KNOM Review '25-01 Vol.28 No.01
https://doi.org/10.22670/knom.2025.28.1.1

자율주행 엣지의 Continual Learning 적응 성능 분석

이 지 은◆, 테오도라 아두푸*, 김 윤 희°

A Performance Analysis of Continual Learning on an Edge for Autonomous Driving

Jieun Lee◆, Theodora Adufu*, Yoonhee Kim°

요 약

자율주행 시스템의 안정성이 중요시되면서 로컬 엣지(Edge)에서 GPU를 기반으로 딥러닝을 적용하는 연구가 진행중이다. 그러나 자율주행용 엣지 딥러닝 학습 과정에서 실시간으로 유입되는 새로운 데이터에 대한 적응력이 뛰어나지 못한다는 문제점이 제기되고 있다. 본 논문에서는 실시간 적응 능력 개선과 정확도 향상을 위해 엣지 상에서 Continual Learning(CL, 연속학습) 알고리즘을 딥러닝 학습 과정에 도입할 것을 제안한다. NVIDIA Jetson AGX 기반 엣지 환경에 EWC와 Rehearsal을 적용한 실험을 통해 CL 알고리즘이 정확도를 손해보지 않으면서 실시간 환경에 빠르게 적응할 수 있음을 보였다.

Key Words : Edge Computing, Continual Learning, EWC, Rehearsal, Resource-constrained Systems, On-device Learning

ABSTRACT

As the reliability of autonomous driving systems becomes increasingly critical, research has been actively conducted on applying deep learning using GPUs as local edge devices. However, a major challenge remains: edge-based deep learning for autonomous driving often lacks the ability to quickly adapt to newly incoming data in real time. In this paper, we propose integrating Continual Learning (CL) algorithms into the deep learning training process on edge devices to enhance real-time adaptability and improve accuracy. Through experiments applying EWC and Rehearsal on an NVIDIA Jetson AGX-based edge environment, we demonstrate that CL algorithms can rapidly adapt to real-time environments without compromising accuracy.

※ 이 논문은 2025년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (과제번호: RS-2025-24534879).
◆ First Author : Sookmyung Women's University, Department of Computer Science, mariewldms@sookmyung.ac.kr
° Corresponding Author : Sookmyung Women's University, Department of Computer Science, yulan@sookmyung.ac.kr
* Sookmyung Women's University, Department of Computer Science, theoadufu@sookmyung.ac.kr
논문번호 : KNOM2025-01-01, Received July 14, 2025; Revised August 08, 2025; Accepted August 25, 2025

Ⅰ. 서 론

최근 고도화된 GPU를 기반으로 자율주행용 엣지 컴퓨팅에 딥러닝을 도입하기 위한 방안이 활발하게 제안되고 있다. 특히, NVIDIA Jetson AGX Orin [1], NVIDIA Jetson AGX Xavier [2] 등 AI 추론이 가능하도록 GPU 및 자원 환경이 최적화된 자율주행용 엣지에서 딥러닝을 수행한 사례가 보고된다. 안전성과 실시간성이 중요하게 여겨지는 자율 주행 환경에서, 다양한 특성의 교통 데이터를 학습하기 위한 엣지에서의 딥러닝은 필수 불가결한 요소로 여겨지며 실 시스템에 도입하기 위한 접근이 시도되고 있다.

일반적으로 딥러닝 모델을 통해 학습 및 추론이 이루어지는 고성능 GPU와 달리 엣지는 GPU의 제한된 연산 능력과 메모리 용량의 한계로 인해 딥러닝 시간, 성능 측면에서 제약이 따른다. 문제 해결을 위해 엣지에 적합한 추론 알고리즘 최적화 기법 [3]이 제안되고 있지만, 추론이 아닌 새로운 데이터 학습 과정에서의 문제를 해결하지는 못한다. 또한 자율주행용 엣지는 실시간 유입되는 다양한 교통 데이터에 적응해야 하기 때문에 빠른 적응이 더욱 중요하다.

Continual Learning(CL, 연속 학습)은 동적인 환경에서 새로운 과제를 학습했을 때 기존 과제에 대한 성능이 급격하게 저하되는 파괴적 망각(Catastrophic forgetting) 현상을 방지하기 위해 제시된 방법이다 [4]. 그림 1은 CL의 이론적 핵심 및 학습 파이프라인을 도식화한 구조도다 [4]. 그림 1의 a는 CL이 다른 성격의 데이터 유입에 적응하는 것의 중요성을, 그림 1의 b는 CL의 기준점(안정성, 일반화, 유연성)을, 그림 1의 c는 학습 과정에 방법론별 CL이 도입되는 지점을, 그림 1의 d는 이론과 방법을 기반으로 CL의 실제 적용 과정을 보인다. CL 방법론을 학습 과정에 적용해 과거 데이터 학습 기억을 보존하면서 새로운 데이터를 학습하는 과정의 균형을 맞추는 것이 CL의 핵심 주안점이다. CL 알고리즘은 Computer Vision(CV), Natural Language Processing(NLP), Reinforcement Learning(RL)등의 분야에 적용될 수 있으며, 최근 연구에서는 실시간 얼굴 이미지 검출 및 보호 [5] 등 실생활 문제 해결 목적으로 활용되었다. 이처럼 CL은 유연성(Plasticity)을 기반으로 실시간으로 유입되는 새로운 데이터에 신속하게 적응할 수 있다

