链滴

矩池云 | 新冠肺炎防控：肺炎 CT 检测

作者：matpool
原文链接：https://ld246.com/article/1582797071723
来源网站：链滴

连日来，新型冠状病毒感染的肺炎疫情，牵动的不仅仅是全武汉、全湖北，更是全国人民的心，大家纷以自己独特的方式为武汉加油！我们相信坚持下去，终会春暖花开。

今天让我们以简单实用的神经网络模型，来检测肺炎的CT影像。

第一步：导入我们需要的库

```
from keras.preprocessing.image import ImageDataGenerator, load_img
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, ZeroPadding2D, Conv2D, MaxPooling2D, Ac
ivation
from keras.optimizers import Adam, SGD, RMSprop
from keras.callbacks import EarlyStopping
from keras import backend as K

import tensorflow as tf
config = tf.ConfigProto()
config.gpu_options.per_process_gpu_memory_fraction = 0.9
K.tensorflow_backend.set_session(tf.Session(config=config))

import os
import numpy as np
import pandas as np
import cv2

from glob import glob

import matplotlib.pyplot as plt
%matplotlib inline
```

第二步：数据查看

2.1 先确认下我们数据的目录结构：

在chest_xray文件夹中，我们将数据分成了训练病例数据(train), 测试病例数据(test), 验证病例数据(val
；

每个训练数据，测试数据，验证数据的文件夹中我们又分成了正常的病例数据(normal), 肺炎病例数据
pneumonia)。

```
print("训练病例数据")
print(os.listdir("chest_xray"))
print(os.listdir("chest_xray/train"))
print(os.listdir("chest_xray/train/"))
```

训练病例数据

['test', 'train', 'val', '.DS_Store']

['NORMAL', '.DS_Store', 'PNEUMONIA']

['NORMAL', '.DS_Store', 'PNEUMONIA']

```
print("测试病例数据")
print(os.listdir("chest_xray"))
print(os.listdir("chest_xray/test"))
print(os.listdir("chest_xray/test/"))
```

测试病例数据

['test', 'train', 'val', '.DS_Store']

['NORMAL', '.DS_Store', 'PNEUMONIA']

['NORMAL', '.DS_Store', 'PNEUMONIA']

```
print("验证病例数据")
print(os.listdir("chest_xray"))
print(os.listdir("chest_xray/val"))
print(os.listdir("chest_xray/val/"))
```

验证病例数据

['test', 'train', 'val', '.DS_Store']

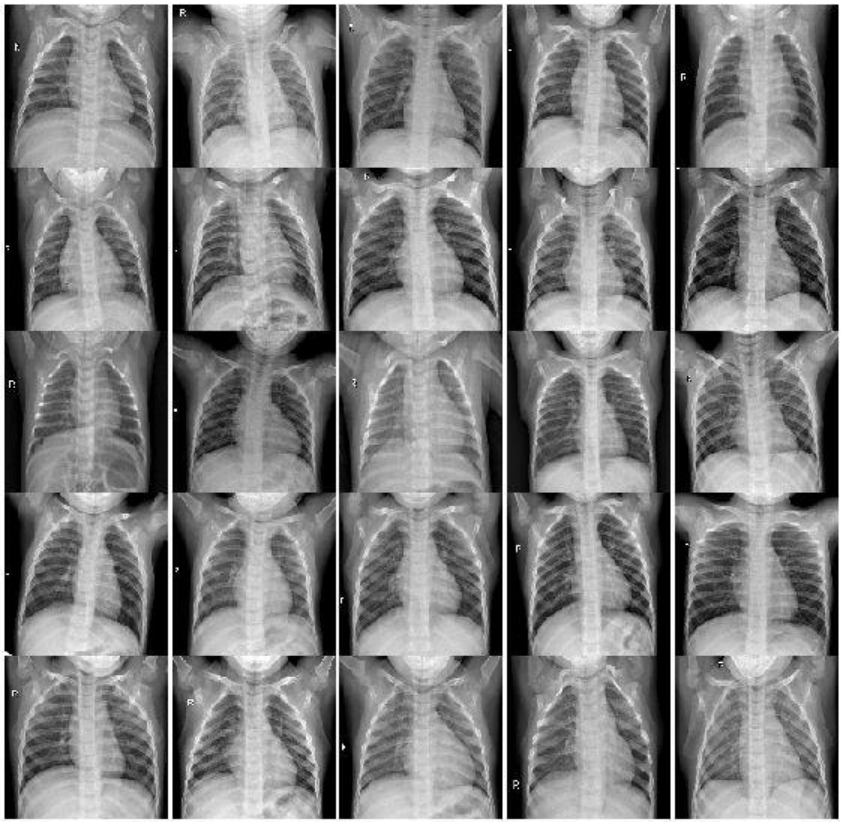
['NORMAL', '.DS_Store', 'PNEUMONIA']

['NORMAL', '.DS_Store', 'PNEUMONIA']

