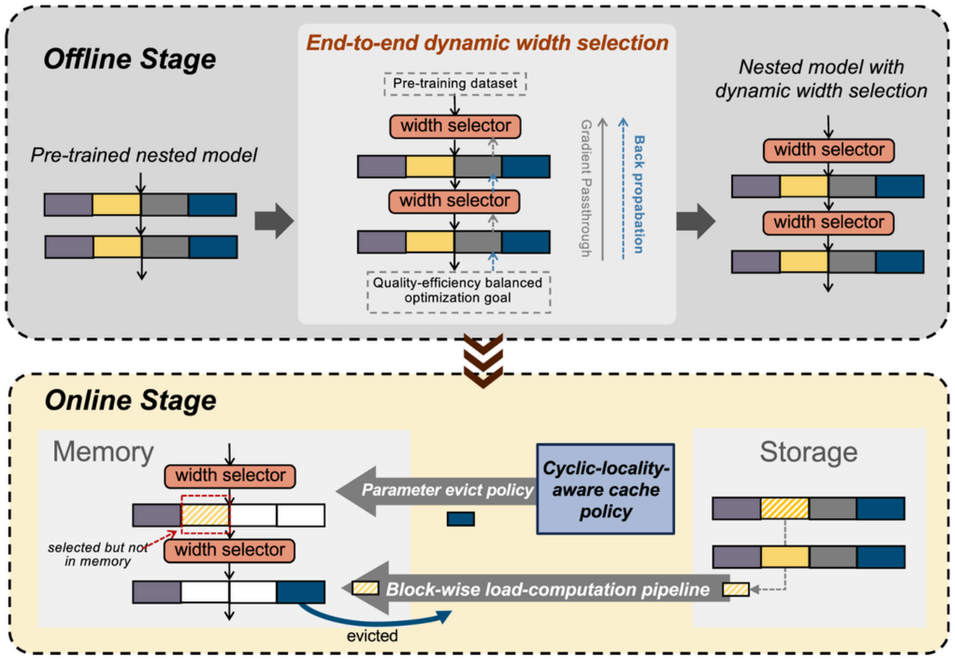
2026.11.20

DWI
: Efficient On-Device LLM Decoding Through Dynamic Width Selection on Nested Models

중첩 모델의 동적 폭 선택을 활용한 온디바이스 LLM 디코딩 효율화



Chen, W., et al. “DWI: Efficient On-Device LLM Decoding Through Dynamic Width Selection on Nested Models.” IEEE Transactions on Mobile Computing, vol. 25, no. 11, pp. 19763–19777, 2026.

Presented by :
학부연구생 김승겸

Contents

01 **서론**
온디바이스 실행과 메모리 제약

02 **배경과 동기**
동적 희소성과 중첩 모델

03 **DWI 설계**
폭 선택, 캐시, 파이프라인

04 **구현**

05 **실험 결과**
품질, 속도, I/O 에너지

06 **관련 연구**

07 **결론**

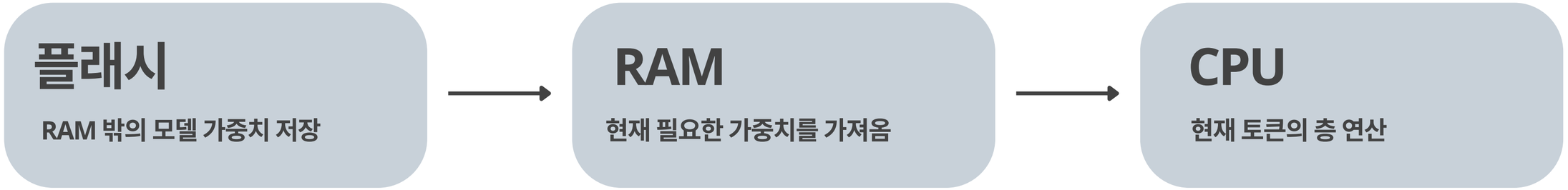
온디바이스 LLM의 필요성과 제약

기기 안에서 LLM을 실행하려는 이유?

- 외부 서버와의 통신 부담 감소
- 불안정한 네트워크 연결 환경에서도 사용 가능
- 기기 내부에서 민감한 데이터 처리
- 스마트폰은 메모리를 OS와 앱이 함께 사용

→ 전체 모델이 RAM에 들어가기 어려운 환경에서 on-flash 실행을 고려한다

On-flash 디코딩과 가중치 로딩



가중치 접근이 디코딩 지연을 유발한다

- 필요한 모델 가중치를 플래시에서 RAM으로 읽는다
- 디코딩을 통해 다음 토큰을 만들 때마다 모델의 여러 층을 통과 → 각 층의 가중치를 반복해서 사용
- 디코딩 중 반복적인 가중치 로딩이 지연을 유발

→ DWI는 토큰 생성 중 반복되는 가중치 로딩 비용을 줄임

단일 요청 디코딩의 반복 접근

토큰마다 층을 순환

현재 토큰에서 첫 층부터 마지막 층까지 실행하고,
다음 토큰에서 같은 순서를 반복

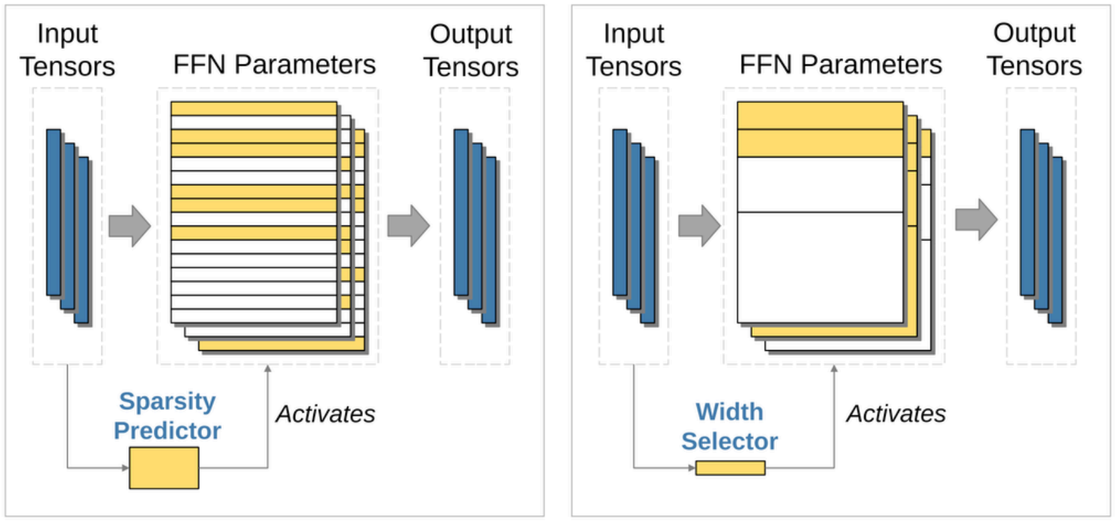
제한된 가중치 캐시

전체 가중치를 보관하지 못하면 층을 재방문할 때
플래시 접근이 발생한다

성능을 결정하는 공급 비용

CPU 계산 능력과 함께, 가중치 접근량 및 저장장치에서 가져오는 시간을 평가

동적 희소성 & 작은 로딩 단위



(a) Scattered Dynamic Sparsity (b) Dynamic Width Selection

Fig. 1. An illustration of previous scattered dynamic sparsity mechanism and our proposed dynamic width selection mechanism in Feed Forward Network.

→ DWI는 입력별 선택을 큰 연속 가중치 구간으로 연결

동적 희소성

입력 특성에 따라 일부 뉴런, 헤드만 활성화하여 계산량과 파라미터 접근량을 감소

기존 방식의 조각 규모

약 2k-4k개 파라미터를 포함하는 작은 텐서 조각을 선택
파라미터 개수와 바이트는 구분

Colocation과 양자화

Colocation 적용시 float16에서 약 12-24KB,
양자화 시 약 3-6KB 또는 그 이하

플래시와의 불일치

흩어진 작은 읽기로는 큰 블록 읽기에 유리한 플래시 대역폭을 충분히 쓰기 어렵다

블록 크기에 따른 플래시 처리량

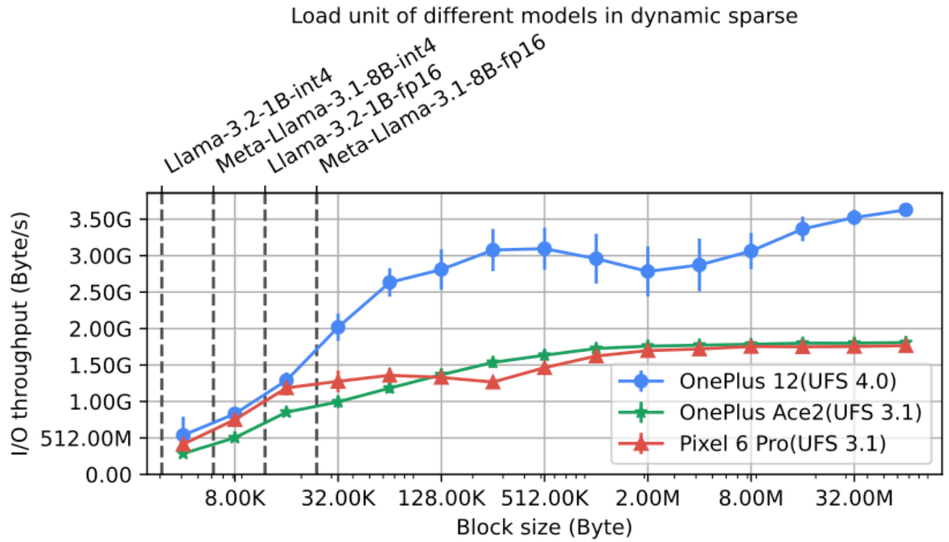


Fig. 2. The achieved flash throughput under random read access, using different block sizes on smartphones. Each data point represents the mean and standard deviation across three runs. Vertical dashed lines mark the typical load-unit sizes of dynamic sparse methods for each listed model, with colocation neuron optimization applied.