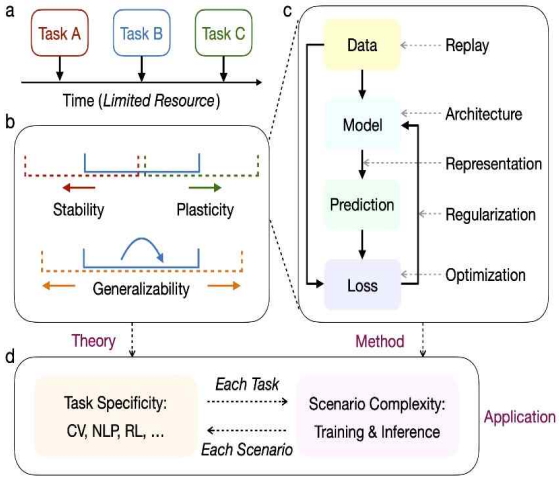

그림 1. CL(Continual Learning)의 개념적 구조 [4]
Fig. 1. A conceptual framework of continual learning

는 장점을 가져 자원 및 환경 제한된 상황에서 해결책이 될 수 있다. 따라서 본 논문은 추론용 엣지의 딥러닝 과정에서 새로운 데이터에 빠르게 적응할 수 있는 방법인 CL 알고리즘의 성능 분석을 제안한다.

본 논문의 구성은 다음 순서를 따른다. 1장의 서론에 이어 2장은 본 논문의 주제와 관련된 연구 및 연구 동기를 서술한다. 3장은 선행 연구를 기반으로, 실험에 활용할 대표적인 CL 알고리즘과 신경망 모델을 소개한 뒤 성능 측정 기준이 될 주요 지표를 제시한다. 4장에서 실험 과정 및 분석 결과를 설명하고, 5장은 결론을 도출한 후 향후 연구를 기술한다.

Ⅱ. 관련 연구 및 연구 동기

본 절에서는 엣지에서 GPU를 활용해 딥러닝을 수행한 선행 연구 사례를 서술하고, 기존 CL 연구를 기반으로 자율주행 엣지에 CL 알고리즘을 도입해야 할 필요성을 보인다.

1. 관련 연구

기존 연구 [6,7]에서는 엣지 내 GPU를 활용해 딥러닝 연산을 수행하고 추론했다. 데이터셋을 통해 학습된 YOLOv8, YOLOv9 모델을 기반으로 추론용 임베디드 컴퓨터인 NVIDIA Jetson AGX Orin에서 추론 과정을 거쳐 모델별 성능을 분석했다 [6]. 또한, 엣지 컴퓨팅의 자원 한계를 개선하기 위해 ConvNet, ResNet50, EfficientNet-B0을 변형한

새로운 아키텍처(CorPAR)를 제안해 추론 과정에서 초당 22프레임을 처리하고, TP Rate(True Positive Rate)가 71% 에 달해 고성능 GPU와 비슷한 성능을 보였다 [7]. 이처럼 엣지는 딥러닝 기반 추론은 가능하지만, 모델 학습은 고성능 GPU에서 진행되었다는 점에서 학습은 미흡하다.

2. 연구 동기

지속적으로 변화하는 환경에 노출되는 자율주행 엣지는 정확한 객체의 주행 위치 예측을 위해 실시간으로 유입되는 데이터에 적응해야 한다. 그러나 엣지 환경은 GPU 연산 성능과 메모리가 제한적이기 때문에 고성능 GPU와 유사한 정확도의 성능을 내기 위해서 더 많은 시간 자원이 필요하다. 이러한 상황에서 CL이 효과적인 해결책이 될 수 있다. CL은 새로운 데이터의 유입에 유연하게 적응하면서 짧은 시간 안에 비슷한 정확도를 도출해 자율주행 엣지 상 자원 제약을 완화할 수 있다.

Ⅲ. CL 기법 분석

본 절에서는 CL 알고리즘 Elastic Weight Consolidation(EWC), Rehearsal을 단순 학습 방식에 적용한 방식을 서술하고, Convolutional Neural Network(CNN) 기반 이미지 분류 모델을 설명한다. 마지막으로 실험 결과를 분석을 위한 주요 성능 지표를 서술한다.

