

**2024年《机器学习》**

**工程报告**

**（完整版）**



**课 程：** 机器学习

**姓名+学号：王卫东（202217597），陈曦（2022217588），周作为（2022217590）**

**姓名+学号：魏海严（202217594），朱炳旭（2022217593），刘东林（2022217598）**

**完成时间：** 2024年5月19日

**一． 工程摘要与每人贡献**

摘要：总体介绍本工程的工作。

每人分工：

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 角色  （组长，组员，独自完成） | 工作量  比例 | 负责内容 |
| 王卫东 | 组长 | 19% | 完成工程报告（完整版）的撰写汇总；负责设计和实施数据增强策略；构建了一个包含注意力模块的CNN架构；实施了模型的训练和验证过程；帮助其他成员编写代码和训练模型 |
| 陈曦 | 组员 | 16% | 基于TensorFlow设计并训练了一个CNN模型 |
| 周作为 | 组员 | 16% | 基于TensorFlow设计了一个CNN模型；审核工程报告（完整版） |
| 魏海严 | 组员 | 18% | 数据增强与预处理；改进卷积神经网络架构；采用集成学习 |
| 朱炳旭 | 组员 | 16% | 通过PyTorch框架中的ResNet18模型，对血细胞进行分类任务 |
| 刘东林 | 组员 | 15% | 基于PyTorch的CNN模型进行了改进，帮助同学搭建环境、安装驱动 |

项目github链接：

<https://github.com/ModestWang/ML-Homework>

引用数据集的链接:

<https://www.kaggle.com/datasets/paultimothymooney/blood-cells/data>

**二、研究背景与意义**

**2.1 研究背景**

血细胞分类是医学领域中的一个重要任务，特别是在血液病、感染和炎症等疾病的诊断中，准确识别血细胞的亚型具有重要意义。传统的血细胞分类依赖于显微镜检查和人工分析，这不仅耗时耗力，还可能受到人为因素的影响，导致结果的不一致性和准确性不足。

**2.2 研究意义**

通过计算机技术实现的血液疾病诊断方法通常是基于图像处理和机器学习技术。这些方法利用数字图像分析来识别和分类血细胞，从而帮助医生进行诊断。这个项目主要涉及医学图像处理和机器学习领域。在这个领域，主要研究方法包括图像分割、特征提取、特征选择、分类器设计等。医学图像处理技术用于处理血液样本的数字图像，提取其中的有用信息；机器学习算法则用于训练分类模型，实现自动化的血液细胞分类和诊断。

**三、模型方法**

本工程中，组内每名成员分别编写了一份代码。主要基于卷积神经网络，一部分成员使用Pytorch构建模型，另一部分使用TensorFlow实现。下面为模型方法汇总。

* 1. **王卫东 的模型**

**3.1.1 模型架构**

卷积层：CNN的核心组成部分，它通过滤波器（或称为卷积核）在图像上滑动，提取局部特征。每个滤波器都能够捕捉到图像的不同特征，例如边缘、角点或更复杂的纹理。

批量归一化层：这些层在训练过程中对每个小批量数据的激活输出进行归一化处理，有助于加速训练过程，提高模型的稳定性。

注意力模块：注意力机制可以使模型在处理图像时更加关注于图像的重要部分，从而提高分类的准确性。在本模型中，注意力模块通过一个卷积层和Sigmoid激活函数来实现，它为输入特征图的每个位置生成一个注意力权重。

池化层：池化层用于降低特征图的空间维度，减少参数数量，从而减轻过拟合并减少计算量。

全连接层：在经过一系列卷积和池化操作后，特征图被展平，并通过全连接层进行分类。全连接层的作用是将学习到的局部特征组合起来，进行最终的分类判断。

**3.1.2 训练过程**

在训练过程中，模型使用交叉熵损失函数（CrossEntropyLoss）和Adam优化器。交叉熵损失函数是分类任务中常用的损失函数，它衡量的是模型输出的概率分布与真实标签的概率分布之间的差异。Adam优化器是一种自适应学习率的优化算法，它结合了动量法和RMSProp算法的优点，能够在不同的参数上使用不同的学习率，通常能够快速收敛。

**3.1.3 评估指标**

模型的性能通过准确率（Accuracy）、F1分数和AUC分数来评估。准确率是最直观的性能指标，它表示模型正确分类的样本占总样本的比例。F1分数是精确率和召回率的调和平均，它在数据不平衡时是一个很好的性能指标。AUC分数衡量的是模型对于正负样本排序能力的一个指标，它的值越接近1，表示模型的性能越好。

**3.1.4 可视化**

为了更好地理解模型的学习过程和性能，我们使用了混淆矩阵和损失曲线图进行可视化。混淆矩阵显示了模型在各个类别上的预测结果，而损失曲线图则展示了训练和验证过程中损失的变化，有助于我们监控模型是否出现过拟合或欠拟合。

**3.2 陈曦 的模型**

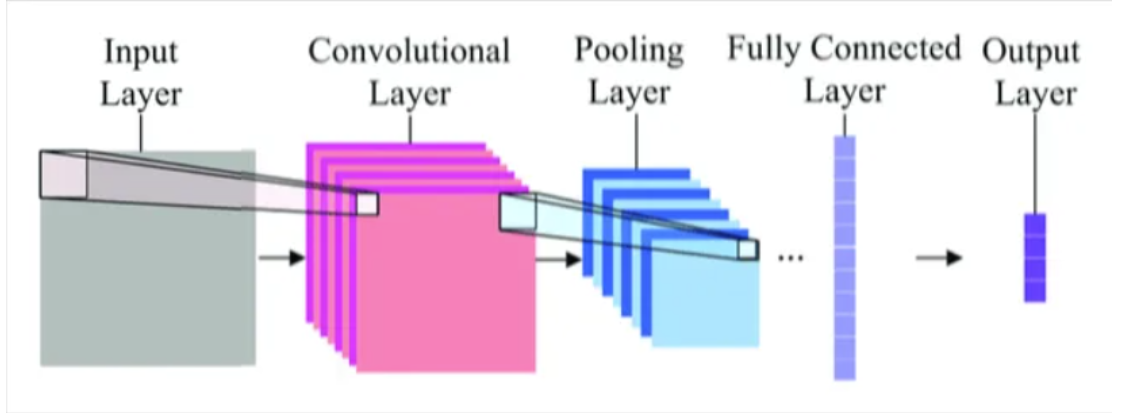
**3.2.1 卷积神经网络（CNN）**

卷积神经网络（Convolutional Neural Network, CNN） 是一种深度学习模型，特别适用于处理图像数据。CNN通过局部连接和权重共享的方式来提取图像的局部特征，具有强大的特征提取能力。它主要由以下几个部分组成：

