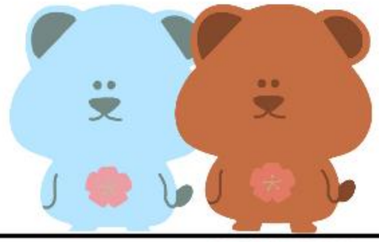
Continual Learning with Deep Generative Replay

인공지능학과 2277018 예서연

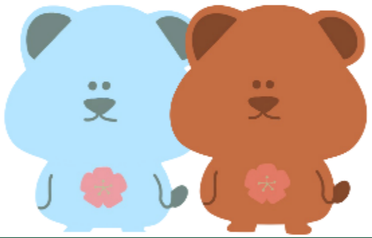
EWHA EURON

목차



EWHA EURON

Introduction



EWHA EURON

#01 Introduction

인간과 큰 영장류는 끊임없이 새로운 기술을 배우고 지식을 축적함
이를 가능케 하는 것이 유동적 기억 시스템

새로운 정보가 들어오면 통합하며, 과거 기억의 교란 없이 인지 능력을 확장함
-> 시냅스의 가소성과 안정성 사이 균형을 맞춤

이를 신경망으로 구현하고자 하는 것이 연속 학습(Continual Learning)
그러나 연속 학습은 파괴적 망각(Catastrophic Forgetting)으로 인해 어려움

파괴적 망각 이미 학습된 task에 대한 성능이 새로운 task에서 유지되지 않으며 오히려 떨어지는 현상

#01 Introduction

이를 해결하기 위해 기존 연구는 과거 데이터를 저장하는 단편적 기억 저장 시스템에 의존함

매 task의 input 샘플을 누적시켜 새로운 task에서 다루는 input 샘플과 함께 재생(replay)시킴
-> 신경망의 파라미터들이 여러 task에 함께 최적화되도록 함

성능 저하 방지(+)
각 task로만 학습된 개별 신경망과 비슷한 성능을 보임

한계(-)
- 지난 input을 저장하고 재생시키기 위해 넓은 메모리 공간을 필요로 함
- 실생활의 문제와 잘 부합하지 않을 수 있음

#01 Introduction

사람과 영장류는 제한된 경험에서도 새로운 지식을 배울 수 있으며, 과거 기억을 계속 보존함
인공신경망과 달리 인간에게는 기억 시스템이 개별적이고 상호작용이 가능한 형태로 존재함

- 해마: 최근의 경험을 빠르게 인코딩하며, 기억 추적 뉴런을 재활성화시킴
- 기억 추적 뉴런은 단기간 지속되며, 자는 동안 또는 의식적/무의식적 회상이 이루어질 때 재활성화됨
  (기억을 떠올릴 때 기억과 관련된 위치에서 발사되는 뉴런으로, 기억과 기억이 발생된 장소를 연결함)
- 신피질: 위와 같이 기억 추적 뉴런이 활성화되면 해마에서 인코딩된 경험이 재생되며, 이는 신피질 내에 통합됨

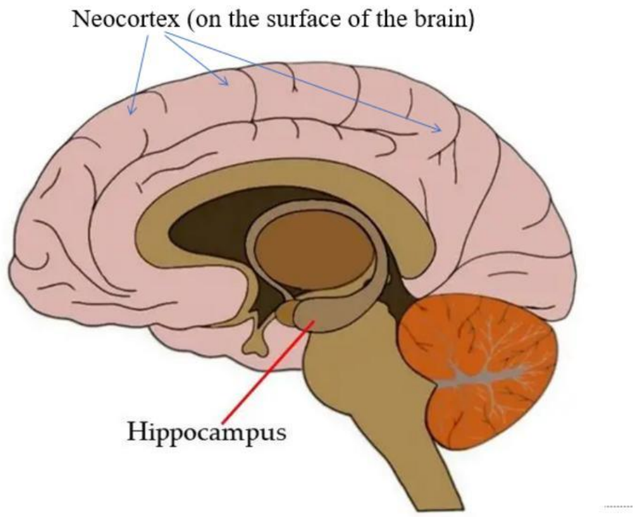
이러한 이중 기억 시스템을 모방한 것이 Complementary Learning System (CLS)

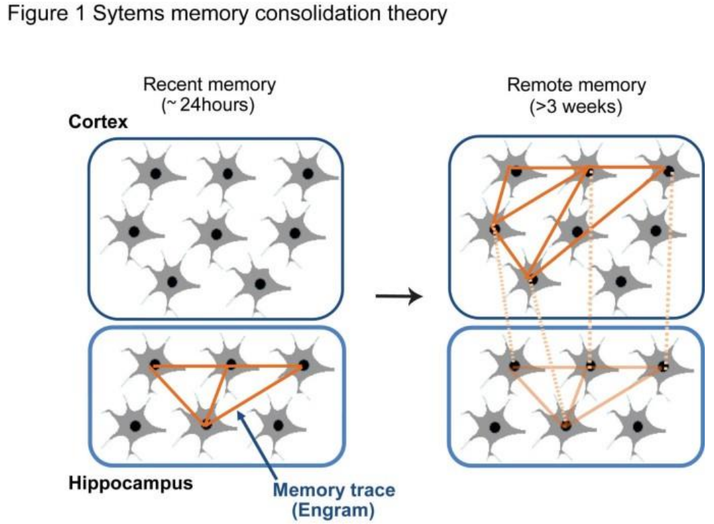
#01 Introduction

그러나, 기억 추적 뉴런을 재활성화하면 통합된 기억에 문제가 발생할 수 있음
(e.g., 경험한 적 없는 거짓 기억을 만들어냄)

