**《深度学习》**

结课论文

|  |  |
| --- | --- |
| **时间：** | **2022.06.08** |
| **学生姓名：** | **理信19.2陶李涛** |
| **班级学号：** | **201910030209** |

基于卷积神经网络的脑癌诊断

摘要

脑癌肿瘤计算机辅助诊断(CAD)系统在医学检测和诊断中日益重要. 为了区分核磁共振图像中的脑区肿瘤与非肿瘤. 利用深度学习和迁移学习的方法. 设计一种脑癌CAD系统.

在BTMRI数据集上[1] , 首先对数据集进行不平衡处理和数据增强[7] ; 利用卷积神经网络(CNN)提取特征. 选取分类性能最高的一层作为微调节点. 其后维度较低层为连接新网络节点；采用固定微调节点前的网络层不可训练. 其余层可训练的方式微调. 分别基于在ImageNet预训练的VGG16[2] 、EfficientNetB0[3] 、ResNet50[4] 构建的CAD系统. 性能准确率高于98%. 其中基于VGG16和ResNet50搭建的系统性能突出. 可用于其他医学图像病例诊断的任务作为预训练模型[10] .

**关键词：深度卷积神经网络; ResNet50; 迁移学习; 计算机辅助诊断系统**

# 1 引言

脑肿瘤是大脑中异常细胞的集合或肿块. 你的头骨, 它包围着你的大脑, 非常僵硬. 在这种受限空间内的任何生长都可能导致问题. 脑肿瘤可以是癌性（恶性）或非癌性（良性）. 当良性或恶性肿瘤生长时, 它们会导致颅骨内的压力增加. 这可能会导致脑损伤, 并可能危及生命.

图1 脑肿瘤癌症位置示例图

脑肿瘤的早期发现和分类是医学影像领域的一个重要研究领域, 因此有助于选择最方便的治疗方法来挽救患者的生命.

在改善健康诊断方面应用深度学习方法正在提供有影响力的解决方案. 根据世界卫生组织 (WHO) 的规定, 正确的脑肿瘤诊断包括检测、脑肿瘤位置识别以及根据恶性程度、等级和类型对肿瘤进行分类. 这项使用磁共振成像 (MRI) 诊断脑肿瘤的实验工作包括检测肿瘤、根据等级、类型和肿瘤位置识别对肿瘤进行分类. 这种方法已经在利用一个模型对不同分类任务的脑 MRI 进行分类方面进行了实验, 而不是针对每个分类任务使用单独的模型[7] .

为解决传统机器学习在兵力图像诊断方面的性能不足和纯粹人工阅片导致的误诊或者错诊等问题, 基于卷积神经网络 (CNN) 的多任务分类可用于肿瘤的分类和检测[9] . 脑肿瘤位置的识别也使用基于 CNN 的模型通过分割脑肿瘤来完成[11] .

# 2 技术路线

## 2.1 VGG-16 卷积神经网络

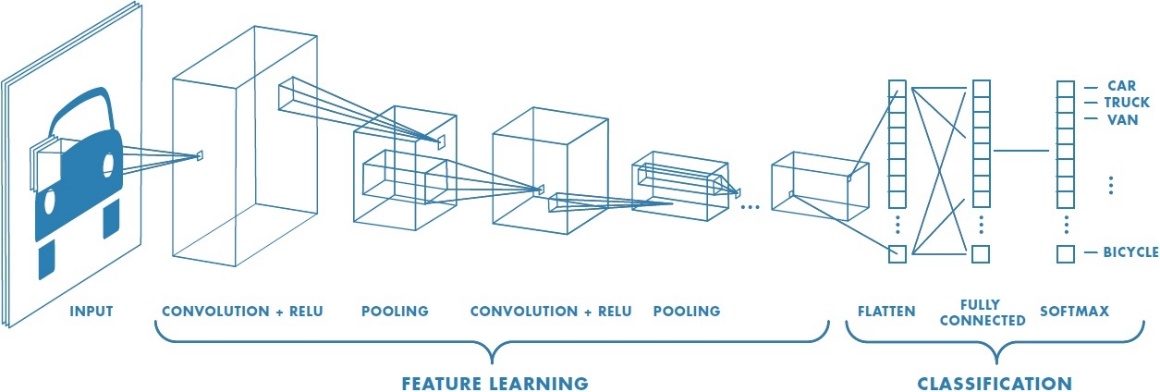
VGG是Oxford的Visual Geometry Group 的组提出的（大家应该能看出VGG名字的由来了）. 该网络是在 ILSVRC 2014上的相关工作, 主要工作是证明了增加网络的深度能够在一定程度上影响网络最终的性能. VGG有两种结构, 分别是VGG16和VGG19, 两者并没有本质上的区别, 只是网络深度不一样[1] a)[2] [2] , 模型结构如下图所示：

图2 VGG模型图

VGG16相比AlexNet的一个改进是采用连续的几个3x3的卷积核代替AlexNet中的较大卷积核（11x11, 7x7, 5x5）. 对于给定的感受野（与输出有关的输入图片的局部大小）, 采用堆积的小卷积核是优于采用大的卷积核, 因为多层非线性层可以增加网络深度来保证学习更复杂的模式, 而且代价还比较小（参数更少）.

简单来说, 在VGG中, 使用了3个3x3卷积核来代替7x7卷积核, 使用了2个3x3卷积核来代替5\*5卷积核, 这样做的主要目的是在保证具有相同感知野的条件下, 提升了网络的深度, 在一定程度上提升了神经网络的效果.

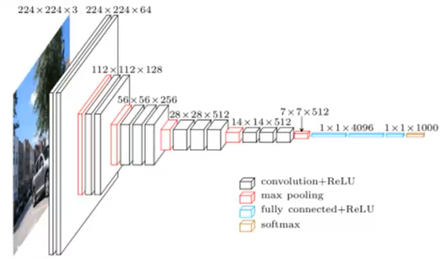
比如, 3个步长为1的3x3卷积核的一层层叠加作用可看成一个大小为7的感受野（其实就表示3个3x3连续卷积相当于一个7x7卷积）, 其参数总量为 3x(9xC^2) , 如果直接使用7x7卷积核, 其参数总量为 49xC^2 , 这里 C 指的是输入和输出的通道数. 很明显, 27xC^2小于49xC^2, 即减少了参数；而且3x3卷积核有利于更好地保持图像性质.

