About this notebook

- PyTorch resnext50_32x4d starter code
- GroupKFold 4 folds
- training code is here

If this notebook is helpful, feel free to upvote :)

Directory settings

In [1]:
```python
# ====================================================
# Directory settings
# ====================================================
import os

MODEL_DIR = '../input/ranzcr-resnext50-starter-models/'
OUTPUT_DIR = './'
if not os.path.exists(OUTPUT_DIR):
    os.makedirs(OUTPUT_DIR)

TEST_PATH = '../input/ranzcr-clip-catheter-line-classification/test'
```

CFG

In [2]:
```python
# ====================================================
# CFG
# ====================================================
class CFG:
    debug=False
    num_workers=4
    model_name='resnext50_32x4d'
    size=600
    batch_size=64
    seed=42
    target_size=11
    target_cols=['ETT - Abnormal', 'ETT - Borderline', 'ETT - Normal',
                 'NGT - Abnormal', 'NGT - Borderline', 'NGT - Incompletely Imaged', 'NGT - Normal',
                 'CVC - Abnormal', 'CVC - Borderline', 'CVC - Normal',
                 'Swan Ganz Catheter Present']
    n_fold=4
    trn_fold=[0, 1, 2, 3]
```

Library

In [3]:
```python
# ====================================================
# Library
# ====================================================
import sys
sys.path.append('../input/pytorch-image-models/pytorch-image-models-master')

import os
import math
import time
import random
import shutil
from pathlib import Path
from contextlib import contextmanager
from collections import defaultdict, Counter

import scipy as sp
import numpy as np
import pandas as pd

from sklearn import preprocessing
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import StratifiedKFold, GroupKFold, KFold

from tqdm.auto import tqdm
from functools import partial

import cv2
from PIL import Image

from matplotlib import pyplot as plt

import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.optim import Adam, SGD
import torchvision.models as models
from torch.nn.parameter import Parameter
from torch.utils.data import DataLoader, Dataset
from torch.optim.lr_scheduler import CosineAnnealingWarmRestarts, CosineAnnealingLR, ReduceLROnPlateau

from albumentations import (
    Compose, OneOf, Normalize, Resize, RandomResizedCrop, RandomCrop, HorizontalFlip, VerticalFlip,
    RandomBrightness, RandomContrast, RandomBrightnessContrast, Rotate, ShiftScaleRotate, Cutout,
    IAAAdditiveGaussianNoise, Transpose
    )
from albumentations.pytorch import ToTensorV2
from albumentations import ImageOnlyTransform

import timm

from torch.cuda.amp import autocast, GradScaler

import warnings
warnings.filterwarnings('ignore')

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

Utils

In [4]:
```python
# ====================================================
# Utils
# ====================================================
def get_score(y_true, y_pred):
    scores = []
    for i in range(y_true.shape[1]):
        score = roc_auc_score(y_true[:,i], y_pred[:,i])
        scores.append(score)
    avg_score = np.mean(scores)
    return avg_score, scores


def get_result(result_df):
    preds = result_df[[f'pred_{c}' for c in CFG.target_cols]].values
    labels = result_df[CFG.target_cols].values
    score, scores = get_score(labels, preds)
    LOGGER.info(f'Score: {score:<.4f}  Scores: {np.round(scores, decimals=4)}')


@contextmanager
def timer(name):
    t0 = time.time()
    LOGGER.info(f'[{name}] start')
    yield
    LOGGER.info(f'[{name}] done in {time.time() - t0:.0f} s.')


def init_logger(log_file=OUTPUT_DIR+'inference.log'):
    from logging import getLogger, INFO, FileHandler,  Formatter,  StreamHandler
    logger = getLogger(__name__)
    logger.setLevel(INFO)
    handler1 = StreamHandler()
    handler1.setFormatter(Formatter("%(message)s"))
    handler2 = FileHandler(filename=log_file)
    handler2.setFormatter(Formatter("%(message)s"))
    logger.addHandler(handler1)
    logger.addHandler(handler2)
    return logger

