Lifelong Learning with DEN

(Dynamically Expandable Networks)

KAIST 전산학부 기계학습 및 지능연구실(MLILAB)
석사과정 이준영
2018-08-08

이 슬라이드는 ICLR 2018에 발표된
Yoon, Jaehong 등이 저술한
"Lifelong Learning with Dynamically Expandable Networks.” 논문을
스터디원들에게 공유하기 위해 만든 슬라이드입니다.

Introduction

Contents

- Prerequisites
- Related Works
- Dynamically Expandable Networks
- Experiments
- Conclusion and Discussion

Prerequisites

l1-Regularizer

Results in parameter selection

$$\min_{w} J(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_1$$

l1-regularizer를 사용하게 되면,
무슨 효과가 있을까?

$$\|\boldsymbol{w}\|_1 = \sum_{w_j} |w_j|$$

$$\|\boldsymbol{w}\|_1 \leq \lambda$$

As the dimensionality of w increases, the norm ball will have increasingly more number of corners.

* 2018 봄 KAIST 황성주 교수님 전산학부 딥러닝 개론 수업 슬라이드에 발췌.

Prerequisites

l1-Regularizer

Results in parameter selection

$$\min_{w} J(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_1$$

l1-regularizer를 사용하게 되면,
무슨 효과가 있을까?

-> Dimension Reduction 효과!
다시 말하면, 변수를 Sparse하게 해주는 효과가 있음. Why?

$$\|\boldsymbol{w}\|_1 = \sum_{w_j} |w_j|$$

$$\|\boldsymbol{w}\|_1 \leq \lambda$$

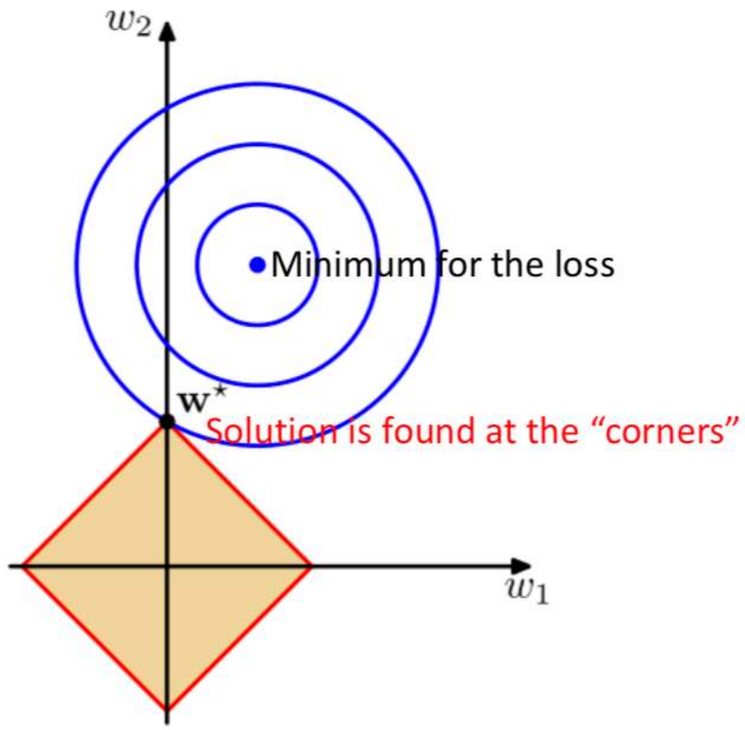
As the dimensionality of w increases, the norm ball will have increasingly more number of corners.

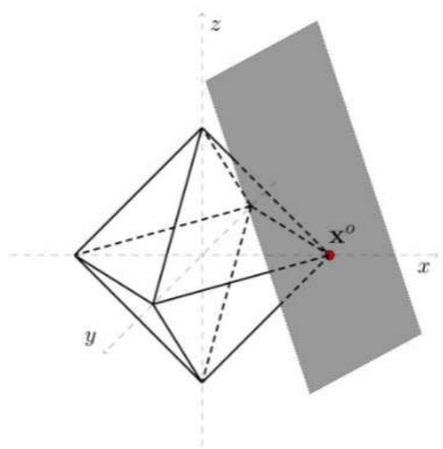
* 2018 봄 KAIST 황성주 교수님 전산학부 딥러닝 개론 수업 슬라이드에 발췌.

Prerequisites

l1-Regularizer

**l1-norm이 특정 값(상수)이 되도록 hard-constraint를 주면,
높은 확률로 솔루션이 절편에서 나온다.
즉, 솔루션이 sparse하도록 유도하는 것이다.
솔루션이 sparse하면,
dimension이 reduction되는 효과도 얻을 수 있다.**





* 2018 봄 KAIST 황성주 교수님 전산학부 딥러닝 개론 수업 슬라이드에 발췌.

Prerequisites

l1-Regularizer

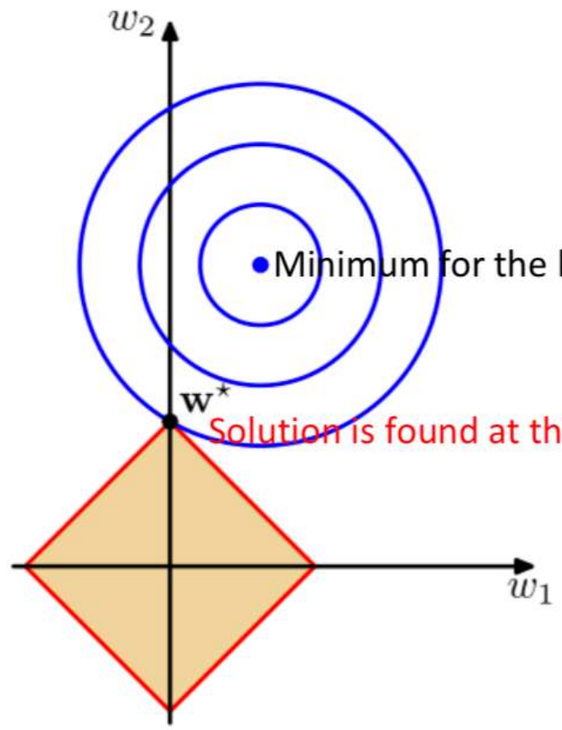
**다만, hard-constraint가 아니고,
실제로 regularizer로 사용될 때에는
0에 가까운 값이 많아지는 효과가 있다.
그래서, 실제로 sparse하게 만들고 싶으면,
일정 값 이하에 대한 weight를 0으로
만들어버리면 된다.**

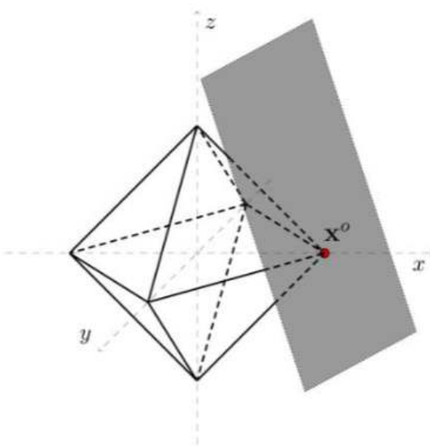
Results in parameter selection

$$\min_{w} J(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_1$$

$$\|\boldsymbol{w}\|_1 = \sum_{w_j} |w_j|$$

$$\|\boldsymbol{w}\|_1 \leq \lambda$$





As the dimensionality of w increases, the norm ball will have increasingly more number of corners.

* 2018 봄 KAIST 황성주 교수님 전산학부 딥러닝 개론 수업 슬라이드에 발췌.

Prerequisites

Multi-task Learning

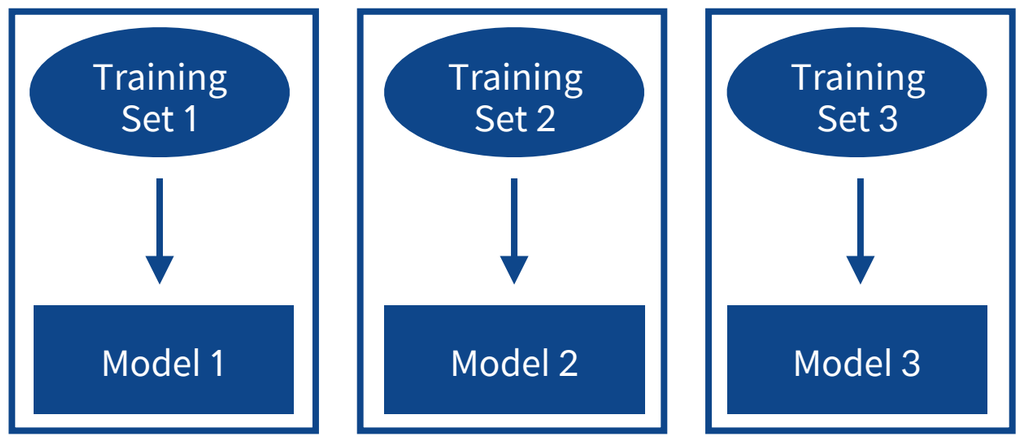
- Task 란?
  - 머신러닝 알고리즘이 수행해야 하는 작업.
  - Ex. 개와 고양이 구별하기, 손글씨로 쓴 숫자를 보고 어떤 숫자인지 맞추기.

Prerequisites

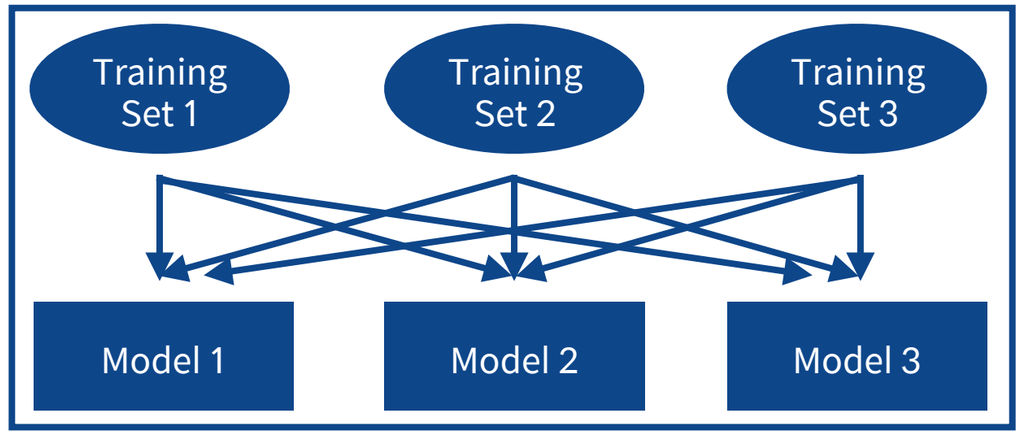
Multi-task Learning

- Task 란?
  - 머신러닝 알고리즘이 수행해야 하는 작업.
  - Ex. 개와 고양이 구별하기, 손글씨로 쓴 숫자를 보고 어떤 숫자인지 맞추기.

- Multi + Task
  - 여러 개의 Task를 동시에 학습하는 방법론.



<Single-task Learning>



<Multi-task Learning>

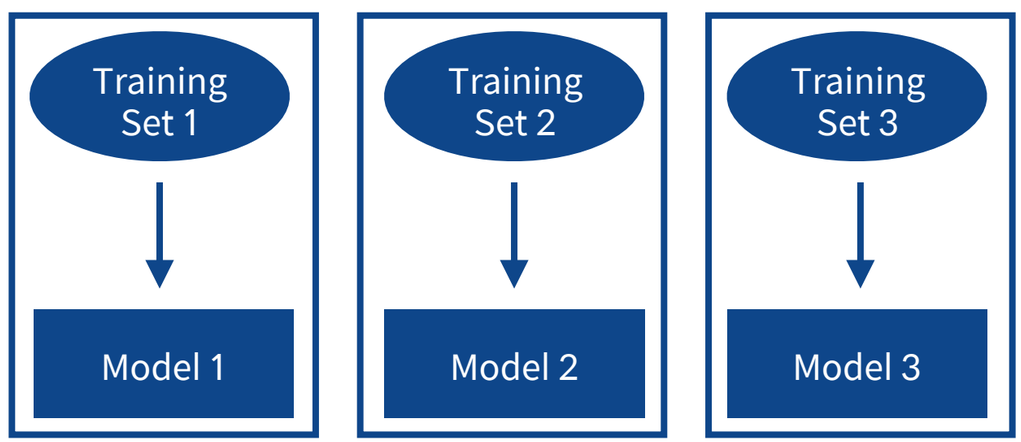
Prerequisites

Multi-task Learning

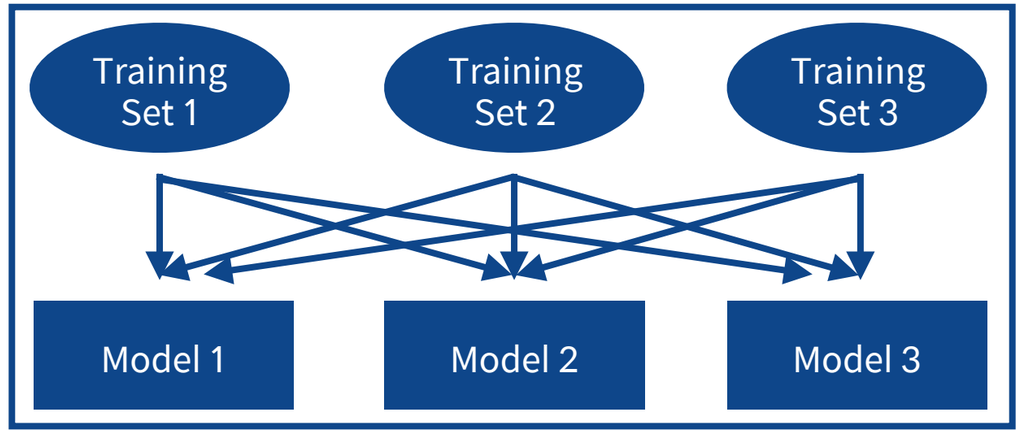
- Task 란?
  - 머신러닝 알고리즘이 수행해야 하는 작업.
  - Ex. 개와 고양이 구별하기, 손글씨로 쓴 숫자를 보고 어떤 숫자인지 맞추기.

- Multi + Task
  - 여러 개의 Task를 동시에 학습하는 방법론.

- 왜 동시에 학습해야 하는가?
  - 각각을 따로 학습할 때 보다 더 성능이 향상될 수 있기 때문!
  - 개와 고양이를 구별하는 Task와 늑대와 호랑이를 구별하는 Task가 있다고 할 때, 둘을 각각 학습시키는 것보다 둘을 한번에 학습시키면 더 성능이 좋아질 것을 기대할 수 있지 않을까?



<Single-task Learning>



<Multi-task Learning>

Prerequisites

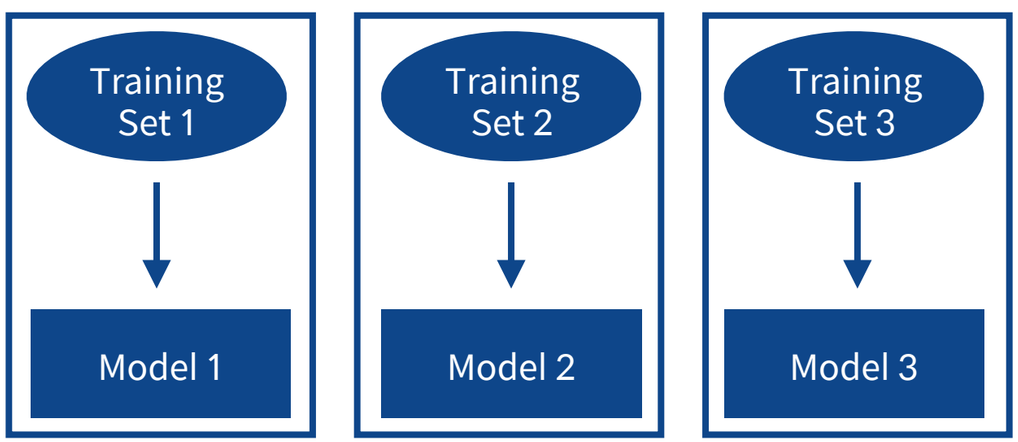
Multi-task Learning

- Task 란?
  - 머신러닝 알고리즘이 수행해야 하는 작업.
  - Ex. 개와 고양이 구별하기, 손글씨로 쓴 숫자를 보고 어떤 숫자인지 맞추기.

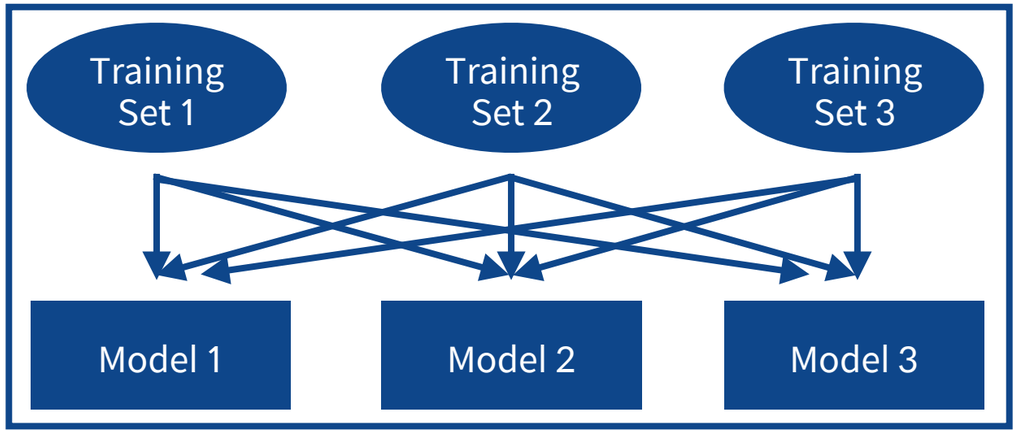
- Multi + Task
  - 여러 개의 Task를 동시에 학습하는 방법론.

- 왜 동시에 학습해야 하는가?
  - 각각을 따로 학습할 때 보다 더 성능이 향상될 수 있기 때문!
  - 개와 고양이를 구별하는 Task와 개와 호랑이를 구별하는 Task가 있다고 할 때, 둘을 각각 학습시키는 것보다 둘을 한 번에 학습시키면 더 성능이 좋아질 것을 기대할 수 있지 않을까?

- Knowledge Transfer
  - Single-task Learning보다 Multi-task Learning의 성능이 좋아졌을 때, 한 Task의 지식이 다른 Task로 전이되어서 성능이 좋아졌다고 말함.
  - Multitask Learning의 목표는 지식 전이가 일어나도록 하는 것!



