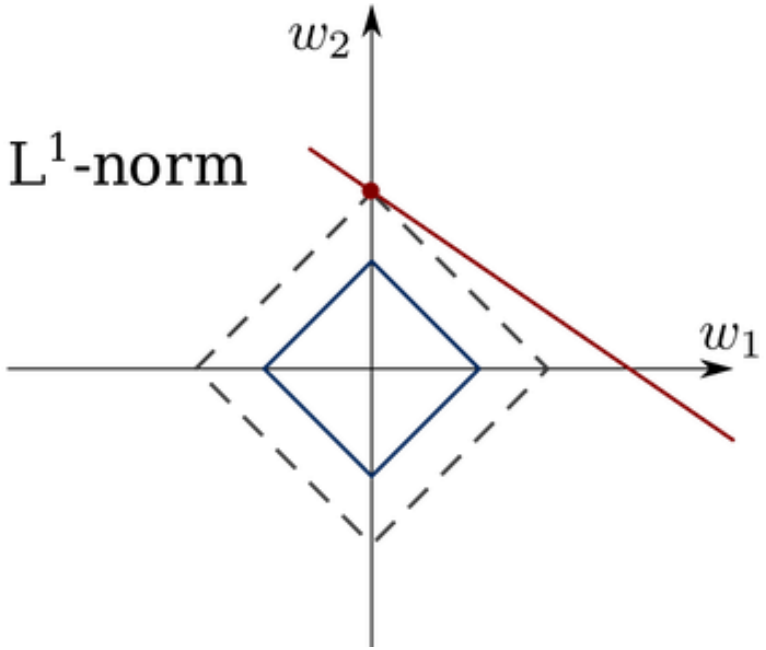
LIFELONG LEARNING WITH DYNAMICALLY EXPANDABLE NETWORKS

데이터사이언스학과 빅데이터 관리 및 응용 연구실
석사과정 김민선

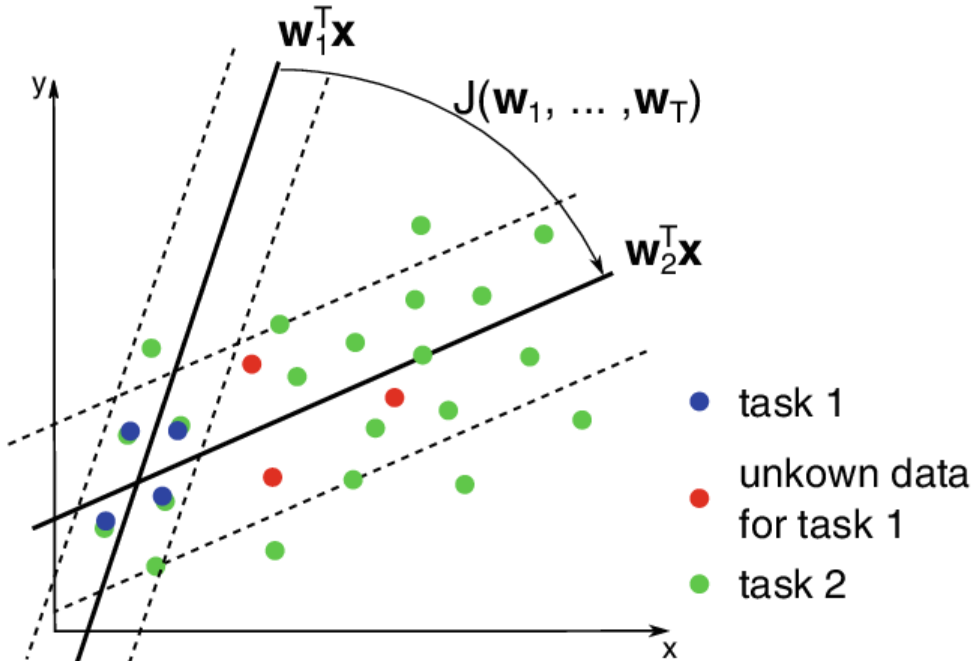
2023-03-10

Preliminaries

- L1-Regularizer
  : The lasso gives a sparse solution in which $w_1 = 0$.

