**부산대학교 정보컴퓨터공학부**

**2023년 전기 졸업과제 중간보고서**

**딥러닝 기반 워크로드 분석을 통한**

**SSD 성능 개선**

분과 : D

팀명 : 빡공

지도교수 : 안성용

201824476 박영훈

201824472 박선민

201724409 공희찬

# 

# **차 례**

[**차 례 2**](#_gjdgxs)

[**1 과제 배경 및 목표 3**](#_60t3jp28d884)

[1.1 과제 배경 3](#_oqn0hcrj8p1g)

[1.2 과제 목표 3](#_imugvbywtcf6)

[**2 요구조건 및 제약 사항 분석에 대한 수정사항 4**](#_1fob9te)

[2.1 요구조건 4](#_3znysh7)

[2.2 제약사항 4](#_2et92p0)

[**3 설계 상세화 및 변경 내역 5**](#_tyjcwt)

[3.1 모델 설계 5](#_2gklrtci4ekr)

[3.2 모델 구조 5](#_nwm262q821fq)

[**4 보고 시점까지의 과제 수행 내용 및 중간 결과 7**](#_26di4c2i3afz)

[4.1 데이터 전처리 7](#_hprqibhhdmmz)

[4.2 레이블링 9](#_bwj7wpqb8xjs)

[4.3 기계학습 모델 설계 및 시뮬레이터 제작 11](#_ifuahdw73up)

[**5 갱신된 과제 추진 계획 12**](#_bg10f0ecaqmr)

[**6 구성원별 진척도 13**](#_61ls8b84jgdh)

[**7 참고문헌 13**](#_d89yp924u200)

# **1 과제 배경 및 목표**

## 1.1 과제 배경

SSD(Solid-State Drive)는 플래시 메모리를 기반으로 한 비휘발성 저장 장치이다. 오늘날 NAND 플래시 기반 SSD는 HDD(Hard-Disk Drive)에 비해 높은 I/O 처리량, 신뢰성, 그리고 낮은 지연시간 등의 강점 덕분에 널리 사용되고 있다. 하지만 SSD는 읽기와 쓰기, 삭제의 단위가 다르고 데이터를 덮어쓰기 할 수 없는 성질 때문에 성능 저하가 발생할 수 있다. 읽기와 쓰기는 페이지(Page) 사이즈 단위로 실행되며, 삭제는 블록(Block) 사이즈 단위로 실행된다. 또한 SSD 드라이브의 페이지는 덮어쓰기 될 수 없는 특성을 가지며, 데이터를 수정하기 위해서는 쓰기 전에 삭제(Erase before Write) 해야 하는 성질 때문에 I/O 요청의 응답시간이 늘어나고, 이는 SSD의 성능 하락을 일으킨다. 또한, 셀마다 쓰기 횟수가 제한되어 있기 때문에 너무 빈번하게 쓰기, 삭제 과정을 거친다면 사용 불가능한 상태 (wearing off)가 된다. 만약 동일한 페이지에 쓰기가 집중될 경우, 해당 페이지가 속한 블록은 지속적인 삭제가 발생하게 되어, 블록에 속한 다른 페이지들도 함께 수명이 단축되게 된다.

## 1.2 과제 목표

SSD의 성능을 개선하기 위해서는 데이터의 입출력 패턴을 분석하는 것이 매우 중요하므로, 본 연구에서는 딥러닝 기법을 이용하여 데이터의 접근 패턴을 분석하고, 이를 이용해 SSD의 성능을 개선하는 기법을 연구하고자 한다. 우리는 우선 입출력 데이터를 Hot 데이터와 Cold 데이터로 분류할 것이다. Hot 데이터는 빈번하게 변경되는 데이터를 의미하며, Cold 데이터는 드물게 변경되는 데이터를 의미한다. 데이터를 분류하는 이유는, 만약 페이지의 일부는 Cold 데이터, 그리고 나머지 일부는 Hot 데이터를 가진다면, Wear Leveling을 위하여 Hot 데이터가 Garbage Collection될 때마다 Cold 데이터도 같이 옮겨 다녀야 할 것이다. 이렇게 Hot 데이터와 함께 Cold 데이터가 따라 다녀야 한다면, Write Amplification은 심해질 것이다. 그러므로, 데이터를 분류하여 각 데이터를 최대한 서로 다른 페이지로 분리함으로써 데이터의 Write Amplication을 줄이며, Garbage collection이 좀 더 효율적으로 동작하도록 할 것이다. 필요에 따라 Hot 데이터와 Cold데이터로 구분 짓는 것뿐만 아니라, Warm 데이터와 Cool 데이터를 포함시키는 것이 더 효과적이라면 데이터의 라벨링을 유동적으로 변경할 것이다. 즉 우리는 기계학습을 통하여 Hot data와 Cold data로 분류하는 기준을 세우고 입출력 워크로드 분석에 적합한 학습 모델을 개발할 것이다. 또한 모델의 학습 결과를 이용하여 FTL의 성능을 개선하기 위하여 Garbage Collection 알고리즘을 구현할 것이며, 구현한 알고리즘은 SSD 시뮬레이션을 통하여 기존 알고리즘과 성능을 비교하고, 평가할 것이다.