→ 가중치 접근량과 읽기 단위의 크기를 함께 최적화

측정

스마트폰 플래시의 랜덤 읽기를 3회 측정 후
평균과 편차 제시

논문에서의 대표 비교

12KB 블록에서 약 1GB/s, 32MB 블록에서 약 3GB/s의 처리량을 비교

총 바이트 외의 요인

읽는 양이 같아도 작은 조각을 반복 접근하는지,
큰 블록으로 읽는지에 따라 시간이 달라짐

양자화와의 관계

가중치 크기가 줄어들면서 읽기 단위도 작아지면 플래시
특성과의 불일치가 커질 수 있다

중첩 모델의 폭과 연속 접두부

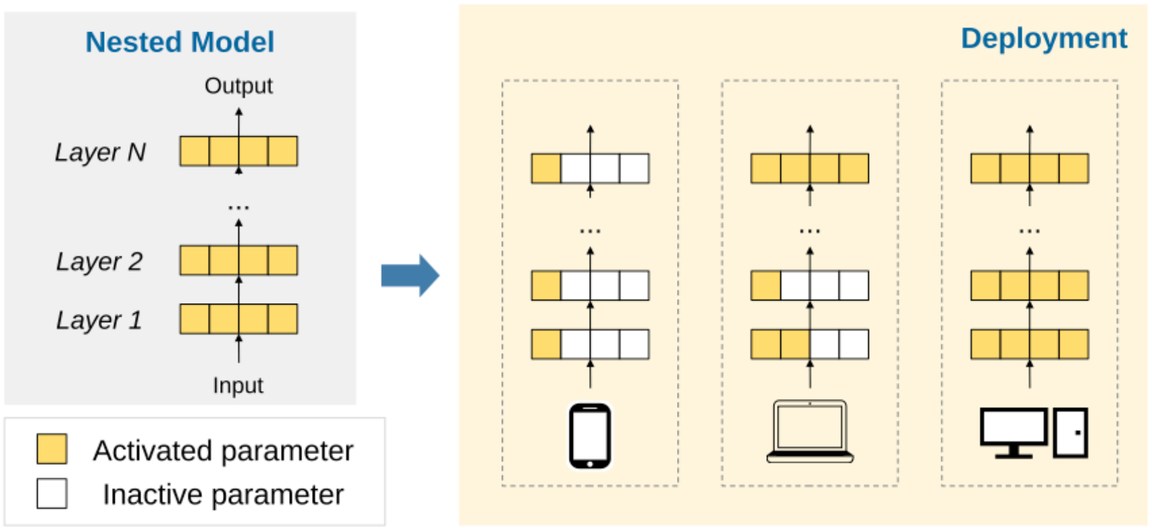


Fig. 3. Nested models. (Left) Nested weight structure. Critical parameters are concentrated at the start indices, enforced by the joint training loss. (Right) Elastic deployment. A nested model allows devices with varying capabilities (e.g., phone vs server) to simply slice effective sub-models from the same weight file without retraining.

- 중첩 모델: 작은 하위 모델의 가중치가 큰 모델의 앞부분에 포함되도록 구성된 모델

폭의 의미

FFN은 활성 중간 뉴런 수, Attention, Mamba는 활성 헤드 수로 폭을 조절

접두부 포함 관계

작은 하위 모델의 가중치는 큰 모델의 앞쪽 채널 및 헤드 구간에 포함

폭을 늘리는 방법

공통 앞부분을 유지하면서 인접한 추가 가중치 구간을 사용

중첩 학습의 필요성

다양한 폭을 함께 학습하여 작은 하위 모델도 유효한 출력을 만듦

중첩 모델의 학습을 위한 손실함수

$$L_{\text{total}} = \sum \alpha_i L_{\text{width}_i}$$

L_{width_i}

폭 i의 하위 모델을 사용했을 때 학습 손실

α_i

해당 손실에 부여하는 가중치

앞쪽 가중치의 역할

작은 폭도 함께 최적화하여 중요한 의미 정보가 낮은 인덱스에 집중되도록 유도

- 중첩 모델 자체의 학습 원리

오프라인 학습과 온라인 실행

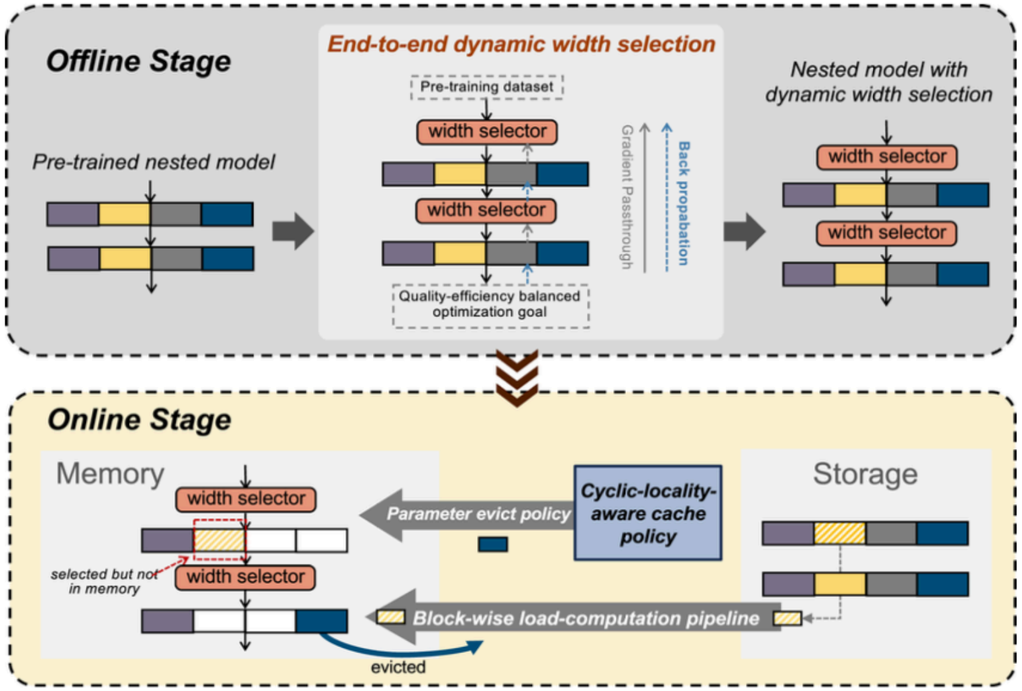


Fig. 4. System overview of DWI, consisting of an offline end-to-end width-selector training stage and an online inference stage with cyclic-locality-aware caching policy and block-wise load-computation pipeline.

오프라인

각 층에 폭 선택기를 넣고 최종 출력 품질과 활성화 비율을 함께 최적화

온라인 폭 선택

현재 은닉 상태로 토큰별·층별 폭을 결정
사용 폭은 입력에 따라 달라짐

가중치 관리

필요한 블록의 캐시 존재 여부를 확인하고, 부족한 공간을 확보하여 추가 블록을 읽음

실행 중첩

RAM의 활성 블록을 계산하는 동안 없는 블록을 비동기로 로딩

폭 선택 확률과 실제 실행 폭

$$\mathbf{g}_l = [g_{l,1}, g_{l,2}, g_{l,3}, \ldots, g_{l,n}], \quad \sum_{i=1}^{n} g_{l,i} = 1. \tag{2}$$

$$\mathbf{g}_{l,\text{hard}} = \text{onehot}(\text{argmax}(\mathbf{g}_l)), \tag{3}$$

g_l

후보 확률, 현재 은닉 상태를 선형 계층과 Softmax에 넣어
후보 폭의 확률을 얻음. 확률의 합은 1

실험의 네 후보

전체 폭, 1/2, 1/4, 1/8 중 하나 선택

argmax와 onehot

가장 높은 확률의 후보 위치를 찾고, 해당 위치만 1인 벡터로 변환

선택 단위

각 토큰의 각 층에서 하나의 폭을 결정. 모든 층에 같은 폭을 강제하지 않음

선택 마스크와 층 출력

$$\mathbf{O}_l = \sum_{i=1}^{n} \left(\mathbf{O}_{l,i} \cdot g_{l,\text{hard},i} \right), \qquad (4)$$