-> 해마는 단순한 재생 버퍼 이상의 역할을 하며, 오히려 생성 모델에 가깝다는 것을 알 수 있음

본 논문에서 소개하는 프레임워크는 이러한 원리를 이용함!
generator를 통해 해마의 역할을 수행하는 것 (생성 모델을 통해 이전 기억의 재생이 이루어짐)





#01 Introduction

Deep Generative Replay 프레임워크

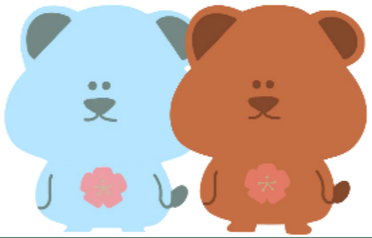
과거의 데이터를 보유하지 않고도 순차적으로 신경망을 학습시키기 위한 방법론
가짜 데이터를 생성하고 현재 데이터와 함께 재생시킴으로써 이전에 습득한 지식을 유지함

가짜 데이터는 GANs 프레임워크를 통해 생성됨
이렇게 생성된 데이터는 이전 task의 solver의 응답과 짝지어짐

이는 새로운 task의 데이터와 합쳐지고,
본 프레임워크를 구성하는 scholar 모델의 input이 되어 generator와 solver 신경망을 업데이트함

이러한 과정을 통해 scholar 모델은 이전 task로부터 배운 지식을 잊어버리는 일 없이
새로운 task 또한 학습할 수 있게 됨 (신경망 구조가 달라도 상관없음)

Related Works



#02 Related Works

Catastrophic Forgetting

파괴적 망각 또는 파괴적 간섭이라는 용어는 1980년대 처음 제안된 개념
원인이 알려지지 않은 신경망의 구조적 한계로 소개됨

신경망은 input의 특징을 파라미터로 기억하므로
모델을 새로 훈련시키는 것은 곧 이전에 얻은 지식을 거의 잊어버리게 하는 결과를 낳음 (파라미터가 변경됨)

이를 해결하기 위해 다양한 솔루션이 제시됨

- Elastic Weight Consolidation
- Learning without Forgetting
- Complementary Learning System
- Deep Generative Replay

#02 Related Works (1) Comparable methods

이전 task의 데이터에 따로 접근하지 않고, 현재의 학습에서 가중치의 변화를 최소화하는 방식

- 정규화 방법론 적용: 드롭아웃, L2 정규화 등
- Elastic Weight Consolidation(EWC): 이전 task에 대한 중요도를 판단하여, 특정 가중치만을 보호함

task별로 다른 파라미터를 사용하여 신경망을 확장시키는 방식

- input과 가까운 층은 파라미터를 공유하되, output 층은 task별로 결과를 출력하도록 함
  - output 층은 간섭에서 자유롭지만, 이전 층은 변경되므로 여전히 older task에 대한 성능 저하를 야기함
- Learning without Forgetting(LwF): 위 방식을 채택하고 공유된 파라미터의 변화를 최소화함
  - 미세 조정을 통해 이전 task의 지식을 간접적으로 유지시킴

#02 Related Works (2) Complementary Learning System(CLS) theory

Complementary Learning System(CLS)

과거 데이터에 아예 접근하지 않는 가짜 재현(pseudorehearsal) 기법을 사용
기억 신경망을 통해 생성한 가짜 input과 가짜 target을 task 신경망에 입력함

해마의 구조를 모방하여, 복잡한 데이터를 이용하는 연속 학습에의 활용이 가능하다고 주장되었으나
실생활에서 나타나는 것과 같은 고차원의 가짜 입력을 생성하지 못해 확장성이 보장되지 않음

Deep Generative Replay와의 차이점

반면 generative replay 프레임워크는 이와 달리 이전 input의 분포로부터 가짜 input을 생성함
또한 생성한 이전 데이터와 실제 현재 데이터를 앙상블하여 입력

-> 실제 이전 데이터를 모두 누적시켜 사용한 것과 동일한 성능을 낼 수 있음

#02 Related Works (3) Deep Generative Models

본 모델은 GANs 프레임워크를 채택하여 관측한 샘플과 유사한 가짜 샘플을 생성하도록 함

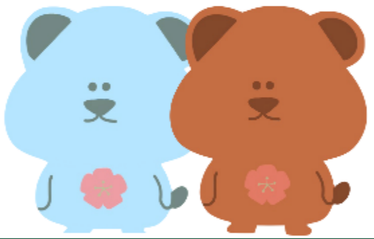
Generative Adversarial Networks(GANs) 프레임워크

generator G와 discriminator D 간의 제로섬 게임을 정의함
D는 생성된 샘플과 실제 샘플의 분포를 비교하여 둘을 구별하는 방법을 학습함
G는 최대한 실제 분포에 가깝게 모방하는 방법을 학습함

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

< objective of two networks >

Generative Replay



EWHA EURON

#03 Generative Replay (0) Terminologies

본 연속 학습 프레임워크에서
해결할 N개 task의 시퀀스를 다음과 같이 정의함

$$\mathbf{T} = (T_1, T_2, \cdots, T_N)$$

task T_i에 대하여 훈련될 때, D_i를 따르는 샘플이 주어짐

Definition 1
하나의 task T_i는 훈련 샘플 (x_i, y_i)에 대하여 모델이 데이터 분포 D_i에 가까워지도록 최적화함

위 모델을 scholar라고 하며, 새로운 task를 학습하고 그로부터 얻은 지식을 다른 신경망에 가르침
가르치는 것과 배우는 것 둘 다 가능하다는 점에서 앙상블 모델의 teacher-student 프레임워크와 구별됨