# **2 요구조건 및 제약 사항 분석에 대한 수정사항**

## 2.1 요구조건

과제의 목표는 Hot/Cold 데이터를 구분하기 위한 머신러닝 모델을 설계하고 학습하여 SSD 쓰기 성능을 향상시키는 것이다. GC(Garbage Collection) 알고리즘을 통해 성능을 향상시킬 것이며 기존 SSD에 사용되는 알고리즘과 학습을 통해 얻어진 GC 알고리즘을 비교 평가해야 한다.

## 2.2 제약사항

이번 과제에서의 주요 제약 사항은 실제 SSD에 구현된 모델을 적용하여 실험할 수 없다는 점이다. 이 문제를 해결하기 위해 우리는 기존에 사용되는 GC 알고리즘과 우리가 개발한 머신러닝 모델의 출력을 비교하는 방식으로 성능을 평가할 것이다. 즉 에뮬레이터가 아닌 시뮬레이터의 구현을 통해 성능 평가가 이루어진다.

# **3 설계 상세화 및 변경 내역**

## 3.1 모델 설계

우리는 시계열 데이터로서 사용되는 워크로드 데이터를 다루어야 하기 때문에 RNN(Recurrent Neural Network)을 선택했다. 이 모델의 구조는 아래 그림 1과 동일하다.

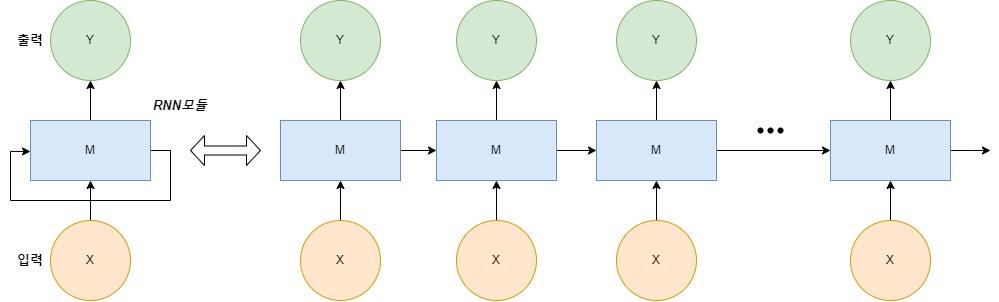


그림 1: 기본적인 RNN 구조

그러나 RNN 모델은 규모가 큰 데이터셋에서 두 가지 주요 문제가 발생한다. 첫 번째 문제는 기울기 소멸(Vanishing Gradient) 문제로, 신경망을 통과하면서 작은 미분값이 반복적으로 곱해져서 값이 거의 0에 수렴되는 현상이다. 이로 인해 학습이 어려워지며, 신경망의 전반부에서 데이터가 손실되는 문제가 발생할 수 있다. 두 번째 문제는 기울기 폭발(Exploding Gradient) 문제로, 반대로 신경망을 통과하면서 큰 미분값이 반복적으로 곱해져서 가중치가 매우 커지는 현상이다. 이는 모델의 예측 능력을 저하시킬 수 있다. 이러한 문제에 대해 여러 가지 해결책이 제시되었고, 그 중에서 LSTM(Long Short Term Memory)을 채택했다. LSTM은 RNN과 비교하여 메모리 유지가 더 오래 지속되므로, 우리가 다루는 Hot/Cold 분류를 위해 장기간 기억이 필요한 LBA(Logical Block Address) 데이터셋에 적합하다고 판단했다.

