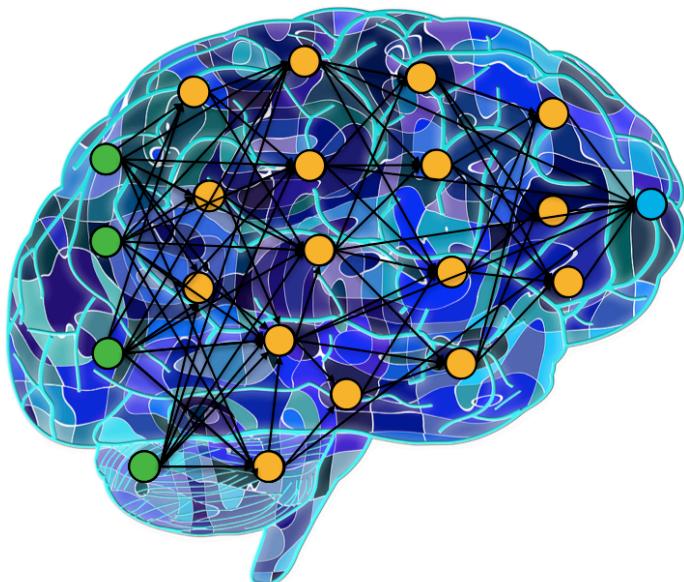


# 人工神经网络课程综合报告

input layer      hidden layer(s)      output layer



张逸凡

2025 春夏学期

# 目录

<b>1 报告 1：多层神经网络的构建与测试</b>	<b>1</b>
1.1 问题简介 . . . . .	1
1.2 模型与算法介绍 . . . . .	1
1.2.1 多层神经网络的前向传播和反向传播 . . . . .	1
1.2.2 反向传播算法的计算图表示方法 . . . . .	2
1.2.3 网络参数的选取 . . . . .	2
1.2.4 损失函数 . . . . .	3
1.3 数据集和算例参数介绍 . . . . .	3
1.3.1 MNIST . . . . .	3
1.3.2 fashionMNIST . . . . .	3
1.4 结果罗列与汇总 . . . . .	3
1.4.1 MNIST . . . . .	3
1.4.2 fashionMNIST . . . . .	5
1.5 结论与心得 . . . . .	6
<b>2 报告 2：利用深度学习框架比较从头训练和微调</b>	<b>7</b>
2.1 问题简介 . . . . .	7
2.2 模型与算法介绍 . . . . .	7
2.2.1 ResNeXt . . . . .	7
2.2.2 DenseNet . . . . .	9
2.3 数据集和算例参数介绍 . . . . .	9
2.3.1 数据集介绍 . . . . .	9
2.3.2 算例参数介绍 . . . . .	9
2.4 结果罗列与汇总 . . . . .	11
2.4.1 ResNeXt50 应用于 CIFAR-10 数据集微调结果 . . . . .	11
2.4.2 ResNeXt 应用于 CIFAR-10 数据集从头训练结果 . . . . .	11
2.4.3 DenseNet 应用于 CIFAR-10 数据集微调训练结果 . . . . .	12

2.4.4 DenseNet 应用于 CIFAR-10 数据集从头训练结果 . . . . .	13
2.5 结论与心得 . . . . .	13
2.6 代码可用性 . . . . .	14
<b>3 报告 3：利用多模态深度学习模型进行喉癌诊断</b>	<b>15</b>
3.1 问题简介 . . . . .	15
3.2 模型与算法介绍 . . . . .	15
3.2.1 模型框架 . . . . .	15
3.2.2 对比学习 . . . . .	16
3.2.3 参数选择 . . . . .	17
3.2.4 性能评估指标 . . . . .	17
3.3 结果罗列与汇总 . . . . .	17
3.3.1 数据收集 . . . . .	17
3.3.2 样本临床特征 . . . . .	18
3.3.3 模型在内部验证集上的表现 . . . . .	18
3.3.4 模型在所有验证集上的表现 . . . . .	19
3.4 结论与心得 . . . . .	20
3.5 代码可用性 . . . . .	20
<b>参考文献</b>	<b>21</b>

## 序言

这份综合报告记录了我在 2025 年春夏学期选修“人工神经网络与算法”这门课程的三次课程报告。总听人们讲“反向传播”、“Transformer”、“超参数”等等术语，上完这个课程后也总算能在这种时候点头微笑了。

这三次课程报告，记录了我从一行一行手搓前馈神经网络，到使用经典的深度学习框架进行调优，再到利用多模态深度学习网络结合对比学习解决医疗上的实际问题，逐步加深，就如同一个神经网络一样。

现在所有学科都在向 AI 靠拢，统计学也是不可避免的。我所选择的今后的科研方向——生物统计学，也注定离不开深度学习。假如哪一天我在科研中用到了深度学习网络，还要感谢这份启蒙报告。

2025.5.6

# 报告 1：多层神经网络的构建与测试

## 1.1 问题简介

在现实生活中，我们经常会遇到分类问题：在医疗领域，医生需通过患者病变区域的影像判断病变类型；在人脸识别领域，机器需通过提取对象的特征，对其进行归类（如年龄，性别等）；在天气预测领域，我们也要通过结合多种指标，对天气状况进行分类预测。为满足我们在这些领域的分类的需求，多层神经网络是近年来备受关注的一个研究领域<sup>1-3</sup>（尽管多层神经网络的功能远不止如此，还可以完成回归、生成、特征学习等任务）。在此课程报告中，我将呈现我对多层神经网络及其实现方法的理解，并通过具体的数据集测试不同的实现方法，最终展示结果。

## 1.2 模型与算法介绍

### 1.2.1 多层神经网络的前向传播和反向传播

在此次报告中，我们采用的模型是多层感知机 (MLP)，它具有以下特点：

- (i) 至少拥有一个隐藏层；
- (ii) 为全连接神经网络，意味着相邻的层之间的神经元两两相连；
- (iii) 同层之间的神经元无连接，信号从输入层到输出层单向传播。

基于以上特点，我们假定我们的感知机有  $L$  层神经元，第  $l$  层神经元个数为  $M_l$ ，给定

$x_0 = x_{input}$  为输入，那么我们可以写出前向传播的递推式 ( $k = 1, 2, \dots, L$ )：

$$\begin{cases} \mathbf{Z}^{(k)} = \mathbf{W}^{(k)} \mathbf{x}^{(k-1)} + \mathbf{b}^{(k)} \\ \mathbf{x}^{(k)} = f_k(\mathbf{Z}^{(k)}) \end{cases} \quad (1.1)$$

这里  $\mathbf{Z}^{(k)}, \mathbf{x}^{(k)}, \mathbf{b}^{(k)}$  是  $M_k$  维向量， $\mathbf{x}^{(k-1)}$  是  $M_{k-1}$  维向量， $\mathbf{W}^{(k)}$  是  $M_k \times M_{k-1}$  维矩阵， $f_k$  表示第  $k$  层神经网络的激活函数。通过上式的前向传播，我们可以最终得到  $\mathbf{x}^{(L)}$  的值作为输出。

于是，训练这个神经网络的本质就是找到合适的  $\mathbf{W}^{(k)}$  和  $\mathbf{b}^{(k)}$ ,  $k = 1, 2, \dots, L$ 。为实现这个目的，我们首先需要确定一个损失函数  $\mathcal{L}(\mathbf{x}^{(L)}, \mathbf{y})$  描述预测值  $\mathbf{x}^{(L)}$  与真实值  $\mathbf{y}$  之间的损失。对于训练集中一个取定的样本，输入  $x_0$  与真实标签  $\mathbf{y}$  是取定的，因此损失函数可以表示为  $\mathbf{W}^{(k)}, \mathbf{b}^{(k)}$ ,  $k = 1, 2, \dots, L$  的函数。

我们经常使用梯度下降法来寻找使得损失函数达到极小值的参数的数值解（假定初值  $w_{ij(l)}^{(0)}, \mathbf{b}_{(l)}^{(0)}$  取定）：

$$\begin{cases} w_{ij(l)}^{(n)} = w_{ij(l)}^{(n-1)} - \eta \frac{\partial \mathcal{L}}{\partial w_{ij}^{(l)}} \\ \mathbf{b}_{(l)}^{(n)} = \mathbf{b}_{(l)}^{(n-1)} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{b}_{(l)}} \end{cases} \quad (1.2)$$

其中  $\eta$  表示学习率， $n$  表示迭代轮数。于是，问题的核心转化为求损失函数关于参数  $w_{ij(l)}, \mathbf{b}_{(l)}$  的偏导数。

基于这种问题的特点，我们可以利用误差反向传播法求损失函数关于参数的梯度。具体来

讲，由链式法则：

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w_{ij}^{(l)}} = \frac{\partial \mathbf{Z}^{(l)}}{\partial w_{ij}^{(l)}} \frac{\partial \mathcal{L}}{\partial \mathbf{Z}^{(l)}} \\ \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}} = \frac{\partial \mathbf{Z}^{(l)}}{\partial \mathbf{b}^{(l)}} \frac{\partial \mathcal{L}}{\partial \mathbf{Z}^{(l)}} \end{cases} \quad (1.3)$$

而第一个式子右端第一项  $\frac{\partial \mathbf{Z}^{(l)}}{\partial w_{ij}^{(l)}} = (0, \dots, x_j^{(l-1)}, \dots, 0)^T$ ，第二个式子右端第一项  $\frac{\partial \mathbf{Z}^{(l)}}{\partial \mathbf{b}^{(l)}} = I_{m(l)}$ ，都是给定的。我们将两式共同的第二项  $\frac{\partial \mathcal{L}}{\partial \mathbf{Z}^{(l)}}$  定义为  $\delta^{(l)}$ ，那么：

$$\begin{aligned} \delta^{(l)} &= \frac{\partial \mathbf{x}^{(l)}}{\partial \mathbf{Z}^{(l)}} \frac{\partial \mathbf{z}^{(l+1)}}{\partial \mathbf{x}^{(l)}} \frac{\partial \mathcal{L}}{\partial \mathbf{Z}^{(l+1)}} \\ &= \text{diag}(f'_l(\mathbf{z}^{(l)}))(W^{(l+1)})^T \delta^{(l+1)} \end{aligned} \quad (1.4)$$

