엣지 AI 환경에서 멀티코어 CPU 및 GPU 메모리 설정 최적화에 관한 실험적 연구

마준익[1], 이세인[2], 전찬욱[3], 문세환[4], 김상윤[5], 이경운[6], 조정훈[7]
[1, 2, 3]경북대학교 전기전자공학부 석사과정
[4, 5]경북대학교 전자공학부 졸업
[6, 7]경북대학교 전기전자공학부 교수
macs6848@knu.ac.kr, lsin07@knu.ac.kr, eddie3618@knu.ac.kr, wkdk777@knu.ac.kr,
carz21@knu.ac.kr, kwlee87@knu.ac.kr, jcho@knu.ac.kr

An Experimental Study on the Optimization of Multicore CPU and GPU Memory Settings in Edge AI Environments

Jun-Ik Ma[1], Se-In Lee[2], Chan-Wook Jeon[3], Se-Hwan Moon[4],
Sang-Yoon Kim[5], Kyung-Woon Lee[6], Jung-Hoon Cho[7]
[1, 2, 3]Dept. of Electrical and Electronic Engineering, Graduate School, KyungPook National University
[4, 5]Dept. of Electronic Engineering, KyungPook National University
[6, 7]Dept. of Electrical and Electronic Engineering, Graduate School, KyungPook National University

요 약

본 논문은 엣지(Edge) 환경에서 인공지능 모델의 추론 성능 최적화를 위한 병렬 연산 전략을 분석하였다. 성능이 상이한 세 가지 엣지 디바이스(Jetson Nano, Jetson Orin Nano, Jetson AGX Orin)와 총 6가지 대표 인공지능 모델(DNN, LSTM, GRU, AlexNet, MobileNet, ResNet)을 대상으로, 멀티코어 CPU 사용 개수 및 GPU 메모리 용량 제한을 독립 변수로 설정하여 실험을 수행하였다. 실험 결과, 재귀적 특성이 강한 모델(LSTM)은 멀티코어 CPU 활용 효과가 제한적이었고, CPU 코어 수 증가에 따른 성능 향상은 4개 이상의 코어부터 둔화되었으며, GPU 메모리 과다 할당은 오히려 성능 저하를 초래하는 것으로 나타났다. 이를 통해 모델 특성에 따른 최적 병렬화 전략 설정이 필수적이며, 무분별한 자원 증가는 오히려 비효율을 초래할 수 있음을 확인하였다. 본 논문은 엣지 디바이스 환경에서 인공지능 모델을 효율적으로 운용하기 위한 병렬 연산 설정에 있어, 단순한 자원 증가가 아니라 모델 특성에 따른 세밀한 최적화 전략이 필수적임을 실험적으로 입증하였다.

1. 서론

인공지능 기술의 급격한 발전은 고성능 환경뿐만 아니라, 성능이 제한된 엣지(Edge) 컴퓨팅 환경에서도 그 활용 범위를 빠르게 확대시키고 있다[1].

일반적으로 인공지능 모델의 추론 과정은 짧은 시간 내에 완료되지만, 모델의 구조가 크고 연산량이 많을 경우, 엣지 환경에서는 제한된 연산 자원으로 인해 추론 속도가 현저히 저하될 수 있다. 이에 따라, 엣지 환경에서 추론 속도 향상을 위한 다양한 성능 최적화 방안이 연구되고 있다[2].

대표적인 성능 개선 방법으로는 모델 구조의 최적화가 있다. 이는 하드웨어 변경 없이도 일정 수준 이상의 효과를 기대할 수 있지만 모델 설계에 대한 깊은 이해와 복잡한 수정 작업이 필요하다[3]. 또 다른 방법은 AI 가속기 및 FPGA와 같은 추가 하드웨어의 사용이다. 이는 상대적으로 직관적인 해결책이지만, 비용 증가 및 물리적 설치 환경의 제약이라는 단점을 동반한다[4].

이와 비교해 멀티코어 CPU나 GPU를 활용한 병렬 연산은 비교적 구현이 용이하고, 다양한 환경에서 적용 가능하다는 장점이 있다[5]. 그러나 이러한 병렬화 전략이 항상 선형적인 성능 향상을 보장하지는 않으며, 모델의 구조적 특성이나 연산 패턴에 따라 효과는 크게 달라질 수 있다[6].

