利用深度学习网络进行黑色素瘤检测的皮肤病变分析

165_3160102186_林新迪　　6_3160104014_陈栩淦
19_3160102998_丁李桑
数学科学学院
数学软件

2018-8-26

摘要

黑色素瘤是一种致命的皮肤癌，可以在皮肤表面的皮肤色素细胞中爆发。黑色素瘤导致75%的皮肤癌相关死亡。然而，由于病变和皮肤之间的对比度低、黑素瘤和非黑色素瘤病变之间的视觉相似性等原因，使得黑素瘤的准确识别极具挑战性。即使皮肤科专家使用皮肤病图像进行诊断，专家的正确诊断率估计为75-84%。本实验使用预训练的卷积神经网络（Convolutional Neural Network, CNN）（如AlexNet、Inception v3等）进行转移学习，使用皮肤病变图像进行再培训；同时自己动手写一个CNN，与前面两者进行对比。在这里，我们使用单个CNN展示皮肤病变的分类，直接从图像端对端训练，仅使用像素和疾病标签作为输入。目的是检测图像是否代表良性痣或恶性黑素瘤。研究结果表明，本实验有着较好的分类能力。

关键字： 皮肤病变、深度学习、卷积神经网络、黑色素瘤检测

目录

表格

插图

1 研究的问题

1.1 背景和意义

- 关于黑色素瘤

  皮肤癌是一个主要的公共卫生问题，美国每年有540万新发皮肤癌病例。五分之一的美国人将在其一生中被诊断出患有皮肤恶性肿瘤。尽管黑色素瘤占美国所有皮肤癌的不到5%，但它们约占所有皮肤癌相关死亡的75%，并且仅在美国每年造成10,000多人死亡。2015年，全球黑色素瘤发病率估计超过350,000例，死亡人数近60,000例。虽然死亡率很高，但早期检测至关重要，因为如果在最早阶段检测到，黑素瘤的存活率超过95%。

图 1: 黑色素瘤（Melanoma），来源：ISIC 2017: Skin Lesion Analysis Towards Melanoma Detection

- 关于皮肤镜检查

  由于皮肤表面发生色素沉着病变，黑色素瘤可通过专家目视检查进行早期检测。它也适用于图像分析的自动检测。鉴于高分辨率相机的广泛应用，可以提高我们筛查和检测麻烦病变的能力的算法具有重要价值。因此，许多中心已经开始自己的自动分析研究工作。但是，各机构之间的集中，协调和比较努力尚未实施。

  皮肤镜检查是一种消除皮肤表面反射的成像技术。通过去除表面反射，增强了更深层皮肤的可视化。先前的研究表明，当与专业皮肤科

医生一起使用时，与标准摄影相比，皮肤镜检查提供了改进的诊断准确性。随着用于智能手机的廉价消费者皮肤镜附件开始进入市场，自动化皮肤镜评估算法对患者护理的积极影响的机会增加。

然而，从皮肤镜检查图像中自动识别黑素瘤仍然是一项艰巨的任务，因为它有几个挑战。首先，皮肤损伤与正常皮肤区域之间的低对比度使得难以分割准确的病变区域。其次，黑素瘤和非黑素瘤病变可能具有高度的视觉相似性，导致难以区分黑素瘤病变与非黑素瘤。第三，皮肤状况的变化，例如在肤色，天然毛发或静脉中，患者在颜色和质地等方面产生不同的黑素瘤外观。

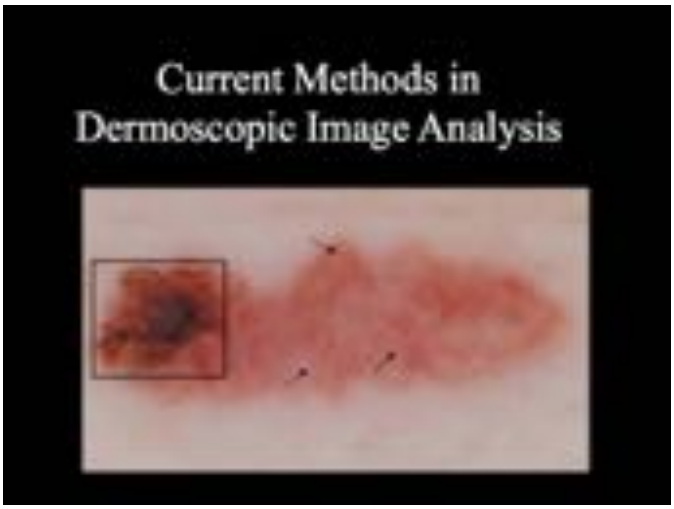

图 2: 皮肤镜（Dermoscopy）， 来源：ISIC 2017: Skin Lesion Analysis Towards Melanoma Detection

1.2 研究现状及发展

随着深度学习算法的不断改进，世界各地的科学家进行了广泛的研究，以产生适当的病变分割结果。深度学习的算法已经被广泛应用到病变分割中，包括Lequan Yu提出的完全卷积残差网络（fully-convolutional residual network，FCRN）用于皮肤病变皮肤镜检查中的分割。[7]Codella等人提出了一种融合卷积神经网络、稀疏编码和支持向量机（SVM）的混合方法来检测黑色素瘤。[1]Kawahara等人采用完全卷积网络提取黑色素瘤识别的多

尺度特征。尽管提出了大量工作，但皮肤病变分割和分类仍然有性能改善的余地。国际皮肤影像协作组织（ISIC）从2016年开始组织黑色素瘤检测挑战，这极大地提高了自动黑色素瘤检测方法的准确性。在ISIC 2017中，宣布了皮肤病变图像的三个处理任务，包括病变分割，皮肤镜特征提取和病变分类。与已经广泛研究的病变分割和分类不同，皮肤镜特征提取是该领域的一项新任务。在ISIC2018中，仍就这个话题提出了挑战任务，包括了三个独立的任务：病变分割、病变属性检测、疾病分类。显然可见，这个皮肤癌病变分割问题，仍然有着很大的发展空间。随着深度学习算法的不断改进和应用，将会有更多的成果出来。

2 实验内容和方案

2.1 实验目的

- 理解深度学习，卷积神经网络转移学习的基本概念
- 熟悉GooLeNet和AlexNet的来源和结构
- 理解转移学习的方法和原理
- 用转移学习的方法实现MATLAB下良性和黑色素瘤的分类
- 动手写简单的卷积神经网络，实现MARLAB下良性和黑色素瘤的分类

2.2 研究内容

- 文献查阅，背景熟悉：阅读Project论文，查询与阅读神经网络识别皮肤病的相关论文，了解黑色素瘤的概念和临床诊断步骤和现有研究成果
- 环境配置：配置MATLAB神经网络工具箱并下载已训练好的卷积神经网络，为算法实现作好准备
- 获取数据：获取数据并根据实验需求对数据进行预处理
- 算法实现一：利用转移学习的方法调整GooLeNet和AlexNet的网络结构，重新训练，测试结果

- 算法实现二：动手写卷积神经网络进行训练，测试结果
- 实验探究：在算法已经实现后，调整参数并探究其对模型分类效果的影响

2.3 关键问题

- 训练数据是实验的基础，如何获取有效准确的数据
- 训练神经网络需耗费大量计算力，如何提高运算速度
- 神经网络结构复杂，如何探究参数对其训练效果的影响

2.4 方法和技术

2.4.1 深度学习

- 机器学习

机器学习（machine learning）是人工智能的一个分支，它致力于研究如何通过计算的手段，利用经验（experience）来改善计算机系统自身的性能。通过从经验中获取知识，机器学习算法摒弃了人为向机器输入知识的操作，转而凭借算法自身来学到所需所有知识。对于传统机器学习算法而言，“经验”往往对应以“特征”（feature）形式存储的“数据”（data），传统机器学习算法所做的事情便是依靠这些数据产生“模型”（model）。

- 特征

设计特征有助于算法学到更好的模型，这是非常困难的事情，特别是随着机器学习的任务越来越复杂多变。

既然模型学习的任务可以通过机器自动完成，那么特征学习这个任务自然完全可以通过机器自己实现。于是，人们尝试将特征学习这一过程也用机器自动的“学”出来，这便是“表示学习”（representation learning）。“深度学习”便是表示学习中的一个经典代表。

- 深度学习

深度学习以数据的原始状态作为算法的输入，经过算法层层抽象将原始数据逐层抽象为自身任务所需的最终特征表示，最后以特征到任务目标

的映射（mapping）作为结束，从原始数据到最终任务目标，“一气呵成”并无夹杂任何人为操作。相比传统机器学习算法仅学得模型这一单一“任务模块”而言，深度学习除了模型学习，还有特征学习、特征抽象等任务模块的参与，借助多层任务模块完成最终学习任务，故称其为“深度”学习。深度学习中的一类代表算法是神经网络算法，包括深度置信网络（deep belief network）、递归神经网络（recurrent neural network）和卷积神经网络（convolution neural network，CNN）。特别是卷积神经网络，目前在计算机视觉、自然语言处理、医学图像处理等领域有着卓越的成就。本实验也正是使用了卷积神经网络。

2.4.2 卷积神经网络

卷积神经网络（convolution neural network，CNN）是一类特殊的人工神经网络，区别于神经网络其他模型（如，递归神经网络等），其最主要的特点是卷积运算操作（convolution operators）。因此，CNN在诸多领域应用特别是图像相关任务上表现优异，诸如，图像分类、图像语义分割、图像检索、物体检测等计算机视觉问题。