图3 VGG网络结构

## 2.2 EfficientNetB0 卷积神经网络

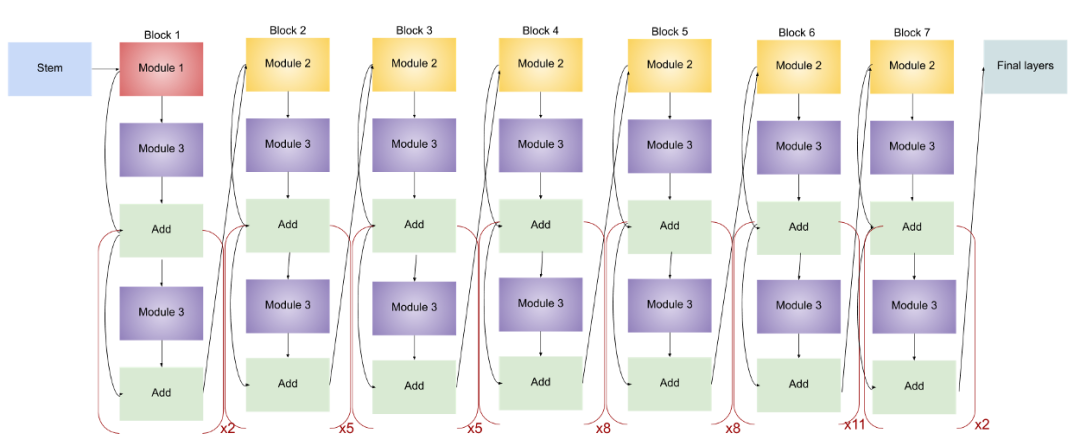
EfficientNets是谷歌大脑的工程师谭明星和首席科学家Quoc V. Le在论文《EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks》中提出[3] . 该模型的基础网络架构是通过使用神经网络架构搜索（neural architecture search）设计得到. 卷积神经网络模型通常是在已知硬件资源的条件下, 进行训练的. 当你拥有更好的硬件资源时, 可以通过放大网络模型以获得更好的训练结果. 为系统的研究模型缩放, 谷歌大脑的研究人员针对EfficientNets的基础网络模型提出了一种全新的模型缩放方法, 该方法使用简单而高效的复合系数来权衡网络深度、宽度和输入图片分辨率.

图4 EfficientNetB0 网络结构图

## 2.3 ResNet50卷积神经网络

ResNet有2个基本的block, 一个是Identity Block, 输入和输出的dimension是一样的, 所以可以串联多个；另外一个基本block是Conv Block, 输入和输出的dimension是不一样的, 所以不能连续串联, 它的作用本来就是为了改变feature vector的dimension

因为 CNN 最后都是要把 image 一点点的 convert 成很小但是 depth 很深的 feature map, 一般的套路是用统一的比较小的kernel（比如VGG都是用3\*3）, 但是随着网络深度的增加, output的 channel 也增大（学到的东西越来越复杂）, 所以有必要在进入Identity Block之前, 用 Conv Block转换一下维度, 这样后面就可以连续接Identity Block, 其实就是在shortcut path的地方加上一个conv2D layer（1\*1 filter size）, 然后在 main path 改变 dimension, 并与 shortcut path 对应起来.

如下图所示：

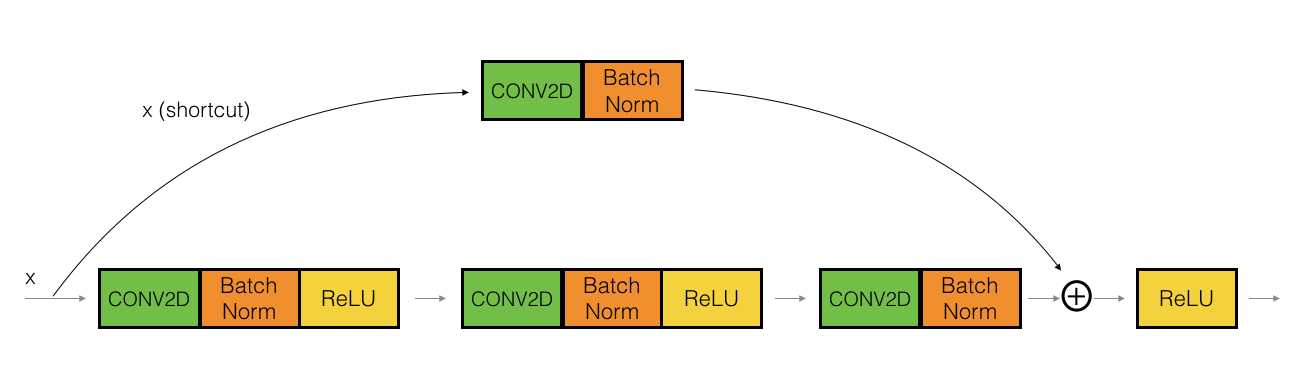


图5 ResNet50主干网络结构图

ResNet50 是在现有训练深度网络的基础上, 提出的一种具有易优化、计算负担小等优点的残差学习框架. 残差用来设计解决退化和梯度问题, 使得网络的性能在深度增加的同时也得以提升. ResNet50 中包含了 49 个卷积层和 1 个全连阶层, 其中, 第二至五阶段 中 的id blockｘ２ 代表的是两个不改变维度的残差块, conv block 代表的是添加维度的残差块, 每个残差块包含三个卷积层, 所以有  个卷积层