2.2 用matpolt 来可视化我们的病例数据：

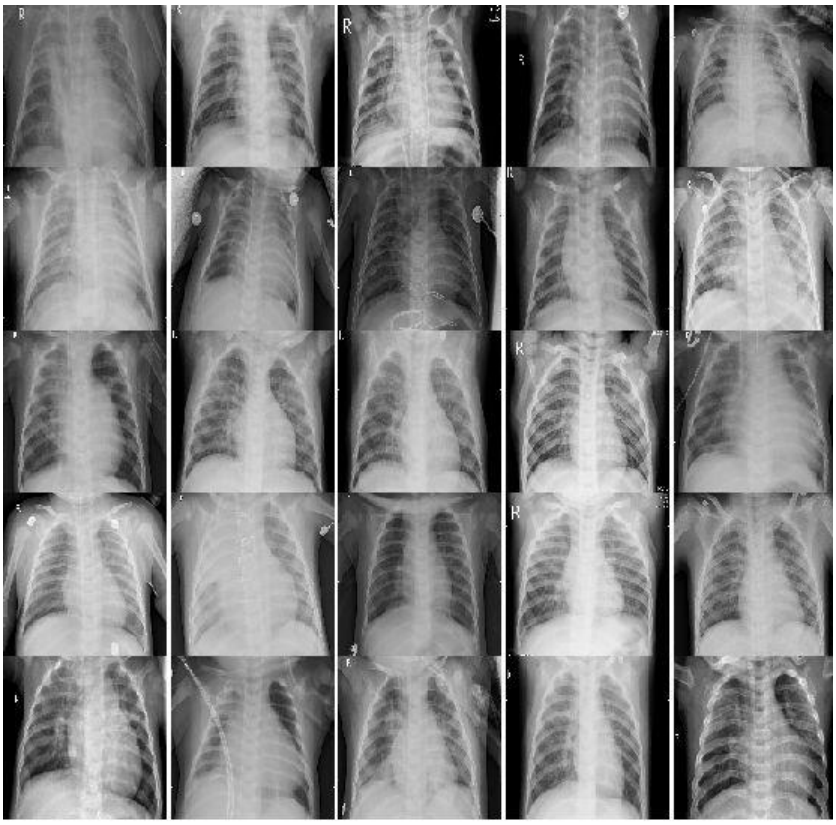
2.2.1 没有肺炎的20个病例的CT图片：

```
multipleImages = glob('chest_xray/train/NORMAL/**')
i_ = 0
plt.rcParams['figure.figsize'] = (10.0, 10.0)
plt.subplots_adjust(wspace=0, hspace=0)
for l in multipleImages[:25]:
    im = cv2.imread(l)
    im = cv2.resize(im, (128, 128))
    plt.subplot(5, 5, i_+1) #.set_title(l)
    plt.imshow(cv2.cvtColor(im, cv2.COLOR_BGR2RGB)); plt.axis('off')
    i_ += 1
```



2.2.2 有肺炎的20个病例的CT图片：

```
multipleImages = glob('chest_xray/train/PNEUMONIA/**')
i_ = 0
plt.rcParams['figure.figsize'] = (10.0, 10.0)
plt.subplots_adjust(wspace=0, hspace=0)
for l in multipleImages[:25]:
    im = cv2.imread(l)
    im = cv2.resize(im, (128, 128))
    plt.subplot(5, 5, i_+1) #.set_title(l)
    plt.imshow(cv2.cvtColor(im, cv2.COLOR_BGR2RGB)); plt.axis('off')
    i_ += 1
```



第三步：数据预处理

3.1 首先先定义一些我们需要使用到的变量

```
# 图片尺寸
image_width = 226
image_height = 226
```

3.2 处理下图片的通道数在输入数据中的格式问题

```
if K.image_data_format() == 'channels_first':
    input_shape = (3, image_width, image_height)
else:
    input_shape = (image_width, image_height, 3)
```

3.3 数据加载和增强

这个案例，我们使用Keras的

ImageDataGenerator来加载我们的数据，并且做数据增强跟处理。

ImageDataGenerator

是keras.preprocessing.image模块中的图片生成器，同时也可以在batch中对数据进行增强，扩充据集大小，增强模型的泛化能力。比如进行旋转，变形，归一化等。

3.3.1 定义训练数据的ImageDataGenerator

- 训练数据的ImageDataGenerator我们做了如下几个处理：
- 将像素值归一化 (rescale)

- 剪切强度(逆时针方向的剪切变换角度)，强度为0.2
- 随机缩放的幅度, 当前为0.2
- 水平翻转

```
train_data_gen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)
```

3.3.2 定义测试数据的ImageDataGenerator

测试数据我们只做了归一化处理

```
test_data_gen = ImageDataGenerator(rescale=1. / 255)
```

3.4 使用我们定义好的ImageDataGenerator从文件夹中读取数据, 其中target_size参数会把我们读的原始数据缩放到我们想要的尺寸

3.4.1 训练数据读取

```
train_generator = train_data_gen.flow_from_directory(
    'chest_xray/train',
    target_size=(image_width, image_height),
    batch_size=16,
    class_mode='categorical')
```

Found 5216 images belonging to 2 classes.