<Single-task Learning>

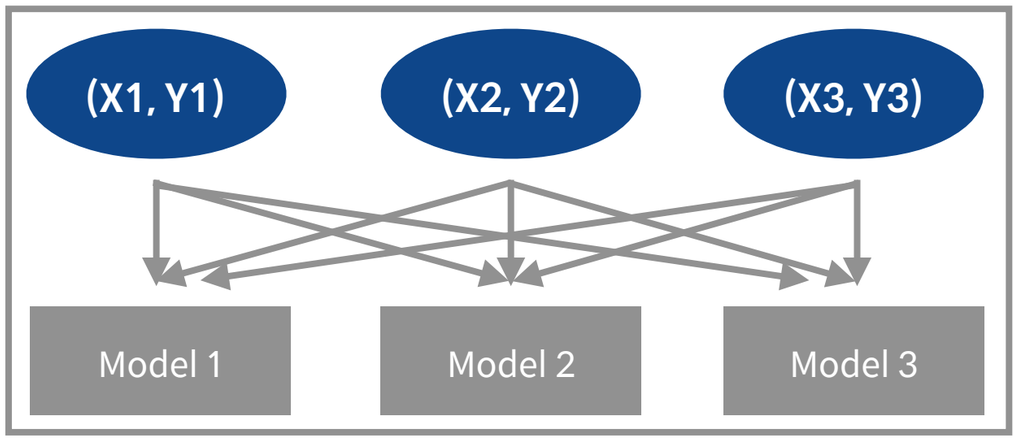


<Multi-task Learning>

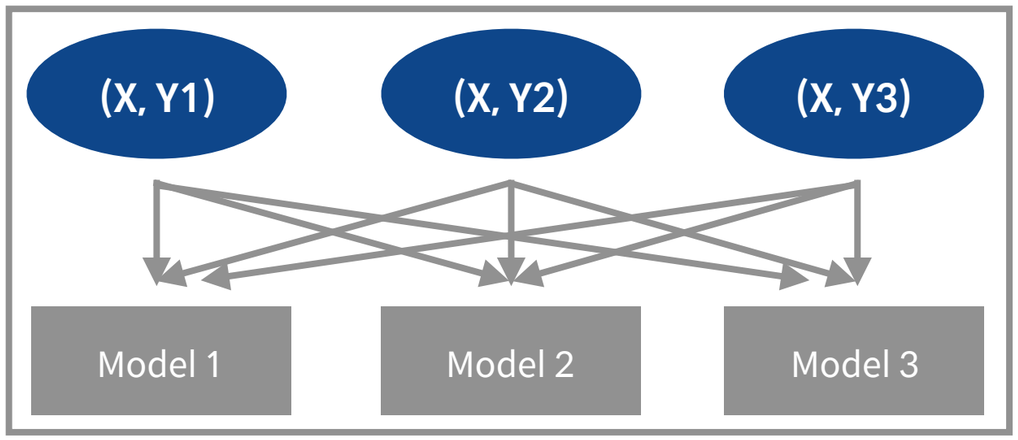
Prerequisites

Multi-task Learning

- Set 에 대한 개념을 혼동하지 않기.
  - 오른쪽 다이어그램 같은 상황도 Multi-tank Learning 임!



- 개와 고양이 사진을 구별하는 Task와 늑대와 호랑이 사진을 구별하는 Task를 같이 수행하는 Multi-task Learning



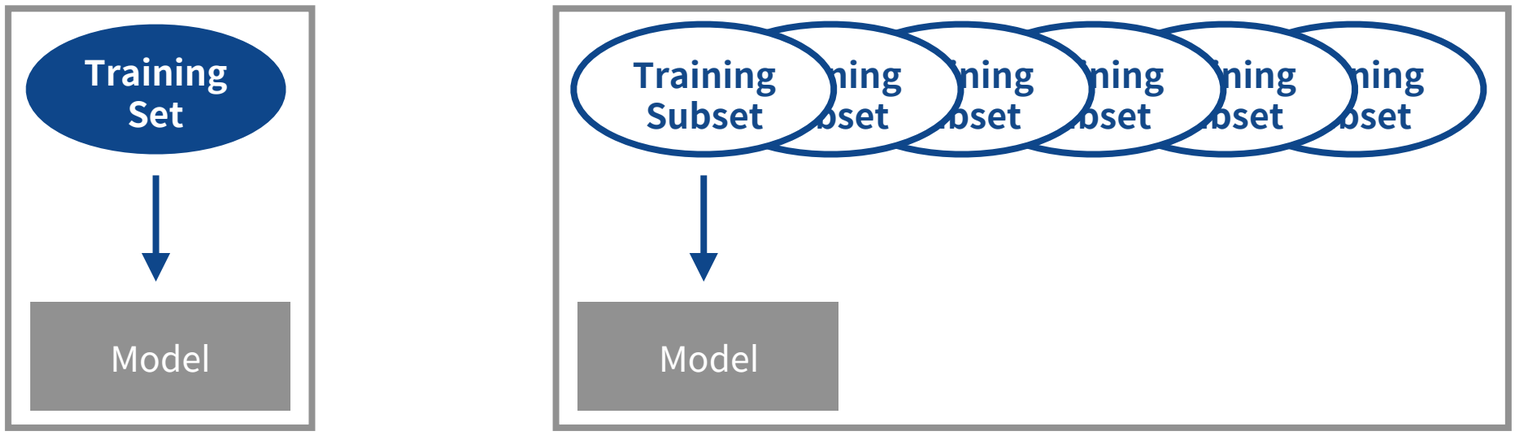
- 한글 손글씨 사진을 보고 초성 중성 종성을 각각 맞추는 Multi-task Learning
- 자율 주행차는 카메라의 입력을 받아, line을 tracing 함과 동시에 장애물의 위치를 파악해야 함.

Prerequisites

Online Learning

- 오프라인 러닝
  - 우리가 아는 일반적인 머신러닝.
  - 모델아! 트레이닝 데이터 이만큼 한번에 다 줄 테니까, 러닝 해봐!

- 온라인 러닝
  - 데이터가 스트림으로 주어지는 상황에서의 머신 러닝.
  - 모델아! 트레이닝 데이터를 한번에 주면 용량이 너무 크니까, 조금씩 나눠서 줄게! 러닝 해봐!
  - 근데 내가 주는 거 저장하기엔 너무 클테니까 트레이닝 데이터는 쓰고 버려!



- 온라인 러닝에서는 **한번 쓴 데이터를 다시 쓸 수 없는 경우가 반드시 존재한다.**
  - 모든 데이터를 1 epoch 에만 쓸 수 있는 것은 아님!
  - 받은 batch data를 이용해 여러 epoch 러닝하고 버려도 됨.

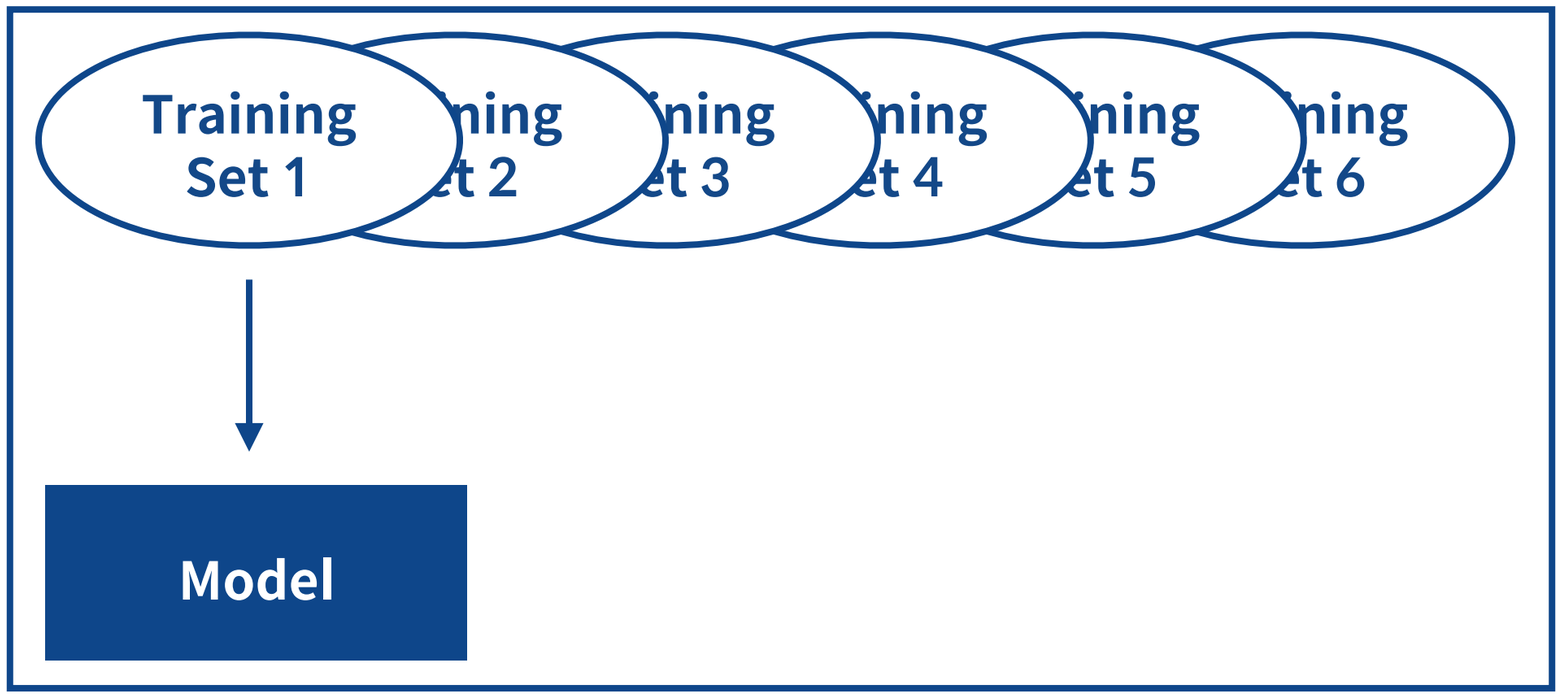
Online Multi-task Learning

= Online Learning + Multi-task Learning

Prerequisite

Lifelong Learning

- Online Multitask Learning의 일종.

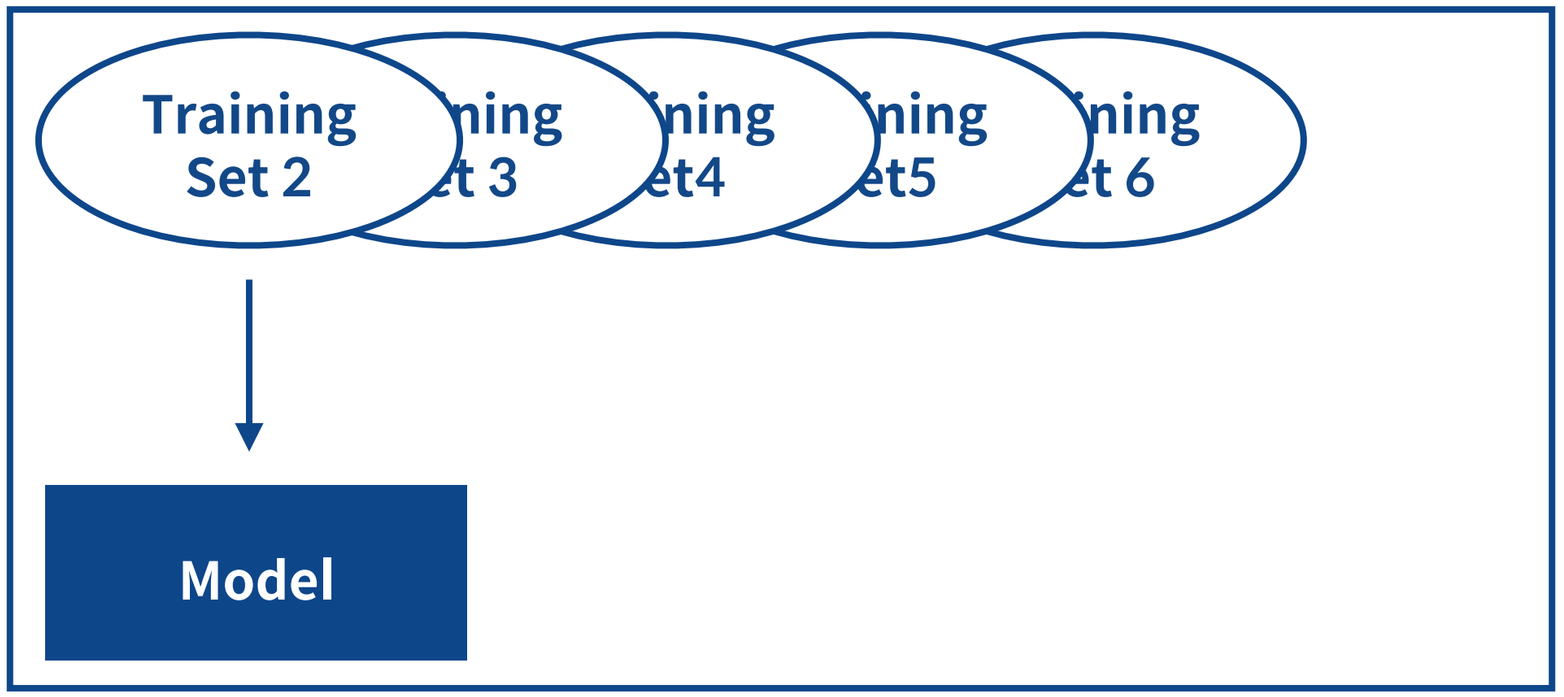


- Task t 에 해당하는 Training Set t가 차례로 주어진다.
- Training Set t는 Task t’(t’ > t)를 학습할 때에는 주어지지 않는다.

Prerequisite

Lifelong Learning

- Online Multitask Learning의 일종.

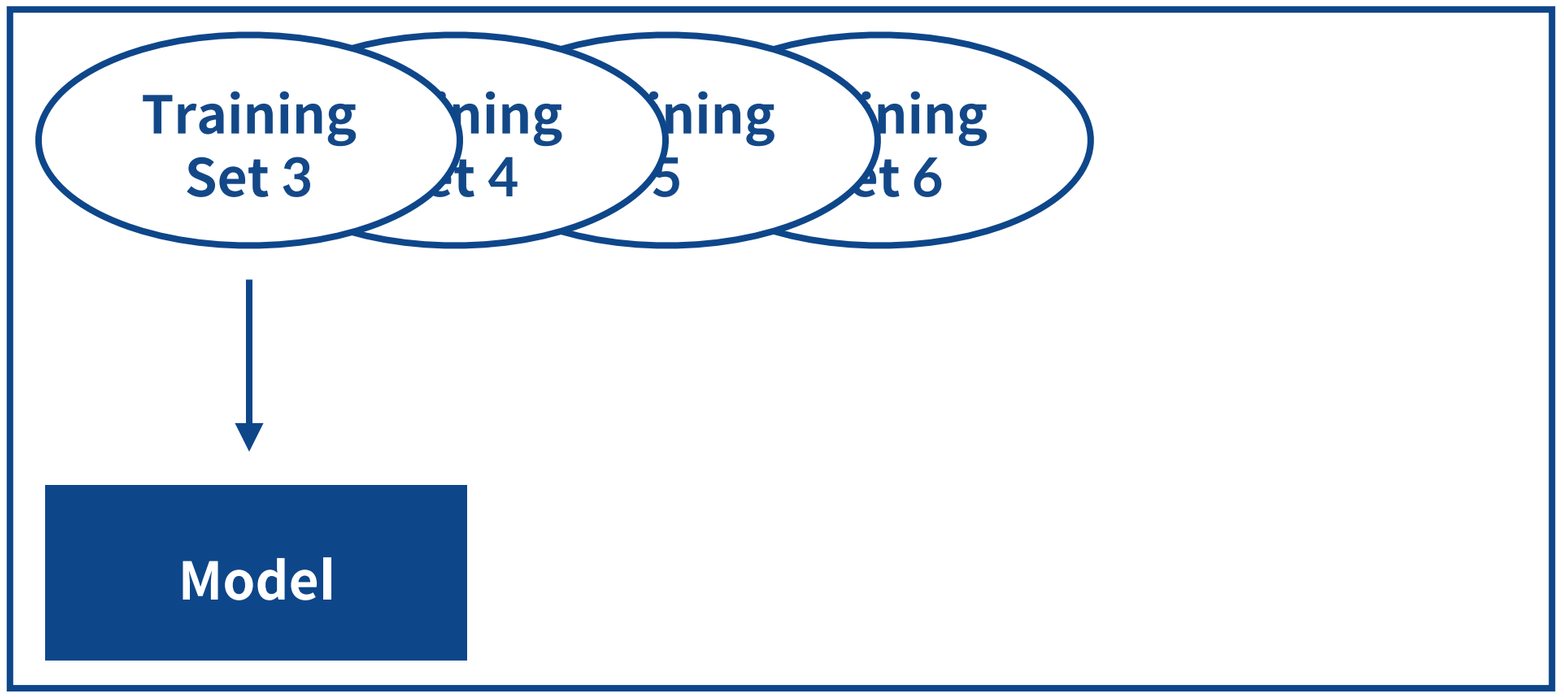


- Task 1에 해당하는 Training Set 1을 모델이 학습하고 나면,
  더 이상 Training Set 1은 접근할 수 없다.

Prerequisite

Lifelong Learning

- Online Multitask Learning의 일종.



- Task 2에 해당하는 Training Set 2를 모델이 학습하고 나면,
  더 이상 Training Set 2은 접근할 수 없다.
- 이후도 마찬가지!

Prerequisite

Lifelong Learning

- 평생 학습 문제 상황은 자율 주행이나 로봇 에이전트 학습과 같이 데이터가 스트림 형태로 도착하는 시나리오에서 자주 등장해 왔음.

- Lifelong Learning 상황에서는 Network Capacity도 효율적으로 써야 한다.
- 에이전트의 총 메모리는 정해져서 나오는데, 에이전트가 평생 학습을 진행하려면 한 Task 당 최소한으로 Network Capacity를 써가면서 학습해야 함.
  - 논문에서 memory efficient 해야 한다는 표현이 나오는데, 여기서 메모리는 스토리지를 포함하는 개념!

- Multi-task Learning이기 때문에 지식 전이도 일어나야 함.
- Forward Knowledge Transfer vs Backward Knowledge Transfer.
  - Forward Knowledge Transfer: 이전 태스크를 미리 학습함으로써, 이후 태스크의 학습을 돕는 상황.
  - Backward Knowledge Transfer: 이후 태스크를 미리 학습함으로써, 이전 태스크의 성능이 증가하는 상황.

Prerequisite

Catastrophic Forgetting

- Lifelong Learning에서는 Task 1, Task 2, ⋯, Task N을 차례로 학습함.

- Task t을 학습함으로써, 계산된 Network weight를 W^t이라고 하자.

- 만약, Task t+1를 학습할 때는 W^t를 초기값으로 사용해 학습하면,
  t가 점점 증가할 수록 이전 Task에 대해 학습한 내용을 잊어버리게 된다.
- 이 현상을 **Catastrophic Forgetting**이라고 함!
- 종종 **의미 변화(Semantic Drift) 현상**이라고도 함!
  - Weight의 의미가 변화했다고 해서...

- Catastrophic Forgetting을 막는 것은 Lifelong Learning의 주요 challenge 중 하나.

Prerequisite

Catastrophic Forgetting 를 어떻게 막을까?

- Naive Approach 1: 모든 t에 대해 W^t를 저장.
  - Catastrophic Forgetting이 일어나지는 않겠지만, 메모리를 비효율적으로 사용할 수 밖에 없음.

- Naive Approach 2: l2 regularization 사용.
  - 새 모델이 이전 모델에서 너무 많이 벗어나는 것을 방지하는 정규화를 적용.
  - 하지만, l2 regularizer와 같은 간단한 정규화는 모델이 새 Task에 대한 지식을 습득하는 것을 방해.

Related Works

Lifelong Learning에 관련된 기존 연구

- [1], [13]은 이전 Task와 새로운 Task 모두에게 좋은 새로운 정규화 방법을 제안.
- 하지만, 신경망의 최종 계수 값보다는 전체 학습 궤도를 고려한다는 한계.
  - 일종의 Relaxation된 문제를 푸는 것!

- [9]는 이전 신경망의 수정을 완전히 차단. 기 학습된 신경망에 대해서는 역전파(Back Propagation)를 막아 신경망이 수정되지 않게 하고, 각 업무에 대해 고정 용량만큼 신경망를 확장해 나가는 방식.
  - Backward Knowledge Transfer가 일어나지 않음.
  - 그래도 Naive Approach 1보다는 메모리를 효율적으로 사용하기도 함.