## 2.5 预训练与迁移学习

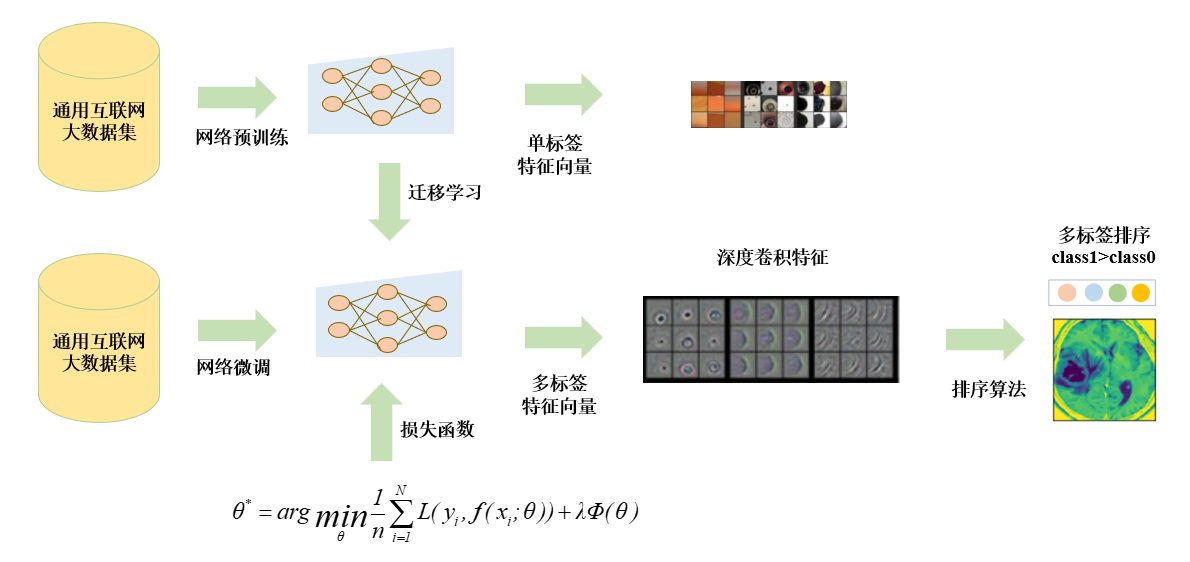
需要搭建一个网络模型来完成一个特定的图像分类的任务. 因此需要随机初始化参数, 然后开始训练网络, 不断调整直到网络的损失越来越小. 在训练的过程中, 一开始初始化的参数会不断变化. 由于模型参数过于繁多, 直接训练的loss函数变化不太明显, 训练开销时间长, 因此可以将训练模型的参数保存下来, 以便训练好的模型可以在下次执行类似任务时获得较好的结果. 这个过程就是 pre-training. 迁移学习中, 以小数据量的样本集作为目标域, 以大量标注的数据集作为源域. 主要有两大类: 目标域与源域图像差别较小时, 通过两个数据集间数学映射关系, 扩充目标域的数据量；目标域与源域图像差别较大时, 建立在卷积神经网络层、特征层之间的迁移. 第二类迁移学习方式适用于本文的小样本数据集医学图像, 利用自然图像数据集ImageNet和深度CNN进行迁移学习[10] .

图6 图像迁移学习原理

在迁移学习过程中, 主流观点认为, 相近的种类迁移效果优于两个相差较大的迁移. ImageNet自然图像与医学肿瘤图像之间有较大差距, 但医学影像领域不存在大规模数据集. 基于以上条件, 本文使用预训练的VGG16、ResNet50、EfficientNetB0等预训练模型来进行训练.

# 3 实验

## 3.1 数据集

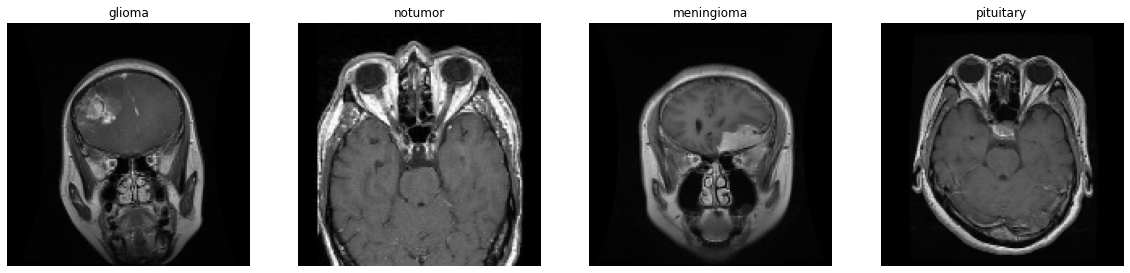
数据集来自kaggle的Brain Tumor MRI Dataset[1] , 该数据集包含7022张人脑 MRI 图像, 分为 4 类：胶质瘤、脑膜瘤、正常垂体.

胶质瘤(meningioma)：源自神经上皮的肿瘤统称为脑胶质瘤, 占颅脑肿瘤的40%～50%, 是最常见的原发性颅内肿瘤.

脑膜瘤(notumor)：是起源于脑膜及脑膜间隙的衍生物, 发病率占颅内肿瘤的19.2%, 50%位于矢状窦旁, 另大脑凸面, 大脑镰旁者多见, 其次为蝶骨嵴、鞍结节、嗅沟、小脑桥脑角与小脑幕等部位, 生长在脑室内者很少, 也可见于硬膜外. 其它部位偶见.

神经胶质瘤(glioma)：是最常见的原发性中枢神经系统肿瘤, 约占所有颅内原发肿瘤的一半, 广义是指所有神经上皮来源的肿瘤, 狭义是指源于各类胶质细胞的肿瘤.

以及正常垂体. 数据集各种类样本如下：

图7 数据集的部分样本

## 3.2 数据集处理及划分

该数据集包含7022张人脑 MRI 图像, 分为 4 类：胶质瘤、脑膜瘤、正常垂体. 对于训练一个神经网络模型, 良好的数据划分极为关键, 将数据集划分为4：1划分, 对于图片大小, 统一设置image\_size = 128

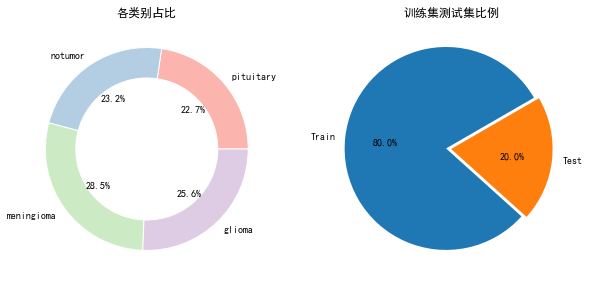


图8 各类别占比及数据集划分比例

3.3 模型搭建

使用tensorflow的keras模块搭建基于ImageNet预训练的模型, 设置最后几层分别为Flatten层、使用relu函数的Dense层、以及最后一层输入为分类类别的全连接层, 选择激活函数为softmax, 且对于最后微调的三层网络中加入Dropout, 随机失活率分别为0.3, 0.2, 对于三个网络, 均基于此搭建模型, 具体结构图如下所示：

