Pneumonia Detection Inception

May 22, 2024

[ ]:

```python
import numpy as np
import tensorflow as tf
from tensorflow import keras
from keras import layers
import tensorflow_addons as tfa
import pandas as pd
import json
import zipfile
import os
import seaborn as sns
import random
import shutil
import time
from PIL import Image
from matplotlib import pyplot as plt
from keras.models import Sequential, Model
from keras.applications import InceptionV3, Xception, InceptionResNetV2
from keras.applications.resnet import preprocess_input
from keras.layers import Dense, Conv2D, Flatten, MaxPooling2D, Dropout,GlobalAveragePooling2D
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint, TensorBoard
import wandb
!mkdir output
!mkdir output/tmp-augmented-images/
random.seed(123)
```

```
/opt/anaconda3/envs/myenv/lib/python3.9/site-
packages/tensorflow_addons/utils/tfa_eol_msg.py:23: UserWarning:

TensorFlow Addons (TFA) has ended development and introduction of new features.
TFA has entered a minimal maintenance and release mode until a planned end of
life in May 2024.
Please modify downstream libraries to take dependencies from other repositories
in our TensorFlow community (e.g. Keras, Keras-CV, and Keras-NLP).

For more information see: https://github.com/tensorflow/addons/issues/2807
```

```
  warnings.warn(
mkdir: output: File exists
mkdir: output/tmp-augmented-images/: File exists
```

[ ]:

```
def resample_data(move_from, move_to, cl, images_to_move=100):
  path = "./" + 'DATASET/data/pneumonia_data'

  classes = os.listdir(path + move_from)

  cl += '/'
  curr_path = path + move_from + cl
  for _, _, files in os.walk(curr_path):
    random.shuffle(files)
    files_to_move = files[:images_to_move]
    for fn in files_to_move:
      shutil.move(curr_path + fn, path + move_to + cl + fn)
      #print('Moved ' + curr_path + fn)

  print('Resampled Images')


move_from, move_to = 'train/', 'test/'
#resample_data(move_from, move_to, 'NORMAL', 200)
# Training images
print('Number of COVID training images:')
!ls DATASET/data/pneumonia_data/train/COVID_19/ | wc -l
print('Number of NORMAL training images:')
!ls DATASET/data/pneumonia_data/train/Normal/ | wc -l
print('Number of PNEUMONIA training images:')
!ls DATASET/data/pneumonia_data/train/Pneumonia// | wc -l
print()

# Validation images
print('Number of COVID training images:')
!ls DATASET/data/pneumonia_data/val/COVID_19/ | wc -l
print('Number of NORMAL validation images:')
!ls DATASET/data/pneumonia_data/val/Normal/ | wc -l
print('Number of PNEUMONIA validation images:')
!ls DATASET/data/pneumonia_data/val/Pneumonia/ | wc -l
print()


# Test images
#resample_data('test/', 'val/', 'PNEUMONIA', 2690)
print('Number of COVID training images:')
!ls DATASET/data/pneumonia_data/test/COVID_19/ | wc -l
print('Number of NORMAL test images:')
```

```
!ls DATASET/data/pneumonia_data/test/Normal/ | wc -l
print('Number of PNEUMONIA test images:')
!ls DATASET/data/pneumonia_data/test/Pneumonia/ | wc -l
```

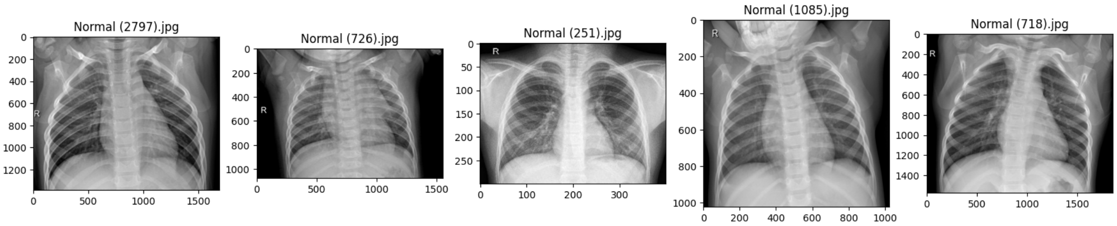
```
Number of COVID training images:
    1100
Number of NORMAL training images:
    3025
Number of PNEUMONIA training images:
    3872

Number of COVID training images:
     171
Number of NORMAL validation images:
     235
Number of PNEUMONIA validation images:
     765

Number of COVID training images:
      10
Number of NORMAL test images:
      10
Number of PNEUMONIA test images:
      20
```