这样，我们得到了  $\delta^{(l+1)}$  与  $\delta^{(l)}$  的递推式。基于这个式子，我们可以先更新第  $l+1$  层的梯度，再更新第  $l$  层的梯度，从而实现反向传播。

## 1.2.2 反向传播算法的计算图表示方法

为了更加直观、具体的将反向传播的计算过程呈现出来，我们可以采用计算图这一工具。计算图的框架由一个有向图构成，其中节点表示某种运算，可以十分基础具体（例如四则运算，指数运算，取倒数，作用激活函数等），而其边表示前向传播的方向。

为计算损失函数关于某参数的梯度，首先我们需要进行前向传播：从输入开始，按顺序“通过”每个节点进行计算，并计算每个步骤所得到的值（我们一般习惯把所得到的值写在箭头上方）。进行完所有运算操作后，我们就得到了最终的输出值，前向传播过程结束。

然后，我们进行反向传播。首先，我们将输出值带入损失函数中。然后，从输出节点开始，计算损失函数相对于每个节点的梯度。通过链式法则，将梯度从输出层反向传播到输入层。每个操作节点根据其输入的梯度计算出自身的梯度，并将其传递给前一层的节点。最终，我们能够通

过计算图直观地得到损失函数关于该参数的梯度值。

### 1.2.3 网络参数的选取

对于网络参数的选取，我们需要选择神经网络的结构（隐藏层个数及每个隐藏层神经元个数）、激活函数形式、损失函数形式、优化算法及其参数（如优化算法中的学习率，ADAM 算法中的  $\beta_1, \beta_2, \epsilon$  等）。

#### 神经网络的结构

对于隐藏层数，由于电脑性能有限，我们选择一个隐藏层；对于隐藏层神经元个数，一个经典的选择是  $n = 28$ （这样，全连接神经网络变为 [784, 28, 10]）。在本报告中，大部分结果（如未标明）对神经元数量进行了适当扩充，选择  $n = 30$ 。并且，在下一节中，我将会尝试不同的神经元个数选择，并对比其表现。

#### 激活函数形式

在本报告中，我将尝试以下三种激活函数：  
Sigmoid 函数：

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \quad (1.5)$$

Tanh 函数：

$$\text{Tanh}(x) = \frac{e^{-x} - e^x}{e^{-x} + e^x} \quad (1.6)$$

Arctan 函数：

$$\text{Arctan}(x) = \tan^{-1}(x) \quad (1.7)$$

#### 优化算法

在本报告中，我将尝试以下三种优化方法：  
梯度下降法 (GD)：

$$\theta^{k+1} = \theta^k - \eta \nabla_{\theta} f(\theta^k; \mathbf{x}, \mathbf{y}) \quad (1.8)$$

随机梯度下降法 (SGD):

$$\theta^{k+1} = \theta^k - \eta \nabla_{\theta} f(\theta^k; x_i, y_i) \quad (1.9)$$

ADAM 法<sup>4</sup>:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t \quad (1.10)$$

这里  $\hat{m}_t = \frac{m_t}{1-\beta_2^t}$ ,  $\hat{v}_t = \frac{v_t}{1-\beta_1^t}$ ;  
 $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$ ,  $v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2$ 。

#### 1.2.4 损失函数

优化算法需要结合损失函数来运行。在本报告中，三种优化方法的损失函数均取为：

$$\mathcal{L}(y, \hat{y}) = \frac{1}{2} \sum_{i=1}^n (y - \hat{y})^2 \quad (1.11)$$

可以看出，这种损失类似于均方误差损失，只是相差一个常数因子。

### 1.3 数据集和算例参数介绍

#### 1.3.1 MNIST

首先，我先尝试了对 MNIST 数据集<sup>5</sup> 进行十分类任务。该数据集包含 60000 个训练样本，10000 个测试样本，每张图片都是  $28 \times 28$  的灰度图，内容为 0-9 的手写数字。对于 ADAM 优化方法，我们选择常用的参数： $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ ,  $\epsilon = 10^{-8}$ 。

由于我的设备性能有限，选用 60000 个训练样本和 10000 个测试样本将会消耗大量时间甚至导致程序崩溃，我选用训练样本中 5000 个样本当作训练集，1000 个样本当作测试集，并且保证两个集合无交集。

#### 1.3.2 fashionMNIST

接着，我尝试了 fashionMNIST 数据集<sup>6</sup>，它是一个替代经典数据集 MNIST 的数据集，包含 70000 张关于衣物的图片 (60000 张训练样本和 10000 张测试样本)，衣物也分为十类。参数选择和样本量的选取与 MNIST 训练集相同。

### 1.4 结果罗列与汇总

#### 1.4.1 MNIST

##### 不同优化算法测试

首先，我测试了最常用的优化方法：随机梯度下降法。首先，我将激活函数选取为 Sigmoid 函数。由于这种方法运行速度较快，我选择训练 10000 个周期，测试这种方法的极限。最终，这种方法在训练集上的精确度达到了 81.38%，在测试集上的精确度达到了 78.80%，效果很好。

然而，用其他两种优化方法：ADAM 法和梯度下降法的效果并不是很好。其中，ADAM 法在第 110 个周期左右就在训练集上达到了 25% 以上的准确率，这比随机梯度下降法在第 130 个周期左右在训练集上达到的准确率 (12%) 要好。然而，从这之后，ADAM 法在训练集上的正确率一直在波动，甚至有所下滑。最终在 1000 轮训练周期后，ADAM 法在训练集上的准确率仅为 27.76%，在测试集上的准确率仅为 26.80%。从图一中可以更加直观地看出，ADAM 算法在达到 28% 左右在训练集内的准确率后开始大幅度波动，而 SGD 算法训练集内准确率稳步上升。由于 ADAM 算法波动的性质，继续训练没有意义，因此图一仅显示前 1000 个周期两种优化算法的性能比较。

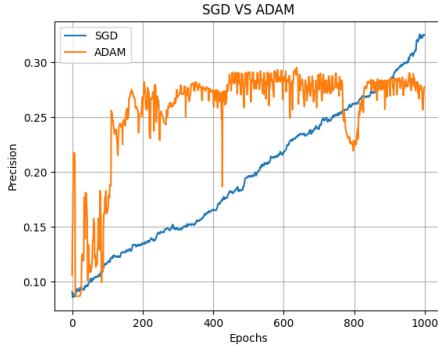


图 1.1: SGD 与 ADAM 比较

对于 GD 算法，最大的问题是运行速度太慢：约 1.2s 才能运行一个周期。这样，若我们想像 SGD 方法那样训练 10000 轮，需要超过三个小时。因此，由于设备性能限制，我们仅研究在前 1000 轮 GD 算法的性质，及其性能与 SGD 方法的比较。GD 方法在前 100 轮内在训练集上的准确率上升很迅速，且在前 1000 轮内准确率都领先于 SGD，最终在 1000 轮后达到了 59.52% 的准确率，在测试集上达到了 55.45% 的准确率，均远高于 SGD 算法（图二）。但由于 SGD 算法的运行速度比 GD 算法的运行速度快 30-40 倍，在实际应用中 SGD 算法更加高效。通过本小节的测试，我们可以总结：SGD 算法是最适合 MNIST 数据集上分类任务的优化算法。

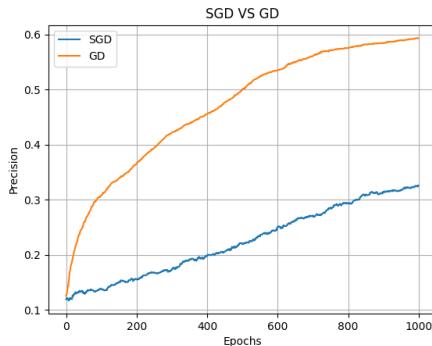


图 1.2: SGD 与 GD 比较

## 不同激活函数测试

在这一小节中，我们把优化方法都设置为最高效的 SGD 方法，测试不同的激活函数的表现。在上一小节中我们已经看到，优化算法选取为 SGD，激活函数选取为 Sigmoid 函数，神经网络的表现很好。下面我将选取 Tanh 函数和 Arctan 函数作为激活函数，并比较其表现与 Sigmoid 函数的表现进行比较。

首先，我们选择 Tanh 函数作为激活函数。在程序运行速度方面，运用 Tanh 函数作为激活函数的程序和运用 Sigmoid 函数作为激活函数的程序运行速度相当，都十分迅速。然而，在运行 100 周期之后，运用 Tanh 函数作为激活函数的程序的准确率开始在 20% 到 60% 范围内剧烈抖动（虽然总体有上升趋势），很不稳定。相比之下，运用 Sigmoid 函数作为激活函数的程序的准确率稳步上升。在 10000 个周期后，运用 Tanh 函数作为激活函数的程序在训练集上仅仅达到了 50.94% 的准确率，在测试集上仅仅达到了 49.90% 的准确率，表现明显劣于用 Sigmoid 函数作为激活函数的程序（训练集准确率：79.58%；测试集准确率：76.20%）（见图 3）。

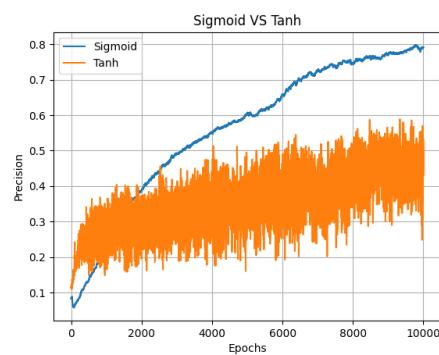


图 1.3: Sigmoid 函数与 Tanh 函数比较

然后，我们选择 Arctan 函数作为激活函数。在程序运行方面，选择 Arctan 函数为激活函数

的程序的运行速度也很迅速。然而，选择 Arctan 函数为激活函数的程序存在与选择 Tanh 函数为激活函数的程序类似的问题：程序的准确率在 20% 到 60% 剧烈波动。在 10000 个周期后，运用 Arctan 函数作为激活函数的程序在训练集仅仅达到 40.78% 的准确率，在测试集上仅仅达到 40.40% 的准确率，表现甚至不如 Tanh 函数（见图 4）。通过以上测试，我们可以得到结论：Sigmoid 函数是最适合在 MNIST 数据集上完成分类任务的激活函数。