Related Works

Dynamically Expandable Networks와 관련된 기존 연구

- 지금까지 학습 중 동적으로 뉴런을 추가함으로써, 신경망 네트워크 용량을 확장하는 신경망을 제시한 연구는 거의 없었음.

- [14]는 학습이 어려운 데이터에 대해 새로운 뉴런을 추가하고 중복을 방지하기 위해 다른 뉴런과 병합하는 방법론을 제시.
- [8]은 Loss를 최소화하는 방향으로 신경망을 학습하고, 각 층이 무한한 수의 뉴런을 가지고 있다는 가정 하에 손실을 줄일 수 있는 각 층의 최소 차원을 찾는 비모수적 신경망 모델을 제안.
- [5]는 부스팅 이론을 기반으로 주어진 손실을 최소화하기 위해 신경망 구조와 계수를 적응적으로 학습할 수 있는 신경망를 제안.
- 소개한 연구들은 모두 다중 업무 학습에 대해 고려하지 않고 있음.
  뉴런을 반복적으로 추가하는 과정을 포함.

- [12]는 새로운 분류가 모델에 주어질 때, 계층 구조를 형성하는 신경망을 점진적으로 학습하는 방법을 제안.
- 그러나, 모든 층에서 뉴런의 증가시킬 수 없고 모델의 분기가 많아진다는 단점이 존재.

Dynamically Expandable Networks

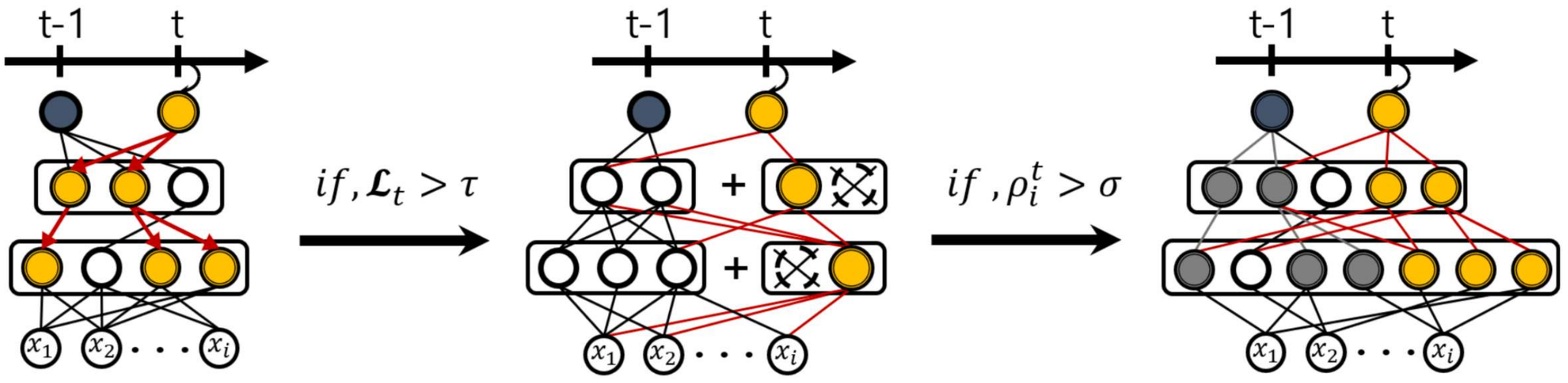
논문의 Goal

- Catastrophic Forgetting이 없으면서,
- 모델이 새로운 Task에 대한 지식을 습득하는 데 방해를 받지 않지 않으면서,
- Forward Knowledge Transfer가 일어나면서,
- Backward Knowledge Transfer도 일어나면서,
- 메모리를 효율적으로 사용하는 평생 학습 방법.

전체 알고리즘

Algorithm 1 Incremental Learning of a Dynamically Expandable Network

Input: Dataset $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_T)$, Thresholds τ, σ
Output: $\boldsymbol{W}^T$
for $t = 1, \ldots, T$ **do**
  if $t = 1$ **then**
    Train the network weights $\boldsymbol{W}^1$ using Eq. 2
  else
    $\boldsymbol{W}^t = SelectiveRetraining(\boldsymbol{W}^{t-1})$ {Selectively retrain the previous network using Algorithm 2 }
  if $\mathcal{L}_t > \tau$ **then**
    $\boldsymbol{W}^t = DynamicExpansion(\boldsymbol{W}^t)$ {Expand the network capacity using Algorithm 3}
  $\boldsymbol{W}^t = Split(\boldsymbol{W}^t)$ {Split and duplicate the units using Algorithm 4 }



Selective Retraining　　Dynamic Expansion　　Split and Duplication

전체 알고리즘

Algorithm 1 Incremental Learning of a Dynamically Expandable Network

Input: Dataset $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_T)$, Thresholds τ, σ
Output: $\boldsymbol{W}^T$
for $t = 1, \ldots, T$ **do**
  if $t = 1$ **then**
    Train the network weights $\boldsymbol{W}^1$ using Eq. 2
  else
    $\boldsymbol{W}^t = SelectiveRetraining(\boldsymbol{W}^{t-1})$ {Selectively retrain the previous network using Algorithm 2 }
    if $\mathcal{L}_t > \tau$ **then**
      $\boldsymbol{W}^t = DynamicExpansion(\boldsymbol{W}^t)$ {Expand the network capacity using Algorithm 3}
  $\boldsymbol{W}^t = Split(\boldsymbol{W}^t)$ {Split and duplicate the units using Algorithm 4 }

첫번째 Task일 때는,
일반 DNN 학습하듯 학습하되,
l1-Regularizer를 통해 Sparse하게 학습함!

$$\underset{\boldsymbol{W}^{t=1}}{\text{minimize}}\, \mathcal{L}(\boldsymbol{W}^{t=1}; \mathcal{D}_t) + \mu \sum_{l=1}^{L} \|\boldsymbol{W}_l^{t=1}\|_1$$

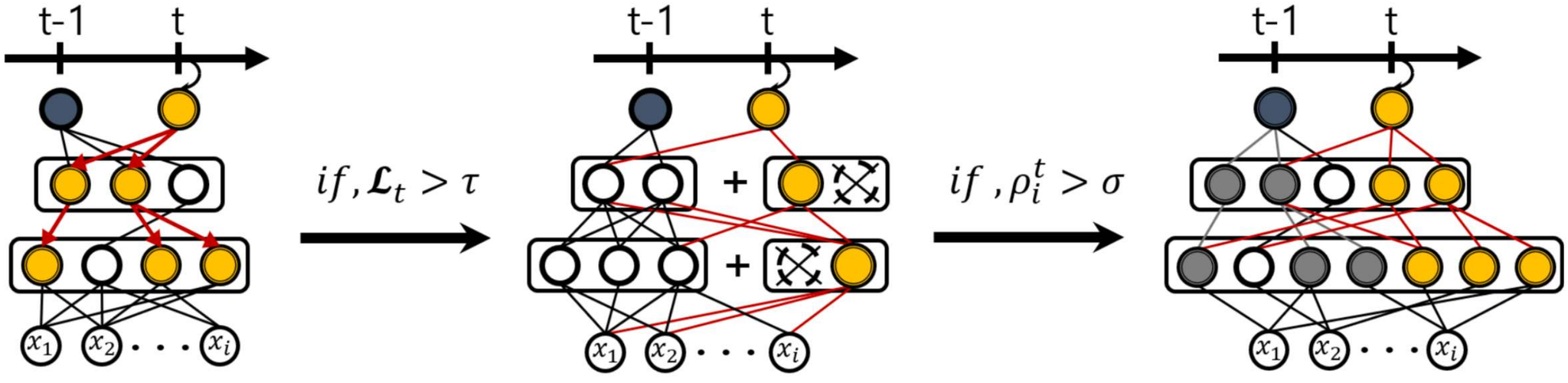
Dynamically Expandable Networks

전체 알고리즘

Algorithm 1 Incremental Learning of a Dynamically Expandable Network

Input: Dataset $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_T)$, Thresholds τ, σ
Output: W
for $t = 1, \ldots, T$ **do**
  if $t = 1$ **then**
    Train the network weights $\boldsymbol{W}^1$ using Eq. 2
  else
    $\boldsymbol{W}^t = SelectiveRetraining(\boldsymbol{W}^{t-1})$ {Selectively retrain the previous network using Algorithm 2 }
    if $\mathcal{L}_t > \tau$ **then**
      $\boldsymbol{W}^t = DynamicExpansion(\boldsymbol{W}^t)$ {Expand the network capacity using Algorithm 3}
    $\boldsymbol{W}^t = Split(\boldsymbol{W}^t)$ {Split and duplicate the units using Algorithm 4 }

두 번째 Task 부터는 이전 Task까지 학습한 Weight를 활용해 학습을 진행함!
학습은 총 3단계로 구성되는데,
1. Selective Retraining 2. Dynamic Expansion 3. Split and Duplication



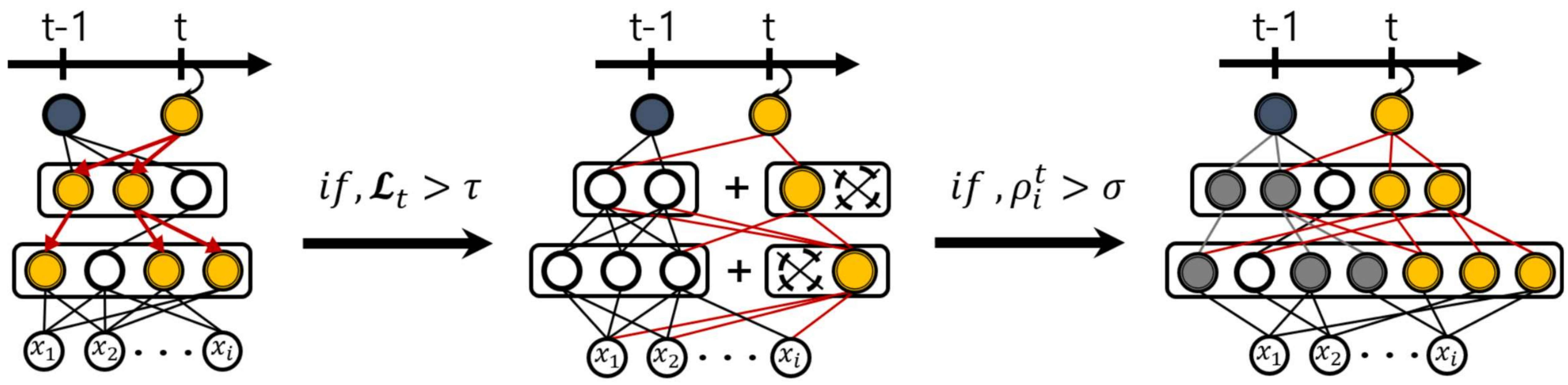
Selective Retraining　　Dynamic Expansion　　Split and Duplication

전체 알고리즘

Algorithm 1 Incremental Learning of a Dynamically Expandable Network

첫 번째 단계인 Selective Retraining은 말 그대로 선별적으로 재학습 하는 단계!
새로운 Task를 학습시키기 위해, 주요한 Weight만을 업데이트하는 과정.
모든 Weight를 학습시키지 않고 몇 개의 edge만을 선별적으로 업데이트 한다.
(왜 선별적으로 업데이트 하는지는 Split and Duplication 까지 설명을 들으면 알 수 있음.)

Input: Dataset $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_T)$, Threshold τ
Output: $\boldsymbol{W}^T$
for $t = 1, \ldots, T$ **do**
  if $t = 1$ **then**
    Train the network weights $\boldsymbol{W}^1$ using Eq. 2
  else
    $\boldsymbol{W}^t = SelectiveRetraining(\boldsymbol{W}^{t-1})$ {Selectively retrain the previous network using Algorithm 2 }
    if $\mathcal{L}_t > \tau$ **then**
      $\boldsymbol{W}^t = DynamicExpansion(\boldsymbol{W}^t)$ {Expand the network capacity using Algorithm 3}
    $\boldsymbol{W}^t = Split(\boldsymbol{W}^t)$ {Split and duplicate the units using Algorithm 4 }



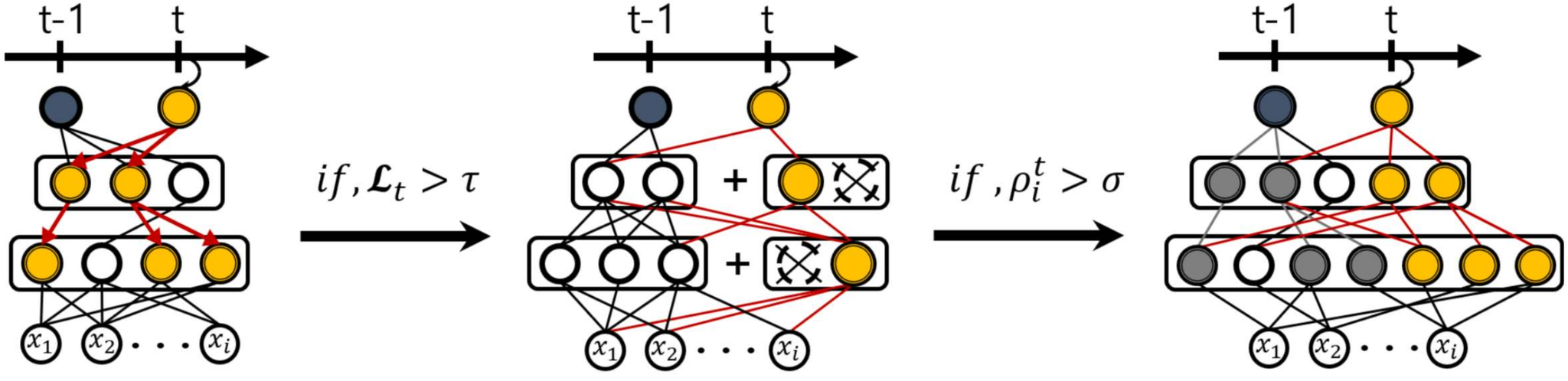
Selective Retraining | Dynamic Expansion | Split and Duplication

전체 알고리즘

Algorithm 1 Incremental Learning of a Dynamically Expandable Network

Input: Dataset $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_T)$, Thresholds τ, σ
Output: $\boldsymbol{W}^T$
for $t = 1, \ldots, T$ **do**
  if $t == 1$ **then**
    Train the network weights $\boldsymbol{W}^t$ using Eq. 2
  else
    $\boldsymbol{W}^t = SelectiveRetraining(\boldsymbol{W}^{t-1})$ {Selectively retrain the previous network using Algorithm 2 }
    if $\mathcal{L}_t > \tau$ **then**
      $\boldsymbol{W}^t = DynamicExpansion(\boldsymbol{W}^t)$ {Expand the network capacity using Algorithm 3}
    $\boldsymbol{W}^t = Split(\boldsymbol{W}^t)$ {Split and duplicate the units using Algorithm 4 }

두 번째 단계인 Dynamic Expansion은
만약, Selective Retraining으로 학습을 시켰는데,
새 Task에 대한 학습이 제대로 일어나지 않았다면(=새 Task에 대한 loss가 너무 크다면),
네트워크 내에 hidden unit을 추가하는 과정이다.



Selective Retraining　　Dynamic Expansion　　Split and Duplication

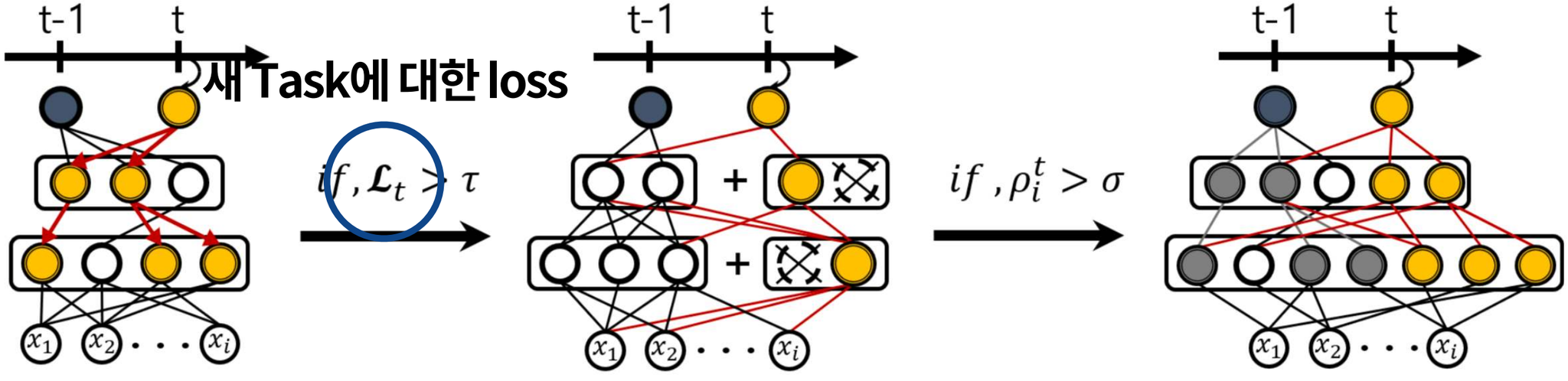
Dynamically Expandable Networks

전체 알고리즘

Algorithm 1 Incremental Learning of a Dynamically Expandable Network

Input: Dataset $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_T)$, Thresholds τ, σ
Output: $\boldsymbol{W}^T$
for $t = 1, \ldots, T$ **do**
  if $t == 1$ **then**
    Train the network weights $\boldsymbol{W}^t$ using Eq. 2
  else
    $\boldsymbol{W}^t = SelectiveRetraining(\boldsymbol{W}^{t-1})$ {Selectively retrain the previous network using Algorithm 2 }
    if $\mathcal{L}_t > \tau$ **then**
      $\boldsymbol{W}^t = DynamicExpansion(\boldsymbol{W}^t)$ {Expand the network capacity using Algorithm 3}
    $\boldsymbol{W}^t = Split(\boldsymbol{W}^t)$ {Split and duplicate the units using Algorithm 4 }

두 번째 단계인 Dynamic Expansion은
만약, Selective Retraining으로 학습을 시켰는데,
새 Task에 대한 학습이 제대로 일어나지 않았다면(=새 Task에 대한 loss가 너무 크다면),
네트워크 내에 hidden unit을 추가하는 과정이다.