[ ]:

```
def viewImagesFromDir(path, num=5):
  #Display num random images from dataset. Rerun cell for new random images.␣
 ↪The images are only single-channel

  img_paths_visualise = sorted(
        os.path.join(path, fname)
        for fname in os.listdir(path)
        if fname.endswith(".jpg")
  )

  random.shuffle(img_paths_visualise)

  fig, ax = plt.subplots(1, num, figsize=(20, 10))
  print(num)
  for i in range(num):
    ax[i].imshow(Image.open(img_paths_visualise[i]))
    index = img_paths_visualise[i].rfind('/') + 1
    ax[i].title.set_text(img_paths_visualise[i][index:])

  fig.canvas.draw()
  time.sleep(1)
```

```
viewImagesFromDir('DATASET/data/pneumonia_data/train/Normal/', num=5)
```

```
5
```



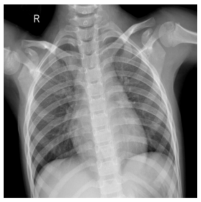
[ ]:

```
base_dir = 'DATASET/data/pneumonia_data'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'val')

# Directory with our training covid 19 pictures
train_covid_dir = os.path.join(train_dir, 'COVID_19')

# Directory with our training normal pictures
train_normal_dir = os.path.join(train_dir, 'NORMAL')

# Directory with our training pneumonia pictures
train_pneumonia_dir = os.path.join(train_dir, 'PNEUMONIA')

# Directory with our validation covid 19 pictures
validation_covid_dir = os.path.join(validation_dir, 'COVID_19')

# Directory with our validation normal pictures
validation_normal_dir = os.path.join(validation_dir, 'NORMAL')

# Directory with our validation pneumonia pictures
validation_pneumonia_dir = os.path.join(validation_dir, 'PNEUMONIA')
```

[ ]:

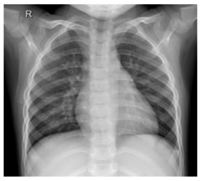
```
# Set up matplotlib fig, and size it to fit 4x4 pics
import matplotlib.image as mpimg
nrows = 6
ncols = 4

fig = plt.gcf()
fig.set_size_inches(ncols*4, nrows*6)
pic_index = 100
train_covid_fnames = os.listdir( train_covid_dir)
train_normal_fnames = os.listdir( train_normal_dir )
train_pneumonia_fnames = os.listdir( train_pneumonia_dir )
```

```
next_covid_pix = [os.path.join(train_covid_dir, fname)
                for fname in train_covid_fnames[ pic_index-8:pic_index]
               ]
next_normal_pix = [os.path.join(train_normal_dir, fname)
                for fname in train_normal_fnames[ pic_index-8:pic_index]
               ]

next_pneumonia_pix = [os.path.join(train_pneumonia_dir, fname)
                for fname in train_pneumonia_fnames[ pic_index-8:pic_index]
               ]

for i, img_path in enumerate(next_normal_pix+next_pneumonia_pix+next_covid_pix):
  # Set up subplot; subplot indices start at 1
  sp = plt.subplot(nrows, ncols, i + 1)
  sp.axis('Off') # Don't show axes (or gridlines)

  img = mpimg.imread(img_path)
  plt.imshow(img)

plt.show()
```

R

R

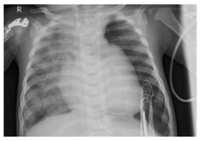
R

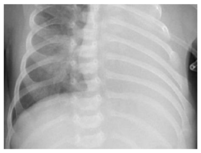
R


R

R

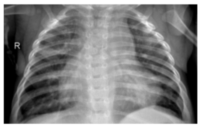
R


R

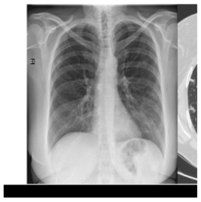


R

R

R

R

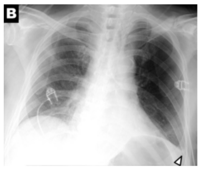
R

C


c

L

DX
SEDUTO


B

[ ]:

```
# Add our data-augmentation parameters to ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255., rotation_range = 40,␣
 ↪width_shift_range = 0.2, height_shift_range = 0.2,shear_range = 0.2,␣
 ↪zoom_range = 0.2, horizontal_flip = True)

test_datagen = ImageDataGenerator( rescale = 1.0/255. )
```