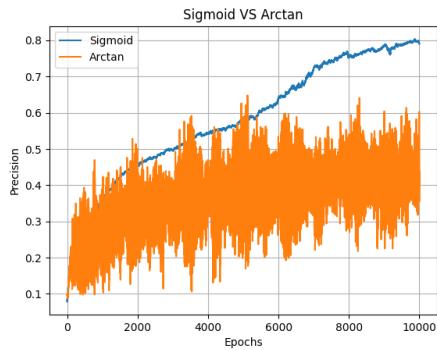


图 1.4: Sigmoid 函数与 Arctan 函数比较

### 不同隐藏层神经元数目性能测试

对于隐藏层神经元数目，一个经典的选择是  $n = 28$ ，然而这个参数显然是可以更改的。在本报告中，我将隐藏层数量分别更改为  $n = 26, 28, 30, 32$ ，并比较其表现差异。直觉上来看，若隐藏层神经元数目过少，则有欠拟合的风险；反之，若隐藏层神经元数目过多，则有过拟合的风险。

选择激活函数为 Sigmoid 函数，优化算法为 SGD 算法，并训练 10000 轮，得到的运行结果如图 5 所示。可见，在前 8000 轮内，不同隐藏层神经元数量的性能排序约为： $n = 30$  优于  $n = 26$  优于  $n = 32$  优于  $n = 28$ ，并没有明显规律。然而，在 10000 轮训练过后，四条曲线基

本重合，最终不同隐藏层神经元数目在训练集上的准确率由表 1 所示。可见，结果都比较令人满意，且差别并不明显。由此我们得到结论：隐藏层神经元在 26 – 32 范围内的表现差别并不明显，且其差异可解释性不强。若在实际应用中，可以通过交叉验证来选择这一参数。

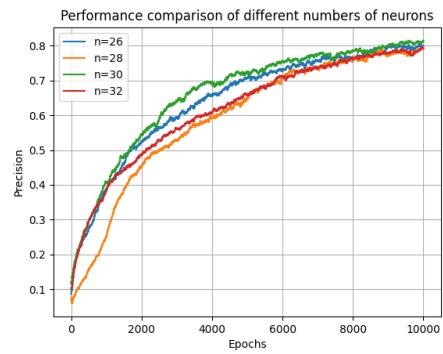


图 1.5: 隐藏层神经元数目对表现的影响

	$n = 26$	$n = 28$	$n = 30$	$n = 32$
Precision	78.3	76.6	80.5	77.1

表 1.1: 不同隐藏层神经元数目在测试集上的准确率比较

#### 1.4.2 fashionMNIST

在本小节中，我们测试所搭建的神经网络在 fashionMNIST 数据集上的表现。由上个小节的结论，我们知道最优的优化算法和激活函数的搭配为：SGD 算法和 Sigmoid 函数，因此本小节的神经网络中选择 SGD 作为优化算法，选择 Sigmoid 函数作为激活函数。

如图 6 所示，fashionMNIST 数据集在训练集的分类准确率在前 2000 个周期内上升迅速（优于 MNIST 数据集）。然而，在 3000 个周期后，MNIST 数据集在训练集的分类准确率实现了反

超。最终，在 10000 个周期后，fashionMNIST 数据集在训练集的准确率为 70.44%，在测试集上的准确率为 68.30%，均低于 MNIST 数据集的分类准确率（训练集：81.27%，测试集：81.50%）。这是符合预期的，因为对衣物图片的分类难度直观上来讲比对手写数字图片的分类难度要高。

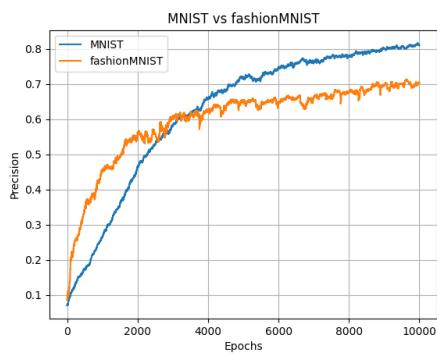


图 1.6: MNIST 数据集与 fashionMNIST 数据集比较

## 1.5 结论与心得

在本报告中，我们测试了不同的激活函数（Sigmoid 函数、Tanh 函数、Arctan 函数）、不同的优化算法（GD、SGD、ADAM）以及不同的隐藏层神经元个数 ( $n = 26, 28, 30, 32$ ) 对多层神经网络分类表现的影响，并且运用两个不同的经典数据集（MNIST, fashionMNIST）进行测试。我们可以得出结论：运用 Sigmoid 函数作为激活函数，SGD 算法作为优化算法，并选取  $n = 30$  作为隐藏层神经元个数是最优的方案，兼具准确率与实用性。其他的组合方案都有不同程度的弱点。

通过本次报告，我详细、具体的掌握了多层神经网络的构建方法，并且体会到了其强大之处，以及选择合适的参数、激活函数、优化算法的重要性。

# 报告 2：利用深度学习框架比较从头训练和微调

## 2.1 问题简介

随着卷积神经网络在近年来的快速发展，许多各式各样的网络相继产生，旨在缓解梯度消失、梯度爆炸、参数量过多等传统卷积神经网络所拥有的问题，从而提升模型的性能。ResNeXt<sup>7</sup> 和 DenseNet<sup>8</sup> 就是其中最经典的例子。事实上，这两个网络早已被应用到除 ImageNet<sup>9</sup> 之外的许多图片集当中，也被应用于多种不同的领域：例如，ResNeXt 被应用于语音识别处理<sup>10</sup>、植物学分类<sup>11</sup> 等领域，而 DenseNet 被应用于医疗<sup>12</sup>、网络安全<sup>13</sup> 等领域，其影响力可见一斑。同时，这两个神经网络框架能够被应用于多个领域，也体现了其泛化能力。

在本报告中，我们尝试将 ResNeXt 和 DenseNet 的框架应用于除 ImageNet 以外的图片集上进行分类任务，并且比较微调和从头训练在多个指标上的差异，从而对这两个经典的卷积神经网络结构拥有更深刻的理解。

## 2.2 模型与算法介绍

### 2.2.1 ResNeXt

ResNeXt 是一个由 Facebook AI Research 团队在 2016 年首次在论文“Aggregated Residual Transformations for Deep Neural Networks”<sup>7</sup> 中

提出的卷积神经网络，它是在经典的残差网络 (ResNet<sup>14</sup>) 基础上发展起来的，结合了 ResNet 和 Inception 的优势，不过又与二者不同。为了理解 ResNeXt 的想法，我们首先介绍残差网络 (ResNet) 的核心思想。

ResNet 的核心思想在于跳层连接 (Skip Connect)<sup>14</sup>，直观的讲就是用一个恒等映射链接若干层神经网络。严格来讲，如果我们将一个神经网络的构建模块记为

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x} \quad (2.1)$$

这里， $\mathbf{x}, \mathbf{y}$  分别是输入向量和输出向量， $\mathcal{F}$  为一个非线性函数，通常包括卷积、激活等操作， $\{W_i\}$  为可学习参数，那么输出  $\mathbf{y}$  实际上可以被视为  $\mathcal{F}(\mathbf{x}, \{W_i\})$  和恒等映射  $\mathbf{x}$  的求和 (图 1)，恒等映射的存在是 ResNet 与传统 CNN 最大的区别。

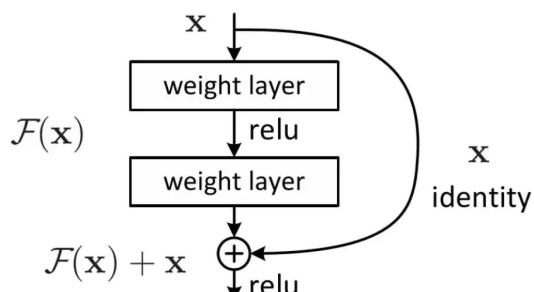


图 2.1: ResNet 网络的基本模块

我们可以从两个角度理解为何要增加恒等映射。首先，可以从缓解梯度消失的角度来理解。假设我们有正向传播表达式：

$$\begin{cases} f = f(\mathbf{x}, W), \\ \sigma = \sigma(f), \\ \hat{y} = g(\sigma), \\ L = L(y, \hat{y}). \end{cases} \quad (2.2)$$

其中， $f, \sigma, g, L$  分别表示卷积、激活函数、分类器和损失函数。回忆反向传播公式：

$$\frac{dL}{dW} = \frac{df}{dW} \frac{d\sigma}{df} \frac{d\hat{y}}{d\sigma} \frac{dL}{d\hat{y}} \quad (2.3)$$

若上式中等号右侧四个式子有一项接近于 0(如 Sigmoid 函数梯度  $< 1$ )，那么随着神经网络不断加深，梯度将会收敛到 0，这很大程度上影响了神经网络的深度。

然而，若我们像 (1) 式那样，在等号右侧加上恒等映射，那么即使  $\frac{d\mathcal{F}}{dx} \approx 0$ ，若记  $\mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x} = g(\mathbf{x}, \{W_i\})$ ，则我们仍有  $\frac{dg}{dx} = \frac{d\mathcal{F}}{dx} + 1$ 。于是，梯度从这一个构建模块传播到前一个构建模块时不会剧烈衰减，从而缓解了梯度消失的问题。

其次，顾名思义，ResNet 将神经网络的目标从“拟合复杂函数”转变为“拟合残差”，而后者往往更易优化(用 SGD 等优化方法求解)。不仅如此，恒等映射在一定程度上保证了浅层特征能够直接通过残差项传递到深层，从而保证了特征复用。