- 基本结构

总体来说，卷积神经网络是一种层次模型，其输入是原始数据，如RGB图像、原始音频数据等。卷积神经网络通过卷积（convolution)操作、汇合（pooling）操作和非线性激活函数（non-linear activation function）映射等一系列操作的层层堆叠，将高层语义信息逐层由原始数据输入层中抽取出来，逐层抽象，这一过程便是“前馈运算”（feed-forward）。 其中，不同类型操作在卷积神经网络中一般称作“层”：卷积操作对应“卷积层”，汇合操作对应“汇合层”等等。最终，卷积神经网络的最后一层将其目标任务（分类、回归等）形式化为目标函数。通过计算预测值与真实值之间的误差或损失（loss），凭借反向传播算法（back-propagation）将误差或损失由最后一层逐层向前反馈（back-forward），更新每层参数，并在更新参数后再次前馈，如此往复，直到网络模型收敛，从而达到模型训练的目的。

比如，具体地，在图像处理中，卷积神经网络的数据层通常是RGB颜色空间的图像：H行，W列，3个通道（分别为R, G, B），在此记作x^1。x^1经过第一层操作（参数为ω^1）可得x^2，x^2作为第二层操作层ω^2的输入，可得x^3……直到第L层，以损失函数的计算（z）作为过程的结束。

$$x^1 \to \omega^1 \to \omega^2 \to \cdots \to x^L \to \omega^L \to z. \tag{1}$$

若y是输入x^1对应的真实标记，则损失函数表示为

$$z = \mathscr{L}(x^L, y), \tag{2}$$

其中，函数$\mathscr{L}(\cdot)$中的参数即ω^L。对于层中的特定操作，可为单独卷积操作、汇合操作、非线性映射或者其他操作/变化，也可以是不同形式的操作/变换的组合。

损失函数经常采用L_2损失函数

$$z = \mathscr{L}(x^L, y) = \frac{1}{2}||x^L - y||^2$$

或者交叉熵（cross entropy）损失函数

$$z = \mathscr{L}(x^L, y) = -\sum_i y_i log(p_i),$$

其中$p_i = \frac{exp(x_i^L)}{\sum_{j=1}^{C} exp(x_j^L)}$.

- 前馈运算

卷积神经网络的前馈运算（feed-forward）比较直观。假设此时网络已训练完毕，各个参数已收敛到最优解，此时可用此网络进行图像类别的预测。预测过程实际上就是一次网络的前馈运算：将测试机图像作为网络输入x^1传入网络，经过多层操作，直至输出$x^L \in \mathbb{R}^C$。如此，可通过下士得到输入图像x^1对应的预测标记：

$$\arg\min_i \ x_i^L \tag{3}$$

- 反馈运算

由于神经网络模型不仅是非凸函数，而且非常复杂，这带来了优化求解的困难。所以才多采用随机梯度下降法（Stochastic Gradient Descent，SGD）和误差反向传播（error back propogation，BP）进行模型参数更新。

算法 1 随机梯度下降法（Stochastic Gradient Descent，SGD）

输入:

输出:

Randomly shuffle dataset;
repeat
    for $i = 1, ..., m$ **do**
        $\theta_j = \theta_j + (y^i - h_\theta(x^i))x_j^i, j = 0, ...n.$
    end for
until 达到停止条件

2.4.3 转移学习

- 转移学习

跟传统的监督式机器学习算法相比，深度神经网络目前最大的劣势是所需的资源极大。尤其是当我们在尝试处理现实生活中诸如图像识别、声音辨识等实际问题的时候。一旦你的模型中包含一些隐藏层时，增添多一层隐藏层将会花费巨大的计算资源。

因此，我们可以使我们在他人训练过的模型基础上进行小改动便可投入使用，这也就是“转移学习”。

神经网络需要用数据来训练，它从数据中获得信息，进而把它们转换成相应的权重。这些权重能够被提取出来，迁移到其他的神经网络中，我们“迁移”了这些学来的特征，就不需要从零开始训练一个神经网络了 。

- 预训练模型

预训练模型(pre-trained model)是前人为了解决类似问题所创造出来的模型。我们在解决问题的时候，不用从零开始训练一个新模型，可以从在类似问题中训练过的模型入手。一个预训练模型可能对于我们的应用不是完全的对的上，但是可以为我们节省大量的时间和资源。

当在训练经网络的时候，我们希望网络能够在多次正向反向迭代的过程中，找到合适的权重。通过使用之前在大数据集上经过训练的预训练模型，我们可以直接使用相应的结构和权重，将它们应用到我们正在面对的问题上。这样一来，就直接将预训练的模型“转移”到我们正在应对的问题上。

算法 2 误差反向传播（error back propogation，BP）

输入: 训练集$D=\{(\mathbf{x}_k,\mathbf{y}_k)\}_{k=1}^{m}$; 学习率$\eta$

输出: 连接权值或阈值确定的多层前馈神经网络

function $BP(D,\eta)$

在$(0,1)$范围内随机初始化网络中的所有连接权值和阈值

repeat

for all$(\mathbf{x}_k,\mathbf{y}_k)\in D$ **do**

计算当前样本输出$\hat{\mathbf{y}}_k=f(\beta_j-\theta_j)$

计算输出神经元的梯度

$$\begin{aligned} g_j &= -\frac{\partial E_k}{\partial \hat{y}^k}\cdot\frac{\partial \hat{y}^k}{\partial \beta_j} \\ &= -(\hat{y}_j^k-y_j^k)f'(\beta_j-\theta_j) \\ &= \hat{y}_j^k(1-\hat{y}_j^k)(\hat{y}_j^k-y_j^k) \end{aligned}$$

计算隐层的神经元梯度项

$$\begin{aligned} e_h &= -\frac{\partial E_k}{\partial b_h}\cdot\frac{\partial b_h}{\alpha_h} \\ &= -\sum_{j=1}^{\ell}\frac{\partial E_k}{\partial \beta_j}\cdot\frac{\partial \beta_j}{b_h}f'(\alpha_h-\gamma_h) \\ &= \sum_{j=1}^{\ell} w_{hj}g_j f'(\alpha_h-\gamma_h) \\ &= b_h(1-b_h)\sum_{j=1}^{\ell} w_{hj}g_j \end{aligned}$$

更新权值

$$\Delta w_{hj}=\eta g_j b_h$$

$$\Delta v_{ih}=\eta e_h x_i$$

更新阈值

$$\Delta\theta_j=-\eta g_j$$

$$\Delta\gamma_h=-\eta e_h$$

end for

until 达到停止条件

end function

当预训练模型已经训练得很好，我们就不会在短时间内去修改过多的权重，在迁移学习中用到它的时候，往往只是进行微调(fine tune)。

- 微调模型

  使用微调模型的方法，具体如下：

  - 特征提取

    我们可以将预训练模型当做特征提取装置来使用。具体的做法是，将输出层去掉，然后将剩下的整个网络当做一个固定的特征提取机，从而应用到新的数据集中。

  - 采用预训练模型的结构

    我们还可以采用预训练模型的结构，但先将所有的权重随机化，然后依据自己的数据集进行训练。

  - 训练特定层，冻结其他层

    另一种使用预训练模型的方法是对它进行部分的训练。具体的做法是，将模型起始的一些层的权重保持不变，重新训练后面的层，得到新的权重。在这个过程中，我们可以多次进行尝试，从而能够依据结果找到frozen layers和retrain layers之间的最佳搭配。

- 使用预训练模型的场景和方法

  - 场景一：数据集小，数据相似度高

    在这种情况下，因为数据与预训练模型的训练数据相似度很高，因此我们不需要重新训练模型。我们只需要将输出层改制成符合问题情境下的结构就好。

  - 场景二：数据集小，数据相似度不高

    在这种情况下，我们可以冻结预训练模型中的前k个层中的权重，然后重新训练后面的n-k个层，当然最后一层也需要根据相应的输出格式来进行修改。

    因为数据的相似度不高，重新训练的过程就变得非常关键。而新数据集大小的不足，则是通过冻结预训练模型的前k层进行弥补。

  - 场景三：数据集大，数据相似度不高

在这种情况下，因为我们有一个很大的数据集，所以神经网络的训练过程将会比较有效率。然而，因为实际数据与预训练模型的训练数据之间存在很大差异，采用预训练模型将不会是一种高效的方式。

因此最好的方法还是将预处理模型中的权重全都初始化后在新数据集的基础上重头开始训练。

– 场景四：数据集大，数据相似度高

这就是最理想的情况，采用预训练模型会变得非常高效。最好的运用方式是保持模型原有的结构和初始权重不变，随后在新数据集的基础上重新训练。

2.4.4 数据增强

增加数据集大小和模型普遍性的强大而通用的工具是数据增强。从本质上讲，数据增强是通过转换人为增加数据集大小的过程。

数据增强的方法有很多，包括旋转、剪切、平移、水平镜像、垂直镜像、拉伸、随机裁剪等。

大多数深度学习库都具有用于典型转换的现成功能。我们需要确认哪些变换对我们的数据是有意义的。

2.5 实验方案

2.5.1 实验手段

下面是本实验的主要过程。[2] [4] [5] [6]

- 数据集：ISIC Archive

用于训练，验证和测试网络的图像来自开源的ISIC Dermoscopic Archive。International Skin Imaging Collaboration (ISIC) 提供了最大的公开可用的皮肤病变质控皮肤镜图像集。目前，ISIC档案馆包含13,000多个皮肤镜像，这些图像是从国际领先的临床中心收集的，并从每个中心的各种设备中获取。广泛和国际参与图像贡献旨在确保代表性临床相关样本。初始数据集主要来自美国，国际标准行业分类持续承诺和国际合作者提供的其他数据集。