본 실험에서는 성능이 상이한 세 가지 Edge 디바이스에서 6가지 인공지능 모델을 대상으로, 멀티코어 CPU 수 또는 GPU 메모리 용량을 독립 변수로 설정하여 추론 성능에 미치는 영향을 실험적으로 분석한다.

실험의 목적은 Edge AI 환경에서 인공지능 모델을 효율적으로 운용하기 위한 병렬 연산의 기준을 실험적으로 도출하는 것이다. 더 나아가, 신경망 구조 및 디바이스 성능의 차이에 따른 성능 변화를 분석하고, 실험에 포함되지 않은 모델에도 적용 가능한 참고 자료를 제시하고자 한다.

2. 실험 환경

본 실험에서는 각 모델 유형의 구조적 특성 차이가 병렬 연산 성능에 미치는 영향을 비교 분석하기 위해, 6가지 모델(DNN, LSTM, GRU, AlexNet, MobileNet, ResNet)을 선정하여 실험을 수행하였다. 각 모델은 해당 유형 내에서 연산량, 병렬화 가능성 등에서 차이를 보이도록 구성되었으며, 이를 통해 다양한 조건에서의 성능 변화를 분석하고자 하였다.

실험에 사용된 엣지 디바이스는 NVIDIA의 Jetson 시리즈로, 소형 디바이스인 Jetson Nano, 중간급 성능의 Jetson Orin Nano, 고성능 디바이스인 Jetson AGX Orin을 선택하였다. 이를 통해 동일한 소프트웨어 환경에서 하드웨어 성능의 차이가 병렬 연산에 미치는 영향을 비교할 수 있도록 하였다.

DNN 및 CNN 모델의 경우 일반적인 이미지 분류 성능을 측정하기 위해 CIFAR-10 데이터셋을 사용하였으며, RNN 계열 모델(LSTM, GRU)은 시계열 및 문맥 의존성이 중요한 IMDB 리뷰 데이터셋을 활용하여 텍스트 기반의 분석을 수행하였다.

실험은 두 가지 조건을 독립 변수로 설정하여 각각의 모델에 대해 추론 속도의 변화를 측정하였다:

1. 멀티코어 CPU 사용 개수에 따른 추론 성능 변화
2. GPU 메모리 사용량 제한에 따른 추론 성능 변화

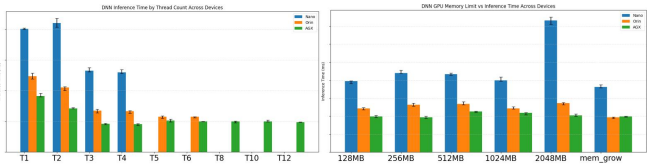
모든 실험은 Docker 기반의 컨테이너 환경을 활용하여 소프트웨어 실행 환경의 일관성을 확보하였다. 초기 실행 시 발생할 수 있는 캐시 준비 및 메모리 할당에 의한 편차를 줄이기 위해 가장 느린 결과와 가장 빠른 결과를 제외한 10회의 실행 결과 평균값을 최종 성능 지표로 활용하였다.

Jetson_Nano		Jetson_Orin		Jetson_AGX	
실험환경		실험환경		실험환경	
운영체제	Ubuntu 18.04	운영체제	Ubuntu 20.04	운영체제	Ubuntu 20.04
플랫폼	L4T 32.7.5	플랫폼	L4T 35.4.1	플랫폼	L4T 35.4.1
CUDA	10.2 버전	CUDA	11.4 버전	CUDA	11.4 버전
cuDNN	8.2 버전	cuDNN	8.6.0 버전	cuDNN	8.6.0 버전
Jetpack	4.6.4 버전	Jetpack	5.1.2 버전	Jetpack	5.1.2 버전
docker	20.10.21 버전	docker	24.0.7 버전	docker	24.0.7 버전
기기	Jetson_Nano	기기	Jetson_Nano_Orin	기기	Jetson_AGX_Orin

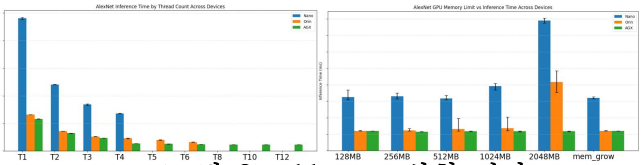
<그림 1> 실험 환경

본 실험은 그림1에 표기된 환경으로 진행되었다.

