# **차 례**

[1 과제 배경 및 목표](#_djeboe1oahxm) 1

[1.1 과제 배경](#_y108gk897gk6) 1

[1.2 과제 목표](#_wq2v0lka1ike) 2

[2. 기존 연구](#_2viq8x6l6emf) 4

[3. 제약사항과 해결방안](#_qpon3rj01p5c) 5

[4. 시스템설계](#_d00ro9ytl15o) 6

[4.1 머신러닝 모델](#_1yn4x1hx79e4) 6

[4.1.1 RNN](#_lfiujd8lmyeu) 7

[4.1.2 LSTM](#_m3sfqw4kkxep) 7

[4.2 학습데이터](#_vnk2tzxmhyov) 8

[4.3 시뮬레이션](#_beq15492kac9) 9

[5 개발 일정 및 업무 분담](#_18bqjbbt4nbs) 10

[5.1 개발 일정](#_tuv0xy99a2h) 10

[5.2 업무 분담](#_7wtecjtmfzbu) 10

# 

# **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. 또한 SSD는 HDD의 주요 I/O 지연 원인인 디스크를 회전시키거나 arm을 움직이는 것과 같은 기계적인 동작을 하지 않기 때문에, HDD보다 성능이 뛰어나다.[1] 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. 예를 들어 “삼성 SSD 840 EVO” 제품의 경우, 2048KB의 블록 사이즈를 가지며 각 블록은 256개의 8KB 페이지를 가진다.[2] 이 SSD에 1Byte의 데이터를 읽거나 쓰기 위해서는 반드시 8KB를 읽거나 써야 하고, 1Byte의 데이터를 삭제하기 위해서는 블록 사이즈 단위로 삭제해야 하기 때문에 2048KB의 데이터를 삭제해야만 한다.

| **Parameter** | **Value** |
| --- | --- |
| Page size | 4096 bytes for data |
| Block size | 512KB (128 pages per block) |
| Page read time | 60μs (maximum) |
| Page program time | 0.8ms (typical) |
| Block erase time | 1.5ms (typical) |
| Endurance | 5K program/erase cycles |

표 1. Operational parameters of Samsung NAND flash memory (K9GAG08U0M)[3]

또한 SSD 드라이브의 페이지는 덮어쓰기 될 수 없는 특성을 가지며, 데이터를 수정하는 과정은 다음과 같다. 새로운 버전의 데이터는 “free” 상태의 페이지에 기록되어야 하고, 예전 버전의 데이터가 기록된 페이지는 “invalid” 상태로 기록해야 한다. 만약 새로 쓰고자 하는 블록에 “invalid” 상태의 페이지들을 가지고 있다면, 그 페이지들은 삭제(Erase) 작업 후에 사용할 수 있다. 이 상태로 계속 메모리가 사용되다가 “invalid” 상태의 페이지들이 어느정도 쌓였을 때, 적절한 시점에 이 페이지들을 한꺼번에 삭제하게 되며 이러한 과정을 Garbage Collection이라고 한다.

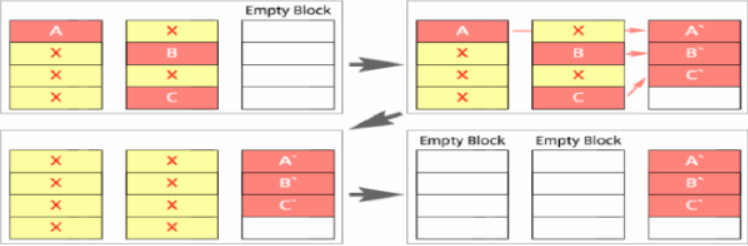


그림 1. Garbage Collection[4]

쓰기 전에 삭제(Erase before Write) 해야 하는 성질 때문에 I/O 요청의 응답시간이 늘어나고, 이는 SSD의 성능 하락을 일으킨다. 또한, SSD는 플래시 메모리이므로 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. 그러므로, 셀 간의 작업을 분산하여 각 블록들이 P/E(Program/Erase) Cycle 한계에 동시에 도달하도록 하는 것이 FTL(Flash Translation Layer)의 중요한 목표 중 하나이다.