在选择数据时，仅选择标记为恶性（malignant）黑素瘤或良性（benign）痣的图像。这排除了少数具有几种不同诊断的图像。数据集由3183个图像（1523个良性痣和1660个恶性黑素瘤）。每个图像都通过组织病理学进行标记，组织病理学是指病理学家对手术提取的标本进行活组织检查或检查。没有根据有关皮肤病变患者的背景信息进行区分。

- 数据集拆分

我们将数据集拆分为三个子数据集：训练数据集，验证数据集和测试数据集。训练数据的比例为60%，验证和测试数据的比例为所有图像的20%。这些被随机分配到数据记录中。再加上两类的数量差不多，我们在拆分数据集的时候保证了图片来自恶性图像类的概率近似为$p = 0.5$。

ISIC数据集的另一个问题是每个图像的可变大小。为了对网络进行无故障培训，我们对图像进行了预处理。图像的最大中心平方，并缩小到对应CNN架构所需的数据量大小。

- 数据增强

通过增加数据集进行人工增加，以获得更多样化的神经网络训练样例。在我们的方法中，我们使用许多不同的增强，每个增强已经一次应用于一个图像。

- 数据增强的方法：
  - 水平镜像
  - 垂直镜像
  - 水平随机平移，区间为[-30, 30]
  - 垂直随机平移，区间为[-30, 30]

- 训练算法

我们使用AlexNet CNN架构和Google的Inception v3 CNN架构，分别进行转移学习的训练。我们从CNN预训练模型中删除最终的分类图层，并在最后添加新的分类输出图层，使得最终输出的类别数目为2类。

同时对于规模较小的AlexNet CNN架构，我们对所有图层中的参数进行微调；对于Inception v3 CNN架构，我们冻结前几层结构，只微调最后一层或者两层结构。

在训练期间，我们将每个图像的大小调整为对应CNN架构所需的像素尺寸，并利用了预训练网络学习的自然图像特征。

CNN使用反向传播进行训练，使用相同的全局学习率0.001，为较小的值以减慢尚未冻结的已传输图层中的学习速度，同时增大完全连接层的学习速率因子，以加快新最终层中的学习速度。学习速率设置的这种组合导致新层中的快速学习，中间层中的较慢学习，以及在较早的冻结层中没有学习。在进行转学习时，无需为多个时期进行训练，所以设置了6个时期，为整个训练数据集的完整训练周期。并且指定小批量梯度下降法中小批量大小为6个数据图像。

对于DIY的CNN，我们采用传统的方式，以一定InputSize的图像进行传入，经过3通道降维的卷积层、最大池化层，把3维图像提取特征到2维，之后经过多组卷积层、ReLU层、池化层，最后输出一个一个(3×3) 的图像矩阵，最后使用softmax分类器对所得图像矩阵进行处理，得到图像分类结果。

其中利用梯度下降法和误差反向传播算法进行反馈运算。每次处理为1个数据图像。

在训练过程中，使用训练集进行训练，使用验证数据集进行验证网络。

- 测试集分类

最终，当训练结束后，使用已训练完成的微调网络对测试图像集进行分类，并计算分类准确度。不关注单独某一类的分类性能是比较算法的最通用方法。 因此，我们使用经验测量和准确性来区分不同类别的正确标签的数量：

T_benign = true benign: number of examples predicted benign that are actually benign;

F_benign = false benign: number of examples predicted benign that are actually malignant;

$T\_malignant$ = true malignant: number of examples predicted malignant that are actually malignant;

$F\_malignant$ = false malignant: number of examples predicted malignant that are actually benign.

我们采用下面的指标，分别来刻画算法的有效性:

准确度Accuracy：指分类器正确分类的记录总数。 分类器的准确性定

义为由模型正确分类的测试集元组的百分比。

$$Accuracy = \frac{T\_benign + T\_malignant}{T\_benign + T\_malignant + F\_benign + F\_malignant}$$

敏感性Sensitivity：指真正的良性比率。

$$Sensitivity = \frac{T\_benign}{T\_benign + F\_malignant}$$

特异性Specificity：表示测试或诊断方法为未患病的患者设置正确（即恶性）诊断的速率。

$$Specificity = \frac{T\_malignant}{T\_malignant + F\_benign}$$

平衡精度Balanced accuracy（BACC）：平衡精度，可以定义为在任一类上获得的平均精度。

$$Balanced\_accuracy = \frac{\frac{T\_benign}{T\_benign+F\_benign} + \frac{T\_malignant}{T\_malignant+F\_malignant}}{2}$$

精度Precision：在预测的结果中良性的比例。

$$Precision = \frac{T\_benign}{T\_benign + F\_benign}$$

F-测量F-measure:：将Precision和Recall两者结合为调和平均值。

$$F\_measure = \frac{2 * Precision * Sensitivity}{Precision + Sensitivity}$$

2.5.2 转移学习的技巧

这一节将展示AlexNet CNN架构和Google的Inception v3 CNN架构在转移学习和微调的具体使用

- AlexNet CNN架构

AlexNet CNN网络的结构如图3，一共有五个卷积层和三个完全连接的层，可见图4。

该网络的最后三层配置为1000个类。针对新的分类问题微调这三个层。从预训练网络中提取除最后三层之外的所有层。通过用完全连接的图层，softmax图层和分类输出图层替换最后三个图层，将图层转移到新的分类任务，将完全连接的图层设置为2类。这就完成了AlexNet网络的微调。

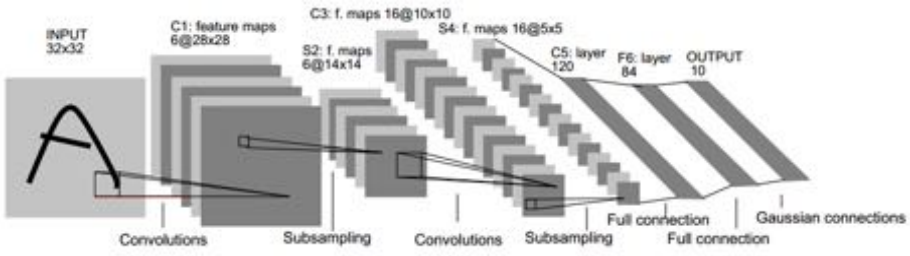

图 3: AlexNet CNN架构的结构

25x1 Layer array with layers:

1	'data'	Image Input	227x227x3 images with 'zerocenter' normalization
2	'conv1'	Convolution	96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0]
3	'relu1'	ReLU	ReLU
4	'norm1'	Cross Channel Normalization	cross channel normalization with 5 channels per element
5	'pool1'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
6	'conv2'	Convolution	256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]
7	'relu2'	ReLU	ReLU
8	'norm2'	Cross Channel Normalization	cross channel normalization with 5 channels per element
9	'pool2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
10	'conv3'	Convolution	384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
11	'relu3'	ReLU	ReLU
12	'conv4'	Convolution	384 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
13	'relu4'	ReLU	ReLU
14	'conv5'	Convolution	256 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
15	'relu5'	ReLU	ReLU
16	'pool5'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
17	'fc6'	Fully Connected	4096 fully connected layer
18	'relu6'	ReLU	ReLU
19	'drop6'	Dropout	50% dropout
20	'fc7'	Fully Connected	4096 fully connected layer
21	'relu7'	ReLU	ReLU
22	'drop7'	Dropout	50% dropout
23	'fc8'	Fully Connected	1000 fully connected layer
24	'prob'	Softmax	softmax
25	'output'	Classification Output	crossentropyex with 'tench' and 999 other classes

图 4: AlexNet CNN架构的各个层详细信息

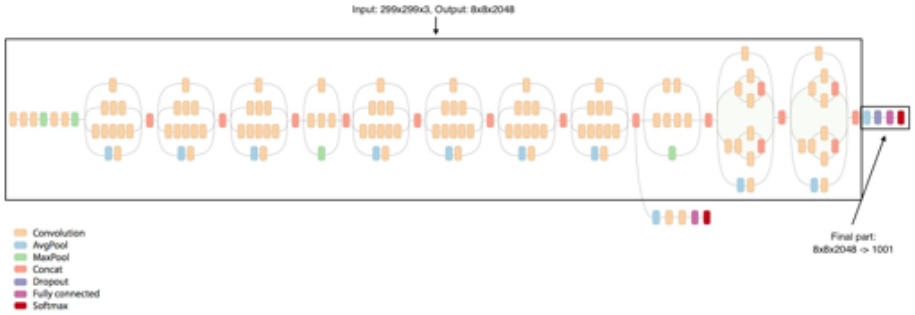

图 5: Inception v3 CNN架构的结构

- Inception v3 CNN架构

Inception v3 CNN是一个有着316层的卷积神经网络，其结构可见图5。

为了采用这个预训练网络进行转移学习以对新图像进行分类，我们替换网络的最后三层，'loss3-classifier'， 'prob'，和 'output'，这三层包含如何在网络中提取成各个类别的概率和标签的功能结合起来的信息。并且在图层图中添加三个新图层：完全连接图层，softmax图层和分类输出图层。并将网络中剩余的最后一个传输层（'pool5-drop_7x7_s1'）连接到新层，如图6。

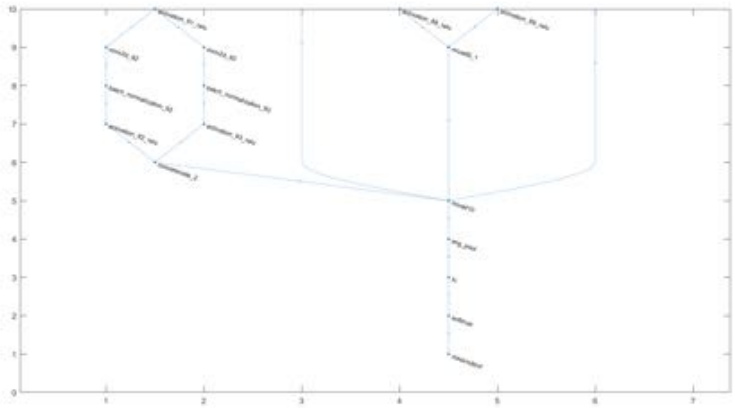

图 6: Inception v3的新图层及其连接结构

由于Inception v3网络非常大，所以我们采用冻结前280层的权重的方法来进行微调，将这些层中的学习速率设置为零。前280层包含了mixed9模块以前的所有图层，如图7。