LOGGER = init_logger()


def seed_torch(seed=42):
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed(seed)
    torch.backends.cudnn.deterministic = True

seed_torch(seed=CFG.seed)
```

Data Loading

In [5]:
```python
train = pd.read_csv('../input/ranzcr-clip-catheter-line-classification/train.csv')
folds = train.copy()
Fold = GroupKFold(n_splits=CFG.n_fold)
groups = folds['PatientID'].values
for n, (train_index, val_index) in enumerate(Fold.split(folds, folds[CFG.target_cols], groups)):
    folds.loc[val_index, 'fold'] = int(n)
folds['fold'] = folds['fold'].astype(int)
display(folds.groupby('fold').size())
```

```
fold
0    7521
1    7521
2    7521
3    7520
dtype: int64
```

In [6]:
```python
oof_df = pd.DataFrame()
for fold in CFG.trn_fold:
    valid_folds = folds[folds['fold']==fold].reset_index(drop=True)
    check_point = torch.load(MODEL_DIR+f'{CFG.model_name}_fold{fold}_best.pth', map_location=device)
    for c in [f'pred_{c}' for c in CFG.target_cols]:
        valid_folds[c] = np.nan
    valid_folds[[f'pred_{c}' for c in CFG.target_cols]] = check_point['preds']
    LOGGER.info(f"========== fold: {fold} result ==========")
    get_result(valid_folds)
    oof_df = pd.concat([oof_df, valid_folds])
oof_df = oof_df.reset_index(drop=True)
LOGGER.info(f"========== CV ==========")
get_result(oof_df)
```

```
========== fold: 0 result ==========
Score: 0.9358  Scores: [0.9385 0.9434 0.9893 0.9466 0.9448 0.9786 0.98   0.8888 0.8125 0.8764
 0.9946]
========== fold: 1 result ==========
Score: 0.9356  Scores: [0.9311 0.9604 0.9892 0.931  0.9323 0.9723 0.9827 0.8872 0.8166 0.8897
 0.9996]
========== fold: 2 result ==========
Score: 0.9390  Scores: [0.9777 0.95   0.9879 0.9383 0.9229 0.978  0.9822 0.886  0.8233 0.8828
 0.9998]
========== fold: 3 result ==========
Score: 0.9351  Scores: [0.9697 0.9432 0.9904 0.9286 0.9303 0.9803 0.9812 0.8768 0.8045 0.8835
 0.9973]
========== CV ==========
Score: 0.9353  Scores: [0.9468 0.9496 0.989  0.9367 0.9313 0.9774 0.981  0.8842 0.8136 0.882
 0.9972]
```

In [7]:
```python
test = pd.read_csv('../input/ranzcr-clip-catheter-line-classification/sample_submission.csv')
print(test.shape)
test.head()
```

```
(3582, 12)
```

Out[7]:

	StudyInstanceUID	ETT - Abnormal	ETT - Borderline	ETT - Normal	NGT - Abnormal	NGT - Borderline	NGT - Incompletely Imaged	NGT - Normal	CVC - Abnormal
0	1.2.826.0.1.3680043.8.498.46923145579096002617...	0	0	0	0	0	0	0	0
1	1.2.826.0.1.3680043.8.498.84006870182611080091...	0	0	0	0	0	0	0	0
2	1.2.826.0.1.3680043.8.498.12219033294413119947...	0	0	0	0	0	0	0	0
3	1.2.826.0.1.3680043.8.498.84994474380235968109...	0	0	0	0	0	0	0	0
4	1.2.826.0.1.3680043.8.498.35798987793805669662...	0	0	0	0	0	0	0	0

In [8]:
```python
if CFG.debug:
    test = test.head()
```

Dataset

In [9]:
```python
# ====================================================
# Dataset
# ====================================================
class TestDataset(Dataset):
    def __init__(self, df, transform=None):
        self.df = df
        self.file_names = df['StudyInstanceUID'].values
        self.transform = transform

    def __len__(self):
        return len(self.df)

