Banya AI 1

A Language-Model Training Engine Using Hand-Derived Closed-Form Exact Adjoints Without Automatic Differentiation

자동미분 없이 수동으로 유도한 정확한 전치로 학습하는 언어모델 엔진

한혁진 (bokkamsun@gmail.com) · ORCID 0009-0007-4132-1707

반야 연구 시리즈 제1편

본 문서 CC BY 4.0 · 코드 Apache-2.0

초록

본고는 자동미분(automatic differentiation, 컴퓨터가 미분을 자동으로 대신 해 주는 라이브러리) 없이 언어모델(language model, 다음 낱말을 예측하는 신경망)을 학습하는 엔진(engine, 학습을 실제로 돌리는 핵심 계산기)을 제시한다. 이 엔진은 신경망을 이루는 모든 연산 원시(primitive, 더 쪼갤 수 없는 기본 연산)마다 그 야코비안 전치(Jacobian transpose, 기울기를 뒤로 흘릴 때 쓰는 전치 연산. 수반 adjoint 라고도 한다)를 수동으로 유도한 닫힌형(closed-form, 반복 없이 한 번의 수식으로 값이 확정되는 형태) 수식으로 직접 구현한다. 본고의 초점은 표준 구조를 수동으로 옮겨 적는 재구현이 아니라, 자동미분 학습 루프에 정확히 물리기 어려운 비표준 연산 — 순환행렬(circulant matrix) 채널 혼합과 분배함수(partition function, softmax 의 나눗셈 상수) 없는 혼합 — 의 정확한 전치에 있다. 분배함수 없는 혼합은 엔진의 학습 경로에 실재하며, 순환행렬 채널 혼합은 닫힌형 전치의 유도와 유한차분 검증으로 제시한다. 엔진의 채널 변환은 필터 사다리다. 그렇게 계산되는 기울기(gradient, 손실을 줄이는 방향)는 표준 오차역전파(backpropagation, 줄여서 backprop)와 수치적으로 동일하되, 자동미분 그래프(graph, 연산을 기록해 두는 자료구조)를 전혀 만들지 않는다. 측정 결과, 자동미분 라이브러리는 메모리에 올라오지 않았고, 정규화 스케일은 블록 수 2에서 10까지 일정했으며, 수동으로 유도한 전치와 유한차분(finite difference)의 비율 중앙값은 0.9997이었다. 나란한 학습 대조에서는 같은 소형 아키텍처와 같은 초기값과 같은 배치 열로 표준 역전파 기준 구현과 300스텝을 나란히 돌려, 손실 궤적의 차이가 전 구간 1e-06 미만이고 최종 손실이 1.1039로 동일했다. 나아가 이 엔진이 주는 확정 이득을 정량화한다. 첫째, 자동미분으로는 학습 루프에 정확히 물리기 어려운 순환행렬 채널 혼합이나 분배함수 없는 연산을 정확한 전치로 학습할 수 있다(가능성 이득). 둘째, 순환행렬 채널 혼합은 밀집 행렬 대비 파라미터를 약 1024배 감소시킨다(구조적 사실). 셋째, 블록축 배치 실행은 순차 대비 혼합 연산에서 최대 5.07배 빨랐다. 핵심 주장은 "역전파를 쓰지 않는다"가 아니라 "자동미분을 수동으로 유도한 정확한 전치로 대체하면 무엇이 가능하고 무엇이 저렴해지는가"이다.

목차

용어 범례

용어를 아래 표에 정의한다. 같은 개념에는 같은 이름만 사용하며, 일곱 편의 본문과 그림이 모두 이 표를 따른다. 코드 기준은 동봉 코드에서 그 개념을 구현한 식별자이고, 일반 용어는 학계에서 통용되는 가장 가까운 이름이다. 다를 때만 칸을 채웠다.

용어 범례 1. 엔진과 신용

용어	뜻	코드 기준	일반 용어
정확한 전치	각 연산의 수반을 손으로 유도해 닫힌형으로 구현한 역방향 계산. 이 엔진과 신용 사슬의 고유명	`banya_core.py`	수동 역전파
수반	순전파의 짝 연산. 기울기를 정확히 거꾸로 되돌려 흘린다	`bwd` 커널들	adjoint
신용	어떤 지점의 은닉에 대한 손실 기울기. 파라미터가 결과에 진 책임	델타(δ)	그레이디언트
신용 사슬	캐시를 역순으로 되짚으며 수반을 곱해 신용을 입력까지 되돌리는 골격	`backward`	오차역전파
수반 정합	수반 식이 유한차분과 같은 값을 내는지 확인하는 검사	각 편 프로브	gradient check
순환행렬 채널 혼합	채널 혼합을 순환행렬로 두고 rfft 성분곱으로 계산하는 연산		순환 합성곱
상대 거리 혼합	거리에만 의존하는 공유 계수로 자리들을 섞는 혼합	`m_mat_w_lag`	Toeplitz 혼합
시간혼합	다른 자리의 정보가 현재 자리로 섞여 드는 경로의 총칭		토큰 혼합
혼합 헤드	혼합을 병렬로 나누는 머리 수		어텐션 헤드
필터 사다리	넓은 필터 몇 개부터 좁은 필터 여러 개까지 스케일이 깊이를 따라 변하는 콘볼루션 채널 변환	`m_mat_w_filter`	콘볼루션 필터뱅크
창	모델이 한 번에 보는 토큰 자리들. 측정 단위로는 고정 길이 텍스트 조각	`T`	컨텍스트 윈도
프로브	논문 수치를 목표값과 함께 재생하는 검증 스크립트	`probe` 폴더	재현 스크립트
얼린 모델	학습을 멈춰 고정한 발달 단계 모델	`model` 폴더 npz	frozen checkpoint
홀드아웃	학습에 쓰지 않은 평가 텍스트		held-out set

용어 범례 2. 바이토큰과 게이트

용어	뜻	코드 기준	일반 용어
바이토큰	토큰 하나를 데이터면과 연산면의 직교 쌍으로 두는 구조	`USE_BITOKEN`	
데이터면	토큰의 내용을 담는 임베딩 면. 더하기 자리로 들어간다	`m_mat_w_data_axis`	토큰 임베딩
연산면	토큰의 연산 지시를 담는 임베딩 면. 곱하기 자리로만 들어간다	`m_mat_w_operate_axis`	
더하기 자리, 곱하기 자리	데이터면이 실리는 잔차 덧셈 위치와 연산면이 작용하는 게이트 곱셈 위치		잔차 연결, 게이팅
게이트	각 채널의 통과량을 0에서 1로 정하는 밸브. 직전 토큰의 연산면이 주입된다	`m_mat_w_gate`	게이팅
한 칸 이동	직전 토큰의 연산면이 다음 자리의 게이트로 주입되는 한 자리 밀림		
연산이 곧 가중치	연산면 임베딩이 라벨 없이 가중치 역할로 창발하는 현상		
일관 연산자	어떤 데이터에도 일관된 방향으로 작용하는 연산자		
방향 일관성	같은 연산이 여러 데이터를 꺾는 방향들의 평균 코사인. 연산자 창발의 지표	`p_direction_consistency`	
회전 연산자 게이트	연산면을 오일러 회전각으로 읽는 게이트. 본문 축약은 회전 게이트	`paper7_rotation.py`	
기본 연산	게이트가 한 홉 지시로 곧바로 표현할 수 있는 연산		프리미티브
깊이 배제 판별	깊이를 1블록으로 제한해 게이트만 변수로 남긴 의도된 편파 측정	제7편 프로브	
자기 되먹임	자기 출력을 입력으로 되먹이며 끝 상태를 채점하는 채점 방식		자기회귀 롤아웃

