对于使用ReLU激活函数的模型较为有效。经过Kaiming初始化方案后，模型参数服从正态分布，其中是输入特征数。这样的参数组合保证了特征信息在经过ReLU激活函数前后方差基本一致，从而保证了特征在正向传播时的分布稳定。以下是VGG16、ResNet50、ResNeXt-CBAM三种模型在应用Kaiming初始化方案前后的训练曲线：**

**插入训练曲线，六个图，每个模型的损失、准确率各一个图**

**./training\_record文件夹下有三个模型文件夹，prototype是初始化之前，with\_weight\_init是初始化之后，.csv文件里是数据**

**对三种不同的CNN模型进行对比试验后，我们得到了这些模型在测试集上的准确率、混淆矩阵和Kappa系数：**

**插入训练结果图表，这个先不急**

**对上述图表数据进行分析，我们可以得出结论：ResNeXt-CBAM模型在当前任务上表现最好。VGG16模型由于没有引进残差学习方法，模型深度对模型训练造成了一定的阻碍，难以有效优化；另外，VGG16模型的参数量过大，模型复杂度高，有很大的过拟合风险。ResNet50模型相较于VGG16在效果上改善明显，但应对噪声和背景干扰的能力逊于ResNeXt-CBAM。由于引入了自注意力机制，ResNeXt-CBAM在当前任务中可以将注意力更多地放在叶片的形状与纹理上，对不同类别图像的特征提取能力更强，因此在当前任务上表现更优。在背景与目标叶片的区别较小、背景复杂的测试用例上，ResNeXt-CBAM模型的预测准确率也高于剩余两种模型。基于上述论证，我们选择ResNeXt-CBAM模型做后续实验，以对其进行进一步优化。**

1. **ResNeXt-CBAM模型超参数搜索实验**

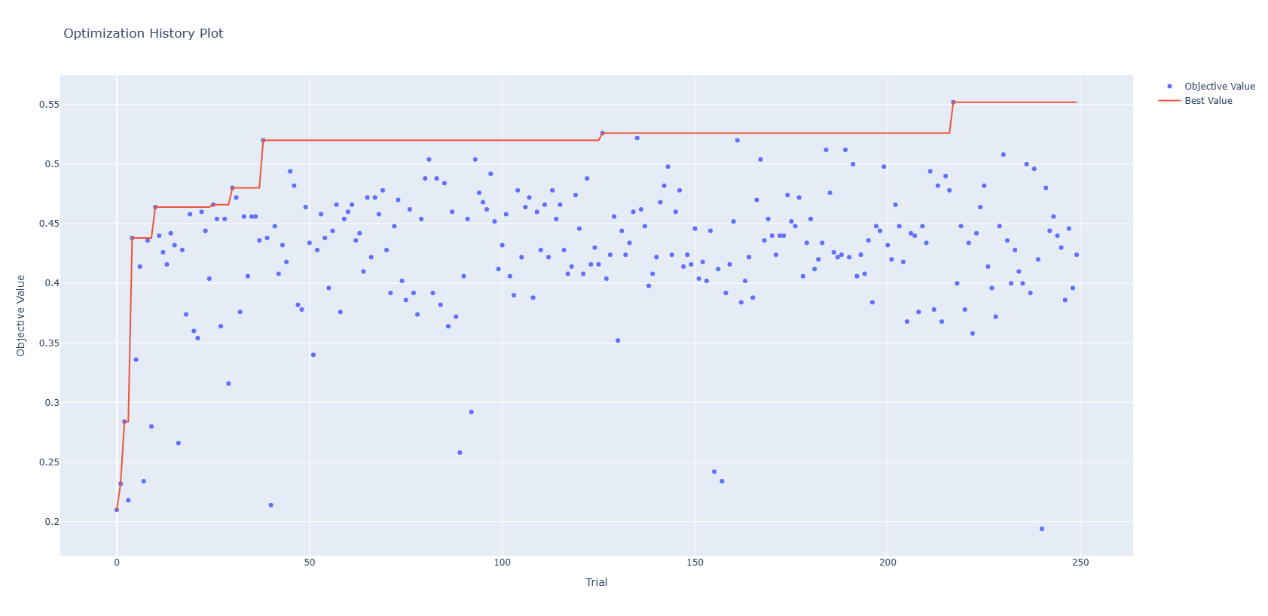
**在模型超参数搜索实验中，我们对比使用了SSA搜索算法、TPE搜索算法。下表为模型的待搜索超参数以及其取值范围：**