    def __getitem__(self, idx):
        file_name = self.file_names[idx]
        file_path = f'{TEST_PATH}/{file_name}.jpg'
        image = cv2.imread(file_path)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        if self.transform:
            augmented = self.transform(image=image)
            image = augmented['image']
        return image
```

Transforms

In [10]:
```python
# ====================================================
# Transforms
# ====================================================
def get_transforms(*, data):

    if data == 'train':
        return Compose([
            Resize(CFG.size, CFG.size),
            Normalize(
                mean=[0.485, 0.456, 0.406],
                std=[0.229, 0.224, 0.225],
            ),
            ToTensorV2(),
        ])

    elif data == 'valid':
        return Compose([
            Resize(CFG.size, CFG.size),
            Normalize(
                mean=[0.485, 0.456, 0.406],
                std=[0.229, 0.224, 0.225],
            ),
            ToTensorV2(),
        ])
```

MODEL

In [11]:
```python
# ====================================================
# MODEL
# ====================================================
class CustomResNext(nn.Module):
    def __init__(self, model_name='resnext50_32x4d', pretrained=False):
        super().__init__()
        self.model = timm.create_model(model_name, pretrained=pretrained)
        n_features = self.model.fc.in_features
        self.model.fc = nn.Linear(n_features, CFG.target_size)

    def forward(self, x):
        x = self.model(x)
        return x
```

Helper functions

In [12]:
```python
# ====================================================
# Helper functions
# ====================================================
def inference(model, states, test_loader, device):
    model.to(device)
    tk0 = tqdm(enumerate(test_loader), total=len(test_loader))
    probs = []
    for i, (images) in tk0:
        images = images.to(device)
        avg_preds = []
        for state in states:
            model.load_state_dict(state['model'])
            model.eval()
            with torch.no_grad():
                y_preds = model(images)
            avg_preds.append(y_preds.sigmoid().to('cpu').numpy())
        avg_preds = np.mean(avg_preds, axis=0)
        probs.append(avg_preds)
    probs = np.concatenate(probs)
    return probs
```

inference

In [13]:
```python
# ====================================================
# inference
# ====================================================
model = CustomResNext(CFG.model_name, pretrained=False)
states = [torch.load(MODEL_DIR+f'{CFG.model_name}_fold{fold}_best.pth') for fold in CFG.trn_fold]
test_dataset = TestDataset(test, transform=get_transforms(data='valid'))
test_loader = DataLoader(test_dataset, batch_size=CFG.batch_size, shuffle=False,
                         num_workers=CFG.num_workers, pin_memory=True)
predictions = inference(model, states, test_loader, device)
# submission
test[CFG.target_cols] = predictions
test[['StudyInstanceUID'] + CFG.target_cols].to_csv(OUTPUT_DIR+'submission.csv', index=False)
test.head()
```

100% 56/56 [03:58<00:00, 4.26s/it]

Out[13]:

	StudyInstanceUID	ETT - Abnormal	ETT - Borderline	ETT - Normal	NGT - Abnormal	NGT - Borderline	NGT - Incompletely Imaged	NGT - Normal	CV(Abnorn
0	1.2.826.0.1.3680043.8.498.46923145579096002617...	0.022955	0.320129	0.446900	0.003614	0.008296	0.012233	0.969640	0.0266
1	1.2.826.0.1.3680043.8.498.84006870182611080091...	0.000093	0.000134	0.000266	0.000209	0.000217	0.000174	0.000036	0.0084
2	1.2.826.0.1.3680043.8.498.12219033294413119947...	0.000221	0.000134	0.000230	0.000864	0.000494	0.000177	0.000070	0.0063
3	1.2.826.0.1.3680043.8.498.84994474380235968109...	0.005454	0.023865	0.031240	0.075304	0.026674	0.932984	0.036249	0.1660
4	1.2.826.0.1.3680043.8.498.35798987793805669662...	0.000440	0.000646	0.000951	0.000985	0.003028	0.000067	0.001673	0.0078