## 

## 3.2 모델 구조

이번 과제에서 다루는 LSTM(Long Short Term Memory)에는 각기 다른 특성을 가진 여러 구조들이 존재한다. 이러한 다양한 LSTM 모델 중에서 우리는 사용되는 데이터셋에 가장 좋은 출력을 내는 모델을 선정해야 한다. 따라서 각 모델을 실험하고 그 결과를 비교해봐야 할 필요가 있다. 첫 번째로, 기본적인 단변량 LSTM 모델에 대한 실험을 진행했으며, 이 모델의 구조는 아래 그림 2과 동일하다.

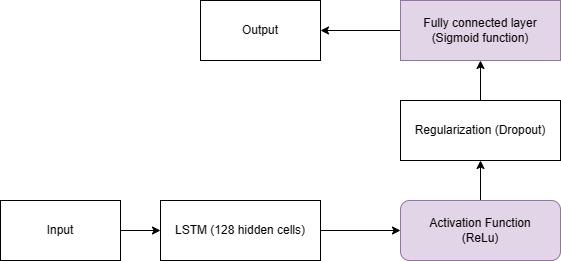


그림 2: 기본적인 LSTM 구조

단변량 LSTM 모델을 사용하여 학습할 때, 우리는 LBA(Logical Block Address)를 특징값으로만 활용했다. LSTM은 한 계층만을 사용하고 활성화(Activation) 함수로는 ReLU(Rectified Linear Unit)를 선택했다. ReLU는 양수에 대해서는 값을 그대로 반환하고, 음수에 대해서는 0으로 치환하는 활성화 함수이다. 이 함수를 사용하면 기울기 소실 문제(Vanishing Gradient)가 발생하지 않고 학습 속도가 빨라지는 장점이 있다. 우리가 다루는 학습 데이터는 기울기 소실 문제가 발생할 수 있으므로, ReLU를 사용하기로 결정했다. 또한 정규화(Normalization) 함수로는 Dropout을 채택했다. 이 모델은 Hot/Cold 이진 분류를 수행하는데, 이를 위해 마지막 계층에서 sigmoid 함수를 사용하여 예측값을 0과 1로 변환했다. 이러한 활성화 함수와 정규화 함수, 그리고 신경망 계층들은 다른 함수들과 비교하며 학습 결과를 확인하고 실험을 진행할 계획이다. 최적화 알고리즘으로는 Adam 경사하강법 알고리즘을 사용했다. 이는 각 단계에서 가중치를 최적화하고 기울기에 가해지는 편향을 조정하여 학습 속도를 개선한다. 마찬가지로 이진 분류 모델이므로 교차 엔트로피 함수를 손실함수로 사용했다.

# **4 보고 시점까지의 과제 수행 내용 및 중간 결과**

## 4.1 데이터 전처리

실험에 사용할 데이터는 UMass Trace Repository의 OLTP Application I/O Trace[1] 데이터이다. 각 Trace Record는 단일 I/O command를 의미하며, 5개의 필드로 나누어진다. 데이터의 예시는 아래와 같다.

| Application Specific Unit | Logical Block Address | Size(byte) | Opcode | Timestamp(sec) |
| --- | --- | --- | --- | --- |
| 1 | 240840 | 3072 | w | 0.026214 |

표 1: 전처리 수행 전의 I/O Trace

표 1의 데이터는 시작으로부터 0.026214초 지난 시점에, Application Specific Unit 1의 Logical Block Address 240840번지에 3072byte의 쓰기가 발생했다는 것을 의미한다. 이 trace에서 Logical Block Address의 Block은 HDD(Hard Disk Drive)의 sector를 의미하며, sector 1개의 크기는 512byte이다. trace에서 Size의 단위는 byte이며, Timestamp의 단위는 second(초)이다. Application Specific Unit은 현재 진행하는 실험에 의미가 없으므로 제외할 필드이며, 실험의 주된 목적은 Write Amplication 감소이므로 Opcode가 “w” 즉, Write인 데이터만을 다룰 것이다.