Definition 2
하나의 scholar H는 튜플 <G, S>로, generator G는 실제와 비슷한 샘플을 만드는 생성 모델이며
solver S는 θ를 파라미터로 하는 task를 해결하는 모델임

solver는 task sequence T를 모두 수행할 수 있어야 하며,
전체 목적함수는 다음을 최소화함
(task sequence 내 전체 task의 loss 합)

$$\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D}[L(S(\boldsymbol{x};\theta),\boldsymbol{y})]$$

D: 전체 데이터 분포, L:손실함수

#03 Generative Replay (1) Proposed Method

가장 최신
scholar 모델
단일 훈련

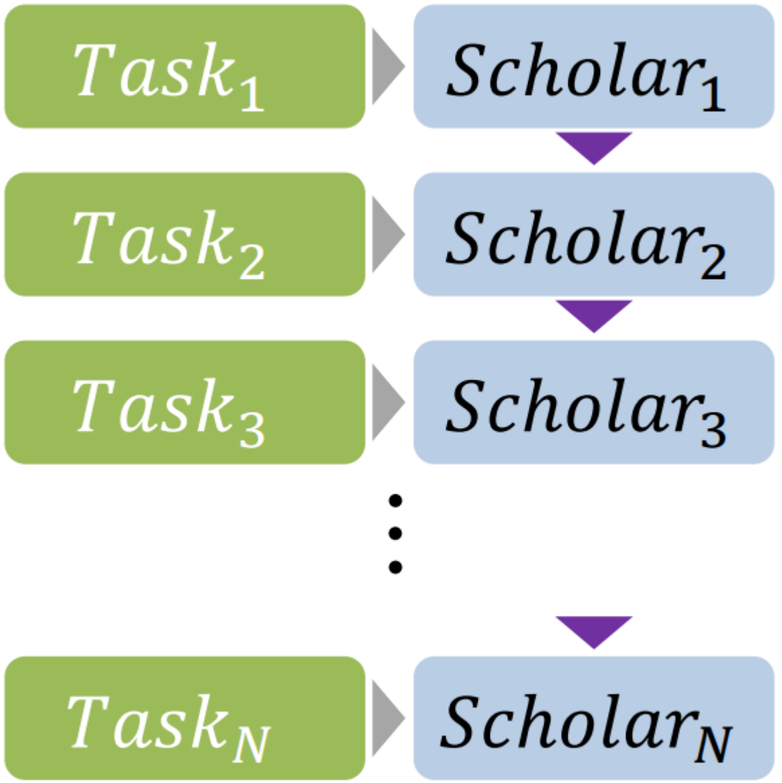
$$H_n$$

(n > 1)

$$=$$

scholar 모델의
전체 시퀀스 훈련

$$(H_i)_{i=1}^{N}$$



N번째 scholar H_n0이
현재 task T_n 및
이전 scholar H_{n-1}로부터 얻은 지식까지 학습하기 때문

#03 Generative Replay (1) Proposed Method

scholar 모델로부터 다른 scholar를 훈련시키는 것은 다음 두 과정으로 진행됨

1. 새로운 generator가 현재 task의 input x를 받고, 이전 task들의 input x' 를 replay함
   이때 두 데이터의 샘플이 섞이며, 그 비율은 task의 중요도에 따라 정해짐
2. generator는 누적된 input 공간을 재구성할 수 있도록 학습되며,
   새로운 solver는 위 데이터에서 input과 target을 매칭하도록 학습됨
   (replayed target y' == past solver가 replayed input x' 를 받아 출력한 결과)

i번째 solver에 대한 훈련 손실함수

θ_i: i번째 scholar 신경망의 파라미터, r: ratio

$$L_{train}(\theta_i) = r\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D_i}[L(S(\boldsymbol{x};\theta_i),\boldsymbol{y})]+(1-r)\mathbb{E}_{\boldsymbol{x}'\sim G_{i-1}}[L(S(\boldsymbol{x}';\theta_i),S(\boldsymbol{x}';\theta_{i-1}))]$$

실제 데이터 분포

생성된 이전 데이터 분포

* $\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D}[L(S(\boldsymbol{x};\theta),\boldsymbol{y})]$

EWHA EURON

#03 Generative Replay (1) Proposed Method

i번째 solver에 대한 훈련 손실함수

θ_i: i번째 scholar 신경망의 파라미터, r: ratio

$$L_{train}(\theta_i) = r\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D_i}[L(S(\boldsymbol{x};\theta_i),\boldsymbol{y})]+(1-r)\mathbb{E}_{\boldsymbol{x}'\sim G_{i-1}}[L(S(\boldsymbol{x}';\theta_i),S(\boldsymbol{x}';\theta_{i-1}))]$$

실제 데이터 분포

생성된 이전 데이터 분포

i번째 solver에 대한 테스트 손실함수

$$L_{test}(\theta_i) = r\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D_i}[L(S(\boldsymbol{x};\theta_i),\boldsymbol{y})] + (1-r)\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D_{past}}[L(S(\boldsymbol{x};\theta_i),\boldsymbol{y})]$$