2.5.3 CNN-DIY结构

采用卷积神经网络（CNN），并通过对网络进行训练来识别皮肤癌的良性与恶性。

卷积神经网络的具体构成框架如下。[3]

首先输入一张$(299 \times 299 \times 3)$ 的图片（图片的像素点为(299×299)，3通道，图片每个像素点的数值映射为$[0, 1]$之间的数值），接着用$(8 \times 8 \times 3)$，步伐($step$)为1的卷积核对图像进行作用，并使用ReLU函数进行整流，这时得

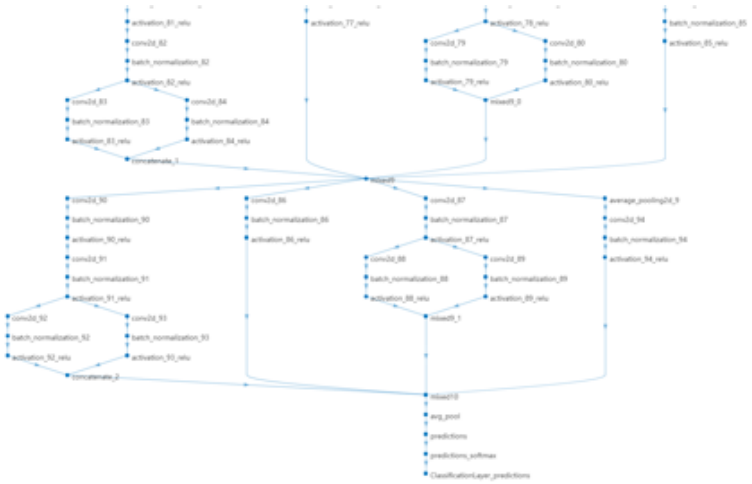

图 7: Inception v3的mixed9以后的图层结构

到的图像矩阵已经转变为二维，再使用最大池化（$max-pooling$）对提取的特征进行压缩，之后再用(7×7) 卷积核构成的卷积层与ReLU 函数对图像矩阵进行处理，再依次通过max-pooling池化层，(7×7)卷积层，ReLU层，max-pooling池化层，(5×5)卷积层，ReLU层，max-pooling池化层，(5×5)卷积层，ReLU层，max-pooling池化层，(3×3) 卷积层，ReLU层，输出一个(3×3) 的图像矩阵，最后使用softmax分类器对所得图像矩阵进行处理，得到图像分类结果。

- 程序内容

本程序由多个子程序组成，各子程序的具体功能如下，具体代码见4.3节。

- $Conv\_Layer\_2d$：二维卷积层，输入一个二维矩阵，二维卷积核与一个偏置，使用ReLU函数整流，输出卷积运算所得的新矩阵；
- $Conv\_Layer\_3d$：二维卷积层，输入一个三维矩阵，三维卷积核与一个偏置，使用ReLU函数整流，输出卷积运算所得的新矩阵；
- $max\_pooling$：二维池化层，输入一个二维矩阵，输出经最大池化作用后的新矩阵；
- $softmax\_layer$：softmax分类器，输入一个二维矩阵，相同大小的二

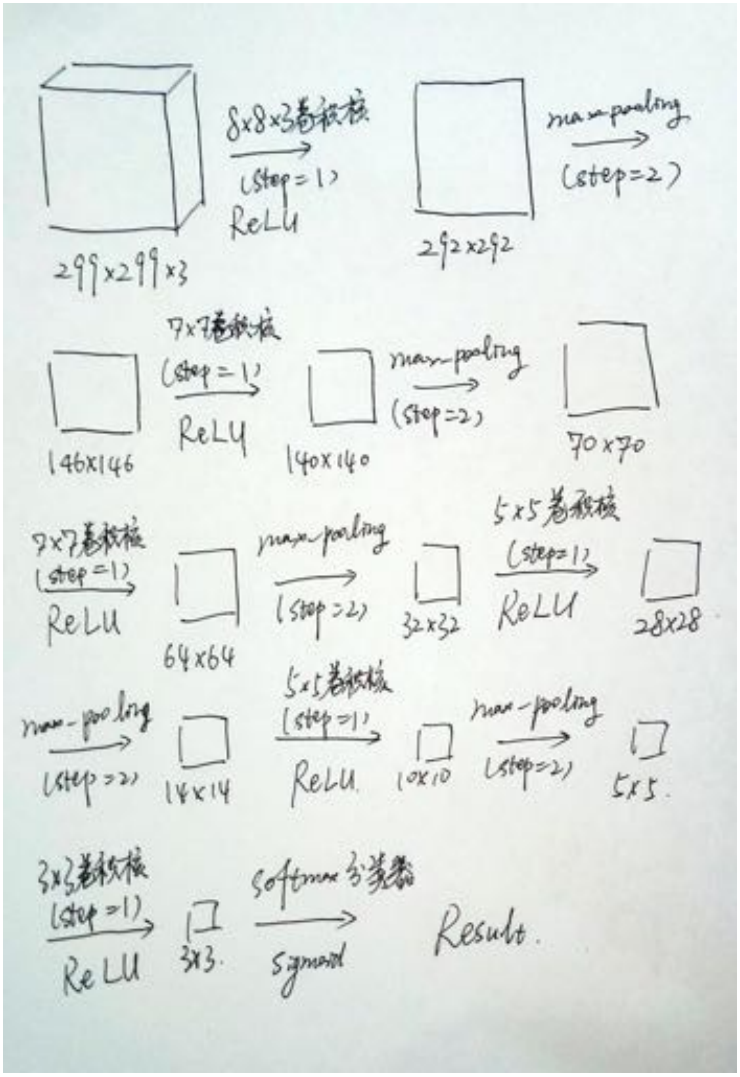
299x299x3
8x8x3卷积核
(step=1)
ReLU
292x292
max-pooling
(step=2)
146x146
7x7卷积核
(step=1)
ReLU
140x140
max-pooling
(step=2)
70x70
7x7卷积核
(step=1)
ReLU
64x64
max-pooling
(step=2)
32x32
5x5卷积核
(step=1)
ReLU
28x28
max-pooling
(step=2)
14x14
5x5卷积核
(step=1)
ReLU
10x10
max-pooling
(step=2)
5x5
3x3卷积核
(step=1)
ReLU
3x3
Softmax分类器
Sigmoid
Result

图 8: CNN-DIY结构

维卷积核与一个偏置，在进行卷积计算后，使用sigmoid函数将运算结果投影到[0, 1]区间；

- $err\_backward\_2d$：对二维卷积层进行误差反向传播；
- $err\_backward\_3d$：对三维卷积层进行误差反向传播；
- $err\_backward\_2d\_softmax$：对softmax分类器进行误差反向传播；
- $err\_backward\_3d\_easy$：对三维卷积层进行误差反向传播，但只计算权重，偏置上的误差传播，以减少实际计算量；
- SGD：利用随机梯度下降法对神经网络进行训练，训练使用的卷积神经网络如上一节所说明；
- CNN_net：利用卷积神经网络对图像进行识别；
- $percision$：脚本文件，训练神经网络，并计算神经网络的正确率。

