

文章

[Hao Ma](#) · 三月 25, 2021 阅读大约需 12 分钟

Covid-19 肺部 X 射线分类和 CT 检测演示 (HealthShare 临床查看器与第三方AI PACS viewer整合)

Covid-19 肺部 X 射线分类和 CT 检测演示 **关键字**：COVID-19，医学影像，深度学习，PACS Viewer 和 HealthShare。

目的

在这场史无前例的新冠疫情笼罩之下，我们竭尽所能为客户提供支援，同时利用先进的 AI 技术观察着不同的疫情战线。

去年，我简单提及了一个[深度学习演示环境](#)。在这个漫长的复活节周末，我们就来看一看现实世界的图像，在 Covid-19 肺部 X 射线数据集上测试运行一些深度学习模型以进行快速分类，并见证这类用于 X 射线甚至 CT 的工具如何通过 docker 等方式快速部署到云端，实现及时的“AI 分诊”并协助放射科医生。

这只是一个 10 分钟快速笔记，希望通过简单的方法帮助各位上手实践。

范围

本演示环境中使用了以下组件，这是我目前能找到的最简单的形式：

- 一个小型**匿名开放数据集**，共 3 种类型：Covid-19 肺与细菌性肺炎肺与正常透明肺。
- 一组**深度学习模型**，如用于肺部 X 射线分类的 Inception V3 模型
- 带有 Jupyter Notebook 的 **Tensorflow 1.13.2** 容器
- 用于 **GPU 工具**的 **Nvidia-Docker2** 容器
- 配备 Nvidia T4 GPU 的 **AWS Ubuntu 16.04 VM**（如果不重新训练预训练的模型，笔记本电脑的 GPU 就足够了）

以及，

- “AI 辅助 CT 检测”的演示容器。
- 第三方 Open PACS Viewer 的演示容器。
- HealthShare Clinical Viewer 的演示实例。

以下不在演示范围内：

- PyTorch 越来越受欢迎（下次会用到）
- TensorFlow 2.0 在演示环境中的运行速度过慢（因此我暂时回转到 1.13 版）
- AutoML 等多模型集成（它们在现实世界中越来越流行，但单一老式模型对这个小数据集来说已经足够了）
- 任何现实世界位置的 X 射线和 CT 数据。

免责声明

这个演示更多是关于技术方法，而不是特定领域的临床试验。基于 CT 与 X 线等证据的 Covid-19 检测已在网上广泛流传，对其有正面评价，也有负面评价。疫情期间，它在各个国家/地区和文化中发挥着不同的作用。

另外，本文的正文和布局可能会根据需要进行修改。 本文完全是出自“开发者”角度的个人观点。

数据

本测试的原始图像来自 [Joseph Paul Cohen](#) 公开的 [Covid-19 肺部 X 射线集](#)，以及开放的

Kaggle 胸部 X 射线集1>中的一些干净的肺，由 Adrian Yu 在 GradientCrescent 仓库1>中收集为一个小型测试集。我[在这里上传了测试数据](#)，供有兴趣的读者进行快速测试。到目前为止，它只包含一个小的训练集：

- 60x Covid-19 肺
- 70x 正常透明肺
- 70x 细菌性肺炎肺

测试

在以下测试中，我根据自己的情况做了些微调：

- Inception V3 模型作为基础，几个 CNN 层作为顶层
- 使用未冻结的底层 Inception 层权重进行迁移学习，以再训练（如果是在笔记本电脑 GPU 上，您只需要冻结预训练的 Inception 层）
- 为弥补目前收集到的少量数据集，略微增加了一些内容。
- 3x 类别而不是二元类别：Covid-19、正常与细菌性（或病毒性）肺炎（我将解释为什么是这三类）
- 计算基本的 3 类混淆矩阵，作为后续可能步骤的模板。

注：选择 InceptionV3 而不是其他流行的基于 CNN 的模型，比如 VGG16 或 ResNet50，并没有什么特别的原因。我只是碰巧最近在其他模型中用它来演示运行了一个骨折数据集，为方便起见就直接重用了。您可以使用[任何偏好的模型](#)重新运行以下 Jupyter Notebook 脚本。