용어 범례 3. 되새김과 발달

용어	뜻	코드 기준	일반 용어
되새김	기울기 없이 임베딩을 이웃끼리 당겨 뭉치게 하는 자기조직화 학습	제2편 프로브	자기조직화
동종군	씨앗에서 코사인 최근접 이웃을 모은 뜻 가까운 개념 집합		클러스터
무게중심 당김, 감쇠	평균 쪽으로 당기고 원본 쪽으로 되돌리는 되새김의 두 힘. 균형에서 평형이 창발한다	`ALPHA`, 감쇠율	
월드먼저	언어 이전에 감각, 공간, 논리 축을 먼저 발달시키는 커리큘럼 순서	`banya_world_data`	커리큘럼 학습
발달 단계	스텝 수로 정의하고 사람 성장 이름을 붙인 체크포인트 단계		
축	임베딩 안에 방향으로 형성되는 성질의 눈금. 감각, 순서, 범주		표현 축

용어 범례 4. 어휘와 메타인지

용어	뜻	코드 기준	일반 용어
음절 원자	어휘 바닥층의 음절 단위 토큰	`AtomTokenizer`	문자 단위 토큰
묶음	음절 원자를 묶어 사전에 올린 개념 토큰. 어휘 단위로만 사용한다		서브워드
캐시 사전 토큰화	원자 아래층에 묶음 사전 위층을 얹은 2층 어휘 체계	제5편 프로브	서브워드 토큰화와 대비
확실, 모호, 모름	자기 출력 분포로 갈리는 세 앎 상태	pmax, 엔트로피	불확실성 추정
라우팅	상태에 따라 답의 처리 경로를 배정하는 것		routing
재해석	모호 히든을 문맥과 함께 재순전파해 확실로 끌어올리는 절차	제6편 프로브	
유사도 정리	데이터면 유사도로 개념이 정돈되는 축		
추론 순회	연산면을 따라 추론을 이어 가는 축		
자기 발화	유사도 정리와 추론 순회를 엮어 스스로 말을 잇는 목표 상태		
반야프레임	시리즈의 배경을 이루는 공리 15개 체계. 본문 주장의 근거는 아님		

기호 범례

이 편의 수식과 본문이 함께 사용하는 기호를 정의한다. 한 식 안에서만 쓰는 국소 기호는 각 식 아래의 기호 줄에 적었다.

기호	한글 명칭	정의 및 의미	자료형
H	채널 폭	은닉 벡터의 차원 수. 정식 설정 1024	정수
T	문맥 길이	한 번에 보는 낱말 자리 수. 정식 설정 128	정수
N	블록 수(깊이)	같은 구조 층의 반복 횟수	정수
V	어휘 크기	출력 낱말 후보의 개수	정수
$h^{(k)}$	k번째 블록 출력	k번째 블록을 통과한 은닉 벡터	벡터
z	로짓	출력 헤드가 낸 낱말별 점수(정규화 전)	벡터
y	정답	실제 다음 낱말의 색인. 출력과 혼동 금지	정수
p	예측 분포	z에 softmax 를 적용한 확률	벡터
L	손실	교차엔트로피 손실	스칼라
f_k	블록 함수	k번째 블록의 순전파 사상	함수
$J_k^\top$	야코비안 전치	f_k의 국소 기울기 행렬의 전치. 신용을 뒤로 흘리는 연산	연산자
δ	신용(델타)	어떤 지점의 은닉에 대한 손실 기울기	벡터
g	로짓 기울기	$\partial L/\partial z$	벡터
r	정규화 스케일	제곱평균 제곱근 정규화의 역제곱평균 계수	스칼라
c	순환 커널	순환행렬을 정의하는 한 벡터	벡터

1. 서론

딥러닝(deep learning, 여러 층 신경망 학습)의 표준 학습법은 오차역전파이며, 실무에서는 이를 자동미분 라이브러리가 대행한다. 자동미분은 순전파(forward pass, 입력에서 출력까지 앞으로 계산) 동안 모든 연산을 그래프에 기록하고, 역방향에서 그 그래프를 되짚어 기울기를 계산한다. 이 방식은

편리하지만 두 가지 제약을 발생시킨다. 첫째, 학습에 넣을 수 있는 연산이 사실상 그 라이브러리가 미분을 지원하는 연산으로 한정된다. 둘째, 신용 배정(credit assignment, 출력의 오차를 각 파라미터의 책임으로 나누는 일)의 경로가 라이브러리 내부에 감춰져 세밀한 개입이 어렵다.

본고는 정반대의 선택을 한다. 자동미분 라이브러리를 전혀 쓰지 않고, 신경망을 이루는 모든 연산의 정확한 전치를 수동으로 유도하여 닫힌형 수식으로 직접 구현한다. 이렇게 얻은 기울기는 근사가 아니라 backprop 과 수치적으로 동일하며, 그 동일한 값을 자동미분 그래프 없이 각 연산의 수반을 역순으로 한 번씩 곱하는 명시적 사슬로 계산한다. 이 선택의 대가는 수동으로 유도한 노동이지만, 그 대가로 얻는 이득을 7절에서 정량화한다.

1.1 기여

- 정확한 전치 신용 사슬 : 순전파를 한 번 수행한 뒤 블록별 캐시를 역순으로 되짚어 각 연산의 수동으로 유도한 정확한 전치를 흘리는 골격. 자동미분 그래프가 없고 값은 backprop 과 동일하다(4절).
- 연산별 닫힌형 전치 : 정규화(RMSNorm), 순환행렬 채널 혼합(고속 푸리에 변환 활용), 상대 거리 혼합(Toeplitz)의 야코비안 전치를 수동으로 유도해 제시한다(5절).
- 검증과 실측 : 세 항목 검증 프로토콜을 제시하고 그 실측 결과를 그림으로 보인다(6절).
- 확정 이득의 정량화 : 가능성 이득(비표준 연산 학습), 파라미터 절감(약 1024배), 배치 속도 이득(최대 5.07배 실측)을 근거와 함께 제시한다(7절).

2. 관련 연구와 위치

backprop 은 각 뉴런의 책임을 계산할 때 순전파 가중치의 전치를 역방향에 그대로 재사용한다. 이 재사용은 뇌에서 불가능하다고 여겨지는 가중치 전달 문제(weight transport problem, 앞으로 갈 때 쓴 가중치를 뒤로 돌아올 때 똑같이 못 쓴다는 제약)로 불린다 [6]. 이를 피하려는 생물학적 타당성 계열로 피드백 정렬(Feedback Alignment) [2], 직접 피드백 정렬(Direct Feedback Alignment) [3], 순전파-순전파(Forward-Forward)와 PEPITA [1], 타깃 전파(Target Propagation) [4], 평형 전파(Equilibrium Propagation) [5], 예측 부호화(Predictive Coding) [7] 가 있다.

이들의 공통점은 정확한 기울기를 포기하고 국소 신호로 근사한다는 것이며, 대규모 깊은 망으로의 확장성이 미해결 난제로 남아 있다 [8]. 본고의 엔진은 이 계열과 목적이 반대이다. 본고는 정확한 기울기를 그대로 계산하되(표 2-1) 자동미분 라이브러리만 걷어낸다. 본고가 대체하는 대상은 근사 학습법이 아니라 자동미분 구현 그 자체이며, 얻는 것은 근사 없이 비표준 구조적 연산을 학습 루프에 정확하게 물릴 자유이다.

자동미분 없이 정확한 기울기로 학습하는 수동 구현 자체는 새로운 일이 아니다. 자동미분 도구가 보급되기 전의 신경망 구현이 본래 그러했고, 근래에도 표준 구조 전체를 수동으로 옮겨 적어 표준 구현과 같은 값을 확인한 공개 구현이 있다 [9]. 본고가 다른 점은 대상이다. 본고가 다루는 것은 표준 구조가 아니라 자동미분에 정확히 물리기 어려운 비표준 연산이며, 그 닫힌형 전치를 유도하고 검증하며 분배함수 없는 혼합은 학습 경로에서 실제로 돌리는 것이 본고의 몫이다.