우선 사용하지 않을 Application Specific Unit 필드와 Opcode필드를 제거하고, Logical Block Address를 Sector Number로 치환하여 Sector Number, Size, Timestamp 필드만 남겨두었다. Size 필드를 sector의 크기인 512byte로 나누면 쓰기 시 차지하는 sector의 개수를 구할 수 있다. 위의 예시 데이터를 새롭게 변환한다면 다음과 같다.

| Sector Number | # of Sectors | Timestamp(sec) |
| --- | --- | --- |
| 240840 | 6 | 0.026214 |

표 2: Application Specific Unit, Opcode 필드를 제외하고 Size를 Sector 단위로 변환

표 2의 데이터는 Sector Number 240840번부터 240845번까지 6개의 sector에 I/O trace의 시작으로부터 0.026214초 후에 쓰기 작업이 실행되었다는 것을 의미한다. 위 예시 데이터의 # of Sectors 필드를 각 Sector Number에 적용한 데이터는 다음과 같다.

| Sector Number | Timestamp(sec) |
| --- | --- |
| 240840 | 0.026214 |
| 240841 | 0.026214 |
| 240842 | 0.026214 |
| 240843 | 0.026214 |
| 240844 | 0.026214 |
| 240845 | 0.026214 |

표 3: # of Sectors 필드를 각 Sector Number에 적용

이번 실험에서는 sector 단위가 아닌 페이지 단위로 레이블링을 진행할 예정이므로, sector를 페이지 단위로 환산해야 한다. sector 1개의 크기는 512byte이고, page 1개의 크기를 4KB로 설정하였으므로 하나의 page는 8개의 sector로 이루어져 있다. 각 sector에 해당하는 page를 부여하여 전처리한 예시는 다음과 같다.

| Page | Timestamp(sec) |
| --- | --- |
| 30105 | 0.026214 |

표 4: Sector Number에 맞는 Page 할당

우리는 이번 실험에서 군집화 진행 시 사용할 매개변수를 page의 접근 빈도, page 접근시간 간격의 평균, page 접근시간 간격의 표준편차로 설정하였다. 위의 전처리된 데이터에서 해당하는 매개변수들을 얻기 위한 방법으로 두 가지 방법을 떠올릴 수 있다.

첫 번째 방법은 이차원 배열을 사용하여 이차원 배열 내부의 일차원 배열을 하나의 페이지라고 생각하고, 일차원 배열의 원소를 sector로 사용하는 것이다. 만약 해당 sector에 쓰기가 발생했다면, 따로 Timestamp를 기록하며 해당 원소의 값을 1만큼 증가시켜 빈도를 기록한다. 두 번째 방법은 Python의 dictionary 자료구조를 활용하여, Sector Number에 해당하는 page를 dictionary의 key로 설정하고, value를 Timestamp로 이루어진 list로 설정하는 방법이다. 실험에서는 두 번째 방법을 선택했는데, 첫 번째 방법은 이상치에 취약하다는 단점이 있기 때문이다. 원시 데이터를 분석해보니, Sector Number가 1억보다 큰 I/O trace가 3건 존재했고, 나머지 모든 I/O trace들의 Sector Number는 1000만보다 작았다. 만약 Sector Number를 8만큼 나눈 크기의 이차원 배열을 선언할 경우, 필요하지 않은 메모리 공간을 많이 차지할 것이다. 또한 두 번째 방법을 채택한다면, 특정 page를 key로 갖는 Timestamp를 담은 list의 크기가 해당 page의 Frequency가 될 것이고, Timestamp를 담은 list를 오름차순으로 정렬하여 시간순으로 나열하였을 때 Time Interval을 구하기도 용이하다. 이와 같은 방법으로 최종적으로 데이터를 전처리한 결과는 다음과 같다.

| Page | Frequency | Time Interval Average | Time Interval Std |
| --- | --- | --- | --- |
| 30105 | 292642 | 0.147616 | 2.953861 |

표 5: 최종 전처리 데이터

표 5의 데이터는 page 30105번지에서 292642회 쓰기가 진행되었으며, page 접근시간 간격의 평균은 0.147616초이고 page 접근시간 간격의 표준편차가 2.953861초임을 의미한다. 만약 page에 단 한번만 접근한 경우(Frequency가 1인 경우)에는 접근시간 간격을 구할 수 없으므로, 해당 page의 접근시간 간격의 평균을 모든 데이터의 접근시간 간격 평균값들 중 최댓값으로 치환하였고, 접근시간 간격의 표준편차는 0으로 치환하였다.

