1번 문제

첨부파일 개요

- ans.py: 주어진 stream에 대해 주기를 예측하는 DNN 알고리즘 구현체 (세부문제 2)

1 세부 문제 1

```
class LFSR128():
  def __init__(self, s):
    # 초기값을 저장한다
    self.s = s & ((1 << 128) - 1)

  def step(self):
    # 출력값을 저장해 놓는다
    o = self.s & 1

    # 다음 MSB를 계산한다
    n = 0
    for i in [0,1,2,7]:
        n ^= (self.s >> i) & 1

    # 상태를 업데이트한다
    self.s = (self.s >> 1) | (n << 127)

    # 출력값을 반환한다
    return o
```

2 세부 문제 2

2.1 목표

주어진 stream을 보고 주기를 예측하는 DNN기반 알고리즘을 제시한다. stream과 함께 LSFR의 크기가 주어진다고 가정하였다.

2.2 구조

전반적인 알고리즘 구조는 그림 1에서 확인할 수 있으며 총 전처리 단계, 학습 단계, 추론 단계로 총 3단계로 구성된다.

1. 1단계는 전처리 단계로, 주어진 stream을 학습 데이터 셋으로 바꾸는 과정이다. stream을 연속해서 n개의 bit (LSFR의 크기) 만큼 자른 벡터를 모델의 입력 데이터로 만들고, 그 다음 1-bit를 onehot encoding 한 2차원 벡터를 타겟 데이터로 설정한다.

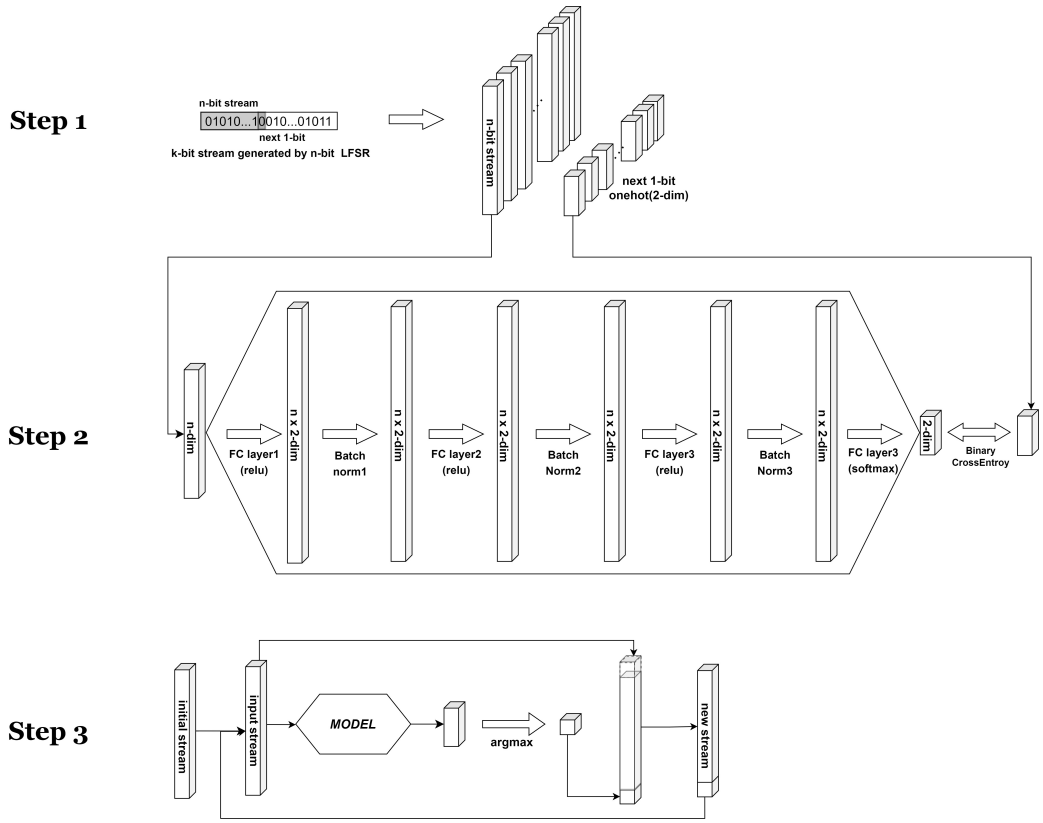

Figure 1: 1단계 전처리 과정, 2단계 모델 학습 과정, 3단계 주기 추론 과정으로 구성되어있다.

