Deep Learning

X-ray Image Binary Classification

U Kang
Seoul National University

In This Lecture

- Chest X-ray Image
- Binary classification from x-ray image
- Implement a deep learning model using tensorflow

Outline

- **Introduction**
- Data
- Preprocessing Codes

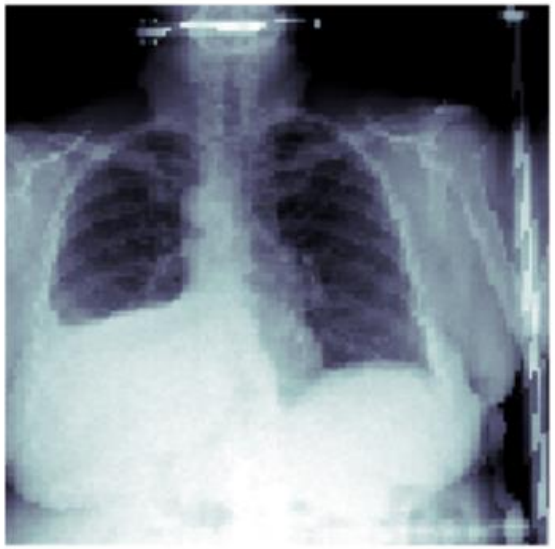
Motivation

- Detecting chest disease could have a significant impact on world health
  - In this practice, we detect a chest disease from a chest x-ray image

Goals

- Classify the chest x-ray image data into one of two categories
  - One category is normal (= No Finding), and the other is abnormal (=Effusion)

Effusion



Problem Definition

- **Given:** chest x-ray image data

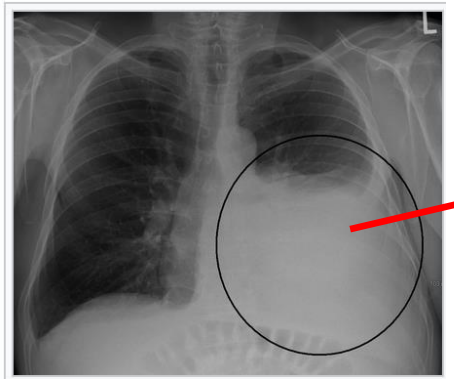
- **Classify:** the data into the correct categories
  - No Finding
  - Effusion

Category description (1)

- **No Finding**
  - Do not find any disease in a x-ray image

Category description (2)

- **(Pleural) effusion**
  - Excess fluid that accumulates in the pleural cavity, the fluid-filled space that surrounds the lungs
  - A pleural effusion appears as an area of whiteness on a standard posteroanterior chest X-ray
  - Reference (wiki) - https://en.wikipedia.org/wiki/Pleural_effusion



A large left sided pleural effusion as seen on an upright chest X-ray

Outline

- ☑ Introduction
- ☐ **Data**
- ☐ Preprocessing Codes

Training Dataset

- 4,000 chest x-ray images
- Image size is (256, 256) or (64, 64)
- There are two categories
  - Normal (class 0)
  - Effusion (class 1)
- NOTE: the dataset is imbalanced
  - 800 effusion out of 4,000 images (20%)

Providing training data

- CSV file (training_labels.csv)
  - There is label information for each x-ray image data
  - The first column is the id of each x-ray image data
  - The second column is the label of each x-ray image data

Image Index	Finding Labels
00001650_011.png	Effusion
00010007_064.png	No Finding

- PNG file in training directory
  - X-ray image data

Outline

- ☑ Introduction
- ☑ Data
- ➡ ☐ **Preprocessing Codes**

Import libraries

- Import the libraries such as tensorflow, numpy, and so on
- For using cv2 package, you should install opencv package in your environment
  - pip install opencv-python

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
from glob import glob
%matplotlib inline
import matplotlib.pyplot as plt
from itertools import chain
import tensorflow as tf
import cv2
```

Loading the Dataset (1)

- Using the pandas library, we prepare the input data
- The variable *all_xray_df* is a DataFrame instance
- It is a 2D table of instances and features

```
all_xray_df = pd.read_csv('./xray-chest/training_labels.csv')
all_image_paths = {os.path.basename(x): x for x in
                   glob(os.path.join( './xray-chest/training', '*.png'))}
print('Scans found:', len(all_image_paths), ', Total Headers', all_xray_df.shape[0])
all_xray_df['path'] = all_xray_df['Image Index'].map(all_image_paths.get)
all_xray_df.sample(3)
```

Loading the Dataset (2)

- You can check the contents:
  - all_xray_df
  - all_xray_df.sample(3)

	Image Index	Finding Labels	path
4463	00011367_000.png	No Finding	./training/00011367_000.png
1794	00010352_001.png	No Finding	./training/00010352_001.png
666	00009659_007.png	Effusion	./training/00009659_007.png

Loading the Dataset (3)

- Labeling the dataset by the class:
- No Finding (=Normal) is 0 otherwise 1

```
all_xray_df['Label'] = all_xray_df['Finding Labels'].map(lambda x: 0 if x=='No Finding' else 1.0)
all_xray_df.sample(10)
```

	Image Index	Finding Labels	path	Label
1194	00010071_001.png	No Finding	./training/00010071_001.png	0.0
958	00010007_001.png	No Finding	./training/00010007_001.png	0.0
1501	00010227_000.png	No Finding	./training/00010227_000.png	0.0
267	00003974_002.png	Effusion	./training/00003974_002.png	1.0
4345	00011311_001.png	No Finding	./training/00011311_001.png	0.0
728	00009925_002.png	No Finding	./training/00009925_002.png	0.0

Loading the Dataset (4)

- We define the function to load the image data

```
def load_data(data_df):
    train_filesname = data_df['path'].values
    train_labels = np.stack(data_df['Label'].map(lambda x: [0] if x==0 else [1]))
    x_train = []
    for filename in train_filesname:
        img = cv2.imread(filename)
        x_train.append(img)
    return np.asarray(x_train,dtype=np.float16), np.asarray(train_labels,dtype=np.float32)
```

Test measure (1)

- **F_1 score**
  - The **F_1 score** (also **F-score** or **F-measure**) is a measure of a test's accuracy in analysis of binary classification
  - It considers both the precision *p* and the recall *r* of the test to compute the score

$$F_1 = \frac{2}{\frac{1}{p} + \frac{1}{r}} = 2 \times \frac{p \times r}{(p + r)}$$

$$p = \frac{true\ positive}{true\ positive + false\ positive}$$

$$r = \frac{true\ positive}{true\ positive + false\ negative}$$

Test measure (2)

- **F_1 score**
  - Consider Effusion as positive
  - True-positive
    - # of predicted Effusion over # of actual Effusion
  - False-negative
    - # of predicted Normal over # of actual Effusion
  - False-positive
    - # of predicted Effusion over # of actual Normal
  - True-negative
    - # of predicted Normal over # of actual Normal

Reference for measure

- $\mathbf{F_1}$ score
  - Wikipedia - https://en.wikipedia.org/wiki/F1_score
  - Library – scikit-learn in python

Questions?