1. CL 알고리즘

그림 1의 c는 다섯 종류의 CL 방법론이 학습 과정에서 도입되는 구간을 표기한다. CL 방법론에는 Regularization-Based Approach, Replay-Based Approach, Optimization-Based Approach, Representation-Based Approach, Architecture-Based Approach가 있다 [4]. 본 연구는 대표적인 CL 기법 중 Regularization-Based Approach와 Replay-Based Approach 방식을 활용한다.

EWC는 Regularization-Based Approach에 해당하는 알고리즘이다. 기존 loss에 정규화된 term을 더해 모델의 가중치 변화를 최소화하는 방식으로, 중요한 가중치일수록 페널티로 값이 변화하지 않아 이전 기억을 보존한다 [8]. EWC를 백본(Backbone) 모델에 적용하는 과정에서 배치 단위로 학습이 수행될 때마다 수식 (1) 방식과 같이 새로 들어온 데이터에 대한 loss $L_{\mathrm{new}}(\theta,\mathrm{D})$와 $F_i(\theta_i-\theta_i^*)^2$을 더한 값에 가중치 λ/2를 곱한 결과를 더해 loss를 갱신한다. F_i는 각 파라미터의 중요도를 의미하며, $(\theta_i-\theta_i^*)^2$는 파라미터의 변화량을 제곱한 결과다.

$$L_{EWC}(\theta,D)=L_{\mathrm{new}}(\theta,\mathrm{D})+\frac{\lambda}{2}\sum_{\mathrm{i}}\mathrm{F_i}(\theta_{\mathrm{i}}-\theta_{\mathrm{i}}^*)^2 \quad (1)$$

Rehearsal은 Replay-Based Approach에 해당하는 알고리즘이다. 버퍼에 기존 학습 샘플을 저장하고 새로운 학습이 진행될 때 현재 태스크와 함께 재학습하여 망각 현상(catastrophic forgetting)을 방지하는 방법이다 [9]. 리플레이 버퍼에 학습 값의 일부를 저장하고, 새로운 학습 데이터를 이용해 학습할 때 버퍼 내 샘플을 재학습한다. 따라서 수식 (2)의 방식으로 산출한 새로운 학습 내용 기반 loss $L(\theta,D_{\mathrm{new}})$와 버퍼 기반 loss $L(\theta,D_{mem})$에 가중치 μ를 곱한 값을 더해 최종 loss를 갱신한다 [10].

$$L_{Re}(\theta,D)=L(\theta,D_{\mathrm{new}})+\mu\mathrm{L}(\theta,\mathrm{D_{mem}}) \quad (2)$$

2. CNN 모델 선정

이미지 데이터셋을 학습하며 CL 알고리즘을 적용할 대표적인 합성곱 신경망 기반 이미지 분류 모델로 ResNet50과 EfficientNet_B0을 선정했다. ResNet50은 잔차 연결(residual connection)을 도입해 딥러닝 모델의 깊이를 효과적으로 늘릴 수 있도록 설계된 안정적인 대표적 CNN 모델이다 [11]. EfficientNet_B0은 모델의 깊이, 너비, 해상도를 균형 있게 확장하는 방식(compound scaling)을 통해 적은 파라미터로 높은 정확도를 달성하는 경량화된 최신 모델이다 [12].

3. 성능 지표

CL의 성능을 파악하기 위한 3가지 기준점으로 [4] 실험 결과를 분석할 성능 지표를 선정했다.

첫 번째로, 새로운 데이터셋이 학습되는 과정에서 이전 기억을 보존하는 안정성(Stability) 평가 지표로 Root Mean Squared Error(RMSE)를 활용한다. RMSE는 CL 알고리즘 적용 학습이 완료된 후 평가용 데이터셋의 일부를 기반으로 미래 위치를 예측하고, 이 예측값과 실제값의 차이를 측정하는 회귀 평가 지표이다.

두 번째로, 불규칙하고 다양한 입력에 대응할 수 있는지 확인하는 일반화(Generalizability) 지표로 딥

러닝 과정에서의 loss를 활용한다. 새로운 분포의 데이터셋이 투입되었을 때의 loss 변화를 기반으로 모델의 일반화 능력을 간접적으로 평가한다.

마지막으로, 새로운 데이터셋에 빠르게 적응하는지를 확인하기 위한 유연성(Plasticity) 지표로 Converge Step(CS)과 Loss Reduction Rate(LRR)를 활용한다.

$$CS = \min\left\{t \in [1, T] | \overline{L}_t \leq \epsilon\right\} \quad (3)$$

CS는 누적 평균 loss값의 추세에서 최종 평균 loss와 차이를 비교하며 오차가 ε이 되는 지점 최소 스텝을 확인해 기준점에 얼마나 빠르게 수렴하는지 판정한다. 수식 (3)에서 t는 스텝 변수, T는 전체 스텝 수, $\overline{L}_t$는 t 스텝의 누적 평균 loss 값이다.

$$LRR = \frac{\overline{\mathrm{L}}_{\max} - \overline{\mathrm{L}}_{\mathrm{T}}}{T - t_{\max}} \quad (4)$$

LRR은 수식 (4)와 같이 최대 loss값에서 최종 loss값을 뺀 값을 step 간격으로 나누어 절댓값 기울기를 이용해 개선 속도를 파악한다. $\overline{L}_{\max}$는 최대 누적 평균 loss, $\overline{L}_T$는 최종 누적 평균 loss, T는 전체 스텝 수, $t_{\max}$는 $\overline{L}_{\max}$인 지점의 스텝이다.