3 实验结果和结论

3.1 实验结果

3.1.1 Alexnet的转移学习

- 初始图像样本

  在读取训练集之后，随机挑选16个训练集图像进行可视化，大部分病变区域较为集中，肉眼能够分辨较大差异，如图。

- 训练进程与结果

  图10是Alexnet转移学习的训练进程与结果图像，图像中有训练时所选择的主要参数以及训练时的正确率和损失函数的随训练循环次数和周期的变化图像。

- 测试结果样本

  利用训练好的卷积神经网络对随机四个样本进行分类并将结果可视化，见图11。

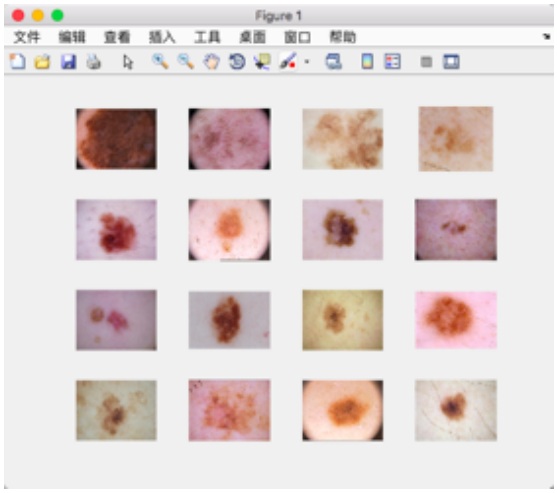

图 9: Alexnet网络初始图像样本

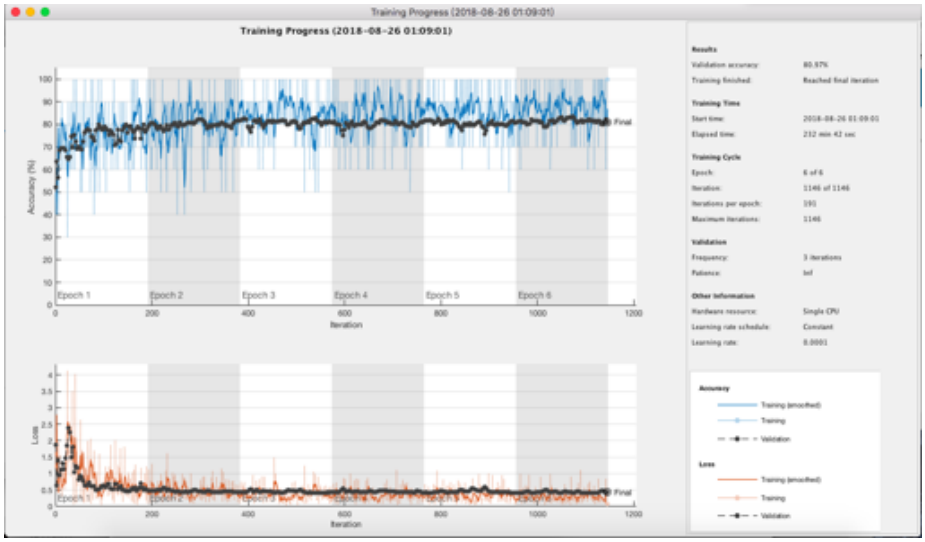

图 10: Alexnet训练进程与结果

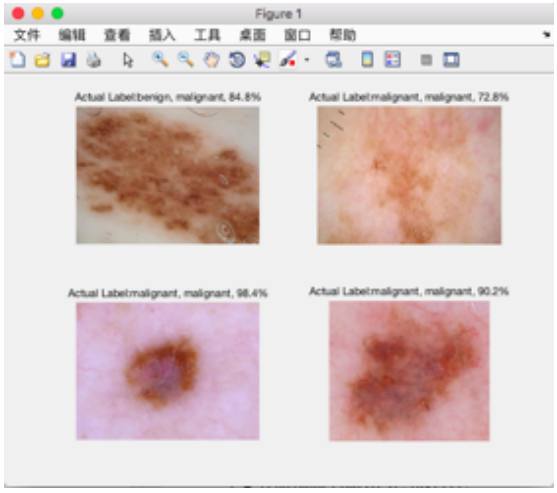

图 11: Alexnet训练结果样本

3.1.2 inceptionv3的转移学习

- 初始图像样本

  在读取训练集之后，随机挑选16个训练集图像进行可视化，大部分病变区域较为集中，肉眼能够分辨较大差异，如图12。

- 训练进程与结果

  图13是inceptionv3转移学习的训练进程与结果图像，图像中有训练时所选择的主要参数以及训练时的正确率和损失函数的随训练循环次数和周期的变化图像。

- 测试结果样本

  利用训练好的卷积神经网络对随机四个样本进行分类并将结果可视化，见图14。

3.2 测试结果指标

针对测试结果我们采用五个指标衡量模型的好坏，分别为准确率（Accuracy），敏感性（Sensitivity），特异性（Specifivity）以及平衡准确率（Balancedaccuracy）。

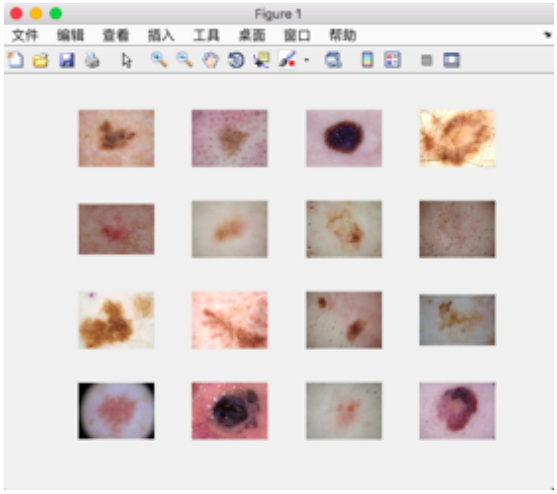

图 12: inceptionv3网络初始图像样本

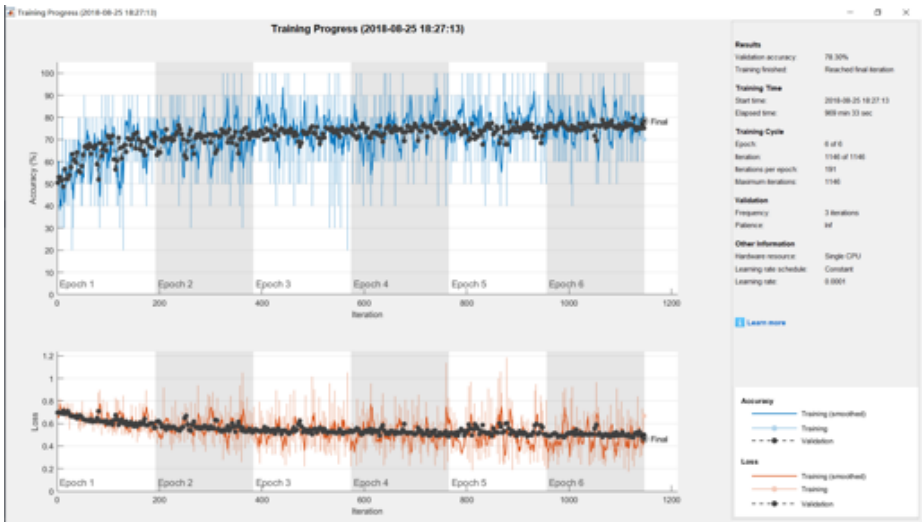

图 13: inceptionv3训练进程与结果

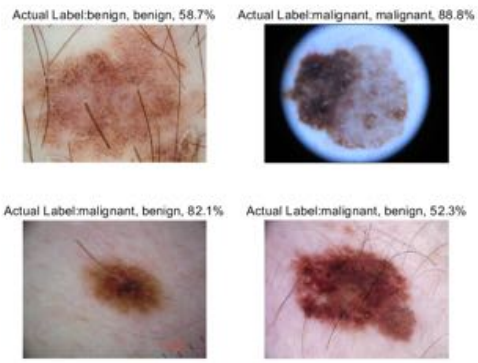

图 14: inceptionv3训练结果样本

T_benign = true benign: number of examples predicted benign that are actually benign;

F_benign = false benign: number of examples predicted benign that are actually malignant;

$T\_malignant$ = true malignant: number of examples predicted malignant that are actually malignant;

$F\_malignant$ = false malignant: number of examples predicted malignant that are actually benign.

我们采用下面的指标，分别来刻画算法的有效性:

准确度Accuracy：指分类器正确分类的记录总数。 分类器的准确性定义为由模型正确分类的测试集元组的百分比。

$$Accuracy = \frac{T\_benign + T\_malignant}{T\_benign + T\_malignant + F\_benign + F\_malignant}$$

敏感性Sensitivity：指真正的良性比率。

$$Sensitivity = \frac{T\_benign}{T\_benign + F\_malignant}$$

特异性Specificity：表示测试或诊断方法为未患病的患者设置正确（即恶性）诊断的速率。

$$Specificity = \frac{T\_malignant}{T\_malignant + F\_benign}$$

CNN	Accuracy	Sensitivity	Specificity	Balancedaccuracy	Precision	F-measure
Alexnet	0.8289	0.8403	0.8195	0.8274	0.7934	0.8162
Inception v3	0.7865	0.7649	0.8082	0.7870	0.8000	0.7821
CNN-DIY	0.6272	0.6008	0.6564	0.6285	0.6586	0.6284

表 1: 测试结果指标

平衡精度Balanced accuracy（BACC）：平衡精度，可以定义为在任一类上获得的平均精度。

$$Balanced\_accuracy = \frac{\frac{T\_benign}{T\_benign+F\_benign} + \frac{T\_malignant}{T\_malignant+F\_malignant}}{2}$$

精度Precision：在预测的结果中良性的比例。

$$Precision = \frac{T\_benign}{T\_benign + F\_benign}$$

F-测量F-measure:：将Precision和Recall两者结合为调和平均值。

$$F\_measure = \frac{2 * Precision * Sensitivity}{Precision + Sensitivity}$$

我们从表1中可以看到，最好的网络是Alexnet，其次是Inception v3，最后是CNN-DIY。前两者都是转移学习，利用已训练好的神经网络进行微调，结果也都在80%左右。而最后一个是自己动手写的卷积神经网络，总体准确率偏低，在60%左右，而且更耗时间。可见转移学习的方法，能够有效的提高卷积神经网络的准确率。

3.3 CNN-DIY模型检验

对CNN-DIY训练网络识别结果进行检验，判断它是否确实具备辅助识别的能力。

取原假设：H_0：训练网络的正确识别率$\eta \leq \frac{1}{2}$，备择假设：H_1：训练网络的正确识别率$\eta > \frac{1}{2}$，显著性水平$\alpha = 0.01$。

训练网络对训练集中955个图像进行识别，成功识别k个图像的概率为$\frac{C_n^k}{2^n}$，设ξ为成功识别的图像个数，它是一个随机变量，其分布近似于正态分布$N(955\eta, 955\eta(1-\eta))$，原假设的接受域为

$$\bar{D} = x : \frac{x - 955\eta}{\sqrt{955\eta(1-\eta)}} \leq \Phi^{-1}(1-\alpha)$$

由于D关于η单调递减，只须要讨论$\eta = \frac{1}{2}$ 的情况即可。将$\alpha = \frac{1}{2}$代入，这时得到拒绝域为$D = x : x \geq 526$，而测试所得正确图像个数为599¿526，因此在显著性水平$\alpha = 0.01$的条件下拒绝原假设，接受备择假设，训练网络显著地有$> \frac{1}{2}$的识别效率。