# **1.2 과제 목표**

SSD의 성능을 개선하기 위해서는 데이터의 입출력 패턴을 분석하는 것이 매우 중요하다. 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 것 보다 Warm 데이터와 Cool 데이터를 포함시키는 것이 더 효과적이라면, 데이터의 라벨링을 유동적으로 변경할 것이다.

즉 우리는 기계학습을 통하여 Hot data와 Cold data로 분류하는 기준을 세우고, 그 단계를 몇 단계로 나눌 것인가에 대한 판단을 하여 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 기존 연구**

SSD (Solid State Drive)의 성능향상을 도모하기 위해 Hot/Cold 데이터 분류를 사용하는 기존 연구 사례는 많이 존재한다. 이러한 연구들은 주로 SSD의 성능을 향상시키기 위해 데이터의 온도(데이터에 얼마나 자주 액세스되는지)에 따라 데이터를 분류하는 방법을 사용한다. 이를 통해 Hot 데이터는 빠른 응답 시간을 보이는 고속 SSD에 저장되고, Cold 데이터는 비교적 느린 저장장치에 저장되어 저장장치 용량을 효과적으로 확장할 수 있다.

다음은 이와 관련된 대표적인 연구 사례들이다.

1. "Optimizing NAND Flash-Based SSD Performance for Hot and Cold Data Streams" (2012)

이 논문은 Hot/Cold 데이터 분류를 사용하여 NAND 플래시 기반 SSD의 성능을 향상시키는 방법을 제안한다. 이 논문에서는 Hot/Cold 데이터를 분류하는 알고리즘을 설계하고, Hot 데이터를 더 빠른 SSD에, Cold 데이터를 느린 저장장치에 저장함으로써 전체 SSD의 성능을 향상시켰다.

2. "Adaptive Block Size Management for NAND Flash-Based SSDs with Hot and Cold Data Streams" (2014)

이 논문은 Hot/Cold 데이터 분류를 사용하여 NAND 플래시 기반 SSD의 성능을 더욱 향상시키기 위해 블록 크기를 조정하는 알고리즘을 제안한다. 이를 통해 Hot 데이터를 더 빠른 SSD에 저장하고, Cold 데이터를 더 느린 저장장치에 저장하여 전체적인 성능을 향상시켰다.

3. "Hot-Cold Data Management for Solid State Drives" (2017)

이 논문은 Hot/Cold 데이터 분류를 사용하여 SSD의 성능을 향상시키기 위한 여러 가지 기술을 제안한다. 이 논문에서는 Hot/Cold 데이터를 분류하는 알고리즘을 설계하고, 데이터 압축 및 암호화 기술을 사용하여 저장장치 용량을 더욱 효과적으로 확장할 수 있도록 한다. 또한, 이 논문에서는 SSD의 성능을 향상시키기 위한 새로운 Flash 파일 시스템(F2FS)을 소개한다.

4. "A Study on Hot and Cold Data Separation Method for Solid State Drives" (2018)

이 논문에서는 SSD에서 데이터를 Hot/Cold로 분류하여 성능을 향상시키는 방법에 대해 연구하였다. 이를 위해 SSD의 가상 주소 매핑 테이블(VAT)에 대한 연구를 통해 Hot/Cold 데이터를 구분하고, Hot 데이터는 빠른 레벨의 저장장치에, Cold 데이터는 느린 레벨의 저장장치에 저장함으로써 전체적인 성능을 향상시켰다.