卷积层（Convolutional Layer）：通过卷积运算提取图像的局部特征，通常使用多个卷积核（filter）来学习不同的特征。

池化层（Pooling Layer）：对卷积层提取的特征进行降维，减小特征图的尺寸，降低计算复杂度，并在一定程度上实现平移不变性。常见的池化操作包括最大池化（Max Pooling）和平均池化（Average Pooling）。

全连接层（Fully Connected Layer）：将前面层提取的特征映射到最终的输出类别。通过全连接层可以实现分类任务。



**3.2.2 模型结构展示**



**3.3 周作为 的模型**

**3.3.1 模型介绍**

这个Sequential模型定义了一个卷积神经网络，用于处理图像分类任务。

卷积层：用于提取图像中的特征。

批量归一化层：对神经网络的每一层输出进行标准化处理，使得每一层的输出具有相近的分布。这有助于加速训练过程、提高模型性能，并减少对初始化权重的敏感性。

最大池化层：对特征图进行下采样，减少空间维度（高度和宽度），同时保留最重要的特征。这有助于减少计算量，并增加感受野的大小。

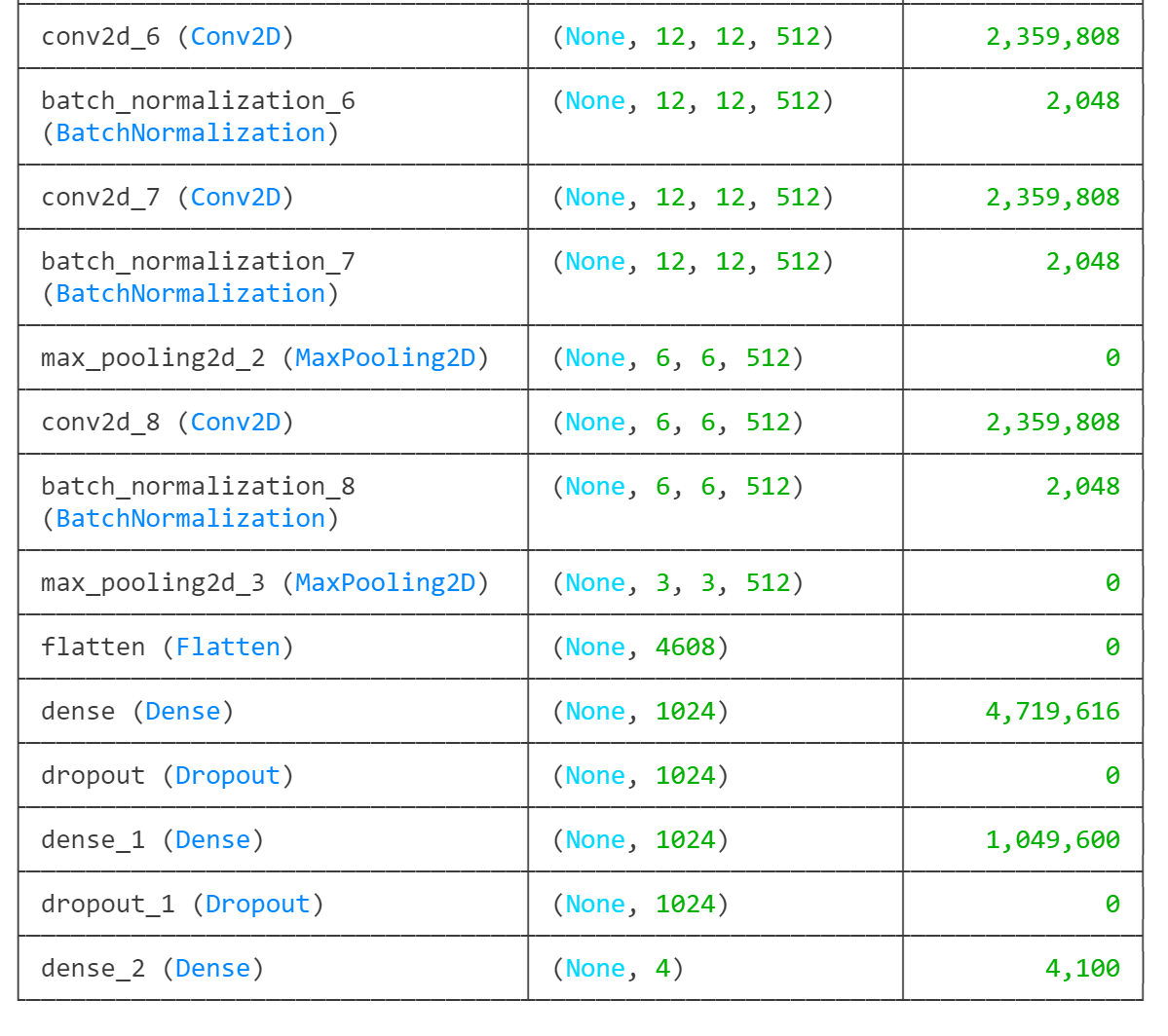
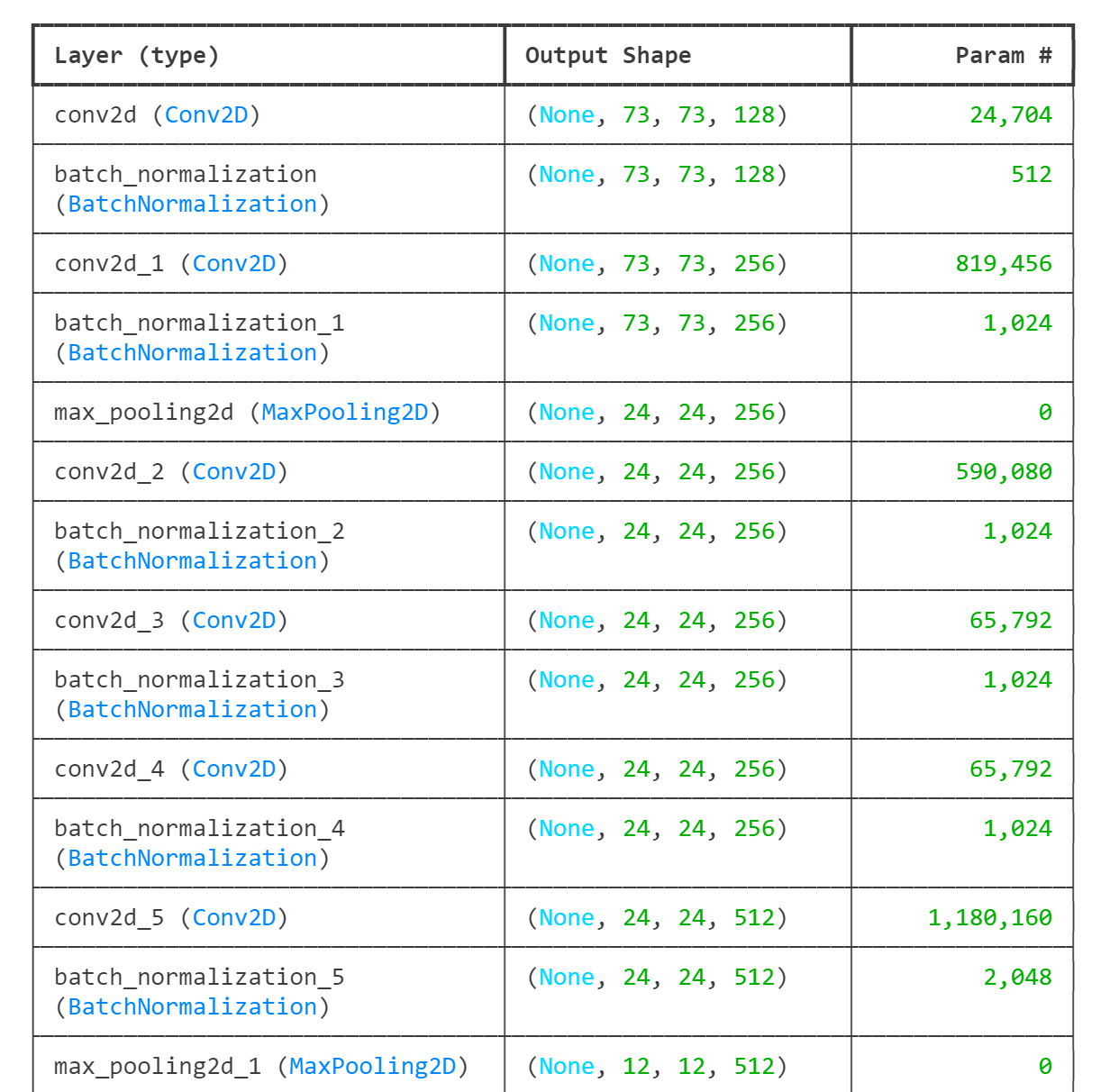
Flatten层：将多维的输入一维化，常用于从卷积层到全连接层的过渡。