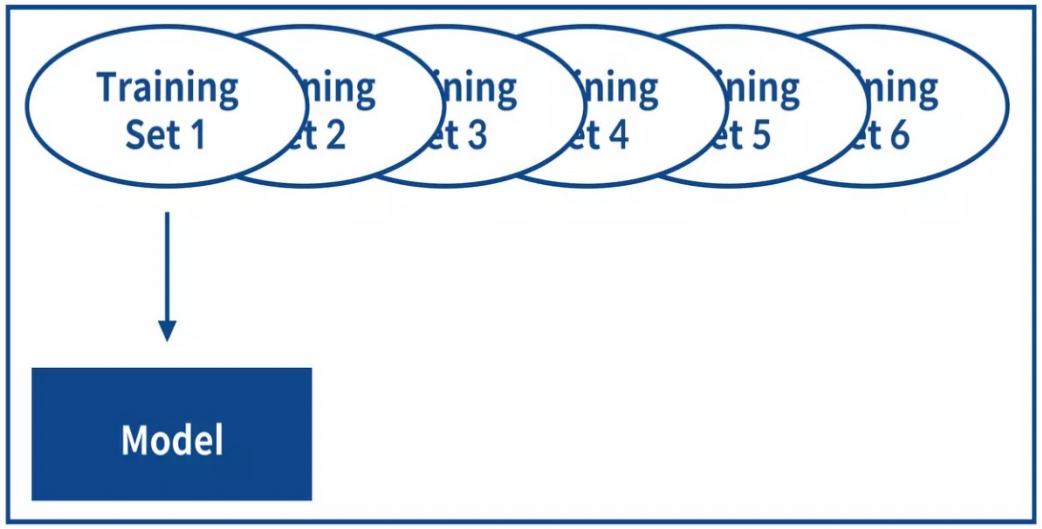


- Multi-task Learning
  : Knowledge transfer is the objective of multi-task learning



Preliminaries

- Lifelong Learning
  : Task t에 해당하는 Training Set t가 차례로 주어진다.



- Knowledge Transfer
  - Forward: 이전 task를 미리 학습함으로써, 이후 task의 학습을 돕는 상황
  - Backward: 이후 task를 미리 학습함으로써, 이전 task의 성능이 증가하는 상황

Related Works

- Dynamically Expandable Networks
  1) Zhou et al.
  2) Philipp et al.
  3) Cortes et al.|

  → Multi-task Learning에 대한 고려x, 뉴런을 반복적으로 추가하는 과정

  Xiao et al. 새로운 분류가 모델에 주어질 때, 계층 구조를 형성하는 신경망을 점진적으로 학습하는 방법을 제안. 다만, 모든 층에서 뉴런을 증가시킬 수 없고 모델의 분기가 많아진다는 단점 존재.

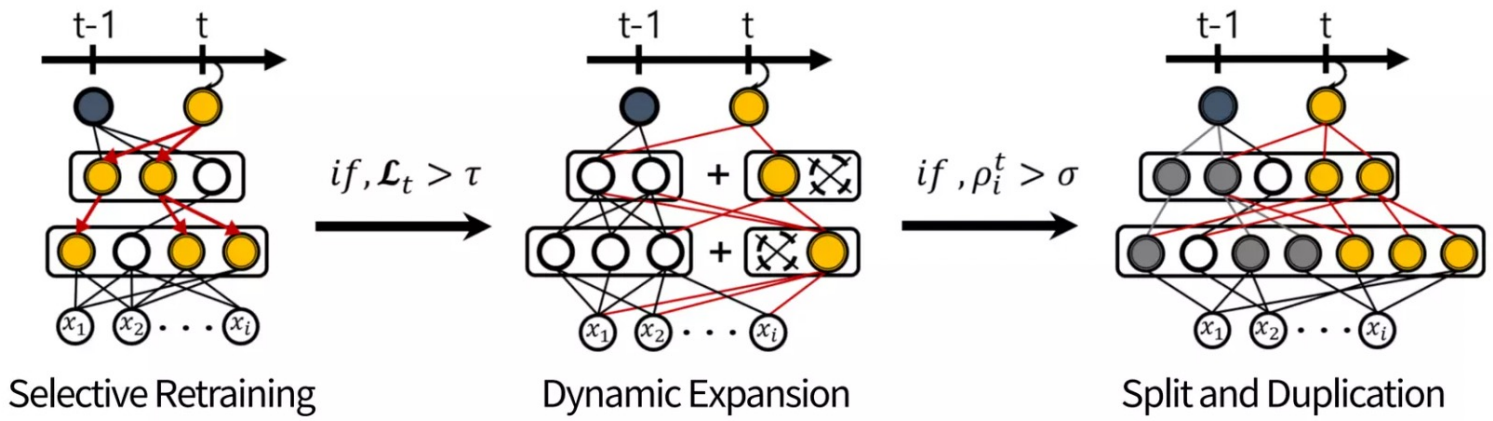
Contribution

- Prevent Catastrophic Forgetting
- Memory Efficient
- Forward & Backward Knowledge Transfer
- Learn the Knowledge of New Tasks