표 2-1. 신용 배정 방식의 분류와 본고의 위치

계열	기울기	별도 역방향 경로	순전파 횟수	본고와의 관계
backprop (자동미분)	정확한 기울기	자동미분 그래프	1	같은 값, 다른 구현
피드백 정렬,직접 피드백 정렬	근사(랜덤 정렬)	고정 랜덤 행렬	1	목적 반대(근사)
순전파-순전파,PEPITA	근사(국소 목적)	없음(순전파 2회)	2	목적 반대(근사)
타깃 전파,평형 전파,예측 부호화	근사(목표,평형, 예측오차)	오토인코더,완화, 예측뉴런	다수	목적 반대(근사)
수동 구현(표준 구조) [9]	정확한 기울기 (backprop 동일)	없음(수동으로 유도)	1	같은 값. 대상이 표준 구조에 한정
본고 (수동으로 유도한 정확한 전치)	정확한 기울기 (backprop 동일)	없음(수동으로 유도한 닫힌형)	1	자동미분 자체를 대체

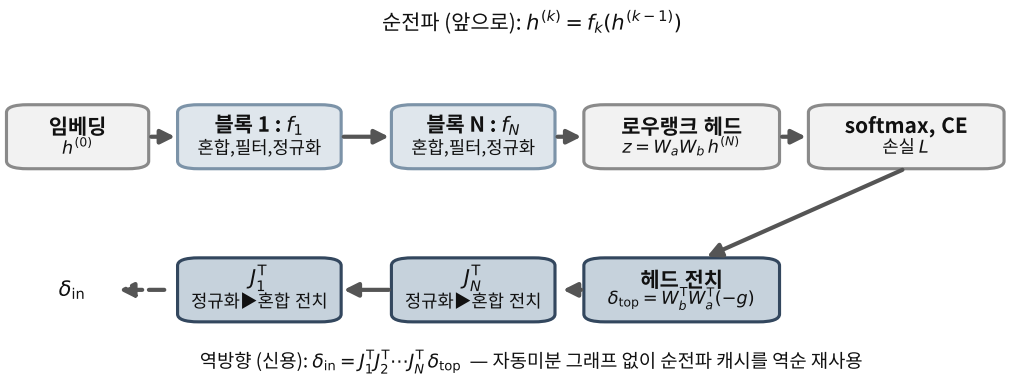
본고는 표의 마지막 줄에 위치한다. 정확성은 첫 줄(backprop)과 같고, 자동미분 그래프가 없다는 점만 다르다. 다섯째 줄의 수동 구현 계열과는 값이 같고 대상 연산이 다르다.

3. 표기와 전체 구조

기호는 앞머리의 기호 범례에 정의했다. 같은 개념에는 같은 문자만 사용하며 본문 수식과 그림이 모두 그 표를 따른다.

모델의 순전파는 임베딩(embedding, 낱말을 벡터로 바꾸는 표)에서 시작해 N개의 동일 구조 블록을 지나 로우랭크 출력 헤드(low-rank head, 큰 출력 행렬을 두 작은 행렬의 곱으로 근사한 것)로 로짓을 내고, softmax 와 교차엔트로피로 손실을 맺는다. 한 블록의 내부는 상대 거리 혼합, 게이트, 채널 필터 사다리, 정규화 잔차로 이루어진다. 전체 경로를 그림 3-1에 보인다.

그림 3-1. 전체 순전파 파이프라인과 블록 내부, 그리고 되돌아오는 정확한 전치 신용



위쪽은 순전파(임베딩 ▶ N개 블록 ▶ 로우랭크 헤드 ▶ 손실), 아래쪽은 역방향 정확한 전치 신용 사슬이다. 각 블록 내부는 거리 혼합,게이트,필터 사다리,정규화 잔차로 구성되며, 역방향은 이 순서를 정확히 뒤집는다. 추가 순전파 없이 한 번의 순전파에서 저장한 캐시만으로 신용이 완성된다.

4. 정확한 전치 신용 사슬

4.1 순전파 합성과 신용 사슬

순전파는 블록 함수의 합성이다.

식 4-1. 순전파 합성

$$h^{(N)} = (f_N \circ f_{N-1} \circ \cdots \circ f_1)(h^{(0)}), \qquad z = W_a W_b \, h^{(N)}$$

수식

기호: $h^{(0)}$ 는 임베딩 출력, f_k 는 k번째 블록 함수, $\circ$ 는 함수 합성(먼저 f_1, 그다음 f_2), $W_a W_b$ 는 로우랭크 헤드, z 는 로짓이다.
작동 방식: 입력 은닉을 블록에 차례로 통과시켜 최종 은닉 $h^{(N)}$ 을 얻고 헤드로 낱말별 점수 z 를 낸다.
실제 동작: 합성 $\circ$ 는 "f_1 의 결과를 f_2 에 넣고 그 결과를 다시 f_3 에 넣는" 연쇄 대입이다. $N = 3$ 이면 $h^{(3)} = f_3(f_2(f_1(h^{(0)})))$ 이다.

손실 L 의 입력 은닉에 대한 기울기는 연쇄법칙(chain rule, 합성함수의 미분 규칙)에 따라 각 블록 야코비안의 전치를 역순으로 곱한 것이다.

식 4-2. 정확한 전치 신용 사슬

$$\delta_{\text{in}} = \frac{\partial L}{\partial h^{(0)}} = J_1^\top J_2^\top \cdots J_N^\top \, \delta_{\text{top}}, \qquad \delta_{\text{top}} = \frac{\partial L}{\partial h^{(N)}}$$

수식

기호: δ_{top} 은 최종 은닉에 대한 손실 기울기(출발 신용), J_k 는 k번째 블록의 국소 야코비안, $J_k^\top$ 는 그 전치, δ_{in} 은 입력까지 되돌아온 신용이다.
작동 방식: 손실의 기울기는 마지막 블록의 전치부터 첫 블록의 전치까지 차례로 곱해 만들어진다. 이것이 backprop 이 계산하는 바로 그 값이며, 본고는 이 곱을 자동미분 그래프 없이 각 $J_k^\top$ 를 수동으로 유도한 닫힌형으로 구현해 실현한다.
실제 동작: 전치 $J_k^\top$ 는 "f_k 가 앞으로 섞은 것을 정확히 반대로 되섞어 신용을 나누는" 연산이다. 곱의 순서가 역순이므로 구현은 블록을 뒤에서 앞으로 훑으며 신용 벡터를 갱신한다.

식 4-2를 숫자로 확인한다. 두 블록 $N = 2$, 은닉 차원 2, 각 블록이 선형 $f_k(x) = A_k x$ 이면 $\delta_{\text{in}} = A_1^\top A_2^\top \delta_{\text{top}}$ 이다. $A_1 = \begin{pmatrix} 1 & 3 \\ 0 & 1 \end{pmatrix}$, $A_2 = \begin{pmatrix} 2 & 0 \\ 1 & 1 \end{pmatrix}$ 로 둔다.

단계	계산	값
출발 신용	δ_{top}	$(1,\,0)$
블록 2 전치	$u = A_2^\top \delta_{\text{top}}$	$(2,\,0)$
블록 1 전치	$\delta_{\text{in}} = A_1^\top u$	$(2,\,6)$

같은 값을 backprop 으로 계산해도 $(2,\,6)$ 이다. 본고는 이 $(2,\,6)$ 을 그래프 없이 전치의 역순곱으로 직접 얻는다.

