🧡

[11주차] Continual Learning with Deep Generative Replay

0. Abstract

- **고질적 망각 문제와 해결 방안**
  - **고질적 망각 문제**: 인공지능 모델이 새로운 과제를 학습할 때 이전에 학습했던 내용을 잊어버리는 문제
  - **단순 데이터 재현의 한계**: 이전 데이터를 모두 재현하면 문제를 해결할 수 있지만, 대용량 메모리가 필요하고 실제 응용에서는 이전 데이터에 접근하기 어려운 경우가 많음
- Deep Generative Replay
  - 생성 모델(Generator): 이전 과제의 데이터를 생성하는 역할
  - 과제 해결 모델(Solver): 새로운 과제를 해결
  - 두 모델이 협력하여 이전 과제의 데이터를 생성하고 새로운 과제와 함께 학습
- 실험 결과
  - 이미지 분류 과제에서 Deep Generative Replay 프레임워크가 기존 방보다 우수한 성능을 보임
  - 이전 과제에 대한 성능 유지와 새로운 과제 학습 능력이 향상됨

1. Introduction

- **인간과 영장류의 지속적 학습 능력**
  - 인간과 큰 영장류는 평생 동안 새로운 기술을 배우고 지식을 축적할 수 있는 능력이 있음
  - 작은 척추동물인 설치류에서도 신경 연결이 1년 이상 지속되는 것이 관찰됨

    - 영장류는 새로운 정보를 통합하고 인지 능력을 확장하면서도 과거 기억을 크게 방해하지 않음

  ➡️ **시냅스 가소성과 안정성의 균형을 통해 가능한 유연한 기억 시스템의 결과**

- **인공 신경망의 고질적 망각 문제**
    - 반면 깊은 신경망에서는 '고질적 망각(catastrophic forgetting)' 현상이 발생
    - 새로운 과제를 학습하면 이전에 학습했던 과제의 성능이 급격히 저하됨
    - 이는 인공 신경망이 입력과 출력을 암묵적 매개변수로 표현하기 때문에, 새로운 목표를 향해 학습하면 이전 지식이 거의 완전히 망각되기 때문임
    - 이 문제는 깊은 신경망을 통해 **순차적으로 다중 과제를 학습하는 지속 학습**의 핵심 장애물이 되고 있음
- **기존 해결 방안의 한계**
    - 과거 데이터를 저장하고 재현하는 에피소딕 메모리 시스템이 제안됨
    - 이를 통해 새로운 과제 데이터와 함께 과거 데이터를 반복 학습하면 개별 과제 학습과 유사한 성능을 보임
    - 하지만 이 방식의 주요 단점은 과거 입력 데이터를 저장하고 재현하기 위해 큰 작업 메모리가 필요하다는 것
    - 실제 상황에서는 이전 데이터에 접근하기 어려운 경우가 많음
- **인간 뇌의 보완적 학습 시스템에서 영감**
    - 인간과 영장류는 제한된 경험에서도 새로운 지식을 학습하고 과거 기억을 유지할 수 있음 ➡️ 이는 **해마와 신피질로 구성된 상호작용하는 이중 기억 시스템** 때문
    - 해마 시스템은 최근 경험을 빠르게 인코딩하고, 이 기억 흔적은 수면이나 회상 중에 재활성 ➡️ 이후 신피질에서 반복적인 재활성화를 통해 기억이 공고화됨
    - 이러한 메커니즘은 강화 학습 에이전트 훈련에서 경험 재현(experience replay)에 영감을 주었음
- **생성 모델로서의 해마**
    - 최근 연구에 따르면 해마는 단순한 재현 버퍼 이상의 역할을 함
    - 기억 흔적의 재활성화는 유연한 결과를 초래할 수 있으며, 특정 기억 흔적을 자극하면 실제 경험하지 않은 거짓 기억이 생성될 수 있음
    - 이는 해마가 재현 버퍼보다는 생성 모델에 더 잘 비유될 수 있음을 시사함