全连接层：在Flatten层之后，全连接层用于整合前面提取的特征，并输出最终的分类结果。通常使用softmax作为激活函数。

Dropout层：在训练过程中，随机将网络中的一部分神经元“丢弃”掉，以减少过拟合，定义了丢弃神经元的比例为50%。

使用model.fit()方法来训练模型，传入训练数据、验证数据、批次大小、训练轮数等参数。

**3.3.2 模型结构展示**



**3.4 魏海严 的模型**

在本工程中，采用了深度学习中的卷积神经网络（CNN）及其改进形式来实现血细胞图像的分类任务。主要使用的方法和理论如下：

**3.4.1 数据增强（Data Augmentation）**

通过对训练数据进行随机翻转、旋转、颜色抖动等操作，增加数据的多样性，提高模型的泛化能力。

**3.4.2 卷积神经网络（CNN）**

CNN是一种专门处理图像数据的深度学习模型，通过卷积层提取图像的空间特征，通过池化层减少特征图的维度，同时保留关键信息。

**3.4.3 残差网络（Residual Networks，ResNet）**

引入残差模块，通过跳跃连接（skip connections）来缓解深层网络的梯度消失问题，使得更深层的网络可以更有效地训练。

**3.4.4 模型的系统介绍：**

设计的模型是一个改进的卷积神经网络（CNN），用于图像分类任务。首先，模型接受输入图像（RGB格式，即3个通道），然后通过一系列的卷积、池化和全连接层对图像进行处理，最终输出图像的分类结果。

模型的主体结构包括卷积层、批归一化层、残差块和全连接层。卷积层用于提取图像的特征，批归一化层有助于加速模型训练并提高模型的泛化能力。残差块是模型的核心组件，通过残差连接（residual connection）可以解决深度神经网络中的梯度消失和梯度爆炸问题，从而使网络更易于训练和优化。

在残差块中，首先通过一个卷积层和批归一化层对输入进行处理，然后再经过另一个卷积层和批归一化层处理。同时，利用残差连接将输入直接加到最后的输出上，以确保信息的流动。这样的设计使得模型能够更好地捕捉图像中的细节信息，并且具有更好的训练效果。

模型的结构采用了多层的残差块，并在每个残差块之间插入了最大池化层，以逐渐减小特征图的尺寸。最后，通过全连接层将特征图展开成一维向量，并经过几层全连接层进行分类，最终输出图像的类别。

此外，模型还包括了dropout层，用于在训练过程中随机丢弃部分神经元，以防止过拟合。整个模型的设计旨在提高图像分类的准确性和泛化能力，使其适用于各种不同的图像分类任务。

**3.4.5 集成学习（Ensemble Learning）**

训练多个独立的CNN模型，并将它们的预测结果进行综合，以提高分类的准确性和稳定性。

**3.4.6 优化算法**

使用Adam优化算法，结合学习率调度器（learning rate scheduler）动态调整学习率，加速模型收敛并避免过拟合。

损失函数：使用交叉熵损失函数（Cross Entropy Loss）来衡量模型预测与真实标签之间的差异。

**3.5 朱炳旭 的模型**

**3.5.1 使用ResNet18模型和权重的神经网络模型**

首先导入了必要的库和ResNet18\_Weights。然后定义了一个名为Net的类，继承自nn.Module。在初始化函数中，首先实例化了一个ResNet18模型，并根据参数决定是否冻结模型。接着获取了模型最后一层全连接层的输入特征数，并将全连接层替换为一个输出维度为4的线性层。接着定义了一个冻结模型参数的方法freeze\_resnet，通过将模型参数的requires\_grad属性设置为False来实现。最后定义了forward方法，将输入x传入ResNet模型中并返回输出。

