Deep RL 기반 Neural Network Pruning 구현

최종 결과 발표

N팀
20196638 장재호

프로젝트 목표

Neural Network를 pruning 하기 위한 심층강화학습 기반 알고리즘 구현

팀원: 20196638 장재호 [의료AI융합전공]

구현 코드: https://github.com/mainjj/RL_Pruner

1. 서론

1.1. 문제 설명

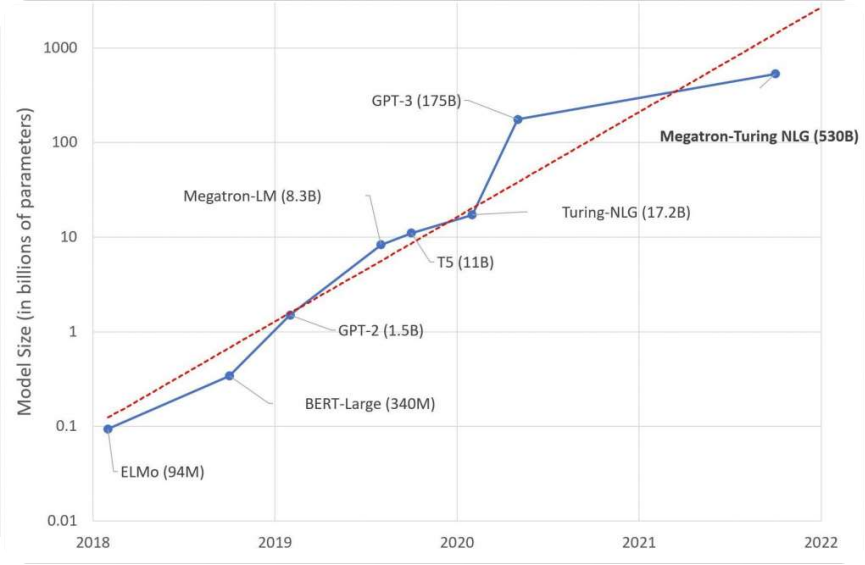
1.2. 딥러닝으로 해결이 어려운 이유

1.3. 강화학습으로 도전하는 이유

1.1 문제 설명

점점 **커지는** 신경망 **규모**
그에 따른 **많은 Resource**가 필요하다

- DNN을 **학습에** 상당한 **resource**가 **필요**
  추론에도 많은 resource가 **필요**
- 리소스가 **제한된 환경**에 **배포**해야 할 때 이런 **문제**는 커진다.



[년도별 model size]

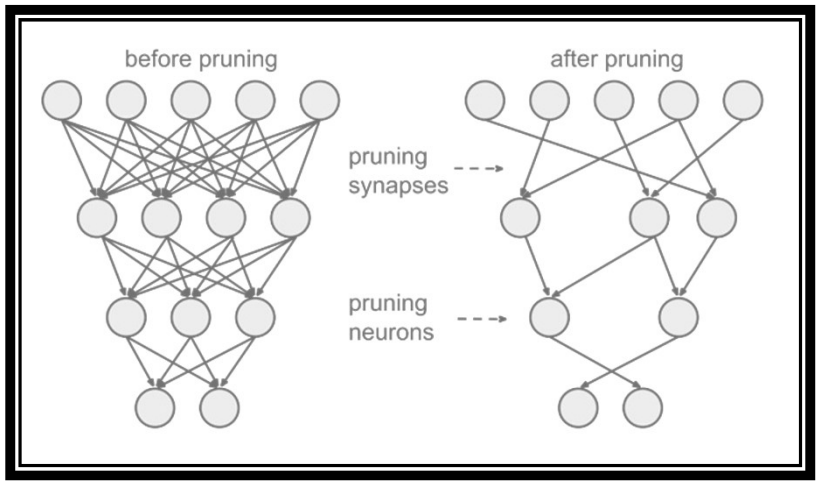
Ref: https://blogs.microsoft.com/ai-for-business/ai-at-scale-technology/

1.1 문제 설명

모델 크기 감소을 위해 **경량화**를 진행

Pruning

- **중요도**가 낮은 **weight**들을 **제거**하여 파라미터 수를 줄이는 방법



Pruning의 **단점**

- 중요도가 **낮은 weight**를 **알 수 없음**
- **시행착오**가 필요함
- **전문가**들의 **개입**이 필요함

Ref: https://blogs.microsoft.com/ai-for-business/ai-at-scale-technology/

1.2 딥러닝으로 해결이 어려운 이유

- **Supervised learning**의 경우
  - **제거해야** 할 **weight**를 **미리 알 수 없기** 때문에 어렵다.