5."Exploiting Hot and Cold Data Stream Access Patterns for Efficient Solid State Drive Design" (2018)

이 논문에서는 Hot/Cold 데이터 분류를 사용하여 SSD의 성능을 향상시키기 위한 방법에 대해 제안한다. 이를 위해 Hot/Cold 데이터를 식별하기 위한 알고리즘을 설계하고, Hot 데이터는 빠른 레벨의 저장장치에, Cold 데이터는 느린 레벨의 저장장치에 저장함으로써 전체적인 성능을 향상시켰다. 또한, 이 논문에서는 SSD의 페이지 매핑 알고리즘을 수정하여 성능을 더욱 향상시키는 방법을 제안한다.

6. "Integrating Hot-Cold Data Separation with Storage Virtualization for Flash Memory Storage System" (2019)

이 논문에서는 Hot/Cold 데이터 분류를 사용하여 플래시 메모리 기반 저장 시스템의 성능을 향상시키기 위한 방법을 제안한다. 이를 위해 스토리지 가상화를 사용하여 Hot/Cold 데이터를 식별하고, Hot 데이터는 빠른 레벨의 저장장치에, Cold 데이터는 느린 레벨의 저장장치에 저장함으로써 전체적인 성능을 향상시켰다.

이러한 연구들은 Hot/Cold 데이터 분류를 사용하여 SSD의 성능을 향상시키는 다양한 방법을 제시하고 있다. 이를 통해 저장장치 용량을 더욱 효과적으로 활용할 수 있고, 더 높은 성능을 얻을 수 있다.

# **3 제약사항과 해결방안**

오픈소스로 주어진 동일한 데이터 (I/O trace)에 대해 Cold/Hot 데이터를 적절히 분류하여 학습한 모델의 성능과, 기존의 GC(Garbage Collection) 알고리즘을 이용한 기존 SSD의 성능을 비교할 것이다. 이를 위해서 SSD 에뮬레이터 혹은 실제 SSD를 이용하여 직접 데이터를 기록하고 성능을 평가해야 하는데, 실제 SSD를 이용하여 실험을 진행하는 것은 본 주제의 영역을 벗어나기 때문에, SSD 에뮬레이터를 이용한 가상환경에서 실험을 진행하고 분석할 것이다. 이를 위해서 우리는 우선 오픈소스로 주어지는 에뮬레이터에 우리가 학습시킨 모델을 적용할 수 있는지 확인할 것이다. 오픈소스로 주어지는 에뮬레이터에는 GC 알고리즘이 이미 구현이 되있을 것이므로, 모델을 적용할 수 있는지 여부만 확인하여 결과를 확인하면 된다. 만약 에뮬레이터에 우리가 만든 모델을 적용할 수 없다면, 직접 SSD 시뮬레이터를 구현하고, 이에 맞는 GC 알고리즘 또한 우리가 직접 작성해야할 것이다. 이렇게 직접 만든 SSD 시뮬레이터 환경이나, 오픈소스로 주어진 SSD 에뮬레이터 환경에서 동일한 I/O 트레이스에 대하여 우리가 만든 모델의 성능을 평가할 것이다.

# **4 시스템설계**

목적 : SSD의 Garbage Collection(GC) 과정 시 발생하는 Write Amplification(WA) 최소화

시스템 작동 방식

1. 머신러닝을 통해 데이터의 수정 빈도(Hot/Cold)를 학습한다.
2. 학습된 기준으로 데이터를 군집화하여 같은 군집을 같은 블록에 위치시킨다.
3. 수정이 잦은 데이터 같은 블록에 위치하여 WA 최소화 된다.

# **4.1 머신러닝 모델**

예측 모델 LSTM, 분류 모델 K-means

데이터가 Hot/Cold 중 어떤 종류인지 알면 지도학습을 통해 분류할 수 있겠지만, 우리의 실험에서는 모르기 때문에 비지도학습을 통해 군집화 해야 한다. 군집화의 대표 모델인 K-means 모델을 사용한다.