Ⅳ. 성능평가

본 절에서는 실험에 사용된 자율주행 차량의 모션 예측을 위한 데이터셋과 실험용으로 선정한 엣지 Nvidia Jetson Orin AGX에 대해 설명한다. 이후 시나리오 설계 과정 및 엣지에서 CL 성능 분석을 진행한 결과를 서술한다.

1. 데이터

실험은 Woven by Toyota [13] 에서 제공한 예측 데이터셋(prediction-dataset)을 사용했다. 해당 데이터셋은 zarr 포맷으로 제공되며, 자율주행 모빌리티 장치의 주변 환경을 평균 10Hz의 속도로 112시간동안 캡처한 것이다. 해당 데이터 셋에는 100개의 장면(여러 에이전트가 존재하는 도로 위의 연속적인 상황), 24838개의 프레임(한 시점의 스냅샷), 1893736개의 에이전트(자동차, 사람 등 도로 위의 객체), 316008개의 신호등이 존재한다. 해당 구조의 zarr 파일을 기반으로 l5kit 파이썬 라이브러리의 AgentDataset 객체를 통해 학습에 적합한 데이터셋을 추출했다. 데이터 산출에 필요한 설정 조건은 제공된 agent motion config.yaml 파일의 기준을 따랐으며 [14], 그 결과 111634개의 데이터셋이 산출되었다. 생성된 각각의 데이터셋은 하나의 에이전트에 초점을 맞춘 단일 에이전트 예측용 데이터셋이다. 데이터셋은 한 에이전트의 래스터화된 이미지, 현재 및 과거 위치, 미래 타깃 위치, 방향 각도, 차량 크기 정보 등 에이전트의 주행 상황에 대한 지표를 포함한다.

생성된 111634개의 데이터셋은 PyTorch의 DataLoader를 통해 배치 단위로 분류되어 ResNet50과 EfficientNet을 통해 학습된다. 백본 모델은 해당 에이전트의 미래 위치를 예측하며, 해당 값을 실제 미래 위치와 비교해 loss를 산출한다.

본 연구에서는 엣지 상에서 CL이 수행되는 상황을 구현하기 위해 전체 데이터셋을 동일한 크기의 세 개의 데이터셋으로 분할한다. CL 학습 알고리즘 적용을 위해 세 가지로 쪼개진 각 데이터셋은 명확히 구분될 수 있는 특징을 가져야 하므로, 에이전트의 속도를 기준으로 단계를 분류했다.

데이터셋은 세 가지의 다른 주행 환경에서 데이터를 추출하는 상황을 가정한다. 각 데이터셋의 속도(target velocities) 속성의 첫 번째 원소값을 기준으로 모든 데이터셋을 작은 값부터 차례대로 정렬한 뒤 세 개의 단계로 쪼갰다. 첫 번째 단계는 0.0m/s-0.2m/s의 속도로 주행하는 데이터셋의 집합이다(Low-velocity Task). 주차장에서 정지되어있거나 정차하는 환경에서 데이터를 수집하는 상황을 가정한다. 두 번째 단계는 0.2m/s - 2.09m/s의 속도로 주행하는 데이터셋의 집합이다(Moderate-velocity Task). 주행속도가 제한되어 있거나 정체되는 구간에서 서행하는 에이전트들의 데이터를 수집하는 상황을 가정한다. 마지막 단계는 2.09m/s - 30.99m/s의 속도로 주행하는 데이터들의 집합이다(High-velocity Task). 고속도로를 비롯한 일반적인 도로주행 상황에서 데이터를 수집하는 상황을 가정한다.

2. 실험 환경

엣지인 NVIDIA Jetson AGX Orin에서 데이터를 학습하고 추론했다. Jetson AGX Orin 시리즈는 AI 시스템을 엣지에서 구현할 수 있도록 설계된 임베디드 시스템이다. 학습 및 추론이 진행된 Jetson

AGX Orin 32GB는 Ampere Architecture 기반 GPU와 8코어 CPU가 하나의 32GB 메모리를 공유하는 SoC(System on Chip) 아키텍처다 [15]. 실제 AI 기반 안전운전 시스템, 지능형 교통 모니터링 시스템에 사용 [16]되므로 자율주행 데이터셋을 활용한 검증에 적합하다. 표 1은 본 연구에 사용된 엣지의 연산 성능 및 사양을 기술한다 [15].