    - 심층 볼츠만 기계나 변분 자동 인코더와 같은 심층 생성 모델은 관찰된 입력을 잘 모방할 수 있음
- **Deep Generative Replay 프레임워크의 제안**
    - 이에 착안하여 저자들은 과거 데이터를 참조하지 않고 순차적으로 딥 신경망을 학습하는 새로운 접근법을 제안함
    - 생성 적대 신경망(GAN) 프레임워크를 사용하여 과거 데이터를 모방하는 생성 모델을 학습함
    - 생성된 가짜 데이터와 과거 과제 해결기의 출력을 쌍으로 제공하여 새로운 과제 학습 시 함께 학습함
    - 이를 통해 새로운 과제를 학습하면서도 기존 지식을 유지할 수 있으며, 다른 모델에게도 생성된 입력-출력 쌍을 제공할 수 있음
    - 실제 데이터에 접근할 필요가 없어 프라이버시 문제가 있는 상황에서도 활용 가능

2. Related Works

2.1 Comparable methods

- 이전 과제 데이터에 대한 접근이 제한된 상황에 초점을 맞춘 연구들을 다룸
- 이러한 연구들은 이미 공고화된 가중치의 변경을 최소화하는 방식으로 네트워크 매개변수를 최적화하는 데 주력했음
- 드롭아웃, L2 정규화 등의 정규화 기법이 새로운 학습으로 인한 간섭을 줄이는 데 도움이 된다고 제안됨
- 또한 가중치의 중요도를 기반으로 특정 가중치를 보호하는 Elastic Weight Consolidation(EWC) 기법이 소개됨
- 다중 과제 순차 학습을 위해 입력에 가까운 층은 공유하고 출력층은 독립적으로 하는 등 네트워크 구조를 변경하는 방식도 시도되었음
- 학습률 감소 등의 기법도 망각 감소에 도움이 되는 것으로 알려졌음

2.2 Complementary Learning System(CLS) theory

- 고질적 망각 해결을 위해 보완적 네트워크 구조를 설계하는 연구들

- 이전 과제 데이터에 접근할 수 없는 경우, 메모리 네트워크에서 생성한 의사 입력과 의사 출력을 과제 네트워크에 제공하는 의사 재현 기법이 제안되었음
- 하지만 이 방식은 단순한 이진 패턴 결합 과제에서만 효과적이었고, 고차원 실세계 데이터에는 적용하기 어려웠음
- 본 논문에서 제안하는 생성 재현 프레임워크는 과거 입력 분포를 학습한 생성 모델을 사용한다는 점에서 기존 의사 재현 기법과 차별화됨

2.3 Deep Generative Models

- 심층 생성 모델
- 생성 모델은 관찰 가능한 샘플을 생성하는 모델을 의미하며, 특히 심층 신경망 기반 심층 생성 모델은 주어진 실제 분포에 대한 생성 샘플의 가능도를 최대화함
- 변분 자동 인코더(VAE)와 생성적 적대 신경망(GAN)은 이미지와 같은 복잡한 샘플을 모방할 수 있음
- GAN은 생성기와 판별기 간의 제로섬 게임을 정의하여, 생성기는 실제 분포를 최대한 모방하고 판별기는 생성 샘플과 실제 샘플을 구분하도록 학습함

3. Generative Replay

Definition 1️⃣: Task(Ti)

- 각 과제 Ti는 데이터 분포 Di에서 추출된 학습 데이터 (xi, yi)를 사용하여 모델을 최적화하는 것을 목표로 함
- 각 과제는 특정 데이터 분포에 대해 모델의 성능을 향상시키는 것

Definition 2️⃣: Scholar(H)

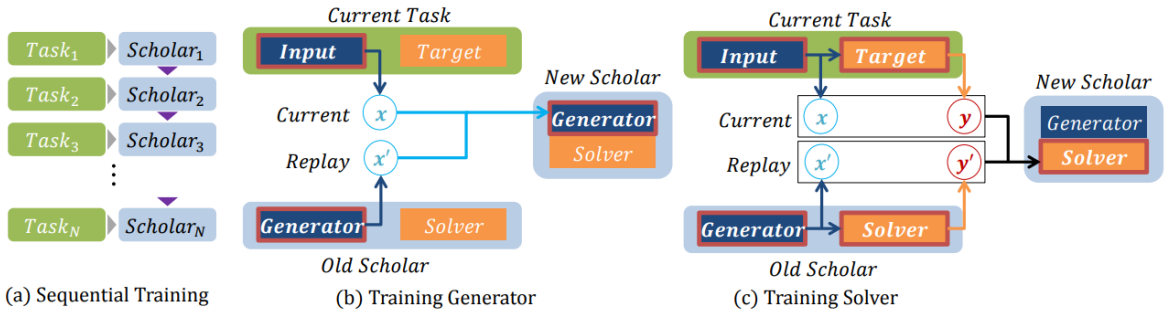
- Scholar H는 생성 모델 G와 과제 해결 모델 S로 구성된 튜플 <G, S>
- G는 실제와 유사한 샘플을 생성하는 생성 모델
- S는 과제를 해결하는 모델로, 매개변수 θ로 표현됨
- Scholar는 새로운 과제를 학습할 수 있고, 다른 네트워크에 자신의 지식을 전달 가능
    - 기존의 교사-학생 프레임워크와 다르게 학습과 전달이 **동시에** 가능