시계열 데이터 예측을 위해 여러 예측 알고리즘이 사용될 수 있지만, 대부분은 미래 를 예측할 때 과거 데이터만을 사용할 수 있다는 한계가 있다. 그러나 LSTM은 다양한 요소들을 종합적으로 고려하여 예측할 수 있기 때문에, 연구에서 발생하는 문제를 해결하기에 이상적인 모델로 여겨진다. LSTM은 장기적인 데이터 의존성을 학습할 수 있는 반복 신경망(RNN)의 한 종류로, 시계열 예측 작업에 효과적이라는 것이 입증됐다.

위의 모델들은 Google에서 제공하는 오픈소스 머신러닝 라이브러리 Tensorflow에서 사용 가능하다. 우리는 Tensorflow를 사용하여 K-means로 데이터를 군집화 하여 이를 LSTM 모델에 학습시켜 Hot/Cold 실시간으로 분류하여 SSD GC과정으로 인한 WA를 줄인다.

**4.1.1 K-means**

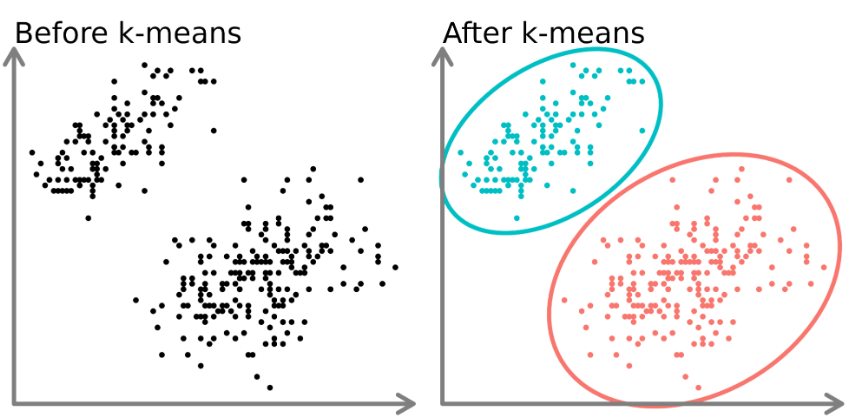


그림 2. K-means 적용

K-means 알고리즘

1. 군집의 개수(K) 설정
2. 초기 중심점 설정
3. 데이터를 군집에 할당
4. 중심점 갱신
5. 데이터를 군집에 재할당
6. 4,5 과정을 중심점의 위치가 고정될 때까지 반복

1단계에서 군집의 개수 설정을 어떻게 하냐에 따라 결과가 크게 달라지며 터무니 없는 결과가 나올 수도 있다. 그래서, 군집의 개수를 설정하는 방법론으로 몇 가지 방법들이 존재하는데, 우리는 Elbow Method를 사용한다.

Elbow Method : 하나씩 K를 증가시키고 가장 유의미한 성능 증가 지점을 K로 설정

# **4.1.1 RNN**

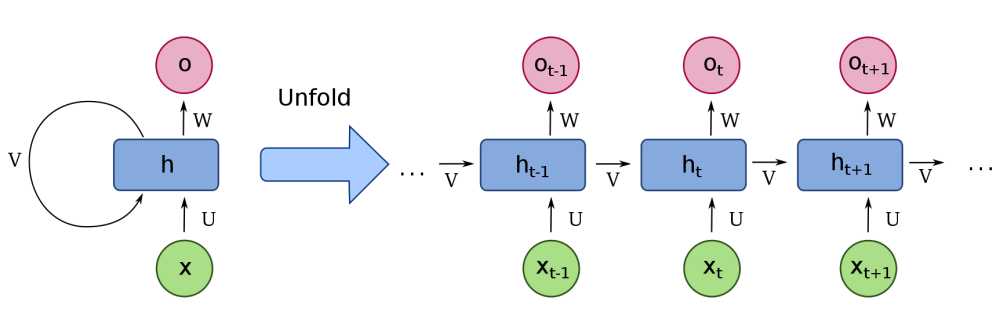
****

그림 3. RNN 구조

RNN은 이전 계층의 출력을 숨겨진 상태로 입력으로 사용할 수 있는 신경망이다. 예를 들 어 영화 장면마다 어떤 상황이 벌어지는지 분류해야 한다. 이때 현재 어떤 상황이 일어나는 지를 이해하려면 이전에 어떤 상황이 일어났는지를 이해해야 한다. 이렇듯 이전 데이터의 정보가 현재 정보를 처리하는 데에 필요한 상황에 RNN이 사용된다.

