


2차년도 주요 결과물

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

- 결과물명 : 실험 데이터로 학습한 이상 탐지 알고리즘 (SW) (딥러닝 기반의 이상 탐지 모델 개발을 위한 합성 데이터 생성(SW))
- 작성일자 : 2022년 12월 07일

과학기술정보통신부 SW컴퓨팅산업원천기술개발사업
“2차년도 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 “실험 데이터로 학습한 이상 탐지 알고리즘 (SW)” 중 “딥러닝 기반의 이상 탐지 모델 개발을 위한 합성 데이터 생성” 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 모델 학습 및 평가를 위한 데이터 합성
- 실제 태양광 BMS 운용 데이터 대상
- 주요 데이터 칼럼 (전압, 전류, SOC, 온도, 전압차) 활용

2. 기법

- 정상 운영 BMS 데이터의 주요 칼럼에 대해 시계열 이상치를 합성하여 딥러닝 기반의 모델 학습을 위해 활용한다.
- 정상 데이터만을 사용하여 이상 데이터를 합성 및 활용하므로 실제 배터리 이상 데이터의 부족 문제를 보완한다.
- 일반적인 시계열 이상치 분류를 참고하여 태양광 BMS 영역에 적합한 이상치 합성 기법을 4가지로 정의하고 (상수값 변환, 외부 구간 대체, 피크값 추가, 가우시안 노이즈 추가) 이를 무작위 방식으로 합성하여 데이터를 생성한다.

3. 소프트웨어

가. 주요 기능

- 데이터 합성 옵션값 설정

```
# Configurations for anomaly synthesis
REPLACING_RATE = (0.0005, 0.1) # min-max rate of the length of replacing interval for anomalies

# probabilities of anomaly types
SOFT_REPLACING = 0.5
UNIFORM_REPLACING = 0.25
PEAK_NOISING = 0.15
WHITE_NOISING = 0.1

# anomaly synthesis options
MAX_EXTERNAL_INTERVAL_RATE = 0.7
MAX_UNIFORM_VALUE_DIFFERENCE = 0.1
MIN_PEAK_ERROR = 0.1
WHITE_NOISE_LEVEL = 0.003

# data value ranges
# V      # I      # SOC      # temp      # gap
VALUE_RANGE = np.array([[660, 820], [-280, 410], [0, 100], [10, 40], [0, 0.2]]).transpose()

# data infos
N_COLUMNS = 5
N_ANOMALY_TYPES = 4
```

- 상수값 대체 함수

```
# Synthesize anomalies. - uniform replacing
syn_data = npy_data.copy()
_anomaly_label = anomaly_label.copy()
interval_len = replacing_lengths[1]
abnormal_column_idx = abnormal_columns[1] < 0.5
if not(abnormal_column_idx.any()):
    abnormal_column_idx[np.random.randint(0, N_COLUMNS)] = True

mean_values = syn_data[target_indices[1]:target_indices[1]+interval_len, abnormal_column_idx].mean(axis=0)
syn_data[target_indices[1]:target_indices[1]+interval_len, abnormal_column_idx]\
    = np.random.uniform(np.maximum(mean_values-MAX_UNIFORM_VALUE_DIFFERENCE, 0),
                        np.minimum(mean_values+MAX_UNIFORM_VALUE_DIFFERENCE, 1))[None, :]

_anomaly_label[target_indices[1]:target_indices[1]+interval_len] = 1
syn_data = np.concatenate((syn_data, additional_data, _anomaly_label), axis=1)

file_name = single_type_data_dir+date+'_1.npy'
meta_data.append([_date+'_1', 'uniform', (target_indices[1], target_indices[1]+interval_len),
                  tuple(np.argwhere(abnormal_column_idx).flatten()), (date in test_clear_date)])
np.save(file_name, syn_data)
```