我也在这篇帖子中附加了以下 Jupyter Notebook 文件。下面也是为了快速说明。

1. 导入必要的库：

```
# import the necessary packages
from tensorflow.keras.layers import AveragePooling2D, Dropout, Flatten, Dense, Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import optimizers, models, layers
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.resnet50 import ResNet50

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import cv2
import os
```

2. 加载[此处提供的示例图像文件](#)

```
# set learning rate, epochs and batch size
```

```
INIT_LR = 1e-5    <span style="color:#999999;"># This value is specific to what mo
del is chosen: Inception, VGG or ResNet etc.</span>
EPOCHS = 50
BS = 8

print("Loading images...")
imagePath = "./Covid_M/all/train" # change to your local path for the sample images
imagePaths = list(paths.list_images(imagePath))

data = []
labels = []

# read all X-Rays in the specified path, and resize them all to 256x256
for imagePath in imagePaths:
    label = imagePath.split(os.path.sep)[-2]
    image = cv2.imread(imagePath)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = cv2.resize(image, (256, 256))
    data.append(image)
    labels.append(label)

#normalise pixel values to real numbers between 0.0 - 1.0
data = np.array(data) / 255.0
labels = np.array(labels)

# perform one-hot encoding for a multi-class labeling
label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(labels)
labels = to_categorical(integer_encoded)

print("... .. ", len(data), "images loaded in multiple classes:")
print(label_encoder.classes_)

Loading images...
... .. 200 images loaded in 3x classes:
['covid' 'normal' 'pneumonia_bac']
```

3. 添加基本数据增强, 重新构建模型, 然后开始训练

```
# split the data between train and validation.
(trainX, testX, trainY, testY) = train_test_split(data, labels, test_size=0.20, strat
ify=labels, random_state=42)

# add on a simple Augmentation. Note: too many Augumentation doesn't actually help in
this case - I found during the test.
trainAug = ImageDataGenerator(rotation_range=15, fill_mode="nearest")

#Use the InveptionV3 model with Transfer Learning of pre-
trained "ImageNet"'s weights.
#note: If you choose VGG16 or ResNet you may need to reset the initial learning rate
at the top.
baseModel = InceptionV3(weights="imagenet", include_top=False, input_tensor=Input(sha
pe=(256, 256, 3)))
#baseModel = VGG16(weights="imagenet", include_top=False, input_tensor=Input(shape=(2
56, 256, 3)))
#baseModel = ResNet50(weights="imagenet", include_top=False, input_tensor=Input(shape
=(256, 256, 3)))

#Add on a couple of custom CNN layers on top of the Inception V3 model.
```

```
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(4, 4))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(64, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(3, activation="softmax")(headModel)

# Compose the final model
model = Model(inputs=baseModel.input, outputs=headModel)

# Unfreeze pre-trained Inception "ImageNet" weights for re-
training since I got a Nvidia T4 GPU to play with anyway
#for layer in baseModel.layers:
#    layer.trainable = False

print("Compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="categorical_crossentropy", optimizer=opt, metrics=["accuracy"])

# train the full model, since we unfroze the pre-trained weights above
print("Training the full stack model...")
H = model.fit_generator( trainAug.flow(trainX, trainY, batch_size=BS), steps_per_epoc
h=len(trainX) // BS,
                        validation_data=(testX, testY), validation_steps=len(testX)
                        // BS, epochs=EPOCHS)

... ..
Compiling model...
Training the full stack model...
... ..
Use tf.cast instead.
Epoch 1/50
40/40 [=====] - 1s 33ms/sample - loss: 1.1898 - acc: 0.3000
20/20 [=====] - 16s 800ms/step - loss: 1.1971 - acc: 0.3812
- val_loss: 1.1898 - val_acc: 0.3000
Epoch 2/50
40/40 [=====] - 0s 6ms/sample - loss: 1.1483 - acc: 0.3750
20/20 [=====] - 3s 143ms/step - loss: 1.0693 - acc: 0.4688 -
val_loss: 1.1483 - val_acc: 0.3750
Epoch 3/50
... ..
... ..
Epoch 49/50
40/40 [=====] - 0s 5ms/sample - loss: 0.1020 - acc: 0.9500
20/20 [=====] - 3s 148ms/step - loss: 0.0680 - acc: 0.9875 -
val_loss: 0.1020 - val_acc: 0.9500
Epoch 50/50
40/40 [=====] - 0s 6ms/sample - loss: 0.0892 - acc: 0.9750
20/20 [=====] - 3s 148ms/step - loss: 0.0751 - acc: 0.9812 -
val_loss: 0.0892 - val_acc: 0.9750
```