목표 함수

- Scholar S는 전체 과제 시퀀스 T에 대한 손실 함수 L의 기댓값을 최소화하는 것이 목표
- 모든 과제에 대한 손실의 합을 최소화하는 것이 목표

- 각 과제 Ti에 대해 해당 데이터 분포 Di에서 샘플링된 데이터로 학습함

3.1 Proposed Method



1 Sequential Training on Scholar Model

- 이전 Scholar 모델 Hn-1의 최신 복사본을 참조하여 새로운 Scholar 모델 Hn을 학습하는 것과 동일
- 전체 학습 절차는 N개의 Scholar 모델 (H1, H2, ..., HN)을 순차적으로 학습하는 것으로 볼 수 있음

2 Generator와 Solver의 독립적인 학습

- 새로운 Scholar 모델 Hn을 학습할 때, **생성기 G**와 **해결기 S**를 독립적으로 학습
- **생성기 G**는 현재 과제의 입력 x와 이전 과제의 재현 입력 x'을 혼합하여 학습
- **해결기 S**는 실제 데이터와 재현 데이터의 혼합 비율 r에 따라 입력과 타겟을 학습

3 학습 손실 함수

- 해결기 S의 학습 손실 함수 Ltrain(θi)는 실제 데이터 손실과 재현 데이터 손실의 가중합으로 구성됨
- 여기서 재현 데이터 손실은 이전 해결기 Si-1의 출력을 타겟으로 사용

4 테스트 손실 함수

- 테스트 시에는 실제 데이터 손실과 과거 데이터 분포 Dpast에 대한 손실의 가중합으로 계산됨
- 이는 모델이 원래 과제들에 대해 잘 수행하도록 하기 위함

5 생성기와 해결기의 구현

- **생성기 G**는 GAN 프레임워크를 사용하여 구현할 수 있음

- **해결기 S**는 과제 시퀀스를 해결할 수 있는 적절한 아키텍처를 가져야 함

3.2 Preliminary Experiment

	$Solver_1$ →	$Solver_2$ →	$Solver_3$ →	$Solver_4$ →	$Solver_5$
Accuracy(%)	98.81%	98.64%	98.58%	98.53%	98.56%

1️⃣ **실험 목적**

- 주요 실험에 앞서, 학습된 Scholar 모델만으로도 빈 네트워크를 학습시킬 수 있음을 보여주기 위함

2️⃣ **실험 데이터 및 설정**

- MNIST 필기체 숫자 데이터베이스를 사용하여 분류 작업을 수행
- 이전 Scholar 모델의 생성적 재현을 통해 처음부터 Scholar 모델 시퀀스를 학습

3️⃣ **실험 결과**

- 전체 테스트 데이터에 대한 분류 정확도를 Table 1에 나타냄
- 첫 번째 Solver는 실제 데이터로 학습되었고, 이후 Solver들은 이전 Scholar 네트워크로부터 학습되었음

