基于 VGG 网络 SVM 肺部感染分类模型

生物科学与医学工程学院

2023 年 6 月 16 日

目录

1 摘要

新冠肺炎爆发以来， 我们对于肺部疾病的关注越来越多， 一个高效的肺部感染分类模型能帮助医生快速便捷判断是否感染。基于 VGG 网络能有效提取纹理特征、分辨更多细节等特点与 SVM(Support Vector Machine) 在二分类问题上的优越性， 我们利用 VGG 神经网络对肺部 X 光影像进行处理后利用 SVM 训练判断肺部是否感染模型，有助于判断肺部是否感染。

关键词：肺部感染， VGG 神经网络， SVM

2 研究背景

在过去三年全球都在经历新型冠状病毒 (COVID-19) 的流行， 而这种病毒一般会导致呼吸道系统出现症状， 其中肺炎是最常见的临床表现， 尤其对于众多老年人或自身就有肺部疾病的人， 感染新冠肺炎后往往会出现肺部大面积感染， 以致难以医治， 就是在新冠爆发时期我们多次听闻的“白肺”。因此， 快速而精准的判断一个人是否患有肺炎变得尤为重要。利用胸部 X 光数据集， 我们可以通过深度学习模型来进行自动化的肺炎检测。这个模型可以帮助医生提高诊断的准确性和效率，同时也能够为大众自检或判断提供更加方便快捷的方法。在肺部患病感染判断模型的基础上， 我们还可以进一步改进训练判断身体其他部位是否感染患病的模型。

3 模型原理

3.1 VGG 神经网络

VGG 网络是一种经典的卷积神经网络结构， 其全称为 Visual Geometry Group network。它的主要特点是采用了相同尺寸的小型卷积核和池化核， 并且具有比较深的网络结构。 VGG 网络在 2014 年 ILSVRC 比赛中取得了优异的成绩， 得到了广泛的应用。以下是一些关于 VGG 网络介绍：

1. VGG 网络结构十分简洁， 其基本架构是由一个或多个卷积层、池化层和全连接层组成， 其中每个卷积层使用相同数量的若干个卷积核进行卷积操作； 每个池化层则使用更大的池化核（通常是 2x2）来降低采样率。

2. VGG 网络通过使用多个卷积层和池化层来提取输入图像的高层次特征， 这些特征可以用于图像分类、目标检测等多个领域。

3. VGG 网络采用了一种预处理的方法，通过先训练一个简单的浅层网络（如 VGG11），然后使用这个网络的权重对更复杂的网络结构（如 VGG19）进行初始化，以便更快地收敛。

3.2 支持向量机 SVM

SVM（支持向量机， Support Vector Machine）是一种常见的机器学习算法， 其主要用于二分类问题和回归问题。 SVM 通过将数据映射到高维空间来寻找一个最优的超平面， 以将不同类别的样本分开。以下是一些关于 SVM 的介绍：

1. SVM 的核心思想是寻找一个最优的超平面来将不同类别的样本区分开。对于线性可分的情况， SVM 可以找到一个线性超平面来实现分类； 对于非线性可分的情况， SVM 可以通过核函数将原始数据映射到高维特征空间中，从而找到一个非线性超平面。

2. SVM 的只关注距离超平面最近的一些点， 这些点被称为支持向量。在训练模型时， SVM 会通过最大化支持向量到超平面的距离（即间隔）来寻找最优的超平面。

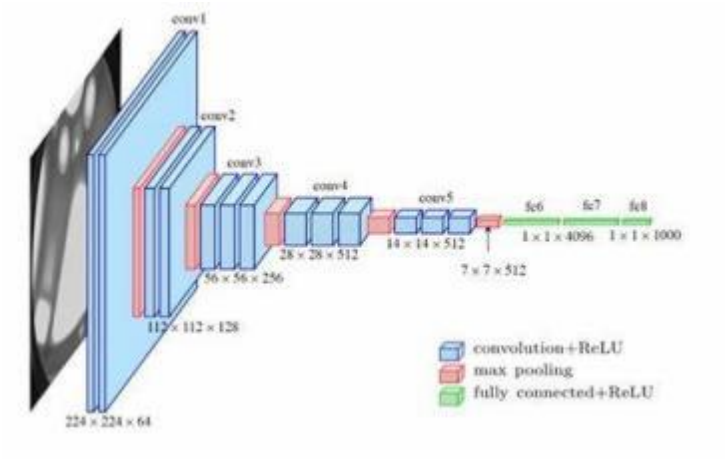

图 1: VGG 网络的结构

3. SVM 还具有一个重要的参数 C，它控制了模型的复杂度。当 C 很小时， SVM 会选择较简单的模型， 减少过拟合的风险； 而当 C 很大时， SVM 会选择更复杂的模型， 提高模型的泛化能力。

4. SVM 还可以通过使用核函数来处理非线性问题。常用的核函数包括线性核、多项式核、高斯核等， 它们可以将原始数据映射到高维空间中， 以便 SVM 寻找一个非线性的超平面。 SVM 是一种非常流行的机器学习算法， 其通过寻找最优的超平面来实现数据分类和回归。 SVM 不仅适用于线性可分问题，还可以通过核函数处理非线性问题，具有很强的泛化能力。