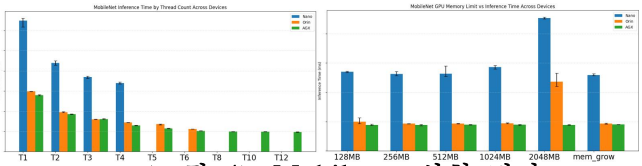
3. 실험 결과



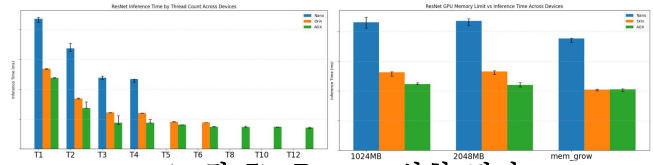
<그림 2 DNN 실험 결과>



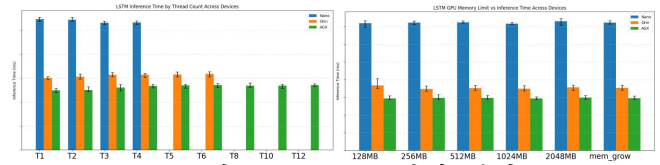
<그림 3> Alexnet 실험 결과



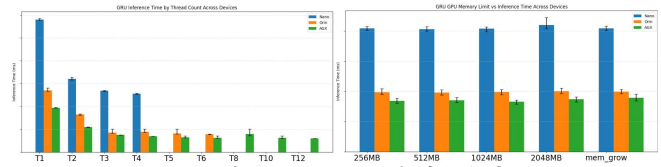
<그림 4> Mobilenet 실험 결과



<그림 5> Resnet 실험 결과



<그림 6> LSTM 실험 결과



<그림 7> GRU 실험 결과

MobileNet과 같은 경량화된 모델은 디바이스 간 성능 차이를 줄이는 데 기여하기 때문에 Orin과 AGX 간의 성능 차이가 상대적으로 미미하였다. 반면, 경량화가 이루어지지 않은 DNN 및 LSTM 모델에서는 Orin과 AGX 간에 뚜렷한 성능 차이가 나타났다. 이 결과를 통해 경량화가 충분히 이루어진 모델을 사용할 경우, 상대적으로 성능이 낮은 디바이스를 사용하는 것이 더욱 효율적이라는 사실을 확인할 수 있다.

디바이스별 CPU 코어 수는 Nano가 4개, Orin이 6개, AGX가 12개로 구성되어 있다.

3-1. 주요 현상 및 원인 분석

실험 결과, 다음 세 가지 주요 현상이 관찰되었다:

(1) GRU와 LSTM 간의 성능 차이

(2) CPU 코어 수 증가에 따른 성능 변화의 비선형성

(3) GPU 메모리 사용량 초과 시 성능 저하

각 현상의 원인을 분석하고, 관련 이론적 배경을 논의한다.

3-1-1. GRU와 LSTM 간의 성능 차이

GRU와 LSTM은 모두 재귀적 특성을 가지는 모델이다. 그러나 본 실험에서는 CPU 코어 수가 증가할 때 GRU에서는 성능 향상이 나타난 반면, LSTM은 그렇지 않았다. 이는 두 모델의 연산 의존성 차이에 기인한다. LSTM은 내부 상태가 순차적으로 갱신되어야 하며, 이전 연산이 완료되어야 다음 연산이 가능하다[7]. 이에 비해 GRU는 비교적 독립적으로 연산이 이루어질 수 있어 병렬화 효율이 높다[8].

따라서 GRU는 CPU 코어 수 증가에 따른 병렬 처리 이점을 효과적으로 활용한 반면, LSTM은 연산 의존성으로 인해 동일한 이점을 얻지 못했다.