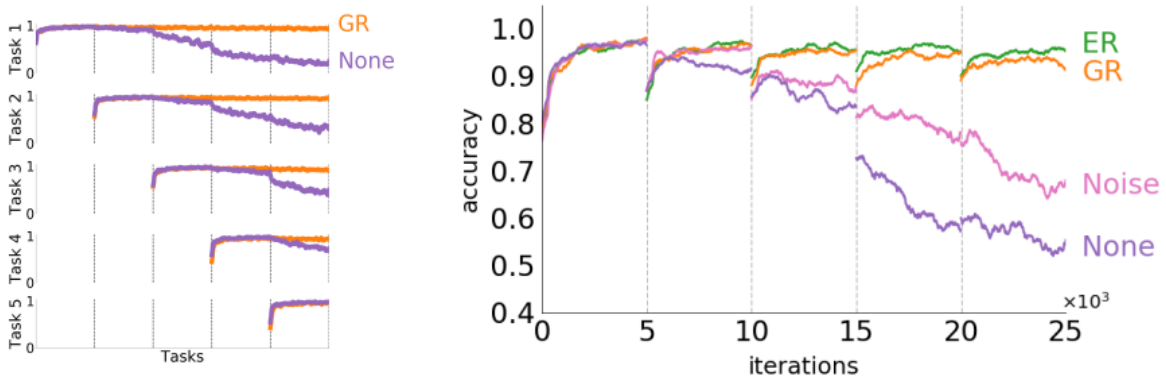
4️⃣ **결과 분석**

- 실험 결과, Scholar 모델이 정보를 손실하지 않고 지식을 전달할 수 있음을 확인

➡️ **학습된 Scholar 모델만으로도 새로운 네트워크를 효과적으로 학습시킬 수 있다!**

4. Experiments

4.1 Learning independent tasks



1 실험 목적

- 생성적 재현 프레임워크의 적용 가능성을 다양한 순차적 학습 환경에서 보여주는 실험
- 특히 서로 독립적인 과제들을 순차적으로 학습하는 실험을 수행함

2 실험 데이터 및 설정

- MNIST 필기체 숫자 데이터베이스를 사용하여 이미지 분류 문제를 해결
- 각 과제마다 입력 이미지의 픽셀 값을 고유한 무작위 순열로 섞어 과제 간 독립성을 높임
  ➡️ 이를 통해 네트워크의 기억 유지 능력을 측정할 수 있음

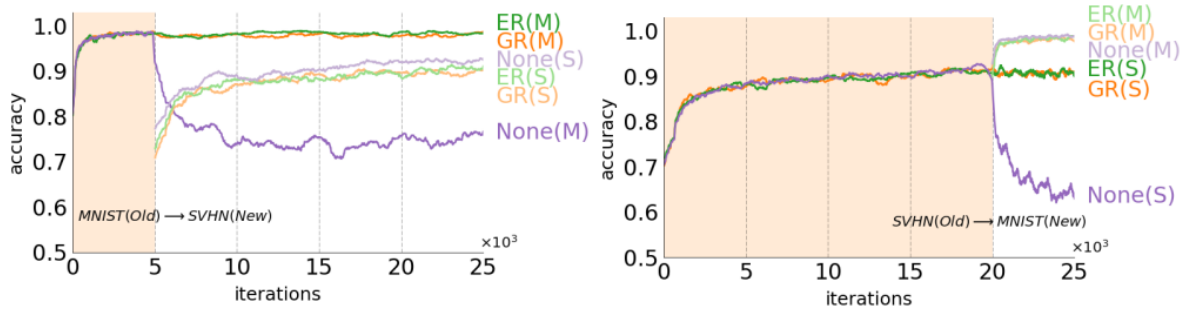
3 실험 결과

- 생성적 재현을 사용한 Solver(GR)는 이전 과제의 성능을 유지하며 순차적으로 학습할 수 있었음
- 반면 단순히 Solver만 학습한 경우(None)에는 치명적 망각이 발생했음
- 무작위 가우시안 노이즈를 재현한 경우(Noise)에도 성능 저하를 막을 수 ❌

4 결과 분석

- 생성적 재현 기반 학습(GR)은 이전 과제 데이터를 효과적으로 회상하여 성능 유지 가능

4.2 Learning new domains



(a) MNIST to SVHN

(b) SVHN to MNIST

1️⃣ **실험 목적**

- 독립적인 과제들을 순차적으로 학습하는 것보다는 서로 관련된 도메인들을 학습하는 것이 더 효율적일 수 있음
- 생성적 재현 프레임워크가 새로운 도메인을 학습할 때 어떤 성능을 보이는지 확인하고자 함

2️⃣ **실험 데이터 및 설정**

- MNIST 데이터셋과 Street View House Number (SVHN) 데이터셋을 순차적으로 학습하는 실험을 진행
- 두 데이터셋은 공간적 구조가 유사하여 관련성이 있는 도메인으로 볼 수 있음