Algorithm

Algorithm 1 Incremental Learning of a Dynamically Expandable Network

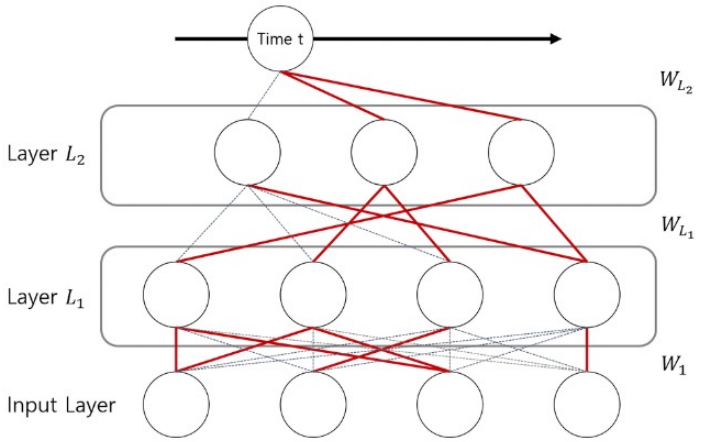
Input: Dataset $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_T)$, Thresholds τ, σ
Output: $\boldsymbol{W}^T$
for $t = 1, \ldots, T$ **do**
  if $t = 1$ **then**
    Train the network weights $\boldsymbol{W}^1$ using Eq. 2
  else
    $\boldsymbol{W}^t = SelectiveRetraining(\boldsymbol{W}^{t-1})$ {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 }



Algorithm

if $t = 1$ **then**
Train the network weights $\boldsymbol{W}^1$ using Eq. 2

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

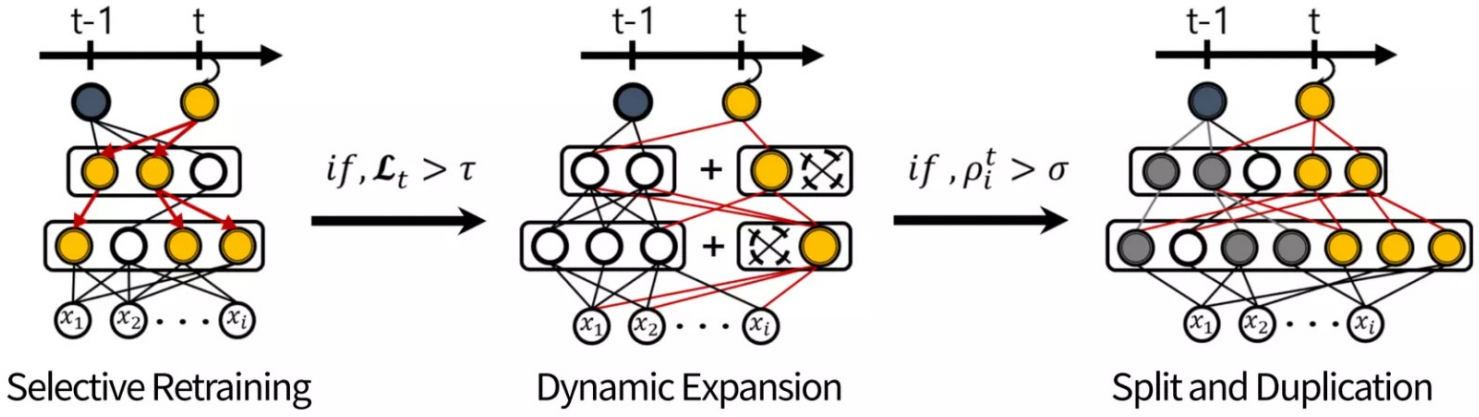


$$\underset{\boldsymbol{W}^{t=1}}{\text{minimize}}\, \mathcal{L}(\boldsymbol{W}^{t=1}; \mathcal{D}_t) + \mu \sum_{l=1}^{L} \|\boldsymbol{W}_l^{t=1}\|_1$$