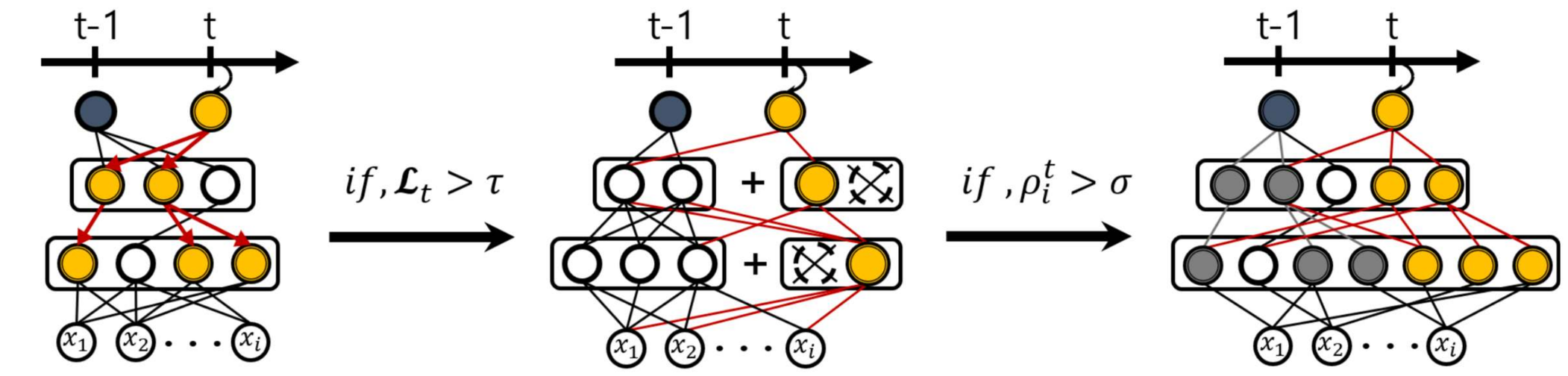
Selective Retraining　　Dynamic Expansion　　Split and Duplication

전체 알고리즘

Algorithm 1 Incremental Learning of a Dynamically Expandable Network

Input: Dataset $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_T)$, Thresholds τ, σ
Output: $\boldsymbol{W}^T$
for $t = 1, \ldots, T$ **do**
  if $t = 1$ **then**
    Train the network weights W^1 using Eq. 2
  else
    $\boldsymbol{W}^t = SelectiveRetraining(\boldsymbol{W}^{t-1})$ {Selectively retrain the previous network using Algorithm 2 }
    if $\mathcal{L}_t > \tau$ **then**
      $\boldsymbol{W}^t = DynamicExpansion(\boldsymbol{W}^t)$ {Expand the network capacity using Algorithm 3}
    $\boldsymbol{W}^t = Split(\boldsymbol{W}^t)$ {Split and duplicate the units using Algorithm 4 }

세 번째 단계인 Split and Duplication은
Selective Retraining과정에서 너무 많이 바뀐 edge들을
떼서 새로운 hidden units을 만들어주는 과정이다.
이 과정으로 catastrophic forgetting을 방지할 수 있다.



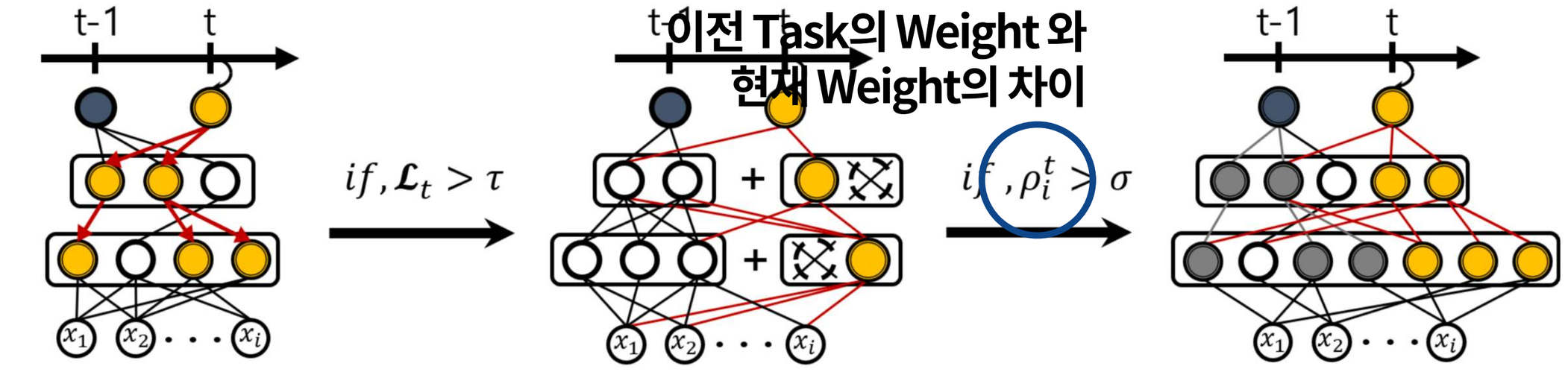
Selective Retraining　　Dynamic Expansion　　Split and Duplication

전체 알고리즘

Algorithm 1 Incremental Learning of a Dynamically Expandable Network

Input: Dataset $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_T)$, Thresholds τ, σ
Output: $\boldsymbol{W}^T$
for $t = 1, \ldots, T$ **do**
  if $t = 1$ **then**
    Train the network weights W_1 using Eq. 2
  else
    $\boldsymbol{W}^t = SelectiveRetraining(\boldsymbol{W}^{t-1})$ {Selectively retrain the previous network using Algorithm 2 }
    if $\mathcal{L}_t > \tau$ **then**
      $\boldsymbol{W}^t = DynamicExpansion(\boldsymbol{W}^t)$ {Expand the network capacity using Algorithm 3}
      $\boldsymbol{W}^t = Split(\boldsymbol{W}^t)$ {Split and duplicate the units using Algorithm 4 }

세 번째 단계인 Split and Duplication은
Selective Retraining과정에서 너무 많이 바뀐 edge들을
떼서 새로운 hidden units을 만들어주는 과정이다.
이 과정으로 catastrophic forgetting을 방지할 수 있다.



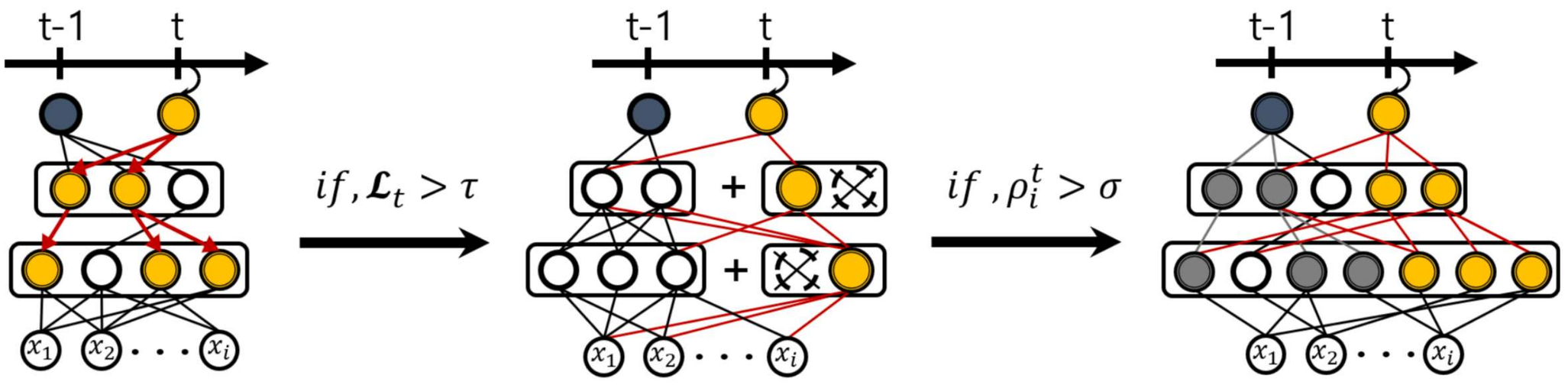
Selective Retraining　　Dynamic Expansion　　Split and Duplication

전체 알고리즘

Algorithm 1 Incremental Learning of a Dynamically Expandable Network

Q. 왜 Selective Retraining에서 선별적으로 업데이트 했을까?

Input: Dataset $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_T)$, Thresholds τ, σ
Output: $\boldsymbol{W}^T$
for $t = 1, \ldots, T$ **do**
  if $t = 1$ **then**
    Train the network weights $\boldsymbol{W}^1$ using Eq. 2
  else
    $\boldsymbol{W}^t = SelectiveRetraining(\boldsymbol{W}^{t-1})$ {Selectively retrain the previous network using Algorithm 2 }
    if $\mathcal{L}_t > \tau$ **then**
      $\boldsymbol{W}^t = DynamicExpansion(\boldsymbol{W}^t)$ {Expand the network capacity using Algorithm 3}
    $\boldsymbol{W}^t = Split(\boldsymbol{W}^t)$ {Split and duplicate the units using Algorithm 4 }



Selective Retraining　　Dynamic Expansion　　Split and Duplication

전체 알고리즘

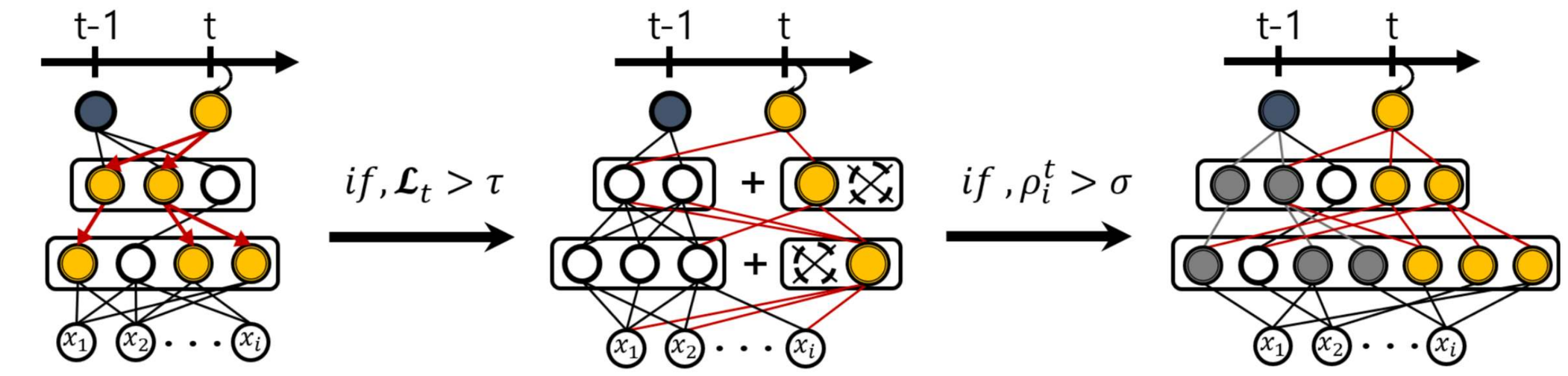
Algorithm 1 Incremental Learning of a Dynamically Expandable Network

Input: Dataset $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_T)$, Thresholds τ, σ
Output: $\boldsymbol{W}$
for $t = 1, \ldots, T$ **do**
  if $t == 1$ **then**
    Train the network weights $\boldsymbol{W}^1$ using Eq. 2
  else
    $\boldsymbol{W}^t = SelectiveRetraining(\boldsymbol{W}^{t-1})$ {Selectively retrain the previous network using Algorithm 2 }
    if $\mathcal{L}_t > \tau$ **then**
      $\boldsymbol{W}^t = DynamicExpansion(\boldsymbol{W}^t)$ {Expand the network capacity using Algorithm 3}
    $\boldsymbol{W}^t = Split(\boldsymbol{W}^t)$ {Split and duplicate the units using Algorithm 4 }

Q. 왜 Selective Retraining에서 선별적으로 업데이트 했을까?

선별적으로 업데이트하지 않았다면, Split and Duplication 단계에서 복제해야할 hidden unit이 너무 많아질 수 있음.

그러면, 효율적인 학습이라는 lifelong learning의 목표를 달성할 수 없음.



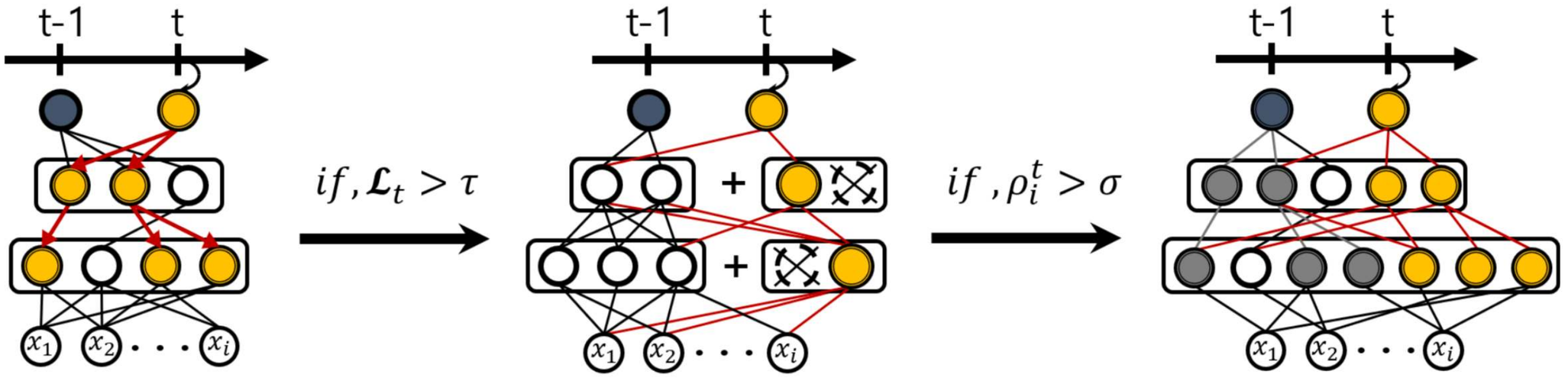
Selective Retraining　　Dynamic Expansion　　Split and Duplication

전체 알고리즘

Q. 왜 Split and Duplication 단계에서 Selective Retraining에서 바뀐 모든 weight에 해당하는 unit을 업데이트 하지 않고, 일정 threshold 이상 바뀐 것에 대해서만 복제 했을까?

Algorithm 1 Incremental Learning of a Dynamically Expandable Network

Input: Dataset $D = (D_1, \ldots, D_T)$, Thresholds τ, σ
Output: $\boldsymbol{W}^T$
for $t = 1, \ldots, T$ **do**
  if $t = 1$ **then**
    Train the network weights $\boldsymbol{W}^1$ using Eq. 2
  else
    $\boldsymbol{W}^t = SelectiveRetraining(\boldsymbol{W}^{t-1})$ {Selectively retrain the previous network using Algorithm 2 }
    if $\mathcal{L}_t > \tau$ **then**
      $\boldsymbol{W}^t = DynamicExpansion(\boldsymbol{W}^t)$ {Expand the network capacity using Algorithm 3}
    $\boldsymbol{W}^t = Split(\boldsymbol{W}^t)$ {Split and duplicate the units using Algorithm 4 }



Selective Retraining　　Dynamic Expansion　　Split and Duplication

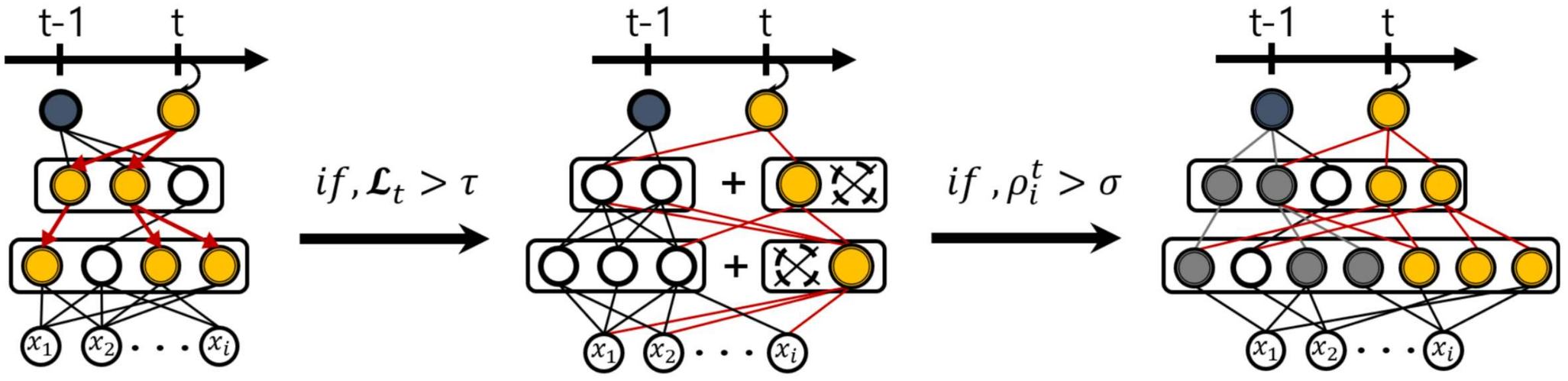
전체 알고리즘

Q. 왜 Split and Duplication 단계에서 Selective Retraining에서 바뀐 모든 weight에 해당하는 unit을 업데이트 하지 않고, 일정 threshold 이상 바뀐 것에 대해서만 복제 했을까?

모든 복제하게 되면 Catastrophic Forgetting은 전혀 일어나지 않겠지만,
메모리 비효율 적이고, Backward knowledge transfer가 일어날 수 없다.
일정 threshold 이상 바뀐 것만 복제함으로써 Catastrophic Forgetting의 여지가 생기면서
Backward knowledge transfer의 여지가 생기는 것임!

Algorithm 1 Incremental Learning of a Dynamically Expandable Network

Input: Dataset $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_T)$, Thresholds τ, σ
Output: $\boldsymbol{W}$
for $t = 1, \ldots, T$ **do**
  if $t = 1$ **then**
    Train the network with $\boldsymbol{W}^{t-1}$ for $\mathcal{D}_t$
  else
    $\boldsymbol{W}^t = SelectiveRetraining(\boldsymbol{W}^{t-1})$ {Selectively retrain the previous network using Algorithm 2 }
    if $\mathcal{L}_t > \tau$ **then**
      $\boldsymbol{W}^t = DynamicExpansion(\boldsymbol{W}^t)$ {Expand the network capacity using Algorithm 3}
    $\boldsymbol{W}^t = Split(\boldsymbol{W}^t)$ {Split and duplicate the units using Algorithm 4 }

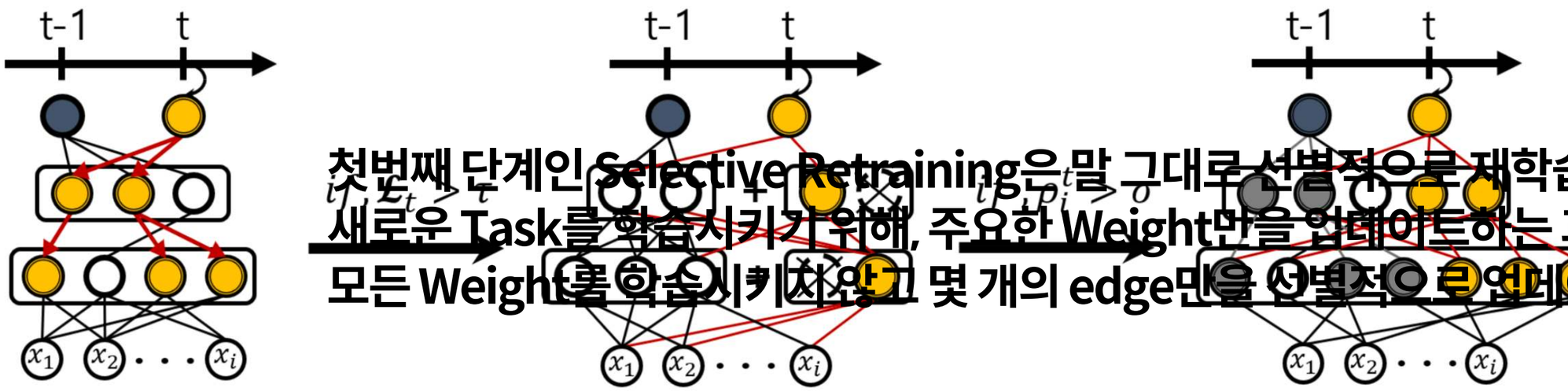


Selective Retraining　　Dynamic Expansion　　Split and Duplication

Selective Retraining

Algorithm 2 Selective Retraining

Input: Datatset $\mathcal{D}_t$, Previous parameter $\boldsymbol{W}^{t-1}$
Output: network parameter $\boldsymbol{W}^t$
Initialize $l \leftarrow L-1$, $S = \{o_t\}$
Solve Eq. 3 to obtain $\boldsymbol{W}^t_{L,t}$
Add neuron i to S if the weight between i and o_t in $\boldsymbol{W}^t_{L,t}$ is not zero.
for $l = L-1, \ldots, 1$ **do**
  Add neuron i to S if there exists some neuron $j \in S$ such that $\boldsymbol{W}^{t-1}_{l,ij} \neq 0$.
Solve Eq. 4 to obtain $\boldsymbol{W}^t_S$