2. 2단계는 1단계에서 전처리한 데이터로 딥러닝 모델을 학습시키는 과정이다. 모델의 구조는 3개의 linear layer와 3개의 batch normalization layer로 구성되어 있으며 마지막 layer를 제외한 layer의 output dimension은 LSFR의 크기의 2배로 설정하였다. 활성화 함수는 마지막 layer를 제외하고 relu를 사용했으며 마지막 layer에서는 softmax함수를 사용했다. 오차함수는 binarycrossentropy 함수를 사용하였고, 최적화 알고리즘으로는 Adam을 사용하였으며 과적합(overfitting) 방지를 위해 weight decay를 사용하였다. 데이터셋은 훈련용 데이터(training data)와 검증 데이터(validation data)로 나누고 훈련용 데이터로만 학습을 진행하였다. 학습은 미니 배치 방식으로하였고 한 iter마다 20 epoch만큼 학습을 진행하였다. 모델의 일반화 성능이 매우 중요하기 때문에 iter 학습 후 검증 데이터셋에서 정확도가 일정 횟수 이상 연속해서 100% 일 때 학습을 종료하도록 하였다.

3. 3단계는 학습이 끝난 모델로 주어진 stream의 주기를 추론하는 단계이다. 추론 시작 전 먼저 주어진 bit stream 중 제일 앞의 n-bit stream을 initial streram으로 정한다. 추론은 루프로 진행되며 루프의 진행은 다음과 같다. n-bit의 input stream을 모델의 input으로 넣고 출력한 2차원 onehot 벡터를 1-bit로 변환한다. 변환한 1-bit를 input stream 제일 뒤에 붙이고 제일 앞의 1-bit를 제거하여 n-bit의 new stream를 만든다. 다음 루프에서 이 new stream을 input stream으로 사용한다. 제일 처음 루프의 input stream은 initial stream으로 하며 new stream이 initial stream과 같다면 몇 번째 루프인지 출력하고 루프를 종료한다.

다음은 알고리즘의 pseudo code이다.

Algorithm 1 stream 주기 예측

Input: k-bit stream **S**, size of LSFR **n**
$D_{train}, D_{validation} \leftarrow preprocess\_stream\_data(\mathbf{S}, val)$ ▷ 데이터 전처리
$init\_stream \leftarrow S[:n]$ ▷ initial stream 설정
$input\_stream \leftarrow init\_stream$
$i \leftarrow 0$
while $i < N$ **do**
  $DNN\_model\_train(D_{train}, epoch)$ ▷ epoch 만큼 모델 학습
  $val\_acc \leftarrow cal\_val\_acc(DNN\_model, D_{validation})$ ▷ 검증 데이터 정확도 계산
  if $val\_acc == 100$ **then**
    $i \leftarrow i + 1$
  else
    $i = 0$
  end if
end while
for $1 \leq i \leq 2^n$ **do**
  $new\_bit \leftarrow DNN\_model(input\_stream)$
  $input\_stream \leftarrow input\_stream[1:]$
  $input\_stream.append(new\_bit)$
  if $input\_stream == init\_stream$ **then**
    return i
  end if
end for

2.3 실험 결과

자원의 한계로 인해 16-bit 크기의 LFSR에서 검증하였다. 모델의 learnig rate는 1e-4, weight_decay는 0.3, test accuracy 100이 20 iter 연속으로 나오는 경우 학습을 종료하였다. 학습은 stream의 전체 주기의 0.2 만큼의 stream 만 주어진다고 설정하였다. 그중 0.75 만큼의 stream을 학습 데이터로 사용하였고 나머지는 검증데이터로 사용하였다. 주기가 짧은 stream의 경우 학습 데이터가 부족하여 정확하지 않은 실험 결과를 얻었지만 2^{13} 이상 주기의 stream의 경우 100 정확도로 추론 가능하였다. 실험 결과는 다음과 같다.

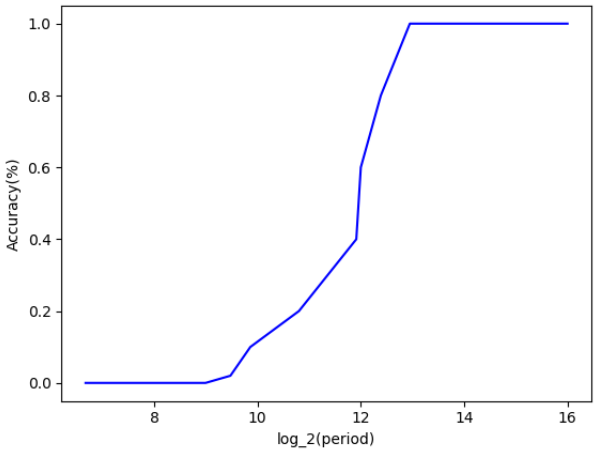

Figure 2: 실험 결과