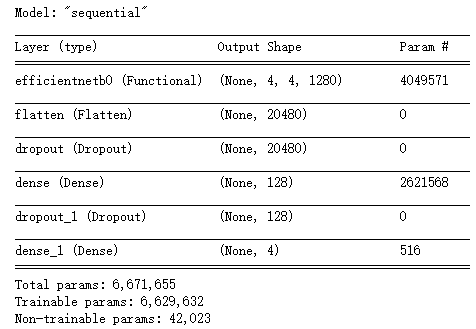
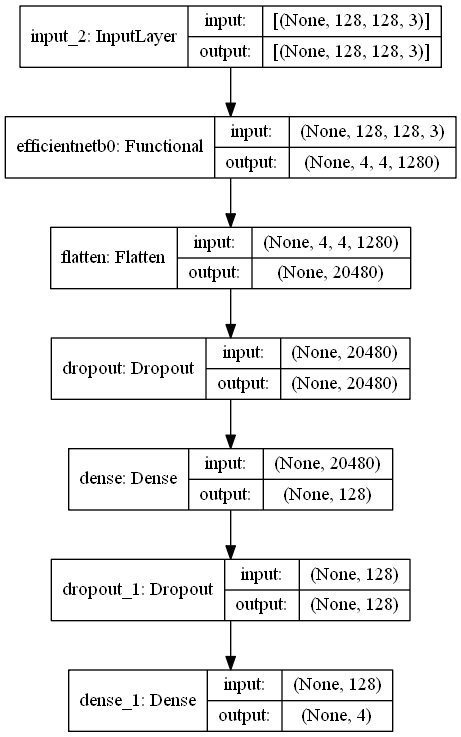
 

图9 EfficientNetB0网络结构图

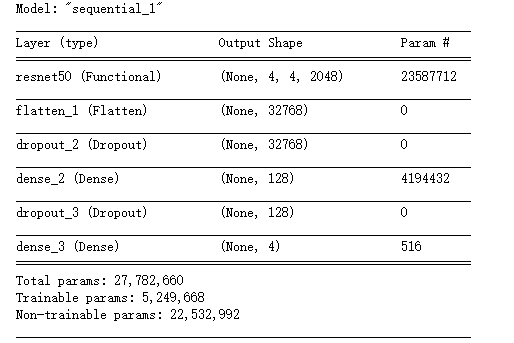
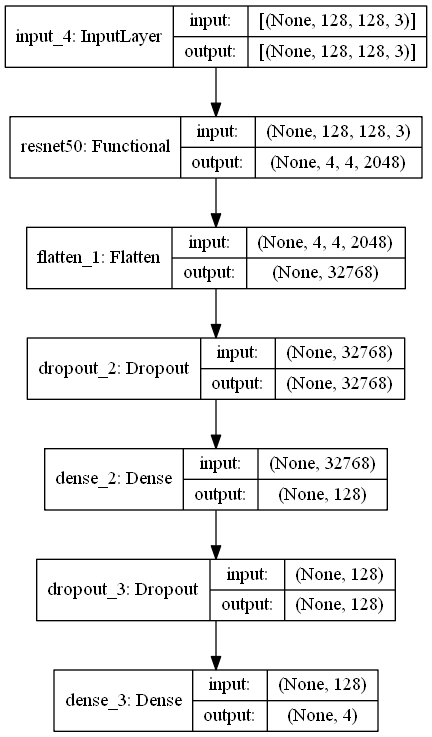
 

图10 ResNet50网络结构图

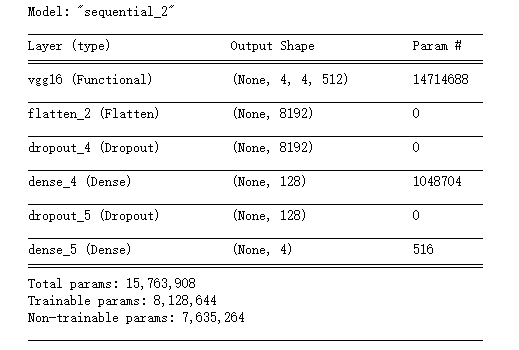
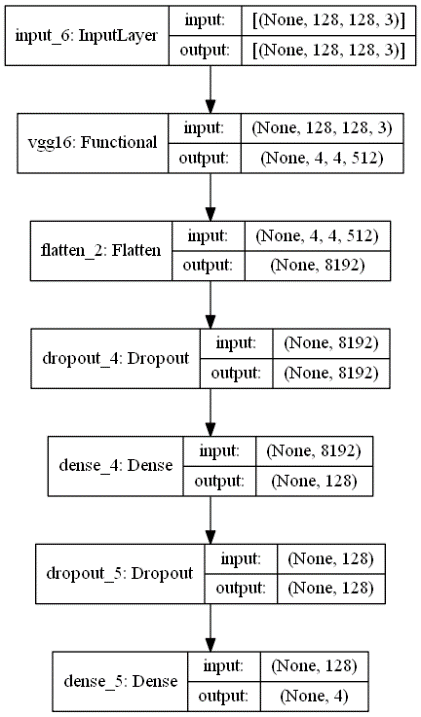
 

图11 vgg16网络结构图

基于以上模型结构, 对现有数据集进行训练.

## 3.4 训练参数

对于三个网络的训练过程中, 使用相同的超参数进行训练, 限于现有显卡环境, 设置的batch\_size比较小, 而且只训练了20轮, 但是数据集图片量较小, 且模型迁移精度效果较好, 因此就以如下参数进行的训练.