이전 데이터가 누적된 분포

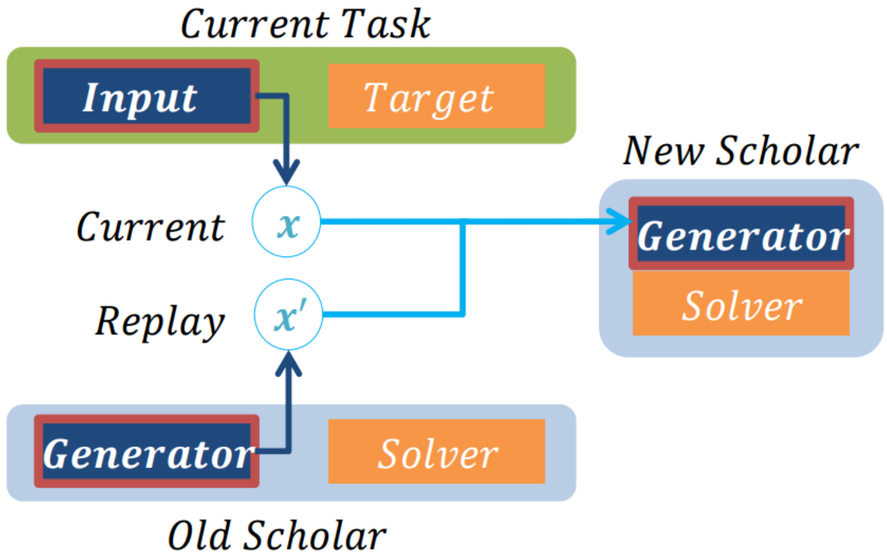
테스트는 모델을 원래 task에 대하여 평가하는 것을 목표로 함

첫 번째 solver에 대해서는 replay될 데이터가 없으므로,
2번째 항은 i = 1일 때 무시됨 (훈련, 테스트 손실함수에서 모두 동일함)

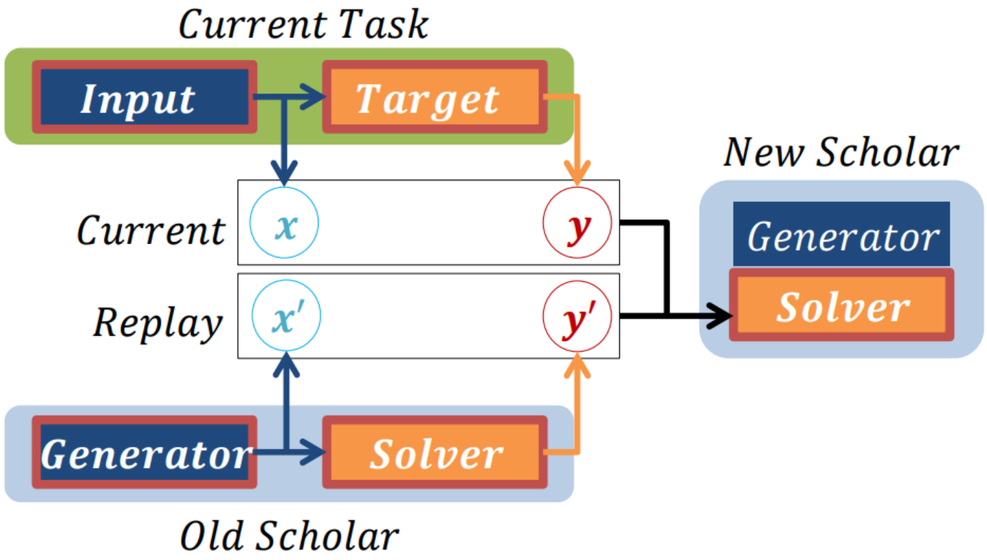
#03 Generative Replay (1) Proposed Method

[새로운 generator]
실제 샘플 x와
이전 generator로부터 replayed 샘플 x'가
혼합된 데이터의 분포를 복제하도록 훈련됨

[새로운 solver]
실제 input-target 쌍 (x, y)와
replayed input-target 쌍 (x', y')로부터 학습함
(replayed target y' == 이전 solver에
생성된 input x'를 넣어 얻은 값)



(b) Training Generator



(c) Training Solver

#03 Generative Replay (2) Preliminary Experiment

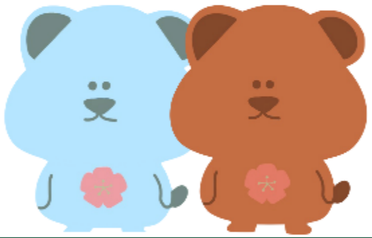
주 실험에 앞서, 훈련된 scholar 모델 하나만으로 새로운 신경망을 훈련시킬 수 있음을 보임

MNIST 손글씨 분류 예제에서, 각 solver의 테스트 정확도가 유지됨을 관찰함
-> scholar 모델이 정보 손실 없이 지식을 잘 전달함!

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

Solver_1은 실제 데이터로 학습하였으며, 이후 solver는 모두 이전 scholar 신경망으로부터 학습함

Experiments



EWHA EURON

#04 Experiments

다양한 순차적 학습 환경을 설정하여 generative replay 프레임워크가 잘 작동함을 보임

본 프레임워크는 생성 모델의 성능이 task 성능에 있어 유일한 제약임
다시 말해, 생성 모델이 최적일 경우 누적된 데이터 전체로 신경망을 학습시켰을 때와 같은 성능이 됨

가능한 좋은 성능을 내기 위해, generator의 학습에서 WGAN-GP를 이용함

GAN의 한 변형으로,
Wasserstein Distance와 Gradient Penalty를 이용함

[이후 표기]
GR: generative replay 이용 시
ER: exact replay 시 (누적된 데이터 전체 이용하여 학습, generator가 완벽할 경우를 상정)
Noise: 생성된 샘플이 실제 분포를 전혀 따르지 않는 worst case
None: 데이터 생성 없이 solver만을 학습시킨 신경망