4 MATLAB代码

4.1 Alexnet的转移学习

```
clc;
clear;

%% Load Data
    imds = imageDatastore('data', ...
        'IncludeSubfolders',true, ...
        'LabelSource','foldernames');
    imds = shuffle(imds);

    % Divide the data into training, validation and testing data sets.
    % Use 60% for training, 20% for validation and 20% for testing.
    [imdsTrain,imdsValidation,imdsTest] = splitEachLabel(imds,0.6,0.2,'r

    % Display some sample images.
    numTrainImages = numel(imdsTrain.Labels);
    idx = randperm(numTrainImages,16);
    figure
    for i = 1:16
        subplot(4,4,i)
        I = readimage(imdsTrain,idx(i));
        imshow(I)
    end

%% Load Pretrained Network
```

```
26      net = alexnet;
27      inputSize = net.Layers(1).InputSize;
28
29      % Remove the last three layers of alexnet
30      layersTransfer = net.Layers(1:end-3);
31
32      % Add three new layers, fullyConnectedLayer, softmaxLayer,
33      % classificationLayer
34      numClasses = numel(categories(imdsTrain.Labels));
35      layers = [
36          layersTransfer
37          fullyConnectedLayer(numClasses,'WeightLearnRateFactor',20,'BiasL
38          softmaxLayer
39          classificationLayer];
40
41  %% Data augmentation
42      % Use an augmented image datastore to automatically resize the images
43      % Specify additional augmentation operations to perform on the images
44      % Data augmentation helps prevent the network from overfitting
45      % and memorizing the exact details of the training images.
46
47      pixelRange = [-30 30];
48      imageAugmenter = imageDataAugmenter( ...
49          'RandXReflection',true, ...
50          'RandXTranslation',pixelRange, ...
51          'RandYTranslation',pixelRange);
52      augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain, ...
53          'DataAugmentation',imageAugmenter);
54
55      augimdsValidation = augmentedImageDatastore(inputSize(1:2),imdsValida
56      augimdsTest = augmentedImageDatastore(inputSize(1:2),imdsTest);
57
58  %% Train Network
```

```
    options = trainingOptions('sgdm', ...
    'MiniBatchSize',10, ...
    'MaxEpochs',6, ...
    'InitialLearnRate',1e-4, ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',3, ...
    'ValidationPatience',Inf, ...
    'Verbose',false, ...
    'Plots','training-progress');

    netTransfer = trainNetwork(augimdsTrain,layers,options);
%% Classify Testing Images
    % Classify the testing images using the fine-tuned network,
    % and calculate the classification accuracy.
    [YPred,scores] = classify(netTransfer,augimdaTest);
    T_benign = sum(YPred == imdsTest.Labels ...
                    & imdsTest.Labels == "benign");
    T_malignant = sum(YPred == imdsTest.Labels ...
                    & imdsTest.Labels == "malignant");
    F_benign = sum(YPred ~= imdsTest.Labels ...
                    & imdsTest.Labels == "benign");
    F_malignant = sum(YPred ~= imdsTest.Labels ...
                    & imdsTest.Labels == "malignant");
    accuracy = mean(YPred == imdsTest.Labels);
    Sensitivity = T_benign / (T_benign + F_malignant);
    Specificity = T_malignant / (T_malignant + F_benign);
    Balanced_accuracy = ((T_benign / (T_benign + F_benign)) ...
                + (T_malignant / (T_malignant + F_malignant))) / 2;
    Precision = T_benign / (T_benign + F_benign);
    F_measure = 2 * Precision * Sensitivity / (Precision + Sensitivity);

    % Display four sample validation images with predicted labels
    % and the predicted probabilities of the images having those labels.
```

```
    figure
    for i = 1:4
        subplot(2,2,i);
        I = readimage(imdsTest,idx(i));
        imshow(I);
        label = YPred(idx(i));
        label_true = imdsTest.Labels(idx(i));
        title("Actual Label:" + string(label_true) + ", " ...
            + string(label) + ", " ...
            + num2str(100*max(scores(idx(i),:)),3) + "%");
    end
```

4.2 inceptionv3的转移学习

```
clc
clear

%% Load Data
    imds = imageDatastore('F:\\MATLAB\\Data', ...
        'IncludeSubfolders',true, ...
        'FileExtensions','.jpg', ...
        'LabelSource','foldernames');
    imds = shuffle(imds);
    numLabelimds = countEachLabel(imds);

    % Divide the data into training, validation and testing data sets.
    % Use 60% for training, 20% for validation and 20% for testing.
    [imdsTrain, imdsValidation, imdsTest] = splitEachLabel(imds,...
                                            0.6, 0.2,'randomized');

    % Display some sample images.
    numTrainImages = numel(imdsTrain.Labels);
    idx = randperm(numTrainImages,9);
```

```
    figure
    for i = 1:9
        subplot(3,3,i)
        I = readimage(imdsTrain,idx(i));
        imshow(I)
        label = imdsTrain.Labels(idx(i));
        title(string(label));
    end

%% Load Pretrained Network
    net = inceptionv3;

    % Extract the layer graph from the trained network
    lgraph = layerGraph(net);
    analyzeNetwork(lgraph)
    % Plot the layer graph.
    figure('Units','normalized','Position',[0.1 0.1 0.8 0.8]);
    plot(lgraph)

    % The first element of the Layers is the image input layer.
    inputSize = net.Layers(1).InputSize;

%% Replace Final Layers
    % To retrain, replace the last three layers of the network, which
    % contain information on how to combine the features
    % that the network extracts into class probabilities and labels.
    lgraph = removeLayers(lgraph, {'predictions',...
                                   'predictions_softmax',...
                                   'ClassificationLayer_predictions'});
    % Add three new layers to the layer graph
    numClasses = numel(categories(imdsTrain.Labels));
    newLayers = [
        fullyConnectedLayer(numClasses,'Name','fc',...
```

```
                                'WeightLearnRateFactor',10,...
                                'BiasLearnRateFactor',10)
        softmaxLayer('Name','softmax')
        classificationLayer('Name','classoutput')];
    lgraph = addLayers(lgraph,newLayers);

    % Connect the last transferred layer remaining in the network
    % to the new layers
    lgraph = connectLayers(lgraph,'avg_pool','fc');
    figure('Units','normalized','Position',[0.3 0.3 0.4 0.4]);
    plot(lgraph)
    ylim([0,10])

%% Freeze Initial Layers
    % Extract the layers and connections of the layer graph
    % and select which layers to freeze.
    % Use the freezeWeights to set the learning rates to zero
    % for the first several layers.
    % Use the createLgraphUsingConnections to reconnect all the layers
    % in the original order.
    layers = lgraph.Layers;
    connections = lgraph.Connections;

    layers(1:280) = freezeWeights(layers(1:280));
    lgraph = createLgraphUsingConnections(layers,connections);

%% Data augmentation
    % Use an augmented image datastore to automatically resize the images
    % Specify additional augmentation operations to perform on the images
    % Data augmentation helps prevent the network from overfitting
    % and memorizing the exact details of the training images.
    pixelRange = [-30 30];
    imageAugmenter = imageDataAugmenter( ...
```

```
        'RandXReflection',true, ...
        'RandYReflection',true, ...
        'RandXTranslation',pixelRange, ...
        'RandYTranslation',pixelRange);
    augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain, ...
        'DataAugmentation',imageAugmenter);
    augimdsValidation = augmentedImageDatastore(inputSize(1:2),imdsValida
    augimdaTest = augmentedImageDatastore(inputSize(1:2),imdsTest);

%% Train Network
    % Specify the training options.
    % Set InitialLearnRate to a small value to slow down learning
    % in the transferred layers that are not already frozen.
    % In the previous step, increasing the learning rate factors
    % for the fully connected layer to speed up learning
    % in the new final layers.
    % This combination of learning rate settings results in
    % fast learning in the new layers,
    % slower learning in the middle layers,
    % and no learning in the earlier, frozen layers.
    options = trainingOptions('sgdm', ...
        'MiniBatchSize',10, ...
        'MaxEpochs',6, ...
        'Shuffle','every-epoch', ...
        'InitialLearnRate',1e-4, ...
        'ValidationData',augimdsValidation, ...
        'ValidationFrequency',3, ...
        'ValidationPatience',Inf, ...
        'Verbose',1 , ...
        'VerboseFrequency', 100, ...
        'Plots','training-progress');
    net = trainNetwork(augimdsTrain,lgraph,options);

```

```
%% Classify Testing Images
    % Classify the testing images using the fine-tuned network,
    % and calculate the classification accuracy.
    [YPred,probs] = classify(net,augimdaTest);
    T_benign = sum(YPred == imdsTest.Labels ...
                    & imdsTest.Labels == "benign");
    T_malignant = sum(YPred == imdsTest.Labels ...
                    & imdsTest.Labels == "malignant");
    F_benign = sum(YPred ~= imdsTest.Labels ...
                    & imdsTest.Labels == "benign");
    F_malignant = sum(YPred ~= imdsTest.Labels ...
                    & imdsTest.Labels == "malignant");
    Accuracy = mean(YPred == imdsTest.Labels);
    Sensitivity = T_benign / (T_benign + F_malignant);
    Specificity = T_malignant / (T_malignant + F_benign);
    Balanced_accuracy = ((T_benign / (T_benign + F_benign)) ...
                + (T_malignant / (T_malignant + F_malignant))) / 2;
    Precision = T_benign / (T_benign + F_benign);
    F_measure = 2 * Precision * Sensitivity / (Precision + Sensitivity);

    % Display four sample validation images with predicted labels
    % and the predicted probabilities of the images having those labels.
    idx = randperm(numel(imdsTest.Files),4);
    figure
    for i = 1:4
        subplot(2,2,i)
        I = readimage(imdsTest,idx(i));
        imshow(I)
        label = YPred(idx(i));
        label_true = imdsTest.Labels(idx(i));
        title("Actual Label:" + string(label_true) + ", " ...
            + string(label) + ", " ...
            + num2str(100*max(probs(idx(i),:)),3) + "%");
```

```
end
```