表1 训练过程部分超参数

|  |  |
| --- | --- |
| **参数名称** | **值** |
| epochs | 20 |
| Batch\_size | 16 |
| loss | categorical\_crossentropy |
| optimizer | Adam |
| metrics | accuracy |
| image\_size | 128 |

对于训练过程中, 还使用tensorboard进行存储训练过程日志, 方便查看训练过程中的信息, 以及对于训练超参数学习率使用ReduceLROnPlateau的方法, 其超参数设置如下：

表2 ReduceLROnPlateau参数

|  |  |
| --- | --- |
| **参数名称** | **值** |
| monitor | val\_accuracy |
| factor | 0.3 |
| patience | 2 |
| min\_delta | 0.001 |
| mode | auto |
| verbose | 1 |

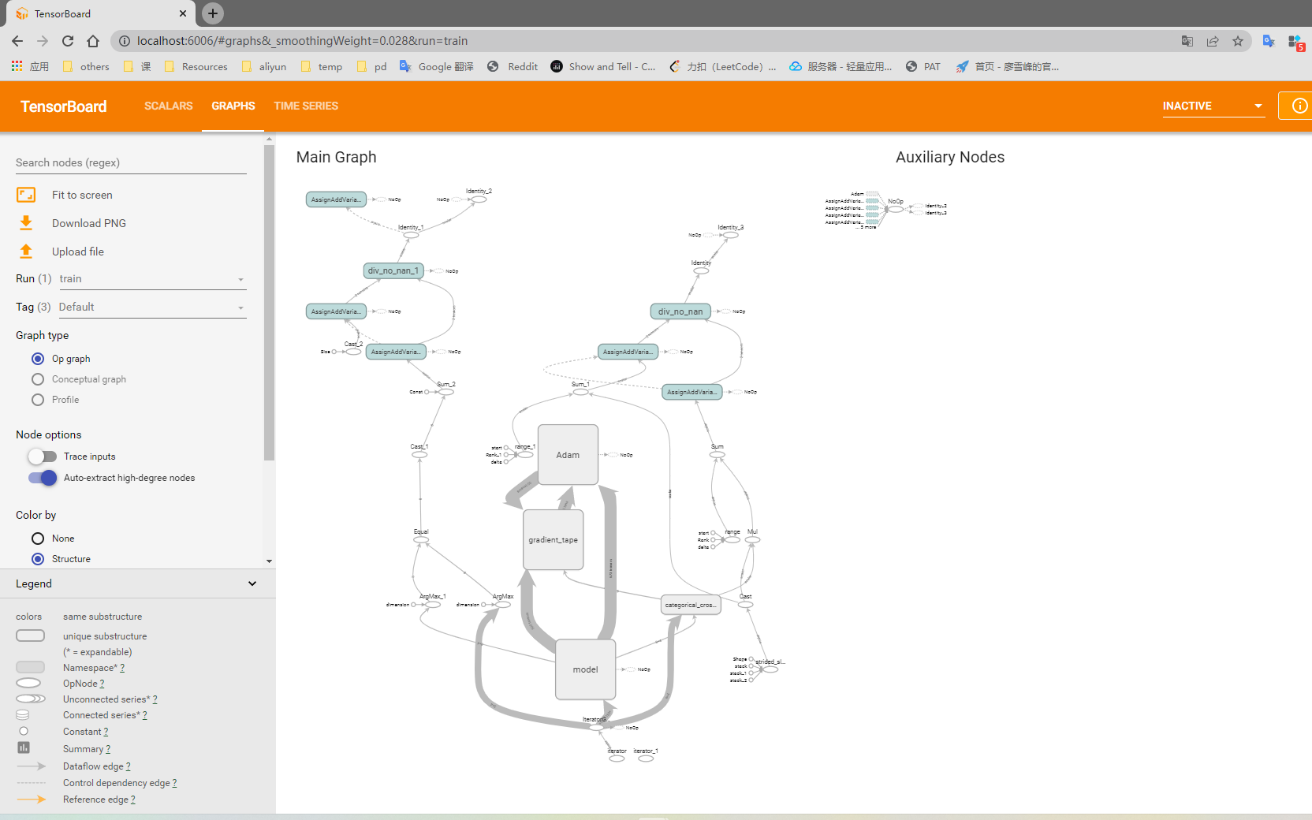
Tensorboard记录如下：

图12 tensorboard模型结构图

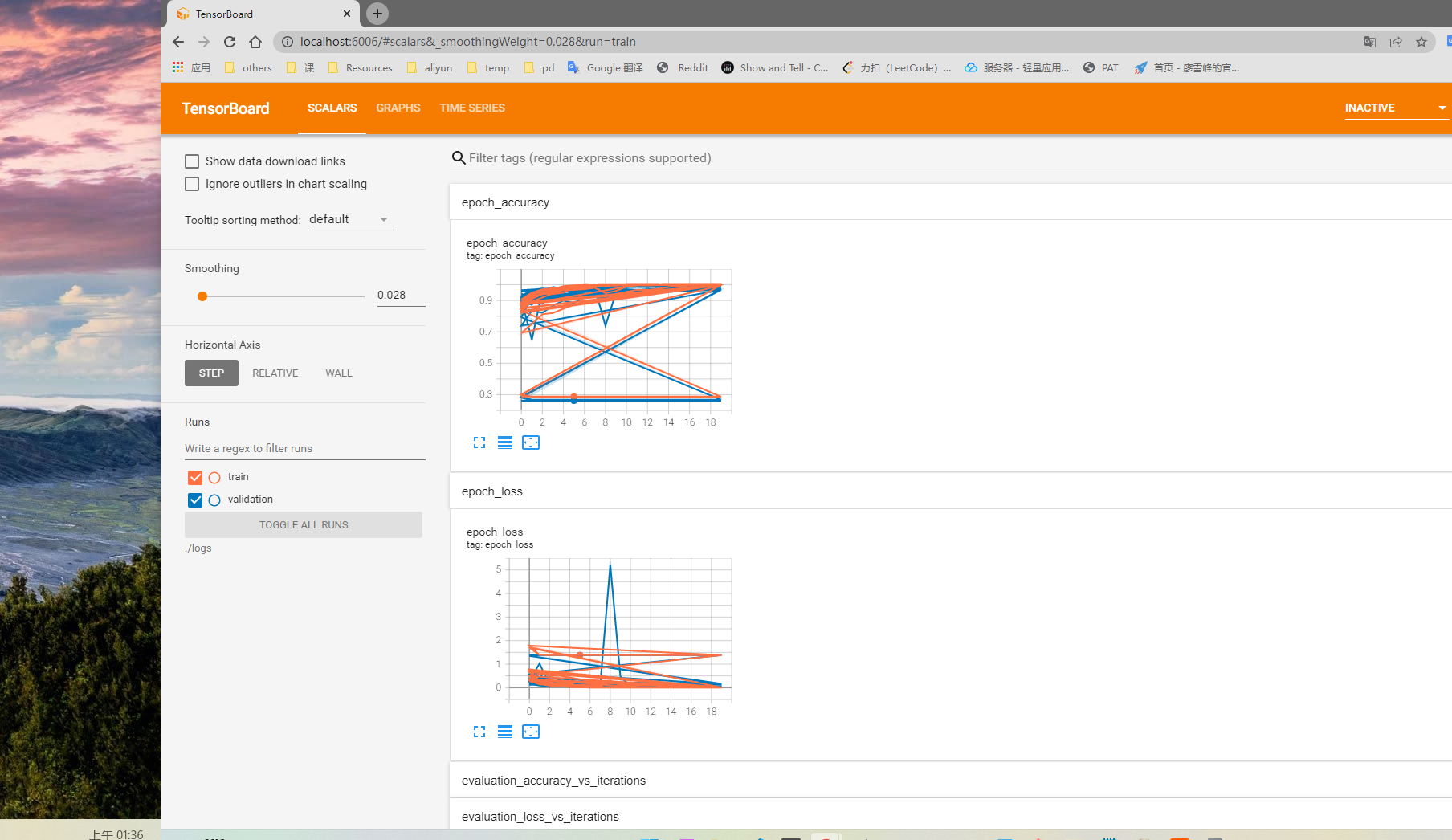


图13 tensorboard记录模型训练页面

以及记录各个epochs的各项数据, 使用如附录2所示的运行环境, （相关代码及数据集源文件, 已上传到github, 请点击我的github:https://github.com/scdctlt/course\_work）

## 3.5 模型结果

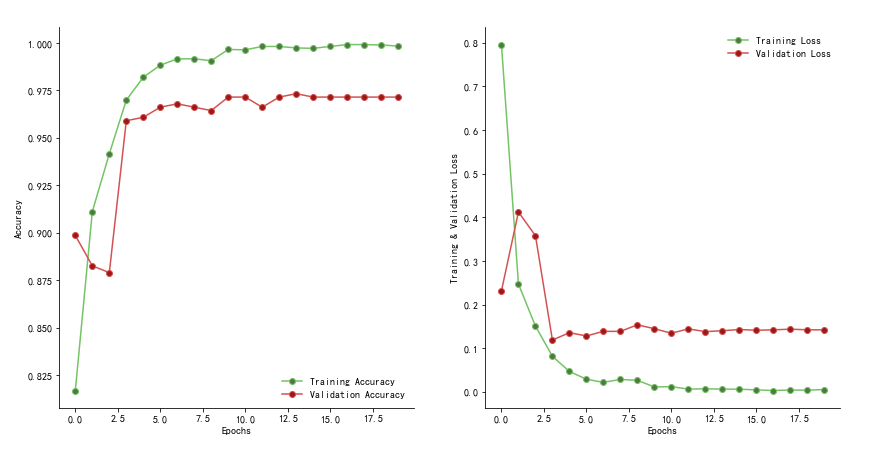
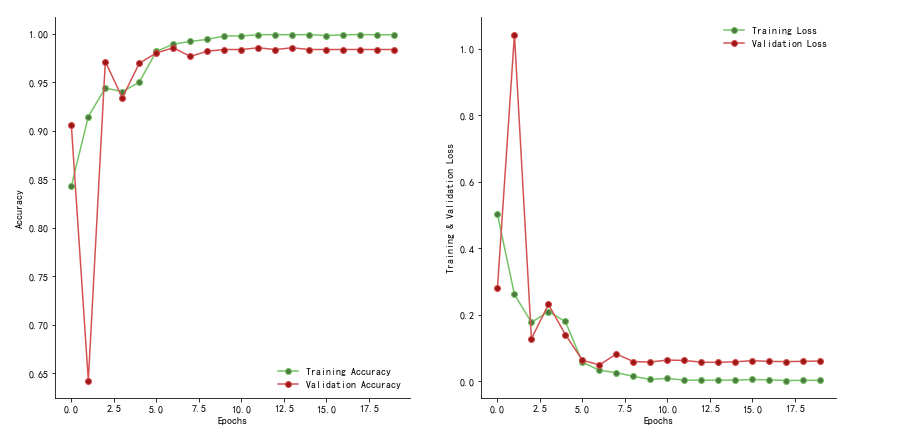
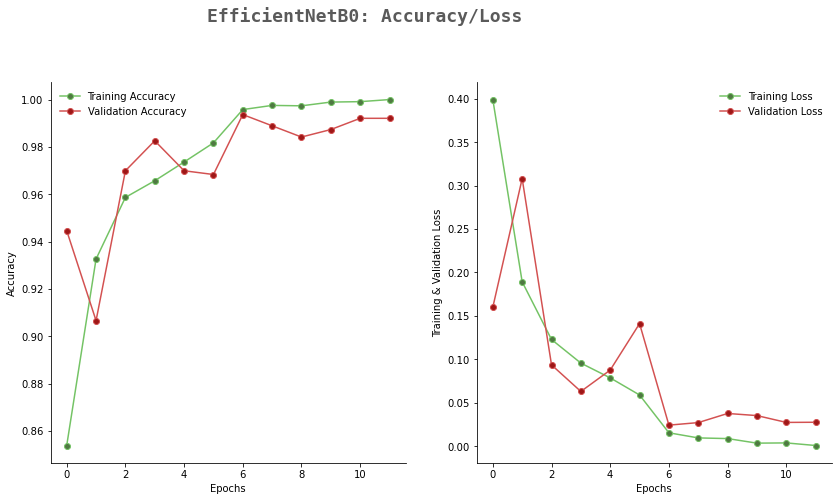
对于以上几个经典的卷积神经网络, 其识别准确率已经达到98%以上, 具有良好的可迁移性能, 其训练的结果如下：