#04 Experiments (1) Learning independent tasks

연속 학습 연구에서 주로 사용되는 실험 환경으로, MNIST 데이터셋에서
각 데이터의 픽셀 값을 임의로 재배치하여 task별로 input이 다르도록 함 (각 task는 대부분의 경우 모두 독립적임)

solver는 이러한 input을 원래 class로 분류해야 함

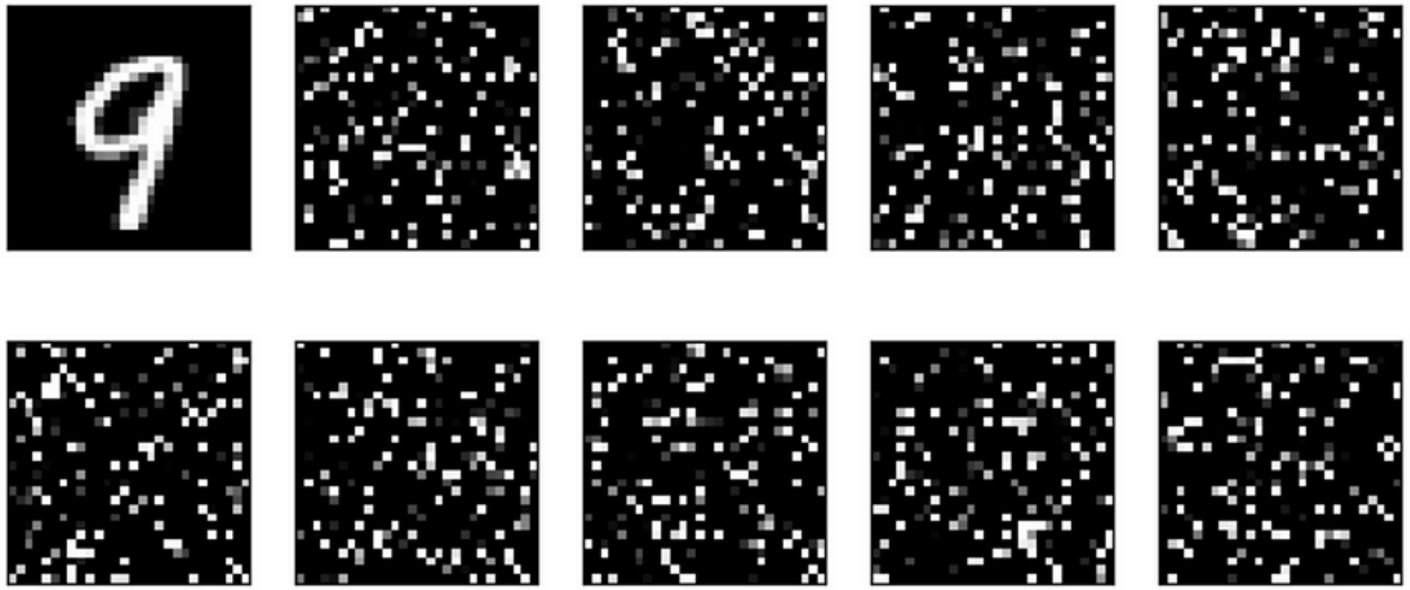
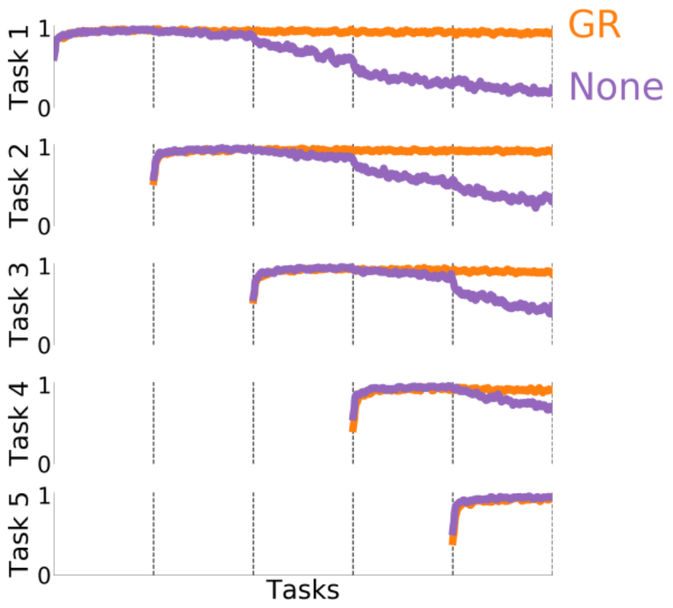
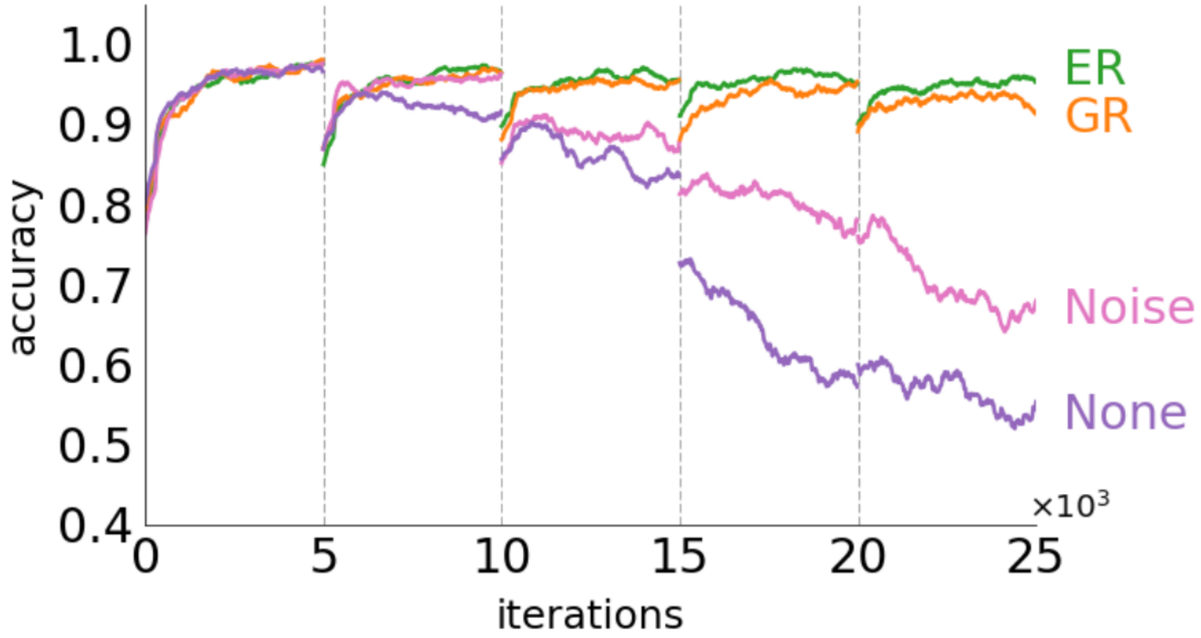