- 외부 구간 대체 함수

```
# Synthesize anomalies. - soft replacing
syn_data = npy_data.copy()
_anomaly_label = anomaly_label.copy()
interval_len = replacing_lengths[0]
abnormal_column_idx = abnormal_columns[0] < 0.5
if not(abnormal_column_idx.any()):
    abnormal_column_idx[np.random.randint(0, N_COLUMNS)] = True
replacing_column_idx = np.random.choice(N_COLUMNS, size=len(abnormal_column_idx[abnormal_column_idx]), replace=False)

replacing_index = np.random.randint(0, total_data_len-interval_len+1)
external_interval = total_npy_data[replacing_index:replacing_index+interval_len, replacing_column_idx].copy()
target_interval = syn_data[target_indices[0]:target_indices[0]+interval_len, abnormal_column_idx].copy()

weights = np.concatenate((np.linspace(0, MAX_EXTERNAL_INTERVAL_RATE, num=interval_len//2),
                           np.linspace(MAX_EXTERNAL_INTERVAL_RATE, 0, num=(interval_len+1)//2)), axis=None)
syn_data[target_indices[0]:target_indices[0]+interval_len, abnormal_column_idx] = weights[:, None] * external_interval\
    + (1 - weights[:, None]) * target_interval

_anomaly_label[target_indices[0]:target_indices[0]+interval_len] = 1
syn_data = np.concatenate((syn_data, additional_data, _anomaly_label), axis=1)

file_name = single_type_data_dir+date+'_0.npy'
meta_data.append([_date+'_0', 'replacing', (target_indices[0], target_indices[0]+interval_len),
                  tuple(np.argwhere(abnormal_column_idx).flatten()), (date in test_clear_date)])
np.save(file_name, syn_data)
```

- 피크값 추가 함수

```
# Synthesize anomalies. - peak noising
syn_data = npy_data.copy()
_anomaly_label = anomaly_label.copy()
interval_len = replacing_lengths[2]
abnormal_column_idx = abnormal_columns[2] < 0.5
if not(abnormal_column_idx.any()):
    abnormal_column_idx[np.random.randint(0, N_COLUMNS)] = True

peak_indices = np.random.randint(target_indices[2], target_indices[2]+interval_len,
                                size=len(abnormal_column_idx[abnormal_column_idx]))
peak_values = syn_data[peak_indices, abnormal_column_idx].copy()

peak_errors = np.random.uniform(np.minimum(0, MIN_PEAK_ERROR-peak_values), np.maximum(0, 1-peak_values-MIN_PEAK_ERROR))
peak_values = peak_values + peak_errors + ((peak_errors > 0).astype(int) * 2 - 1) * MIN_PEAK_ERROR
syn_data[peak_indices, abnormal_column_idx] = peak_values

_anomaly_label[peak_indices] = 1
syn_data = np.concatenate((syn_data, additional_data, _anomaly_label), axis=1)

file_name = single_type_data_dir+date+'_2.npy'
meta_data.append([date+'_2', 'peak', (np.min(peak_indices), np.max(peak_indices)+1),
                tuple(np.argwhere(abnormal_column_idx).flatten()), (date in test_clear_date)])
np.save(file_name, syn_data)
```

- 가우시안 노이즈 추가 함수

```
# Synthesize anomalies. - white noising
syn_data = npy_data.copy()
_anomaly_label = anomaly_label.copy()
interval_len = replacing_lengths[3]
abnormal_column_idx = abnormal_columns[3] < 0.5
if not(abnormal_column_idx.any()):
    abnormal_column_idx[np.random.randint(0, N_COLUMNS)] = True

noised_data = syn_data[target_indices[3]:target_indices[3]+interval_len, abnormal_column_idx]\
    + np.random.randn(interval_len, len(abnormal_column_idx[abnormal_column_idx])) * WHITE_NOISE_LEVEL
noised_data[noised_data > 1] = 1
noised_data[noised_data < 0] = 0
syn_data[target_indices[3]:target_indices[3]+interval_len, abnormal_column_idx] = noised_data

_anomaly_label[target_indices[3]:target_indices[3]+interval_len] = 1
syn_data = np.concatenate((syn_data, additional_data, _anomaly_label), axis=1)

file_name = single_type_data_dir+date+'_3.npy'
meta_data.append([date+'_3', 'white_noise', (target_indices[3], target_indices[3]+interval_len),
                tuple(np.argwhere(abnormal_column_idx).flatten()), (date in test_clear_date)])
np.save(file_name, syn_data)
```

나. 사용 환경

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

다. 사용 방법

- 입력과 출력 데이터는 parquet 형식을 사용함
- 전체 코드는 노트북 파일 형식(.ipynb)으로 데이터 파일을 지정한 후 일련의 과정을 따라 이상치를 합성할 수 있음
- 월 단위 데이터를 입력하면 모델 학습을 위한 학습 데이터(training data)와 평가 데이터(test data)를 나누고, 평가 데이터에 이상치를 합성하여 저장함
- 데이터 정보로 완전 충전이 된 날(통상적인 맑은 날)의 구분과 데이터 충전 구간이 기록됨

라. 코드

OpenESS/SNU_AI/data_preprocess/ess_preprocess/synthetic_data_generation.ipynb