文章
Michael Lei · 五月 10, 2021 阅读大约需 10 分钟

通过深度学习解释和研究 Covid-19 X 射线分类器

关键字：深度学习，Grad-CAM，X 射线，Covid-19，HealthShare，IRIS

目的

在复活节周末，我谈到了一些针对 Covid-19 肺的深度学习分类器。
演示结果还算不错，似乎与当时有关该主题的一些学术研究刊物相吻合。 但它真的足够“ 好 ”吗？

最近，我偶然收听了一个关于“ 机器学习中的可解释性 ”的在线午餐网络讲座，Don
在演讲的最后谈到了这个分类结果：

(a) Husky classified as wolf (b) Explanation

Figure 11: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task.

	Before	After
Trusted the bad model	10 out of 27	3 out of 27
Snow as a potential feature	12 out of 27	25 out of 27

Table 2: "Husky vs Wolf" experiment results.

上图也出现在“ Why Should I Trust You? ” Explaining the Predictions of Any Classifier 这篇研究论文中。
我们可以看到，分类器实际上经过训练，以背景像素（如雪等野生环境）作为主要输入，对宠物狗和野狼进行分类。

这关乎我过去的兴趣，现在也激起一些好奇：

- 我们如何“ 观察 ”这些通常以“ 黑盒 ”形式表示的 Covid-19 分类器，了解哪些像素实际上促成了“ Covid-19 肺 ”结果？
- 在这种情况下，我们可以利用的最简单的形式或工具是什么？

这也是篇简单的 10 分钟笔记。 最后，我会谈到为什么它也与我们即将推出的全新 IRIS 和 HealthShare 功能有关。

范围

幸运的是，过去几年中，各种 CNN 衍生分类器都有了方便的工具：

- CAM（类激活图）：其应用在这里和这里得到了充分解释。
- Grad-CAM（梯度加权类激活）：这是 CAM 更通用的版本，让我们能够观察整个模型中的任何 CNN 层。

我们将使用 **Grad-CAM** 对我们上一篇帖子中的 Covid-19 肺分类器进行快速演示。

"Tensorflow 2.2.0rc + Jupyter" Docker 在配备 Nvidia T4 GPU 的 AWS Ubuntu 16.04 服务器上使用。 TensorFlow 2 提供了简单的**梯度带**实现。

这是我在 Ubuntu 服务器上启动的快速笔记：

docker run -itd --runtime=nvidia -v /zhong/tf/:/tf -p 8896:8888 -p 6026:6006 --name tf-gpu2 tensorflow/tensorflow:2.2.0rc2-gpu-py3-jupyter

方法

您可以放心地在此处忽略以上 Grad-CAM 研究出版物中引用的数字。

这里引用这些数字只是为了对后面使用的 Python 代码进行连续的原始提案（第 4 页和第 5 页）
交叉检查，也希望能提供更好的结果透明度。

(1)：为了得到任意类 c 的宽度 u 和高度 v 的类判别定位图，我们首先计算类 c 的得分相对于卷积层的特征图 Ak 的梯度 yc（softmax 前）。 这些回流的梯度被全局平均池化，得到目标类的神经元重要性权重 ak。

(2)：计算出目标类 c 的 ak 后，我们对激活图进行加权组合，然后执行 ReLU。
这就得到了与卷积特征图大小相同的粗略热图。

测试

现在，我们来尝试一下目前能找到的最简单的编码：

1. 导入软件包

```
import tensorflow as tf;
print(tf.version)
```

2.2.0-rc2

```
import tensorflow as tf
import tensorflow.keras.backend as K
```

from tensorflow.keras.applications.inceptionv3 import InceptionV3
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.inceptionv3 import preprocessinput, decodepredictions
import numpy as np
import os
import imutils
import matplotlib.pyplot as plt
import cv2

2. 加载我们之前训练和保存的模型

newmodel = tf.keras.models.loadmodel('savedmodel/inceptionV3')
newmodel.summary()

