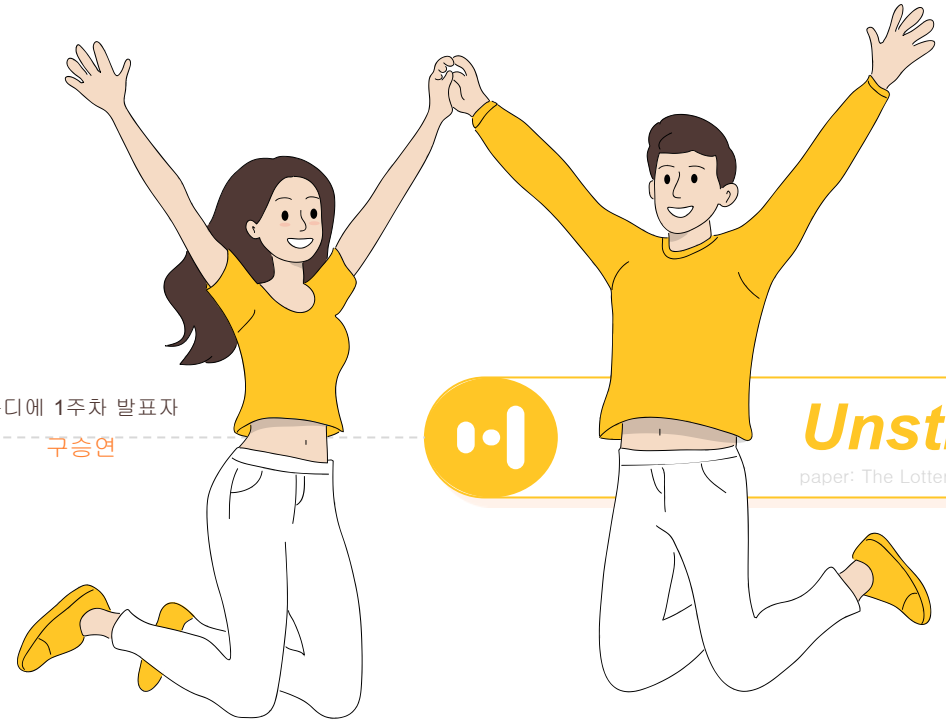


온디에 1주차 발표자

구승연

Unstructured Pruning & LTH

paper: The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks

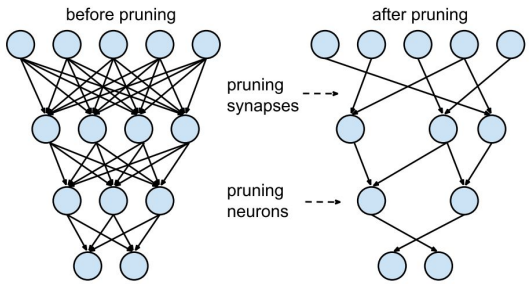
목차

Pruning 이란?

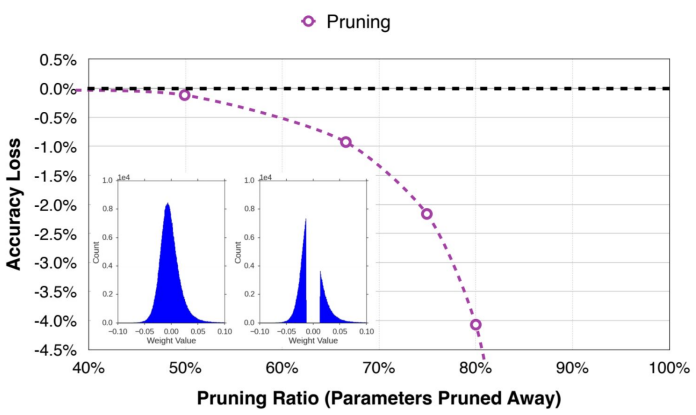
- Pruning 이란?