$O_{l,i}$

층 l을 후보 폭 i로 실행했을 때의 출력

$g_{l,hard,i}$

선택한 폭에서만 1이고, 나머지는 0인 선택 마스크

식(4)의 합산 의미

마스크가 1인 항만 남으므로 선택된 폭의 출력이 최종 출력이 됨

추론 시 실행

선택하지 않은 폭의 연산은 생략

Gradient Passthrough와 선택기 학습

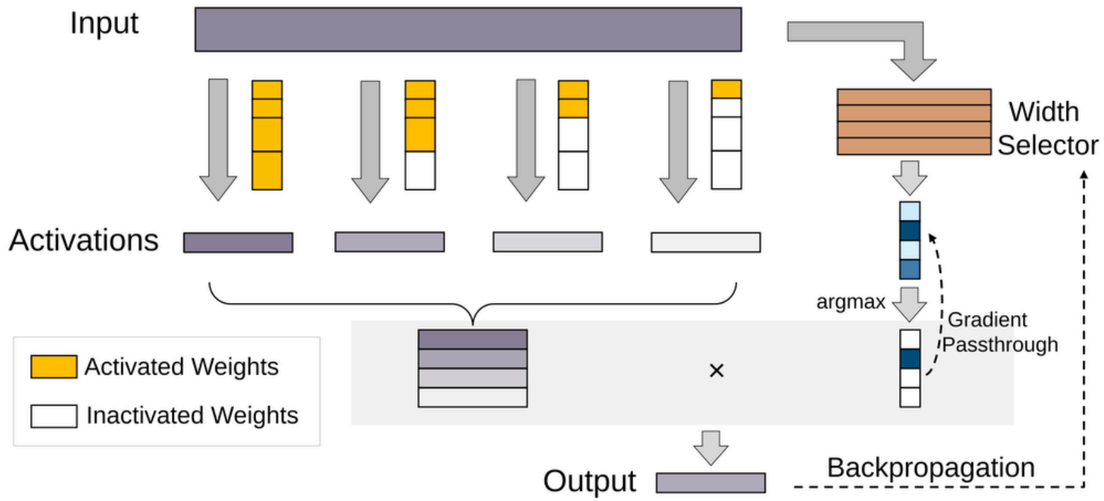


Fig. 5. End-to-end width-selector training with gradient passthrough method.

이산 선택의 문제

argmax·onehot은 일반적인 기울기 전달을 막아
폭 선택기에 학습 신호를 전달할 수 없음

순전파

하나의 폭을 고르는 hard 선택과 활성화 마스킹을 유지

역전파

$g_{l,hard,i}$ 의 기울기를 연속 확률 g_l 에 직접 전달.
STE 개념을 적용

직접 학습

출력 품질을 예측하는 별도의 surrogate model 없이
최종 출력 손실로 선택기를 학습한다.

- STE(Straight-Through Estimator: 이산 선택을 학습하기 위한 기울기 추정 방법

품질·효율 손실의 결합

$$Loss = Loss_{\mathrm{q}} + \lambda \cdot Loss_{\mathrm{e}} \tag{5}$$

$$Loss_{\mathrm{q}} = Loss_{\mathrm{CE}}\left(Output_{\mathrm{model}}, Label_{\mathrm{dataset}}\right) \tag{6}$$

$Loss_q$

품질 손실, 모델 출력과 데이터셋 정답 사이의 교차 엔트로피를 사용

$Loss_e$

효율 손실, 평균 활성화 비율이 목표를 초과할 때 벌점 부여

λ

효율 손실의 가중치,
효율 벌점을 전체 학습 손실에 얼마나 강하게 반영할지 결정

공동 최적화의 목적

큰 폭만 사용하는 선택과 품질을 과도하게 희생하는 선택
사이의 균형을 학습

평균 활성화 비율에 대한 효율 벌점

$$Loss_{\mathrm{e}} = \mathrm{ReLU}\left(\mathrm{Mean}_{i\in\mathrm{Tokens},j\in\mathrm{Layers}}\left(\frac{width_{i,j}}{fullwidth_j}\right) - \tau\right) \tag{7}$$

폭의 정규화

$\frac{width_{i,j}}{fullwidth_j}$ 는 토큰 i, 층 j에서 전체 폭 중 사용한 비율

토큰·층 전체의 평균

위치마다 폭을 다르게 배분하되 전체 평균 활성화 수준을 조절

τ와 ReLU

평균이 tau 보다 높으면 양의 벌점 생성
목표 이하에서는 효율 손실이 0

지표의 범위

tau는 평균 활성화 비율의 기준
각 층의 고정 폭이나 전체 RAM 사용률이 아님

후보 폭의 경계와 가중치 블록

$$\begin{cases} b_1 = [0, w_1] \\ b_i = [w_{i-1}, w_i] \text{ when } i > 1 \end{cases} \quad (8)$$

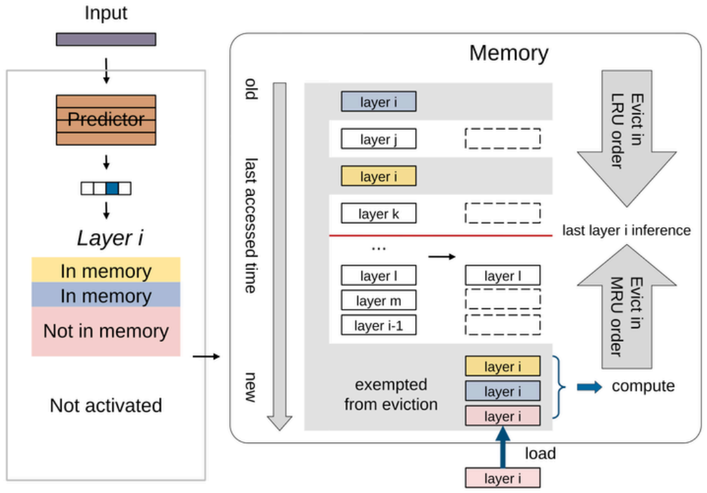


Fig. 6. Cyclic-locality-aware cache policy, which prioritizes cold data not accessed since the last inference of layer i. When not available, it prioritizes the newest data.

가중치 블록 구간

b_1은 시작부터 w_1까지
이후 b_i는 이전 후보 폭과 현재 후보 폭 사이의 추가 구간

선택 폭과 필요 블록

폭 w_i를 실행하려면 앞쪽 블록 b_1부터 b_i까지 필요

부분 재사용

선택한 폭에 필요한 블록 중 RAM에 있는 블록은 재사용하고,
RAM에 없는 블록만 플래시에서 로딩

가중치 캐시

모델 파라미터 블록을 보관
Attention의 KV 상태 저장 공간과 역할이 다름

순환 지역성을 고려한 캐시 퇴출

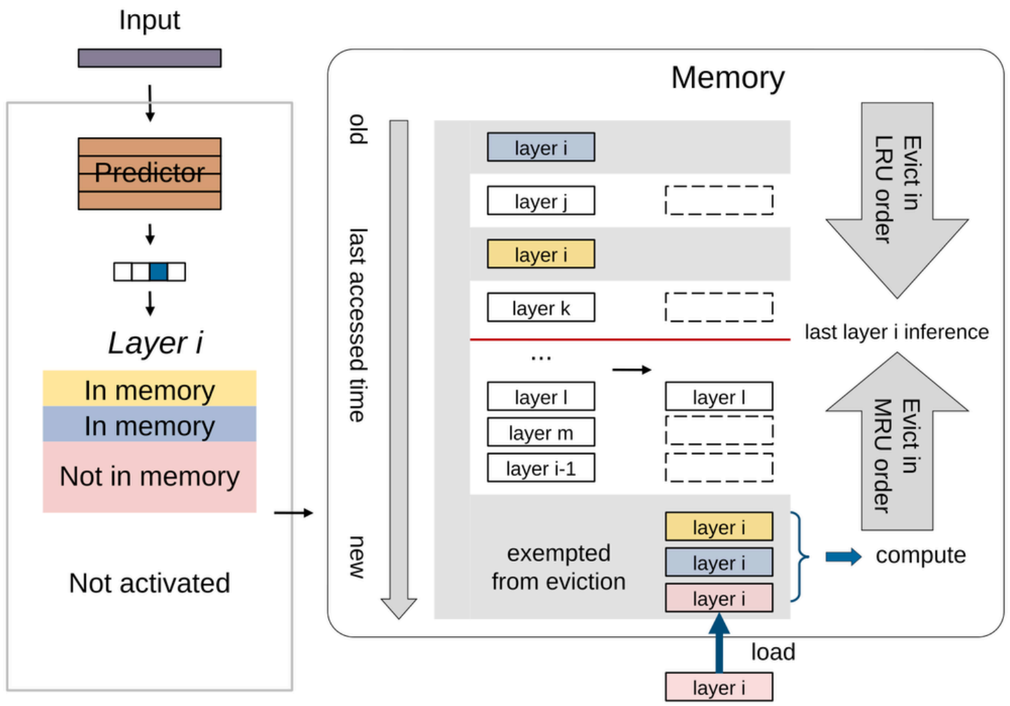


Fig. 6. Cyclic-locality-aware cache policy, which prioritizes cold data not accessed since the last inference of layer i. When not available, it prioritizes the newest data.