4.2 손실 기울기의 출발점과 로우랭크 헤드 전치

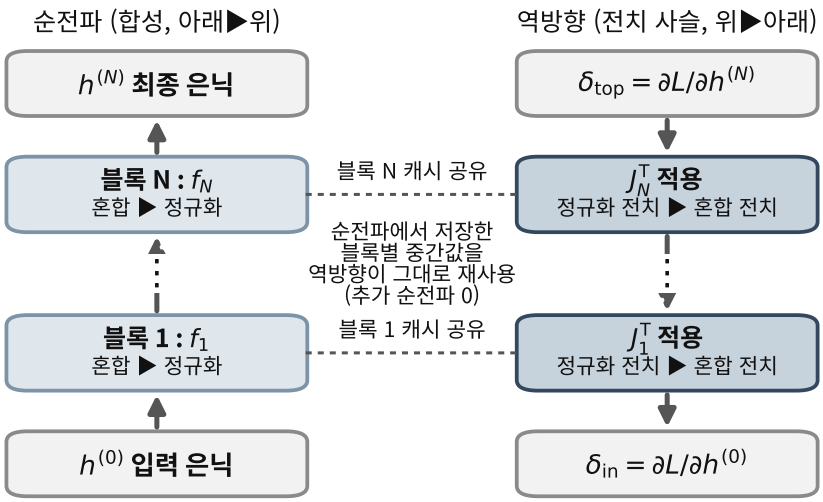
식 4-3. softmax,교차엔트로피 기울기와 헤드 전치

$$p = \text{softmax}(z), \quad g = \frac{\partial L}{\partial z} = p - \mathbf{1}_y, \qquad \delta_{\text{top}} = W_b^\top \left(W_a^\top (-g) \right)$$

수식

기호: p 는 예측 분포, $\mathbf{1}_y$ 는 정답 위치만 1인 원핫 벡터, g 는 로짓 기울기, W_a, W_b 는 로우랭크 헤드의 두 인자 행렬, $-g$ 는 하강 방향을 담은 신용이다.
작동 방식: softmax 와 교차엔트로피를 함께 미분하면 항이 소거되어 "예측에서 정답을 뺀" 형태만 남는다. 큰 헤드 전치 $W_{\text{head}}^\top$ 는 작은 두 전치 $W_a^\top, W_b^\top$ 를 순서대로 적용해 같은 값으로 얻는다.
실제 동작: $g = p - \mathbf{1}_y$ 는 정답 확률은 올리고 나머지는 내리는 힘이다. 음부호는 신용을 처음부터 하강 방향으로 흘려 이후 갱신을 더하기로 통일하기 위한 규약이다.

그림 4-1. 순전파 합성과 역방향 전치 사슬의 대칭



순전파(왼쪽)가 임베딩에서 블록을 아래에서 위로 쌓아 올린 것을, 역방향(오른쪽)은 위에서 아래로 각 블록의 전치 $J_k^\top$ 를 곱해 정확히 되짚는다. 각 블록의 전치는 순전파의 역순(정규화 전치 다음 혼합 전치)으로 내부를 되짚으며, 가로 점선은 순전파에서 저장한 캐시를 역방향이 재사용함을 나타낸다.

5. 연산별 닫힌형 전치

정확한 전치 사슬이 성립하려면 블록을 이루는 모든 연산이 각자의 닫힌형 전치를 가져야 한다. 본절은 대표 연산 세 가지의 수동으로 유도한 전치를 보인다. 이들은 자동미분에 잘 맞지 않는 구조적 연산이지만, 수동으로 유도하면 정확한 전치가 닫힌형으로 나온다. 세 가지 중 정규화와 상대 거리 혼합은 엔진의 학습 경로에 들어 있고, 순환행렬 채널 혼합은 유도와 수반 검증으로 제시한다(엔진의 채널 변환은 필터 사다리다, 3절). 순환,콘볼루션 연산의 아키텍처적 역할과 스케일,깊이 설계 자체는 후속 논문의 주제이며, 여기서는 엔진이 이런 연산까지 정확한 전치로 미분한다는 사실만을 다루며(11절), 그 수반 정합은 6절 프로브의 유한차분 대조로 확인한다.

5.1 정규화(RMSNorm)의 전치

식 5-1. RMSNorm 순전파와 전치

$$y_i = g_i\, x_i\, r, \quad r = \left(\tfrac{1}{H}\textstyle\sum_j x_j^2 + \epsilon\right)^{-1/2}, \qquad \delta^x = r\,a - \tfrac{r^3}{H}\,x\,\textstyle\sum_j a_j x_j, \quad a = \delta^y \odot g$$

수식

기호: x 는 입력 은닉, g 는 채널별 게인(gain, 배율), r 은 역제곱평균 스케일, ϵ 은 0 나눗셈 방지 상수, δ^y 는 출력 쪽 신용, $a = \delta^y \odot g$ 는 게인 반영 신용($\odot$ 은 성분별 곱), δ^x 는 입력 쪽 신용이다.
작동 방식: 순전파는 벡터를 자기 크기로 나눠 스케일을 일정하게 만든다. 전치는 그 정규화가 성분들을 서로 얽어 놓은 만큼을 정확히 되푼다. 둘째 항 $-\frac{r^3}{H}x\sum a_j x_j$ 이 얽힘 보정이다.
실제 동작: $\sum_j a_j x_j$ 는 신용과 입력의 내적(inner product, 방향 일치도)이다. 이 한 수를 빼 정규화가 만든 성분 간 상호작용을 상쇄한다. 이 전치가 깊은 블록에서도 신용이 사그라들지 않게 지킨다(그림 6-2의 실측으로 확인).

5.2 순환행렬 채널 혼합의 전치

식 5-2. 순환 혼합의 순전파,전치,커널 기울기

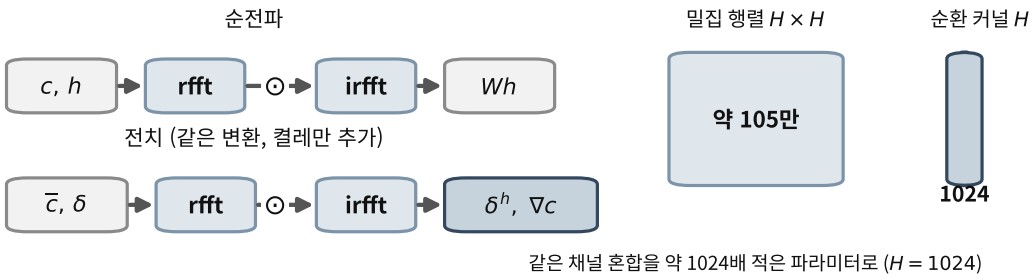
$$Wh = \mathrm{irfft}\big(\,\mathrm{rfft}(c) \odot \mathrm{rfft}(h)\big), \quad \delta^h = \mathrm{irfft}\big(\overline{\mathrm{rfft}(c)} \odot \mathrm{rfft}(\delta)\big), \quad \nabla c = \mathrm{irfft}\big(\overline{\mathrm{rfft}(h)} \odot \mathrm{rfft}(\delta)\big)$$

수식

기호: c 는 순환 커널(첫 열만 저장), h 는 입력, rfft,irfft 는 실수 고속 푸리에 변환과 역변환, $\overline{(\cdot)}$ 는 복소 켤레(conjugate), δ 는 출력 쪽 신용, δ^h 는 입력 쪽 신용, ∇c 는 커널 기울기이다.
작동 방식: 순환행렬의 곱은 순환 합성곱(circular convolution)이라 곱셈정리에 의해 주파수 영역의 성분별 곱이 된다. 전치는 켤레만 취하고, 커널 기울기는 입력과 신용의 교차상관(cross-correlation)이다. 세 식이 모두 같은 변환으로 얻어진다.
실제 동작: 밀집 행렬은 $H \times H$ 개의 수를 갖지만 순환행렬은 커널 H 개만으로 정해진다. $H = 1024$ 이면 밀집은 약 1.05×10^6 개, 순환은 1024 개로 약 1024배 적다(그림 7-1). 곱의 비용도 $O(H^2)$

에서 $O(H \log H)$ 로 감소한다. 위 세 식의 수반 정합은 6절 프로브에서 내적 항등과 유한차분 대조로 검증한다.

그림 5-1. 순환행렬 채널 혼합의 순전파와 전치가 같은 푸리에 변환을 공유



순전파(위)와 전치(아래)가 동일한 푸리에 변환을 쓰고, 전치는 커널에 켤레만 취한다. 오른쪽은 밀집 행렬($H \times H$, 약 105만)과 순환 커널(H, 1024)의 파라미터 규모 비교이다.