표 1. 실험에 사용된 엣지(Jetson AGX Orin 32GB)주요 사양
Table 1. Configuration of the Edge AI (NVIIDA Jetson AGX Orin)

Specification	Details
Model	Jetson AGX Orin 32GB
AI Performance	200 TOPS(INT8)
GPU	NVIDIA Ampere architecture (1792 NVIDIA CUDA cores, 56 Tensor Cores)
CPU	8-core ARM Cortex-A78AE v8.2 64-bit CPU 2MB L2 +4MB L3
Memory	32GB 256-bit LPDDR5 204.8 GB/s
Power	15W-40W

3. 실험 시나리오

CL 알고리즘의 성능 평가 전 자원 활용 및 결과에 영향을 미치는 변인을 통제했다. 그림 2, 그림 3은 학습에 앞서 데이터셋 샘플링 비율과 배치(batch) 크기를 결정하기 위해 수행한 실험 결과다.

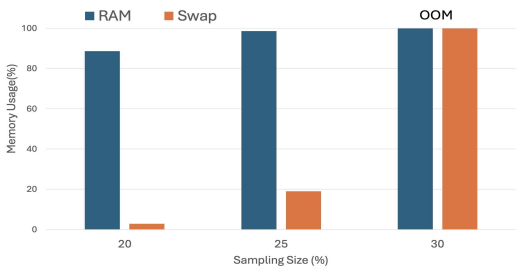

그림 2. 데이터셋 샘플링 크기에 따른 학습 시작 시점 RAM, Swap 점유율
Fig. 2. RAM and Swap Usage at Training Start According to Dataset Sampling Size

실험 전 데이터셋을 메모리에 적재하는 과정에서 주 메모리 용량이 32GB인 Jetson AGX Orin은 전체 데이터 적재가 불가능했다. 10% 단위로 샘플링한 결과 그림 2와 같이 30%의 데이터셋을 샘플링했을 때부터 더 이상 잔여 메모리가 존재하지 않아

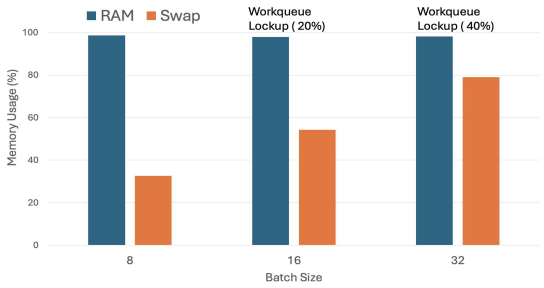

그림 3. 딥러닝 중 배치 크기에 따른 엣지의 최대 RAM, Swap 점유율
Fig. 3. Maximum RAM and Swap Usage on Edge During Deep Learning by Batch Size

OOM(Out Of Memory)으로 작업이 중단됐다. 25% 샘플링 결과 OOM은 발생하지 않았으나 RAM 점유율은 98.6%, Swap 메모리 점유율은 20%였다. Swap 메모리는 RAM이 가득 찼을 때 비활성 페이지를 디스크로 이동해 물리 메모리를 확보하기 위한 보조 수단으로, 메모리 부족을 완화하지만 RAM보다 속도가 느리다 [17]. 따라서 RAM 잔여 공간이 없는 상태에서 CPU, GPU기반 추론 시 속도가 느려져 12%의 잔여 RAM 공간이 있는 20% 데이터셋으로 수치를 고정했다. 실험에 활용된 전체 데이터셋은 22325개이며, 각 세 개의 단계는 전체 데이터의 1/3개인 7442개의 데이터셋을 포함한다.

일반적으로 모델 학습 과정에서 배치 크기는 GPU 메모리 할당량에 비례한다. 따라서 데이터셋을 7442개로 고정하고 배치 크기를 조정했다. 그림 3은 배치 크기에 따른 최대 메모리 점유율을 나타낸다. 학습 상황에서 배치 크기가 8일 때는 5번의 실험 중 5번 모두 정상적으로 학습 및 추론이 가능했다. 반면 배치 크기가 16일 때는 20%, 배치 크기가 32일 때는 40%의 확률로 워크큐 락업(Workqueue Lockup) [18,19]이 발생했다. 워크큐 락업은 8개의 CPU 코어 중 특정 CPU 내의 커널 워크큐 내 워커에서 특정 작업이 오랫동안 진행되어 대기한 채로 작업이 중단되는 버그다. 배치 크기가 16, 32일 때 Swap 메모리를 50% 이상 사용해 CPU가 Swap으로부터 데이터를 불러오는 과정에서 스레드가 중단되었다. 따라서 중단없이 결과를 추출하기 위해 배치 크기를 8로 결정했다.