층의 순환 순서

토큰마다 층을 같은 순서로 재방문하므로
방금 쓴 층이 즉시 다시 필요하지 않음

cold 블록 우선 퇴출

현재 층의 직전 방문 이후 접근하지 않은 cold 블록 중
가장 오래된 블록을 퇴출

cold가 없는 경우

가장 최근에 접근한 블록을 퇴출하여 순환 순서상
가까운 재사용을 고려

현재 연산 보호

현재 층의 블록은 퇴출 대상에서 제외

블록별 로딩·연산 파이프라인

$$\mathbf{o} = \sum_{i=1}^{n} \mathbf{o}_i + \mathbf{c} \tag{9}$$

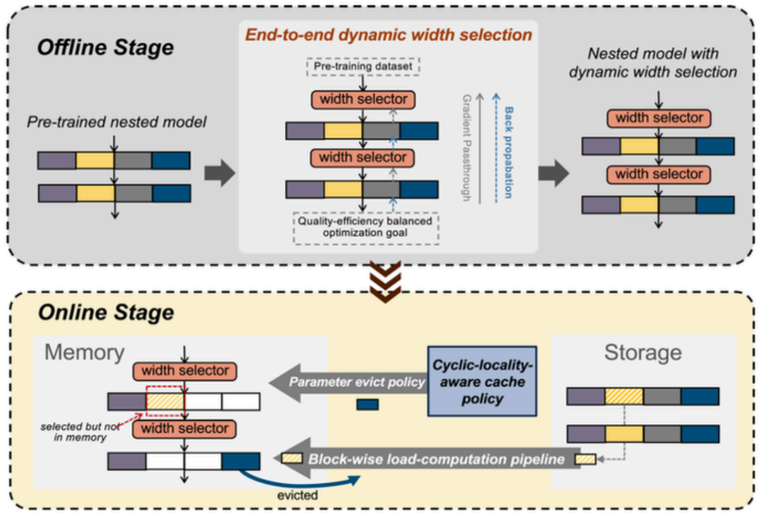


Fig. 4. System overview of DWI, consisting of an offline end-to-end width-selector training stage and an online inference stage with cyclic-locality-aware caching policy and block-wise load-computation pipeline.

준비된 블록부터 계산

RAM에 있는 활성 블록의 연산을 먼저 실행

부족한 블록은 비동기 로딩

계산과 동시에 없는 활성 블록을 플래시에서 로딩

o_i와 C

o_i는 블록별 중간 출력
합산 후 편향 값이 존재할 때 C를 한 번 더 더함

중첩 실행의 범위

적용 가능한 블록 분해를 활용
로딩이 끝나지 않은 필요 블록에서는 대기가 발생

비활성 블록의 상태 갱신과 Backfill

$$\mathbf{State}_{b,j} = \mathrm{Backfill}(\mathbf{State}_{b,i}, \{\mathbf{Input}_k \mid i < k \leq j\}) \quad (10)$$

- b : 갱신할 블록
- i : 갱신할 블록의 상태를 마지막으로 갱신한 토큰 위치
- j : 블록이 다시 선택된 현재 토큰 위치

예시: 토큰 3이 마지막 갱신 시점, 토큰 6에서 다시 활성화

1. 토큰 4,5: 블록의 상태 갱신을 미루고 버퍼에 저장됨
2. 토큰 6: 토큰 3의 상태에서 시작해, 보관한 입력 4·5와 현재 입력 6을 순서대로 반영
3. 결과: 토큰 4·5·6에서 상태를 매번 갱신했을 때와 같은 토큰 6의 상태를 얻음

갱신을 생략하면

블록이 선택되지 않으면 가중치를 읽지 않고 상태 갱신을 미룸
Attention의 KV 캐시나 Mamba의 상태에 해당 토큰의 정보가 아직 반영 되지 않음

입력 보관

나중에 상태를 갱신할 수있도록, 블록이 비활성인 동안 들어온 입력을 버퍼에 저장

Backfill

마지막 상태 $State_{b,i}$에 누락된 입력을 적용해 $State_{b,j}$ 까지 상태 전이를 보충

추가 비용

일반 Backfill은 현재 토큰용 가중치를 재사용해 추가 가중치 읽기가 없다

밀린 상태 갱신의 한도와 강제 동기화

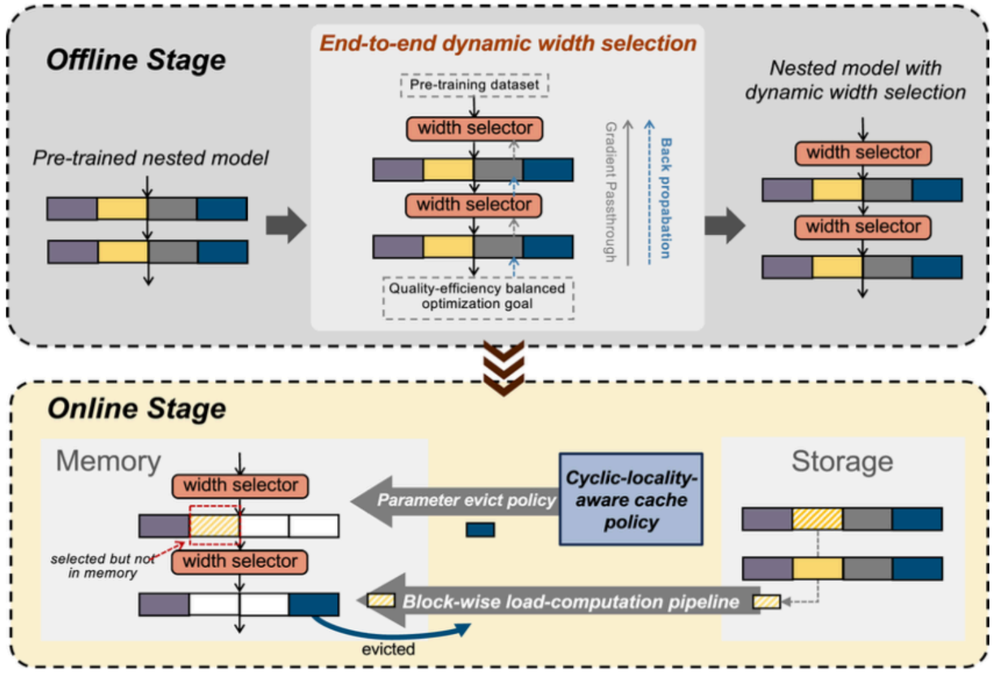


Fig. 4. System overview of DWI, consisting of an offline end-to-end width-selector training stage and an online inference stage with cyclic-locality-aware caching policy and block-wise load-computation pipeline.

N_{lim} : 동기화 임계값

누락 입력과 보충 작업이 무한히 커지지 않도록
전체 폭의 상태를 동기화

입력 버퍼 크기 상한

$N_{lim} \times d_{hidden} \times N_{layer}$개 원소 저장
보관할 토큰 수 x 토큰당 입력 벡터 크기 x 층 수

전체 폭 갱신

한도에 도달하면 전체 폭을 사용해,
아직 갱신이 밀려 있는 블록까지 현재 시점으로 동기화

오래된 입력 해제

한 구성요소의 모든 블록에 특정 토큰까지의 입력이 반영되면,
그 토큰까지의 입력을 버퍼에서 지울 수 있음

llama.cpp 기반 CPU 구현

지원 모델 및 실행 환경

- MatFormer-OLMo 1.3B와 MatMamba-LM 1.4B
- Arm CPU 기반 프로토타입

조건부 실행 노드

연산을 블록 단위로 나누어 선택한 경우에만
해당 노드를 실행

가중치 관리

- libaio로 필요한 가중치를 비동기 사전 로딩
- 계산 시점에 준비되지 않은 블록만 로딩 완료까지 대기
- 메모리 예산을 초과하면 캐시 정책에 따라 퇴출

모델별 동기화

MatMamba-LM의 배치 정규화 구조에서는
블록 출력 지점에서 결과를 모으기 전에
정규화 지점에서 블록 연산을 동기화

모델과 데이터셋

TABLE I
MODELS AND DATASETS USED IN OUR EXPERIMENTS

Model	Size	Architecture	Nested Component	Datasets
MatFormer-OLMo-1.3B	0.71B	Transformer	FFN	HellaSwag [42], XWinograd [43], PIQA [44], ARC (Easy Set and Challenge Set) [45], OpenBookQA [46], GLUE (RTE) [47], SuperGLUE (BoolQ, COPA) [48]
MatMamba-LM-1.4B	1.34B	Mamba-2	Mamba-2 block	WebQuestions [49], LAMBADA [50], Winogrande, XWinograd, PIQA, ARC (Easy Set and Challenge Set), OpenBookQA, SuperGLUE (COPA)

MatFormer-OLMo

- FFN의 중첩 폭을 조절
- HellaSwag·PIQA·ARC·OpenBookQA 등으로 평가

MatMamba-LM

- Mamba-2의 중첩 폭을 조절
- WebQuestions·LAMBADA·PIQA·ARC 등

평가 방법

LM Evaluation Harness의 one-shot 설정, 각 모델에 지정한 과제 집합에서 평균 품질을 비교

DWI / 실험 설정

DWI 실험에서 사용한 데이터셋

폭 선택기 학습 **C4** 정제한 웹 텍스트 · 기존 모델의 가중치는 고정하고 폭 선택기만 학습

정확도 평가

예시 1개를 보여준 뒤 평가하는 one-shot 설정

데이터셋	어떤 문제를 푸나	평가 모델
HellaSwag	일상 상황에 가장 자연스럽게 이어지는 내용 고르기	MatFormer-OLMo
WinoGrande	문맥과 상식으로 빈칸에 들어갈 사람이나 사물 고르기	MatMamba-LM
XWinograd	여러 언어에서 대명사가 가리키는 대상 판단하기	**두 모델 모두**
PIQA	일상 목표를 이루는 두 방법 중 물리적으로 적절한 방법 고르기	**두 모델 모두**
ARC Easy / Challenge	초·중등 수준의 과학 객관식 문제 풀기	**두 모델 모두**
OpenBookQA	기초 과학 사실과 상식을 결합해 정답 고르기	**두 모델 모두**
RTE (GLUE)	앞 문장이 참일 때 뒤 문장도 참인지 판단하기	MatFormer-OLMo
BoolQ (SuperGLUE)	지문을 읽고 질문에 예 또는 아니요로 답하기	MatFormer-OLMo
COPA (SuperGLUE)	사건의 원인이나 결과로 더 그럴듯한 선택지 고르기	**두 모델 모두**
WebQuestions	사실 질문에 짧게 답하기 · 정답과 일치하는지 평가	MatMamba-LM
LAMBADA	앞 문맥을 바탕으로 글의 마지막 단어 예측하기	MatMamba-LM