**3.6 刘东林 的模型**

**3.6.1 使用卷积神经网络（CNN）**

模型有6个卷积层、6个池化层、3个Dropout层、5个全连接层。每个卷积层和全连接层后加ReLU激活函数。

输入为240 × 240 × 3，经过第一个卷积层后240 × 240 × 32，经过第一个池化层后120 × 120 × 32；经过第二个卷积层后120 × 120 × 64，经过第二个池化层后60 × 60 × 64；经过第三个卷积层后60 × 60 × 128，经过第三个池化层后30 × 30 × 128；经过第四个卷积层后30 × 30 × 256，经过第四个池化层后15 × 15 × 256；经过第五个卷积层后15 × 15 × 512，经过第五个池化层后7 × 7 × 512；经过第六个卷积层后7 × 7 × 1024，经过第六个池化层后3 × 3 × 1024。

在第二个、第四个、第六个池化层后加一层随机dropout。

**四、系统设计**

受篇幅限制，这里我们选取其中比较有代表性的两个模型进行说明（其余模型详见各成员的个人报告），分别是王卫东和魏海严设计的模型，也是开始时使用和后面改进的模型。

**4.1 王卫东 的设计**

**4.1.1. 导入部分常用库**

numpy 和 torch 是进行科学计算和深度学习的基础库。

torch\_directml 是用于在不支持CUDA的设备上运行PyTorch代码的库。

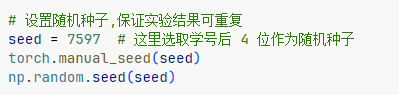
torch.nn 包含了构建神经网络所需的模块和损失函数。

DataLoader 和 random\_split 用于数据加载和分割。

transforms 和 datasets 用于数据预处理和加载。

time 用于获取时间

**4.1.2. 设置随机种子**



将随机种子设置为学号后四位，用于torch和numpy。随机种子确保了实验的可重复性，使得每次运行代码时初始化的权重和数据分割都保持一致。

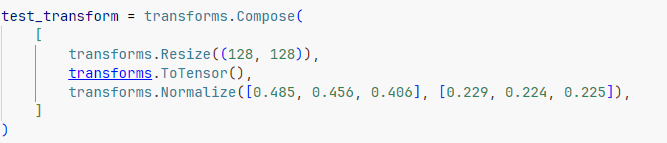
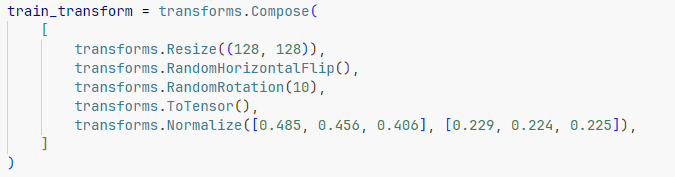
**4.1.3. 数据预处理**

用train\_transform 和 test\_transform 定义了图像的预处理步骤。

在PyTorch的transforms模块中，Normalize和Resize是两种常用的数据预处理操作。它们的作用如下：

Resize((128, 128))：这个操作会将图像的大小调整为128x128像素。图像尺寸的统一对于神经网络的输入是必要的，因为网络需要固定大小的输入张量。此外，调整图像大小也可以减少模型的计算负担，特别是当原始图像非常大时。

Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])：这个操作会对图像进行标准化处理。给定的两个数组分别代表了RGB通道的均值和标准差。通过这种方式，每个通道的像素值会减去均值并除以标准差。这样做的目的是将图像数据规范化到一个标准的分布中，通常有助于模型的训练效率和性能。

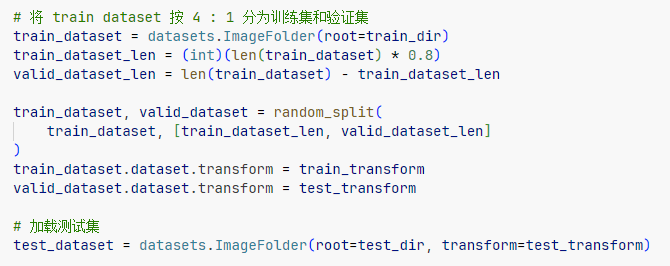


在train\_transform中添加了RandomHorizontalFlip()和RandomRotation()。这是一种数据增强技术，用于训练深度学习模型时增加数据的多样性。

RandomHorizontalFlip()：这个变换会以一定的概率（默认为0.5，即50%）水平翻转图像。这意味着图像有一半的机会被镜像反转。这种数据增强可以帮助模型学习到不依赖于水平方向的特征，从而提高模型对于水平位置变化的鲁棒性。

RandomRotation(10)：这个变换会随机地将图像旋转一个在[-10, 10]度的角度范围内的值。这样的随机旋转可以使模型对于图像的轻微旋转变化更加不敏感，增强模型对于旋转变化的泛化能力。

**4.1.4. 数据集分割**



首先指定了训练集和测试集的路径，然后使用使用 random\_split 将训练数据集按照 4 : 1 分为训练集和验证集，用于交叉验证。

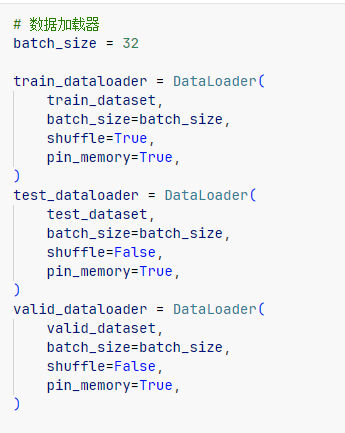
将数据集分割为测试集和验证集有几个好处，这是机器学习和深度学习实践中的一个常见做法。验证集可以用来评估模型的性能，而不会影响最终的测试集结果。这有助于我们了解模型在未见过的数据上的表现。如果我们有多个模型或者多种算法，验证集可以帮助我们选择表现最好的模型。