可以看到，在最终的全局平均池化之前，模型中 4D 的最后一个 CNN 层被称为 "**mixed10** " 。

3. 计算 Grad-CAM 热图

下面是一个实现了上述 Grad-CAM 公式 (1) 和 (2) 的简单版本热图。 这篇帖子对其做出了解释。

with tf.GradientTape() as tape:
 lastconvlayer = model.getlayer('mixed10')
 iterate = tf.keras.models.Model([model.inputs], [model.output, lastconvlayer.output])
 modelout, lastconvlayer = iterate(testX)
 classout = modelout[:, np.argmax(modelout[0])]
 grads = tape.gradient(classout, lastconvlayer)
 pooledgrads = K.mean(grads, axis=(0, 1, 2))
heatmap = tf.reducemean(tf.multiply(pooledgrads, lastconvlayer), axis=-1)

在我们的示例中，它将生成一个热图 NumPy 数组 (27, 6, 6)。 然后，我们可以将它重新调整为原始 X 射线图像尺寸并叠加在 X 射线图像上方。

不过，在这种情况下，我们将使用略详细的版本，这篇帖子对此也有很好的解释。 它组成了一个函数，Grad-CAM 热图已调整为原始 X 射线图的大小：

# import the necessary packages
from tensorflow.keras.models import Model
import tensorflow as tf
import numpy as np
import cv2

class GradCAM:
 def init(self, model, classIdx, layerName=None):
 self.model = model
 self.classIdx = classIdx
 self.layerName = layerName
 if self.layerName is None:
 self.layerName = self.findtargetlayer()

def findtargetlayer(self):
 for layer in reversed(self.model.layers):
 # check to see if the layer has a 4D output
 if len(layer.outputshape) == 4:
 return layer.name
raise ValueError("Could not find 4D layer. Cannot apply GradCAM.")

def computeheatmap(self, image, eps=1e-8):
 gradModel = Model(
 inputs=[self.model.inputs],
 outputs=[self.model.getlayer(self.layerName).output,
 self.model.output])
record operations for automatic differentiation
 with tf.GradientTape() as tape:
 inputs = tf.cast(image, tf.float32)
 (convOutputs, predictions) = gradModel(inputs)
 loss = predictions[:, self.classIdx]
**# use automatic differentiation to compute the gradients
 grads = tape.gradient(loss, convOutputs)**
# compute the guided gradients
 castConvOutputs = tf.cast(convOutputs > 0, "float32")
 castGrads = tf.cast(grads > 0, "float32")
 guidedGrads = castConvOutputs * castGrads * grads
convOutputs = convOutputs[0]
 guidedGrads = guidedGrads[0]
weights = tf.reducemean(guidedGrads, axis=(0, 1))
 cam = tf.reducesum(tf.multiply(weights, convOutputs), axis=-1)

resize the heatmap to oringnal X-Ray image size
 (w, h) = (image.shape[2], image.shape[1])

heatmap = cv2.resize(cam.numpy(), (w, h))

# normalize the heatmap
 numer = heatmap - np.min(heatmap)
 denom = (heatmap.max() - heatmap.min()) + eps
 heatmap = numer / denom
 heatmap = (heatmap * 255).astype("uint8")

return the resulting heatmap to the calling function
 return heatmap

4. 加载 Covid-19 肺部 X 射线图

现在，加载一个从未在模型训练和验证过程中使用过的测试 X 射线图。 （也已上传到上一篇帖子中）

filename = './test/nejmoa2001191f1-PA.jpeg'
orignal = cv2.imread(filename)
plt.imshow(orignal)
plt.show()

调整为 256 x 256，归一化为像素值在 0.0 到 1.0 之间的 numpy 数组 “ dataXG ”。

orig = cv2.cvtColor(orignal, cv2.COLORBGR2RGB)
resized = cv2.resize(orig, (256, 256))
dataXG = np.array(resized) / 255.0
dataXG = np.expanddims(dataXG, axis=0)

5. 进行快速分类

现在可以调用上面新加载的模型进行快速预测：

preds = newmodel.predict(dataXG)
i = np.argmax(preds[0])
print(i, preds)

0 [[0.9171522 0.06534185 0.01750595]]

因此它被归类为 0 型 - Covid-19 肺，概率为 0.9171522。

6. 计算 Grad-CAM 热图

# Compute the heatmap based on step 3
cam = GradCAM(model=newmodel, classIdx=i, layerName='mixed10') # find the last 4d shape "mixed10" in this case
heatmap = cam.computeheatmap(dataXG)

#show the calculated heatmap
plt.imshow(heatmap)
plt.show()