4. 为验证结果绘制混淆矩阵：

```
print("Evaluating the trained model ...")
predIdxs = model.predict(testX, batch_size=BS)

predIdxs = np.argmax(predIdxs, axis=1)
```

```
print(classification_report(testY.argmax(axis=1), predIdxs, target_names=label_encoder.classes_))
```

```
<span style="color:#999999;"># calculate a basic confusion matrix</span>
cm = confusion_matrix(testY.argmax(axis=1), predIdxs)
total = sum(sum(cm))
acc = (cm[0, 0] + cm[1, 1] + cm[2, 2]) / total
sensitivity = cm[0, 0] / (cm[0, 0] + cm[0, 1] + cm[0, 2])
specificity = (cm[1, 1] + cm[1, 2] + cm[2, 1] + cm[2, 2]) / (cm[1, 0] + cm[1, 1] + cm[1, 2] + cm[2, 0] + cm[2, 1] + cm[2, 2])
```

```
<span style="color:#999999;"># show the confusion matrix, accuracy, sensitivity, and specificity</span>
print(cm)
print("acc: {:.4f}".format(acc))
print("sensitivity: {:.4f}".format(sensitivity))
print("specificity: {:.4f}".format(specificity))
```

```
<span style="color:#999999;"># plot the training loss and accuracy</span>
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["acc"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_acc"], label="val_acc")
plt.title("Training Loss and Accuracy on COVID-19 Dataset")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig("../Covid19/s-class-plot.png")
```

Evaluating the trained model ...

	precision	recall	f1-score	support
covid	1.00	1.00	1.00	12
normal	1.00	0.93	0.96	14
pneumonia_bac	0.93	1.00	0.97	14
accuracy			0.97	40
macro avg	0.98	0.98	0.98	40
weighted avg	0.98	0.97	0.97	40

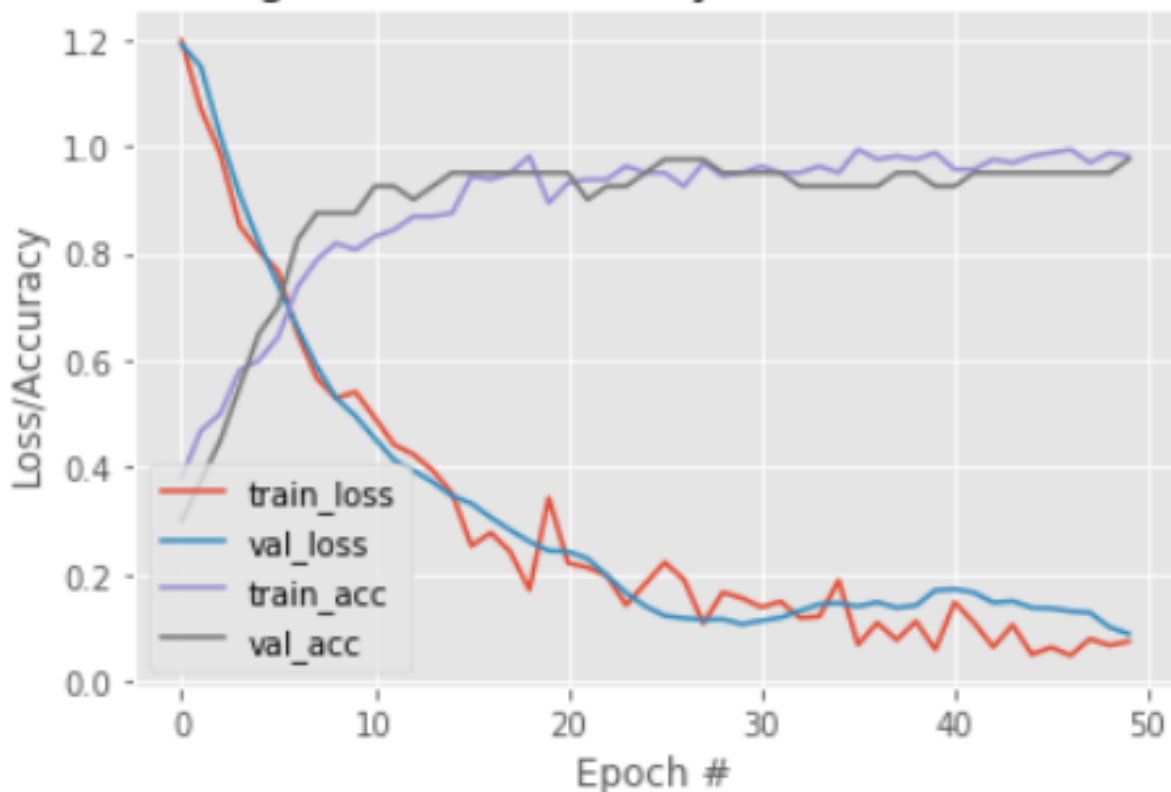
```
[[12  0  0]
 [ 0 13  1]
 [ 0  0 14]]
```