[ ]:

```
train_generator = train_datagen.flow_from_directory(train_dir, batch_size = 32,␣
 ↪class_mode = 'categorical', target_size = (150, 150))
validation_generator = test_datagen.flow_from_directory(validation_dir,␣
 ↪batch_size = 32, class_mode = 'categorical', target_size = (150, 150))
```

```
Found 7997 images belonging to 3 classes.
Found 1171 images belonging to 3 classes.
```

[ ]:

```
from keras.applications.inception_v3 import InceptionV3
base_model = InceptionV3(input_shape = (150, 150, 3), include_top = False,␣
 ↪weights = 'imagenet')
```

[ ]:

```
for layer in base_model.layers:
    layer.trainable = False
```

[ ]:

```
from keras.optimizers import RMSprop

x = layers.Flatten()(base_model.output)
x = layers.Dense(1024, activation='relu')(x)
x = layers.Dropout(0.2)(x)

# Add a final sigmoid layer with 1 node for classification output
x = layers.Dense(3, activation='softmax')(x)

model = tf.keras.models.Model(base_model.input, x)

# Use the legacy Keras optimizer
optimizer_legacy = tf.keras.optimizers.legacy.RMSprop(learning_rate=0.0001)

model.compile(optimizer = optimizer_legacy, loss = 'categorical_crossentropy',␣
 ↪metrics = ['acc'])
```

[ ]:

```
import scipy
print(scipy.__version__)
from keras.callbacks import LearningRateScheduler
def lr_scheduler(epoch, lr):
    # Decay the learning rate by a factor of 0.1 every 10 epochs
    if epoch % 10 == 0 and epoch != 0:
        lr = lr * 0.1
    return lr

# Define a learning rate scheduler callback
#lr_scheduler_callback = LearningRateScheduler(lr_scheduler)

# Pass lr_scheduler_callback to the callbacks parameter of the fit method
history = model.fit(train_generator,
                    validation_data=validation_generator,
                    epochs=20)
```

```
1.13.0
Epoch 1/20
250/250 [==============================] - 75s 293ms/step - loss: 1.4277 - acc:
0.7770 - val_loss: 0.5299 - val_acc: 0.8087
Epoch 2/20
250/250 [==============================] - 73s 290ms/step - loss: 0.4409 - acc:
0.8418 - val_loss: 0.2771 - val_acc: 0.8915
Epoch 3/20
250/250 [==============================] - 72s 289ms/step - loss: 0.3788 - acc:
0.8698 - val_loss: 0.6650 - val_acc: 0.7754
Epoch 4/20
250/250 [==============================] - 73s 290ms/step - loss: 0.3654 - acc:
0.8752 - val_loss: 0.2360 - val_acc: 0.9069
Epoch 5/20
250/250 [==============================] - 72s 289ms/step - loss: 0.3412 - acc:
0.8895 - val_loss: 0.2275 - val_acc: 0.9163
Epoch 6/20
250/250 [==============================] - 73s 290ms/step - loss: 0.3445 - acc:
0.8842 - val_loss: 0.3488 - val_acc: 0.8693
Epoch 7/20
250/250 [==============================] - 72s 287ms/step - loss: 0.3292 - acc:
0.8921 - val_loss: 0.1807 - val_acc: 0.9334
Epoch 8/20
250/250 [==============================] - 72s 288ms/step - loss: 0.3014 - acc:
0.9035 - val_loss: 0.7495 - val_acc: 0.8070
Epoch 9/20
250/250 [==============================] - 72s 289ms/step - loss: 0.3194 - acc:
0.8975 - val_loss: 0.2734 - val_acc: 0.9009
Epoch 10/20
250/250 [==============================] - 72s 287ms/step - loss: 0.2934 - acc:
0.9021 - val_loss: 0.2458 - val_acc: 0.9129
Epoch 11/20
250/250 [==============================] - 72s 287ms/step - loss: 0.2965 - acc:
0.9055 - val_loss: 0.2825 - val_acc: 0.9009
Epoch 12/20
250/250 [==============================] - 72s 289ms/step - loss: 0.2835 - acc:
0.9095 - val_loss: 0.2207 - val_acc: 0.9266
```

```
Epoch 13/20
250/250 [==============================] - 72s 289ms/step - loss: 0.2803 - acc:
0.9141 - val_loss: 0.3204 - val_acc: 0.8958
Epoch 14/20
250/250 [==============================] - 72s 286ms/step - loss: 0.2949 - acc:
0.9105 - val_loss: 0.2303 - val_acc: 0.9231
Epoch 15/20
250/250 [==============================] - 72s 287ms/step - loss: 0.2758 - acc:
0.9093 - val_loss: 0.2962 - val_acc: 0.9095
Epoch 16/20
250/250 [==============================] - 72s 289ms/step - loss: 0.2733 - acc:
0.9102 - val_loss: 0.1993 - val_acc: 0.9308
Epoch 17/20
250/250 [==============================] - 73s 290ms/step - loss: 0.2578 - acc:
0.9137 - val_loss: 0.4504 - val_acc: 0.8668
Epoch 18/20
250/250 [==============================] - 74s 296ms/step - loss: 0.2656 - acc:
0.9175 - val_loss: 0.2109 - val_acc: 0.9377
Epoch 19/20
250/250 [==============================] - 74s 295ms/step - loss: 0.2676 - acc:
0.9158 - val_loss: 0.3218 - val_acc: 0.8992
Epoch 20/20
250/250 [==============================] - 74s 294ms/step - loss: 0.2628 - acc:
0.9175 - val_loss: 0.3142 - val_acc: 0.9095
```

