**基于VGG-16的医学成像诊断检测**

​

# 1：作业简介：医学成像诊断检测

## ****1.1问题描述：****

随着 CT 和 MRI 技术的提高，医生需要查看的医学影像极大地增加。通过计算机视觉，采用人工智能的诊断成像检测可以提高临床医生的工作效率，增强成像解释，并协助异常检测、鉴别诊断和工作列表优先级排序。

## ****1.2预期解决方案：****

我们期待您参考英特尔的类似实现方案，基于我们提供的胸部 X 光图像数据，训练一个神经网络模型，有效诊断医学影像图片是否为肺炎患者——这里推理时间和二分类准确度（F1分数）将作为评分的主要依据。

## ****1.3数据集****

<https://filerepo.idzcn.com/dataset/assignment_2.zip>

链接：<https://pan.baidu.com/s/1KNdSIwQHiDrJLT-5K-sPmA>

提取码：fly8

## 1.4图像展示

# 2：数据预处理

## 2.1数据集结构

本项目数据集共由三部分组成，分别包含为test，train，val文件夹。

图形用户界面, 应用程序

描述已自动生成​

数据集结构 

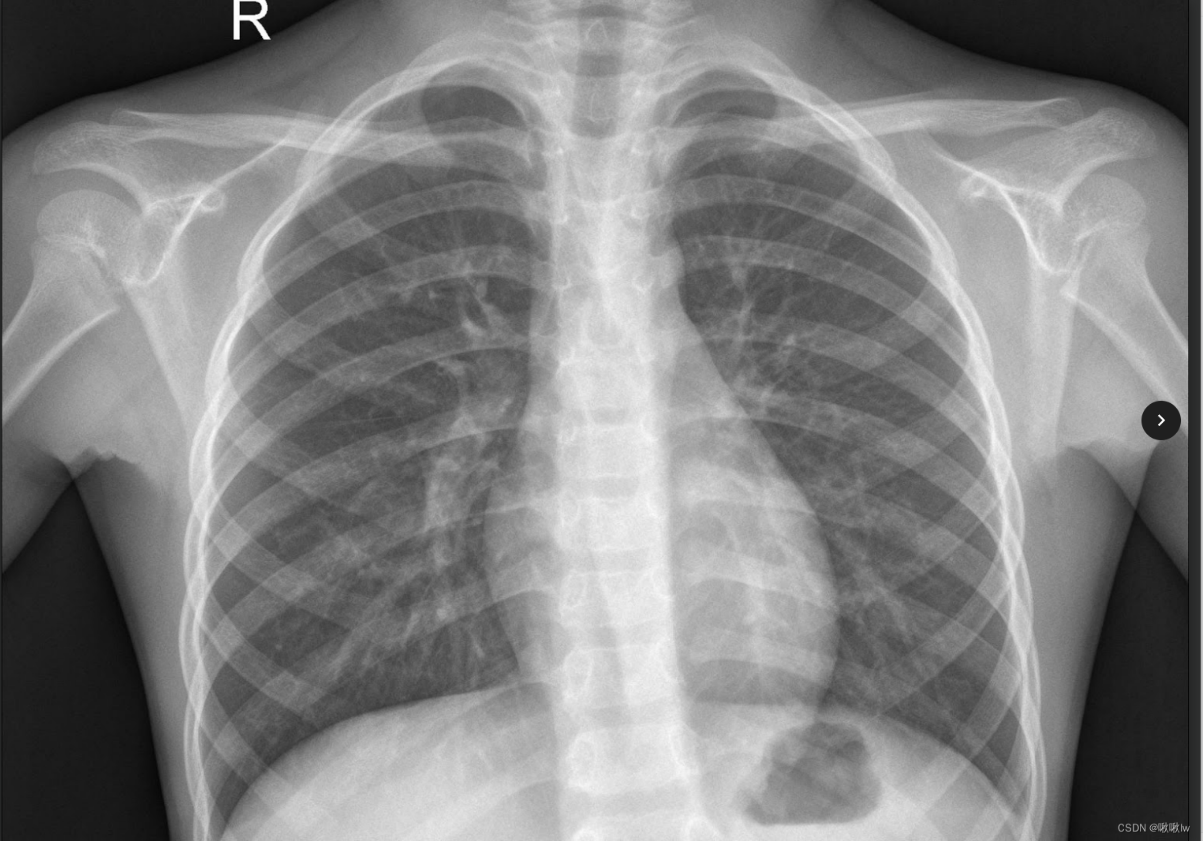
每个文件夹下有两个子文件夹，分别为NORMAL和PNEUMONIA子文件夹。

文本

中度可信度描述已自动生成​

子文件夹结构 

其中， PNEUMONIA子文件夹下包含了正常的肺部图像。

​ NORMAL子文件夹图像展示 

         PNEUMONIA子文件夹下包含了肺炎图像

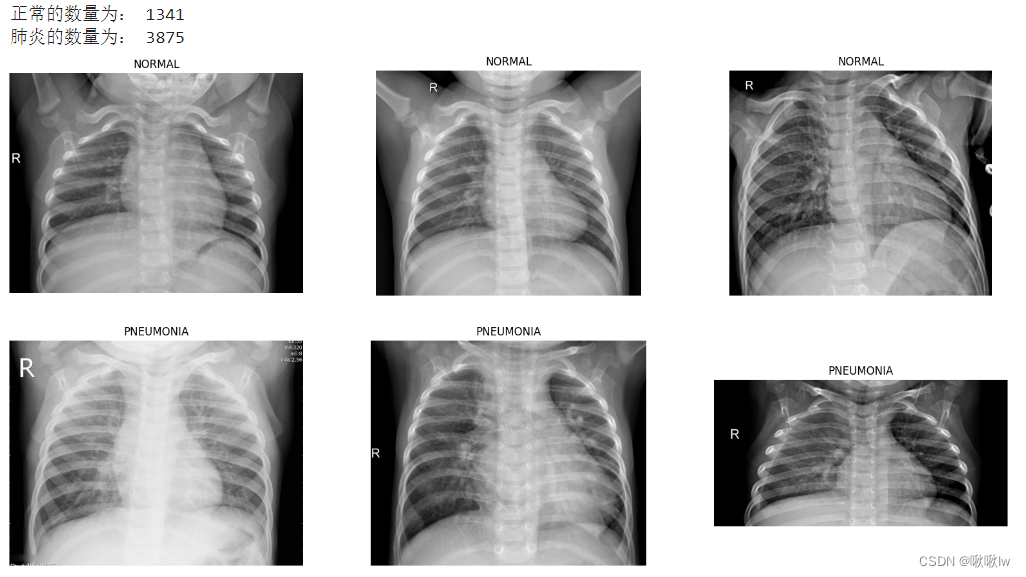
## 2.2探索性数据分析

在这里，我分别取了train数据集下的随机不重样3个正常，3个肺炎的图像进行展示。

文本

描述已自动生成

​

​ 随机图像抽样

图片包含 游戏机, 旧