acc: 0.9750

sensitivity: 1.0000

specificity: 1.0000

Training Loss and Accuracy on COVID-19 Dataset



从上图可以看出，得益于“迁移学习”的好处，即使只有很小的数据集和不到 5 分钟的快速训练，得出的结果也并不差：12 个 Covid-19 肺全部正确分类，总共 40 个肺中只有 1 个正常肺被错划为“细菌性肺炎”肺。

5. 为测试真实 X 射线图绘制混淆矩阵

接下来，我们再深入一些，发送真实的 X 射线图测试这个轻度训练的分类器有多有效。

我将上述训练集或验证集中尚未使用的 27 个 X 射线图像上传到模型：

9 个 Covid-19 肺，9 个正常肺，9 个细菌性肺。（图像也附在本篇帖子中。）

我只在第 2 步中修改了一行代码，确保它从不同路径加载测试图像：

```
...
imagePathTest = "./Covid_M/all/test"
...
```

使用上面训练好的模型开始预测：

```
predTest = model.predict(dataTest, batch_size=BS)
print(predTest)
predClasses = predTest.argmax(axis=-1)
print(predClasses)
...
```

最后，按照第 5 步重新计算混淆矩阵：

```
testX = dataTest
testY = labelsTest
... ..
```

得出一些真实测试结果：

同样，经过训练的模型似乎能够正确分类所有 Covid-19 肺。对于这么小的数据集来说，这样的结果不算太差。

6. 一些进一步的观察：

我在复活节周末尝试了许多种数据集，注意到从 AI 分类器的角度来看，Covid-19 肺似乎具有一些显著特征，相对更容易与其他正常细菌性或病毒性（流感）肺区分开来。

经过一些快速测试，我还发现，实在很难区分细菌性和病毒性（正常流感）肺。如果有时间的话，我必须用 Ensemble 集群降低它们之间的差异，就像其他 Kaggle 竞争者在这种情况下可能会做的一样。

以上结果是否切实符合临床的真实情况？Covid-19 肺在 X 射线图上真有一些显著特征吗？我并不确定。这要向真正的胸部放射科医生征求意见。就目前而言，我宁愿假设现在的数据集太小，还无法得出最终结论。

下一步：我很想收集更多的真实 X 射线图，用 xgsboot、AutoML 或新的 IRIS IntegratedML 工作台全面观察。还有，我希望我们能够为临床医生和 A&E 分诊医生将 Covid-19 肺按其严重程度进一步分为如 1 级、2 级和 3 级。