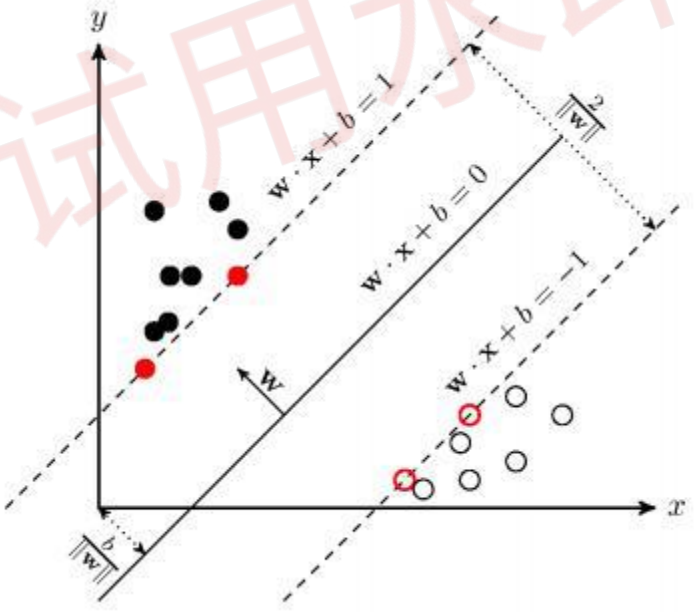

图 2: SVM 原理示意图

4 设计思路

由于肺部有肺、肋骨、气管、大量空气等诸多元素， 因此肺部 X 光影像相较于其他的医学影像，往往表现出很强的纹理性，在进行特征提取时往往需要注意更多的细节部分。

VGG 网络常用于判断肺部影像是因为其在图像识别方面表现优秀。 VGG 网络是一种深度卷积神经网络， 其结构简洁易懂， 使用了同样大小的卷积核尺寸(3x3) 和最大池化尺寸(2x2)， 便于调整和优化。这种网络结构使得 VGG 网络能够逐渐提取出图像特征， 从而更好地对图像进行分类、识别等任务。在肺部影像识别中， 医生通常需要通过对肺部图像进行分类或结构化分析来判断患者的病情。而用 VGG 网络对肺部图像进行训练， 则可以帮助医生对肺部影像进行分类、判断异常等。此外， VGG 网络还有一个优点是， 由于其网络结构简单， 训练时收敛的速度较快， 这对于处理海量数据非常有优势。因此， VGG 网络在肺部影像识别方面的应用非常广泛。

SVM 在分类问题上具有如下优点：

1. 非线性映射是 SVM 方法的理论基础， SVM 利用内积核函数代替向高维空间的非线性映射；

2. 对特征空间划分的最优超平面是 SVM 的目标， 最大化分类边际的思想是 SVM 方法的核心；

3. 支持向量是 SVM 的训练结果，在 SVM 分类决策中起决定作用的是支持向量；

4.SVM 是一种有坚实理论基础的新颖的小样本机器学习方法。它不仅能够解决线性、非线性分类问题，而且泛化能力强。

在医学领域中， SVM 常用于疾病诊断， 如肺结核、乳腺癌等疾病的诊断。 SVM 通过对患者的组织、体液等数据进行学习和验证， 建立模型， 从而实现对患者是否患病的判断。此外， 在 SARS、COVID-19 等传染性疾病的诊断中， 也可以通过 SVM 对患者进行分类, 从而排除疑似病例。这是因为 SVM 不但能够处理线性问题， 还能够有效地处理非线性问题， 将数据映射到高维空间中， 从而更好地发现数据之间的关系， 并给出相应的分类结果。因此， SVM 方法在医学领域中广泛应用，可以帮助医生快速和准确地进行疾病诊断，从而更好地保护人民的健康。

结合上述知识我们设计了这一深度学习神经网络模型， 利用 VGG 网络对肺部信息进行提取后利用 SVM 进行分类器训练，最后得到分类模型，并以此协助判断肺部是否患病。