描述已自动生成​ 计算每个像素位置的平均、方差、最大和最小值

## 2.3提取数据集

在本项目中，为了更好地提取出图像，我构建了一个函数，能够将每个主文件夹下的图片提取出来，并且打好了标签。

## ​2.4数据增强

在这里，我采用了transforms.RandomResizedCrop(64)首先对图像进行随机裁剪，并随机调整裁剪后的图像大小为 64x64 像素。这样的操作有助于模型学习对不同尺寸和位置的物体具有更好的鲁棒性，从而提高泛化能力。

其次，我使用了transforms.RandomHorizontalFlip进行随机水平翻转图像。这个操作通过一定的概率水平翻转图像，以扩充训练数据。这可以帮助模型学到物体在水平方向上的不变性。

文本

描述已自动生成

## 2.5构建数据集

在这里，我对train数据集进行了数据增强操作，test和val数据集并没有进行操作

文本

描述已自动生成

​

# 3：****使用卷积神经网络识别肺炎图像****

## 3.1VGG-16架构

VGG16是由Karen Simonyan和Andrew Zisserman于2014年在论文“VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE SCALE IMAGE RECOGNITION”中提出的一种处理多分类、大范围图像识别问题的卷积神经网络架构，成功对ImageNet数据集的14万张图片进行了1000个类别的归类并有92.7%的准确率。

本项目将分类层的最后一层修改为(1x1x2)即可将分类结果从1000类 修改为二分类。

​

图表

描述已自动生成​VGG-16结构图

## ​3.2****卷积神经网络****

卷积神经网络（Convolutional Neural Network，CNN）是一种专门用于处理具有网格结构数据（如图像和视频）的深度学习模型。CNN 在计算机视觉任务中取得了巨大成功，因为它能够有效地捕获图像中的空间结构信息。