那么，ResNeXt 与 ResNet 有何异同？首先，ResNeXt 继承了 ResNet “残差连接”的思想。不同的是，ResNeXt 运用了“分组卷积”这一技巧。

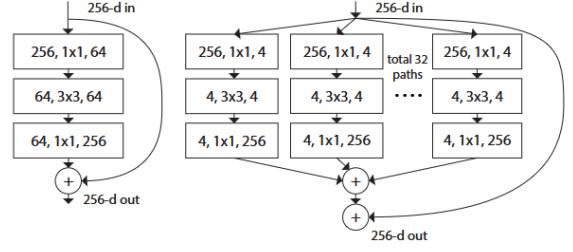


图 2.2: ResNet 与 ResNeXt 基本模块对比

由图 2 所示，一个 ResNeXt 的基本模块实际上将输入的通道分为了 32 个小组(基数)分别学习特征，然后再对所有学到的特征进行整合。直观上来看，图 2 中，ResNeXt 的基本模块要远比 ResNet 的基本模块复杂。然而，事实是，二者的复杂度基本相当。这是由于分组卷积这一技巧实际上是可以减少参数量的(在输入和输出通道数相同的情况下)：假设输入通道数为  $C_{in}$ ，输出通道数为  $C_{out}$ ，卷积核尺寸为  $K \times K$ ， $G$  为基数。那么对于传统的卷积操作，参数量为  $C_{in} \times C_{out} \times K \times K$ ，而对于分组卷积，参数量仅为  $G \times (\frac{C_{in}}{G}) \times (\frac{C_{out}}{G}) \times K \times K = (C_{in} \times C_{out} \times K \times K)/G$ ，将参数量减少了  $K$  倍。

鉴于以上分析，一个 ResNeXt 的构建模块的传播公式可以记为：

$$\mathbf{y} = \mathbf{x} + \sum_{i=1}^C \mathcal{T}_i(\mathbf{x}) \quad (2.4)$$

这里  $C$  为基数。可见，ResNeXt 在保留了残差连接的基础上，使用了“分组卷积”的技巧。但这与 Inception 也略有不同，因为 ResNeXt 在每个分支上运用相同的拓扑结构。

最后，在此贴上 ResNeXt-50 和 ResNeXt-50(32x4d) 的结构图：

stage	output	ResNet-50	ResNeXt-50 (32×4d)
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
		3×3 max pool, stride 2	3×3 max pool, stride 2
conv2	56×56	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128, C=32 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3	28×28	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256, C=32 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
conv4	14×14	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512, C=32 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$
conv5	7×7	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 1024 \\ 3 \times 3, 1024, C=32 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		$25.5 \times 10^6$	$25.0 \times 10^6$
FLOPs		$4.1 \times 10^9$	$4.2 \times 10^9$

图 2.3: ResNeXt 结构图

## 2.2.2 DenseNet

DenseNet 是康奈尔大学等机构由 2017 年在论文 “Densely Connected Convolutional Networks”<sup>8</sup> 中提出的，其主要思想为通过密集连接提升特征重用和梯度流动，显著减少参数量并提升模型性能。一个五层的 DenseNet 基础模块如图 4 所示。

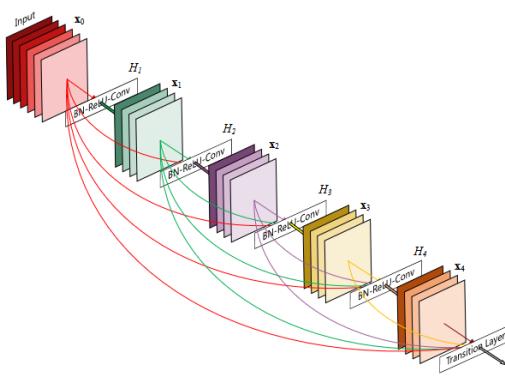


图 2.4: DenseNet 网络的基本模块

从图中可以看出，与传统的卷积神经网络不同，DenseNet 中每一层神经网络用其之前每一层神经网络的输出作为输入，并且其输入用作之

后每一层神经网络的输出。于是，一个  $L$  层的 DenseNet 共有  $\frac{L(L+1)}{2}$  个连接（与之形成对比，传统的神经网络共有  $L$  个连接）。这也就是说：

$$\mathbf{x}_l = H_l([\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{l-1}]) \quad (2.5)$$

其中， $[\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{l-1}]$  表示由第 0 层至第  $l-1$  层输出的特征图的拼接结果。

我们再一次在此贴上 DenseNet 的结构图

Layers	Output Size	DenseNet-121 ( $k=32$ )	DenseNet-169 ( $k=32$ )	DenseNet-201 ( $k=32$ )	DenseNet-161 ( $k=48$ )
Convolution	112×112				
Pooling	56×56			3×3 max pool, stride 2	
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	28×28			1×1 conv	
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	14×14			1×1 conv	
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 36$
Transition Layer (3)	7×7			1×1 conv	
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$
Classification Layer	1×1			7×7 global average pool	10000 fully-connected, softmax

图 2.5: DenseNet 结构图

## 2.3 数据集和算例参数介绍

### 2.3.1 数据集介绍

本报告中，应用的数据集为 CIFAR-10 数据集<sup>15</sup>，它是一个经典的计算机视觉数据集，广泛用于图像分类任务的基准测试。这个数据集拥有 50000 个训练数据、10000 个测试数据以及 10 个类别，图片分辨率为  $32 \times 32$ ，是一个难度适中的测试数据集。

### 2.3.2 算例参数介绍

#### ResNeXt50 微调参数介绍

首先，我们对 ResNeXt50 这个网络进行微调。我们选用的是 ResNeXt50-32x4d 这个版本。首先，我们需要解决的问题是由于原模型是用与 ImageNet 的 1000 分类任务的，而我们的任务为 10 分类任务，因此需要修改最后一层的全连接

层。这里，我们将最后一层神经元个数修改为 10，再作用 SoftMax 函数。注意到 ImageNet 输入的分辨率 ( $224 \times 224$ ) 与 CIFAR-10 输入的分辨率 ( $32 \times 32$ ) 也不同，但不必担心，因为 ResNeXt 会在最后一个卷积层后连接一个全局池化层（池化到  $1 \times 1$ ），因此输入分辨率的差异是不影响 ResNeXt 的工作的。

接着，我们对训练集的图片进行了随机增强（以 0.5 概率随机水平翻转以及随机裁剪），旨在增强模型的泛化能力，同时缓解过拟合。

对于 Batch Size，我们设置为 128，也就是说每批次处理 128 个图片做梯度下降、反向传播。这样的好处是，比单样本训练更加高效，同时比全数据集训练更节省内存。这样，训练集每一个周期需要训练 391 个批次；验证集每一个周期需要训练 79 个批次。

对于优化器，我们选择 SGD 算法，具体参数为：学习率设置为 0.01，动量设置为 0.9，权重衰减设置为  $5e^{-4}$ 。具体来说，权重衰减本质上是对可学习的权重  $w$  做一个二范数惩罚：

$$Loss_{new} = Loss + \frac{WD}{2} \sum w^2 \quad (2.6)$$

这里 WD 代表权重衰减参数，这等价于说

$$w_{t+1} = w_t - \eta(\nabla w_t + WDw_t) \quad (2.7)$$

这里  $\eta$  为学习率。在此基础上，我们再引进动量参数，表达式为：

$$\begin{cases} v_t = Mv_{t-1} + \eta(\nabla w_t + WDw_t) \\ w_{t+1} = w_t - v_t \end{cases} \quad (2.8)$$

这里动量用  $M$  表示。它的意义是，参数的更新不仅仅依赖于当前梯度，还依赖于梯度的历史数据。

对于损失函数，我们选择交叉熵损失：

$$H(y, \hat{y}) = - \sum_{i=1}^{10} y_i \log(\hat{y}_i) \quad (2.9)$$

其中  $\hat{y}$  为模型输出的 SoftMax 概率。

此外，为了加快收敛，我们采用学习率调度机制：每 7 个训练周期，学习率降为原本的 0.1。训练的总周期数选为 20。

## ResNeXt50 从头训练参数介绍

对于 ResNeXt50-32x4d 的从头训练任务，大部分的参数以及算法和微调任务相同。当然，最大的区别就是我们不加载预训练的权重，所有可学习的权重全部随机初始化。

此外，我们对一些参数进行了调整：由于对于从头学习任务来说，前几个训练周期损失函数距离收敛还很远，因此初始学习率我们从 0.01 上调至 0.1；总训练周期数由 20 个训练周期上调至 120 个学习周期；学习率调度机制从原先的每 7 个训练周期学习率衰减为原先的 0.1 倍改为在第 30 和 60 个训练周期学习率衰减为原先的 0.1 倍。

## DenseNet121 微调参数介绍

这里，大部分参数与 ResNeXt50 微调参数相同，仅列举不同点：

- (1) 由于 DenseNet 一般需要较小学习率，这里将学习率由 0.01 调整为 0.002，将学习率调度的步长由 7 调整为 5；
- (2) 由于 DenseNet 占用较大缓存，因此批次量由 128 调整为 64。

## DenseNet121 从头训练参数介绍

这里，大部分参数与 ResNeXt50 从头训练参数相同，仅列举不同点：

- (1) 学习率由 0.1 调整为 0.05；
- (2) 批次量由 128 调整为 64；
- (3) 训练周期由 120 减少至 90。

## 2.4 结果罗列与汇总

### 2.4.1 ResNeXt50 应用于 CIFAR-10 数据集微调结果

首先，我们绘制了训练集和验证集的准确率以及损失关于训练周期的函数（图 6）。从图中可以看到一个有趣的现象：在前两个训练周期，训练集的准确率比验证集低，并且训练集的损失比验证集高；在第三个周期之后，训练集的表现才逐渐形成了反超。这可能是因为我们对训练集的图片进行了随机的增强。总的来讲，训练集和验证集的准确率和损失都在第 8 个训练周期后逐渐收敛；在 20 个训练周期之后，训练集上的准确率达到了 95.77%，验证集上的准确率达到了 89.58%，而验证集上的最佳准确率（于第 18 个训练周期达到）为 89.65%。