4.3 CNN-DIY

- *Conv_Layer_2d*：二维卷积层，输入一个二维矩阵，二维卷积核与一个偏置，使用ReLU函数整流，输出卷积运算所得的新矩阵；

```
%img is the image inputed
%w is the kernel of the convolutional calculation, b is the bias
function y=Conv_Layer_2d(img,w,b)
img_size=size(img);
weigh_size=size(w);
new_img=zeros(img_size(1)-(weigh_size(1)-1),...
    img_size(2)-(weigh_size(2)-1));
for i=1:img_size(1)-(weigh_size(1)-1)
    for j=1:img_size(2)-(weigh_size(2)-1)
        new_img(i,j)=sum(sum(w.*...
            img(i:i+weigh_size(1)-1,j:j+weigh_size(2)-1)))+b;
        %apply convolutional calculation
    end
end
y=max(0,new_img);
%apply ReLU function
end
```

- *Conv_Layer_3d*：二维卷积层，输入一个三维矩阵，三维卷积核与一个偏置，使用ReLU函数整流，输出卷积运算所得的新矩阵；

```
%img is the image inputed
%w is the kernel of the convolutional calculation, b is the bias
function y=Conv_Layer_3d(img,w,b)
img_size=size(img);
weigh_size=size(w);
new_img=zeros(img_size(1)-(weigh_size(1)-1),...
    img_size(2)-(weigh_size(2)-1),...
```

```
    img_size(3)-(weigh_size(3)-1));
for i=1:img_size(1)-(weigh_size(1)-1)
    for j=1:img_size(2)-(weigh_size(2)-1)
        for k=1:img_size(3)-(weigh_size(3)-1)
            new_img(i,j,k)=sum(sum(sum(w.*img(i:i+weigh_size(1)-1,...
                j:j+weigh_size(2)-1,k:k+weigh_size(3)-1))))+b;
            %apply convolutional calculation
        end
    end
end
y=max(0,new_img);
%apply ReLU function
end
```

- *max_pooling*：二维池化层，输入一个二维矩阵，输出经最大池化作用后的新矩阵；

```
%apply max_pooling to dispose the image
%in this function, the max_pooling function is applied on an area of
%and the step is 2
%img is the image inputed
function y=max_pooling(img)
img_size=size(img);
new_img=zeros(img_size(1)/2,img_size(2)/2);
for i=1:img_size(1)/2
    for j=1:img_size(2)/2
        new_img(i,j)=max(max(img(2*i-1:2*i,2*j-1:2*j)));
    end
end
y=new_img;
end
```

- *softmax_layer*：softmax分类器，输入一个二维矩阵，相同大小的二维卷积核与一个偏置，在进行卷积计算后，使用sigmoid函数将运算结

果投影到[0, 1]区间；

```
%img is the feature image inputed
%wis the weigh of the convolutional kernel, b is the bias
function y=softmax_layer(img,w,b)
img_size=size(img);
weigh_size=size(w);
new_img=zeros(img_size(1)-(weigh_size(1)-1),...
    img_size(2)-(weigh_size(2)-1));
for i=1:img_size(1)-(weigh_size(1)-1)
    for j=1:img_size(2)-(weigh_size(2)-1)
        new_img(i,j)=sum(sum(w.*img(i:i+weigh_size(1)-1,...
            j:j+weigh_size(2)-1)))+b;
    end
end
y=(1+exp(-new_img)).^(-1);
%apply sigmoid function to map the result to a certain value between
%1
end
```

- *err_backward_2d*：对二维卷积层进行误差反向传播；

```
%In this function, we apply ReLU function for rectification
%err_matric is the matrix which accomodates the errors
%fd_matric is the matrix in CNN net before the err_matrix
%fd_weigh is the weigh matrix before the err_matrix
%fd_bias is the bias before the err_matrix
%matric is the matrix before the err_matrix which accomodates the
%transmission error
%weigh is the err_weigh before the err_matrix which accomodates the
%transmission error
%bias is the err_bias before the err_matrix which accomodates the
%transmission error
function [matric,weigh,b]=...
    err_backward_2d(err_matric,fd_matric,fd_weigh,fd_bias)
```

```
err_matric_size=size(err_matric);
%get the size of err_matric
w_size=size(fd_weigh);
%get the size of fd_weigh
weigh=zeros(w_size(1),w_size(2));
matric=zeros(err_matric_size(1)+w_size(1)-1,...
    err_matric_size(2)+w_size(2)-1);
fd_trans=Conv_Layer_2d(fd_matric,fd_weigh,fd_bias);
%apply Conv_Layer_2d function to get the forward result
b=0;
for i=1:err_matric_size(1)
    for j=1:err_matric_size(2)
        if fd_trans(i,j)~=0
            %consider whether forward result is zero or not, if the r
            %is zero, then the error can't be accumlulated backward
            for k1=1:w_size(1)
                for k2=1:w_size(2)
                    matric(i+k1-1,j+k2-1)=matric(i+k1-1,j+k2-1)+...
                        fd_weigh(k1,k2)*err_matric(i,j);
                    weigh(k1,k2)=weigh(k1,k2)+...
                        fd_matric(i+k1-1,j+k2-1)*err_matric(i,j);
                    b=b+err_matric(i,j);
                    %apply complex derivative to support the backward
                    %accumulation
                end
            end
        end
    end
end
end
```

- *err_backward_3d*：对三维卷积层进行误差反向传播；

```
%In this function, we apply ReLU function for rectification
```

```
%err_matric is the matrix which accomodates the errors
%fd_matric is the matrix in CNN net before the err_matrix
%fd_weigh is the weigh matrix before the err_matrix
%fd_bias is the bias before the err_matrix
%matric is the matrix before the err_matrix which accomodates the
%transmission error
%weigh is the err_weigh before the err_matrix which accomodates the
%transmission error
%bias is the err_bias before the err_matrix which accomodates the
%transmission error
function [matric,weigh,b]=...
    err_backward_3d(err_matric,fd_matric,fd_weigh,fd_bias)
err_matric_size=size(err_matric);
%get the size of err_matric
if length(err_matric_size)==2
    err_matric_size(3)=1;
end
w_size=size(fd_weigh);
%get the size of fd_weigh
weigh=zeros(w_size(1),w_size(2),w_size(3));
matric=zeros(err_matric_size(1)+w_size(1)-1,...
    err_matric_size(2)+w_size(2)-1,...
    err_matric_size(3)+w_size(3)-1);
fd_trans=Conv_Layer_3d(fd_matric,fd_weigh,fd_bias);
%apply Conv_Layer_3d function to get the forward result
b=0;
for i=1:err_matric_size(1)
    for j=1:err_matric_size(2)
        for k=1:err_matric_size(3)
            if fd_trans(i,j,k)~=0
                %consider whether forward result is zero or not, if t
                %result is zero, then the error can't be accumlulated
                %backward
```

```
                for k1=1:w_size(1)
                    for k2=1:w_size(2)
                        for k3=1:w_size(3)
                            matric(i+k1-1,j+k2-1,k+k3-1)=...
                                matric(i+k1-1,j+k2-1,k+k3-1)+...
                                fd_weigh(k1,k2,k3)*err_matric(i,j,k);
                            weigh(k1,k2,k3)=weigh(k1,k2,k3)+...
                                fd_matric(i+k1-1,j+k2-1,k+k3-1)*...
                                err_matric(i,j,k);
                            b=b+err_matric(i,j,k);
                            %apply complex derivative to support the
                            %backward error accumulation
                        end
                    end
                end
            end
        end
    end
end
end
```

- *err_backward_2d_softmax*：对softmax分类器进行误差反向传播；

```
%In this function, we apply sigmoid function for rectification
%err_matric is the matrix which accomodates the errors
%fd_matric is the matrix in CNN net before the err_matrix
%fd_weigh is the weigh matrix before the err_matrix
%fd_bias is the bias before the err_matrix
%matric is the matrix before the err_matrix which accomodates the
%transmission error
%weigh is the err_weigh before the err_matrix which accomodates the
%transmission error
%bias is the err_bias before the err_matrix which accomodates the
%transmission error
```

```
function [matric,weigh,b]=...
    err_backward_2d_softmax(err_matric,fd_matric,fd_weigh,fd_bias)
err_matric_size=size(err_matric);
%get the size of err_matric
w_size=size(fd_weigh);
%get the size of fd_weigh
weigh=zeros(w_size(1),w_size(2));
matric=zeros(err_matric_size(1)+...
    w_size(1)-1,err_matric_size(2)+w_size(2)-1);
fd_trans=softmax_layer(fd_matric,fd_weigh,fd_bias);
b=0;
for i=1:err_matric_size(1)
    for j=1:err_matric_size(2)
        for k1=1:w_size(1)
            for k2=1:w_size(2)
                matric(i+k1-1,j+k2-1)=matric(i+k1-1,j+k2-1)+...
                    fd_weigh(k1,k2)*err_matric(i,j)*...
                    fd_trans(i,j)*(1-fd_trans(i,j));
                weigh(k1,k2)=weigh(k1,k2)+fd_matric(i+k1-1,j+k2-1)*..
                    err_matric(i,j)*fd_trans(i,j)*(1-fd_trans(i,j));
                b=b+err_matric(i,j)*fd_trans(i,j)*(1-fd_trans(i,j));
                %apply complex derivative to support the backward err
                %accumulation
            end
        end
    end
end
end
```