7. 在原始 X 射线图上显示热图

# Old fashioned way to overlay a transparent heatmap onto original image, the same as above
heatmapY = cv2.resize(heatmap, (orig.shape[1], orig.shape[0]))
heatmapY = cv2.applyColorMap(heatmapY, cv2.COLORMAPHOT) # COLORMAPJET, COLORMAPVIRIDIS, COLORMAPHOT
imageY = cv2.addWeighted(heatmapY, 0.5, orignal, 1.0, 0)
print(heatmapY.shape, orig.shape)

# draw the orignal x-ray, the heatmap, and the overlay together
output = np.hstack([orig, heatmapY, imageY])
fig, ax = plt.subplots(figsize=(20, 18))
ax.imshow(np.random.rand(1, 99), interpolation='nearest')
plt.imshow(output)
plt.show()

(842, 1090, 3) (842, 1090, 3)

这似乎表明我们的 Covid-19 演示分类器 “ 相信 ” 患者的 “ 右侧气管旁带 ” 周围出现了一些 “ 浑浊 ” 问题？

我不是很明白，这要请教真正的放射科医生。

那么，接下来再尝试一些从现实世界案例提交到 GitHub 仓库中的测试图像：

filename = './test/1-s2.0-S0929664620300449-gr2lrg-b.jpg'

0 [[9.9799889e-01 3.8319459e-04 1.6178709e-03]]

这似乎也是合理的 Covid-19 解释，表明问题更多地发生在左心线区域？

再试试另一个随机测试 X 射线图：

filename = '../CovidM/all/test/covid/radiol.2020200490.fig3.jpeg'

0 [[0.9317619 0.0169084 0.05132957]]

没想到这并不完全正确，但看起来好像也不算太离谱？
它显示了两个问题区域，左侧为主要问题，右侧为部分问题，这与放射科医师的标记应该是有些对应的？
（希望它不是在人类标记上训练 - 这是可解释性问题的另一个层面）。

我要在这里打住了，我猜不会有太多人对 X 射线感兴趣。

原因

我个人对“ 可解释性 ”和“ 可理解性 ”以及相关技术方法的重要性有着深切的体会。
在此领域的任何尝试都是值得的，无论它有多么微不足道。
最终，“ 数据公平 ”、“ 数据公正 ”和“ 数据信任 ”将建立在其数字经济过程透明化的基础之上。
另外，它现在开始能为人所用了。 25 年前，当年轻的我在 1995
年的夏天忙着写博士论文的时候，我甚至不指望对被广泛用作黑盒的所谓“ 神经网络 ”有任何了解。 当时的 AI
更像是逻辑推理机“ 专家系统 ”，“ 神经网络 ”只是被称为“ 神经网络 ”，而“ 深度学习 ”尚未诞生。
现在，越来越多的研究和工具不断涌现，可供 AI 开发者轻松使用。

最后，具体到这个演示，我很欣赏这种工具的一点是，它甚至不需要以像素级的标记作为起点，而是试图自动生成肺部病变区域，实现一种类似半自动标记的效果。 这在实际应用中是有意义的。
我记得去年，我的一位放射科医生朋友为了帮助我获取一些骨折数据，一遍又一遍地生成 U-Net
训练的像素标签，这很伤眼睛。

未来计划

回到正题。 得益于过去 10 多年深度学习的快速发展，医学影像已成为 AI 领域比较成熟的方向。
这是值得我们深入研究的。 不过，接下来如果有时间，我希望我们能在 NLP 方面多做一些尝试。

致谢

所有来源均已根据需要插入上述文本。 如有其他需要，我还将提供更多引用。

免责声明：

再次说明，我现在写这篇快速笔记是为了防止相关信息随时间遗失。 本文完全是出自“ 开发者 ”角度的个人观点。
内容和文本随时可能会被更改或完善。
以上内容主要是展示技术思想和方法，而不是临床解释。临床解释需要放射科专家根据大量优质数据建立黄金规则。

#HealthShare #InterSystems IRIS #InterSystems IRIS for Health

---

源 URL: https://cn.community.intersystems.com/post/%E9%80%9A%E8%BF%87%E6%B7%B1%E5%BA%A6%E5%AD%A6%E4%B9%A0%E8%A7%A3%E9%87%8A%E5%92%8C%E7%A0%94%E7%A9%B6-covid-19-x-%E5%B0%84%E7%BA%BF%E5%88%86%E7%B1%BB%E5%99%A8