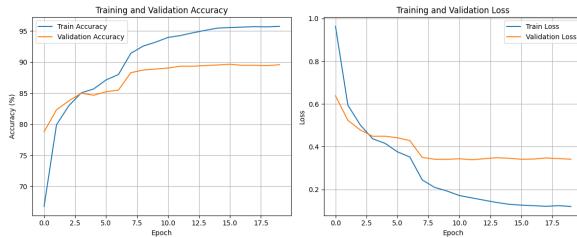


图 2.6: ResNeXt 微调训练曲线

不仅如此，我们可以绘制出训练集和验证集的混淆矩阵：

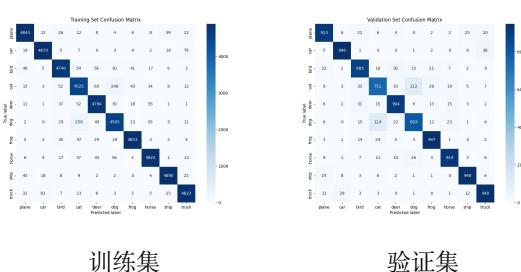


图 2.7: ResNeXt 微调训练集和验证集混淆矩阵

可以看出，对于训练集以及验证集，绝大部分图片位于对角线元素，这能够说明模型良好的性能。有趣的是，除了对角线元素之外，位于混淆矩阵 (6, 4) 和 (4, 6) 的元素也较多（这个现象对于训练集和验证集都成立），这说明相对于其他物体，猫和狗相对难以区分，这也是符合直觉的。

不仅如此，为直观感受微调后 ResNeXt50 网络的分类性能，我们还可以对全连接层之前的这一层（全局平均池化后的输出）做 t-SNE<sup>16</sup> 降维后的可视化。由于全局平均池化后通道数为 2048，每个图片实际上是一个 2048 维的向量，我们对验证集上的图片随机抽取 1000 个图片进行 t-SNE 降维（图 8）。

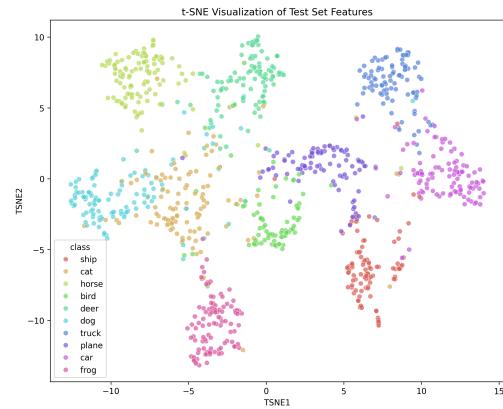


图 2.8: ResNeXt 微调 t-SNE 降维可视化

可以看出，在图中相同类别的图片分布相对集中。同样的，我们也可以看到猫和狗类的点有一部分重叠，说明 ResNeXt50 网络对猫和狗的分类效果并没有其他类别那么理想。

### 2.4.2 ResNeXt 应用于 CIFAR-10 数据集从头训练结果

首先，我们绘制训练集和验证集的准确率以及损失关于训练周期的图像（图 9）。首先，在

120 个训练周期后，训练集的准确率达到了惊人的 98.27%，高于微调的结果 (95.77%) 但验证集的准确率却只有 87.32%，低于微调的结果 (89.58%)。训练集和验证集准确率差异如此之大，说明模型若不使用预训练好的参数，更容易将注意力集中在训练集的某些无关紧要的特征上。不仅如此，观察损失曲线，我们可以发现在第 60 个训练周期之后，训练集的损失反而增加了。这说明我们的从头训练出现了一定程度的过拟合。实际上，训练 60 个周期已经足够训练集的准确率收敛。

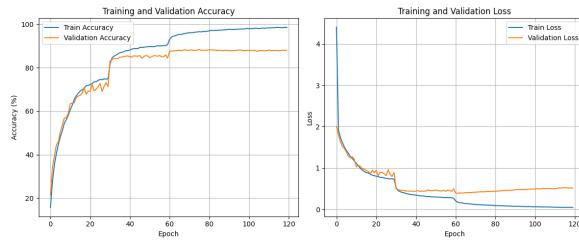


图 2.9: ResNeXt 从头训练曲线

此外，与微调相同，我们同样绘制了训练集与验证集的混淆矩阵 (图 10) 以及验证集随机抽取 1000 个样本的 t-SNE 降维结果 (图 11)，其形状特征与微调的结果大体类似：

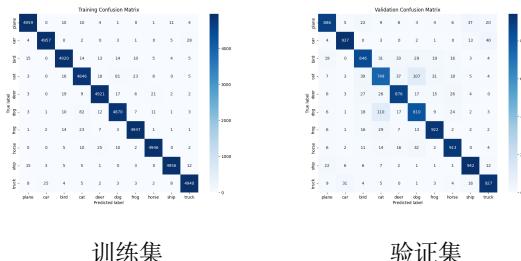


图 2.10: ResNeXt 从头训练训练集和验证集混淆矩阵

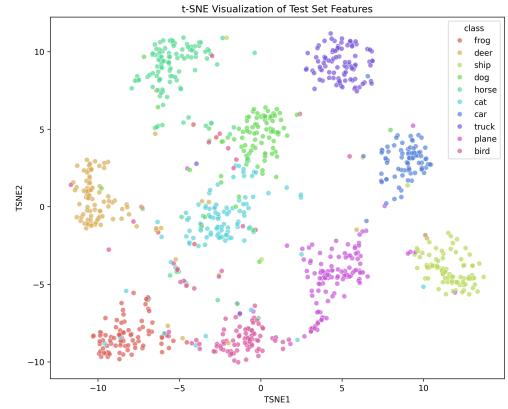


图 2.11: ResNeXt 从头训练 t-SNE 降维可视化

#### 2.4.3 DenseNet 应用于 CIFAR-10 数据集微调训练结果

相比于 ResNeXt，DenseNet 的训练时长较长，一个训练周期在 T4 GPU 上大约需要 2 分钟。经过 20 个训练周期，训练集、验证集的准确率和损失曲线如图 12 所示：

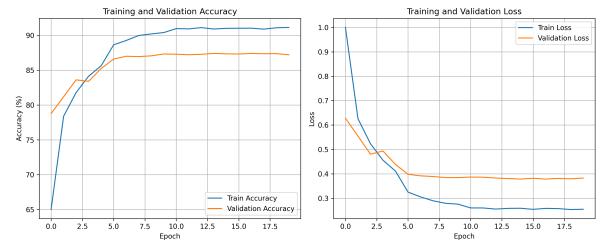


图 2.12: DenseNet 微调训练曲线

从图中可以看出，训练集和验证集收敛到的准确率均低于 ResNeXt(91.17% vs 95.77%; 87.13% vs 89.65%)。我尝试调整 DenseNet 的许多参数，正确率仍然无法超过 ResNeXt。这可能是因为 DenseNet 的密集连接会保留大量中间特征，而 CIFAR-10 的 32x32 输入分辨率可能无法充分发挥其优势，导致特征冗余和计算浪费。类似的，我们绘制了混淆矩阵与 t-SNE 降维可视化。可以看出，DenseNet 的 t-SNE 降维可视化

化结果明显不如 ResNeXt(图 14)。

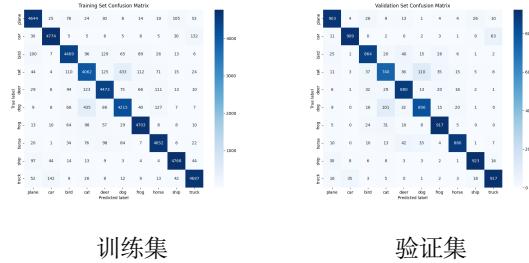


图 2.13: DenseNeXt 从头训练训练集和验证集混淆矩阵

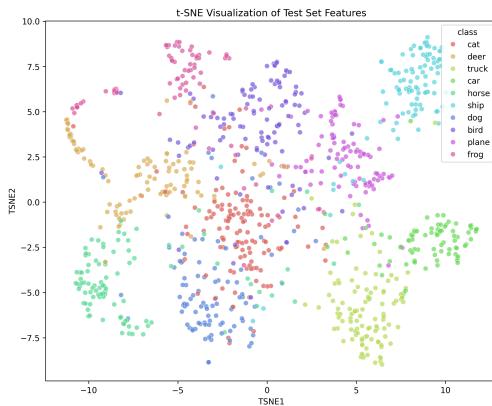


图 2.14: DenseNet 微调 t-SNE 降维可视化

#### 2.4.4 DenseNet 应用于 CIFAR-10 数据集从头训练结果

我们在 T4 GPU 上训练 90 个周期，时长约为 3 小时。训练曲线如图 15 所示：

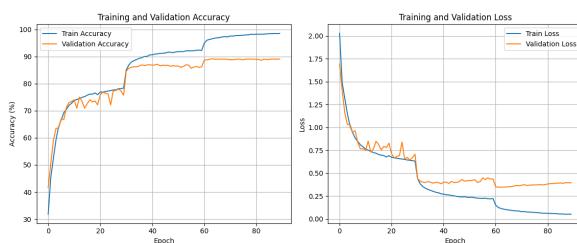


图 2.15: DenseNet 从头训练曲线

有趣的是，与微调不同，从头训练 DenseNet 的表现强于 ResNeXt (验证集 89.15% vs 87.32%)，具体的原因有待进一步分析。虽然从损失曲线来看，从第 60 个训练周期开始出现轻微的过拟合，但是 60 周期后验证集准确率基本保持稳定。