- *err_backward_3d_easy*：对三维卷积层进行误差反向传播，但只计算权重，偏置上的误差传播，以减少实际计算量；

```
%In this function, we apply ReLU function for rectification
%err_matric is the matrix which accomodates the errors
```

```
%fd_matric is the matrix in CNN net before the err_matrix
%fd_weigh is the weigh matrix before the err_matrix
%fd_bias is the bias before the err_matrix
%weigh is the err_weigh before the err_matrix which accomodates the
%transmission error
%bias is the err_bias before the err_matrix which accomodates the
%transmission error
function [weigh,b]=...
    err_backward_3d_easy(err_matric,fd_matric,fd_weigh,fd_bias)
err_matric_size=size(err_matric);
%get the size of err_matric
if length(err_matric_size)==2
    err_matric_size(3)=1;
end
w_size=size(fd_weigh);
%get the size of fd_weigh
weigh=zeros(w_size(1),w_size(2),w_size(3));
fd_trans=Conv_Layer_3d(fd_matric,fd_weigh,fd_bias);
%apply Conv_Layer_3d function to get the forward result
b=0;
for i=1:err_matric_size(1)
    for j=1:err_matric_size(2)
        for k=1:err_matric_size(3)
            if fd_trans(i,j,k)~=0
                %consider whether forward result is zero or not, if t
                %result is zero, then the error can‘t be accumlulated
                %backward
                for k1=1:w_size(1)
                    for k2=1:w_size(2)
                        for k3=1:w_size(3)
                            weigh(k1,k2,k3)=weigh(k1,k2,k3)+...
                                fd_matric(i+k1-1,j+k2-1,k+k3-1)*...
                                err_matric(i,j,k);
```

```
                                b=b+err_matric(i,j,k);
                                %apply complex derivative to support the
                                %backward error accumulation
                            end
                        end
                    end
                end
            end
        end
end
end
```

- *SGD*：利用随机梯度下降法对神经网络进行训练，训练使用的卷积神经网络如上一节所说明；

```
%in the function, wi and bi refers to the relative weighs and bias
%img is the image matrix
%label is the image label
function [w1,w2,w3,w4,w5,w6,w7,b1,b2,b3,b4,b5,b6,b7]=...
    SGD(img,label,a,w1,w2,w3,w4,w5,w6,w7,b1,b2,b3,b4,b5,b6,b7)
img_1=Conv_Layer_3d(img,w1,b1);
img_2=max_pooling(img_1);
img_3=Conv_Layer_2d(img_2,w2,b2);
img_4=max_pooling(img_3);
img_5=Conv_Layer_2d(img_4,w3,b3);
img_6=max_pooling(img_5);
img_7=Conv_Layer_2d(img_6,w4,b4);
img_8=max_pooling(img_7);
img_9=Conv_Layer_2d(img_8,w5,b5);
img_10=max_pooling(img_9);
img_11=Conv_Layer_2d(img_10,w6,b6);
result=softmax_layer(img_11,w7,b7);
%construt the CNN net
err=label-result;
```

```
[img_11_err,w7_err,b7_err]=err_backward_2d_softmax(err*a,img_11,w7,b7
[img_10_err,w6_err,b6_err]=err_backward_2d(img_11_err,img_10,w6,b6);
img_9_err=err_backward_maxpooling(img_10_err,img_9);
[img_8_err,w5_err,b5_err]=err_backward_2d(img_9_err,img_8,w5,b5);
img_7_err=err_backward_maxpooling(img_8_err,img_7);
[img_6_err,w4_err,b4_err]=err_backward_2d(img_7_err,img_6,w4,b4);
img_5_err=err_backward_maxpooling(img_6_err,img_5);
[img_4_err,w3_err,b3_err]=err_backward_2d(img_5_err,img_4,w3,b3);
img_3_err=err_backward_maxpooling(img_4_err,img_3);
[img_2_err,w2_err,b2_err]=err_backward_2d(img_3_err,img_2,w2,b2);
img_1_err=err_backward_maxpooling(img_2_err,img_1);
[w1_err,b1_err]=err_backward_3d_easy(img_1_err,img,w1,b1);
%get the backward error by err_backward function
w1=w1+w1_err;
w2=w2+w2_err;
w3=w3+w3_err;
w4=w4+w4_err;
w5=w5+w5_err;
w6=w6+w6_err;
w7=w7+w7_err;
b1=b1+b1_err;
b2=b2+b2_err;
b3=b3+b3_err;
b4=b4+b4_err;
b5=b5+b5_err;
b6=b6+b6_err;
b7=b7+b7_err;
%modify the weighs and biases according to the backward errors
end
```

- CNN_net：利用卷积神经网络对图像进行识别；

```
function result=CNN_net(img,w1,w2,w3,w4,w5,w6,w7,b1,b2,b3,b4,b5,b6,b7
img_1=Conv_Layer_3d(img,w1,b1);
```

```
img_2=max_pooling(img_1);
img_3=Conv_Layer_2d(img_2,w2,b2);
img_4=max_pooling(img_3);
img_5=Conv_Layer_2d(img_4,w3,b3);
img_6=max_pooling(img_5);
img_7=Conv_Layer_2d(img_6,w4,b4);
img_8=max_pooling(img_7);
img_9=Conv_Layer_2d(img_8,w5,b5);
img_10=max_pooling(img_9);
img_11=Conv_Layer_2d(img_10,w6,b6);
result=softmax_layer(img_11,w7,b7);
end
```

- *percision*：脚本文件，训练神经网络，并计算神经网络的正确率。

```
clc
clear

%deside the learning rate
tic

imds = imageDatastore('F:\\MATLAB\\Data', ...
    'IncludeSubfolders',true, ...
    'FileExtensions','.jpg', ...
    'LabelSource','foldernames');
imds = shuffle(imds);%randomize the images
numLabelimds = countEachLabel(imds);%get the labels of images
[imdsTrain, imdsTest] = splitEachLabel(imds,...
                                        0.7,'randomized');
Trainnum=length(imdsTrain.Labels);
Testnum=length(imdsTest.Labels);

inputSize = [299 299 3];
augimdsTrain = augmentedImageDatastore(inputSize,imdsTrain);
```

```
augimdsTest = augmentedImageDatastore(inputSize,imdsTest);
%adapt the size of training set and testing set to the suitable one

%decide the initial weighs and biases
a_array = 10:-1/Trainnum * 9.9:0.1;
w1 = (randn(8,8,3));
w2 = (randn(7,7));
w3 = (randn(7,7));
w4 = (randn(5,5));
w5 = (randn(5,5));
w6 = (randn(3,3));
w7 = (randn(3,3));
b1=0;
b2=0;
b3=0;
b4=0;
b5=0;
b6=0;
b7=0;

%apply SGD method to train the CNN net
for i=1:Trainnum
    a = a_array(i);
    [image,img_label] = readByIndex(augimdsTrain,i);
    if imdsTrain.Labels(i)=='benign'
        new_label=1;
    else
        new_label=0;
    end
    [w1,w2,w3,w4,w5,w6,w7,b1,b2,b3,b4,b5,b6,b7]=...
        SGD(im2double(cell2mat(image.input)),...
        new_label,a,w1,w2,w3,w4,w5,w6,w7,b1,b2,b3,b4,b5,b6,b7);
    i
```

```
    toc
end

Pre_Label=zeros(Testnum);
T_benign=0;
T_malignant=0;
F_benign=0;
F_malignant=0;

%apply the testing set to validify the trained CNN net
for j=1:Testnum
    [image,img_label]=readByIndex(augimdsTest,j);
    if CNN_net(im2double(cell2mat(image.input)),...
            w1,w2,w3,w4,w5,w6,w7,b1,b2,b3,b4,b5,b6,b7)>0.5
        Pre_Label(j)=1;
        if imdsTest.Labels(j)=='benign'
            T_benign=T_benign+1;
        else
            F_benign=F_benign+1;
        end
    else
        Pre_Label(j)=0;
        if imdsTest.Labels(j)=='benign'
            F_benign=F_benign+1;
        else
            T_benign=T_benign+1;
        end
    end
    j
    toc
end

%abtain the parameter to consider the performance of the CNN net in t
```

```
%practice
Accuracy = mean(YPred == imdsTest.Labels);
Sensitivity = T_benign / (T_benign + F_malignant);
Specificity = T_malignant / (T_malignant + F_benign);
Balanced_accuracy = ((T_benign / (T_benign + F_benign)) ...
            + (T_malignant / (T_malignant + F_malignant))) / 2;
Precision = T_benign / (T_benign + F_benign);
F_measure = 2 * Precision * Sensitivity / (Precision + Sensitivity);
```

参考文献

[1] Noel Codella, Junjie Cai, Mani Abedini, Rahil Garnavi, Alan Halpern, and John R Smith. Deep learning, sparse coding, and svm for melanoma recognition in dermoscopy images. In *International Workshop on Machine Learning in Medical Imaging*, pages 118–126. Springer, 2015.

[2] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115, 2017.

[3] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[4] Yuexiang Li and Linlin Shen. Skin lesion analysis towards melanoma detection using deep learning network. *Sensors*, 18(2):556, 2018.

[5] Ilker Ali OZKAN and Murat KOKLU. Skin lesion classification using machine learning algorithms. *International Journal of Intelligent Systems and Applications in Engineering*, 5(4):285–289, 2017.

[6] Patric Ridell and Henning Spett. Training set size for skin cancer classification using google' s inception v3, 2017.

[7] Lequan Yu, Hao Chen, Qi Dou, Jing Qin, and Pheng-Ann Heng. Automated melanoma recognition in dermoscopy images via very deep resid-

ual networks. *IEEE transactions on medical imaging*, 36(4):994–1004, 2017.