Figure 1: 10 different permutations of the same image. These permutations are used to create 10 different tasks.

#04 Experiments (1) Learning independent tasks



(a)



(b)

(a) 이전 task에 대한 성능은
실제 데이터 또는 의미 있는 가짜 데이터를
replay하지 않으면 저하됨

(b) 이전 task에 대한 평균 테스트 정확도
- GR: 성능이 유지되어 ER과 가까움
- Noise: 랜덤 가우시안 노이즈는 유의미한 영향을 주지 못함
- None: 파괴적 망각 현상이 일어남

#04 Experiments (2) Learning new domains

일반적으로, 같은 신경망으로 독립적인 task를 훈련시키는 것은 공유될 정보가 없어 비효율적임
그러나 generative replay는 다중 task 해결에 있어 이점을 가짐

다중 task에 하나의 모델을 사용하는 것은 다음과 같은 장점이 있음

1. 한 도메인에서의 지식이 다른 도메인에서의 더 나은, 더 빠른 이해를 도움
   (두 도메인이 완전히 독립적이지 않은 경우)

2. 여러 도메인의 일반화를 통해 보편적인 지식을 얻는다면, 이를 완전히 새로운 도메인에도 적용할 수 있음
   - 이는 어린아이가 각 카테고리의 특성을 유추하고 새로운 물체가 속하게 될 카테고리를 예측하는 것과 비슷함

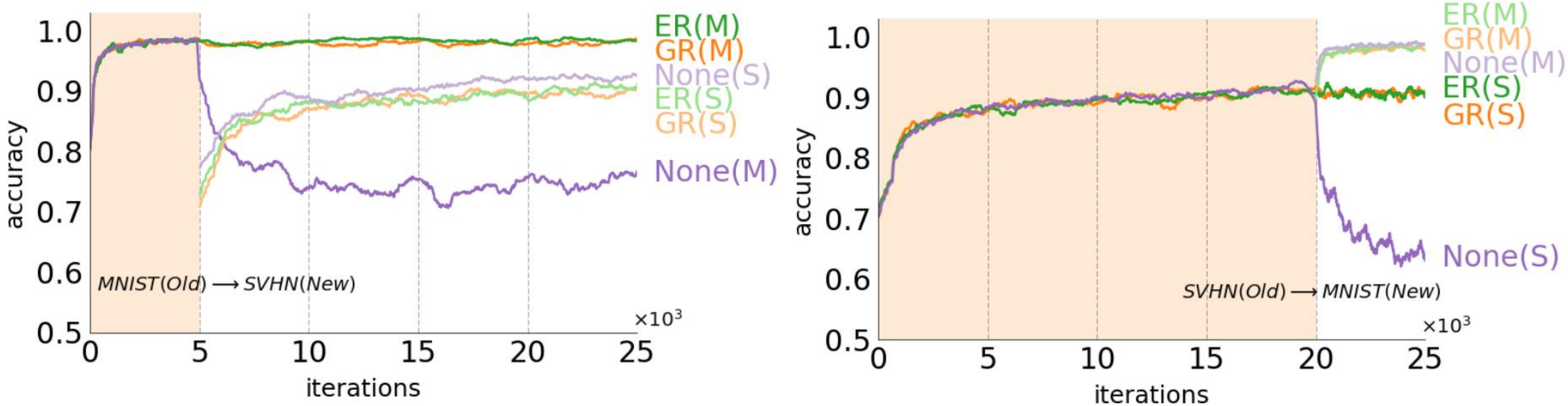
#04 Experiments (2) Learning new domains

generative replay를 통해 새로운 도메인에서의 지식을 통합할 수 있는지 테스트함

MNIST와 Street View House Number(SVHN) 데이터셋을 사용하여,
MNIST->SVHN, SVHN->MNIST 2가지 경우로 순차적 학습을 진행함

순차적 학습 진행 후, generator가 재생해낸 이전 데이터 (iteration 횟수별)