# **4.1.2 LSTM**

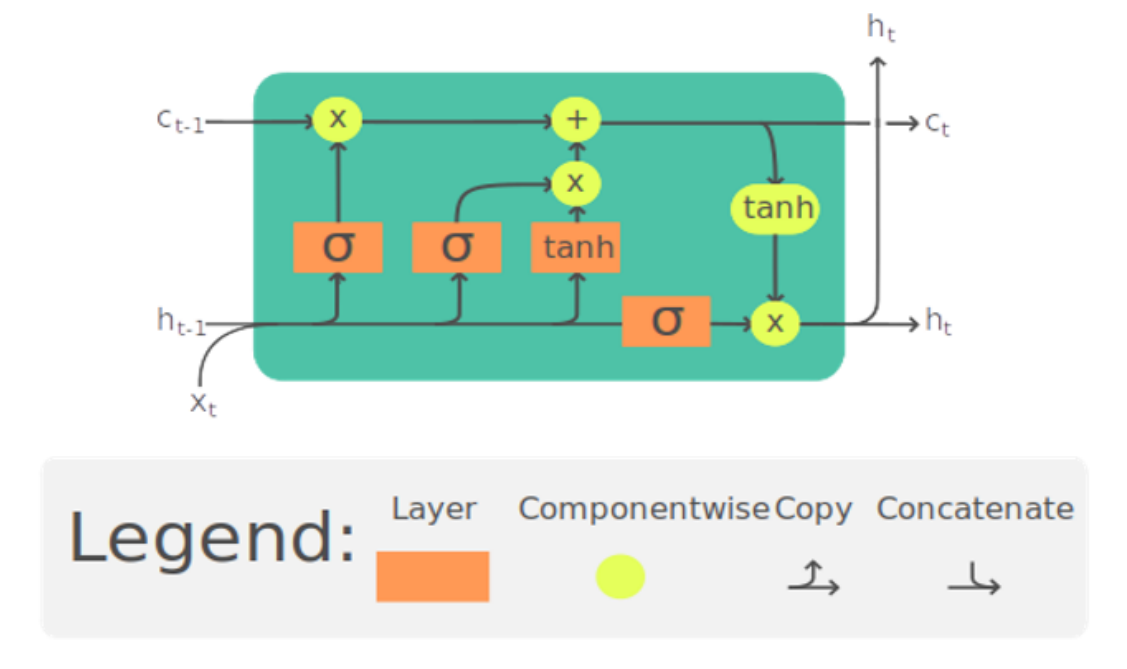


그림 4. LSTM 구조

LSTM은 정보를 더 긴 시간 동안 유지하는 메모리 셀을 가져 더 장기적인 관계성을 학습할 수 있다. 더하여 내부에 사용되는 게이트들이 입력된 정보가 메모리에 저장될지, 언제 출력될지, 언제 망각할지 결정한다. 이를 고려할 때 학습 데이터가 시간과 접근 빈도 등에 영향을 받으므 로 LSTM이 더 나은 선택이 될 것이라 예상한다.

# **4.2 학습데이터**

학습 데이터 (SSD I/O Trace) 제공 사이트

* SNIA(Storage Networking Industry Association)
* UMass Trace Repository

이들 데이터는 제공되는 데이터마다 조금씩 다르지만, 주로 다음 양식을 따른다.

0,20939840,8192,W,0.554041

각 필드는 쉼표로 구분된다.

