💻 You Only Look Once (YOLO) Training and Cross Validation

Group: 4NN

Dharani Palanisamy (z5260276)

Faiyam Islam (z5258151)

Pooja Saianand (z5312416)

Priya Nandyal (z5312288)

The notebook computes loss function values, precision, recall, and mean average precision curves using cross validation of the dataset. We will also analyse the evaluation metrics such as mAP and IoU and instigate a conclusion on the accuracy of this object detection method.

We are running this notebook on GPU.

We've used the following sources which assisted our approach for YOLO:

https://www.kaggle.com/code/kimse0ha/vinbigdata-eda-infer-analysis-with-yolov5

https://www.kaggle.com/code/awsaf49/vinbigdata-cxr-ad-yolov5-14-class-infer/notebook

https://www.kaggle.com/code/mrutyunjaybiswal/vbd-chest-x-ray-abnormalities-detection-eda/notebook

https://www.kaggle.com/code/jamsilkaggle/quick-data-analysis-with-yolov5-at-a-glance

Importing datasets and installing packages

In [ ]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

In [ ]:
```
!pip install --upgrade seaborn
```

In [ ]:
```
import numpy as np, pandas as pd
from glob import glob
import shutil, os
import matplotlib.pyplot as plt
from sklearn.model_selection import GroupKFold
from tqdm.notebook import tqdm
import seaborn as sns
```

Analyzing the train dataset and obtaining class information

In [ ]:
```
dim = 512
fold = 4
train_df = pd.read_csv(f'../input/vinbigdata-{dim}-image-dataset/vinbigdata/train.csv')
train_df.head()
```

In [ ]:
```
train_df['image_path'] = f'/kaggle/input/vinbigdata-{dim}-image-dataset/vinbigdata/train/'+ train_df.image_id+('.png' if dim!='original' else '.jpg')
train_df.head()
```

In [ ]:
```
# Getting the number of classes
train_df = train_df[train_df.class_id!=14].reset_index(drop = True)
```

In [ ]:
```
# Pre-processing the dataset by computing the values of x_min,x_max,x_mid,y_min,y_max,y_mid,height and width

train_df['x_min'] = train_df.apply(lambda row: (row.x_min)/row.width, axis =1)
train_df['x_max'] = train_df.apply(lambda row: (row.x_max)/row.width, axis =1)
train_df['x_mid'] = train_df.apply(lambda row: (row.x_max+row.x_min)/2, axis =1)

train_df['y_min'] = train_df.apply(lambda row: (row.y_min)/row.height, axis =1)
train_df['y_max'] = train_df.apply(lambda row: (row.y_max)/row.height, axis =1)
train_df['y_mid'] = train_df.apply(lambda row: (row.y_max+row.y_min)/2, axis =1)

train_df['h'] = train_df.apply(lambda row: (row.y_max-row.y_min), axis =1) #height
train_df['w'] = train_df.apply(lambda row: (row.x_max-row.x_min), axis =1) #width

train_df['area'] = train_df['w']*train_df['h']
train_df.head()
```

In [ ]:
```
# The number of rows in the train_df for the feature columns
features = ['x_min', 'y_min', 'x_max', 'y_max', 'x_mid', 'y_mid', 'w', 'h', 'area']
y = train_df['class_id']
X = train_df[features]
X.shape, y.shape
```

In [ ]:
```
# To get a list of the class names
class_ids, class_names = list(zip(*set(zip(train_df.class_id, train_df.class_name))))
class_list = list(np.array(class_names)[np.argsort(class_ids)])
class_list = list(map(lambda x: str(x), class_list))
class_list
```

Splitting the dataset

In [ ]:
```
# Split the dataset and drop all the x-rays that do not contain any abnormality
fold = 4
gkf  = GroupKFold(n_splits = 5)
train_df['fold'] = -1
for fold, (train_idx, val_idx) in enumerate(gkf.split(train_df, groups = train_df.image_id.tolist())):
    train_df.loc[val_idx, 'fold'] = fold
train_df.head()
```