디코딩 속도 측정

PIQA · OpenBookQA · ARC-Challenge에서 뽑은 프롬프트로 토큰당 평균 생성 시간 측정

스마트폰은 프롬프트 18개 × 최대 100토큰 · Raspberry Pi는 3개 × 최대 20토큰

출처 Chen et al., DWI, Section V-A 및 Table I

폭 선택기 학습 설정

TABLE II
TRAINING SETTINGS

Model	Steps	Tokens	λ	Learning rate	Training time
MatFormer-OLMo	100	$\leq$ 6.55M	1	1×10^{-3} [a]	$\approx$ 0.5 hour
MatMamba-LM	500	$\leq$ 32.8M	10	1×10^{-2}	$\approx$ 1.5 hours

[a] Used with a cosine annealing scheduler, configured with a period of 1000 steps. Since we only run the first 100 steps, the learning rate remains close to the initial value throughout training.

기존 모델 고정

- 데이터 C4로 선택기만 학습
- 각 선택기는 은닉 차원 2048 x 너비 옵션 4의 선형 계층
- 폭 후보는 4개(전체, 절반, 1/4, 1/8)

선택기 규모

- MatFormer: 8개, 약 64K 파라미터
- MatMamba: 48개, 약 384K 파라미터

학습 환경

- 단일 A800 80GB, 시퀀스 길이 2048
- AdamW 사용, 유효 배치 크기 32
- 각각 100·500 스텝
- 약 0.5·1.5시간

추론 기기 및 가중치 캐시 예산

TABLE III
THE EDGE DEVICES USED IN OUR EXPERIMENTS

Device	SoC	Memory	Storage
OnePlus 12	Snapdragon 8 Gen 3	24GB	UFS 4.0
OnePlus Ace2	Snapdragon 8+ Gen 1	16GB	UFS 3.1
Pixel 6 Pro	Google Tensor	12GB	UFS 3.1
Raspberry Pi 4B	Broadcom BCM2711	4GB	microSD UHS-I

CPU 추론 환경

- 스마트폰 3종, 라즈베리 파이 4B
- Oneplus 12는 6스레드, 나머지는 4스레드 사용

메모리 예산 적용 범위

- FFN과 Mamba-2 블록 가중치에만 RAM 예산 적용
- 폭 선택기 등 별도로 상주하는 구성요소는 메모리 제한 X

디코딩 속도 측정

- PIQA·OpenBookQA·ARC-Challenge 에서 프롬프트를 추출해 토큰당 평균 생성 시간 측정
- 스마트폰: 18개 프롬프트, 최대 100토큰
- 라즈베리파이: 3개 프롬프트, 최대 20 토큰