4. 실험 분석

실험은 2가지 딥러닝 모델 ResNet50과

표 2. CL 알고리즘 학습 성능 지표 비교
Table 2. Continual Learning Performance Comparison

Backbone	Method	Average Loss			Accuracy	Training Time	CS (ε=10)	LRR
		Low velocity Task	Moderate velocity Task	High velocity Task	RMSE			
EfficientNet	Naive	0.62	2.46	35.86	12.03	592.66	406	0.21
	EWC	0.61	2.45	35.97	12.81	1135.94	443	0.35
	Rehearsal	1.19	3.89	146.07	8.30	1127.38	208	0.09
ResNet50	Naive	0.62	2.56	43.63	13.83	639.64	428	0.21
	EWC	0.62	2.56	42.68	11.96	1098.96	488	0.37
	Rehearsal	1.26	4.53	168.64	10.45	1192.22	181	0.08

EfficientNet에서 CL 미적용 단순 학습 방식, EWC, Rehearsal의 성능을 측정하여 총 6가지 상황에서 안정성, 유연성, 일반화 능력을 비교했다. 딥러닝 모델과 CL 알고리즘을 기반으로 속도별로 분할된 세 데이터셋을 학습한 뒤, 평가 과정에서 성능을 평가한다. 각 실험은 다섯 번씩 수행되었으며, 산출된 결과물은 다섯 번의 실험 결과의 평균값이다. CPU 연산량을 최소화하는 선에서 동일 조건으로 성능을 비교하기 위해 배치 단위의 데이터 로딩 과정에서 하나의 프로세스를 병렬 처리하지 않고 순차 처리했다.

표 2는 6가지 상황에 따른 단계별 loss, 학습이 진행된 학습 시간, 평가 시 정확도를 측정하는 RMSE, Converge Step(CS), Loss Reduction Rate(LRR)이다.

실험 결과, EfficientNet에서 EWC와 Rehearsal 모두 단순 학습 방식에 비해 학습 시간이 1.9배 길었고, ResNet50에서 EWC는 1.7배, Rehearsal은 1.8배 길었다. EWC는 추가로 Fisher Information Matrix를 적용하고, Rehearsal은 리플레이 함수를 기반으로 과거 loss를 추가로 계산해 새로운 loss를 갱신해 매 스텝마다 추가 연산으로 학습 시간이 약 2배 길어졌다.

안정성과 일반화 능력 분석을 위해 학습 과정의 세 단계에서 loss를 측정하고, 학습 종료 후 평가 과정에서 RMSE를 측정했다. loss는 배치 단위로 데이터가 로드될 때 매 스텝마다 산출되며, 표 2에 명시된 loss는 각 단계별 스텝 수인 930개 loss의 평균값이다. RMSE는 속도 별로 분류되지 않은 평가용 데이터셋에서 각 장면의 초반 100개 프레임을 학습한 후, 이어지는 50개 프레임을 예측해 예측 경로와 실제 경로의 차이를 측정해 계산한 평균의 제곱근 오차다. EWC는 EfficientNet, ResNet50에서 각 단계별 loss와 RMSE가 단순 학습 대비 ±2 이내로 미미했다. Rehearsal은 EfficientNet, ResNet50 모두 마지막 단계에서 loss가 단순 학습 방식보다 2배 크지만, RMSE는 3.7, 3.4 만큼 작았다. Rehearsal이 과거 데이터의 일부를 버퍼에 넣어 재학습하기 때문에 새 데이터에 대한 일반화 능력은 부족해 loss가 크고, 이전 기억을 보존하는 안정성이 뛰어나 평가용 데이터에 대한 정확도는 높아 RMSE가 낮은 트레이드오프를 보였다.

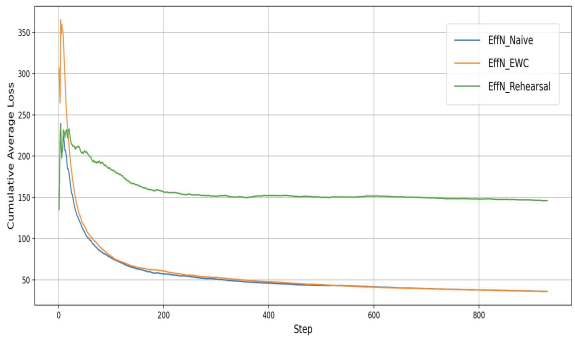

그림 4. 세 번째 학습 데이터셋(High-velocity Task)의 step 별 누적 평균 loss 변화 추이
Fig. 4. Cumulative Average Loss Trend per Step on High-Velocity Task (Task 3)

유연성과 자원 한계를 분석하기 위해 적응 속도, 개선 속도, 메모리 사용량을 측정했다. 첫 번째로, 적응 속도는 Rehearsal이 가장 빨랐다. 그림 4는 마지막 단계(High-velocity Task)에서 스텝별 누적 loss 평균값을 나타낸 그래프로, 노이즈를 제거해 변화 추이를 명확히 보여준다. 세 가지 방식은 930번의 스텝을 거치며 점차 학습되면서 특정 값으로 수렴되는 추이를 보이지만, 속도 및 최종 loss 값에는 차이가 있다. 스텝 1에서 loss는 Rehearsal은