첫 번째 필드는 I/O를 사용하는 애플리케이션 지정자이다.

두 번째는 논리적 블록 주소를 나타내며 세 번째는 레코드 사이즈를 표기한다.

네 번째는 OP코드로써 R(Read), W(Write)로 구성된다.

다섯 번째는 타임 스탬프로서, I/O 트레이스 시작점부터 경과 시간을 기록한 타임 스탬프이다. 이를 비교함으로 레코드 간 시간 차이를 확인할 수 있다

# **4.3 시뮬레이션**

위 과정들을 통해 얻은 모델을 평가하기 위하여 SSD 에뮬레이터를 사용할 것이다. 이때 실 제 SSD 제품에 사용되는 GC 알고리즘을 에뮬레이터에 적용할 것이며, 필요에 따라 직접 GC 알고리즘 시뮬레이터를 작성하여 실험할 것이다.

| 이름 | 개발 및 제공 기관 |
| --- | --- |
| FEMU | Virginia Tech |
| VSSIM | KAIST OS Lab |
| SimpleSSD | KAIST CAMELab |
| SSD Extension for DiskSim Simulation Environment | Microsoft |

표 2. 사용 가능한 SSD 시뮬레이터

이를 통해, 평가 기준 알고리즘과 기계 학습을 통해 얻어낸 모델의 성능을 평가하고 비교하여 개선된 성능을 얻어내고자 한다.

모두 GC 알고리즘을 변경 가능하며, 오픈소스 소프트웨어로 별도의 비용 없이 사용이 가 능하다. FEMU는 C언어로 작성되어 있다. 또한 DRAM을 기반으로 작동하기 때문에 실험상 대용량 저장장치에 접근함에 따라 발생하는 추가적인 딜레이가 발생하지 않게 된다. VSSIM 역 시 C언어로 이루어져 있다. 해당 에뮬레이터는 가상머신으로부터 실시간 워크로드를 받아 사 용하여 실제 SSD 시스템 환경과 동일한 환경에서 실험을 진행할 수 있게 해준다. SIMPLESSD 는 C++로 구성된다. 3가지 에뮬레이터 모두 풀 시스템 SSD 에뮬레이터이고 높은 성능을 보 여주기 때문에 사용환경, 접근성, 구성 프로그래밍 언어 등을 고려하여 선정할 필요가 있다.

# **5 개발 일정 및 업무 분담**

# **5.1 개발 일정**

개발 일정은 아래와 같다.

| **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** | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | **머신러닝 모델 개발** | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  | **신경망 모델 개발** | | | | |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  | **모델 학습 및 결과 비교** | | | |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  | **중간보고서 준비** | | |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **모델 개량** | | | | | | |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **최종 발표/보고서 준비** | | | |

# 

# **5.2 업무 분담**

업무 분담은 아래와 같다.

공희찬 : 데이터 가공 및 Hot/Cold 데이터 분류

박선민 : 머신러닝 모델 개발 및 성능 평가

박영훈 : 신경망 모델 개발 및 에뮬레이터 적용

공통 : 머신러닝 모델 개량 및 자료조사

[1] 주광우, 한재현, 이상진, 이은서, 손용석. SSD를 위한 병렬적 가비지 컬렉션 기법. 한국정보과학회. 2020 Jul;1179–1181.

[2] Kristian Vättö . Samsung SSD 840 EVO mSATA (120GB, 250GB, 500GB & 1TB) Review [Internet]. ANANDTECT. 2014 [cited 2023 May 10]. Available from: https://www.anandtech.com/show/7594/samsung-ssd-840-evo-msata-120gb-250gb-500gb-1tb-review

[3] Samsung Electronics Co. K9XXG08UXM flash memory data sheet, 2007.

[4] dataonair. DATA ON-AIR. 2015 [cited 2023 May 20]. Available from: https://dataonair.or.kr/db-tech-reference/d-lounge/expert-column/?mod=document&uid=52346