## 4.2 레이블링

데이터가 자주 수정되면 Hot, 거의 수정되지 않으면 Cold로 구분하고 필요에 따라 그 사이에도 군집을 추가하여 데이터를 분류하기 위해 비지도 학습을 진행했다. 군집 수를 지정하지 않아도 되고, 이상치 대응 성능이 좋은 DBSCAN(Density-Based Spatial Clustering of Applications with Noise)를 군집화 도구로 고려했다. 하지만, 대용량 데이터를 다뤄야 하기 때문에 DBSCAN보다 단순한 k-means 알고리즘을 사용했다.

k-means의 매개변수로 페이지 접근 빈도, 접근 시간 간격 평균, 접근 시간 간격의 표준편차를 사용했다. 짧은 시간에 집중적으로 수정된 후 긴 시간 동안 수정되지 않은 데이터의 경우, 접근 시간 간격의 평균만으로 구분할 수 없고, 접근 간격의 표준 편차를 추가 해야 했다.

데이터 분석 시, 변수들의 단위 차이로 인해 변수들의 특성을 무시하고 단순히 수치만 큰 값을 기준으로 분류되는 것을 피하기 위해 표준화를 수행하였다. 표준화 함수로 MinMaxScaler를 고려했지만, 이 함수는 데이터의 원래 분포를 유지하는 특성을 가지고 있어 접근 빈도가 낮은 쪽으로 데이터가 편중되어 있는 우리의 연구에 적합하지 않다고 판단했다.

MinMaxScaler 대신 사용한 StandardScaler의 경우 평균과 분산을 각 0과 1로 조정하고 분포 역시 변화하므로, 엘보우 기법으로 k값 4를 얻을 수 있었다.

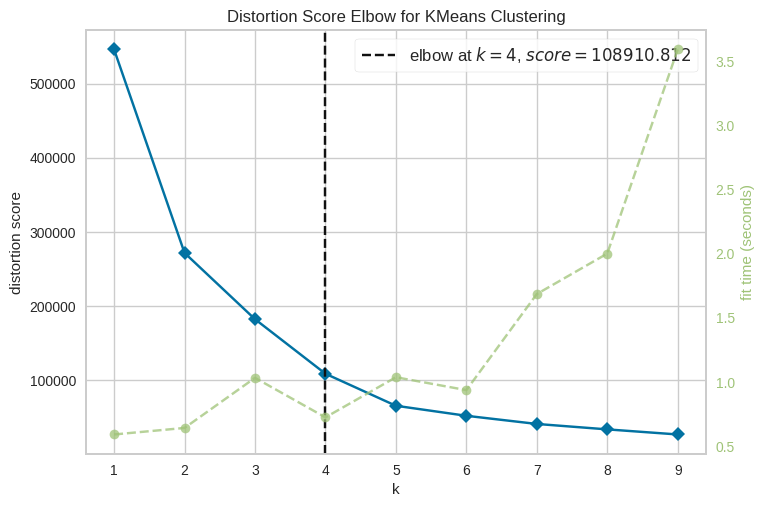


그림 3: 데이터 엘보우 기법 k 설정

| 군집 | 접근 횟수 평균 | 시간 간격 평균 | 시간 간격 표준 편차 | 데이터 개수 |
| --- | --- | --- | --- | --- |
| 0 | 289.19 | 114.17 | 452.46 | 83,276 |
| 1 | 125.37 | 270.10 | 1,490.10 | 48,482 |
| 2 | 7.86 | 13,593.36 | 0 | 4,736 |
| 3 | 88,621.59 | 0.57 | 7.54 | 47 |

표 6: 데이터 군집화 결과 수치

2번 군집이 극히 적게 쓰기 요청된 페이지로 이뤄진 군집 즉, Cold 페이지 군집

3번 군집이 극히 많이 쓰기 요청된 페이지로 이뤄진 군집 즉, Hot 페이지 군집

0번 군집과 1번 군집이 Cold와 Hot 사이의 Warm 군집

각 변수를 고려했을 때, 0번 군집이 1번 군집에 비해 접근 횟수 평균이 크고 시간의 평균과 표준 편차가 작기 때문에 비교적 짧은 시간에 집중적으로 많이 수정되었다고 해석했다.