类似的，我们也绘制出了混淆矩阵以及 t-SNE 可视化：

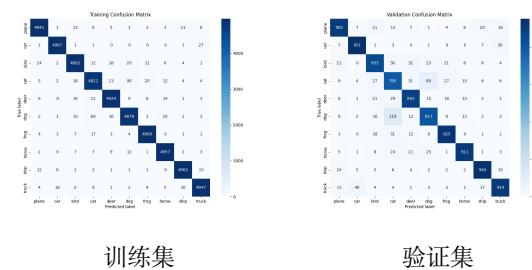


图 2.16: DenseNeXt 微调训练集和验证集混淆矩阵

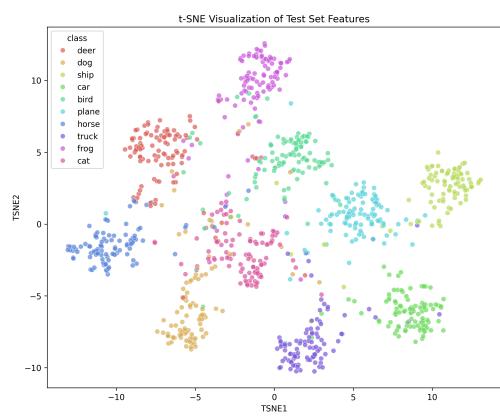


图 2.17: DenseNet 从头训练 t-SNE 降维可视化

## 2.5 结论与心得

在本次报告中，我们测试了两种经典的卷积神经网络——ResNeXt 和 DenseNet 在 CIFAR-10 数据集上微调和从头训练的表现，并绘制了

其训练曲线、混淆矩阵及 t-SNE 降维可视化。总体来讲，两个网络在 CIRFAR-10 的数据集上都达到了不错的表现。

对比两个卷积神经网络，DenseNet 训练时长略长于 ResNeXt；在微调任务上，ResNeXt 表现优于 DenseNet；而在从头训练任务上，DenseNet 表现优于 ResNeXt。总体来说，从头训练任务需要的训练时长远长于微调任务，也存在过拟合的风险，但收敛到的训练集、验证集准确率均高于微调任务。

## 2.6 代码可用性

本论文使用的所有代码均为 python 代码，使用 python 版本为 3.12.7。所有代码均存放于：

[https://github.com/yifanzhang7/  
neuralnetwork-assignment2](https://github.com/yifanzhang7/neuralnetwork-assignment2)

# 报告 3：利用多模态深度学习模型进行喉癌诊断

## 3.1 问题简介

咽喉癌是一种发病率极高的癌症：喉癌在全球头颈部恶性肿瘤中发病率位居第二，而下咽癌约占此类病例的 3%。2022 年，全球新发喉癌与下咽癌病例分别达到 188,960 例和 86,276 例<sup>17</sup>。尽管近年来，癌症的发病率有所下降且治疗方法不断改进，但喉癌与下咽癌的生存率提升甚微，其中下咽癌的 5 年生存率仍维持在 30 – 35% 的较低水平。

本报告中的分类任务为多分类任务：患者的真实标签被分为癌症组、非癌病变组以及正常组。其中，非癌病变组可以被进一步细分为良性病变组、低级别病变和高级别病变。所有的真实标签都由临床病理学得来。因此，本报告的分类任务覆盖了三分类任务以及五分类任务。

## 3.2 模型与算法介绍

### 3.2.1 模型框架

本报告的模型整合了三个不同模态的患者信息：喉镜检查图像、患者主诉和临床特征。一个输入样本的例子由图 1 所示：

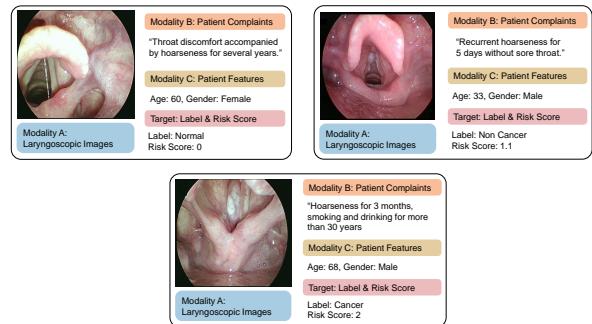


图 1: 输入样本样例

在图像与文本模态处理方面，本研究采用 BEIT3<sup>18</sup> 作为多模态数据感知的骨干网络。在模态特征提取方面，BEIT3 通过在前馈层中引入混合模态专家结构，对 Transformer<sup>19</sup> 编码器模型进行增强，使其能够针对不同模态的输入执行不同类型的非线性变换。

就模态交互而言，BEIT3 采用跨模态共享的统一多头自注意力层，从而使其具备多模态信息交互能力。此外，BEIT3 已在海量图像和文本数据集上进行预训练，在图像、文本及图文样本等多种模态上展现出强大的泛化能力。

对于包含  $N$  个样本的数据集中第  $i$  个样本的联合多模态输入，由患者主诉 (Text)、喉镜检查图像 (Image) 和患者临床特征 (Feature) 组成，可表示为：

$$x_i = \{\text{Text}_i, \text{Image}_i, \text{Feature}_i\}_{i=1,2,\dots,N} \quad (3.1)$$

参考视觉 Transformer 和 BERT 的表示方法, BEIT3 采用共享注意力多模态变换器 (SAMT)<sup>20</sup>, 将文本和图像输入映射为带有 CLS 标记的多重嵌入表示:

$$\begin{cases} V_i, T_i = \text{SAMT}(\text{Text}_i, \text{Image}_i) \\ V_i = [v_{i,\text{cls}}, v_{i,1}, v_{i,2}, \dots, v_{i,m_v}], V_i \in \mathbb{R}^{(m_v+1) \times D} \\ T_i = [t_{i,\text{cls}}, t_{i,1}, t_{i,2}, \dots, t_{i,l_i}], T_i \in \mathbb{R}^{(l_i+1) \times D} \end{cases} \quad (3.2)$$

其中,  $D$  表示嵌入空间的维度,  $m_v$  代表图像经分块嵌入处理后得到的图块数量,  $l_i$  表示第  $i$  个样本中患者主诉文本  $\text{Text}_i$  的长度。

为了将第三个模态: 患者特征与前两个模态融合, 我们通过特征交互网络 (FIN) 将数值型特征向量  $\text{Feature}_i$  映射到与骨干网络生成的图像/文本的 CLS 标记维度相同的特征空间。随后, 对图像表征、文本表征及患者特征表征执行向量进行求和, 最终获得综合多模态表征  $s_i$ :

$$\begin{cases} f_i = \text{FIN}(\text{Features}_i) \\ s_i = f_i + v_{i,\text{cls}} + t_{i,\text{cls}} \end{cases} \quad (3.3)$$

接下来,  $s_i$  会作为患者嵌入网络 (PEN) 的输入, 生成嵌入式的患者特征向量  $z_i$ 。最终, 通过评分网络 (SN) 输出患者的风险评估分值  $\text{score}_i$ 。模型的优化目标是最小化预测风险评估分值  $\text{score}_i$  与真实目标风险分值  $y_i$  之间的均方误差:

$$\begin{cases} z_i = \text{PEN}(s_i), \text{score}_i = \text{SN}(z_i) \\ \mathcal{L}_{\text{reg}} = \frac{1}{N} \sum_{i=1}^N (y_i - \text{score}_i)^2 \end{cases} \quad (3.4)$$

样本的最终预测分类结果由其风险评估分值与预设的各类别风险基线分值之间的距离决定:

$$\text{pred}_i = \operatorname{argmin}_{c_k \in C} |\text{score}_i - c_k| \quad (3.5)$$

其中,  $c_k$  的选取方式将在 2.4 小节中展开。

### 3.2.2 对比学习

对比学习任务是人工智能领域近年来的热门领域之一。该方法首先生成单个样本数据的不同视图, 随后将这些视图的表征在嵌入空间中进行对齐。

在多模态对比学习框架中, 对比语言-图像预训练 (CLIP)<sup>21</sup> 的目标是利用未标注的多模态数据对, 在不同模态的嵌入空间中构建丰富的对比信息, 从而实现多模态信息的对齐。其损失函数可表示为:

$$\begin{cases} l_i^{(v_{i,\text{cls}} \rightarrow t_{i,\text{cls}})} = -\log \text{softmax}\left(\frac{\langle g_v(v_{i,\text{cls}}), g_t(t_{i,\text{cls}}) \rangle}{\tau}\right)_i \\ l_i^{(v_{i,\text{cls}} \rightarrow t_{i,\text{cls}})} = -\log \text{softmax}\left(\frac{\langle g_t(t_{i,\text{cls}}), g_v(v_{i,\text{cls}}) \rangle}{\tau}\right)_i \\ \mathcal{L}_{\text{CLIP}} = \sum_{i=1}^N \frac{1}{2} \left( l_i^{(v_{i,\text{cls}} \rightarrow t_{i,\text{cls}})} + l_i^{(t_{i,\text{cls}} \rightarrow v_{i,\text{cls}})} \right) \end{cases} \quad (3.6)$$

其中,  $g_t, g_v : \mathbb{R}^D \rightarrow \mathbb{R}^D$  分别表示文本嵌入空间和视觉嵌入空间中的非线性映射函数, 用于对比学习。余弦相似度定义为:  $\langle \mathbf{a}, \mathbf{b} \rangle = \frac{\mathbf{a}^\top \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}$  温度系数项:  $\tau \in \mathbb{R}^+$  为可学习参数。

为深入挖掘多模态数据集中的跨模态匹配信息, 本报告采用 CLIP 作为预训练任务, 以增强模型的多模态任务处理能力。

对比学习的概念不仅可应用于图文对的无监督预训练, 还能在监督式下游任务中为网络嵌入空间提供额外监督信号。本报告通过在患者级特征嵌入空间  $Z$  中实施监督对比学习 (SCL)<sup>22</sup>, 显著提升了模型性能。

这里, 监督对比学习的损失函数  $\mathcal{L}_{\text{SCL}}$  的定义为:

$$\mathcal{L}_{\text{SCL}} = - \sum_{i=1}^N \frac{1}{|P(i)|} \sum_{j \in P(i)} \log \frac{\exp(z_i \cdot z_j / \tau)}{\sum_{k \in A(i)} \exp(z_i \cdot z_k / \tau)} \quad (3.7)$$

这里集合

$$P(i) = \{j | j \neq i \wedge \text{label}_i \neq \text{label}_j\}, A(i) = \{k | k \neq i\}.$$