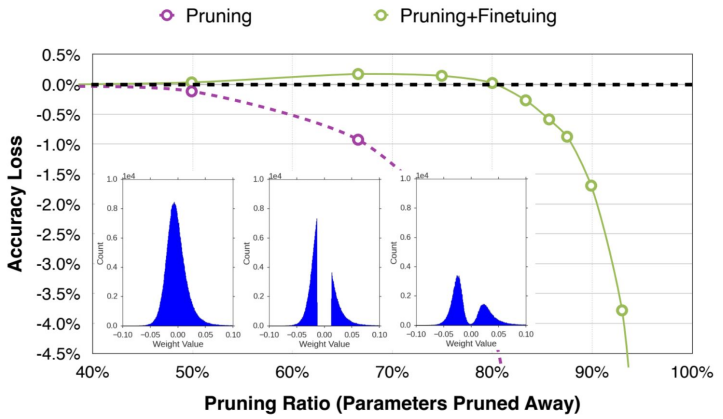
  ***중요도**가 낮은 **weight(parameter)** 을 찾아서 없애자!*

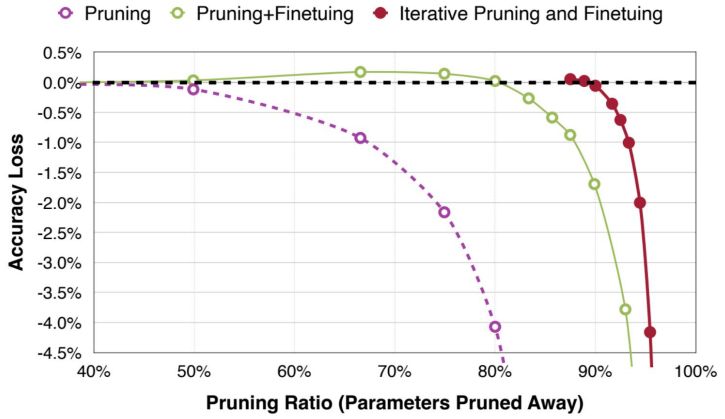
  가장 쉽게 생각해 볼 수 있는 것이, 값이 작은 가중치를 없애는 것(magnitude pruning)



- Pruning의 시각화 및 효과







**MIT Han Lab. (2024, fall) TinyML and Efficient Deep Learning Computing*

Pruning 이란?

- Pruning 이란?
  중요도가 낮은 **weight(parameter)** 을 찾아서 없애자!

  가장 쉽게 생각해 볼 수 있는 것이, 값이 작은 가중치를 없애는 것(magnitude pruning)

- 수식적 표현

Data(D)에 대해
objective function이 가장 작아지도록 W를 선택

$$\min_W \mathcal{L}(W;\mathcal{D}) = \min_W \frac{1}{N}\sum_{i=1}^{N} \ell(W;(\mathbf{x}_i,\mathbf{y}_i)),$$

$$\text{s.t.} \|W\|_0 \leq k.$$

non-zero 파라미터의 개수

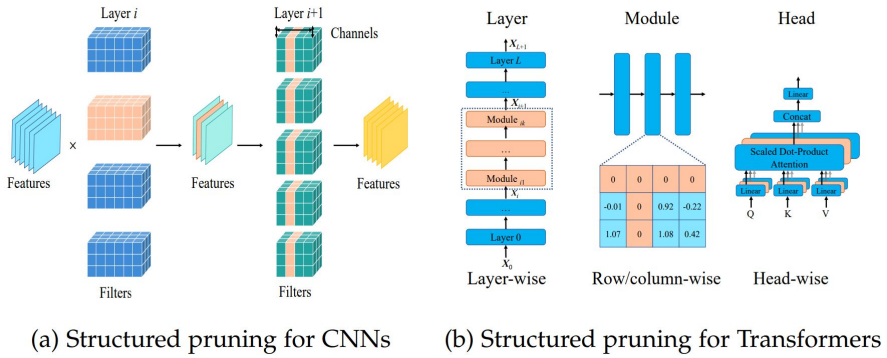
목표하는 파라미터의 수(k < K)
K : 전체 파라미터 수

- Pruning으로 기대할 수 있는 효과
  파라미터의 수가 감소함으로써,
  메모리 사용량 ⬇️, **속도** ⬆️, **에너지** ⬇️

*Cheng, Hongrong, Miao Zhang, and Javen Qinfeng Shi. "A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations." IEEE Transactions on Pattern Analysis and Machine Intelligence (2024).

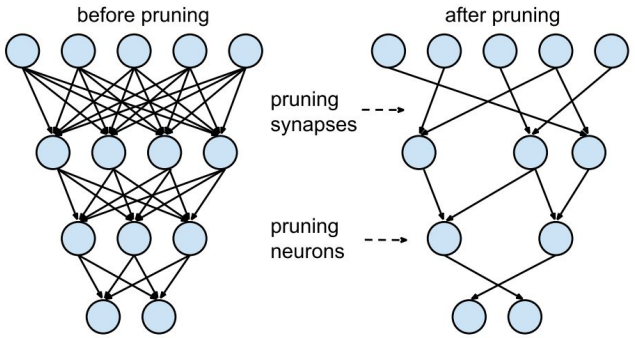
Unstructured Pruning이란?

