


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 : 에너지원의 이상감지 및 수명예측 알고리즘
- 작성일자 : 2023년 11월 17일

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

수행기관	성명/직위	확인
슈어소프트테크(주)	심정민/상무이사	

정보통신기획평가원장 귀하

1. 개요

에너지원의 이상 감지를 위해 DL 모델을 사용하여 이상 감지 모델을 생성하였으며, Point Anomaly 감지를 위해 예측 오차 기반의 DeepAnT 모델, Pattern Anomaly 감지를 위해 재구성 오차 기반의 USAD 모델을 사용하였다. 각 모델에서 감지할 수 있는 이상 데이터 종류가 다르기 때문에 두 모델을 병렬적으로 연결하여 이상 탐지를 수행하도록 구성하였다.

2. 알고리즘

2.1 DeepAnT

CNN 기반으로 이루어진 모델로 학습 시 Convolution Network를 통해 데이터의 특징을 학습한 후 인풋 데이터를 기반으로 예측을 수행하여 예측 오차 기반으로 이상 데이터를 탐지한다.

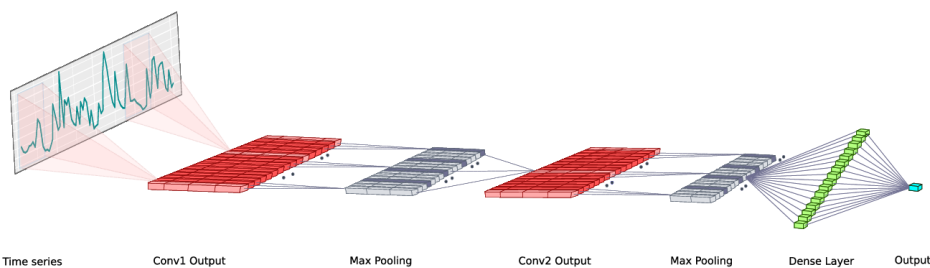


FIGURE 1. DeepAnT architecture for time series prediction: A convolutional neural network with two convolutional layers, two max pooling, and a fully connected layer.

그림 1 DeepAnT 모델 구조도

```
class DeepAnt(nn.Module):
    def __init__(self, seq_len, p_w):
        super().__init__()

        self.convblock1 = nn.Sequential(
            nn.Conv1d(in_channels=1, out_channels=32, kernel_size=3,
padding='valid'),
            nn.ReLU(inplace=True),
            nn.MaxPool1d(kernel_size=2)
        )

        self.convblock2 = nn.Sequential(
            nn.Conv1d(in_channels=32, out_channels=32, kernel_size=3,
padding='valid'),
            nn.ReLU(inplace=True),
            nn.MaxPool1d(kernel_size=2)
        )

        self.flatten = nn.Flatten()

        self.denseblock = nn.Sequential(
            nn.Linear(32, 40),
            #nn.Linear(96, 40), # for SEQL_LEN = 20
            nn.ReLU(inplace=True),
            nn.Dropout(p=0.25),
        )

        self.out = nn.Linear(40, p_w)
```

```
def forward(self, x):
    x =self.convblock1(x)
    x =self.convblock2(x)
    x =self.flatten(x)
    x =self.denseblock(x)
    x =self.out(x)
    return x
```

표 1 DeepAnT 모델 코드 구조

source :

https://github.com/keti-dp/OpenESS/blob/suresoft_upload/SURE_DOC/AI_Model/DEMO/deepant_pred.py

2.2 USAD

USAD 모델은 한 개의 Encoder와 두 개의 Decoder가 연결되어있는 구조로, 학습 시 한 Encoder는 원본과 가장 유사하게 재구축을 수행하고, 다른 Decoder는 원본 데이터는 원본과 가장 유사하게 재구축을 수행하고, Decoder를 거쳐온 데이터는 원본과 다르게 재구축을 수행하도록 학습한다.