135, 단순 학습 방식은 136으로 차이가 없지만 스텝 930의 최종 loss와 ε 차이 나는 값에 도달하는

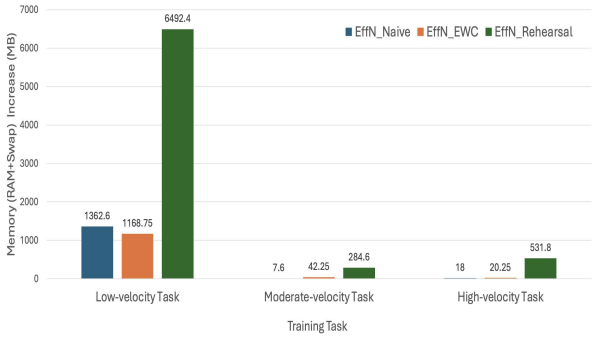

그림 5. 데이터셋 딥러닝 학습 과정에서의 각 단계별 엣지 메모리(RAM+Swap) 변화량(MB)
Fig 5. Memory Usage Change (RAM + Swap, MB) per Task During Deep Learning on Edge Device

데 걸리는 스텝은 Rehearsal이 더 짧았다. 기준점 ε을 10으로 설정 후 CS(Converge Step)를 측정한 결과 표 2와 같이 ResNet50, EfficientNet에서 모두 Rehearsal이 208, 181로 가장 작았다. Rehearsal은 새로운 학습 loss에 버퍼 기반으로 기존의 학습 loss가 더해져서 새로운 loss가 갱신되어 새로운 loss가 반영되는 비율이 단순 학습에 비해 적어 loss값 변동에 제약이 있기 때문이다. 따라서 실제 학습 과정에서 CS를 전체 스텝 수로 설정해도 전체 스텝 수인 930과 모델 최적화 정도에 차이가 없다.

두 번째로, 개선 속도는 EWC가 가장 뛰어났다. LRR을 계산했을 때, 표 2와 같이 EfficientNet과 Resnet50에서 EWC의 절댓값이 각각 0.35, 0.37로 가장 컸다. 그림 4에서 EWC는 첫 번째 스텝 loss가 단순 학습 방식보다 170 만큼 크고, 최대 누적 평균 loss가 350 이상으로 가장 크지만 930 스텝의 loss는 35로 동일하다. EWC는 이전 단계 학습으로 모델이 업데이트된 상황에서, 페널티로 큰 값의 가중치가 변하지 않아 초반 loss는 크지만 개선 속도가 빨라 단순 학습 방식과 동일한 누적 loss 평균으로 수렴하기 때문이다.

마지막으로 CL은 단순 학습 방식보다 비슷하거나 더 많은 메모리를 사용했다. 그림 5는 단계별 세 종류의 데이터가 투입되는 상황에서 전 단계와 비교했을 때 증가한 메모리량(RAM+Swap)을 나타낸다. 단순 학습, EWC, Rehearsal 모두 첫 번째 단계인 가장 느린 속도의 데이터셋이 투입되었을 때 모델 초기화 및 데이터셋 로딩으로 메모리 증가량이 가장 크다. 또한 이후로는 증가량이 10% 미만으로 크게 감소한다는 점은 동일하지만, 메모리 증가 정도에 차이가 있다. 세 단계에서 모두 EWC와 단순 학습의 RAM, Swap 메모리 증가량은 194MB, 35MB, 2MB 차이로 유사했다. EWC는 스텝마다 가중치에 따라 페널티를 조정한다. 매 스텝마다 추가적인 연산이 필요하지만 메모리를 더 사용하지 않아 추가 메모리 부담이 발생하지 않는다. 반면 Rehearsal은 단순 학습 방식보다 메모리 사용량이 5310MB, 277MB, 513MB 높았다. Rehearsal은 이전 데이터셋을 버퍼에 저장해 재학습하며 loss를 두 번 산출하기 때문에 추가적인 메모리 할당이 필요하기 때문이다. CL 알고리즘은 추가 loss 및 행렬 연산으로 CPU 및 GPU 연산량이 많아지면서 단순 학습 방식과 메모리를 비슷하게 사용하거나 더 많이 사용했다.

V. 결 론

본 논문은 자율주행 엣지에서 딥러닝 모델 학습 과정에 CL 알고리즘을 적용한 뒤 학습 과정에서 모델 오차와 추론 정확도를 실험했다. 이어서 엣지의 모델 적응 속도 및 메모리 사용량을 비교했다. 분석 결과 딥러닝 모델에 CL 알고리즘을 적용하면 정확도를 손해보지 않으면서 적응 속도, 개선 속도가 빨라져 실시간 적응 능력이 향상되었다. 향후 연구에서는 제한된 연산 자원과 실시간성이 요구되는 다양한 엣지 환경에서 안정적으로 CL이 수행될 수 있도록 CL 알고리즘의 엣지 환경별 적응성을 높이는 작업이 필요하다.