于是总的损失函数为  $\mathcal{L}_{\text{SCL}}$  和  $\mathcal{L}_{\text{reg}}$  的加权求和：

$$\mathcal{L}_{\text{Total}} = (1 - \lambda)\mathcal{L}_{\text{SCL}} + \lambda\mathcal{L}_{\text{reg}} \quad (3.8)$$

这里  $\lambda$  为超参数。这个目标函数通过将类别信息融入嵌入特征空间的构建过程，使模型能够从监督信号中获取多样化的信息形式，从而显著提升模型性能。整个模型的框架由图 2 所示：

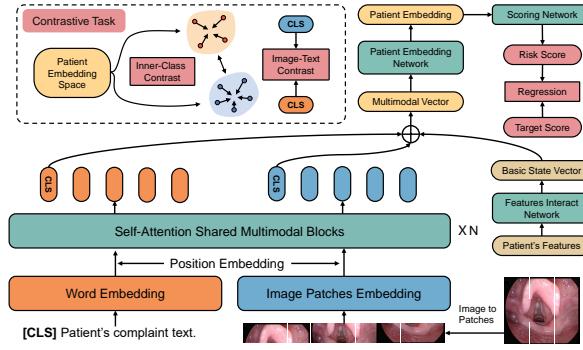


图 2: 模型框架

### 3.2.3 参数选择

在模型训练过程中，我们采用 AdamW 优化器，参数设置如下：学习率  $\eta = 10^{-5}$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ ，批处理大小设为 126，骨干网络选用 BEIT3-base 模型。本实验均在配备 3 块 NVIDIA GeForce RTX 3090 (24GB 显存) GPU 的深度学习服务器上完成。

对于每一类别的风险基线分值  $c_k$ ，我们将正常组、非癌病变组以及癌症组的基线分值分别预设为 0、1 和 2。对于五分类任务中的低级别病变和高级别病变的风险基线分值，我们采用网格搜索的方式确定这两个参数，最终分别设为 1.1 和 1.3：

低级别病变 分值	高级别病变 分值	训练集			内部验证集		
		Acc.	Pre.	Rec.	Acc.	Pre.	Rec.
1·1	1·2	96·51	95·32	96·83	96·06	88·03	88·11
	1·3	96·56	96·13	96·80	96·42	88·91	89·03
	1·4	96·37	97·84	94·08	95·90	87·72	88·10
1·2	1·3	95·92	94·29	96·54	95·59	88·71	88·68
	1·4	96·87	97·71	95·52	96·46	88·04	88·05
1·3	1·4	95·54	97·41	93·95	95·33	88·71	88·66
						88·70	88·63

表 1: 基线分值的确定

### 3.2.4 性能评估指标

模型整体性能评估采用准确率 (Acc.) 及宏平均方法计算的精确率 (Pre.)、召回率 (Rec.) 和 F1 值 (F1.)，这些指标作为多分类任务的性能度量标准：

$$\left\{ \begin{array}{l} \text{Accuracy} = \frac{\sum_{i=1}^{N_c} TP_i}{N} \\ \text{Precision}_{macro} = \frac{1}{N_c} \sum_{i=1}^{N_c} \frac{TP_i}{TP_i + FP_i} \\ \text{Recall}_{macro} = \frac{1}{N_c} \sum_{i=1}^{N_c} \frac{TP_i}{TP_i + FN_i} \\ \text{F1-score}_{macro} = \frac{1}{N_c} \sum_{i=1}^{N_c} \frac{2TP_i}{2TP_i + FP_i + FN_i} \end{array} \right. \quad (3.9)$$

这里，TP、TN、FP 和 FN 分别表示真正例、真负例、假正例和假负例的样本数量。下标  $i$  表示这些数值是针对第  $i$  个类别计算的， $N_c$  则代表类别总数。

## 3.3 结果罗列与汇总

### 3.3.1 数据收集

本报告从中心 1 共收集 1708 例样本，经数据预筛选后获得 1664 例有效样本。这些样本进一步划分为 1424 例训练集和 240 例内部验证集。为确保模型在三大类别（正常、非癌病变、癌症）上的评估效果，内部验证集采用分层随机抽样，每类别各含 80 例数据。

外部验证集由以下三部分组成：中心 2 的 93 例样本、中心 3 的 120 例样本及中心 4 的 167 例样本。图 3 展示了数据收集与划分流程。

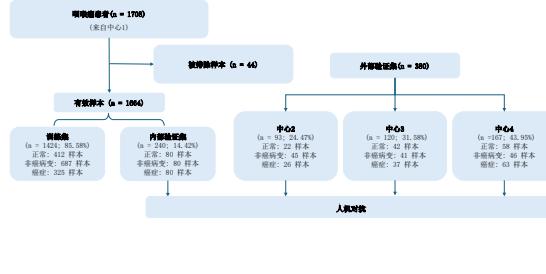


图 3: 数据收集与划分流程图

### 3.3.2 样本临床特征

各数据集中参与者的详细特征（包括年龄、性别及标签分布）如表 2 所示。由表可见，除性别特征外，五个不同数据集在所有其他特征上均存在显著差异。这表明我们需要一个具有强泛化能力的模型，才能在不同数据集上均获得良好的预测性能。这里，假设检验的零假设为各组数据分布相同，运用的检验方法为 Kruskal-Wallis 检验（连续变量）或卡方检验（离散变量）。

特征	训练集 (n=1424)	内部验证集 (n=240)	中心 2 (n=93)	中心 3 (n=120)	中心 4 (n=167)	P 值
年龄	54.94±13.74	57.73±14.30	56.89±13.85	56.44±13.85	57.66±12.52	0.0040
性别						0.19
男	1020(71.63%)	178(74.17%)	72(77.42%)	94(78.33%)	130(77.84%)	
女	404(28.37%)	62(25.83%)	21(22.58%)	26(21.67%)	37(22.16%)	
标签						<0.001
正常	412(28.93%)	80(33.33%)	22(23.66%)	42(35.00%)	58(34.73%)	
非癌病变	687(48.24%)	80(33.33%)	45(48.39%)	41(34.17%)	46(27.54%)	
癌症	325(22.82%)	80(33.33%)	26(27.96%)	37(30.83%)	63(37.72%)	

表 2: 患者基本特征

### 3.3.3 模型在内部验证集上的表现

在本报告中，我们首先讨论模型在三分类任务上的表现（若无特殊说明，模型表现均指三分类表现），然后再讨论模型在五分类任务上的表现。

我们首先在内部验证集上将本研究开发的模型与 ResNet<sup>14</sup>、ViT<sup>23</sup>、Swin Transformer<sup>24</sup> 和

CLIP<sup>21</sup> 等典型人工智能算法进行对比。对于无法直接接收文本输入的模型，我们采用 BERT<sup>25</sup> 来整合文本模态信息。实验结果如表 3 所示：本报告呈现的多模态交互模型整体准确率达 91.67%，精确率 91.67%，召回率 91.87%，F1 分数 91.82%，各项指标均优于对比模型。

模型	训练集				内部验证集			
	Acc.	Pre.	Rec.	F1.	Acc.	Pre.	Rec.	F1.
<b>视觉模型</b>								
Resnet50	96.86	95.95	97.23	96.55	79.17	80.38	79.17	79.05
Resnet101	95.28	94.25	95.86	94.96	81.25	82.67	81.25	81.48
ViT-Base	96.02	95.32	96.00	95.60	81.34	82.45	81.25	81.52
ViT-Large	96.12	95.30	95.93	95.59	82.50	83.05	82.50	82.65
SwinT	95.60	94.84	95.33	95.06	84.17	86.27	84.17	84.32
CLIP-Vision	95.39	94.86	94.65	94.75	84.48	84.46	84.37	84.12
<b>视觉-语言模型</b>								
Resnet50 + BERT	96.65	95.89	97.02	96.42	81.27	84.96	81.25	81.49
Resnet101 + BERT	99.79	99.76	99.76	99.77	82.08	83.32	82.08	82.21
ViT-Large + BERT	98.32	97.79	98.48	98.13	84.58	86.49	84.58	84.85
SwinT + BERT	98.89	96.35	98.43	98.02	85.83	86.26	85.83	85.96
CLIP	99.81	99.22	99.58	98.89	86.67	86.91	86.67	86.74
<b>多模态融合模型</b>								
MFDN (ours)	99.90	99.94	99.87	99.90	91.67	91.67	91.87	91.82

表 3: 多模态模型与传统 AI 模型对比

进一步的，我们想研究多模态模型给出的风险评分本身的质量。我们从定性分析的角度出发，绘制了内部验证集预测风险评分的分布直方图（见图 4），并计算了各类别预测风险评分的均值与方差（见表 4）。

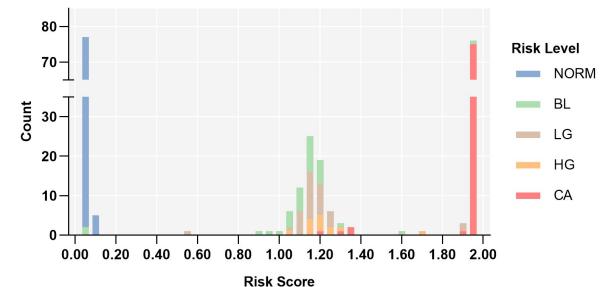


图 4: 内部验证集风险评分直方图

	正常	良性病变	低级别病变	高级别病变	癌症
分数	0.11±0.29	1.17±0.36	1.22±0.22	1.26±0.23	1.91±0.36

表 4: 内部验证集风险评分均值与方差

观察发现，对于癌变类和正常类样本，模型

预测的风险评分大多集中在各类别基线值附近。此外，统计检验（表 5）显示正常类与癌变类相邻分类间的风险评分存在显著分离，这表明模型具有较高的回归准确性。

备择假设	秩和检验		T 检验	
	U-value	P-Value	T-Value	P-Value
NORM< BLL	-7.78	<0.0001	-14.88	<0.0001
BLL<LG	-2.21	0.01	-0.64	0.26
LG<HG	-0.81	0.21	-0.46	0.33
HG<CA	-5.32	<0.0001	-9.62	<0.0001