|  |  |  |  |
| --- | --- | --- | --- |
| 超参数名称 | 变量名 | 类型 | 取值范围 |
| 初始学习率 | learning\_rate | 连续型 |  |
| Adam动量衰减系数 | beta1 | 连续型 |  |
| Adam二阶矩衰减系数 | beta2 | 连续型 |  |
| L2正则化系数 | wd | 连续型 |  |
| dropout概率 | dropout\_rate | 连续型 |  |
| 基数 | cardinality | 离散型 |  |
| 瓶颈宽度 | base | 离散型 |  |
| SEBlock缩减比例 | SE\_reduction\_ratio | 离散型 |  |
| SABlock卷积核尺寸 | SA\_kernel\_size | 离散型 |  |

**通过两种搜索算法对比，我们选择TPE搜索算法对超参数进行优化。由上表可知，候选超参数有离散型，也有连续性，而TPE算法对于混合型搜索空间更为友好。另外，SSA算法在处理每一维取值范围窄、搜索空间维数较高时表现一般。**

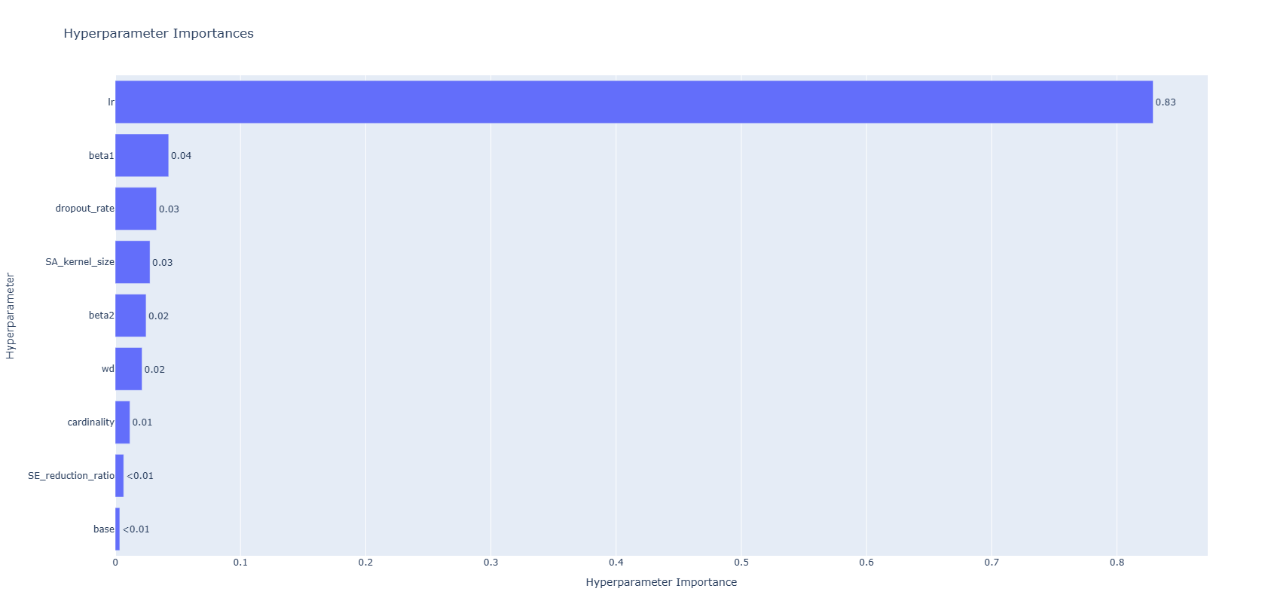
**在超参数搜索实验中，我们评估当前超参数优劣的方法是，以当前超参数在小数据集上对模型进行10次训练，并评估模型在测试集上的准确率。准确率越高，当前超参数的效果越好。**