5 环境配置及说明

5.1 数据库来源及简介

Chest X-Ray Images (Pneumonia) 是一个用于分类肺炎的 X 光图像数据集。该数据集由美国国立卫生研究院(NIH) 提供， 数据集中包含 5,863 张肺部 X 光片， 其中 4,396 张为训练集， 1,167 张为测试集。数据集中的肺部 X 光片大小不一， 但都已经统一调整为最小边长为 256 个像素， 且已经进行了灰度化处理。

5.2 环境配置

环境配置如表 1所示。

5.3 函数库说明

5.3.1 TensorFlow

TensorFlow（以下简称 TF）是由 Google 开源的一款机器学习框架， 它能够帮助开发人员构建和训练各种机器学习模型，如神经网络、深度学习等。 TensorFlow 的主要优点包括以下几点：

1. 灵活性： TensorFlow 支持多种编程语言， 包括 Python 、C++ 和 Java 等， 提供了丰富的 API 和工具，以便开发人员可以灵活地构建和调整机器学习模型。

2. 可扩展性： TensorFlow 能够轻松地部署到各种硬件平台上， 包括 CPU 、GPU 和 TPU 等，以便高效地进行训练和推理。

表 1: 环境配置

No	Environment	Version
1	Windows	11
2	Python	3.11
3	Anaconda3	2023.03
4	Pycharm	Community2023.1
5	Cuda	12.1
6	Pillow	9.5.0
7	Pip	23.1
8	scikit- learn	1.2.2
9	TensorFlow	2.12.0
10	numpy	3.2.2

3. 高性能： TensorFlow 可以通过使用诸如并行计算、硬件加速等技术来优化性能， 从而加快训练速度和提高准确性。

除此之外， TensorFlow 还具有良好的社区支持和文档资料， 这也使得它成为了一个备受欢迎的机器学习框架。

TensorFlow 最常用于图像处理方面的任务， 这是因为它支持高效的矩阵计算和卷积运算， 这些运算在图像处理中是必不可少的。此外， TensorFlow 还提供了许多常用的图像处理函数和工具，如卷积神经网络(CNN)、池化层、序列化图像数据等， 能够帮助开发人员高效地进行图像识别、分类和分割等任务。

5.3.2 Scikit-learn

Scikit-learn 是一款 Python 机器学习库， 它提供了大量的算法和工具， 用于分类、聚类、回归和降维等任务。 Scikit-learn 的主要优点包括以下几点：

1. 易于学习和使用： Scikit-learn 提供了 Python 的 API 接口， 易于 Python 程序员学习和使用，同时还有丰富的文档和代码示例，使得入门门槛较低。

2. 统一的 API 设计： Scikit-learn 采用了统一的 API 设计， 使得不同算法之间切换变得容易，还可以通过封装来简化模型训练的流程。

3. 丰富而又完整的算法实现： Scikit-learn 提供了多种经典和先进的机器学习算法的实现， 如逻辑回归、决策树、支持向量机、随机森林、神经网络等， 而且这些算法的实现都是经过优化和高度可靠性的。

因此， Scikit-learn 成为了一个备受欢迎的开源机器学习库， 被广泛应用于数据处理、特征提取、预测建模等领域。

Scikit-learn 也被广泛应用于图像处理方面的任务，这是因为 Scikit-learn 提供了许多常用的图像处理函数和算法， 如 PCA、LDA、KNN 等， 能够帮助开发人员进行图像分类、聚类和降维等任务。同时， Scikit-learn 还提供了丰富的特征提取和数据预处理方法， 使得机器学习模型在图像处理的任务中表现得更加优秀。

5.3.3 Pillow

Pillow 库（也称为 PIL）是一个 Python 图像处理库， 用于打开、操作和保存多种图像格式。Pillow 库提供了一系列的图像操作函数， 包括裁剪、缩放、旋转、合并、分割、滤镜等功能。此外，它还支持各种文件格式的图片读取和存储，如 BMP 、PNG 、JPEG 等。

5.3.4 Numpy

NumPy 库是一个 Python 数值计算库， 可以用于高效地处理矩阵和数组数据。 NumPy 提供了大量的数学函数和工具， 用于科学计算、数据分析和机器学习等领域。 NumPy 中的数组结构能够高效地存储和处理大量数据， 同时还支持广播、向量化运算和线性代数等操作。在图像处理中，NumPy 可以方便地将图像数据转换为数组形式，并进行各种处理和操作。