5.3 상대 거리 혼합(Toeplitz)의 전치

식 5-3. 상대 거리 혼합의 순전파와 전치

$$x_t^{\text{out}} = x_t + \textstyle\sum_s c_{t-s}\, x_s, \qquad \delta_s^{x} = \delta_s + \textstyle\sum_t c_{t-s}\, \delta_t$$

수식

기호: x_t 는 자리 t 의 은닉, c_{t-s} 는 거리 $t-s$ 에만 의존하는 공유 혼합 계수, δ_t 는 자리 t 의 출력 쪽 신용, δ_s^x 는 자리 s 의 입력 쪽 신용이다.
작동 방식: 같은 거리에는 같은 계수를 쓰므로 계수 수가 자리 수에 선형으로만 증가한다. 전치는 순전파의 첨자 방향을 뒤집은 형태이며, 커널 기울기는 같은 거리 항들을 모아 더하는 산란 합(scatter-add)으로 정확하게 얻는다.
실제 동작: 잔차 항 x_t 는 그대로 통과하는 항등 경로라 전치에서도 δ_s 로 남고, 혼합 항만 방향을 뒤집어 더해진다. 이 항등 경로가 신용을 안정적으로 흘려보내는 지름길이 된다.

6. 검증과 실측

수동으로 유도한 전치가 실제로 정확한 미분인지는 세 항목으로 검사한다. 첫째, 자동미분 라이브러리가 메모리에 올라오지 않았는가. 둘째, 블록을 늘려도 출력과 입력의 크기 비가 $\sqrt{H}$ 근처로 안정한가. 셋째, 수동으로 유도한 전치와 유한차분의 비율 중앙값이 1에 가까운가. 이 검사는 집필 환경에서 측정했다(부록 B).

식 6-1. 유한차분 대비 정합성 비율

$$\rho = \text{median}\left(\frac{\text{수동으로 유도한 전치}}{\text{유한차분}}\right) \approx 1$$

수식

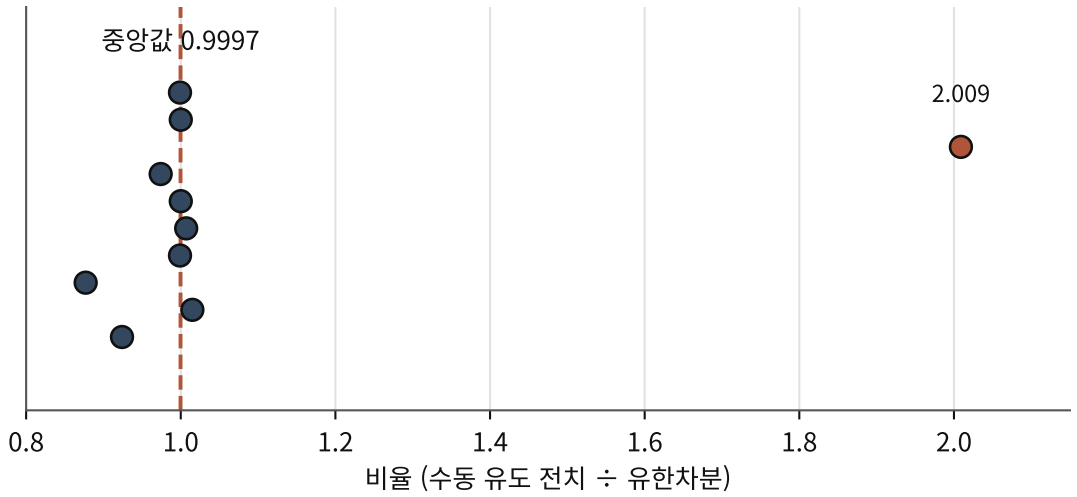
기호: ρ 는 정합성 비율, 유한차분은 파라미터를 아주 조금 흔들어 손실 변화로 기울기를 근사한 값, median 은 중앙값(이상치에 강한 대푯값)이다.
작동 방식: 수동으로 유도한 전치가 정확하면 유한차분과 거의 같아 비율이 1이 된다. 1에서 벗어나면 어느 연산의 전치 유도가 틀렸다는 신호이다.
실제 동작: 유한차분은 값이 0 근처인 소수 항에서 수치 오차로 비율이 크게 튈 수 있으므로 이상치에 강한 중앙값으로 대표한다. 아래 실측이 이를 보인다.

측정

검증은 소형 설정($H = 32$, 블록 6, 문맥 6, 배치 2)에서 측정했다.

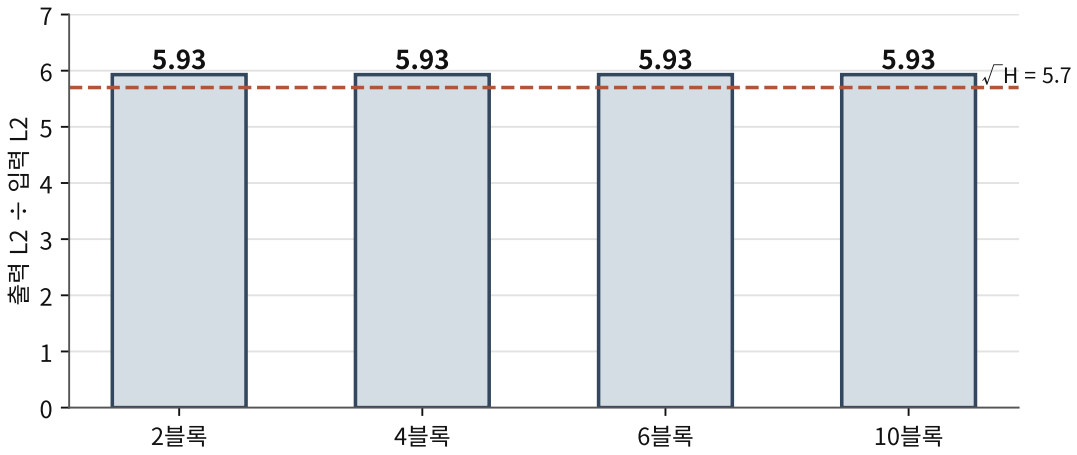
- 자동미분 미사용 : 자동미분 라이브러리 로드 여부 False . 순전파와 역방향이 수동으로 유도한 푸리에 변환,전치 연산만으로 이루어짐.
- 정규화 스케일 안정성 : 블록 수 2, 4, 6, 10 모두에서 출력 대 입력의 크기 비가 5.93 ($\sqrt{H} = 5.7$ 근처)로 일정(그림 6-2).
- 수반 정합 : 비율 중앙값 0.9997 (그림 6-1). 표본 10개 중 9개가 0.877~1.015 에 밀집하고 1개만 유한차분이 0 근처라 2.009 로 튄다.

그림 6-1. 수반 정합 비율 (소형 설정)



비율 10개를 가로축에 점으로 찍었다. 9개가 1.0 왼쪽 가까이(0.877~1.015) 몰려 있고 오른쪽 한 점(2.009)만 유한차분이 0 근처라 튄 것이다. 이상치에 강한 중앙값이 0.9997 로 1.0 에 밀착하므로, 수동으로 유도한 전치가 정확한 미분임을 보인다.

그림 6-2. 정규화 스케일 안정성 (깊이를 늘려도 일정)



블록 수를 2에서 10까지 늘려도 출력 대 입력의 크기 비가 5.93 으로 변하지 않는다. RMSNorm 전치(식 5-1)가 신용의 크기를 층마다 유지하기 때문이며, 이 성질이 깊은 망 학습을 가능하게 한다.

재현

이 절의 세 검증은 부록 C의 준비를 마친 뒤 다음 한 줄로 재생된다.

```
cd probe && python3 paper1_engine_probe.py
```

출력에서 자동미분 라이브러리 로드 여부 False, 정규화 스케일 비 5.93, 수반 정합 비율 중앙값 0.9997, 순환 수반 정합(내적 항등 차이 1e-14 수준, 기울기 비율 중앙값 1.000000)이 본문 수치와 대응한다. 목표값이 출력에 괄호로 같이 인쇄되므로 바로 대조된다.