**4.1.5. 数据加载器**

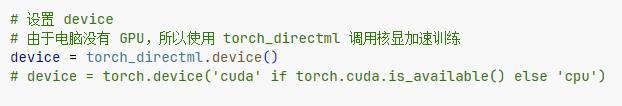
DataLoader 用于创建可迭代的数据加载器，以批量方式加载数据。

对于训练数据，我将DataLoader的shuffle参数设置为 true，这样会对数据进行洗牌，提高模型训练的泛化能力。同时我设置pin\_memory参数为True，将数据保存在显存（虽然使用的是核显）上，减少数据传输过程中的时间损耗，提高训练的速度。

在深度学习中，批量大小（batch size）是一个重要的超参数，它指的是每次训练过程中网络一次处理的数据样本数量。这里，我将批量大小（batch\_size）设置为32。



**4.1.6. 训练设备设置**



由于没有GPU，代码使用 torch\_directml.device() 来使用集成显卡进行训练。

**4.1.7. 定义模型**



**注意力机制模块**：AttentionModule 是一个注意力机制模块，它通过1x1卷积和Sigmoid函数生成注意力图。通过引入注意力机制，网络能够更加关注图像中的重要部分，从而提高特征的表达能力和网络的分类性能。



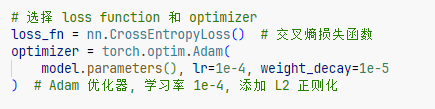
**卷积神经网络**：CNN 是一个卷积神经网络，包含多个卷积层、批量归一化层、池化层、全连接层和Dropout层。CNN 是深度学习中的一种强大的模型，特别适用于图像和视频识别任务。

批量归一化：加速网络训练，提高模型的泛化能力。

最大池化：减少特征维度，降低计算复杂度，同时保留重要特征。

全连接层和 dropout ：将学习到的特征转换为分类结果，dropout 用于减少过拟合。

**4.1.8. 损失函数和优化器**



**损失函数**：使用常用的交叉熵损失函数。

**优化器**：使用Adam优化器，设置学习率为0.0001，并添加L2正则化，正则化系数设置为0.00001，以避免过拟合现象。

**4.1.9. 训练和验证过程**

在多个epoch内进行模型的训练和验证。在每个epoch结束时，输出训练损失、验证损失、准确率和F1分数。

当验证损失降低时，保存模型的状态。

**4.2 魏海严 的设计**

**4.2.1 数据处理流程**

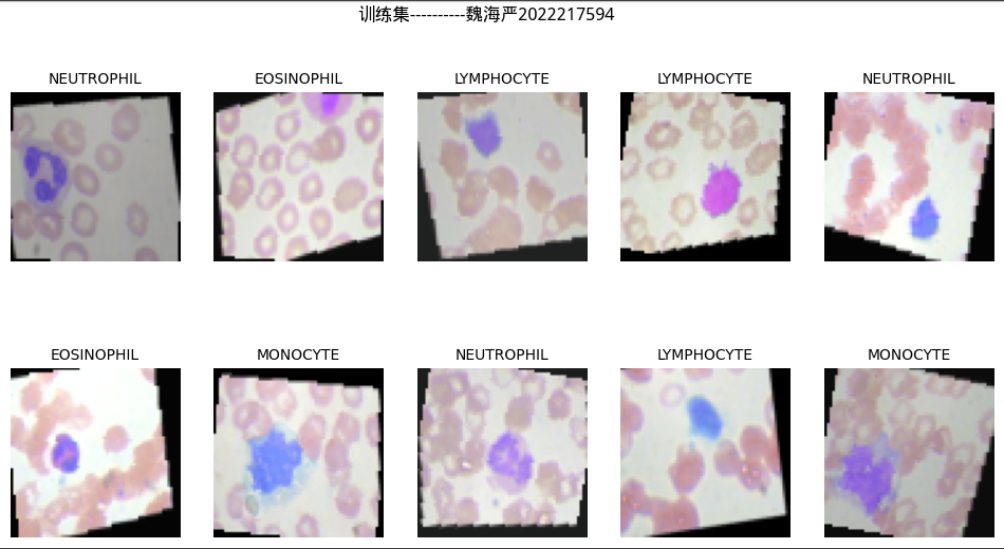
数据加载：使用torchvision.datasets.ImageFolder加载训练集和测试集。