**以下是使用TPE搜索算法对最优超参数组合进行搜索的过程：**

****

**分析搜索过程我们得知，TPE算法可以在开始迭代时较快地找到较优的超参数范围。在之后的迭代之中，算法在一定范围内随机采样，试图寻找到更优的超参数组合，并根据采样评估结果缩小下一次迭代时的搜索范围。**

**根据结果，我们同样能得到不同超参数对结果的影响程度：**

****

**很明显，初始学习率对训练结果的影响最大。**

**下表展示了TPE算法搜寻到的最优超参数组合：**

|  |  |  |  |
| --- | --- | --- | --- |
| 超参数名称 | 变量名 | 类型 | 最佳取值 |
| 初始学习率 | learning\_rate | 连续型 |  |
| Adam动量衰减系数 | beta1 | 连续型 |  |
| Adam二阶矩衰减系数 | beta2 | 连续型 |  |
| L2正则化系数 | wd | 连续型 |  |
| dropout概率 | dropout\_rate | 连续型 |  |
| 基数 | cardinality | 离散型 |  |
| 瓶颈宽度 | base | 离散型 |  |
| SEBlock缩减比例 | SE\_reduction\_ratio | 离散型 |  |
| SABlock卷积核尺寸 | SA\_kernel\_size | 离散型 |  |