Selective Retraining　　Dynamic Expansion　　Split and Duplication

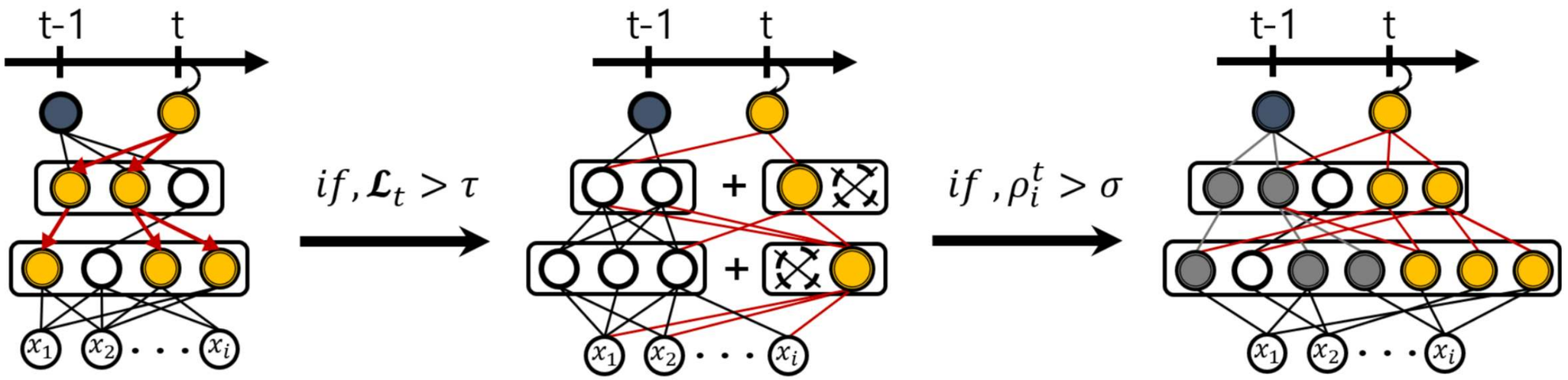
Selective Retraining

Algorithm 2 Selective Retraining

Input: Datatset D_t, Previous parameter W
Output: network parameter $\boldsymbol{W}^t$
Initialize $l \leftarrow L-1$, $S = \{o_t\}$
Solve Eq. 3 to obtain $\boldsymbol{W}^t_{L,t}$
Add neuron i to S if the weight between i and o_t in $\boldsymbol{W}^t_{L,t}$ is not zero
for $l = L-1, \ldots, 1$ **do**
  Add neuron i to S if there exists some neuron $j \in S$ such that $\boldsymbol{W}^{t-1}_{l,ij} \neq 0$.
Solve Eq. 4 to obtain $\boldsymbol{W}^t_S$

$$\underset{\boldsymbol{W}^t_{L,t}}{\text{minimize}}\, \mathcal{L}(\boldsymbol{W}^t_{L,t}; \boldsymbol{W}^{t-1}_{1:L-1}, \mathcal{D}_t) + \mu \|\boldsymbol{W}^t_{L,t}\|_1$$

l1-regularizer를 통해 주요한 역할을 하는 edge들을 찾아냄.
output layer부터 input layer로 가면서 차례로 $W_{l,t}^t$를 구해냄.
($W_{l,t}^t$는 l번째 layer에서 task t의 Weight 값)



Selective Retraining　　Dynamic Expansion　　Split and Duplication

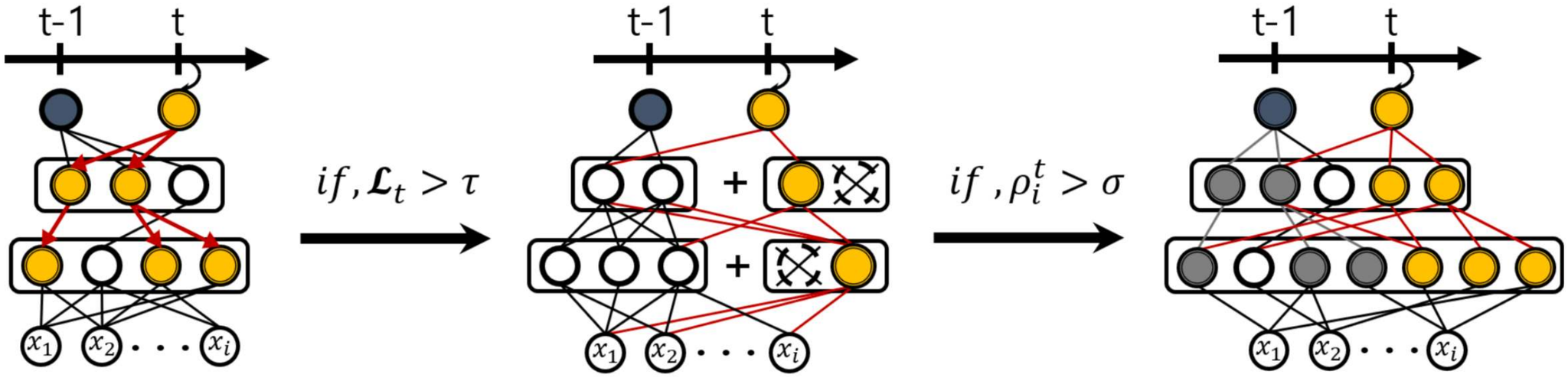
Selective Retraining

Algorithm 2 Selective Retraining

Input: Datatset D_t, Previous parameter W^{t-1}
Output: network parameter W^t
Initialize $l \leftarrow L-1$, $S = \{o_t\}$
Solve Eq. 3 to obtain $W^t_{L,t}$
Add neuron i to S if the weight between i and o_t is not zero
for $l = L-1, \ldots, 1$ **do**
  Add neuron i to S if there exists some neuron $j \in S$ such that $W^{t-1}_{i,j} \neq 0$
Solve Eq. 4 to obtain W^t_S

$$\underset{W^t_{L,t}}{\text{minimize}}\ \mathcal{L}(W^t_{L,t}; W^{t-1}_{1:L-1}, \mathcal{D}_t) + \mu \|W^t_{L,t}\|_1$$

l1-regularizer를 통해 주요한 역할을 하는 edge들을 찾아냄.
output layer부터 input layer로 가면서 차례로 $W_{l,t}^t$를 구해냄.
($W_{l,t}^t$는 l번째 layer에서 task t의 Weight 값)
알고리즘에서는 Final layer와 그 전 layer의 weight 값을 구하는 것처럼 서술했지만, 사실은 모든 Weight에 대해서 미리 다 구해두어야 한다.



Selective Retraining　　Dynamic Expansion　　Split and Duplication

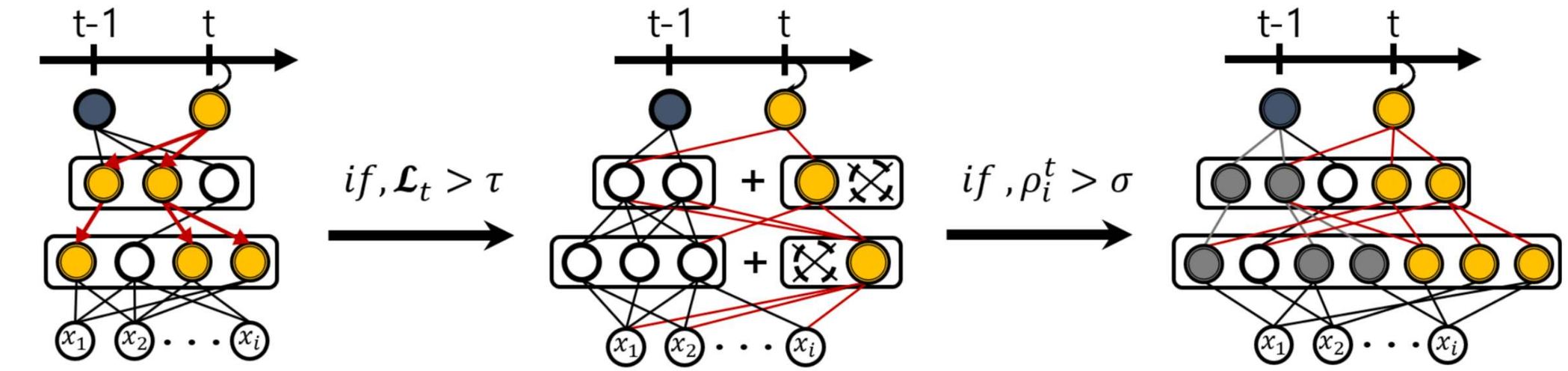
Selective Retraining

Algorithm 2 Selective Retraining

Input: Datatset $\mathcal{D}_t$, Previous parameter $\boldsymbol{W}^{t-1}$
Output: network parameter $\boldsymbol{W}^t$
Initialize $l \leftarrow L-1$, $S = \{o_t\}$
Solve Eq. 3 to obtain $\boldsymbol{W}^t_{L,t}$

Weight가 바뀐 Edge를 찾았으면,
Edge에 연결된 hidden units을 찾는 단계.

Add neuron i to S if the weight between i and o_t in $\boldsymbol{W}^t_{L,t}$ is not zero.
for $l = L-1, \ldots, 1$ **do**
  Add neuron i to S if there exists some neuron $j \in S$ such that $\boldsymbol{W}^{t-1}_{l,ij} \neq 0$.
Solve Eq. 4 to obtain $\boldsymbol{W}^t_S$



Selective Retraining　　Dynamic Expansion　　Split and Duplication

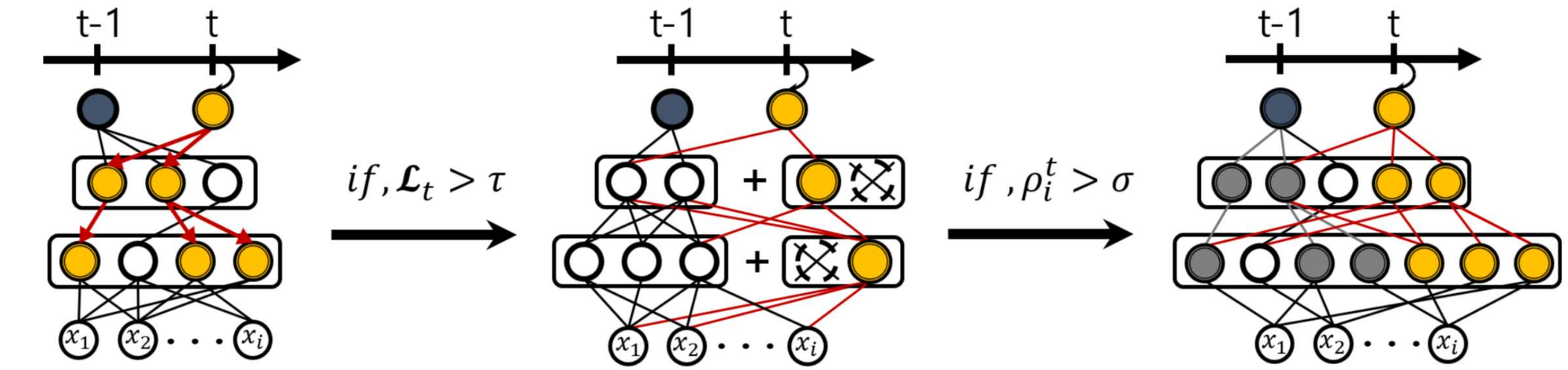
Selective Retraining

Algorithm 2 Selective Retraining

Input: Datatset $\mathcal{D}_t$, Pre
Output: network param
Initialize $l \leftarrow L-1$, S
Solve Eq. 3 to obtain $\boldsymbol{W}_{L,t}$
Add neuron i to S if the weight between it and [illegible] is not zero.
for $l = L-1, \ldots, 1$ **do**
  Add neuron i to S if there exists some neuron $j \in S$ such that $\boldsymbol{W}_{l,ij}^{t-1} \neq 0$.
Solve Eq. 4 to obtain $\boldsymbol{W}_S^t$

$$\underset{\boldsymbol{W}_S^t}{\text{minimize}}\, \mathcal{L}(\boldsymbol{W}_S^t ; \boldsymbol{W}_{S^c}^{t-1}, \mathcal{D}_t) + \mu \|\boldsymbol{W}_S^t\|_2$$

논문의 서술에 따르면 위 식보다는



Selective Retraining　　Dynamic Expansion　　Split and Duplication

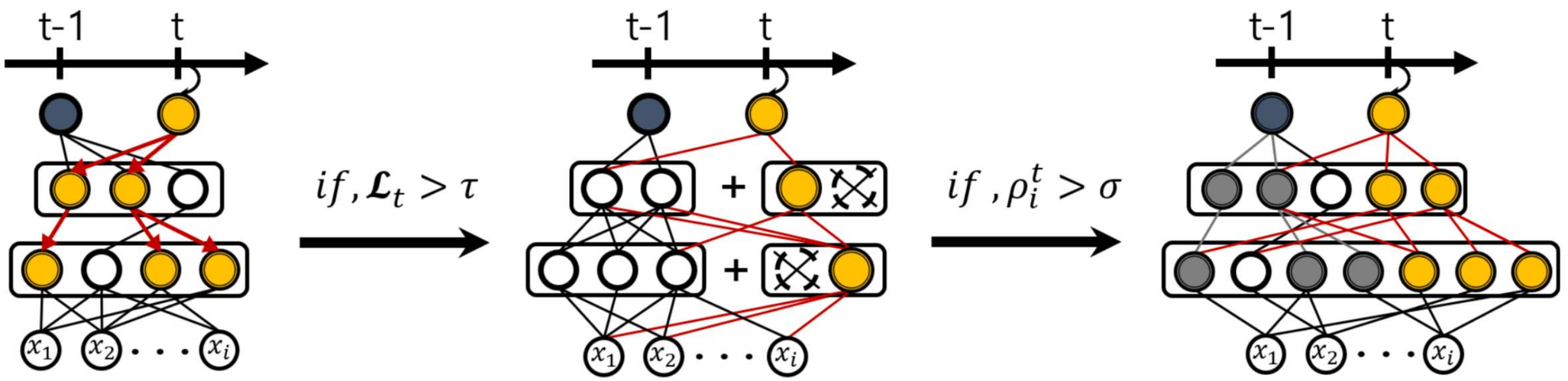
Selective Retraining

Algorithm 2 Selective Retraining

Input: Datatset $\mathcal{D}_t$, Pre
Output: network param
Initialize $l \leftarrow L-1$, S
Solve Eq. 3 to obtain $\boldsymbol{W}_{L,t}$
Add neuron i to S if the weight between $a_{L,i}$ and $\boldsymbol{W}_{L,t}$ is not zero.
for $l = L-1, \ldots, 1$ **do**
  Add neuron i to S if there exists some neuron $j \in S$ such that $W^t_{l,ij} \neq 0$.
Solve Eq. 4 to obtain $\boldsymbol{W}^t_S$

$$\underset{\boldsymbol{W}^t_S}{\text{minimize}}\ \mathcal{L}(\boldsymbol{W}^t_S ; \boldsymbol{W}^{t-1}_S, \mathcal{D}_t) + \mu\|\boldsymbol{W}^t_S\|_2$$

이 식으로 보는 것이 적절할 것!
W_S^t 를 재 학습하는데, W_S^{t-1}을 초기값으로 주고 재학습.
W_S^t 자체가 너무 커지지 않도록 l2 regularization.



Selective Retraining　　Dynamic Expansion　　Split and Duplication

Dynamically Expandable Networks

Selective Retraining

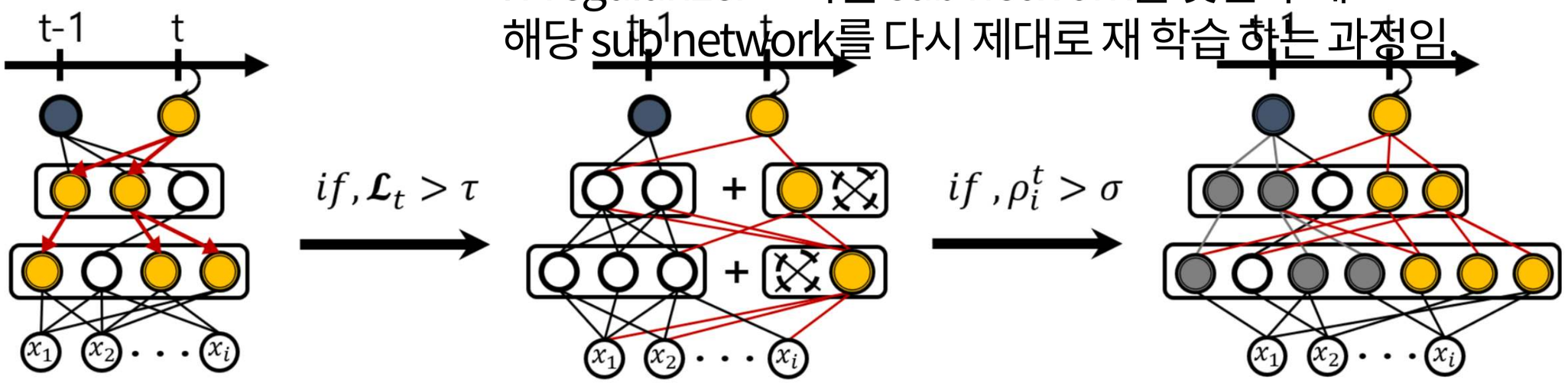
Algorithm 2 Selective Retraining

Input: Datatset $\mathcal{D}_t$, Pre
Output: network param
Initialize $l \leftarrow L-1$, S
Solve Eq. 3 to obtain $\boldsymbol{W}_{L,t}$
Add neuron i to S if the weight between i and $o_{L,t}$ is not zero.
for $l = L-1, \ldots, 1$ **do**
  Add neuron i to S if there exists some neuron $j \in S$ such that $W^t_{l,ij} \neq 0$.
  Solve Eq. 4 to obtain $\boldsymbol{W}^t_S$

$$\underset{\boldsymbol{W}^t_S}{\text{minimize}}\, \mathcal{L}(\boldsymbol{W}^t_S ; \boldsymbol{W}^{t-1}_S, \mathcal{D}_t) + \mu \|\boldsymbol{W}^t_S\|_2$$

이 식으로 보는 것이 적절할 것!
W_S^t 를 재 학습하는데, W_S^{t-1}을 초기값으로 주고 재학습.
W_S^t 자체가 너무 커지지 않도록 l2 regularization.

극히 일부 subnework에 대해서만 학습하므로 빠르게 끝남.
l1 regularizer로 바꿀 sub network를 찾은 후에
해당 sub network를 다시 제대로 재 학습 하는 과정임.