- Structured Pruning과 Unstructured Pruning



(a) Structured pruning for CNNs (b) Structured pruning for Transformers

별다른 제약 없이 모든 weight을 개별적으로 pruning 할 수 있으면 -> unstructured



pruning 하는 단위가 단일 neuron인지, 더 큰 단위인지에 따라 달라짐.

Unstructured Pruning은
- non-structured pruning
- weight-wise pruning
- fine-grained pruning

으로 불리기도 하고, 서베이 논문에서는 **specific acceleration pruning** 으로 구분한다.

*Cheng, Hongrong, Miao Zhang, and Javen Qinfeng Shi. "A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations." IEEE Transactions on Pattern Analysis and Machine Intelligence (2024).

Unstructured Pruning이란?

- (Unstructured) Magnitude Pruning의 과정 – once

$$Importance = |W|$$

중요도는 파라미터 절댓값(Magnitude)

$$v_{\text{thr}} = \texttt{kthvalue}(Importance, \#W \cdot s)$$
$$M = Importance > v_{\text{thr}}$$
$$W = W \cdot M$$

중요도(Magnitude)를 기준으로
마스크(M)을 만들고
파라미터 (W)에 적용

```
def fine_grained_prune(tensor: torch.Tensor, sparsity : float) -> torch.Tensor:
    # Step 1: calculate the #zeros (please use round())
    num_zeros = round(sparsity * num_elements)

    # Step 2: calculate the importance of weight
    importance = torch.abs(tensor)

    # Step 3: calculate the pruning threshold
    threshold = torch.kthvalue(importance.view(-1), num_zeros).values

    # Step 4: get binary mask (1 for nonzeros, 0 for zeros)
    mask = torch.gt(importance, threshold).float()
```

sparsity와 현재 입력된 파라미터의 수를 기반으로 0의 개수 계산

각 파라미터의 중요도 계산

threshold 만들기 (k번째 작은 값을 찾는 함수)

threshold 이하인 위치는 0인 mask 만들기

*https://colab.research.google.com/drive/1Fagq3JQBzCizodyxpHKvWDzfCC7F1RWN

Unstructured Pruning이란?

- (Unstructured) Magnitude Pruning의 과정 – fine-tuning

```python
class FineGrainedPruner:
    def __init__(self, model, sparsity_dict):
        self.masks = FineGrainedPruner.prune(model, sparsity_dict)

    @torch.no_grad()
    def apply(self, model):
        for name, param in model.named_parameters():
            if name in self.masks:
                param *= self.masks[name]

    @staticmethod
    @torch.no_grad()
    def prune(model, sparsity_dict):
        masks = dict()
        for name, param in model.named_parameters():
            if param.dim() > 1: # we only prune conv and fc weights
                masks[name] = fine_grained_prune(param, sparsity_dict[name])
        return masks
```

mask 생성

mask 적용

mask 생성

```python
num_finetune_epochs = 5
optimizer = torch.optim.SGD(model.parameters(), lr=0.01, momentum=0.9, weight_decay=1e-4)
scheduler = torch.optim.lr_scheduler.CosineAnnealingLR(optimizer, num_finetune_epochs)
criterion = nn.CrossEntropyLoss()

best_sparse_model_checkpoint = dict()
best_accuracy = 0
print(f'Finetuning Fine-grained Pruned Sparse Model')
for epoch in range(num_finetune_epochs):
    # At the end of each train iteration, we have to apply the pruning mask
    #    to keep the model sparse during the training
    train(model, dataloader['train'], criterion, optimizer, scheduler,
          callbacks=[lambda: pruner.apply(model)])
    accuracy = evaluate(model, dataloader['test'])
    is_best = accuracy > best_accuracy
    if is_best:
        best_sparse_model_checkpoint['state_dict'] = copy.deepcopy(model.state_dict())
        best_accuracy = accuracy
    print(f'    Epoch {epoch+1} Accuracy {accuracy:.2f}% / Best Accuracy: {best_accuracy:.2f}%')
```

매 iter마다 mask를 적용한다. (fixed mask)

*https://colab.research.google.com/drive/1Fagq3JQBzCizodyxpHKvWDzfCC7F1RWN

The Lottery Ticket Hypothesis (LTH)