1. **ResNeXt-CBAM模型优化实验与有效性验证**

**以下是ResNeXt-CBAM模型在50轮深度训练中的训练曲线与训练结果：**

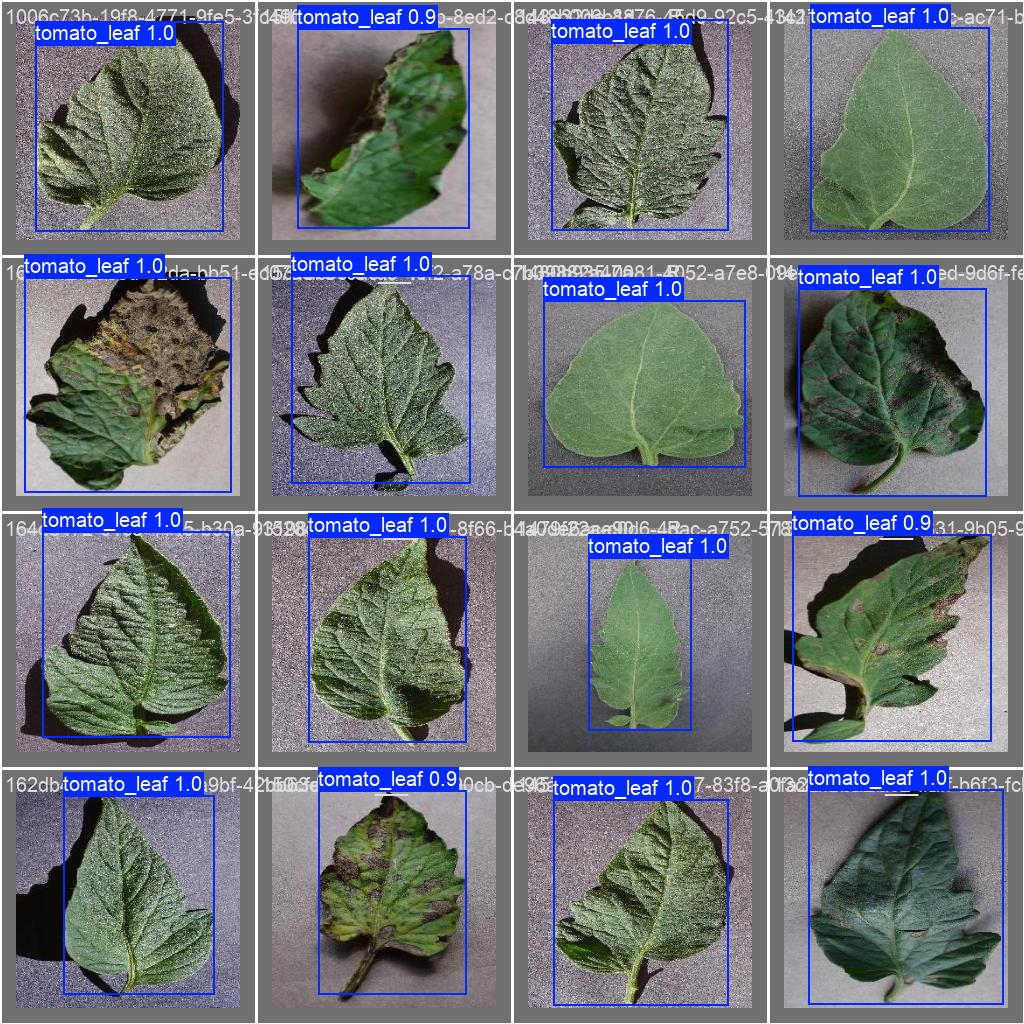
**插入训练曲线与训练结果**

**./training\_record/ResNeXt/after\_hyperparam\_optim中.csv文件**

**（五）模型的扩展与应用**

**为了使训练后的ResNeXt-CBAM能够在更为复杂的图像上同样表现优异，我们同时训练了一个YOLOv8-n模型，用于对图像中的番茄叶片目标进行检测。**

**YOLO是一种常用的图像目标检测模型架构，其基本工作原理是判断输入图像中是否存在目标对象，并使用矩形框紧确地框选出可能是目标的物体，并标注出该物体是目标的可能性。在本研究中，我们选择了1136张形态各异的番茄叶片图像，并手动框选出图像中的目标，并将图像与目标位置作为训练数据送入YOLOv8-n模型进行训练。在进行100轮训练之后，YOLOv8-n模型已经可以较为精确地框选出验证图像中的目标。下图展示了训练后的YOLO模型对一些验证图像中目标的检测情况：**

****

**通过ResNeXt-CBAM和YOLO模型的组合使用，研究得到的模型得以适用于更大的应用场景。以下是组合模型的基本工作原理图：**

**插入工作原理图，先不急**

**使用YOLOv8-n识别待预测图像中的目标并将其裁剪，将裁剪后的对象再送进ResNeXt-CBAM模型进行预测，可以有效排除ResNeXt-CBAM模型因背景图像干扰造成的错误，提高了整个模型预测的鲁棒性，使得模型可以在更广泛的应用场景工作。**

**事实上，YOLOv8模型自身同时拥有目标检测和图像识别的功能，但我们并没有直接使用YOLOv8对图像进行种类预测，而是对图像识别进行单独建模，使目标检测与图像识别功能解耦，这是考虑到模型功能的可拓展性。想要训练性能优秀的YOLO模型，需要大量带有目标选框的、类型齐全的训练数据，这些数据往往只能由人工标注，这无疑会提高整个模型训练的成本。而若将两个功能独立建模，对于YOLO模型而言，就无需关注训练数据的类型是否齐全，而用于分类的ResNeXt-CBAM模型也无需关注目标选框。当模型功能需要扩展时，我们仅需搜集新类型的图片并送入ResNeXt-CBAM训练，无需再对其进行人工目标选框。综上所述，使用目标检测模型、图像识别模型组合模型再扩展性上更胜一筹。**