[ ]:

```python
history = history.history
```

[ ]:

```python
history.keys()
```

[ ]:

```
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
```

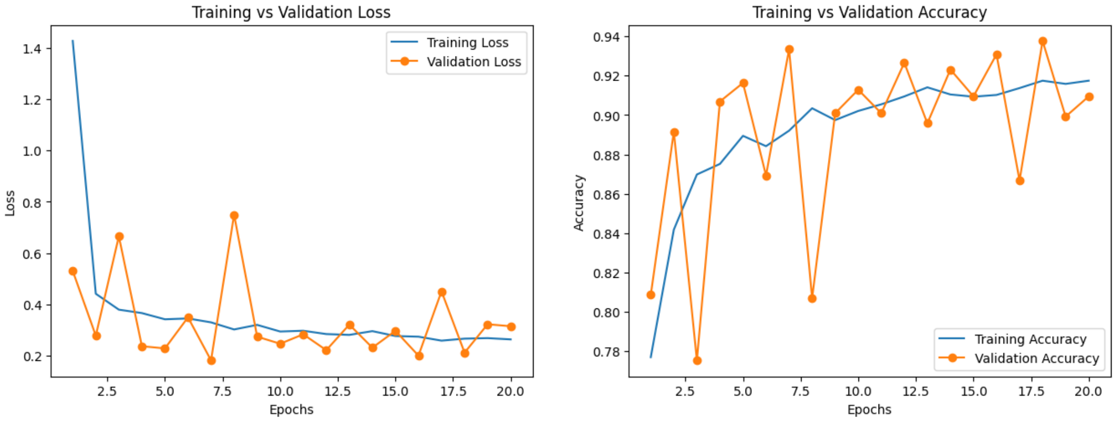
[ ]:

```python
train_loss, val_loss = history['loss'], history['val_loss']
train_acc, val_acc = history['acc'], history['val_acc']
```

[ ]:

```python
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15,5))
epoch_runs = [i+1 for i in range(20)]

ax1.plot(epoch_runs, train_loss, label='Training Loss')
ax1.plot(epoch_runs, val_loss, label='Validation Loss', marker='o')
ax1.set(title='Training vs Validation Loss', xlabel='Epochs',ylabel='Loss')
ax1.legend()

ax2.plot(epoch_runs, train_acc, label='Training Accuracy')
ax2.plot(epoch_runs, val_acc, label='Validation Accuracy', marker='o')
ax2.set(title='Training vs Validation Accuracy',␣
 ↪xlabel='Epochs',ylabel='Accuracy')
ax2.legend()
```

```
plt.show()
```



[ ]:

```
test_dir = 'DATASET/data/pneumonia_data/test'
# Assuming you have a separate test dataset stored in the variable test_dir
test_generator = test_datagen.flow_from_directory(test_dir, batch_size=32,␣
 ↪class_mode='categorical', target_size=(150,150))

# Evaluate the model on the test dataset
test_loss, test_acc = model.evaluate(test_generator)

print("Test Loss:", test_loss)
print("Test Accuracy:", test_acc)
```

```
Found 40 images belonging to 3 classes.
2/2 [==============================] - 0s 53ms/step - loss: 0.3402 - acc: 0.9250
Test Loss: 0.34017261862754822
Test Accuracy: 0.925000011920929
```