#04 Experiments (2) Learning new domains



(a) MNIST to SVHN

(b) SVHN to MNIST

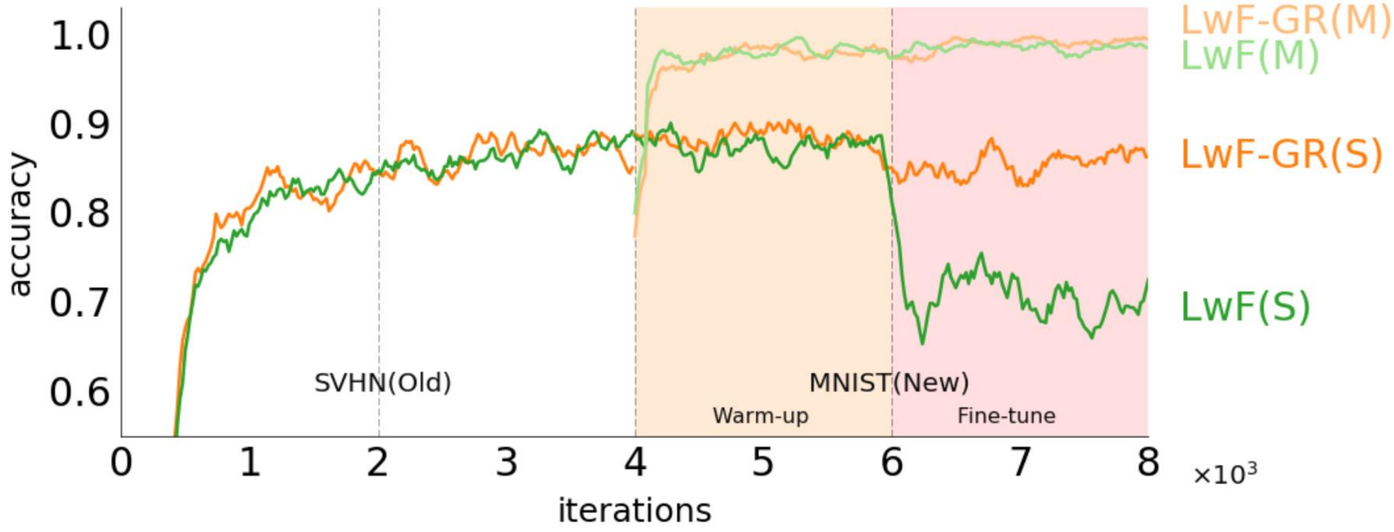
(a), (b) 2개의 다른 도메인에 대한 분류 정확도
- **뚜렷한 선**: 원래 task의 성능
- 흐린 선: 새로운 task의 성능

- GR: 첫 task에서의 성능을 유지했으며, 두 번째 task의 성능 또한 ER과 맞먹음
- None: 이전 도메인에 대한 망각이 일어나 성능이 크게 하락했으나,
  신경망이 두 번째 task에만 최적화되어 새 task에서는 다른 모델보다 조금 더 나은 성능을 보임

#04 Experiments (2) Learning new domains

generative replay는 다른 연속 학습 모델과 호환 가능함
(e.g., LwF는 지난 지식을 불러오기 위해 현재 task의 input을 replay하는데,
이때 지난 task의 input과 유사한 샘플을 만드는 생성형 모델을 이용할 수 있음)

LwF와, LwF에 GR을 적용시킨 LwF-GR의 성능을 비교함
(이전 신경망을 유지하기 위해 task별로 input을 다르게 생성하여 훈련시킴)
LwF가 소개된 논문에서와 같이, 신경망의 초기 부분을 어느 정도 훈련시킨 뒤 전체 신경망을 미세 조정함



GR을 적용하자 신경망이
지난 지식의 대부분을 유지함

미세 조정이 시작되자
첫 task에서의 성능을 잃음
(공유 중인 신경망이 변경돼서)

#04 Experiments (3) Learning new classes

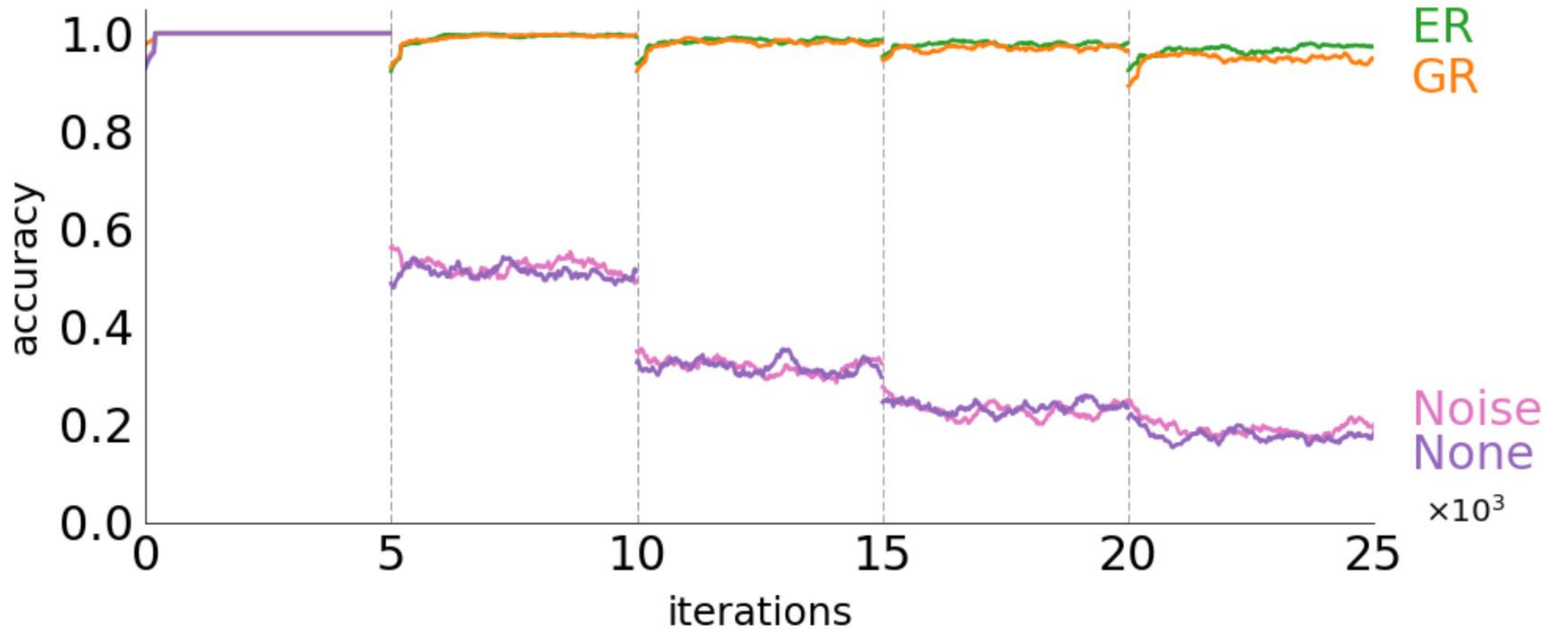
generative replay가 task별로 input과 target이 크게 차이 나는 상황에서도 지난 지식을 잘 기억해낼 수 있음을 보이기 위해, 공통부분이 없는 데이터로 순차적인 훈련을 진행하는 새로운 실험을 제안함