图14 ResNet50训练过程Accuracy/Loss随epochs变化情况图

图15 EfficientNetB0训练过程Accuracy/Loss随epochs变化情况图

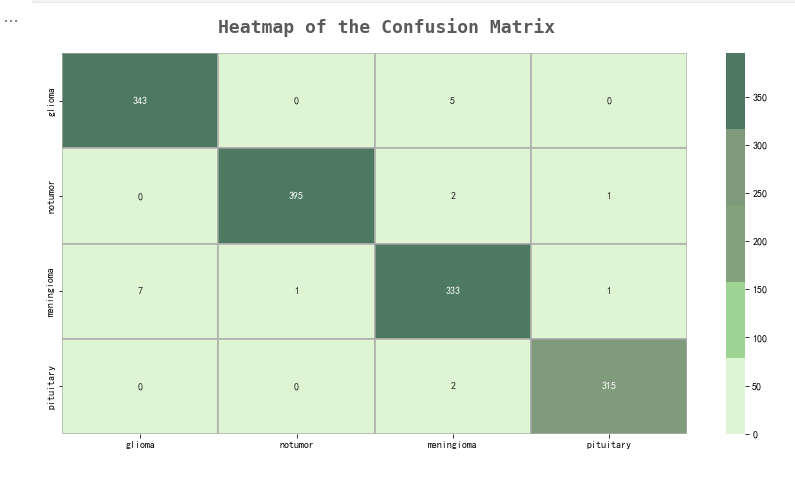
图16 VGG16 训练过程Accuracy/Loss随epochs变化情况图

对于以上训练数据的准确率以及训练过程中的变化, 查阅相关资料得知[10] , 针对小规模样本数据集, 现有的分类模型以及很好的可以学习到其相关表征, 且EfficientNetB0、VGG16和ResNet50都是比较经典的分类网络, 经过微调后可适用于各种分类检测任务, 在此对比了几个模型的情况具体如下表：

表3 各个模型训练描述表

|  |  |  |  |
| --- | --- | --- | --- |
| **模型名称** | **精度** | **参数量** | **训练时间** |
| EfficientNetB0 | 0.9857 | 6,671K |  |
| VGG16 | 0.9863 | 15,763K |  |
| ResNet50 | 0.9898 | 27,782K |  |

这里展示ResNet50的混淆矩阵, 如下图所示：

图17 ResNet50测试集混淆矩阵热力图

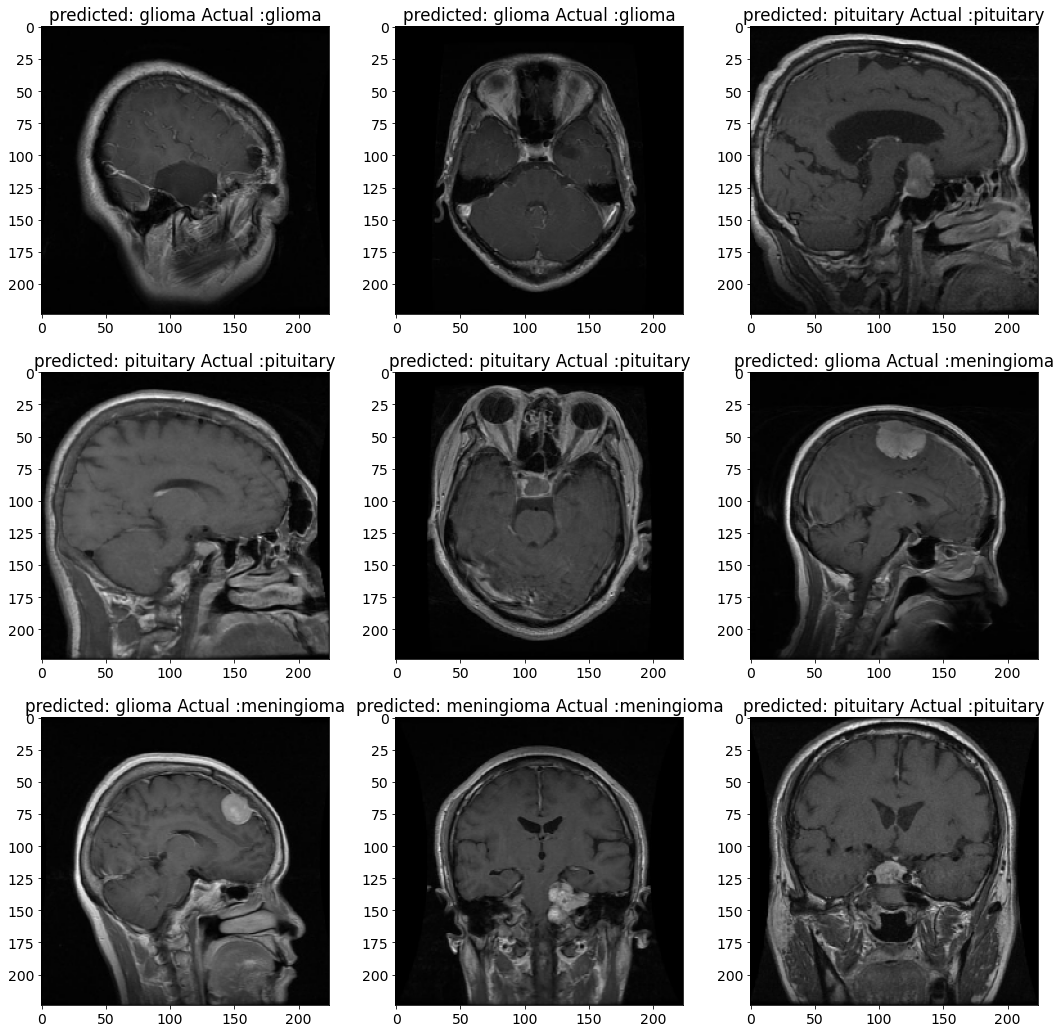
抽取部分样本进行测试, 结果如下

图18 部分数据测试结果

# 4 结论

为了进一步提升脑癌肿瘤 CAD 系统的性能, 在深度学习的基础上引入迁移学习, 并提出二次迁移学习, 提升了分类性能, 缓解了医学数据集引入深度学习困难的问题. 结果表明, 特定的迁移学习方式可以提升网络分类性能, 在可获得相似数据集的情况下, 数据联合训练可以提升分类性能, 二次迁移方法更加有效, 准确率均达到98%以上；迁移学习的特定引入方式、二次迁移学习、数据联合训练的方法均可为小数据集、医学影像数据集的分类任务提供参考依据.