비교군과 평가 지표

Scattered Sparsity

- 입력에 따라 흩어진 가중치를 선택하는 방식
- Deja Vu 예측기와 희소 연산을 이용한 비교군

With Bundling

함께 활성화하는 가중치를 묶어 읽기 단위를 개선

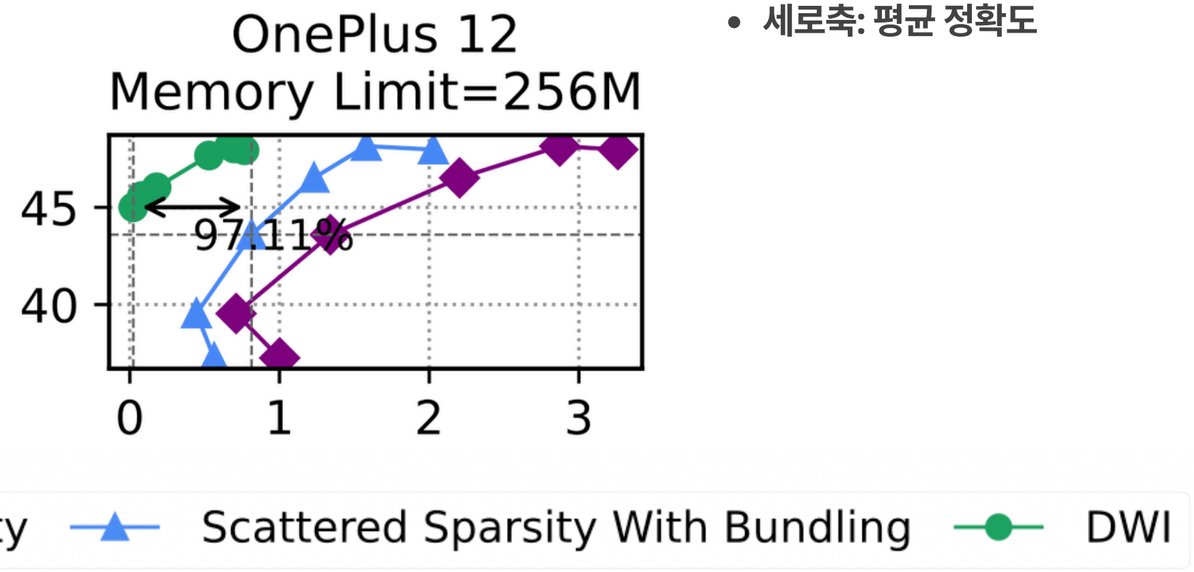
품질 분석 비교군

- Fixed Width는 일정한 폭 사용
- CATS: 활성화 통계에 기반한 임계 방식 사용

평가 지표

정확도, 활성화 비율, 토큰당 시간, 실제 로딩 바이트,
I/O 처리량, 토큰당 에너지

같은 정확도에서 최대 34배 가속



- 가로축: 토큰 당 생성 시간(초)
- 세로축: 평균 정확도

모델과 기기

- MatFormer-OLMo, OnePlus 12

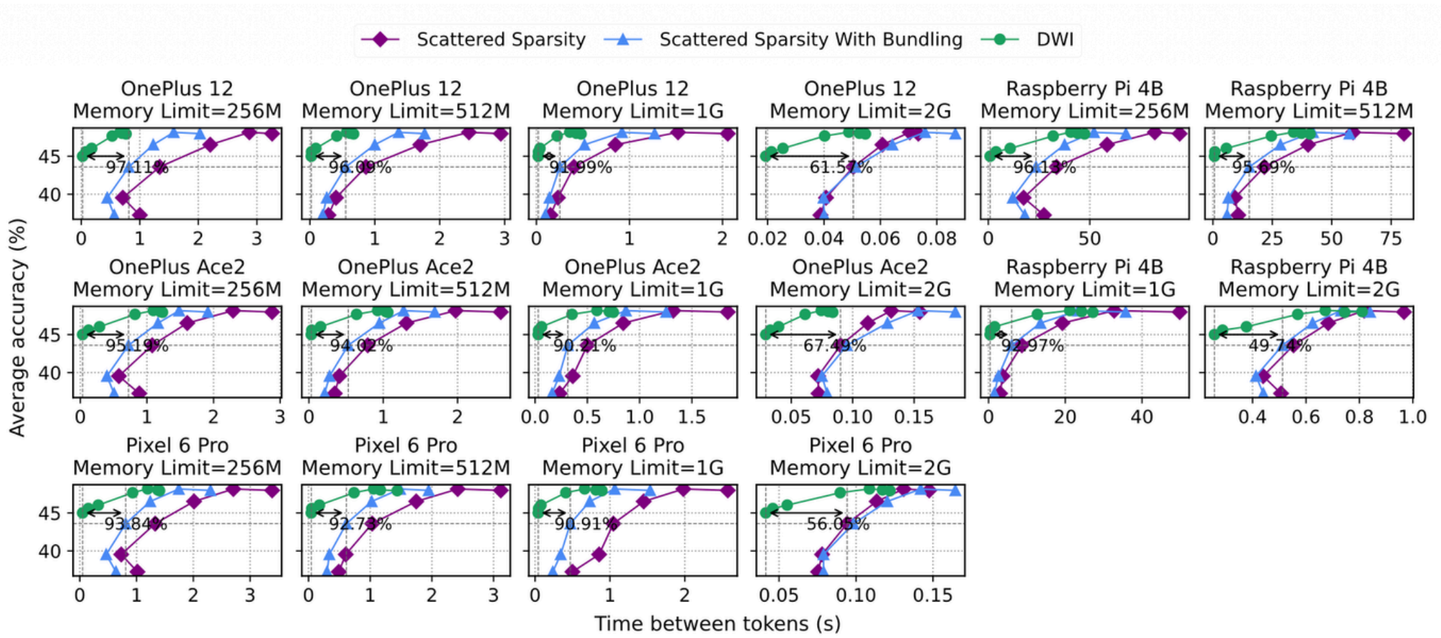
메모리 예산

- 중첩 구성요소 가중치 캐시 256MB

비교 대상

- Scattered Sparsity with Bundling

기기·예산에 따른 MatFormer 결과



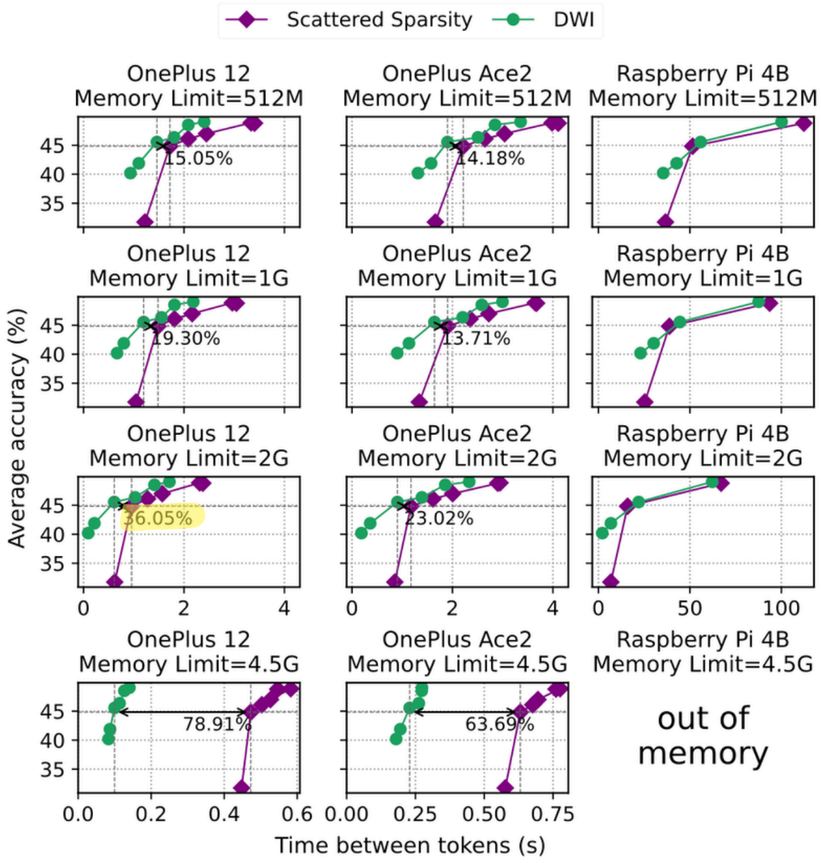
예산 영향

- 예산이 커지면 가중치 재사용이 증가

결과

MatFormer-OLMo에서 256 MB 조건의 토큰당 생성 시간은
네 기기에서 약 94~97% 감소했고, 2 GB에서는 약 50~67% 감소

MatMamba-LM의 품질 속도



대표 결과

OnePlus 12, 2GB 예산에서
비슷한 평균 정확도에서 토큰당 생성 시간 36.05% 감소
Scattered Sparsity 대비 약 1.5배 가속

MatFormer와의 차이

Mamba-2는 가중치 읽기 단위가 이미 커서 Bundling 없이 비교

기기와 예산에 다른 차이

모델 구조와 기존 로딩 단위가 달라지면
같은 활성화 비율에서도 가속 효과가 달라진다

평균 활성화량과 정확도

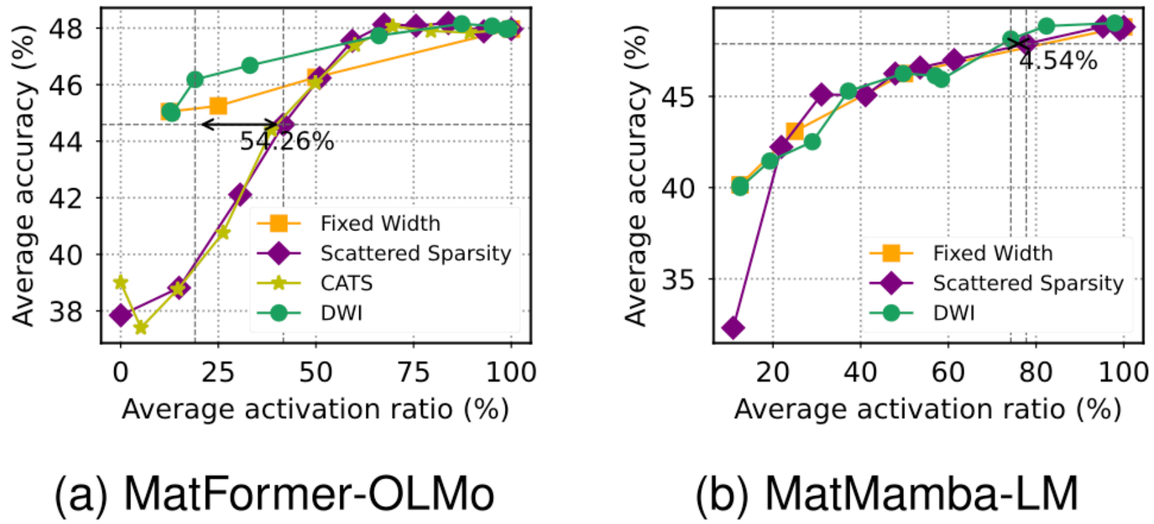


Fig. 9. Average accuracy and activation ratio across downstream tasks for DWI and baselines under diverse sparsity configurations, measured on MatFormer-OLMo and MatMamba-LM. DWI shows similar or higher accuracy at the same activation ratio, compared to Fixed Width and Scattered Sparsity.

동적 선택과 고정 폭

같은 활성화 비율에서 DWI는 비교군과 비슷하거나 더 높은 정확도를 보임
MatFormer-OLMo에서는 Fixed Width 대비 개선 경향이 뚜렷함

폭의 배분 차이

DWI는 토큰·층별로 폭을 배분
고정 폭은 모든 입력에 같은 폭 적용

낮은 활성화 구간

MatFormer에서 scattered·CATS는 50% 아래에서
급격한 품질 저하 발생

중첩 모델의 절충

작은 폭에서 더 완만한 정확도 곡선을 보이지만,
품질 손실 가능성은 남아있다.

활성화 비율과 실제 로딩 바이트

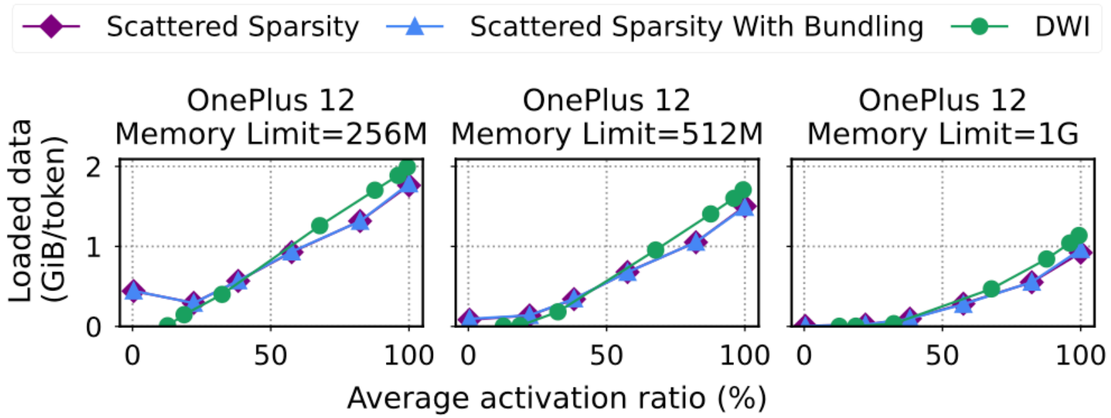


Fig. 10. Average loaded parameters per token during the generation of up to 100 tokens across 18 prompts, measured on MatFormer-OLMo under different memory budgets.