포함된 class가 계속해서 바뀌는 경우, 신경망이 새 target 분포에 맞게 계속 변경되므로
generative replay를 통해 이전 input과 target을 재생해야 신경망을 균형 있게 학습시킬 수 있음

MNIST 데이터셋을 5개의 공통부분 없는 부분집합(2개 class 포함)으로 나눠 각 task의 input으로 사용함
(e.g., 첫 번째 task에서는 0과 1, 두 번째 task에서는 2와 3, …)

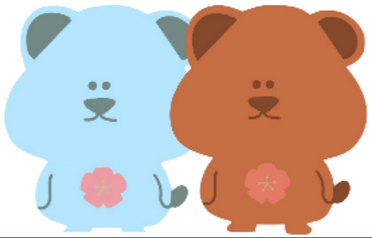
generator가 모든 학습했던 샘플들을 누적시켜 input 분포를 생성해낼 수 있었음

#04 Experiments (3) Learning new classes



- GR: input과 output 분포를 모두 재구성했을 때, 이전에 학습한 class 정보를 상기하여 ER과 유사하게 전체 class를 구별할 수 있게 됨
- None: 이전 class 정보를 완전히 잊어 매번 새로운 데이터에 대해서만 학습함
- Noise: 유의미한 input 분포가 없어 이전 output 분포만을 기억했으며, 이는 지식을 유지하지 못함

Discussion



EWHA EURON

#05 Discussion

Deep Generative Replay 프레임워크의 제안

이전에 학습한 샘플을 모방한 가짜 데이터를 생성 및 재현하여 다중 task에 대한 순차적 학습을 가능케 함
generator와 solver로 구성된 scholar 모델은 task를 기반으로 한 지식의 역할을 함

구조를 조금 변형시켜, 유사한 다른 문제를 해결하도록 할 수도 있음
(e.g., 이전 scholar 모델을 같은 신경망의 복제로 만들어, 훈련 과정을 명시적으로 나눌 필요 없이 다중 task를 학습하도록 할 수 있음)

다른 방법론의 한계

LwF 또는 EWC와 같은 접근은 이전 신경망의 지식을 보호하여 파괴적 망각을 완화할 수 있음을 보임
그러나, 이는 신경망의 가중치 보호를 위한 추가적인 손실 항을 필요로 하여 task 간 성능의 tradeoff 발생 가능
이를 막기 위해 신경망이 훨씬 거대해야 함 (비용이 많이 듦)

특히,
EWC: 모든 task의 학습에서 같은 구조를 유지해야 함
LwF: task 간 연관성에 성능이 크게 좌우되며, 한 task에 대한 훈련 시간이 이전 task의 수에 따라 선형적으로 증가함

#05 Discussion

Deep Generative Replay 메커니즘의 이점과 한계

저장된 신경망으로부터 생성된 input-target 짝으로부터 이전 지식을 유지하여,
task 간 성능의 균형을 맞추고 지식의 전이를 용이하게 함

신경망이 모든 task의 목표에 맞게 공동으로 최적화되고,
이에 따라 generator가 input 공간을 완벽하게 재구성했을 때 최고 성능을 보장함

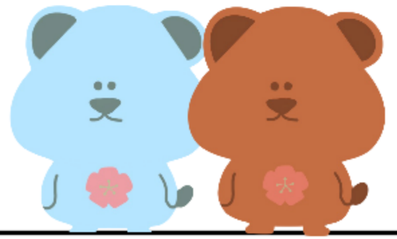
그러나 알고리즘의 성능이 generator의 생성 퀄리티에 크게 구애받는다는 결점이 있음

Generative Replay의 향후 전망 및 혼합 사용 가능성

생성형 모델의 발전이 이루어지면 본 프레임워크의 성능 또한 직접적으로 향상될 것임
강화학습 및 지속적으로 진화하는 신경망의 개발에 활용될 수 있음

EWC와 LwF, generative replay는 모두 다른 레벨에서의 기억 유지에 기여하므로,
이들을 적절히 섞어 연속 학습의 고질적인 문제를 해결할 수 있을 것임

THANK YOU



EWHA EURON