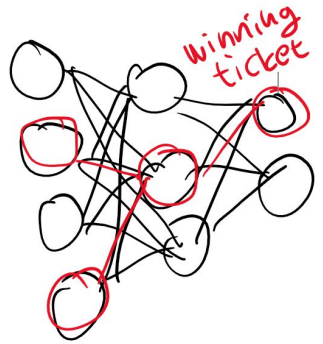
- **LTH 이전에 겪었던 문제**
  - *앞서 pruning이 inference에 효과적임을 언급한 바 있음 >>>*

> - **Pruning으로 기대할 수 있는 효과**
>   *파라미터의 수가 감소함으로써,*
>   *메모리 사용량⬇, 속도⬆, 에너지⬇*

  - ***inference 말고 training할 때도 파라미터 수가 적으면 더 좋지 않을까?***
    However, if a network can be reduced in size, why do we not train this smaller architecture instead in the interest of making training more efficient as well?
  - ***어차피 pruning 한 다음에 다시 학습을 시켜야 하는데, pruning을 먼저 할 수는 없을까?***
  - *하지만, 경험적으로 가지치기가 된 상태에서 학습을 하기 어려움*
    the sparse architectures produced by pruning are difficult to train from the start

이 connection과
이 initialization의 조합이 WINNING TICKET!

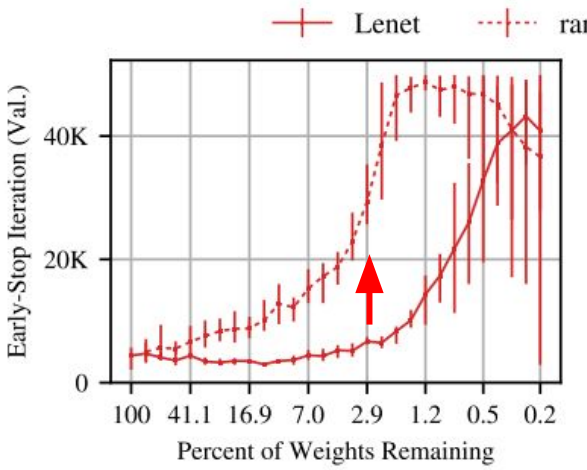
- **LTH가 밝혀낸 것**
  - *가지치기를 한 상태에서 학습을 하기 위해 파라미터를 초기화 할 때,*
    가지치기를 하기 전의 초기값으로 초기화를 하면 학습이 성공적이다!
    (the initialization lottery!!)



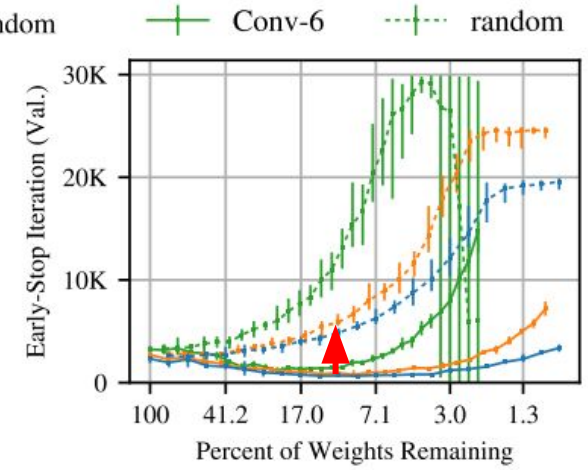
*Frankle, J., & Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.
*https://velog.io/@bismute/The-Lottery-Ticket-Hypothesis%EC%99%80-%EA%B7%B8-%ED%9B%84%EC%86%8D-%EC%97%B0%EA%B5%AC%EB%93%A4-%EB%A6%AC%EB%B7%B0#%EB%85%BC%EB%AC%B8%EC%9D%98-%EB%82%B4%EC%9A%A9

The Lottery Ticket Hypothesis (LTH)