不管怎样，[我还是附上了数据集和 Jupyter Notebook](#)。

部署

在这个“医学影像”领域，我们已经触及了一些快速设置的简易起点。实际上，这个 Covid-19 领域是我在过去一年中用周末时间和长假研究的第 3 个领域。其他包括“人工智能辅助骨折检测”系统和“人工智能辅助眼（眼科）视网膜诊断”系统。

上面的模型也许还太过简陋，但是我们可能很快就必须面对一个常见问题：[我们要如何将其部署为一种“AI 服务”？](#)

这涉及技术堆栈和服务生命周期，还涉及实际的“用例” - 我们要解决哪些问题？它可以提供什么样的实际价值？答案有时并不像技术本身那样清晰。

英国 [RCR \(Royal College of Radiologist\)](#) 草案提出了 2 个简单的用例：“放射科医生的 AI 助手”和“A&E 或基层医疗环境中的 AI 分诊”。我个人表示同意，并认为“AI 分诊”目前来说更有价值。幸运的是，得益于云、Docker、AI 以及我们的 HealthShare 的支持，如今的开发者可以运用更多资源应对这类用例。

例如，以下屏幕截图显示了 AWS 托管的企业级“Covid-19 肺 AI 辅助 CT 检测”服务，及其如何直接嵌入 HealthShare Clinical Viewer 进行演示。与 X 射线类似，DICOM 中的 CT 集可以直接上传或发送到这个打开的 PACS Viewer，点击“AI diagnosis”即可在 10 秒内根据训练的模型给出 Covid-19 概率的定量指示，能够全天候用于快速“AI 分诊”用例。X 射线分类等模型可以在同一患者的相关信息中，以相同的方式在现有 PACS 查看器上部署和调用，帮助一线临床医生。

致谢

测试图像来自 [Joseph Paul Cohen](#) 的 [Covid-19 肺部 X 射线集](#)，以及开放的 [Kaggle 胸部 X 射线集](#) 中的一些干净的肺，由 Adrian Yu 在 [GradientCrescent 仓库](#) 中收集。我还重用了 [PylmageSearch](#) 的 [Adrian](#) 的结构，添加了我自己改进的训练，如“测试”部分所列。还要感谢 [MIM](#) 提供 [基于 AWS 云的 Open PACS Viewer](#) 以及 X 射线和 CT 图像的 AI 模块，用于研究测试数据集。

未来计划

如今，AI 已经“侵入”到人类健康和日常生活的方方面面。根据我高度简化的观点，医疗领域的 AI 应用可能主要有以下几个方向：

- **医学影像**：包括胸部、心脏、眼睛和大脑等的 X 射线、CT 或 MRI 等图像。
- **NLP 理解**：对海量文本资产和知识库的挖掘、理解和学习。
- **人口健康**：包括流行病学等趋势预测、分析和建模。
- **个性化 AI**：一组专为个人训练的 AI/ML/DL 模型，作为个人健康助手与用户一起成长和变老？
- **其他 AI**：如 AlphaGo，甚至用于 3D 蛋白结构预测来对抗 Covid-19 的 AlphaFold 等，这些前沿突破让人印象深刻。

谁也无法预测我们将来能取得什么样的成果。而且，这本来就是一个愿望清单，只要我们别在家里待得太久。

附录 - [文件已上传到此处](#)。其中包括上文使用的图像和提及的 Jupyter Notebook 文件。

在周末从头开始设置和运行可能要花费几个小时的时间。

[#AI](#) [#机器学习](#) [#HealthShare](#)

源

URL:

<https://cn.community.intersystems.com/post/covid-19-%E8%82%BA%E9%83%A8-x-%E5%B0%84%E7%BA%BE%E5%88%86%E7%B1%BB%E5%92%8C-ct-%E6%A3%80%E6%B5%8B%E6%BC%94%E7%A4%BA%EF%BC%88healthshare-%E4%B8%B4%E5%BA%8A%E6%9F%A5%E7%9C%8B%E5%99%A8%E4%B8%8E%E7%AC%A%E4%B8%89%E6%96%B9ai-pacs-viewer%E6%95%B4%E5%90%88%E7%BC%89>