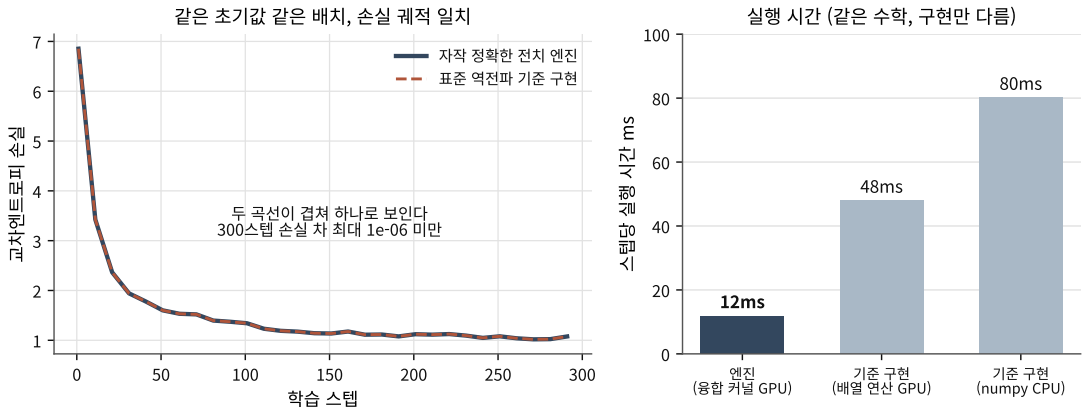
6.1 표준 역전파와의 나란한 학습 대조

유한차분 검사는 한 시점의 기울기만 본다. 수동으로 유도한 전치가 학습 전체에서 표준 역전파와 같은지를 보기 위해, 같은 소형 아키텍처(채널 256, 블록 4, 혼합 헤드 4, 문맥 64, 배치 16)를 같은 초기 가중치와 같은 배치 열 303개로 두 학습기에 나란히 태웠다. 한쪽은 본 엔진이고, 다른쪽은 같은 수식을 numpy 와 cupy 의 표준 배열 연산만으로 옮긴 층별 표준 역전파 기준 구현이다. 기준 구현은 본 엔진의 역방향 코드를 호출하지도 복제하지도 않으며, 그 기울기는 float64(64비트 부동소수점) 중심 유한차분과 먼저 대조했다. 파라미터 10종 전부에서 살아있는 좌표 30개의 상대오차 최대가 1.45e-05다.

측정

300스텝 학습에서 두 손실 궤적이 겹친다. 스텝 0 손실은 완전히 같고(9.5692), 처음 20스텝의 손실 차 최대가 9.54e-07, 300스텝 뒤 차이가 1.19e-07이다(상대 0.00%). 최종 손실은 둘 다 1.1039이며 같은 배치 열을 돈 numpy CPU 기준도 1.1039다. 실행 시간은 스텝당 본 엔진 11.8ms, 기준 구현 같은 GPU 47.9ms, 기준 구현 numpy CPU 80.1ms다.

그림 6-3. 표준 역전파와의 나란한 학습, 궤적 일치와 실행 시간



왼쪽은 300스텝 손실 궤적으로, 본 엔진(실선)과 표준 역전파 기준 구현(점선)이 겹쳐 하나로 보인다. 손실 차 최대가 1e-06 미만이라 수동 유도 전치가 표준 역전파와 같은 기울기임을 학습 전체에서 보인다. 오른쪽 실행 시간에서 기준 구현은 융합 커널이 없는 미최적화 참조 구현이므로, 차이는 전치 방식의 알고리즘 우위가 아니라 커널 융합 등 구현 최적화의 효과다.

이 대조의 범위를 명시한다. 이번 소형 대조가 덮는 경로는 임베딩, 토플리츠 시간혼합, 필터 사다리, RMSNorm, 로우랭크 헤드이고, 게이트와 궁합 주의집중과 바이토큰과 fp16(16비트 반정밀도) 헤드와 FFT 시간혼합과 순환행렬 채널 혼합 경로는 덮지 않는다. 실행 시간 비교는 최적화된 자동미분 프레임워크와의 비교가 아니며 이 소형 설정에 한정된다. 궤적 재현에 필요한 갱신 회계도 밝힌다. 위치 임베딩은 아담이 아니라 확률적 경사하강이고, 필터 치우침도 확률적 경사하강인데 유효 학습률이 채널 수로 나뉘며, 거리커널과 정규화 게인의 아담은 배치 평균이 아니라 합 기울기를 받고, 토큰 임베딩은 배치에 등장한 열만 갱신하는 열단위 아담이다.

재현

이 소절의 나란한 학습은 부록 C의 준비를 마친 뒤 다음 한 줄로 재생된다.

```
cd probe && python3 paper1_small_contrast_probe.py
```

출력에서 기준 구현의 유한차분 대조(살아있는 좌표 30개, 상대오차 최대 1.45e-05), 스텝 0 손실 일치(9.5692), 처음 20스텝 손실 차 최대(1e-06 미만), 최종 손실(양쪽 1.1039), 스텝당 실행 시간이 본문 수치와 대응한다.

7. 이 엔진의 확정 이득

정확한 기울기라는 점만 보면 backprop 과 값이 같으므로 "수동으로 유도하는 이유는 무엇인가"라는 물음이 남는다. 본절은 그 답으로 네 가지 이득을 근거와 함께 제시한다. 이 가운데 둘은 구조에서 곧바로 나오는 사실이고 하나는 실측이다.

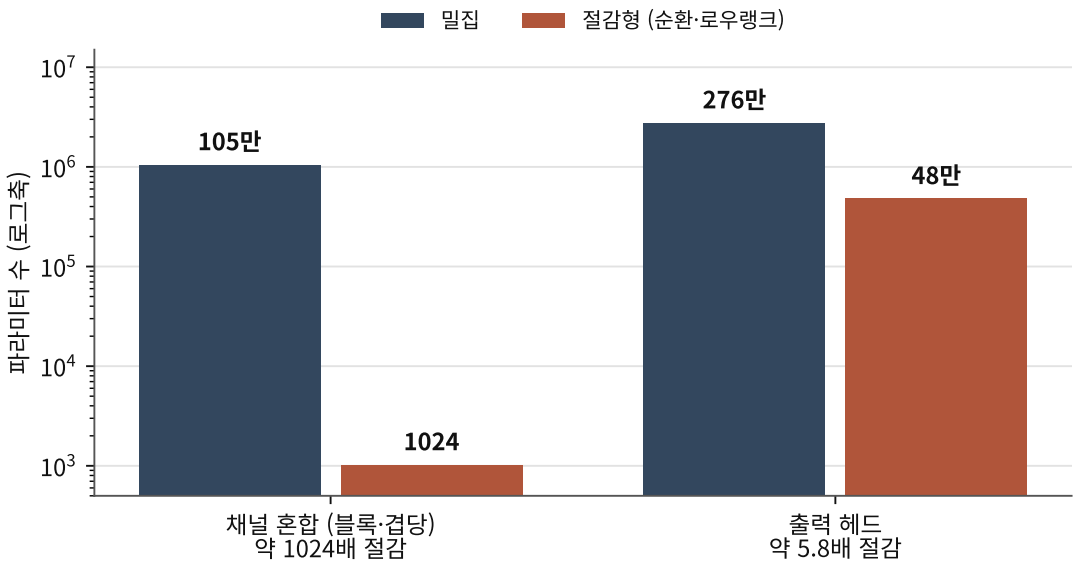
확정 이득

이득 1 — 비표준 연산을 정확히 학습(가능성 이득, 가장 핵심). 자동미분 라이브러리는 순환행렬 채널 혼합이나 분배함수($\div\Sigma$) 없는 어텐션 같은 비표준 원시에 대해 정확한 전치를 기본 제공하지 않는다. 본 엔진은 각 연산의 닫힌형 전치를 수동으로 유도해 이런 연산을 학습 루프에 정확히 물릴 수 있다. 분배함수 없는 혼합은 학습 경로에 실재하고, 순환행렬 채널 혼합 세 식은 수반 정합을 유한차분 대조로 검증했다(6절 프로브). 즉 정확도가 같은데도 이 엔진을 쓰는 이유는 자동미분으로는 학습에 못 무는 구조적 연산을 학습할 수 있기 때문이다. 이 자유가 후속 논문의 어텐션, 아키텍처 설계를 가능하게 한다.