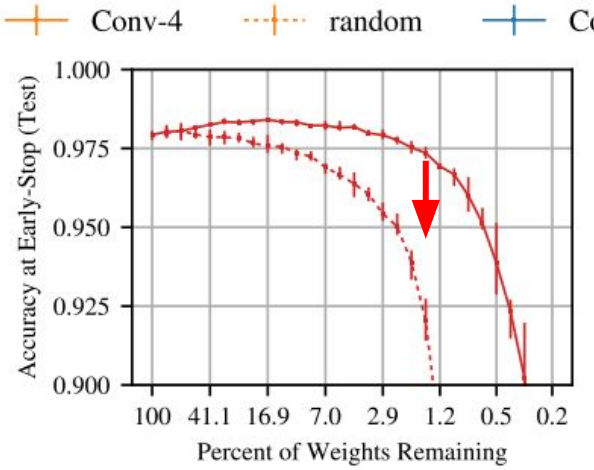
- 경험적 근거



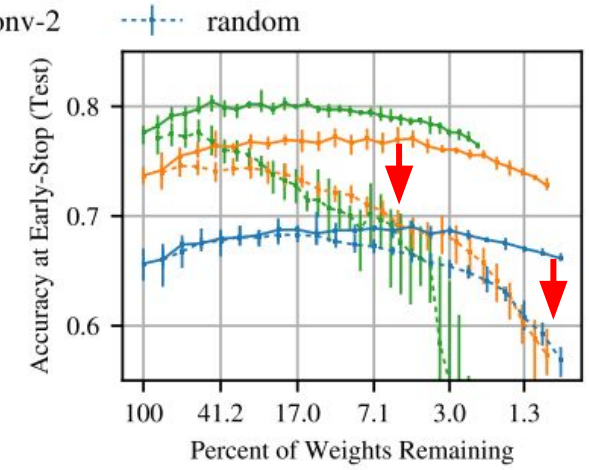
학습 느림



학습 느림



정확도 낮음



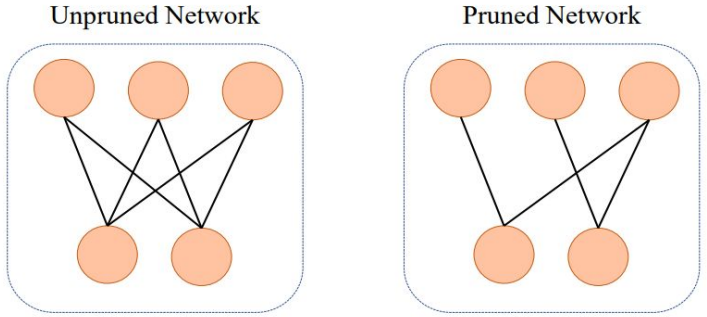
정확도 낮음

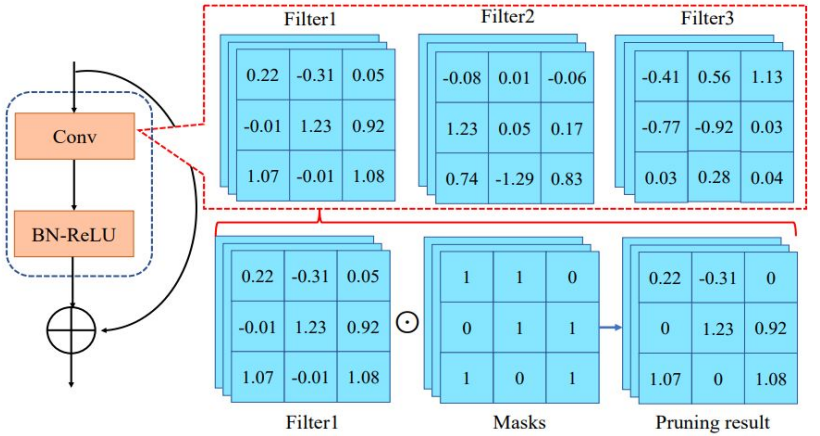
- 이 연구의 한계
  - *Motivation이 "학습을 pruning된 상태로 하자"인데, LTH를 적용하려면 fully-connect 상태로 학습을 한 번 꼭 해야함*
    - ***inference 말고 training할 때도 파라미터 수가 적으면 더 좋지 않을까?***
      However, if a network can be reduced in size, why do we not train this smaller architecture instead in the interest of making training more efficient as well?
    - ***어차피 pruning 한 다음에 다시 학습을 시켜야 하는데, pruning을 먼저 할 수는 없을까?***