数据增强：使用torchvision.transforms对训练数据进行增强处理，如调整大小、随机翻转、旋转和颜色抖动等。

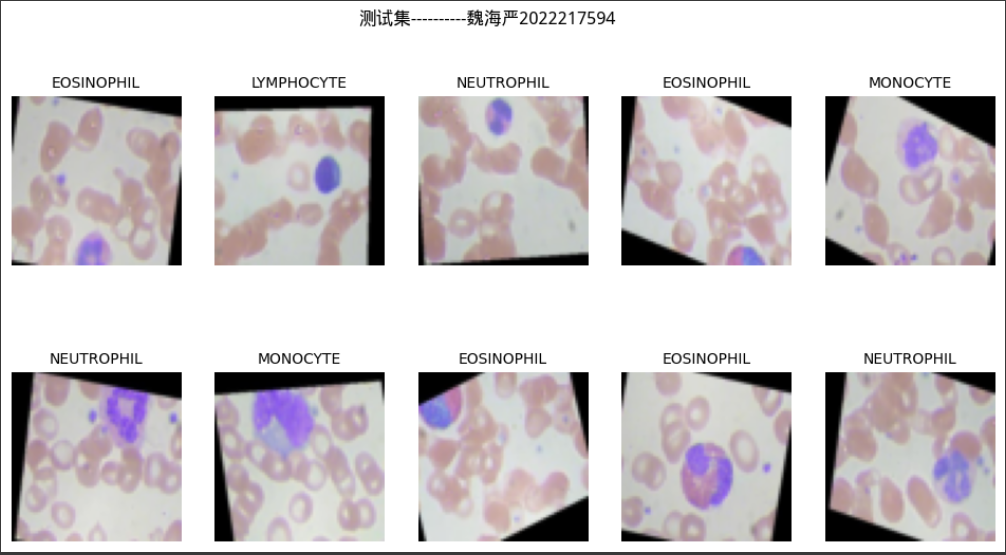
标准化：对图像数据进行标准化处理，使得图像数据在同一尺度上进行训练。

**4.2.2 查看数据：**

浏览训练数据集和测试数据集中各个类相关数据，随机选取部分图像，将图像输出，分别显示训练集和测试集的图像。在浏览数据之前，需要对数据进行逆标准化处理



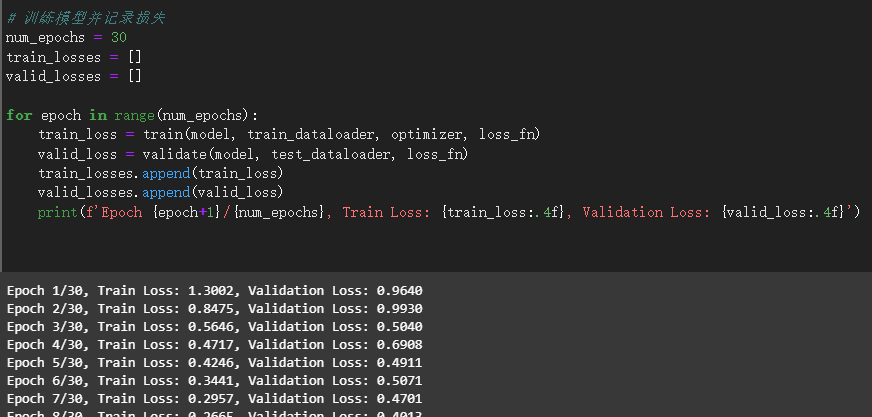
从给定的数据集中随机采样图像，并以网格形式显示这些图像



**4.2.3 模型训练流程**

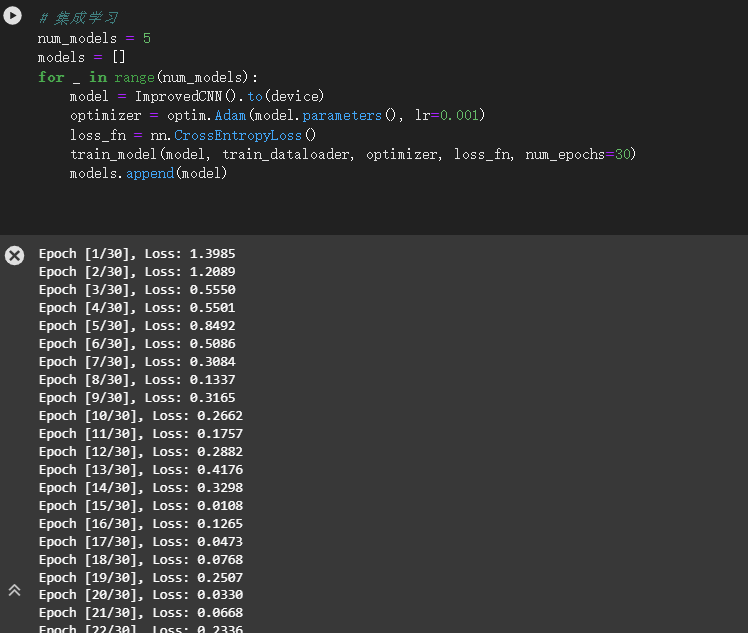
**模型定义：** 定义改进的卷积神经网络（ImprovedCNN），包括卷积层、池化层、残差模块、全连接层和Dropout层。

**单独模型训练**： 使用定义好的模型、优化器（Adam）和损失函数（交叉熵损失）对数据进行训练，设置学习率调度器来动态调整学习率。



**集成模型训练**： 训练多个独立的CNN模型，以构建集成模型。

**4.2.4 模型测试流**



上图为5个学习器的训练损失

**评估集成模型：** 对测试数据集进行预测，通过平均多个模型的预测结果来得到最终的分类结果。(评估指标： 使用准确率（accuracy）、F1分数（F1 score）和ROC AUC分数（AUC score）来评估模型性能。采用**混淆矩阵**评估模型性能。将单独（非集成）模型的性能与集成模型的性能做比较。)