확정 이득

이득 2 — 파라미터 절감(구조적 사실). 채널 혼합을 밀집 행렬($H \times H$) 대신 순환행렬(커널 H)로 두면 파라미터가 블록,겹당 약 1024배 감소한다($H = 1024$: 약 105만 대 1024). 출력 헤드도 로우랭크 분해로 $V \times H$ 대신 $(V + H) \times R$ 만 쓴다. 어휘 $V \approx 2700$, $H = 1024$, $R = 128$ 이면 약 5.8배 감소한다($2700 \times 1024 \approx 276$만 대 $(2700{+}1024) \times 128 \approx 48$만). 이는 측정이 아니라 구조에서 곧바로 계산되는 값이다(그림 7-1).

그림 7-1. 파라미터 절감 (밀집 대 순환, 밀집 헤드 대 로우랭크 헤드)

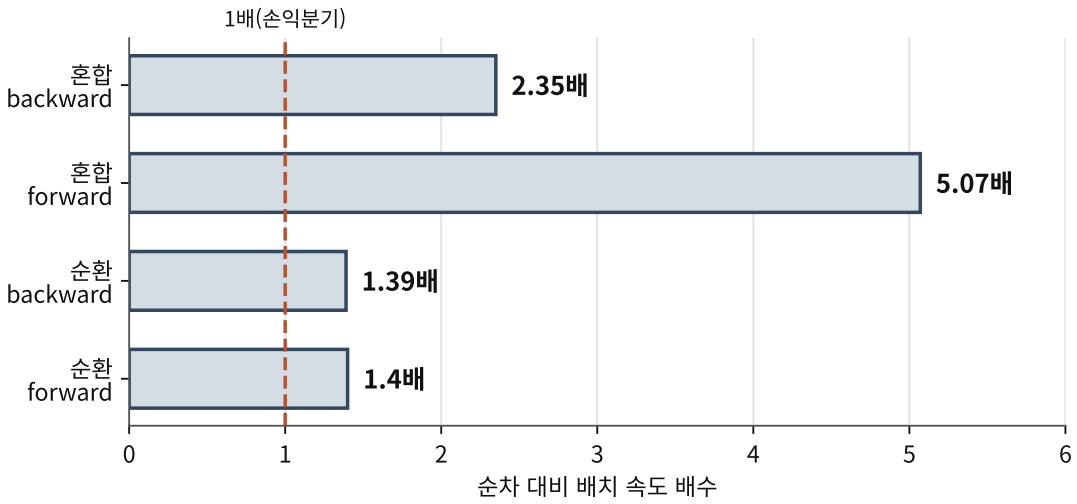


채널 혼합은 밀집 행렬 대비 약 1024배, 출력 헤드는 밀집 대비 약 5.8배 파라미터가 적다. 두 값 모두 구조에서 곧바로 계산되는 사실이다.

확정 이득

이득 3 — 블록축 배치 속도(실측). 여러 블록을 순차로 도는 대신 블록축으로 묶어 한 번에 실행하면 집필 환경의 그래픽 처리장치에서 원시 연산 단독 측정으로 순환행렬 채널 혼합은 약 1.40배, 상대 거리 혼합은 약 5.07배 빨라진다($H = 512$, 배치,문맥 곱 4096, 가운데 블록 15개 기준, 그림 7-2). 신용 경로를 직접 소유하므로 이런 배치 재구성이 자유롭다.

그림 7-2. 블록축 배치의 속도 이득



순차 실행 대비 블록축 배치 실행의 속도 배수이다. 상대 거리 혼합이 5.07배로 가장 크고 순환행렬 채널 혼합도 1.4배 빨라진다. 세로선 1배는 손익분기이며, 값이 클수록 배치가 순차보다 빠름을 뜻한다.

확정 이득

이득 4 — 신용 경로의 완전한 소유. 자동미분이 감춘 역방향 경로를 직접 코드로 갖고 있으므로, 정확한 기울기의 급소만 전정밀도로 지키고 나머지는 반정밀도로 처리하는 혼합정밀, 나온 낱말의 임베딩 열만 갱신하는 열 희소 갱신처럼 연산 단위의 통제가 가능하다. 이는 라이브러리 내부에 신용 경로가 숨은 표준 방식으로는 어렵다.

8. 독자적 기여와 다른 편과의 경계

본고의 독자적 기여는 학습 엔진 그 자체이다. 구체적으로 (1) 정확한 전치 신용 사슬 골격, (2) 연산별 닫힌형 전치(RMSNorm,순환,Toeplitz)의 유도, (3) 세 항목 검증 프로토콜과 실측, (4) 파라미터별 최적화 부호 규약과 혼합정밀,희소 갱신 규율, (5) 위 7절의 확정 이득이다. 이 다섯은 다른 어느 편에서도 다시 주장하지 않는, 본고에만 귀속되는 몫이다.

반대로 다음은 본고의 몫이 아니며 각 후속 편으로 넘긴다. 경계를 분명히 해 같은 결과를 두 번 주장하지 않는다.

표 8-1. 본고와 후속 편의 경계

주제	어느 편의 몫인가
정확한 전치 신용 사슬, 검증, 연산별 전치, 최적화 규율	본고(제1편)
기울기를 전혀 안 쓰는 되새김 자기조직화 학습	제2편
분배함수 없는 어텐션의 성능 비교, 순환행렬 채널 혼합의 아키텍처 스케일,깊이 설계, 직교 데이터,연산 토큰의 성능	제3편
월드먼저 발달 커리큘럼과 창발 축 구조 판독	제4편
자기 분포,기하에서 나오는 제어 신호	제5편
로우랭크 헤드를 이용한 어휘 확장 응용	후속(효율 주제)

요컨대 본고는 어떻게 자동미분 없이 정확하게 학습하는가라는 방법과 그 이득만 다루고, 그 위에 무엇을 얹어 무엇을 얻는가는 각 후속 편이 자기 실측으로 다룬다.

9. 한계와 다음 측정 과제

본고의 기준선은 1인 개발, 자작 정확한 전치 역전파(자동미분 없이 수동 유도) 엔진, 소비자 그래픽 처리장치 한 대라는 레퍼런스 클래스(reference class, 견주어야 할 실제 비교 대상 집단)이다. 아래는 이 방법의 결함이 아니라, 이 레퍼런스 클래스에서 이미 측정된 강점 위에 이어질 다음 측정 과제이다.

- 비교 범위 : 6.1절이 동일 소형 아키텍처의 표준 역전파 기준 구현과 나란한 학습을 측정해 손실 궤적 일치와 최종 손실 동일을 보였다. 다만 그 기준은 미최적화 참조 구현이라, 파이토치 같은 최적화된 자동미분 프레임워크와의 실행 시간과 메모리 비교, 그리고 본 설정 크기에서의 비교는 다음 과제이며 총 학습 시간 우열은 여전히 미증명이다.
- 측정 범위 : 직접 측정한 것은 소형 설정의 세 검증 항목과 나란한 학습 대조와 그래픽 처리장치 배치 속도이다. 정식 설정의 검증값과 대규모 학습 성능은 다음 측정 과제로 남는다.
- 규모 : 더 큰 모델,데이터로의 확장과 하류 과제(downstream task) 평가는 향후 과제이다.
- 연산별 기여 분리 : 각 수동으로 유도한 전치 연산이 최종 성능에 얼마나 기여하는지의 개별 제거 실험(ablation)은 아직 없다.