In [ ]:
```
val_files   = []
train_files = []
val_files += list(train_df[train_df.fold==fold].image_path.unique())
train_files += list(train_df[train_df.fold!=fold].image_path.unique())
print(len(train_files)) #size of train dataset
print(len(val_files)) # size of validation set
```

Creating directories and copying files

In [ ]:
```
os.makedirs('/kaggle/working/vinbigdata/labels/train', exist_ok = True)
os.makedirs('/kaggle/working/vinbigdata/labels/val', exist_ok = True)
os.makedirs('/kaggle/working/vinbigdata/images/train', exist_ok = True)
os.makedirs('/kaggle/working/vinbigdata/images/val', exist_ok = True)
label_dir = '/kaggle/input/vinbigdata-yolo-labels-dataset/labels'
for file in tqdm(train_files): # we use tqdm to see the progress of the copying of files
    shutil.copy(file, '/kaggle/working/vinbigdata/images/train')
    filename = file.split('/')[-1].split('.')[0]
    shutil.copy(os.path.join(label_dir, filename+'.txt'), '/kaggle/working/vinbigdata/labels/train')

for file in tqdm(val_files):
    shutil.copy(file, '/kaggle/working/vinbigdata/images/val')
    filename = file.split('/')[-1].split('.')[0]
    shutil.copy(os.path.join(label_dir, filename+'.txt'), '/kaggle/working/vinbigdata/labels/val')
```

Setting up YOLOv5

In [ ]:
```
# Creating a yaml file
from os import listdir
from os.path import isfile, join
import yaml

cwd = '/kaggle/working/'

with open(join( cwd , 'train.txt'), 'w') as f:
    for path in glob('/kaggle/working/vinbigdata/images/train/*'):
        f.write(path+'\n')

with open(join( cwd , 'val.txt'), 'w') as f:
    for path in glob('/kaggle/working/vinbigdata/images/val/*'):
        f.write(path+'\n')

data = dict(
    train =  join( cwd , 'train.txt') ,
    val   =  join( cwd , 'val.txt' ),
    nc    = 14,
    names = class_list
    )

with open(join( cwd , 'vinbigdata.yaml'), 'w') as outfile:
    yaml.dump(data, outfile, default_flow_style=False)

f = open(join( cwd , 'vinbigdata.yaml'), 'r')
print('\nyaml contents:')
print(f.read())
```

In [ ]:
```
import torch
from IPython.display import Image, clear_output  # to display images and clear outputs
```

In [ ]:
```
# rmtree removes any yolov5 directory existing in the kaggle directory before creating a new one
# Running this function would result in an error if there is not yolov5 directory on the kaggle working directory
shutil.rmtree('/kaggle/working/yolov5')
```

In [ ]:
```
# Setting up yolov5
shutil.copytree('/kaggle/input/yolov5-official-v31-dataset/yolov5', '/kaggle/working/yolov5')
os.chdir('/kaggle/working/yolov5')

clear_output()
print('Setup complete. Using torch %s %s' % (torch.__version__, torch.cuda.get_device_properties(0) if torch.cuda.is_available() else 'CPU'))
```

In [ ]:
```
pwd
```

Training the dataset and cross validation

In [ ]:
```
# training and cross validating the dataset using yolov5
!WANDB_MODE="dryrun" python train.py --img 640 --batch 16 --epochs 30 --data /kaggle/working/vinbigdata.yaml --weights yolov5x.pt --cache
```

Confusion matrix

In [ ]:
```
# A confusion_matrix is produced when training the datasets
plt.figure(figsize=(30,15))
plt.axis('off')
plt.imshow(plt.imread('runs/train/exp/confusion_matrix.png'));
```

Plots for our model

In [ ]:
```
# Training the data computes loss, precision, recall and mAP values for different thresholds
plt.figure(figsize=(30,15))
plt.axis('off')
plt.imshow(plt.imread('runs/train/exp/results.png'));
```