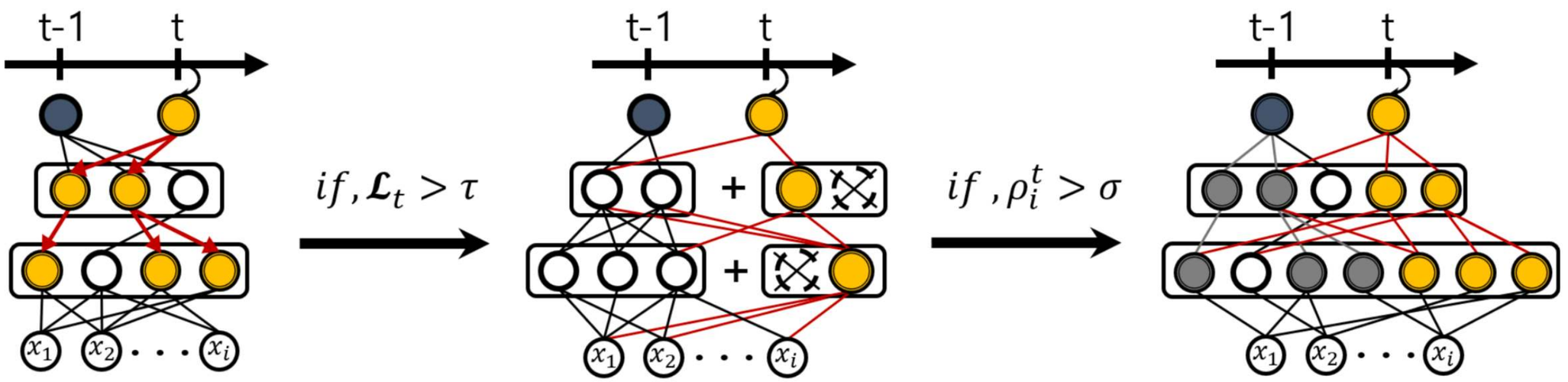
Selective Retraining　　Dynamic Expansion　　Split and Duplication

Dynamically Expansion

Algorithm 3 Dynamic Network Expansion

Input: Datatset $\mathcal{D}_t$, Threshold τ
Perform Algorithm 2 and compute $\mathcal{L}$
if $\mathcal{L} > \tau$ **then**
  Add k units $\boldsymbol{h}^{\mathcal{N}}$ at all layers
  Solve for Eq. 5 at all layers
for $l = L-1, \ldots, 1$ **do**
  Remove useless units in $\boldsymbol{h}_l^{\mathcal{N}}$

두 번째 단계인 Dynamic Expansion은
만약, Selective Retraining으로 학습을 시켰는데,
새 Task에 대한 학습이 제대로 일어나지 않았다면
(=새 Task에 대한 loss가 너무 크다면),
네트워크 내에 hidden unit을 추가하는 과정이다.



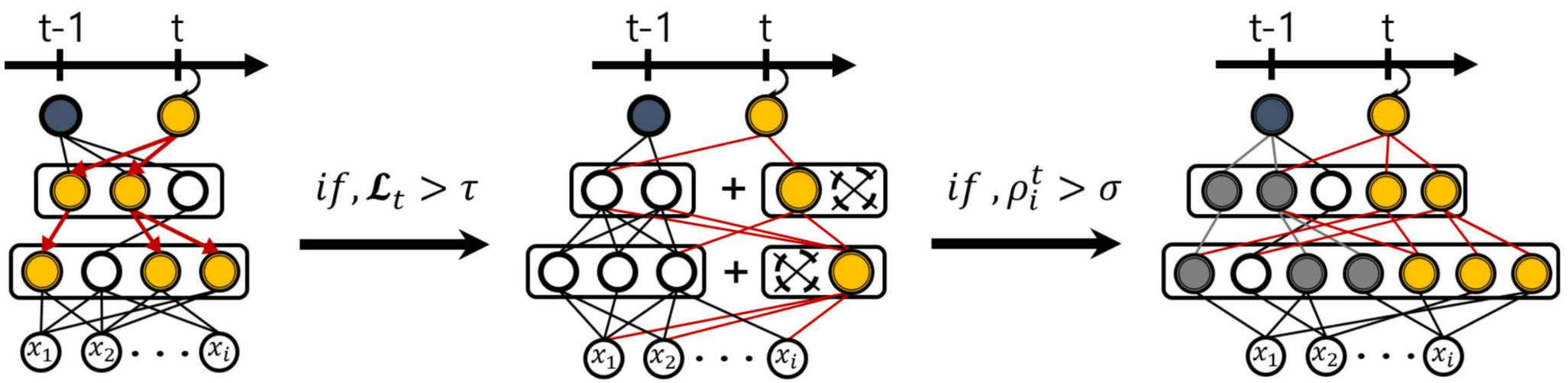
Selective Retraining　　Dynamic Expansion　　Split and Duplication

Dynamically Expansion

Algorithm 3 Dynamic Network Expansion

Input: Datatset $\mathcal{D}_t$, Threshold τ
Perform Algorithm 2 and compute $\mathcal{L}$
if $\mathcal{L} > \tau$ **then**
  Add k units $\boldsymbol{h}^{\mathcal{N}}$ at all layers
  Solve for Eq. 5 at all layers
for $l = L-1, \ldots, 1$ **do**
  Remove useless units in $\boldsymbol{h}_l^{\mathcal{N}}$

Selective Retraining을 했는데,
새 Task에 대한 학습이 제대로 일어나지 않았다면
(=새 Task에 대한 loss가 너무 크다면),
네트워크 내에 고정된 k개의 hidden unit을 추가!



Selective Retraining　　Dynamic Expansion　　Split and Duplication

Dynamically Expansion

Algorithm 3 Dynamic Net

Input: Datatset $\mathcal{D}_t$, Thresh

Perform Algorithm 2 and compute $\mathcal{L}$

if $\mathcal{L} > \tau$ **then**

Add k units $\boldsymbol{h}^{\mathcal{N}}$ at all layers

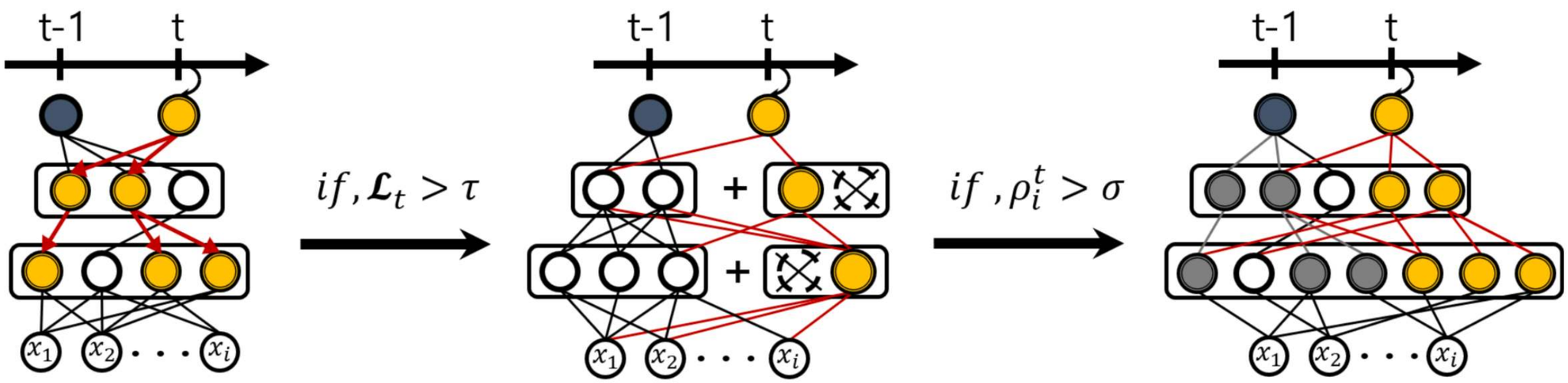
Solve for Eq. 5 at all layers

for $l = L-1, \ldots, 1$ **do**

Remove useless units in $\boldsymbol{h}_l^{\mathcal{N}}$

$$\underset{\boldsymbol{W}_l^{\mathcal{N}}}{\text{minimize}}\ \mathcal{L}(\boldsymbol{W}_l^{\mathcal{N}}; \boldsymbol{W}_l^{t-1}, \mathcal{D}_t) + \mu\|\boldsymbol{W}_l^{\mathcal{N}}\|_1 + \gamma \sum_g \|\boldsymbol{W}_{l,g}^{\mathcal{N}}\|_2$$

네트워크 내에 고정된 k개의 hidden unit을 추가한 이후에,
추가된 hidden unit들을 학습시키는 단계!



Selective Retraining　　Dynamic Expansion　　Split and Duplication

Dynamically Expansion

Algorithm 3 Dynamic Net

Input: Datatset $\mathcal{D}_t$, Thresh

Perform Algorithm 2 and compute $\mathcal{L}$

if $\mathcal{L} > \tau$ **then**

Add k units $\boldsymbol{h}^{\mathcal{N}}$ at all layers

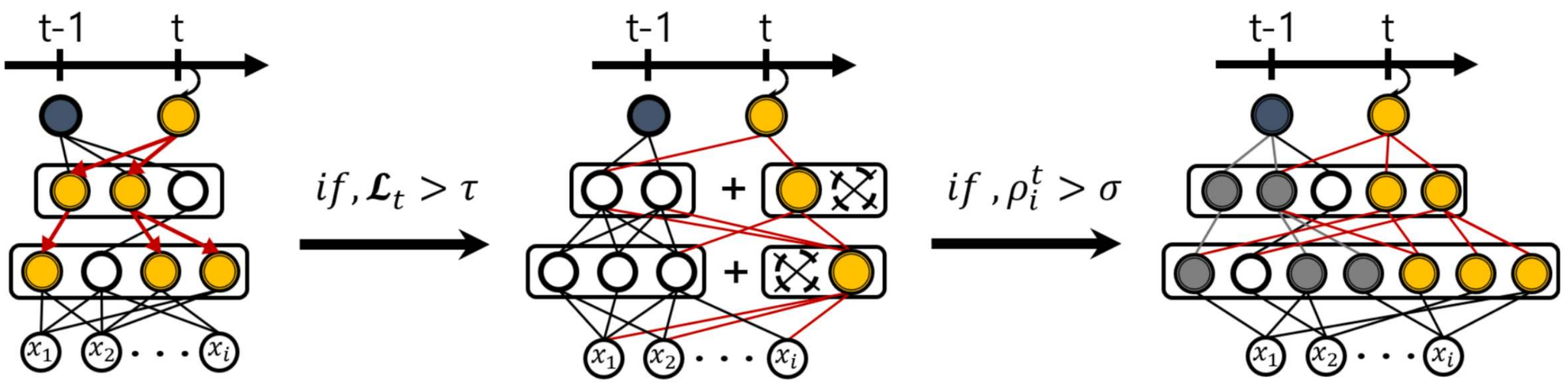
Solve for Eq. 5 at all layers

for $l = L-1, \ldots, 1$ **do**

Remove useless units in $\boldsymbol{h}_l^{\mathcal{N}}$

$$\underset{\boldsymbol{W}_l^{\mathcal{N}}}{\text{minimize}}\ \mathcal{L}(\boldsymbol{W}_l^{\mathcal{N}}; \boldsymbol{W}_l^{t-1}, \mathcal{D}_t) + \mu\|\boldsymbol{W}_l^{\mathcal{N}}\|_1 + \gamma \sum_g \|\boldsymbol{W}_{l,g}^{\mathcal{N}}\|_2$$

여기서도 l1-regularizer로 Weight가 Sparse하게 학습시킴.



Selective Retraining　　Dynamic Expansion　　Split and Duplication

Dynamically Expansion

Algorithm 3 Dynamic Net

Input: Datatset $\mathcal{D}_t$, Thresh

Perform Algorithm 2 and compute $\mathcal{L}$

if $\mathcal{L} > \tau$ **then**

Add k units $\boldsymbol{h}^{\mathcal{N}}$ at all layers

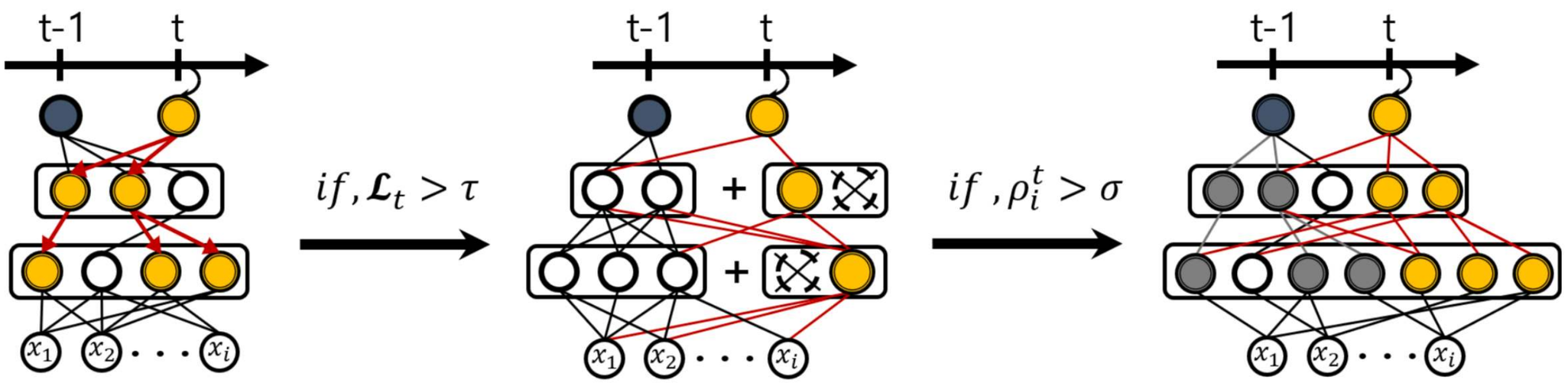
Solve for Eq. 5 at all layers

for $l = L-1, \ldots, 1$ **do**

Remove useless units in $\boldsymbol{h}_l^{\mathcal{N}}$

$$\underset{\boldsymbol{W}_l^{\mathcal{N}}}{\text{minimize}}\ \mathcal{L}(\boldsymbol{W}_l^{\mathcal{N}}; \boldsymbol{W}_l^{t-1}, \mathcal{D}_t) + \mu\|\boldsymbol{W}_l^{\mathcal{N}}\|_1 + \gamma \sum_g \|\boldsymbol{W}_{l,g}^{\mathcal{N}}\|_2$$

Group Sparsity Regularizer. Wen et, al(2016)
g는 각 unit에 들어오는 incoming weights.
불필요한 units을 찾아내는 데 유용.



Selective Retraining　　Dynamic Expansion　　Split and Duplication

Dynamically Expansion

Algorithm 3 Dynamic Net

Input: Datatset $\mathcal{D}_t$, Thresh

Perform Algorithm 2 and compute $\mathcal{L}$

if $\mathcal{L} > \tau$ **then**

Add k units $\boldsymbol{h}^{\mathcal{N}}$ at all layers

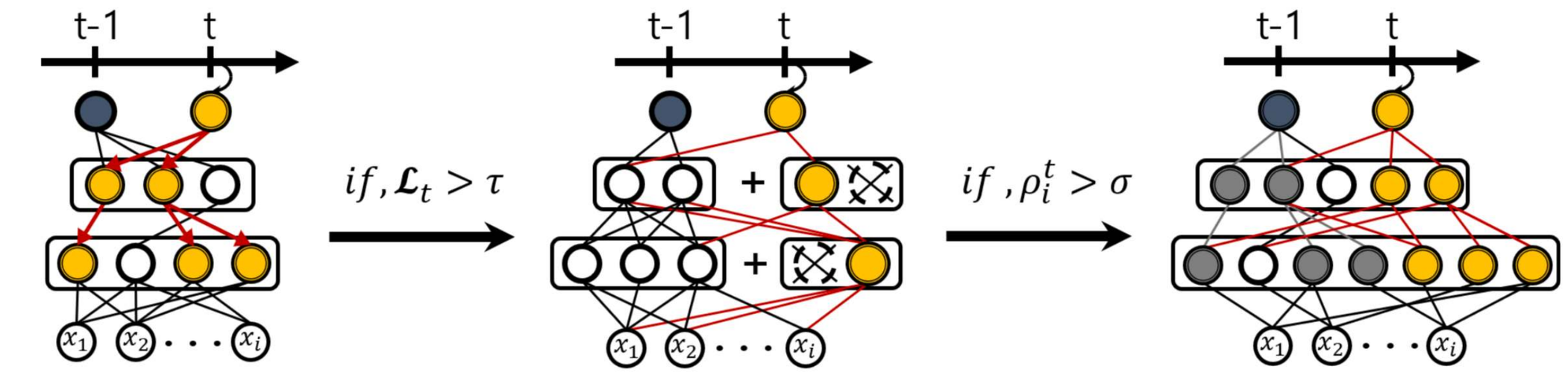
Solve for Eq. 5 at all layers

for $l = L-1, \ldots, 1$ **do**

Remove useless units in $\boldsymbol{h}_l^{\mathcal{N}}$

$$\underset{\boldsymbol{W}_l^{\mathcal{N}}}{\text{minimize}}\ \mathcal{L}(\boldsymbol{W}_l^{\mathcal{N}}; \boldsymbol{W}_l^{t-1}, \mathcal{D}_t) + \mu\|\boldsymbol{W}_l^{\mathcal{N}}\|_1 + \gamma \sum_g \|\boldsymbol{W}_{l,g}^{\mathcal{N}}\|_2$$

위 식을 통해 학습하므로, 불필요한 units을 찾아낼 수 있다.
따라서, 불필요한 units을 제거할 수 있음!
(메모리 효율성을 위해 불필요한 메모리 제거)



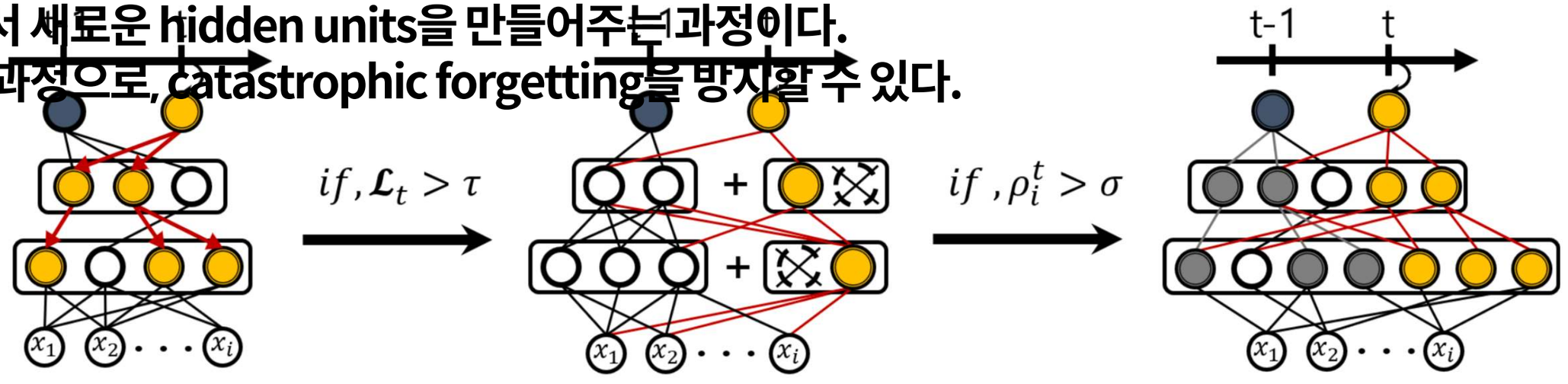
Selective Retraining　　Dynamic Expansion　　Split and Duplication

Split And Duplication

Algorithm 4 Network Split/Duplication

Input: Weight $\boldsymbol{W}^{t-1}$, Threshold σ
Perform Eq. 6 to obtain $\widetilde{\boldsymbol{W}}^t$
for all hidden unit i **do**
  $\rho_i^t = \|\boldsymbol{w}_i^t - \boldsymbol{w}_i^{t-1}\|_2$
  if $\rho_i^t > \sigma$ **then**
    Copy i into i' ($\boldsymbol{w}'$ introduction of edges for i')
Perform Eq. 6 with the initialization of $\widetilde{\boldsymbol{W}}^t$ to obtain $\boldsymbol{W}^t$

세 번째 단계인 Split and Duplication은
Selective Retraining 과정에서 너무 많이 바뀐 edge들을
떼서 새로운 hidden units을 만들어주는 과정이다.
이 과정으로, catastrophic forgetting을 방지할 수 있다.