서로 다른 지표

활성화 비율은 사용 폭의 비율,
로딩량은 RAM에 없어 실제로 읽어온 바이트 수

결과

같은 활성화 비율에서 DWI의 로딩량은 대체로 비교군과 유사

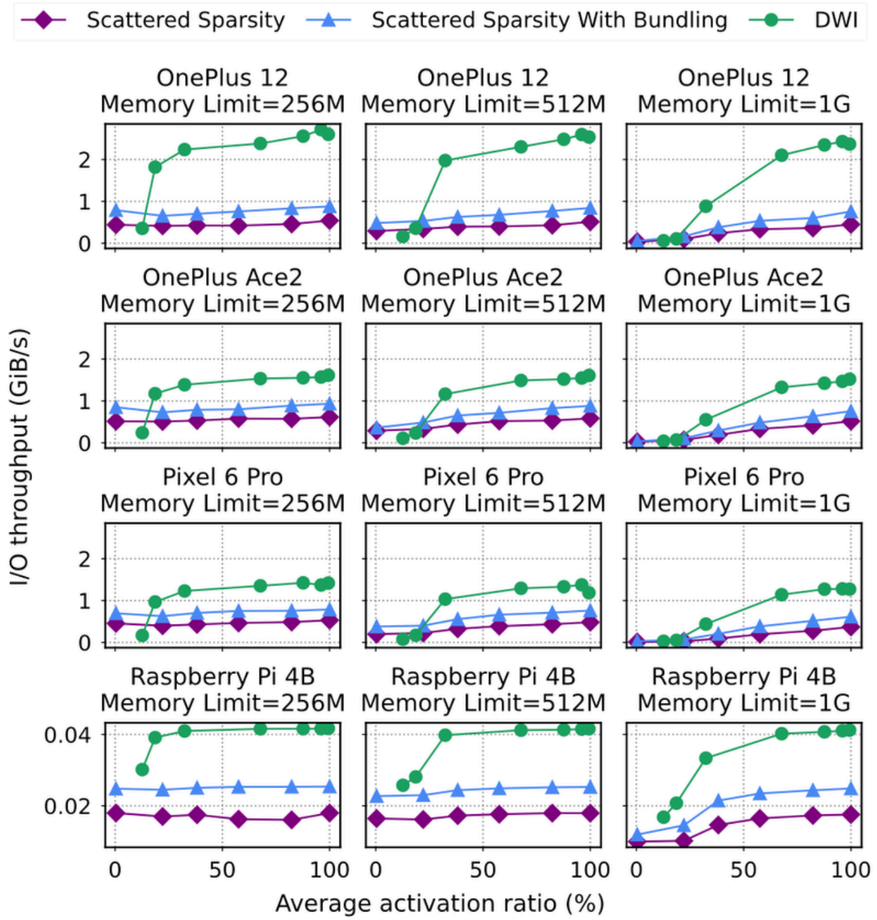
캐시 재사용 효과

예산이 커지면 반복 로딩이 줄어 활성화량과
실제 로딩량이 일대일로 증가하지 않을 수 있음

속도 향상의 원인

로딩량 감소만으로 설명하지 않고,
연속 접근과 처리량도 확인 필요

추론 중 유효 I/O 처리량



지표 정의

전체 로딩 바이트 수를 전체 추론 시간으로 나눈 유효 처리량

분모 범위

저장장치 읽기 작업에 걸린 시간만 분모로 사용한 대역폭 측정과 구분해야 함

결과

비슷한 로딩량에서도 연속 접두부 접근과 로딩·연산 중첩이 실행 효율을 높임

생성 토큰당 에너지

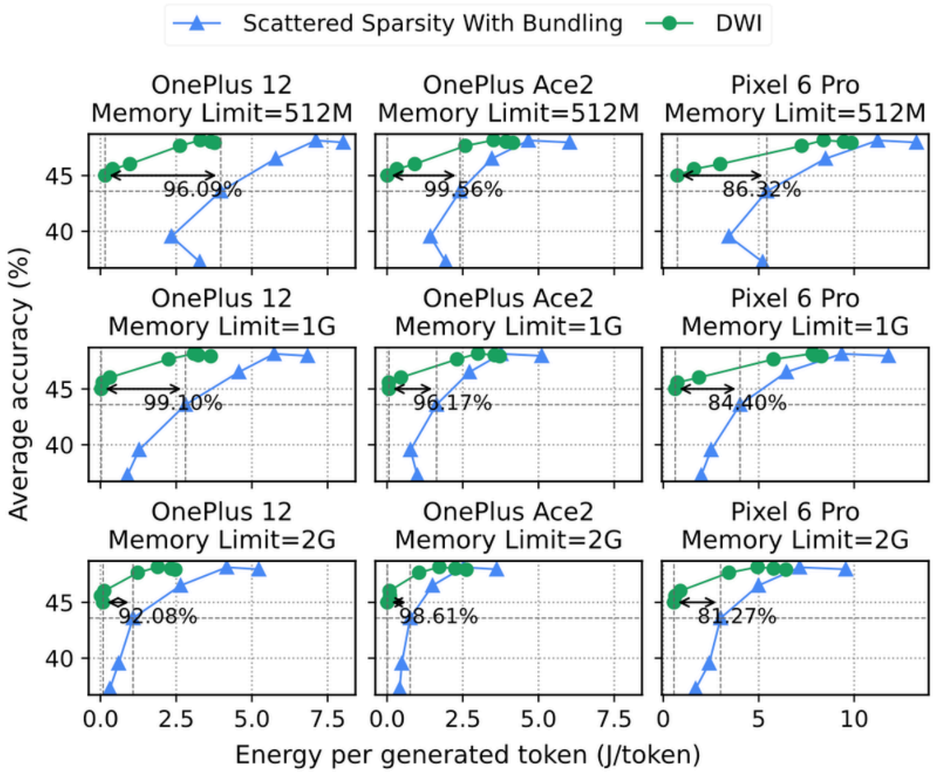


Fig. 12. Per-token energy consumption during the generation of up to 100 tokens across 3 prompts (1 from each dataset), measured on MatFormer-OLMo under different memory budgets.

측정 방식

기기의 전압·전류 정보를 이용해 에너지를 누적하고
생성 토큰 수로 나눔

관찰 결과

DWI가 비교군보다 낮은 토큰당 에너지를 보임
생성 시간 감소가 주요 원인

평가 범위

세 데이터셋에서 각 1개 프롬프트 사용한 축소된 설정에서 수행
순간 전력이 아닌 토큰 완성까지의 누적 에너지

캐시 정책의 로딩량과 실행 속도 비교

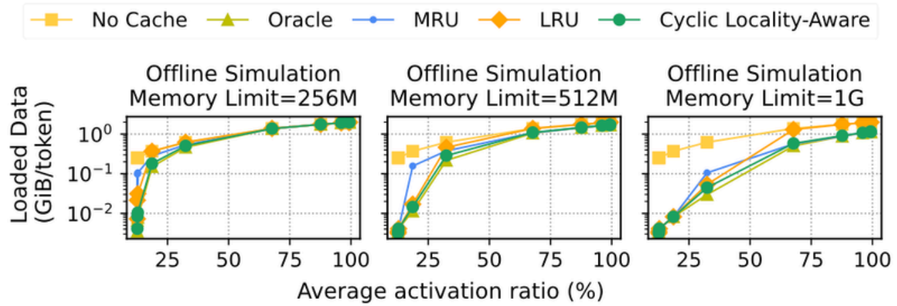


Fig. 13. Offline simulation of per-token loaded data under various cache policies. Measured on MatFormer-OLMo during the generation of up to 100 tokens using a single prompt from OpenBookQA.

LRU·MRU 비교

오래 사용하지 않은 블록 또는 가장 최근 블록을 퇴출하는 정책과 비교

Oracle 기준

미래 접근을 아는 Belady 정책.
실제 온라인 정책의 이상적 비교 기준

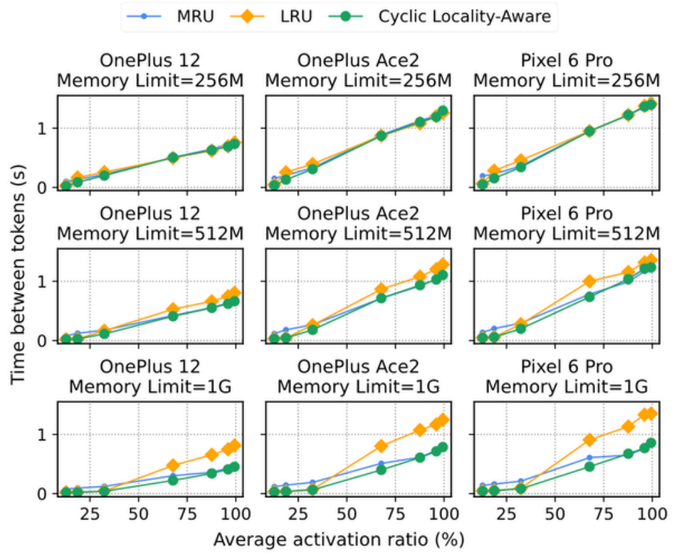
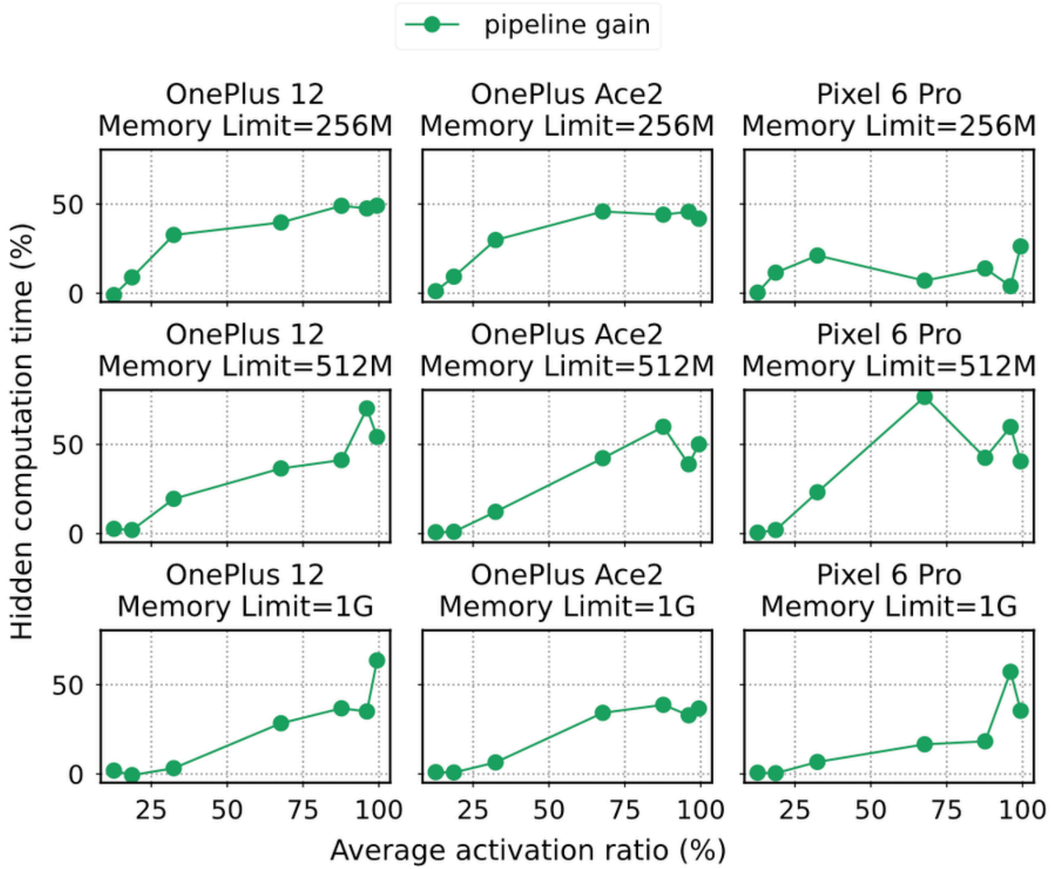


Fig. 14. Comparison of the inference speed of MatFormer-OLMo under various cache policies. Measured during the generation of up to 100 tokens across 3 prompts (1 from each dataset).