**4.2.5 单张图片预测流程：**

图像预处理： 对单张图片进行预处理，包括调整大小和标准化。

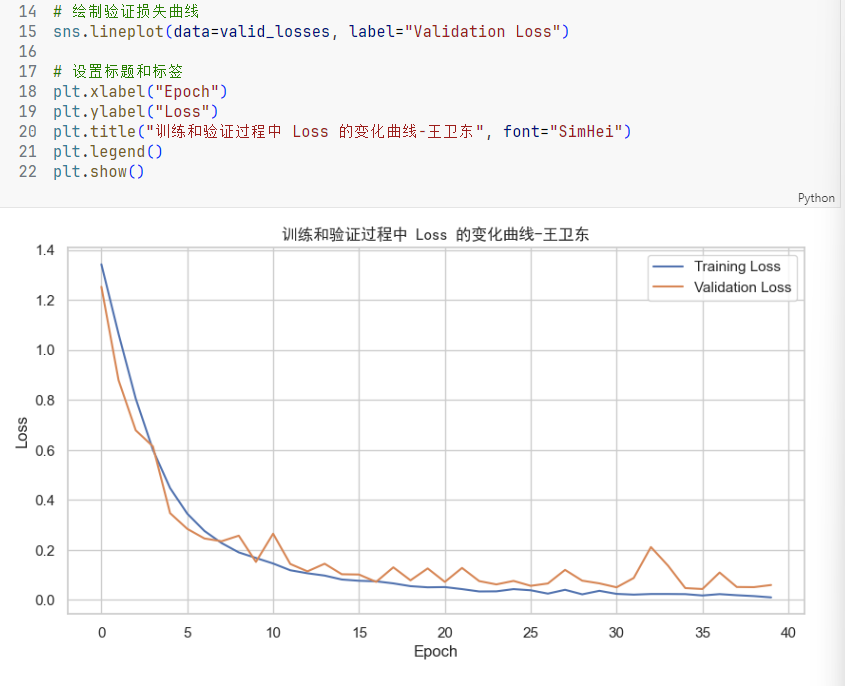
预测： 使用集成模型对预处理后的单张图片进行预测，得到概率分布和最终分类结果。

**五．实验结果分析、对比和讨论**

**5.1 开始时采用的模型得到的结果**

**5.1.1 训练结果可视化**

使用 matplotlib 和 seaborn 绘制损失和准确率的变化曲线。



损失下降：训练损失和验证损失都随着训练周期的增加而稳定下降，这表明模型正在学习并且改进其预测能力。

准确率和F1分数的提升：准确率和F1分数随着训练的进行而显著提高，这表明模型在分类任务上的性能得到了提升。

模型保存点：每当验证损失达到新低时，模型就会保存，它允许我们恢复到最佳性能的模型状态。

**5.1.2 测试集上进行模型评估**

PyTorch的torch.load方法加载之前训练训练过程中得到的的最优模型。

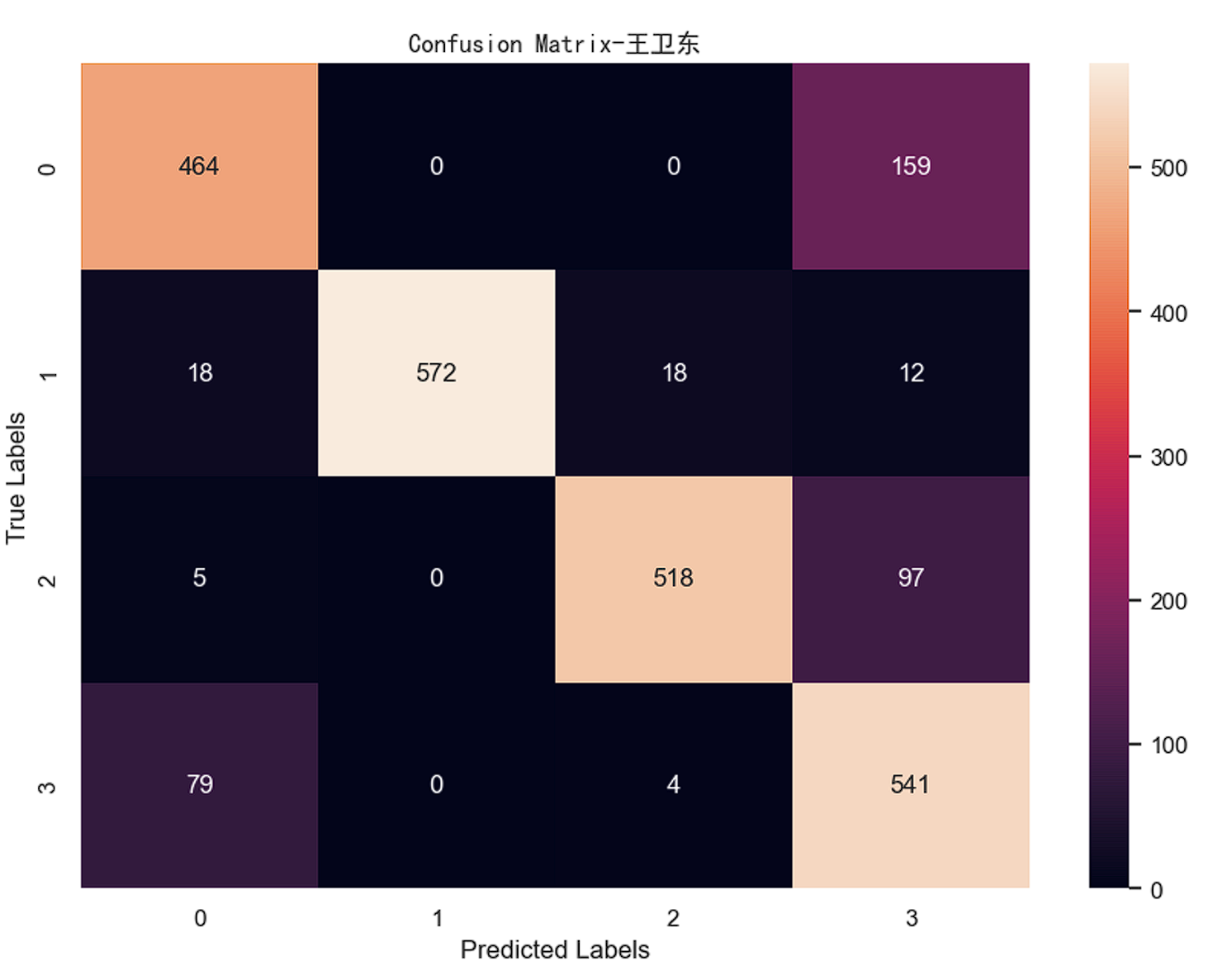
通过调用eval()方法，确保模型在评估模式下运行，这在进行模型推理时是必要的，因为它会关闭模型中的dropout和batch normalization等训练时特有的操作。

迭代测试数据加载器test\_dataloader，将数据移动到指定的设备（如GPU），然后通过模型获取logits。使用torch.softmax计算概率，并将真实标签和预测概率添加到之前初始化的列表中。

计算了模型的准确率、F1分数和AUC分数。np.argmax(pred\_probs, axis=1)用于从预测概率中获取最可能的类别标签。



混淆矩阵是一个重要的工具，可以帮助理解模型在各个类别上的预测性能。



测试准确率：测试准确率为0.8424，这意味着模型能够正确分类约84%的测试样本。对于本次的任务而言，并没有达到预期，模型仍有改进空间。

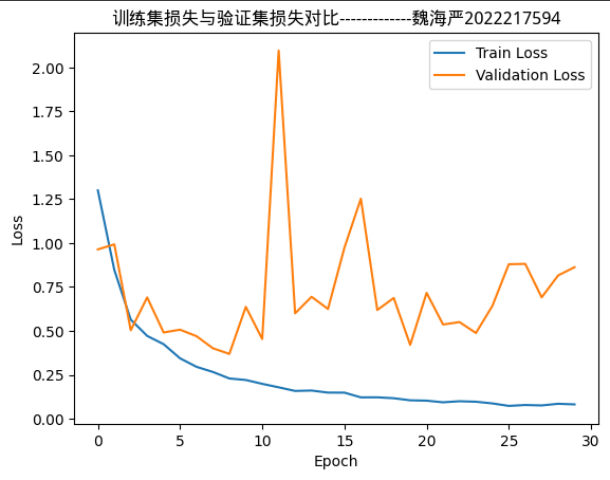
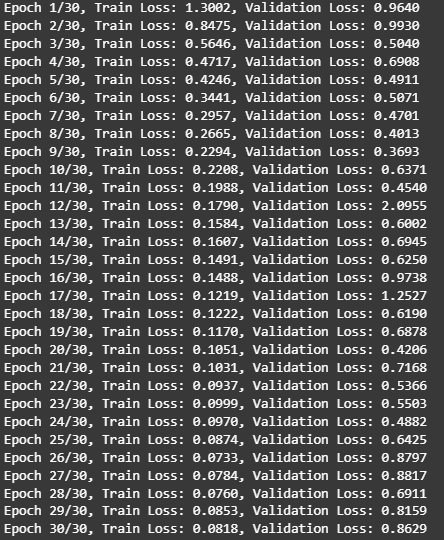
测试F1分数：F1分数为0.8471，这是一个相对较高的分数，表明模型在精确率和召回率之间取得了良好的平衡。

