


1단계(3차년도) 주요 결과물

(과제명) 대규모 분산 에너지 저장장치 인프라의 안전한 자율운영
및 성능 평가를 위한 지능형 SW 프레임워크 개발
(과제번호) 2021-0-00077

- 결과물명 : 전이학습을 완료한 이상 탐지 알고리즘(SW)
- 작성일자 : 2023년 11월 14일

과학기술정보통신부 SW컴퓨팅산업원천기술개발사업
“1단계(3차년도) 주요 결과물”로 제출합니다.

수행기관	성명/직위	확인
서울대학교	강명주/연구책임자	

정보통신기획평가원장 귀하

<목차>

1. 개요	1
가. 목적	1
나. 범위	1
2. 기법	2
3. 소프트웨어	2
가. 주요 기능	2
나. 사용 환경	4
다. 사용 방법	4
라. 코드	5

1. 개요

가. 목적

- 본 문서는 “대규모 분산 에너지 저장장치 인프라의 안전한 자율운영 및 성능 평가를 위한 지능형 SW 프레임워크 개발” 사업의 주요 결과물에 대한 보고서이다.
- 안전 AI 분석 엔진 “전이학습을 완료한 이상 탐지 알고리즘” 소프트웨어에 대한 설명을 한다.

[안전 AI 분석 엔진 SW]

구분	1차년도	2차년도	3차년도	4차년도	5차년도
안전 AI 분석 엔진	물성 진단 지표 후보(SW)	실험 데이터로 학습한 이상 탐지 알고리즘(SW)	전이학습을 완료한 이상 탐지 알고리즘(SW)	최적화된 이상 탐지, 성능 진단 알고리즘(SW)	이상 탐지 알고리즘 및 사고 대응 알고리즘(SW)
	데이터 전처리 알고리즘(SW)	실험 데이터로 학습한 성능 진단 알고리즘(SW)	전이 학습을 완료한 성능 진단 알고리즘(SW)	개선된 이상 탐지, 성능 진단 알고리즘(SW)	상태 진단 알고리즘 및 효율적 운영 분석 결과(SW)

나. 범위

- 딥러닝 모델의 실제 데이터 적용
- 딥러닝 모델의 전이학습을 통한 활용
- 실제 태양광 BMS 운용 데이터 대상
- 주요 데이터 칼럼 (전압, 전류, SOC, 온도, 전압차) 활용

2. 기법

- 정상 운영 BMS 데이터의 주요 칼럼에 대해 딥러닝 기반의 시계열 이상 탐지 모델인 AnomalyBERT를 학습한다.
- 모델은 일반적인 시계열에서 나타나는 이상 상황을 모사하여 해당 이상 구간을 구별하는 방법을 통하여 학습한다.
- 하나의 실제 태양광 BMS 운용 데이터에서 학습한 모델을 다른 태양광 BMS 운용 데이터에서 미세조정 기법을 활용하여 전이학습을 진행한다.
- 모델의 성능을 이상 탐지 평가를 위한 합성 데이터를 통해 평가한다.

3. 소프트웨어

가. 주요 기능

- 시계열 이상 탐지 모델을 하나의 실제 태양광 데이터를 통해 학습 후 파라미터 저장 (훈련 명령어)

Training

We provide the training code for our model. (recommended) For example, to train a model of 6-layer Transformer body on ESS_sionyu dataset, run:

```
python3 train.py --dataset=ESS_sionyu --patch_size=90 --window_sliding=512
```



To train a model on ESS_panli_bank1 dataset with patch size of 2 and customized outlier synthesis probability, run:

```
python3 train.py --dataset=ESS_panli_bank1 --patch_size=2 --soft_replacing=0.5 --uniform_replacing=0.1 --peak_noising=0.1 \
--length_adjusting=0.1
```



If you want to customize the model and training settings, please check the options in `train.py`.

- 저장한 모델을 다른 실제 태양광 데이터에서 미세조정을 통해 전이 학습 (전이 학습 진행을 위한 저장 모델 불러오기)

```
# Load a checkpoint if exists.
if options.checkpoint != None:
    try:
        model.load_state_dict(torch.load(options.checkpoint, map_location='cpu'))
    except:
        loaded_weight = torch.load(options.checkpoint, map_location='cpu')
        loaded_weight['linear_embedding.weight'] = model.linear_embedding.weight
        loaded_weight['linear_embedding.bias'] = model.linear_embedding.bias
        model.load_state_dict(loaded_weight)
```

(저장한 모델을 활용해 전이 학습 진행)

```
python3 train.py --dataset=ESS_panli_bank1 --gpu_id=3 --patch_size=90 --window_sliding=512 --checkpoint='logs/{log_dir}/state_dict.pt' --lr=0.00001 --max_steps=50000
```

- 모델의 성능 평가

```
def score(config, adjust=True):
    label = np.load(config['label'])
    anomaly_rate = config['anomaly_rate']

    def _score(gt, pr, anomaly_rate, adjust):
        # get anomaly intervals
        gt_aug = np.concatenate([np.zeros(1), gt, np.zeros(1)]).astype(np.int32)
        gt_diff = gt_aug[1:] - gt_aug[:-1]

        begin = np.where(gt_diff == 1)[0]
        end = np.where(gt_diff == -1)[0]

        intervals = np.stack([begin, end], axis=1)

        # quantile cut
        pa = pr.copy()
        q = np.quantile(pa, 1-anomaly_rate)
        pa = (pa > q).astype(np.int32).squeeze(1)

        if adjust:
            for s, e in intervals:
                interval = slice(s, e)
                if pa[interval].sum() > 0:
                    pa[interval] = 1

        # confusion matrix
        TP = (gt * pa).sum()
        TN = ((1 - gt) * (1 - pa)).sum()
        FP = ((1 - gt) * pa).sum()
        FN = (gt * (1 - pa)).sum()

        assert (TP + TN + FP + FN) == len(gt)

        # Compute p, r, f1.
        precision = TP / (TP + FP)
        recall = TP / (TP + FN)
        # f1_score = 2*precision*recall/(precision+recall)
        score = (precision + recall) / 2

        return score

    result = []
    for step_id in trange(0, config['end_step']+1, 500):
        prediction = np.load(f"{config['log']}/state/state_dict_step_{step_id}_results.npy")
        ESS_score = _score(label, prediction, anomaly_rate, adjust)

        result.append(ESS_score)
        if ESS_score > 0.90:
            print('first 90% touch epoch:', step_id)
            break

    return result
```

나. 사용 환경

- 본 프로그램은 python 코드로 구현되었음
- Python이 설치된 어떠한 OS 환경에서도 사용이 가능함
- 코드의 구동에는 python의 OS, numpy, pandas, pytorch 라이브러리가 필요함

다. 사용 방법

- 학습 및 평가 데이터는 parquet 형식을 사용하며, 학습한 모델은 pt 형식으로 저장함
- 전체 코드는 AnomalyBERT의 학습 및 평가 코드를 따라 수행되어 학습 데이터 (training data)와 평가 데이터(test data)를 지정한 후 일련의 과정을 따라 훈련 및 성능 평가를 할 수 있음
- 모델을 훈련 데이터에 대해 학습을 진행한 후 학습 완료한 모델을 저장함
- 학습 완료한 모델을 해당 데이터의 평가 데이터에서 성능을 평가함
- 저장한 모델을 다른 훈련 데이터에 대해 미세조정을 통해 전이학습을 진행함
- 전이 학습을 완료한 모델을 해당 데이터의 평가 데이터에서 성능을 평가함

라. 코드

OpenESS/SNU_AI/AnomalyBERT