10. 결론

본고는 자동미분 라이브러리를 수동으로 유도한 닫힌형 정확한 전치로 대체해 언어모델을 학습하는 엔진을 제시했다. 계산되는 기울기는 backprop 과 수치적으로 동일하며(정합성 비율 중앙값 0.9997, 정규화 스케일은 깊이와 무관하게 5.93), 자동미분 그래프가 없어 순환행렬이나 분배함수 없는 연산 같은 비표준 원시를 학습 루프에 정확하게 물릴 수 있다. 그 대가로 얻는 이득은 명확하다. 자동미분이 못 무는 연산을 학습하는 자유, 채널 혼합 약 1024배 파라미터 절감, 블록축 배치 최대 5.07배 속도, 그리고 신용 경로의 완전한 소유이다. 남은 과제는 동일 아키텍처의 표준 backprop 대비 총 학습 시간 비교와 규모 확장이다.

11. 후속 논문 예고

본고의 엔진은 하나의 공통 토대이며, 그 위에 얹히는 각 주제를 후속 논문으로 나눈다. 각 후속 편은 본고의 정확한 전치 신용 사슬과 검증 프로토콜을 인용하고 자기 주제의 실측 근거를 따로 확립한다.

후속 논문 예고

- 제2편 — 되새김 자기조직화 : 본고와 달리 기울기를 전혀 계산하지 않고 임베딩을 이웃끼리 당겨 뭉치는, 실제로 경사가 없는 학습을 다룬다. 감쇠와 당김의 균형으로 목표값 없이 평형이 창발함을 보이고, 내부 되먹임이 표현을 압축하되 없던 관계를 창조하지는 못한다는 음성 결과를 실증한다.
- 제3편 — 직교 데이터,연산 토큰과 콘볼루션 스케일 구조 : 본고가 예시로만 언급한 순환행렬

채널 혼합의 아키텍처적 역할과 깊이,스케일 조합을 정면으로 다루고, 토큰을 데이터면과 연산면 두 직교 임베딩으로 나누는 바이토큰(bi-token) 구조가 라벨 없이 "연산이 곧 가중치"로 창발함을 방향 일관성으로 정량화한다.

- 제4편 — 월드먼저 발달 커리큘럼 학습 : 언어 이전에 감각,공간,논리 축을 먼저 발달시키는 학습 순서가 필터에 축 구조를 형성한다는 가설을, 본고 엔진으로 실제 학습해 창발한 축을 비침습으로 판독한다.
- 제5편 — 자기 분포,기하 모니터링에서 창발하는 제어 신호 : 모델이 자기 출력 분포와 임베딩 기하를 읽어 확실,모호,모름의 라우팅과 학습 동기 신호를 뽑아내는 부분을 측정된 통계량으로만 서술한다.

참고문헌

[1] Hinton, G. (2022). The Forward-Forward Algorithm: Some Preliminary Investigations. arXiv:2212.13345.

[2] Lillicrap, T. P., Cownden, D., Tweed, D. B., Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. Nature Communications 7:13276.

[3] Nøkland, A. (2016). Direct Feedback Alignment Provides Learning in Deep Neural Networks. NeurIPS. arXiv:1609.01596.

[4] Lee, D.-H., Zhang, S., Fischer, A., Bengio, Y. (2015). Difference Target Propagation. ECML/PKDD. arXiv:1412.7525.

[5] Scellier, B., Bengio, Y. (2017). Equilibrium Propagation: Bridging the Gap between Energy-Based Models and Backpropagation. Frontiers in Computational Neuroscience 11:24.

[6] Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., Hinton, G. (2020). Backpropagation and the brain. Nature Reviews Neuroscience 21:335-346.

[7] Whittington, J. C. R., Bogacz, R. (2017). An Approximation of the Error Backpropagation Algorithm in a Predictive Coding Network with Local Hebbian Synaptic Plasticity. Neural Computation 29(5):1229-1262.

[8] Bartunov, S. et al. (2018). Assessing the Scalability of Biologically-Motivated Deep Learning Algorithms and Architectures. NeurIPS. arXiv:1807.04587.

[9] Karpathy, A. (2024). llm.c: LLM training in simple, raw C/CUDA. GitHub. https://github.com/karpathy/llm.c

부록 A. 기본 하이퍼파라미터

표 A-1. 정식 설정

항목	값	항목	값
채널 폭 H	1024	문맥 길이 T	128
블록 수 N	16	블록당 채널 변환 겹수	2
혼합 헤드 수	16	로우랭크 헤드 랭크 R	128
SGD 학습률(위치,바이어스)	0.2	채널 필터 Adam 학습률	0.002
Adam 학습률	0.002	순전파당 갱신	1

부록 B. 측정 환경

표 B-1. 측정에 사용한 컴퓨터 사양

구분	사양
그래픽 처리장치(GPU)	NVIDIA GeForce RTX 3090 Ti (24 GB)
운영체제	Ubuntu 24.04.4 LTS
소프트웨어	Python 3.12.3, numpy 2.5.1, scipy 1.18.0, cupy 14.1.1, CUDA 13.3.1
측정 항목	수반 정합,정규화 안정성(소형 설정), 블록축 배치 속도(GPU)

6절과 7절의 실측 수치는 위 환경에서 측정했다.

부록 C. 재현 안내

본고의 모든 실측 수치는 공개 재현 패키지로 재생된다. 아래 다섯 단계를 순서대로 하면 된다.

1. 요구 환경. NVIDIA 그래픽 처리장치(GPU)와 CUDA, Python 3.12 가 필요하다. 파이썬 패키지는 압축을 푼 폴더에서 다음으로 설치한다. cupy 는 설치된 CUDA 버전에 맞는 배포판을 사용한다(예: CUDA 13 은 cupy-cuda13x).

```
pip install -r requirements.txt
```

2. 코드 받기. 압축 묶음(zip)을 내려받아 푼다. 저장소로 받으려면 github.com/ubmscoin/banya/banya_ai_paper_code 폴더다.

```
wget https://ubmscoin.github.io/banya/banya_ai_paper_code/
banya_ai_paper_code.zip
unzip banya_ai_paper_code.zip && cd banya_ai_paper_code
```

3. 모델 받기. 얼린 모델 3개(합계 471MB)는 제노도 doi.org/10.5281/zenodo.21383724 에 있다. 다음 한 줄이 내려받기와 무결성 검증(sha256, 파일 지문 대조)을 함께 한다. 손으로 받으려면 위 기록에서 세 파일을 받아 model 폴더에 넣고 `sha256sum -c checksums.sha256` 으로 검증한다.

```
cd model && bash download.sh && cd ..
```

4. 말뭉치 만들기. 말뭉치는 원문 배포 없이 생성기가 만든다. 다음 한 줄이 banya_world_data 폴더에 말뭉치 23종을 전부 생성한다.

```
cd corpus_build && bash build_all.sh && cd ..
```

5. 실측 재생. probe 폴더의 프로브가 각 절의 수치를 재생한다. 본고의 수치는 다음 두 프로브가 담당한다. 출력에 목표값이 괄호로 같이 인쇄되므로 본문과 바로 대조된다.

프로브	재생하는 수치	확인할 출력
paper1_engine_probe.py	6절 세 검증과 5.2절 순환 수반 정합	자동미분 로드 False, 스케일 비 5.93, 정합성 중앙값 0.9997, 순환 기울기 비율 중앙값 1.000000
paper1_small_contrast_probe.py	6.1절 나란한 학습	스텝 0 손실 9.5692 일치, 20스텝 손실 차 최대 1e-06 미만, 최종 손실 1.1039, 실행 시간

패키지의 폴더 구성과 파일별 설명은 목차 페이지에 있다.

반야 연구 시리즈 제1편(엔진). 정확한 전치 신용 사슬로 자동미분을 대체한다. 6,7절 수치는 부록 B 환경에서 측정한 값이고, 정식 설정의 대규모 학습 성능은 다음 측정 과제로 남는다. 파라미터,구조 값은 사실이다. 표준 backprop 대비 총 학습 시간 비교는 향후 과제. 되새김, 직교 토큰,콘볼루션,스케일, 발달 커리큘럼, 창발 제어 신호는 각각 제2~5편에서 다룬다.