*Frankle, J., & Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

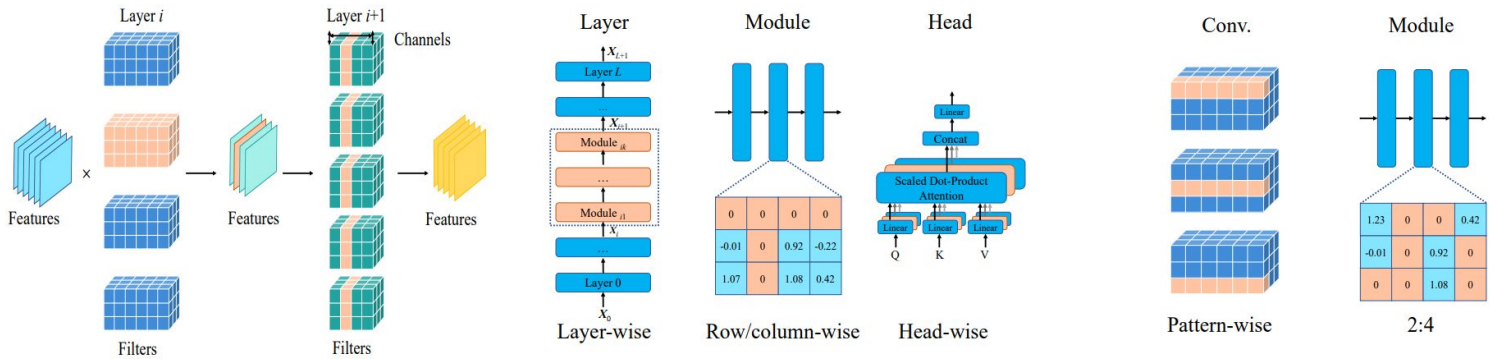
*https://velog.io/@bismute/The-Lottery-Ticket-Hypothesis%EC%99%80-%EA%B7%B8-%ED%9B%84%EC%86%8D-%EC%97%B0%EA%B5%AC%EB%93%A4-%EB%A6%AC%EB%B7%B0#%EB%85%BC%EB%AC%B8%EC%9D%98-%EB%82%B4%EC%9A%A9

Unstructured Pruning의 한계





값이 0이 되면 파라미터가 없어진 것처럼 보이지만,
실제로는 Mask 연산도 해야하고,
전체 행렬 연산에서 줄어든 부분은 없음

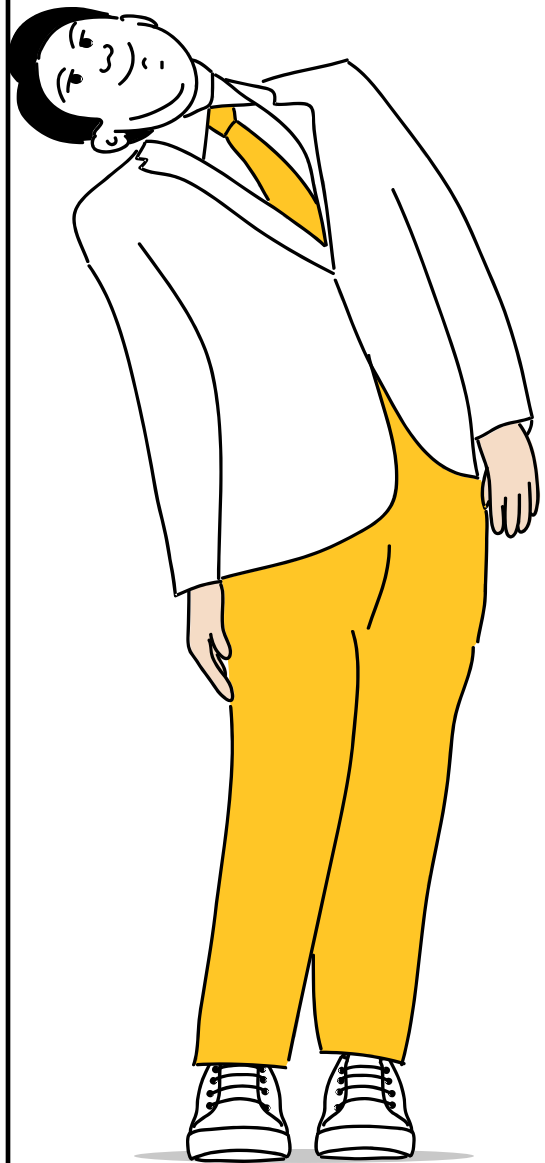


따라서 실제로 연산량을 줄일 수 있는 STRUCTURED PRUNING 필요

*Cheng, Hongrong, Miao Zhang, and Javen Qinfeng Shi. "A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations." IEEE Transactions on Pattern Analysis and Machine Intelligence (2024).

감사합니다

2025. 03. 12 (Wed) 온디에 1주차 구승연



추가로,
https://github.com/google-research/lottery-ticket-hypothesis