3-1-2. CPU 코어 수 증가와 성능 향상의 비선형성

DNN, CNN 모델군에서는 CPU 코어 수가 증가함에 따라 성능이 향상되었지만, 그 향상은 선형적이지 않았다. 이론적으로는 코어 수 증가에 비례하여 병렬 처리 능력이 향상될 것으로 예상되나, 실제로는 다음과 같은 요인들로 인해 제한이 발생한다:

캐시 병목 현상: 공유 캐시 또는 메모리 대역폭 부족으로 인해 데이터 접근 지연 발생[9]

동기화 비용 증가: 다수의 쓰레드 간 동기화 오버헤드로 인한 지연[9]

메모리 병목 현상: 메모리 컨트롤러 및 인터커넥트 포화로 인한 성능 저하[9]

특히 CPU 코어 과다 사용은 전력 소모 및 발열 증가를 초래하여 시스템 안정성을 저하시킬 수 있으며[10], 연산 외의 작업들이 중단되면서 병렬 처리의 이점이 감소하게 된다[11].

3-1-3. GPU 메모리 사용량과 성능 저하

GPU 메모리 사용량이 일정 수준을 초과하면 모든 모델에서 성능 저하가 관찰되었다. 이는 다음과 같은 요인으로 설명할 수 있다:

PCIe 스왑 비용 증가: GPU 메모리가 가득 찼을 경우, 추가 데이터는 느린 CPU 메모리로 스왑되어야 하며, 이 과정에서 속도 저하와 대기 시간 증가가 발생한다[12].

메모리 관리 복잡성 증가: 메모리 사용량이 많아질수록 메모리 할당/해제, 캐시 유지 작업이 복잡해져 오버헤드가 증가한다[13].

실험에서는 mem_grow 옵션을 사용했을 때 대부분 가장 빠른 추론 속도를 기록했으나, mem_grow는 메모리 조각화(fragmentation)를 유발할 수 있다.

4. 결론

실험 결과 및 분석을 통해 다음과 같은 주요 결론을 도출할 수 있었다.

첫째, 모델 구조에 따른 병렬화 효과 차이가 뚜렷하게 나타났다. 재귀적 특성이 강하고 연산 의존성이 높은 모델(RNN, LSTM)은 멀티코어 CPU를 활용하더라도 성능 향상을 기대하기 어려웠으며, 오히려 단일 코어를 사용하는 것이 에너지 효율 측면에서 바람직했다. 반면, GRU처럼 연산 의존성이 낮은 모델은 CPU 병렬 처리를 효과적으로 활용할 수 있어, 적절한 수준의 멀티코어 사용이 성능 향상에 기여함을 확인하였다.

둘째, CPU 코어 수 증가에 따른 성능 향상은 비선형적이었다. CPU 코어 수를 무작정 증가시키는 것은 다양한 제약 요인으로 인해 성능 향상 폭이 급격히 둔화되었으며, 특히 4개 이상의 코어를 사용할 경우 추가적인 성능 이득이 매우 제한적이었다. 따라서, 특정 작업에 필요한 CPU 코어 수를 적절히 조절하는 것이 효율적인 성능 최적화 전략임을 확인 할 수 있었으며, 실험에 사용된 엣지 환경에서는 4개 내외의 CPU 코어를 활용하는 것이 성능과 에너지 효율성 측면에서 최적의 선택임을 확인할 수 있었다.

셋째, GPU 메모리 사용량과 성능 간의 역설적 관계가 확인되었다. GPU 메모리를 과도하게 사용할 경우 오히려 성능 저하가 발생하였다. mem_grow 옵션을 사용할 경우 최상의 추론 속도를 얻을 수 있었으나, 장기 실행 환경이나 대규모 모델에서는 메모리 조각화로 인한 문제가 발생할 가능성이

존재하였다. 이에 따라, 안정적인 성능 확보를 위해서는 상황에 따라 mem_grow 사용 여부를 신중히 선택하고, 필요시 GPU 메모리를 명시적으로 할당하는 전략이 요구된다.

본 연구를 통해 엣지 디바이스 환경에서 인공지능 모델을 효율적으로 운용하기 위한 병렬 연산 설정에 있어, 단순한 자원 증가가 아니라 모델 특성에 따른 세밀한 최적화 전략이 필수적임을 실험적으로 입증하였다.