Selective Retraining　　Dynamic Expansion　　Split and Duplication

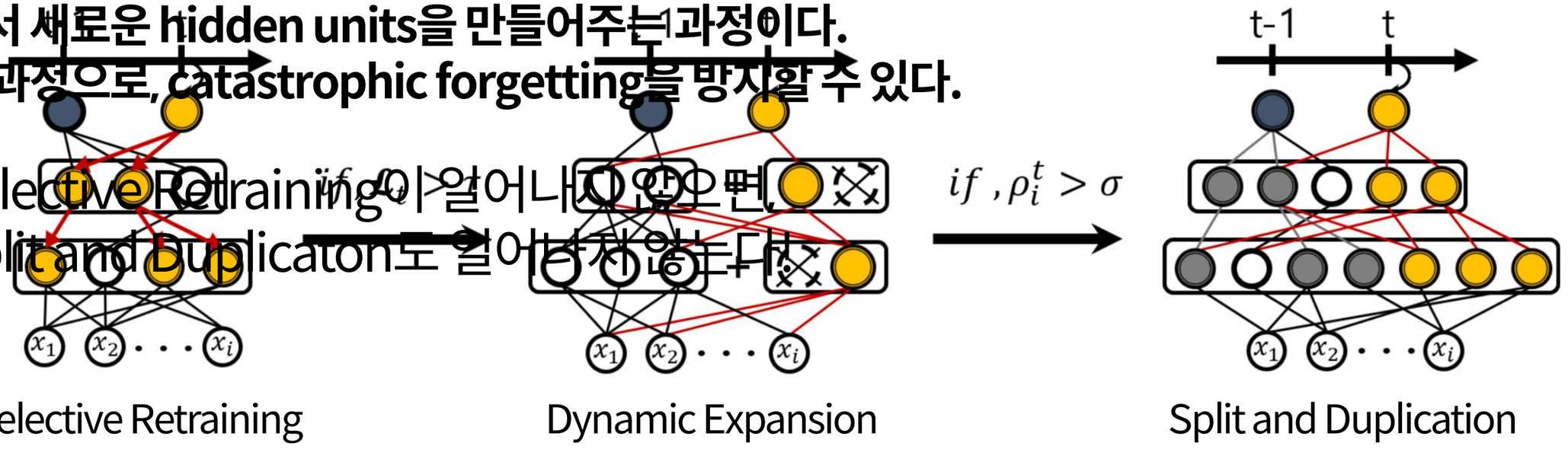
Split And Duplication

Algorithm 4 Network Split/Duplication

Input: Weight $\boldsymbol{W}^{t-1}$, Threshold σ
Perform Eq. 6 to obtain $\widetilde{\boldsymbol{W}}^t$
for all hidden unit i **do**
  $\rho_i^t = \|\boldsymbol{w}_i^t - \boldsymbol{w}_i^{t-1}\|_2$
  if $\rho_i^t > \sigma$ **then**
    Copy i into i' ($\boldsymbol{w}'$ introduction of edges for i')
Perform Eq. 6 with the initialization of $\widetilde{\boldsymbol{W}}^t$ to obtain $\boldsymbol{W}^t$

세 번째 단계인 Split and Duplication은
Selective Retraining 과정에서 너무 많이 바뀐 edge들을
떼서 새로운 hidden units을 만들어주는 과정이다.
이 과정으로, catastrophic forgetting을 방지할 수 있다.

Selective Retraining이 일어나지 않으면,
Split and Duplicaton도 일어나지 않는다.



Selective Retraining　　Dynamic Expansion　　Split and Duplication

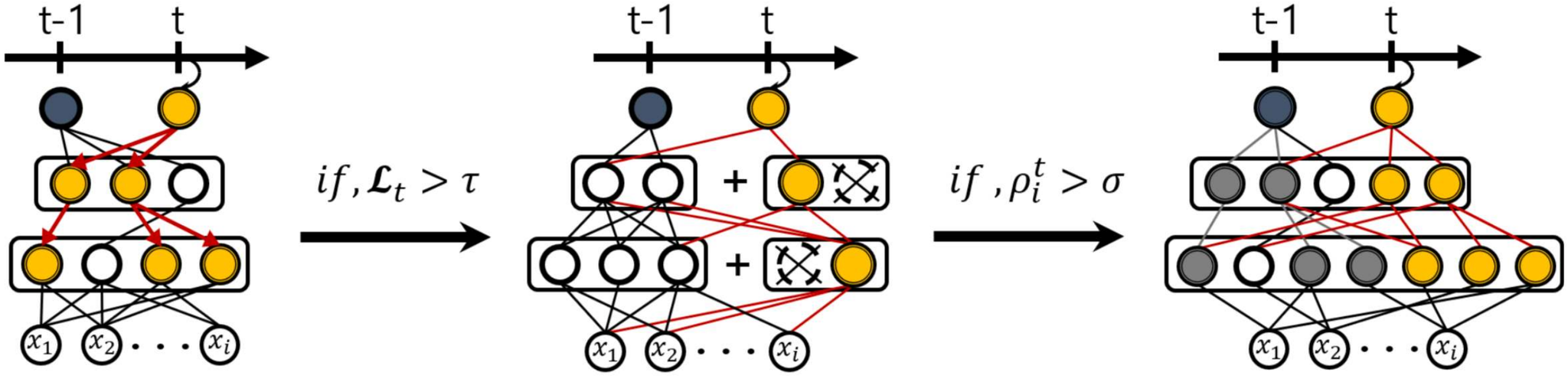
Dynamically Expandable Networks

Split And Duplication

Algorithm 4 Network Split/Duplication

Input: Weight $\boldsymbol{W}^{t-1}$, Threshold σ
Perform Eq. 6 to obtain $\widetilde{\boldsymbol{W}}^t$
for all hidden unit i **do**
  $\rho_i^t = \|\boldsymbol{w}_i^t - \boldsymbol{w}_i^{t-1}\|_2$
  if $\rho_i^t > \sigma$ **then**
    Copy i into i' ($\boldsymbol{w}'$ introduction of edges for i')
Perform Eq. 6 with the initialization of $\widetilde{\boldsymbol{W}}^t$ to obtain $\boldsymbol{W}^t$

모든 hidden units에 대해
이전 태스크까지 학습한 weight와 비교했을 때,
너무 많이 바뀐 weight가 있는 unit의 경우에는 복제한다.



Selective Retraining　　Dynamic Expansion　　Split and Duplication

Split And Duplication

Algorithm 4 Network Split/D

Input: Weight $\boldsymbol{W}^{t-1}$, Thresh

Perform Eq. 6 to obtain $\widetilde{\boldsymbol{W}}^t$

for all hidden unit i **do**

$\rho_i^t = \|\boldsymbol{w}_i^t - \boldsymbol{w}_i^{t-1}\|_2$

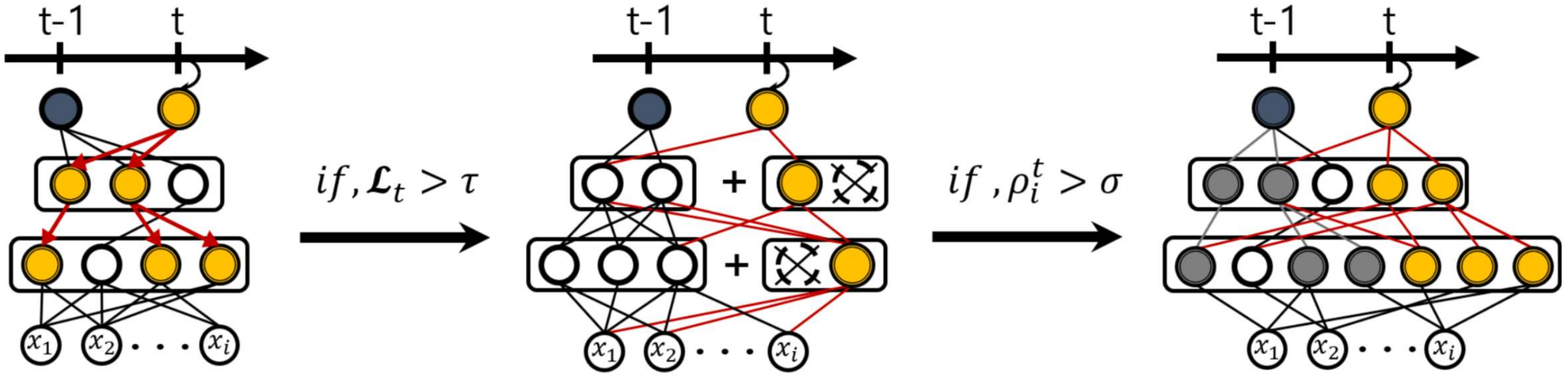
if $\rho_i^t > \sigma$ **then**

Copy i into i' ($\boldsymbol{w}'$ introduction of edges for i')

Perform Eq. 6 with the initialization of $\widetilde{\boldsymbol{W}}^t$ to obtain $\boldsymbol{W}^t$

$$\underset{\boldsymbol{W}^t}{\text{minimize}}\ \mathcal{L}(\boldsymbol{W}^t; \mathcal{D}_t) + \lambda\|\boldsymbol{W}^t - \boldsymbol{W}^{t-1}\|_2^2$$

복제를 진행한 이후에는 복제된 상태를 기준으로,
Fine tuning을 진행한다.



Selective Retraining　　Dynamic Expansion　　Split and Duplication

Dynamically Expandable Networks

Time Stamping

- 각 Task를 학습할 때마다 각 Task에 사용한 unit 수를 기록해 두는 것(레이어 마다).
- Time Stamping을 통해 해당 Task에 해당하는 unit만을 사용해 각 Task를 수행.

- 새로운 Hidden unit의 추가로 인한 기존 업무의 치명적 망각을 방지.
- [9]가 제시한 각 학습 단계까지 학습한 Weight를 저장하는 것보다 더 유연한 전략.

- [9]와 달리 DEN에서는 Backward Transfer가 일어날 수 있음.
  - 분할되지 않았지만, 추후 Task에서 학습된 다른 unit들로부터 이익을 얻을 수 있기 때문.

Experiments

실험 결과

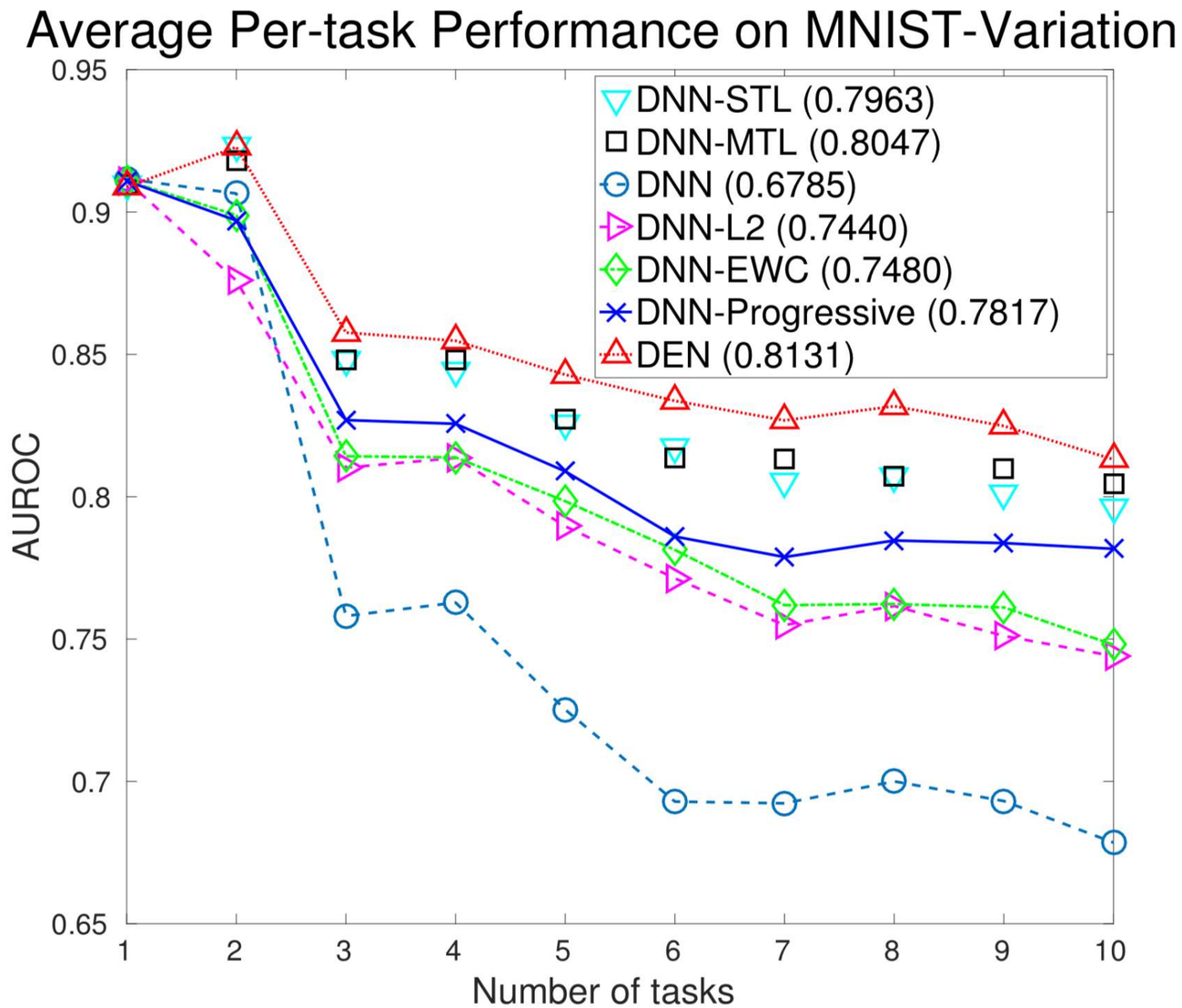
- 많은 실험이 있지만, MNIST-Variation Dataset에 대해서 집중적으로 알아보고자 함.

- MNIST-Variation Dataset이란?
- MNIST 데이터의 image에 random permutation을 준 데이터 셋.
- Lifelong Learning 상황을 위해 만들어진 데이터 셋.
- Knowledge Transfer가 일어날 수 밖에 없는 데이터 셋.

Experiments

실험 결과

- 얼마나 Catastrophic Forgetting이 안일어 났는지를 나타내는 그래프.

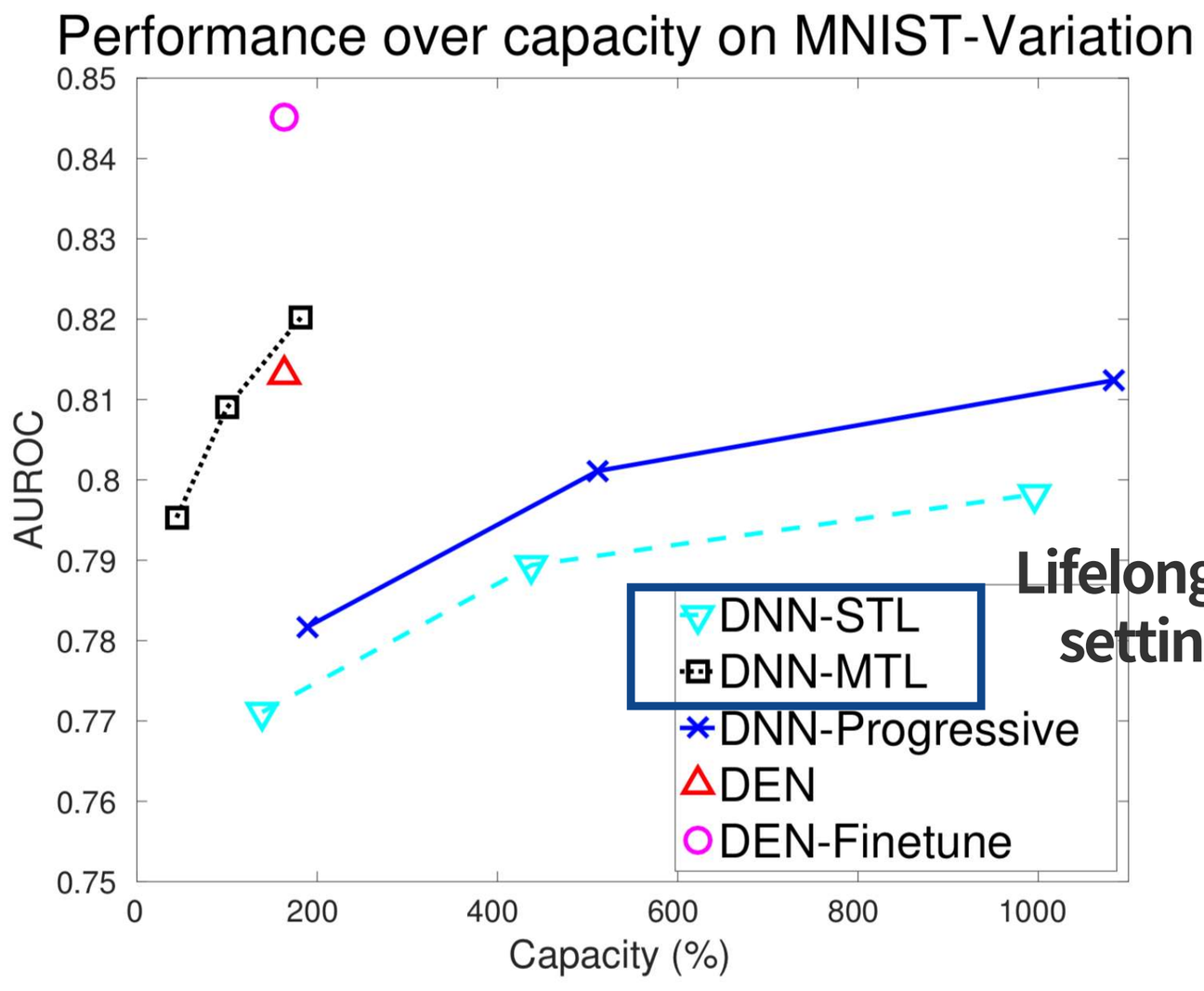


기존의 Lifelong Learning 알고리즘보다 더 좋은 성능을 나타냄.

Experiments

실험 결과

- 얼마나 적은 Capacity를 사용했는지를 나타내는 그래프.