Algorithm

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

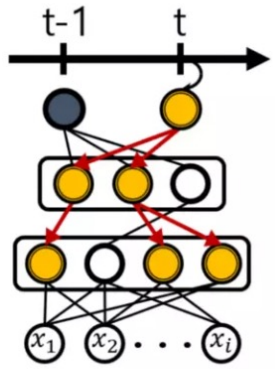
else
$\boldsymbol{W}^t = SelectiveRetraining(\boldsymbol{W}^{t-1})$ {Selectively retrain the previous network using Algorithm 2 }
if $\mathcal{L}_t > \tau$ **then**
$\boldsymbol{W}^t = DynamicExpansion(\boldsymbol{W}^t)$ {Expand the network capacity using Algorithm 3}
$\boldsymbol{W}^t = Split(\boldsymbol{W}^t)$ {Split and duplicate the units using Algorithm 4 }



Selective Retraining

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

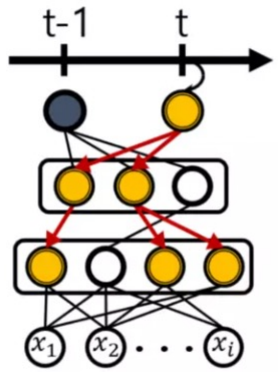
첫번째 단계인 Selective Retraining은 말 그대로 선별적으로 재학습 하는 단계!
새로운 Task를 학습시키기 위해, 주요한 Weight만을 업데이트하는 과정.
모든 Weight를 학습시키지 않고 몇 개의 edge만을 선별적으로 업데이트 한다.

Selective Retraining

Solve Eq. 3 to obtain $\boldsymbol{W}_{L,t}^t$

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

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)

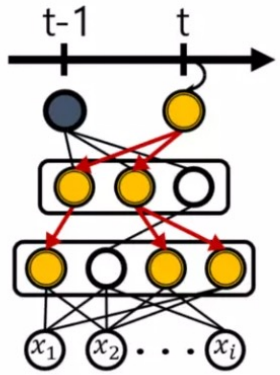


Selective Retraining

Selective Retraining

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

Add neuron i to S if the weight between i and o_t in $\boldsymbol{W}_{L,t}^t$ 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 $\boldsymbol{W}_{l,ij}^{t-1} \neq 0$.



Selective Retraining

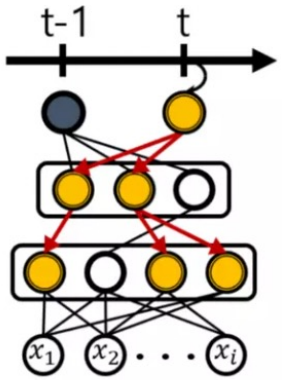
Selective Retraining

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

Solve Eq. 4 to obtain $\boldsymbol{W}_S^t$

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.