6 代码及说明

导入需要的函数库：

```
import numpy as np
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense, Dropout
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report
import tkinter as tk
from tkinter import filedialog
```

设置数据集路径：

```
train_dir = r"E:\ChestXRay2017\chest_xray\train"
test_dir = r"E:\ChestXRay2017\chest_xray\test"
```

图像预处理：

```
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_data = train_datagen.flow_from_directory(train_dir,
target_size=(224, 224), batch_size=32, classes=['NORMAL', 'PNEUMONIA'],
class_mode='binary')
test_data = test_datagen.flow_from_directory(test_dir,
target_size=(224, 224), batch_size=32, classes=['NORMAL', 'PNEUMONIA'],
class_mode='binary')
```

上述代码利用 tf 中 ImageDataGenerator 函数对像素值归一化处理， 将像素值从 [0,255] 归一化到 [0,1], 降低数据尺度， 提高效率。 flow_from_directory() 函数是从指定目录中读取训练数据。下面是对各个参数的解释： train_dir: 训练数据所在目录的路径。

target_size=(224, 224): 将输入的图片缩放为指定大小 (224x224) ，以适应网络的输入尺寸。

batch_size=32: 指定每次训练时，模型一次性读取多少张图片进行训练。

classes=['NORMAL', 'PNEUMONIA']: 指定所读取的图片所属类别的名称，此处将"NORMAL"和"PNEUMONIA" 分别作为 0 和 1 的标签。

class_mode='binary': 指定分类方式，此处为二进制分类。

最终， flow_from_directory() 方法返回一个数据生成器对象 train_data，该对象可以用于迭代训练集中的所有图片并将它们传递给神经网络进行训练。

VGG 模型搭建：

```
vgg_model = VGG16(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))
```

这一行代码创建了一个预训练的 VGG16 模型对象， 并且加载了在 imagenet 数据集上训练得到的权重。 include_top 参数被设置为 False，表示移除全连接层之后， 只保留卷积层结构， 后续我们需要自定义新的全连接输出层。 input_shape 参数指定输入图像的大小和通道数。

```
model = Sequential()
```

这一行代码定义了一个顺序模型对象，并将其赋值给变量 model。

```
model.add(vgg_model)
```

这一行代码将创建好的 VGG16 模型对象添加到新建的顺序模型中。

```
model.add(Flatten())
```

这一行代码向新建的顺序模型添加一个 Flatten 层， 用于将卷积层的输出展平， 形成一维向量，以便送入全连接层进行分类。

```
model.add(Dense(128, activation='relu'))
```

这一行代码向模型添加一个全连接层， 该层有 128 个神经元， 并使用 ReLU 函数作为激活函数。

```
model.add(Dropout(0.5))
```

这一行代码加入了一个 Dropout 层，用于在训练过程中防止过拟合。

```
model.add(Dense(1, activation='sigmoid'))
```

这一行代码添加了最终的全连接层， 该层只有一个神经元， 并使用 Sigmoid 函数作为激活函数，用于输出二分类结果的概率。

```
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy',
tf.keras.metrics.Precision(),
tf.keras.metrics.Recall(),
tf.keras.metrics.AUC()])
```

这一行代码设置了模型的优化器、损失函数和评价指标。其中， 优化器使用 Adam，损失函数使用二元交叉熵，评价指标包括准确率、精确率、召回率和 AUC 值。

```
model . summary ()
```

这一行代码使用 model.summary() 函数来输出模型的结构信息， 包括每一层的输出形状和参数数量等。

```
history = model . fi t ( train_data , epochs =1, validation_data=test_data )
```

这一行代码是模型训练过程。

特征提取：

```
train_features = model . predict ( train_data , verbose =1)
test_features = model . predict ( test_data , verbose =1)

train_labels = np . where ( train_data . c l a s se s == 0 , -1, 1)
test_labels = np . where ( test_data . c l a s se s == 0 , -1, 1)
```

pridict() 返回模型的预测结果,np.where() 是将标签由 0，1 改成我们所需要的-1，1。

SVM 模型训练：

```
svm_model = SVC( kernel= ' linear ' , random_state=10)
svm_model . fi t ( train_features , train_labels )
test_pred = svm_model . predict ( test_features )
acc = svm_model . score ( test_features , test_labels )
cm = confusion_matrix ( test_labels , test_pred )
cf_report = classification_report ( test_labels , test_pred )
```