References

[1] NVIDIA, NVIDIA Jetson Orin, Retrieved July, 14,2025, from https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/

[2] NVIDIA, NVIDIA Jetson Xavior, Retrived July, 14,2025, from https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-series/

[3] Zihen Jiang, Tianqi Chen and Mu Li, “Efficient Deep Learning Inference on Edge Devices”, SysML18, Stanford, CA USA, pp. 1-3, April 2018

[4] Liyuan Wang, Xingxing Zhang, Hang Su and

Jun Zhu, "A Comprehensive Survey of Continual Learning: Theory, Method and Application", in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 46, no. 8, pp. 5362-5383, Aug. 2024

[5] 신종훈, "Faster R-CNN과 Continual Learning을 적용한 실시간 얼굴 검출 및 보호.", 충북대학교, 국내석사학위논문, pp.1-52, 2024

[6] 조홍석, 권희준, 정해영, 이수안, "NVIDIA Jetson AGX Orin 기반의 YOLOv8, YOLOv9 모델을 이용한 화재 탐지", 한국정보과학회, 학술논문집, pp.1764-1766, 2024-06-26

[7] J. N. Chaudhari, H. Galiyawala, P. Sharma, P. Shukla and M. S. Raval, "Onboard Person Retrieval System With Model Compression: A Case Study on Nvidia Jetson Orin AGX," in IEEE Access, vol. 13, pp. 8257-8269, 2025

[8] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinskaet al., "Overcoming Catastrophic Forgetting in Neural Networks," Proc. Natl. Acad. Sci., vol. 114, no. 13, pp. 3521–3526, 2017

[9] E. Verwimp, M. De Lange, and T. Tuytelaars, "Rehearsal revealed: The limits and merits of revisiting samples in continual learning"(2021), Retrived July, 14, 2025, https://arxiv.org/abs/2104.07446.

[10] Shengqin Jiang, Daolong Zhang, Fengna Cheng, Xiaobo Lu, Qingshan Liu, "DuPt: Rehearsal-based continual learning with dual prompts", Neural Networks, Volume 187, pp.1-10, 2025

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun "Deep Residual Learning for Image Recognition"(2015), Retrived July, 14, 2025, https://arxiv.org/abs/1512.03385

[12] Mingxing Tan, Quoc V.Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks"(2019), Retrieved July, 14, 2025, https://arxiv.org/abs/1905.11946v5

[13] Woven By Toyota, Retrived July, 14, 2025, https://woven.toyota/en/

[14] woven-planet, l5kit, Retrieved July, 14, 2025, https://github.com/woven-planet/l5kit/blob/master/examples/agent_motion_prediction/agent_motion_config.yaml

[15] NVIDIA, NVIDIA Jetson AGX Orin Series, Retrived July, 14, 2025, https://www.nvidia.com/content/dam/en-zz/Solutions/gtcf21/jetson-orin/nvidia-jetson-agx-orin-technical-brief.pdf

[16] Advantech, MIC-733-AO5A1, Retrived July, 14, 2025, https://iotmart.advantech.co.kr/

[17] Red Hat, Chapter 15. Swap Space, https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/7/html/storage_administration_guide/ch-swapspace

[18] NVIDIA, Kernel locked on Orin, Retrived July, 14,https://forums.developer.nvidia.com/t/kernel-locked-on-orin/245450/12

[19] The Linux kernel, Workqueue, Retrived July, 14, 2025, https://docs.kernel.org/core-api/workqueue.html

이 지 은 (Jieun Lee)

2022년 3월~현재 : 숙명여자대학교 컴퓨터과학과 학사과정

<관심분야> 엣지 컴퓨팅, 클라우드 컴퓨팅, 딥러닝, 임베디드 시스템

테오도라 아두푸 (Theodora Adufu)

2013년 : 가나대학교 컴퓨터과학 및 경제학과 졸업(학사).
2016년 : 숙명여자대학교 컴퓨터과학과 졸업(석사).
2022년~ : 현재 숙명여자대학교 컴퓨터과학과 박사과정.

<관심분야> 엣지 컴퓨팅, 퀀텀 컴퓨팅, 컨테이너, 클라우드 컴퓨팅, GPU 스케줄링, 딥러닝

김 윤 희 (Yoonhee Kim)

1991년 : 숙명여자대학교 전산학과 졸업(학사).
1996년 : Syracuse University 전산학과 졸업(석사).
2000년 : SyracuseUniversity 전산학과 졸업(박사).
1991년~1994년 : 한국전자통신연구원 연구원
2000년~2001년 : Rochester Institute of Technology 컴퓨터공학과 조교수.
2001년~2016년 : 숙명여자대학교 컴퓨터과학부 교수.
2017년~현재 : 숙명여자대학교 소프트웨어학부 교수.
<관심분야> 엣지 컴퓨팅, 퀀텀 컴퓨팅, 클라우드 시스템, 워크플로우 제어