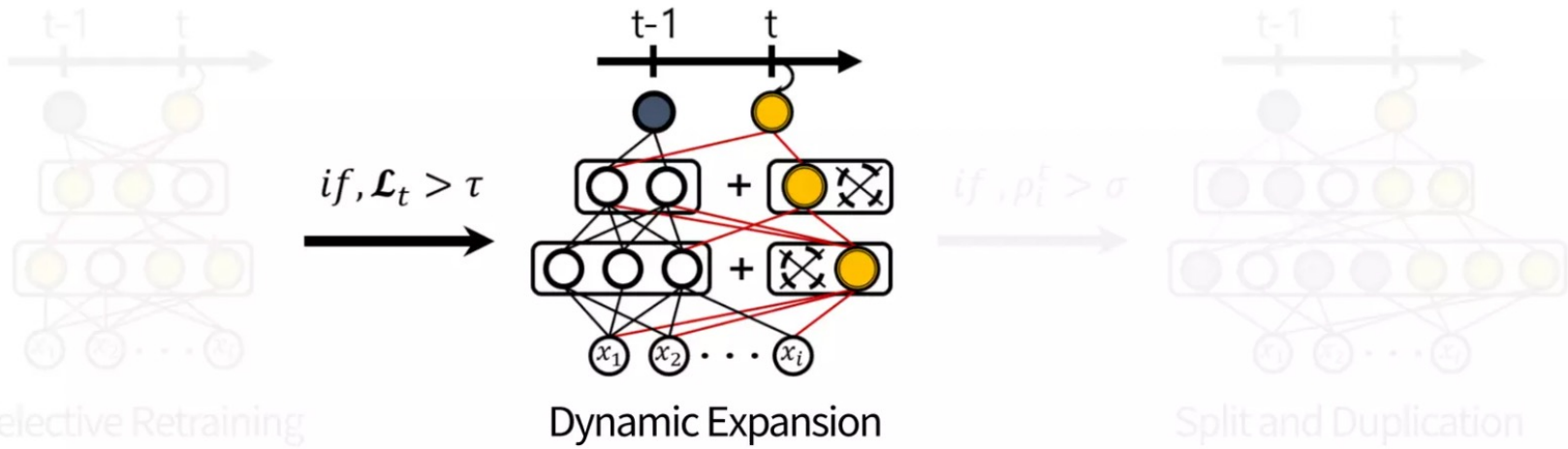
Selective Retraining

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 $\boldsymbol{h}_l^{\mathcal{N}}$

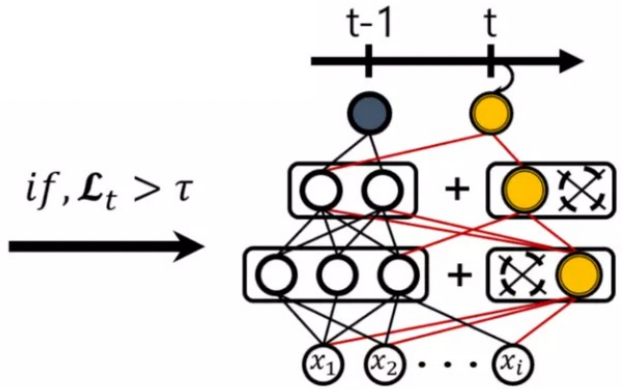
두 번째 단계인 Dynamic Expansion은
만약, Selective Retraining으로 학습을 시켰는데,
새 Task에 대한 학습이 제대로 일어나지 않았다면
(=새 Task에 대한 loss가 너무 크다면),
네트워크 내에 hidden unit을 추가하는 과정이다.



Dynamically Expansion

Perform Algorithm 2 and compute $\mathcal{L}$
if $\mathcal{L} > \tau$ **then**
  Add k units $\boldsymbol{h}^{\mathcal{N}}$ at all layers

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



Dynamic Expansion

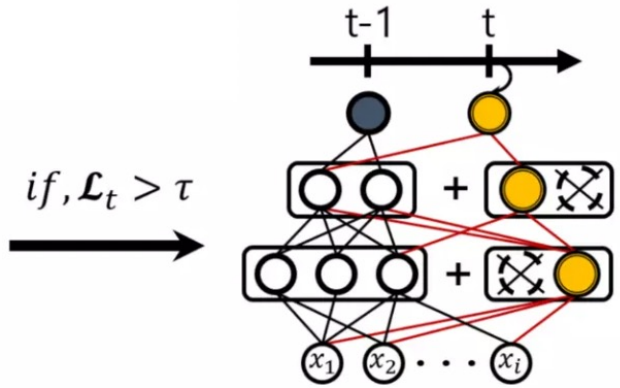
Dynamically Expansion

Group Sparsity Regularizer
-> find useless units!