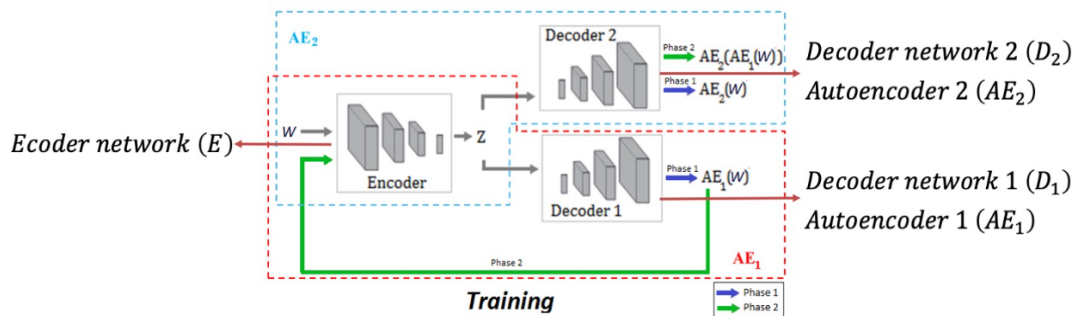


그림 2 USAD 모델 학습 과정

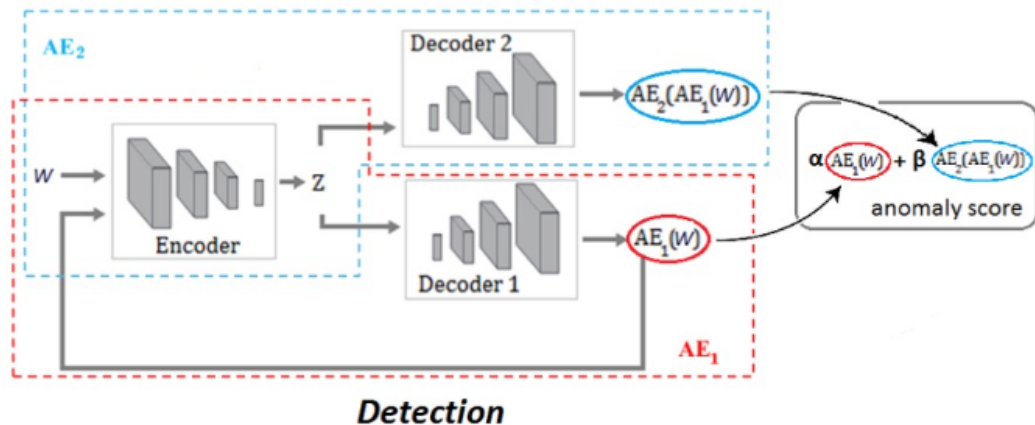


그림 3 USAD 모델 탐지 과정

```
class Encoder(nn.Module):
    def __init__(self, in_size, latent_size):
```

```

super().__init__()
self.linear1 =nn.Linear(in_size, int(in_size/2))
self.linear2 =nn.Linear(int(in_size/2), int(in_size/4))
self.linear3 =nn.Linear(int(in_size/4), latent_size)
self.relu =nn.ReLU(True)

def forward(self, w):
    out =self.linear1(w)
    out =self.relu(out)
    out =self.linear2(out)
    out =self.relu(out)
    out =self.linear3(out)
    z =self.relu(out)
    return z

class Decoder(nn.Module):
    def __init__(self, latent_size, out_size):
        super().__init__()
        self.linear1 =nn.Linear(latent_size, int(out_size/4))
        self.linear2 =nn.Linear(int(out_size/4), int(out_size/2))
        self.linear3 =nn.Linear(int(out_size/2), out_size)
        self.relu =nn.ReLU(True)
        self.sigmoid =nn.Sigmoid()

    def forward(self, z):
        out =self.linear1(z)
        out =self.relu(out)
        out =self.linear2(out)
        out =self.relu(out)
        out =self.linear3(out)
        w =self.sigmoid(out)
        return w

class UsadModel(nn.Module):
    def __init__(self, w_size, z_size):
        super().__init__()
        self.encoder =Encoder(w_size, z_size)
        self.decoder1 =Decoder(z_size, w_size)
        self.decoder2 =Decoder(z_size, w_size)

```

표 2 USAD 모델 코드 구조

Source :
https://github.com/keti-dp/OpenESS/blob/suresoft_upload/SURE_DOC/AI_Model/DEMO/usad_predict.py