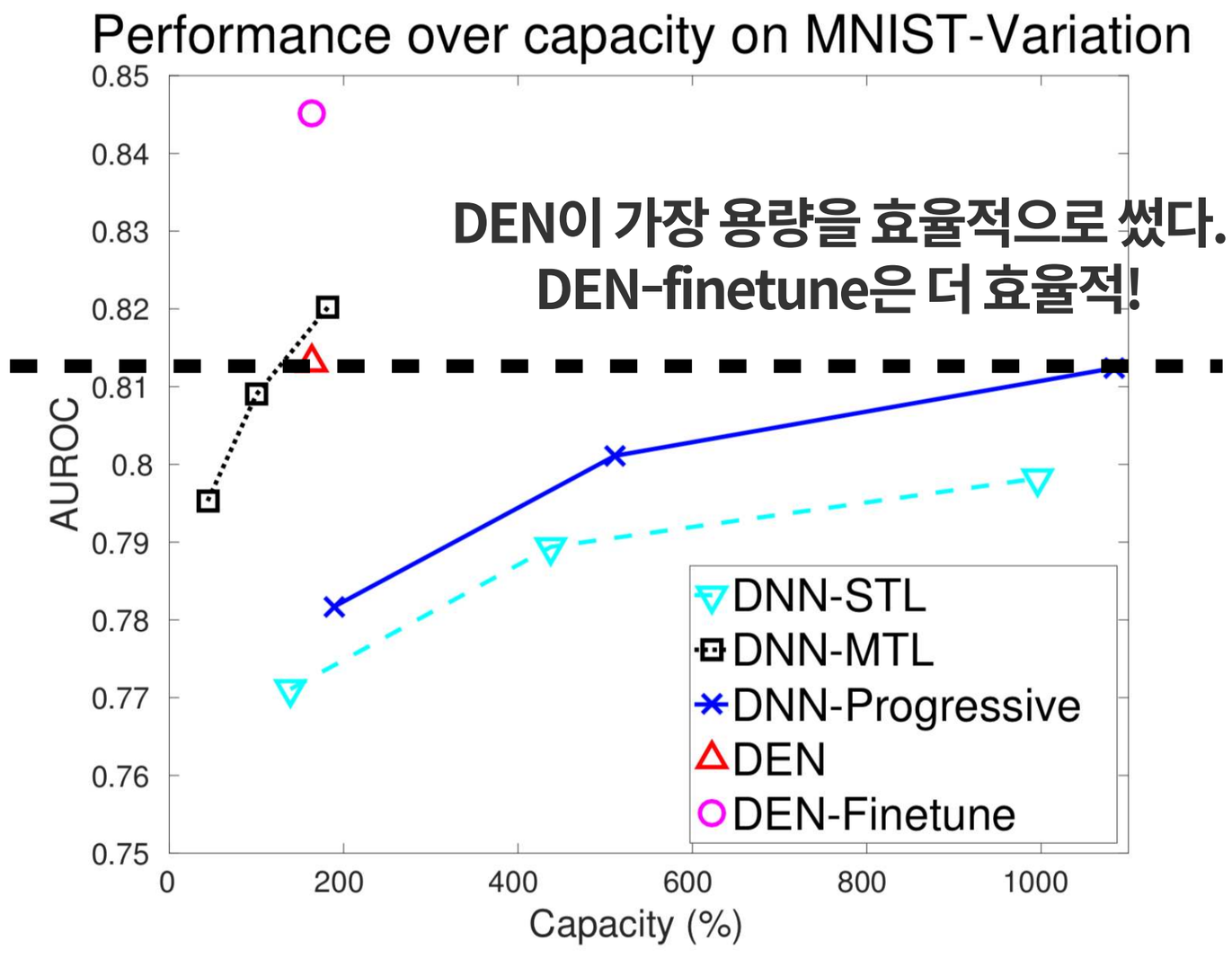


Lifelong Learning setting이 아님.

Experiments

실험 결과

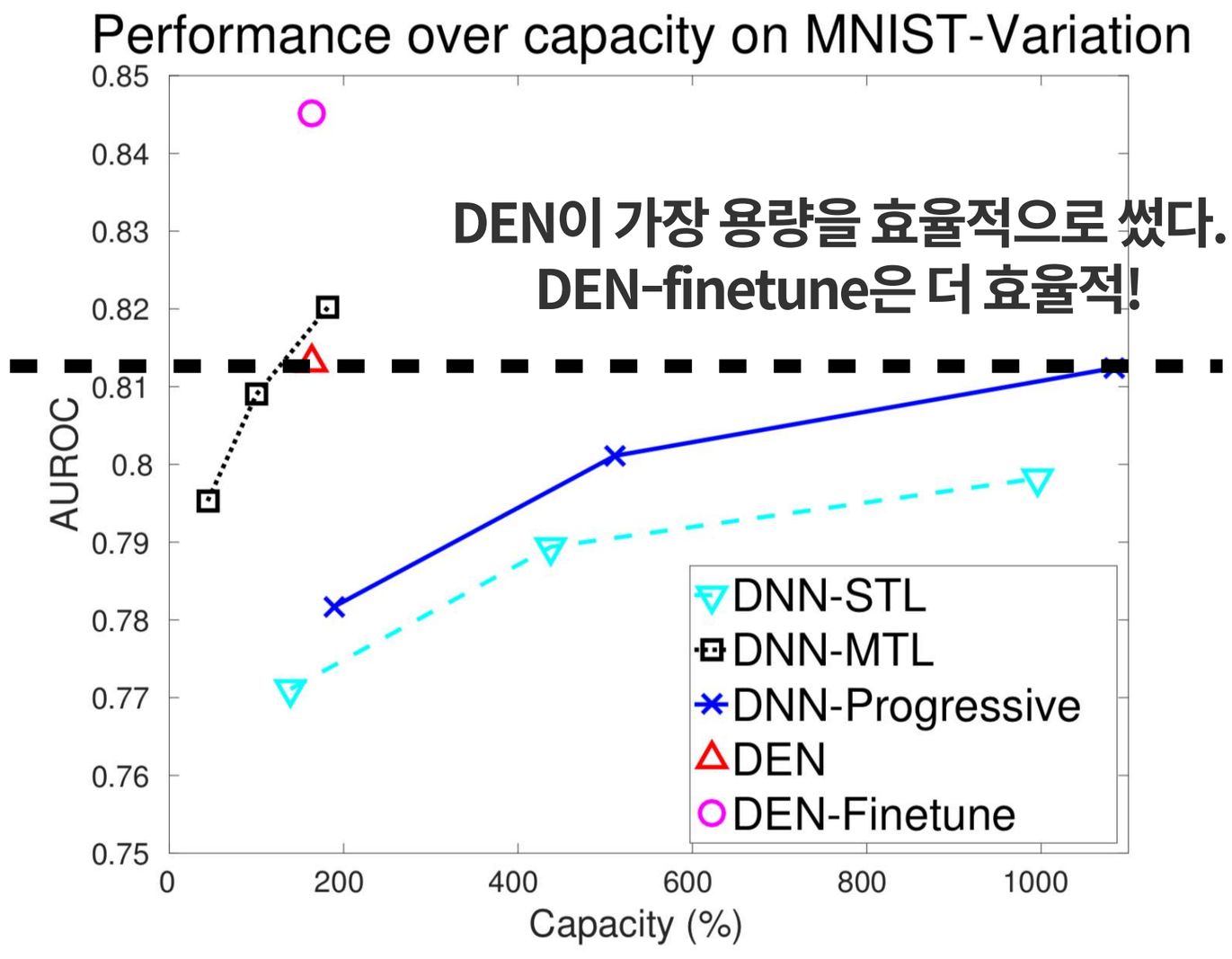
- 얼마나 적은 Capacity를 사용했는지를 나타내는 그래프.



Experiments

실험 결과

- 얼마나 적은 Capacity를 사용했는지를 나타내는 그래프.



기존의 Lifelong Learning 알고리즘들이 사용한 용량의 11.9~60.3% 를 사용함.

Experiments

추가 실험

- DEN 모델에 대해 더 알아보고 싶은 부분이 있어 구현을 해보고자 하였음.
- 다행히, 저자들이 공개한 코드가 존재!
  - https://github.com/jaehong-yoon93/DEN

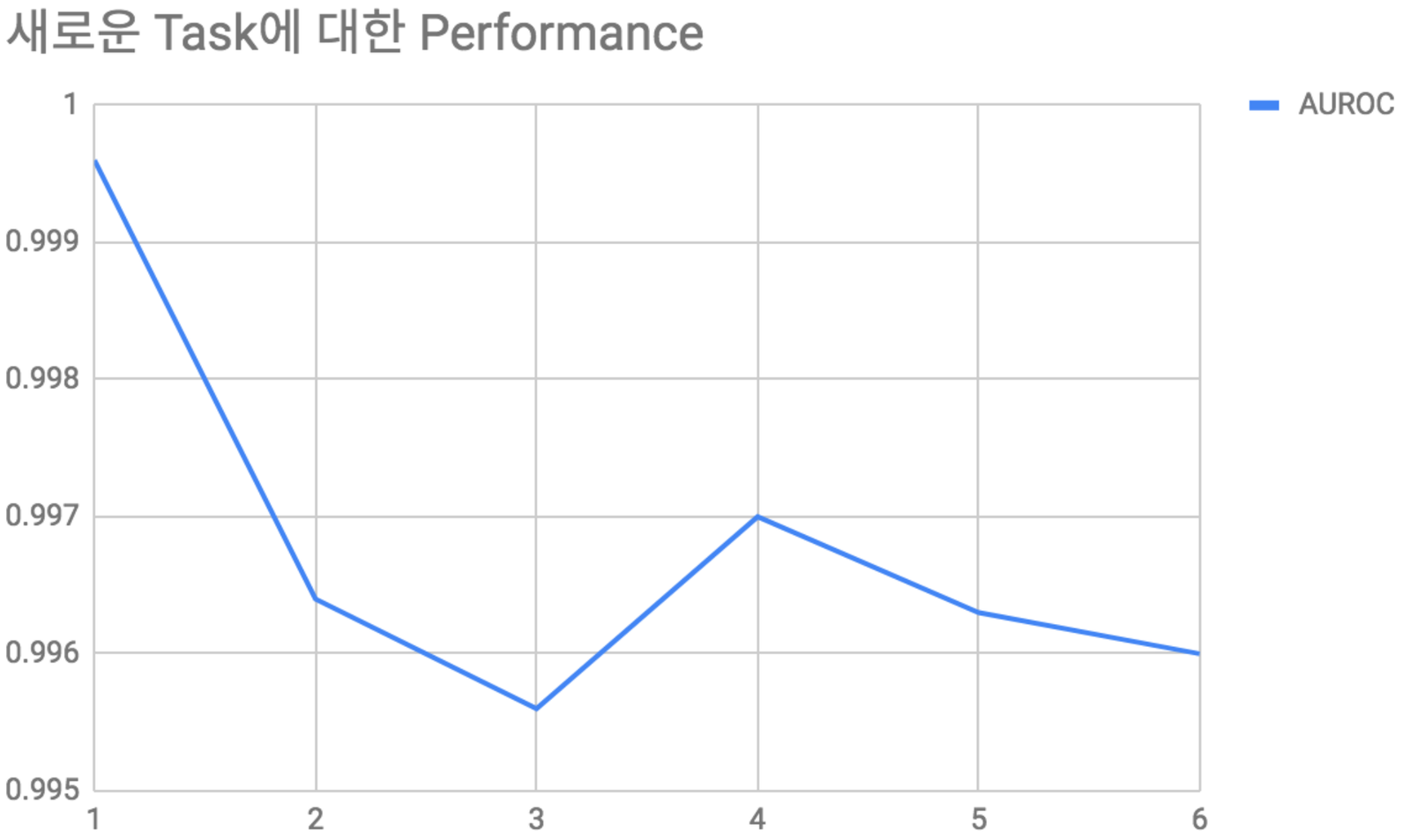
- 하지만, 공개된 코드에 Selective Retraining이 거의 일어나지 않는 버그가 있는 것으로 판단됨.
- 그래서, 고쳐서 실험해봄
  - https://github.com/JoonyoungYi/DEN-tensorflow

- MNIST-Variations 데이터 셋으로 실험한 결과를 리포트 하겠음.

Experiments

실험 결과

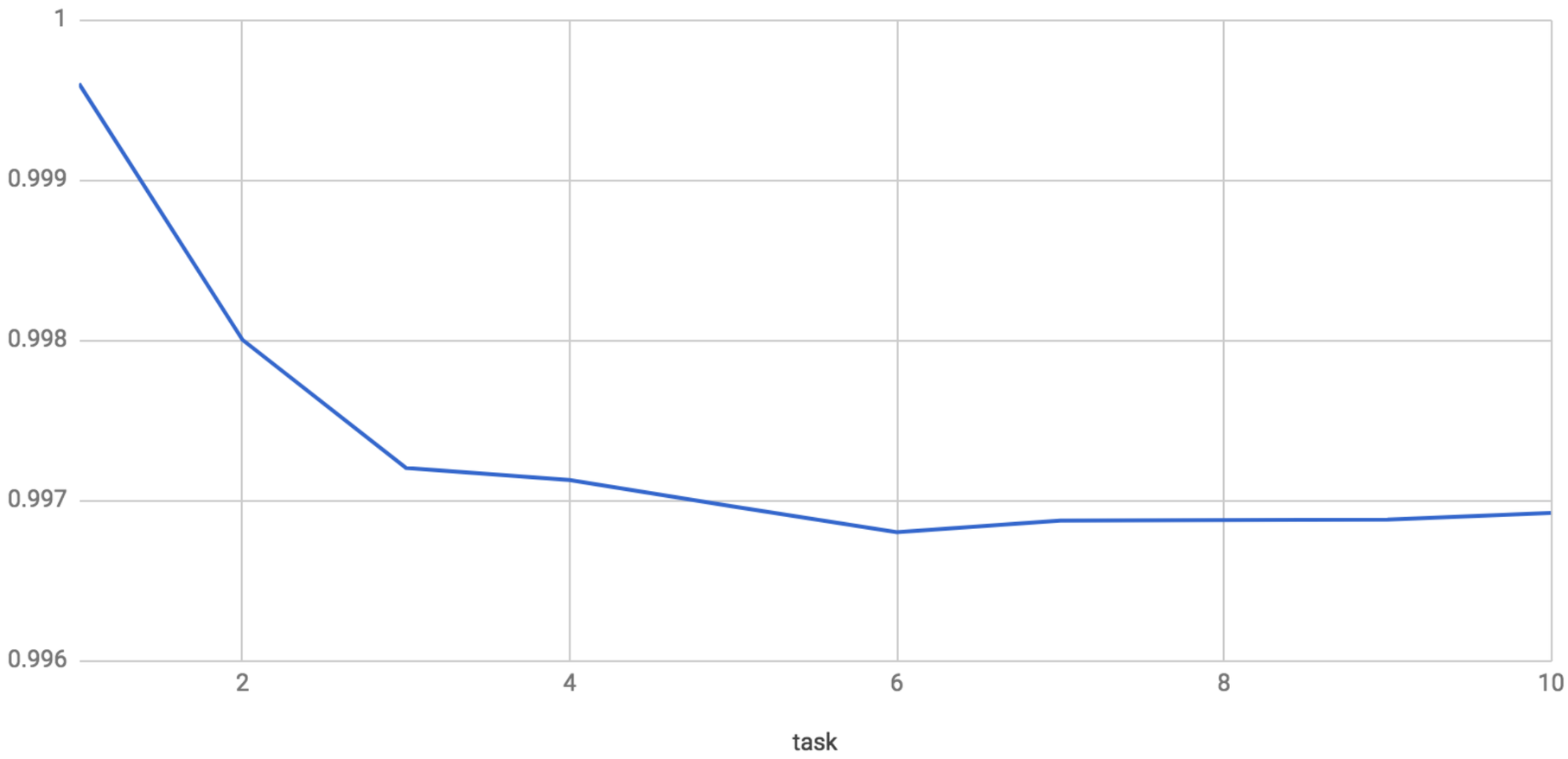
- 모델이 새로운 Task에 대한 지식을 습득하는 데 **방해를 받지 않지 않을까?**
- **방해받지 않는다**는 것을 실험을 통해 알 수 있었음.



Experiments

실험 결과

- 논문에 제시된 결과보다 Catastrophic Forgetting이 덜 일어남.
- 논문에 제시된 실험 결과의 용량과 아마 차이가 있을 것으로 예상.



Experiments

실험 결과

- Backward Knoweldge Transfer가 일어나는 것을 확인할 수 있었음.

Task 진행

각 Task에 대한 성능

DEN-1	1	2	3	4	5	6	7	8	9	10
1	0.9996	0.9996	0.9996	0.9996	0.9996	0.9996	0.9996	0.9996	0.9996	0.9996
2		0.9964	0.9964	0.9963	0.9963	0.9963	0.9963	0.9963	0.9963	0.9963
3			0.9956	0.9956	0.9956	0.9956	0.9957	0.9956	0.9956	0.9956
4				0.997	0.997	0.997	0.997	0.997	0.997	0.997
5					0.9963	0.9963	0.9963	0.9963	0.9963	0.9963
6						0.996	0.996	0.996	0.996	0.996
7							0.9972	0.9972	0.9972	0.9972
8								0.997	0.997	0.997
9									0.9969	0.9969
10										0.9973

종종 Backward Knowledge Transfer가 일어나는 것을 확인.
(이후 Task를 학습하면 오히려 성능이 증가하는 것을 종종 확인 가능)
MNIST-Variations Dataset 특성 상, Backward Knowledge Transfer가
많이 일어나기는 힘들 것!

Conclusion and Discussion

결론

- Catastrophic Forgetting이 **다른 모델보다 덜 했고,**
- 모델이 새로운 Task에 대한 지식을 습득하는 데 **방해를 받지 않지 않았고,**
- Forward Knowledge Transfer가 **일어나며,**
- Backward Knowledge Transfer도 **일어나며,**
- 메모리를 기존 방법 대비 11.9 ~60.3%만 사용해 **효율적인** 평생 학습 방법.

Conclusion and Discussion

한계

- 하이퍼 파라미터가 너무 많아서 일일이 손으로 정해줘야 한다.

- 그래서, 한계를 극복하기 위한 다른 연구가 등장!
- Overcoming Catastrophic Forgetting with Hard Attention to the Task.
- Lifelong Learning에서 Hard Attention을 통해 Catastrophic Forgetting을 해결.

NAME | 이준영(Joonyoung Yi)
EMAIL | joonyoung.yi@mli.kaist.ac.kr joonyoung.yi@kaist.ac.kr
PHONE | +82-10-9765-0885

Appendix

Reference

[1] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. ”Overcoming catastrophic forgetting in neural networks”, Proceedings of the National Academy of Sciences (PNAS) 2017, pp. 201611835.

[3] Eric Eaton and Paul L. Ruvolo. ELLA: An efficient lifelong learning algorithm. In Sanjoy Dasgupta and David Mcallester (eds.), ICML, volume 28, pp. 507–515. JMLR Workshop and Conference Pro- ceedings, 2013.

[5] Corinna Cortes, Xavi Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. Adanet: Adaptive structural learning of artificial neural networks. arXiv preprint arXiv:1607.01097, 2016.

[6] Abhishek Kumar and Hal Daume III. Learning task grouping and overlap in multi-task learning. In ICML, 2012.

[7] Sang-Woo Lee, Jin-Hwa Kim, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic for- getting by incremental moment matching. arXiv preprint arXiv:1703.08475, 2017

[8] George Philipp and Jaime G. Carbonell. Nonparametric neural networks. In ICLR, 2017.

[9] Andrei Rusu, Neil Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. arXiv preprint arXiv:1606.04671, 2016.

[12] Tianjun Xiao, Jiaxing Zhang, Kuiyuan Yang, Yuxin Peng, and Zheng Zhang. Error-driven incremen- tal learning in deep convolutional neural network for large-scale image classification. In Proceedings of the 22nd ACM international conference on Multimedia, pp. 177–186. ACM, 2014.

[13] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In ICML, pp. 3987–3995, 2017.

[14] Guanyu Zhou, Kihyuk Sohn, and Honglak Lee. Online incremental feature learning with denoising autoencoders. In International Conference on Artificial Intelligence and Statistics, pp. 1453–1461, 2012.