두 단계 검증

- 시뮬레이션 로딩량
- 실제 추론 속도

로딩·연산 중첩의 개별 기여



비교 실험

파이프라인 실행과 로딩·연산의 순차 실행을 비교

시간 절약 비율

$$r = \frac{T_{no-pipe} - T_{pipe}}{T_{compute}}$$

- (파이프라인 미적용 시간 - 적용 시간) / 연산 시간

분모 해석

$T_{compute}$ 는 전체 중첩 가중치가 RAM에 들어가는 조건에서 측정한 연산 시간

학습 중 품질과 평균 폭의 수렴

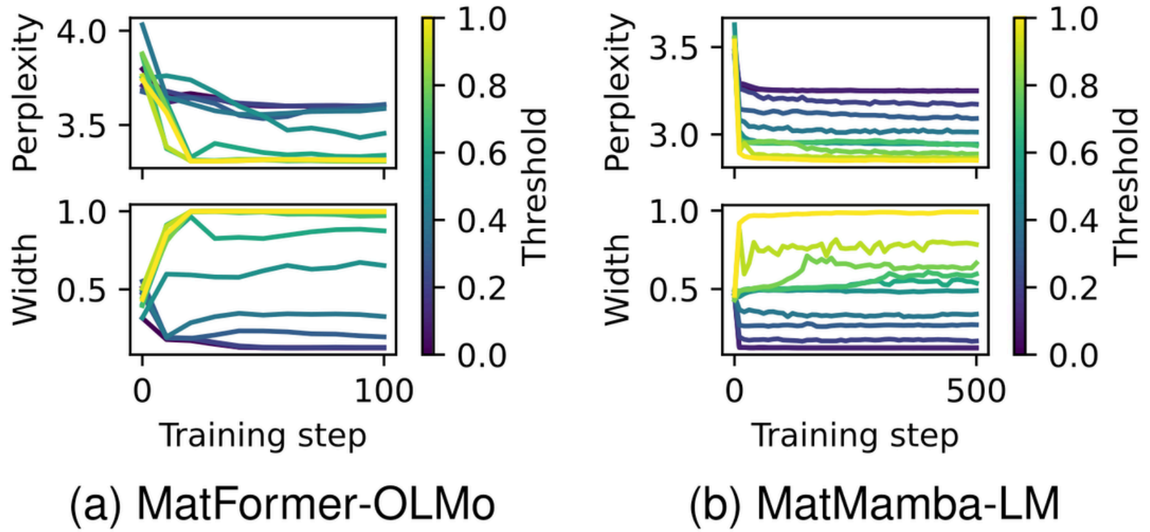


Fig. 16. The perplexity and average width of the trained dynamic-width model, measured on the C4 validation set during the width-selector training process for MatFormer-OLMo and MatMamba-LM on diverse thresholds.

검증 데이터와 지표

C4 검증 부분집합에서 퍼플렉서티와 평균 폭을 관찰

품질·효율 확인

퍼플렉서티는 낮을 수록 예측이 좋음
평균 폭은 선택기가 실제로 어느 정도의 폭을 사용하는지 나타냄

결과

두 모델에서 학습이 진행되며 각 지표가 안정적인 값에 도달

목표값에 따른 학습

서로 다른

τ와 λ의 민감도 분석

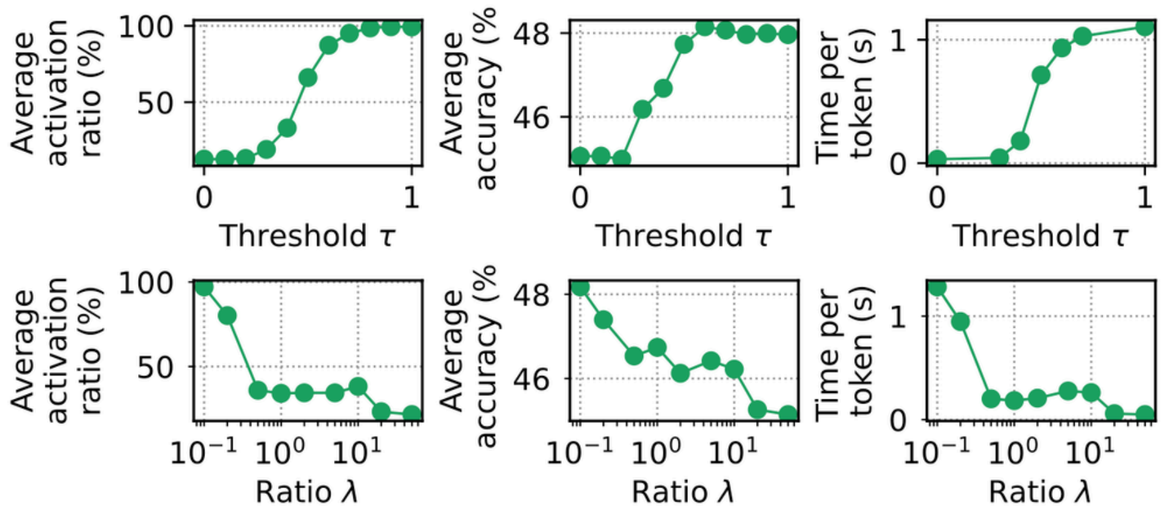


Fig. 17. Sensitivity of MatFormer-OLMo to the threshold τ and weight λ in auxiliary loss. We report average activation ratio, average accuracy on downstream tasks, and decoding time (s/token). Decoding time is measured on OnePlus Ace2 during the generation of up to 100 tokens across 3 prompts with budget = 512MB. Top: τ sensitivity ($\lambda = 1$). Bottom: λ sensitivity ($\tau = 0.4$).

τ 변화

$\lambda = 1$로 고정.

목표 활성화 비율에 따라 폭·정확도·생성 시간이 변화

λ 변화

$\tau = 0.4$로 고정.

$\lambda = 0.5 - 10$에서 활성화량과 성능이 비교적 안정적

생성 시간 측정 조건

MatFormer-OLMo, Oneplus Ace2, 예산 512MB,
3개 프롬프트에서 최대 100토큰 생성

동적 희소성과 중첩 모델 연구 비교

TABLE V
COMPARISON OF DYNAMIC SPARSITY METHODS

System	Training signal	Runtime adaptivity granularity	Storage layout assumption	Memory constraint handling	I/O overlap
DWI	end to end CE + auxiliary loss	discrete per-layer width	prefix-contiguous blocks	cache+swapping	Yes
Flextron	CE from surrogate model	discrete per-layer width	in memory	N/A	N/A
PowerInfer-2	dense-model activation label	per activation tensor	scattered tensors+colocation	cache+swapping	Yes
LLM in a flash	dense-model activation label	per activation tensor	scattered tensors+colocation	cache+swapping	Not mentioned
Deja Vu	dense-model activation label	per activation tensor	in memory	N/A	N/A
static subnet deploy	N/A	N/A	in memory	static subnetwork extraction	N/A

On-flash 희소 추론

가중치 선택·묶기·캐시를 활용하지만 작은 흩어진 접근의 I/O 제약이 남음

동적 폭 선택의 학습

- Flextron은 품질 예측 surrogate model 사용
- DWI는 gradient passthrough로 직접 학습

DWI의 결합 설계

중첩 접두부 선택에 순환 지역성 캐시와 블록 파이프라인을 결합
양자화·증류와는 상보적

결론: 동적 폭 선택과 가중치 관리

동적 폭 선택

중첩 모델에서 토큰과 층별 폭을 정하고
품질 손실과 효율 벌점으로 선택기를 학습

순환 지역성 캐시

반복적인 방문 순서에 맞춰 가중치 블록을
유지하거나 퇴출하여 재로딩을 줄임

블록 파이프라인

준비된 블록을 계산하는 동안 부족한 블록을 로딩
상태 누락은 지연 보충

검증된 성능 범위

MatFormer 최대 34배, MatMamba 약 1.5배
모델/기기/예산/비교군에 따라 서로 다른 결과를 보임

Q&A

Presented by :
학부연구생 김승겸