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

Solve for Eq. 5 at all layers

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



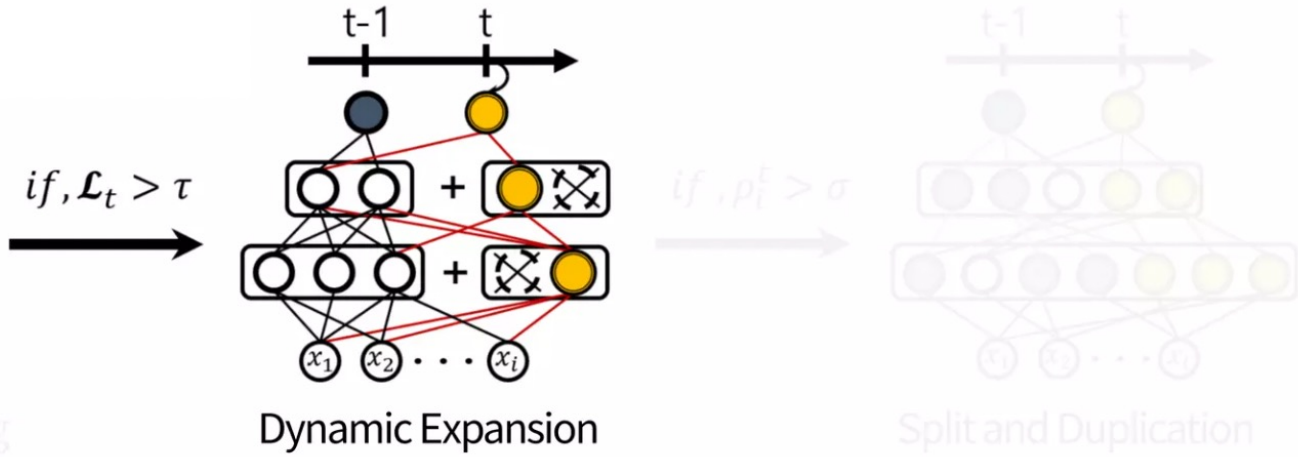
Dynamic Expansion

Dynamically Expansion

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

for $l = L-1, \ldots, 1$ **do**
  Remove useless units in $\boldsymbol{h}_l^{\mathcal{N}}$

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



Split And Duplication

Algorithm 4 Network Split/Duplication

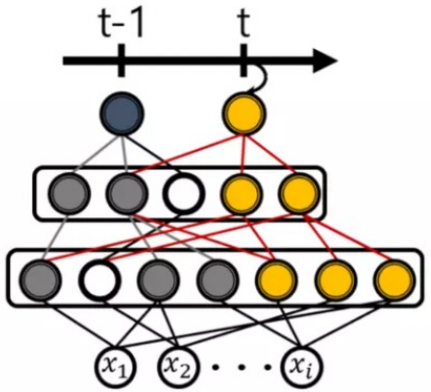
Input: Weight W^{t-1}, Threshold σ
Perform Eq. 6 to obtain $\widetilde{W}^t$
for all hidden unit i **do**
  $\rho_i^t = \|w_i^t - w_i^{t-1}\|_2$
  if $\rho_i^t > \sigma$ **then**
    Copy i into i' (w' introduction of edges for i')
Perform Eq. 6 with the initialization of $\widetilde{W}^t$ to obtain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

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

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!

$if\ , \rho_i^t > \sigma$

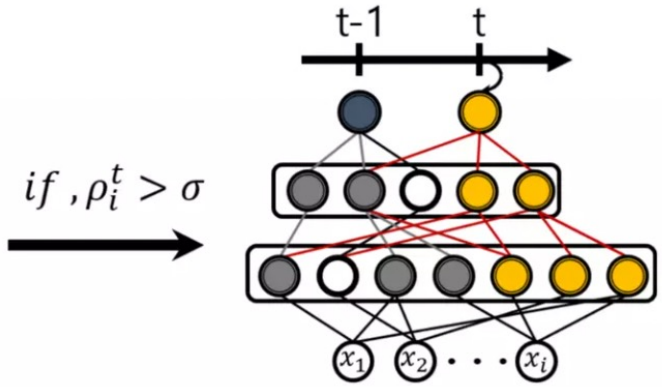


Split and Duplication

Split And Duplication

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')