3️⃣ **실험 결과**

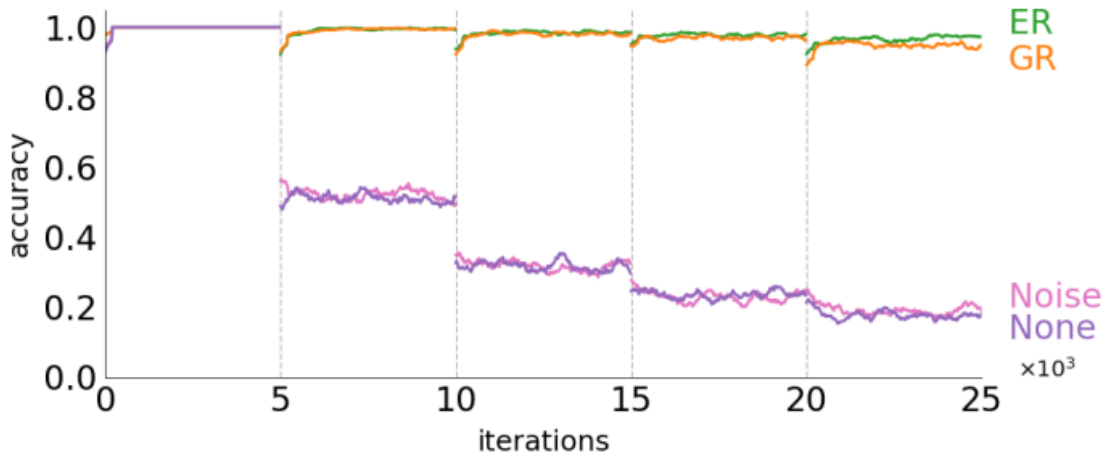
- 단순히 Solver만 학습한 경우(purple), 새로운 과제를 학습하면서 이전 과제의 성능이 크게 떨어짐
- 반면 생성적 재현을 사용한 Solver(orange)는 이전 과제의 성능을 유지하면서 새로운 과제도 학습할 수 있었
- 실제 과거 데이터를 재현한 경우(green)와 성능이 유사
- 새로운 과제만 학습한 경우(dim curves)에는 생성적 재현을 사용한 모델이 약간 더 좋은 성능을 보였

4️⃣ **결과 분석**

- 생성적 재현 프레임워크는 관련된 도메인들을 순차적으로 학습하는 상황에서도 효과적으로 작동함
- 이전 과제의 지식을 유지하면서 새로운 과제를 학습할 수 있었음

➡️ **이는 생성 모델의 품질이 과제 성능의 유일한 제약 요소라는 점을 다시 한번 보여줌**

4.3 Learning new classes



1️⃣ 실험 목적

- 입력과 타겟 데이터가 서로 다른 과제 간에 크게 편향된 경우에도 생성적 재현이 과거 지식을 잘 유지할 수 있는지 확인하고자 함
- 특히 한 번에 일부 클래스만 접근할 수 있는 상황을 가정하고 실험을 진행

2️⃣ 실험 데이터 및 설정

- MNIST 데이터셋을 5개의 상호 배타적인 부분집합으로 나눔
- 각 부분집합에는 2개의 클래스만 포함
- 네트워크를 이 부분집합들을 순차적으로 학습시킴

3️⃣ 실험 결과

- 단순히 네트워크를 독립적으로 학습시킨 경우(purple), 이전에 학습한 클래스를 완전히 망각함
- 과거 출력 분포만 복원한 경우(pink), 과거 지식을 유지하는 데 도움이 되지 않았음
- 입력과 출력 분포를 모두 복원한 생성적 재현 모델(orange)은 이전에 학습한 클래스를 잘 기억할 수 있었음

4️⃣ 결과 분석

- 과거 데이터가 완전히 폐기된다고 가정했기 때문에, 생성기는 현재 입력과 과거 생성기의 출력을 모두 모방하도록 학습되었음
- 이를 통해 생성기는 지금까지 학습한 모든 클래스의 입력 분포를 재현할 수 있게 됨
- 그 결과, 생성기가 생성한 샘플에는 모든 클래스가 균등하게 포함되어 있음

4. Discussion

- Deep Generative Replay의 **장점**
  - 입력-타겟 쌍을 저장된 네트워크에서 생성하므로, 이전 지식과 새로운 지식 간 균형을 쉽게 유지할 수 있음
  - 유연한 지식 전달이 가능하며, 이전 입력 공간이 생성기에 의해 복원되면 전체 성능을 보장받을 수 있음
- Deep Generative Replay의 **한계**
  - 생성기의 품질에 크게 의존하므로, SVHN 데이터셋에서는 성능 저하가 관찰되었음

논문에 대한 의견 및 의문점(꼭지)

➡️ 논문에서는 EWC, LwF, Deep Generative Replay 프레임워크를 혼합하면 더 나은 해결책을 제시할 수 있다고 언급했는데, 이에 대한 구체적인 방안이 궁금함