3.4.2 验证数据读取

```
validation_generator = test_data_gen.flow_from_directory(
    'chest_xray/val',
    target_size=(image_width, image_height),
    batch_size=16,
    class_mode='categorical')
```

Found 16 images belonging to 2 classes.

3.4.3 测试数据读取

```
test_generator = test_data_gen.flow_from_directory(
    'chest_xray/test',
    target_size=(image_width, image_height),
    batch_size=16,
    class_mode='categorical')
```

Found 624 images belonging to 2 classes.

第四步：模型构建

4.1 定义我们的模型

我们的模型层次采用VGG16 网络模型，

原模型链接：

https://gist.github.com/baraldilorenzo/07d7802847aaad0a35d3

4.1.1 VGG

VGG是由Simonyan 和Zisserman在文献《Very Deep Convolutional Networks for Large Scale Image Recognition》中提出卷积神经网络模型，其名称来源于作者所在的牛津大学视觉几何组(Visual Geometry Group)的缩写。

该模型参加2014年的 ImageNet图像分类与定位挑战赛，取得了优异成绩：在分类任务上排名第二在定位任务上排名第一。

VGG结构图：

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

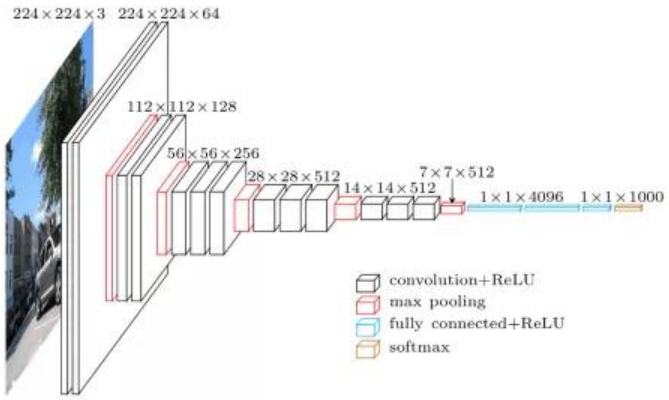
4.1.2 VGG16

VGG模型有一些变种，其中最受欢迎的当然是 VGG-16，这是一个拥有16层的模型。你可以看到它要维度是 224x224x3 的输入数据。

VGG16输入224x224x3的图片，经过的卷积核大小为3x3x3，stride=1，padding=1，pooling为采2x2的Max Pooling方式：

- 输入224x224x3的图片，经过64个卷积核的两次卷积后，采用一次Max Pooling
- 再经过两次128的卷积核卷积之后，采用一次Max Pooling
- 再经过三次256的卷积核的卷积之后，采用Max Pooling
- 重复两次三个512的卷积核卷积之后再Max Pooling
- 三次FC

VGG 16结构图



下面我们使用Keras 建立VGG 16 模型

```
model = Sequential()
model.add(ZeroPadding2D((1,1),input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
```

```
model.add(Conv2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(512, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(512, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(Flatten())
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))
```

4.2 查看下模型概况

```
model.summary()
```

4.3 编译模型

我们使用Adam优化器，并且设置learning rate为0.0001，对验证集的精确度添加early stopping m nitor，并且patinece设置成3，这个参数的意思，当我们有3个连续的epochs没有提升精度，我们就止训练，防止过拟合。

```
optimizer = Adam(lr = 0.0001)
early_stopping_monitor = EarlyStopping(patience = 3, monitor = "val_accuracy", mode="max"
 verbose = 2)
model.compile(loss="categorical_crossentropy", metrics=["accuracy"], optimizer=optimizer)
```

第五步：肺炎CT模型训练

5.1 训练模型

```
history = model.fit_generator(epochs=5, callbacks=[early_stopping_monitor], shuffle=True,
                              validation_data=validation_generator, generator=train_generator,
                              steps_per_epoch=500, validation_steps=10,verbose=2)
```

5.2 模型在训练过程中，训练数据集的精度和损失值会发生变化。

有次可见，我们的模型在训练的时候，精度不断提高，因此看到我们的模型在逐渐收敛到最佳的状态。

```
plt.plot(history.history['accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train'], loc='upper left')
```

```
plt.show()

plt.plot(history.history['loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['train'], loc='best')
plt.show()
```

第六步： 模型在测试集数据上的使用

```
scores = model.evaluate_generator(test_generator)
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

具体的数据结果，欢迎各自进行尝试实验

当前 “新冠肺炎防控-肺炎CT检测” 案例镜像已经在矩池云GPU云共享平台正式上线。

感兴趣的小伙伴可以通过官网“机器租赁” — “我要租赁” — “选择镜像” — “Jupyter 教程 Demo”中尝试使用。