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



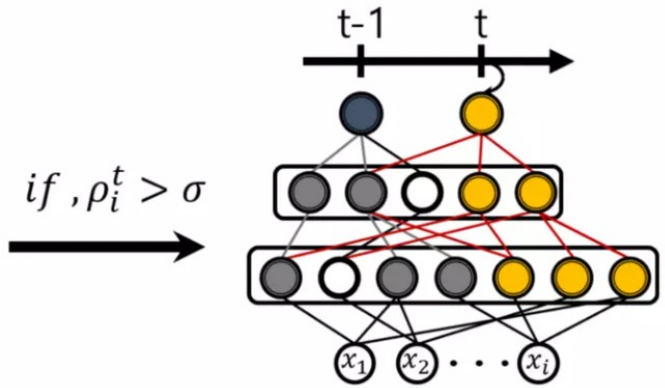
Split and Duplication

Split And Duplication

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

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

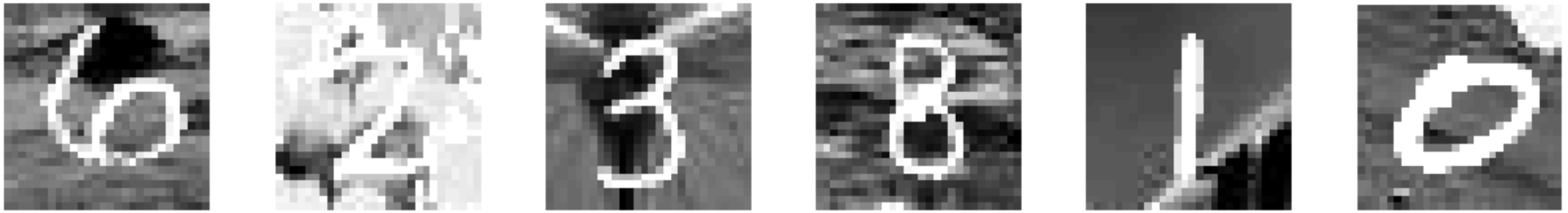
복제를 진행한 이후에는 복제된 상태를 기준으로,
Fine tuning을 진행한다.



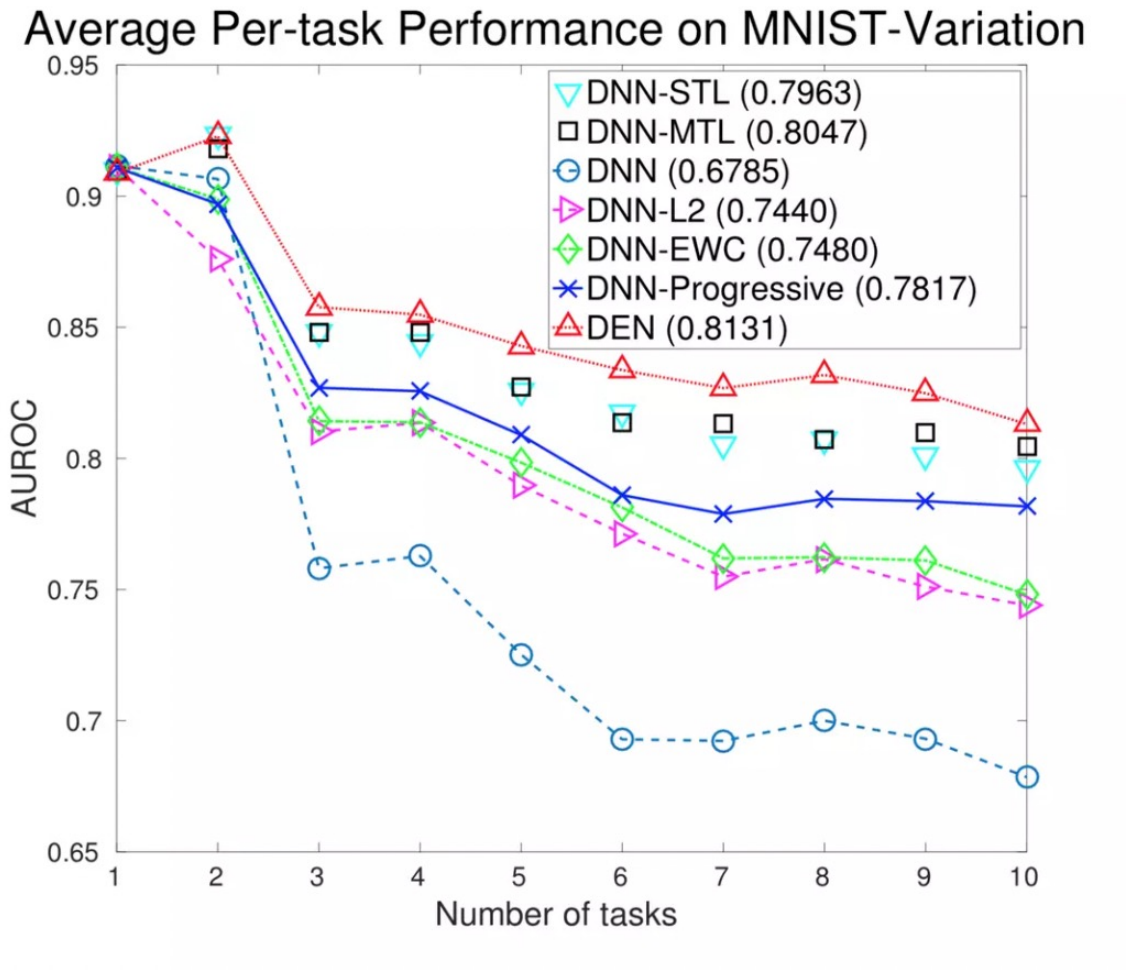
Split and Duplication

Experiment

- MNIST-Variation Dataset
  - Suitable for Lifelong Learning
  - Inevitable for Knowledge Transfer