향후 연구에서는 보다 다양한 신경망 구조와 엣지 디바이스를 대상으로 추가적인 실험을 수행함으로써, 엣지 AI 최적화 전략을 더욱 구체화할 예정이다.

ACKNOWLEDGMENT

이 논문은 2025년도 정부(산업통상자원부)의 재원으로 한국산업기술진흥원의 지원을 받아 수행된 연구임 (RS-2024-00415938, 2024년 산업혁신인재성장지원사업)

참 고 문 헌

[1] Raghubir Singh, Sukhpal Singh Gill, “Edge AI: A survey,” Internet of Things and Cyber-Physical Systems, vol. 3, pp. 71 - 92, 2023. https://doi.org/10.1016/j.iotcps.2023.02.004

[2] Surianarayanan, C., Lawrence, J. J., Chelliah, P. R., Prakash, E., & Hewage, C. (2023). A Survey on Optimization Techniques for Edge Artificial Intelligence (AI). Sensors, 23(3), 1279. https://doi.org/10.3390/s23031279

[3] Mills, K. G., Han, F. X., Salameh, M., Rezaei, S. S. C., Kong, L., Lu, W., Lian, S., Jui, S., & Niu, D. (2021). L2NAS: Learning to Optimize Neural Architectures via Continuous-Action Reinforcement Learning. In Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), Virtual Event, QLD, Australia. ACM. https://doi.org/10.1145/3459637.3482360

[4] Ollivier, S., Li, S., Tang, Y., Cahoon, S., Caginalp, R., Chaudhuri, C., Zhou, P., Tang, X., Hu, J., & Jones, A. K. (2023). Sustainable AI Processing at the Edge. IEEE Micro, 43(1), 19 - 28. https://doi.org/10.1109/MM.2022.3220399

[5] Morgan, N., Yenusah, C., Diaz, A., Dunning, D., Moore, J., Heilman, E., Roth, C., Lieberman, E., Walton, S., Brown, S., et al. (2024). On a Simplified Approach to Achieve Parallel Performance and Portability Across CPU and GPU Architectures. Information, 15(11), 673. https://doi.org/10.3390/info15110673

[6] Ofenbeck, G., Steinmann, R., Caparros, V., Spampinato, D. G., & Püschel, M. (2014). Applying the Roofline Model. In 2014 IEEE 28th International Parallel and Distributed Processing Symposium Workshops (pp. 76 - 85). IEEE. https://ieeexplore.ieee.org/abstract/document/6844463

[7] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder - Decoder for Statistical Machine Translation. arXiv preprint arXiv:1406.1078. https://arxiv.org/abs/1406.1078

[8] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv preprint arXiv:1412.3555. https://arxiv.org/abs/1412.3555

[9] Hill, M. D., & Marty, M. R. (2008). Amdahl's Law in the Multicore Era. IEEE Computer, 41(7), 33 - 38. https://doi.org/10.1109/MC.2008.209

[10] Bhat, G., Gumussoy, S., & Ogras, U. Y. (2017). Power-Temperature Stability and Safety Analysis for Multiprocessor Systems. ACM Transactions on Embedded Computing Systems (TECS), 16(5s), Article 145. https://doi.org/10.1145/3126567

[11] Hager, G., Treibig, J., Habich, J., & Wellein, G. (2016). Exploring performance and power properties of modern multi core chips via simple machine models. Concurrency and Computation: Practice and Experience, 28(2), 189 - 210. https://doi.org/10.1002/cpe.3180

[12] Kehne, J., Metter, J., & Bellosa, F. (2015). GPUswap: Enabling Oversubscription of GPU Memory through Transparent Swapping. In Proceedings of the 11th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '15), ACM, pp. 69 - 82. https://doi.org/10.1145/2731186.2731192

[13] Guo, C., Zhang, R., Xu, J., Leng, J., Liu, Z., Huang, Z., Guo, M., Wu, H., Zhao, S., Zhao, J., & Zhang, K. (2024). GMLake: Efficient and Transparent GPU Memory Defragmentation for Large-scale DNN Training with Virtual Memory Stitching. In Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '24), Volume 2, ACM, pp. 450 - 463. https://doi.org/10.1145/3620665.3640423