上述代买完成了利用特征提取后的数据的 SVM 分类模型训练工作，并计算出相应指标。

判断所选肺部影像是否正常：

```
while True :
    root = tk .Tk ()
    root . withdraw ()

    new_dir = fi l e d i a l o g . askdirectory () # 打 开 文 件 夹对 话 框
    i f not new_dir :
        break
    new_data = test_datagen . flow_from_directory ( os . path . join (new_dir ,
    " test_new " ) , target_size =(224 , 224) , batch_size =1, s h u ffle=False , class_mod

    new_features = model . predict (new_data , verbose =1)
    new_pred = svm_model . predict ( new_features )

    for filename , pred in zip (new_data . filenames , new_pred ) :
        print ( " File ␣name : " , filename )
        i f pred == -1:
            print ( " This ␣ i s ␣a␣NORMAL␣ chest ␣X-ray . " )
        else :
            print ( " This ␣ i s ␣a␣PNEUMONIA␣ chest ␣X-ray . " )
```

上述代码提供文件选择窗口自主选择要判断的图像， 特征提取后利用 SVM 进行判断类型并输出结果。

7 程序运行结果

通常情况下我们将 epochs 设置为 20，以能得出比较准确的结果， 由于机器算力原因我们将 epoch 设置为 5，查看输出结果。运行代码会先出现模型参数等信息，如图 3：

```
Found 5232 images belonging to 2 classes.
Found 624 images belonging to 2 classes.
模型构造完成
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 128)               3211392

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 17,926,209
Trainable params: 17,926,209
Non-trainable params: 0
_________________________________________________________________
```

图 3: 模型参数及相关信息输出

然后会运行第一个 epoch，输出如图 4：其中， ETA 为预测该 epoch 运行剩余时间， loss 为损

```
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
117/164 [====================>.........] - ETA: 7:51 - loss: 0.8836 - accuracy: 0.7155 - precision: 0.7588 - recall: 0.9065 - auc: 0.5979
```

图 4: 模型指标输出

失函数， accuracy，precision，recall，auc 为准确率、精确率、召回率、 Area Under the Curve。模型每次迭代都会带入验证集验证一次所以能实时显示数值，并且观察 loss 函数是否收敛。

训练玩 5 个 epoch 之后会输出五次模型的指标，如图 5，

```
Epoch 1/10
164/164 [==============================] - 1682s 10s/step - loss: 0.6557 - accuracy: 0.7615 - precision: 0.8006 - recall: 0.9037 - auc: 0.7349 - val_loss: 0.8275 - val_accuracy:
Epoch 2/10
164/164 [==============================] - 1646s 10s/step - loss: 0.3709 - accuracy: 0.8581 - precision: 0.8753 - recall: 0.9619 - auc: 0.8878 - val_loss: 0.8049 - val_accuracy:
Epoch 3/10
164/164 [==============================] - 4486s 27s/step - loss: 0.1699 - accuracy: 0.9343 - precision: 0.9381 - recall: 0.9531 - auc: 0.9770 - val_loss: 0.7370 - val_accuracy:
Epoch 4/10
164/164 [==============================] - 1649s 10s/step - loss: 0.1504 - accuracy: 0.9442 - precision: 0.9451 - recall: 0.9593 - auc: 0.9816 - val_loss: 0.8242 - val_accuracy:
Epoch 5/10
164/164 [==============================] - 5291s 32s/step - loss: 0.1517 - accuracy: 0.9478 - precision: 0.9479 - recall: 0.9616 - auc: 0.9799 - val_loss: 0.4765 - val_accuracy:
```

图 5: 模型指标输出

表 2: 模型指标

No	Loss	Accuracy	Precision	Recall	AUC
epoch1	0.6557	0.7615	0.8006	0.9037	0.7349
epoch2	0.3709	0.8301	0.8733	0.9019	0.8878
epoch3	0.1699	0.9343	0.9581	0.9531	0.9770
epoch4	0.1504	0.9442	0.9653	0.9593	0.9816
epoch5	0.1517	0.9478	0.9679	0.9616	0.9799

该五次训练指标如表 2所示

可知在 epoth=5 时即模型可获得较好的指标，各项指标性能较好，具有较强的实用性。

由于能力问题，在最后打开文件并分类部分代码问题一直没得以解决。

8 模型优化

该模型目前只能用于简单的分类任务，对于该模型优化方案由如下几点：

1. 加入残差网络优化模型
2. 通过多层 SVM 进行不同程度感染分类
3. 通过 UNet 处理后进行语义分割可以判断病变部位

9 组员分工

：写代码及运行，报告撰写11320122 朱峥

：收集资料，报告撰写

10 附录

程序运行代码： chest.py

数 据 集 ： https: / / www. kaggle. com/ datasets/ paultimothymooney/ chest- xray- pneumonia