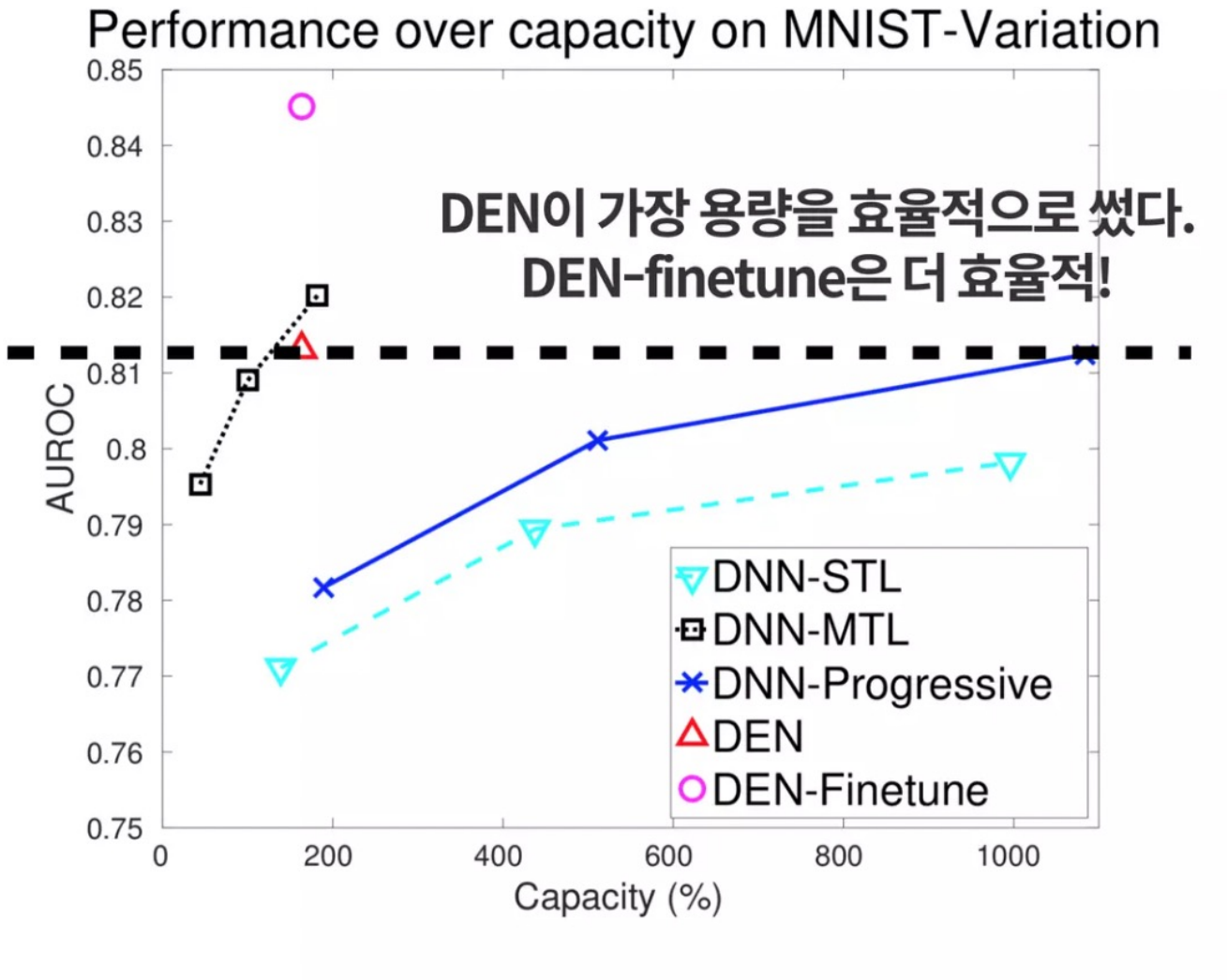


Experiment



기존의 Lifelong Learning 알고리즘보다 더 좋은 성능

Experiment



11.9% ~ 60.3% less capacity
compared to previous lifelong learning

Conclusion

- Conduct each task with limited number of units
- Prevent catastrohpic forgetting with introducing hidden units
- Highly probable backward transfer

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.