测试AUC分数：AUC分数为0.9670，这是一个比较高的分数，表明模型在区分不同类别方面表现出色。

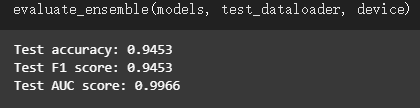
**5.2 改进后的模型得到的结果**

**5.2.1 实验结果分析**

模型训练过程中的损失变化：在训练过程中，通过记录每个epoch的损失值，可以观察到模型的收敛情况。随着训练进行，损失值逐渐减小，表明模型在不断学习和优化。

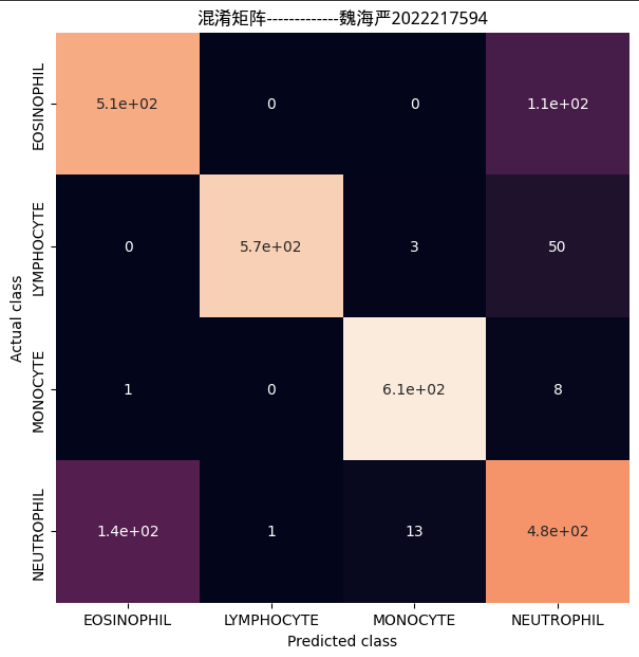


评估结果：在测试数据集上，通过准确率、F1分数和AUC分数来评估模型性能。实验结果显示，集成模型在各项指标上均表现出较单独模型更高的性能，证明了集成学习方法的有效性。



可以看出，集成模型有很高的准确率，F1值和AUC分值，模型效果好。

通过混淆矩阵可以看出，对角线元素占比很高，非对角线方格数据多数趋近于0，说明模型预测精确。



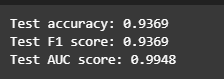
在单张图片预测任务上，可以看到模型预测正确



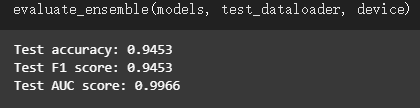
**6.2.2 单模型与集成模型的对比**

集成模型的准确率、F1分数和AUC分数均优于单个模型的结果，表明集成模型通过结合多个模型的优势，能够提高整体性能和鲁棒性。

**单个模型分类指标值**：



**集成模型分类指标值：**



可以看到集成模型性能略有提升。

不同超参数设置的对比：对学习率、批次大小、迭代次数等超参数进行调整，通过实验发现适当的超参数设置能够显著影响模型的训练效果和最终性能。其中通过多次实验发现epoch为30时，学习率取0.001时，batch size = 32时最合适。

**5.3 其他改进想法**

**数据增强：**考虑更多的数据增强技术，如颜色抖动、尺度变换等，以进一步提高模型对不同图像条件的鲁棒性。

**超参数调整：**通过系统地调整学习率、批量大小等超参数，可能会进一步提高模型性能。

**注意力机制改进：**研究更先进的注意力机制，如Transformer模型中的自注意力，以提高模型对关键特征的捕捉能力。

**模型结构：**探索更深或更复杂的网络结构，以提高模型的学习能力。

**六．其他**

在本项目的研究过程中，通过深度学习和集成学习方法对血细胞图像分类进行了深入的探索和实验。这不仅提升了我们的技术能力，也使我们深刻理解了计算机视觉技术在医学诊断中的重要应用。特别是通过实际数据的处理、模型的设计和优化、以及实验结果的分析，我们进一步掌握了如何将理论知识应用于实际问题解决中的技能。

在实际应用方面，自动化血细胞分类技术可以显著提高医疗诊断的效率和准确性，为医生提供有效的辅助工具，减轻工作负担。此外，这项技术还可以应用于其他图像分类任务，如癌细胞检测、病理图像分析等，为医学影像领域的发展提供更多可能性。

**七．对本门课的感想、意见和建议**

**7.1感想：**

《机器学习》这门课让我学到了很多机器学习的相关知识，让我学习到了很多机器学习的算法，同时让我意识到了机器学习在生活中发挥的重要作用，这门课让我对人工智能和机器学习有了一个更加深入的了解。老师在上机器学习课都能让我们很直观的理解他所讲的知识，并且老师讲课的风格很有趣。课程的编程作业以及大作业给我们一个很好的练习平台，加深了我对机器学习的理解。

**7.2意见和建议：**

机器学习领域发展迅速，课程内容应定期更新和扩展，以反映最新的研究进展和技术趋势。可以在课程中引入一些前沿技术和最新研究成果，帮助学生了解该领域的发展趋势和未来方向，激发学生的学习兴趣和创新思维。

此外，可以提供丰富的学习资源帮助学生多渠道学习和掌握知识。提供交互式学习资源，如在线编程环境、实验室和模拟工具。这样的资源可以帮助学生直观地理解算法和技术，并在实践中获得反馈。通过实际项目驱动学习，设立从简单到复杂的项目任务，帮助学生逐步掌握机器学习的应用方法。

还可以定期对学生的学习进度和理解情况进行评估，并提供及时反馈和建议。这有助于学生发现和解决问题，并及时调整学习策略。

最后，我建议学校能够增加《机器学习》课程的时长，如果能给《机器学习》这门课安排一些实验课，我认为我们会有更多的收获。