## 3.3****深度神经网络****

深度神经网络（Deep Neural Network，DNN）是一种神经网络结构，其具有多个隐藏层，使其成为深层次模型。深度神经网络是深度学习的核心组成部分，能够学习和表示更抽象、更复杂的数据特征，适用于各种机器学习任务。

## 3.4更改VGG-16网络结构

传统的VGG-16网络的输出是1000的大小，为了适合本项目，我将网络改成了2分类问题并对一部分网络进行了优化。

文本

描述已自动生成

# 4：在GPU上训练

## 4.1参数设置

在这里我使用了以下几个部分来提高训练的精度：

（1）：交叉熵损失函数 (nn.CrossEntropyLoss())。交叉熵损失对于分类任务是一种常见的损失函数，它在训练期间衡量模型的预测和真实标签之间的差异。

（2）：Adam 优化器 (optim.Adam)。是一种基于梯度的优化算法，通常在深度学习中表现较好。

（3）： ReduceLROnPlateau 学习率调度器(optim.lr\_scheduler.ReduceLROnPlateau)。该调度器在验证集上监测模型性能，并在性能停滞时降低学习率。

截图里有图片

描述已自动生成

## 4.2在GPU上训练50次

在这里，我使用了三个数据集，分别为train，val，test。

1. **训练集（train）**：
   * **用途：** 用于训练机器学习模型。模型通过学习训练集中的样本来调整参数，使其能够捕捉输入数据的模式和特征。
   * **特点：** 训练集通常是最大的数据集，包含用于模型训练的大量样本。高质量、多样性的训练集有助于提高模型的泛化能力，使其在未见过的数据上表现良好。
2. **验证集（val）**：
   * **用途：** 用于调整模型超参数、选择模型架构和进行早停等操作。验证集上的性能评估有助于避免模型在训练集上过拟合，提高对未知数据的泛化能力。
   * **特点：** 验证集通常是从独立于训练集的数据中划分出来的，模型在训练过程中不使用验证集的信息。在训练过程中，通过监控验证集上的性能来调整模型的参数和架构。
3. **测试集（test）**：
   * **用途：** 用于评估训练好的模型的性能。测试集中的样本是模型在训练和验证过程中未曾见过的数据，因此测试集上的性能评估更接近模型在真实场景中的表现。
   * **特点：** 测试集应该是完全独立于训练集和验证集的，确保模型在测试集上的表现不受过拟合或过度调整的影响。测试集上的性能评估是对模型泛化能力的最终验证。

## 4.3查看test数据集F1分数及时间

我发现，在GPU上查看F1分数和时间时，大概F1分数达到0.9以上，时间为9s左右。

​

​

## 4.4保存为VGG16模型并使用模型进行推理测试

这里使用test数据集里面的图像进行推理测试并打印出相应的图像。

图形用户界面, 文本

描述已自动生成

​

文本

描述已自动生成

​

模糊的照片上写着字

描述已自动生成​

# 5：转移到CPU上

## 5.1创建VGG16模型

这里将GPU训练的模型保存到了vgg16.pth中，在CPU上进行加载。

## 5.2尝试直接在CPU上进行训练

这里我发现，在CPU上直接进行推理的话，时间会非常慢。大概为26s左右

文本

描述已自动生成

​

文本

描述已自动生成​

# 6：使用oneAPI组件

## 6.1Transfer Learning with oneAPI AI Analytics Toolkit进行迁移学习

​文本

描述已自动生成

## 6.2使用Intel Extension for PyTorch进行优化

在上一章中，我发现使用CPU直接进行训练的话会相当慢，在这里使用Intel Extension for PyTorch大大提高了速度。大概缩短了一倍的时间，并且F1的值并没有改变。

​

文本

描述已自动生成​

**6.3保存使用Intel Extension for PyTorch进行优化的模型**

文本

描述已自动生成

## 6.4使用 Intel® Neural Compressor 量化模型

这里对优化后的模型vgg16\_optimized.pth进行加载

文本

描述已自动生成

加载完成以后以准确度为评估函数进行量化

量化成功以后会出现如下代码

​日历

描述已自动生成

## 6.5使用量化后的模型在 CPU上进行推理

加载模型

文本

描述已自动生成

进行推理

推理结果

​

模糊的黑白照片

描述已自动生成​

F1分数及推理时间

​

​

​​