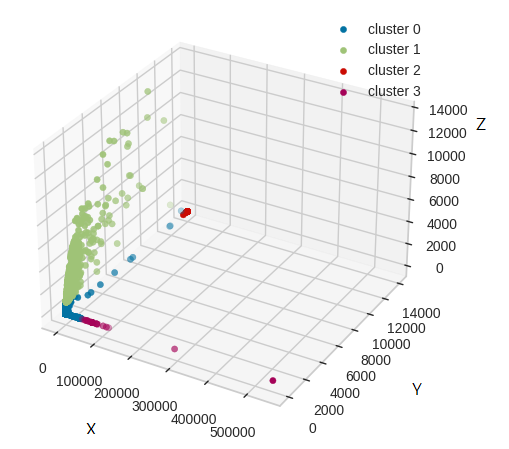


그림 4: 표6의 데이터를 3차원 공간에 표현한 결과

x축은 접근 횟수, y축은 시간 간격 평균, z축은 시간 간격 표준편차이다.

## 4.3 기계학습 모델 설계 및 시뮬레이터 제작

앞선 3.2장에서 설명했듯이, 이번 과제에서 다루는 LSTM(Long Short Term Memory)에는 각기 다른 특성을 가진 여러 구조들이 존재하는데, 이 중 기본적인 단변량 LSTM 모델에 대한 실험을 진행하였다. 더 나아가, Stacked LSTM 모델을 실험하기 위해 모델을 구성하고 있으며 다변량 모델을 구성한 실험 또한 진행할 예정이다. 또한 우리는 설계한 모델을 실험하기 위해 시뮬레이터를 작성중에 있다. Hot/Cold가 구분되지 않은 데이터를 시뮬레이터를 통한 Garbage Collection 진행 시 발생하는 Write Amplification을 기준으로 하여, 우리가 설계한 머신러닝 모델을 통해 Hot/Cold를 분류한 데이터를 시뮬레이터를 통한 Garbage Collection 진행 시 발생하는 Write Amplification을 비교할 것이며, 이 중 가장 Write Amplification이 개선된 모델을 도출하기 위해 다양한 구조를 시도하며 실험을 진행할 예정이다.

# **5 갱신된 과제 추진 계획**

| **5월** | | | | | **6월** | | | | | **7월** | | | | | | **8월** | | | | | **9월** | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1주** | **2주** | **3주** | **4주** | **5주** | **1주** | **2주** | **3주** | **4주** | **5주** | **1주** | **2주** | **3주** | **4주** | **5주** | **6주** | **1주** | **2주** | **3주** | **4주** | **5주** | **1주** | **2주** | **3주** | **4주** | **5주** |
| **착수보고서** | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **데이터 가공 및 모델 공부** | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | **1차 데이터 전처리** | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  | **레이블링 코드 작성** | | | | |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  | **2차 데이터 전처리** | | | | |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  | **중간보고서 준비** | | | | |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **모델 개량, 시뮬레이터 개발** | | | | | | |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **최종 발표/보고서 준비** | | | |

# **6 구성원별 진척도**

| 조원 | 수행 내용 |
| --- | --- |
| 공희찬 | 데이터 전처리 (완료)  머신러닝 모델 개발 (예정) |
| 박선민 | 머신러닝 모델 설계 (진행 중)  시뮬레이터 개발 (예정) |
| 박영훈 | 데이터 레이블링 (완료)  시뮬레이터 개발 (예정) |
| 공통 | 자료 수집 및 요구사항 분석(완료)  머신러닝 모델 개량 및 시뮬레이터 성능 비교 (예정) |

# 

# **7 참고문헌**

[1] <https://traces.cs.umass.edu/index.php/Storage/Storage>

[2] <https://www.ibm.com/kr-ko/topics/recurrent-neural-networks>