# **参考文献**

* + 1. Masuod Nick. Brain Tumor MRI Dataset[EB/OL]. 2021[2022.06.01]. <https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset>.
    2. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[C]. International Conference on Learning Representations, 2015: 1-14.
    3. Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the Inception Architecture for Computer Vision[C]. IEEE Computer Vision and Pattern Pecognition, 2016: 2818-2826.
    4. He K, Zhang X, Ren S, et al. Deep residual learing for image recognition[C] IEEE Computer Vision and Pattern Pecognition, 2016: 770-778.
    5. 王敬, 张宝, 谢晓, 李健, 张莉, 郭华平. 基于广义稀疏逻辑回归的全脑分类[J/OL]. 信阳师范学院学报(自然科学版):1-7[2022-06-02].
    6. 潘嘉诚, 董一鸿, 陈华辉. 基于图神经网络的自闭症辅助诊断综述[J/OL]. 计算机工程: 1-13 [2022-06-02].
    7. 胡伟艺, 苏娴彦, 柯晓婷, 陈焱锋, 赖清泉. 基于深度学习的MRI诊断半月板损伤的研究进展[J]. 磁共振成像, 2022, 13(05): 167-170.
    8. 曾宪华, 纪聪辉, 董倩. 基于生成对抗网络和域一致性的MRI运动伪影校正方法[J]. 中国科学: 信息科学, 2022, 52(05): 822-836.
    9. 张元元, 杜科均, 屈直闯, 树海峰, 余思逊. 机器学习在辅助检测神经退行性疾病中的研究进展[J]. 中国医疗设备, 2022, 37(05): 157-160+169.
    10. 高婧, 毛宁, 谢海柱. 基于医学图像的影像组学和深度学习在乳腺癌淋巴结转移预测中的研究进展[J]. 中华放射学杂志, 2022, 56(05): 583-586.
    11. 唐红梅, 白梦月, 韩力英, 梁春阳. 基于低秩背景约束与多线索传播的图像显著性检测[J]. 电子与信息学报, 2021, 43(05): 1432-1440.

# 附 录

## 附录1：附件文件列表

|  |  |
| --- | --- |
| Jupyter文件 | 基于卷积神经网络的脑癌诊断.ipynb |
| VGG模型 | VGG16\_model.h5 |
| Resnet50模型 | Resnet50\_model.h5 |
| EfficientNetB0模型 | EfficientNetB0\_model.h5 |
| Python包版本 | requirements.txt |

## 附录2：程序运行环境

|  |  |
| --- | --- |
| 系统 | windows11, 64位操作系统 |
| CPU | Intel i5-9300H 2.4GHz 16G |
| GPU | NVIDIA GeForce GTX 1050 3G |

## 附录3：模型训练源程序 (详细见附件)

|  |  |
| --- | --- |
| **程序功能** | **传统方法搭建EfficientNetB0网络源程序** |
| effnet = EfficientNetB0(weights='imagenet',include\_top=False,input\_shape=(image\_size,image\_size,3))  model = effnet.output  model = tf.keras.layers.GlobalAveragePooling2D()(model)  model = tf.keras.layers.Dropout(rate=0.5)(model)  model = tf.keras.layers.Dense(4,activation='softmax')(model)  model = tf.keras.models.Model(inputs=effnet.input, outputs = model)  model.summary()  model.compile(loss='categorical\_crossentropy',optimizer = 'Adam', metrics= ['accuracy'])  tensorboard = TensorBoard(log\_dir = 'logs')  checkpoint = ModelCheckpoint("effnet.h5",monitor="val\_accuracy",save\_best\_only=True,mode="auto",verbose=1)  reduce\_lr = ReduceLROnPlateau(monitor = 'val\_accuracy', factor = 0.3, patience = 2, min\_delta = 0.001,  mode='auto',verbose=1)  epochs=20  batch\_size=16  history = model.fit(X\_train,y\_train,validation\_split=0.1, epochs =epochs, verbose=1, batch\_size=batch\_size,  callbacks=[tensorboard,checkpoint,reduce\_lr])  pred = model.predict(X\_test)  pred = np.argmax(pred,axis=1)  y\_test\_new = np.argmax(y\_test,axis=1)  print(classification\_report(y\_test\_new,pred)) | |

## 附录4：画图源程序

|  |  |
| --- | --- |
| **程序功能** | **绘制Accuracy和Loss随epochs变化情况** |
| filterwarnings('ignore')  epochs = [i for i in range(epochs)]  fig, ax = plt.subplots(1,2,figsize=(14,7))  train\_acc = history.history['accuracy']  train\_loss = history.history['loss']  val\_acc = history.history['val\_accuracy']  val\_loss = history.history['val\_loss']  fig.text(s='EfficientNetB0: Accuracy/Loss',size=18,fontweight='bold',  fontname='monospace',color=colors\_dark[1],y=1,x=0.28,alpha=0.8)  sns.despine()  ax[0].plot(epochs, train\_acc, marker='o',  markerfacecolor=colors\_green[2],  color=colors\_green[3],  label = 'Training Accuracy')  ax[0].plot(epochs, val\_acc, marker='o',  markerfacecolor=colors\_red[2],  color=colors\_red[3],  label = 'Validation Accuracy')  ax[0].legend(frameon=False)  ax[0].set\_xlabel('Epochs')  ax[0].set\_ylabel('Accuracy')  sns.despine()  ax[1].plot(epochs, train\_loss, marker='o',  markerfacecolor=colors\_green[2],  color=colors\_green[3],  label ='Training Loss')  ax[1].plot(epochs, val\_loss, marker='o',  markerfacecolor=colors\_red[2],  color=colors\_red[3],  label = 'Validation Loss')  ax[1].legend(frameon=False)  ax[1].set\_xlabel('Epochs')  ax[1].set\_ylabel('Training & Validation Loss')  fig.show() | |