NORM = 正常, BL = 良性病变, LG = 低级别病变, HG = 高级别病变, CA = 癌症

表 5: 相邻类别风险评分的统计检验

### 3.3.4 模型在所有验证集上的表现

首先，我们绘制出了模型在内部验证集以及三个外部验证集上的混淆矩阵，如图 5 所示：

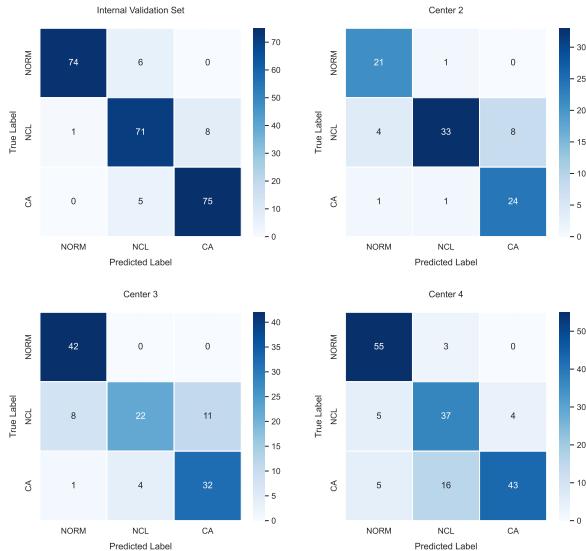


图 5: 模型在所有数据集上的混淆矩阵

接着，我们将我们的模型与 10 位不同资质的医生给出的分类表现做对比（表 6）。其中，10 位医生根据资质分为专家、高年资医生以及低年

资医生，他们接触到的分类信息与模型相同。各分类指标的置信区间由 Bootstrap 给出。

数据集	指标	专家	高年资	低年资	MFDN (ours) <sup>a</sup>
内部验证集	Acc.	86.24 [81.67, 90.79]	83.06 [78.33, 87.50]	77.44 [74.17, 84.58]	<b>91.67</b> [88.33, 95.00]
	Pre.	86.23 [81.71, 90.74]	83.05 [78.69, 87.34]	79.44 [74.37, 84.15]	<b>91.67</b> [88.15, 94.98]
	Rec.	87.49 [83.28, 91.34]	84.61 [80.12, 88.61]	83.04 [78.57, 87.28]	<b>91.87</b> [88.31, 95.15]
	F1.	86.50 [81.94, 90.84]	82.92 [78.14, 87.29]	79.46 [74.12, 84.41]	<b>91.82</b> [88.08, 95.03]
	Acc.	86.22 [79.35, 92.47]	85.30 [77.42, 92.47]	80.65 [72.04, 88.20]	83.87 [76.34, 91.40]
中心 2	Pre.	85.14 [76.90, 92.38]	84.49 [76.88, 91.80]	77.82 [69.20, 85.76]	<b>87.03</b> [80.27, 93.21]
	Rec.	88.43 [82.12, 94.17]	86.90 [79.20, 93.74]	85.05 [76.69, 92.60]	83.35 [75.19, 90.77]
	F1.	86.55 [78.79, 92.80]	85.28 [76.97, 92.18]	79.60 [70.36, 87.60]	82.45 [76.17, 85.25]
	Acc.	80.42 [73.33, 87.50]	77.50 [70.00, 85.00]	73.61 [65.83, 80.83]	<b>80.00</b> [72.50, 86.67]
	Pre.	79.85 [72.83, 86.63]	76.68 [69.40, 83.74]	72.21 [65.66, 78.64]	<b>80.05</b> [73.70, 85.65]
中心 3	Rec.	82.74 [75.99, 88.77]	80.90 [73.68, 87.85]	81.78 [75.10, 87.40]	80.46 [72.85, 87.46]
	F1.	80.28 [72.86, 86.96]	76.86 [68.88, 84.10]	70.82 [63.02, 78.49]	78.66 [70.73, 85.18]
	Acc.	77.53 [70.83, 83.93]	74.60 [67.86, 80.97]	69.84 [62.50, 76.19]	<b>80.36</b> [74.39, 86.31]
	Pre.	78.07 [71.31, 84.25]	75.91 [69.65, 81.87]	71.85 [65.80, 77.67]	<b>80.82</b> [75.00, 86.37]
	Rec.	79.58 [73.74, 85.10]	77.71 [71.41, 83.27]	75.69 [69.59, 81.28]	<b>80.73</b> [74.98, 86.06]
中心 4	F1.	77.58 [70.95, 83.63]	74.29 [67.41, 80.70]	69.48 [62.47, 76.28]	<b>79.82</b> [73.49, 85.44]

<sup>a</sup> 加粗、下划线和斜体分别表示该模型的性能超越专家级喉科医师、高年资喉科医师和低年资喉科医师的水平

表 6: 人机对抗实验结果

由表 6 可以看出，我们的模型在所有数据集上的精确度指标均超越专家组（高出 0.20%-5.44%）。尽管模型在一个外部验证集（中心 2）上表现一般，但在另外两个外部验证集（中心 3 和中心 4）上的表现优于高年资喉科医师。

最后，我们评估了模型在五分类任务中的综合表现。为了公平性，我们首先将模型输出和喉科医师的风险评分调整至最接近的预设基线值，随后计算调整后风险评分与真实标签之间的均方误差 (MSE)。结果如表 7 所示：该模型整体性能显著优于低年资医师 (MSE 降低 0.101-0.204)，达到与高年资医师相当的水平。

组别	喉科医生	内部验证集	中心 2	中心 3	中心 4
专家	A	0.153	0.143	0.117	0.329
	B	0.173	0.108	0.293	0.270
	C	0.114	0.095	0.141	0.143
	D	0.149	0.115	0.148	0.174
	平均值	0.147	0.115	0.175	0.229
高年资	A	0.341	0.120	0.220	0.280
	B	0.147	0.160	0.183	0.220
	C	0.219	0.125	0.254	0.252
	平均值	0.236	0.135	0.219	0.251
	MFDN (ours) <sup>*</sup>	<u>0.082</u>	<u>0.180</u>	<u>0.207</u>	<u>0.239</u>

\* 加粗、下划线和斜体分别表示该模型的性能超越**专家级喉科医师**、**高年资喉科医师**和**低年资喉科医师**的水平。

表 7: 五分类任务人机 MSE 对比

总的来讲，我们的模型在三分类任务上的表现略强于高年资医生，在五分类任务上的表现与高年资医生基本相同，这也说明了五分类任务的难度大于三分类任务。即便如此，我们的模型仍能达到与高年资医生相同的表现，说明多模态信息能够显著提升模型的分类表现。

### 3.4 结论与心得

在本次报告中，我们利用一个多模态的深度学习模型进行咽喉癌的诊断。与传统的 AI 模型不同，本报告所呈现的模型利用了除图像外的文本信息以及患者特征信息，从而提升了模型的性能。不仅如此，我们所呈现的模型的损失函数融合了对比学习的损失函数，并且建立在预训练的 BEIT3 模型基础上。上述所有特点使得我们的模型在真实的外部验证集上拥有略强于高年资医生的分类准确率。

### 3.5 代码可用性

本论文使用的所有代码均为 python 代码，使用 python 版本为 3.12.7。所有代码均存放于：  
<https://github.com/yifanzhang7/neuralnetwork-assignment3>

# 参考文献

- [1] Filippo Amato, Alberto López, Eladia María Peña-Méndez, Petr Vaňhara, Aleš Hampl, and Josef Havel. Artificial neural networks in medical diagnosis. *Journal of Applied Biomedicine*, 11(2):47–58, 2013.
- [2] Manisha M Kasar, Debnath Bhattacharyya, and TH Kim. Face recognition using neural network: a review. *International Journal of Security and Its Applications*, 10(3):81–100, 2016.
- [3] Dires Negash Fente and Dheeraj Kumar Singh. Weather forecasting using artificial neural network. In *2018 second international conference on inventive communication and computational technologies (ICI-CCT)*, pages 1757–1761. IEEE, 2018.
- [4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [5] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.
- [6] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algo-  
rithms. *arXiv preprint arXiv:1708.07747*, 2017.
- [7] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.
- [8] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018.
- [9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [10] Tianyan Zhou, Yong Zhao, and Jian Wu. Resnext and res2net structures for speaker verification. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 301–307, 2021.
- [11] DP Yadav, AS Jalal, Deviram Garlapati, Kaizar Hossain, Ayush Goyal, and Gaurav Pant. Deep learning-based resnext model in phycological studies for future. *Algal research*, 50:102018, 2020.

- [12] Najmul Hasan, Yukun Bao, Ashadullah Shawon, and Yanmei Huang. Densenet convolutional neural networks application for predicting covid-19 using ct image. *SN computer science*, 2(5):389, 2021.
- [13] Jeyaprakash Hemalatha, S Abijah Roseline, Subbiah Geetha, Seifedine Kadry, and Robertas Damaševičius. An efficient densenet-based deep learning model for malware detection. *Entropy*, 23(3):344, 2021.
- [14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [15] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [16] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.
- [17] Jialin Zhou, Ying Xu, Jianmin Liu, Lili Feng, Jinming Yu, and Dawei Chen. Global burden of lung cancer in 2022 and projections to 2050: Incidence and mortality estimates from globocan. *Cancer Epidemiology*, 93:102693, 2024.
- [18] Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, et al. Image as a foreign language: Beit pretraining for vision and vision-language tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19175–19186, 2023.
- [19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [20] Hangbo Bao, Wenhui Wang, Li Dong, Qiang Liu, Owais Khan Mohammed, Kriti Aggarwal, Subhojit Som, Songhao Piao, and Furu Wei. Vlmo: Unified vision-language pre-training with mixture-of-modality-experts. *Advances in Neural Information Processing Systems*, 35:32897–32912, 2022.
- [21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- [22] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *Advances in neural information processing systems*, 33:6827–6839, 2020.
- [23] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiao-

hua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

- [24] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.
- [25] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.