- **Unsupervised learning**의 경우
  - pruning을 [하는 node, 하지 않는 node]로 구분할 수 있겠지만,
  - **강화 학습**과 같이 적절한 보상을 주면서 학습하는 것보다 **더 많은 데이터**가 필요하고 수렴 속도가 **느릴 수 있다**.

1.3 강화학습으로 도전하는 이유

기존의 pruning **방법**은

전문가의 **trial and error**에 의존하여 진행되었다.

이런 **경험적인 과정은** 번거롭고 비효율적이다.

강화학습을 활용해 보다 효율적으로 문제에 접근할 수 있다.

2. 관련 연구 및 기술 동향

2.1 관련 연구 및 기술 동향

Magnitude 기준 pruning

- **크기**(magnitude)가 **작은** weights들을 **제거**하는 방법
- Weight값이 0에 가까울수록 그 값이 모델의 출력에 미치는 영향이 적기 때문

THE LOTTERY TICKET HYPOTHESIS

- 전체 network와 **비슷한 성능**을 내는 **sub-network**(winer ticket)가 **존재**한다는 것을 밝힘
- **Winer ticket**을 **재학습**(retrain)하면 정확도 차이가 크게 안 난다는 사실을 보여줌
- **재학습을** 해야 한다는 **단점 존재**

3. 방법

3.1. 강화학습 시스템 정의

3.2. 심층 신경망 설계

3.3. 정책기반 강화학습 알고리즘 **(pseudo code)**

3.1 강화학습 시스템 정의

Pruning할 DNN: **ResNet50** (PyTorch, CIFAR10 dataset으로 학습)

- **ResNet50**은 **총 54**개의 layer들로 (Convolution, Fully-connected layer) 이루어져 있다.
- **54개의 layer**에 대한 **pruning**을 진행한다.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

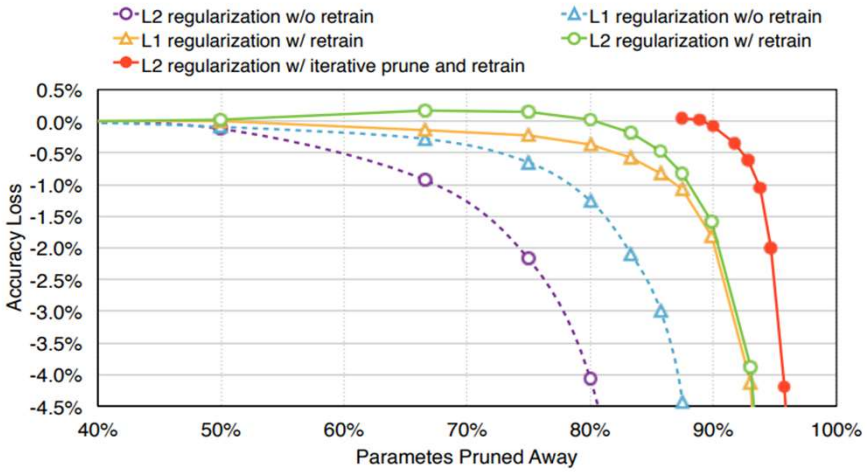
Ref: https://arxiv.org/abs/1512.03385

3.1 강화학습 시스템 정의

Trade off를 가지는 **Sparsity**와 **Accuracy**

Sparsity : 0인 가중치의 비율.

sparsity와 **accuracy** 간의 **균형**을 맞추는 것이 중요합니다.



Ref: Learning both Weights and Connections for Efficient Neural Networks

3.1 강화학습 시스템 정의

✓ **Environment**

- Agent가 DNN(ResNet50)을 pruning하고 reward 및 state를 받을 수 있는 시스템

✓ **Agent**

- **하나의 layer**에 대한 **prune ratio**를 정해준다.
  - prune ratio: 하나의 layer에 있는 weights들을 L1 norm순으로 정렬한 후, 작은 값들을 얼마나 삭제할지 나타내는 비율입니다.

[한번의 time step 예시]

3.1 강화학습 시스템 정의

✓ **State**

1. **model sparsity**: model 전체에서 weights가 0이 된 비율
2. **layer sparsity**: 이번에 pruning할 layer에서 weights가 0이 된 비율
3. **현재 test accuracy** : 현재 model의 테스트 정확도
   - Pruning 이후 측정한 test accuracy
   - **Action**을 하며 값이 **변경**된다.
4. **해당 layer를 가장 최근에 prune한 뒤 측정한 test accuracy**
   - Layer를 pruning한 후 test accuracy를 측정합니다. 그런 다음, 그 값을 **저장**하고 **다음 번에 해당 레이어를 다시 pruning할 때 사용합니다**.
   - 초기값 = prune을 안 한 model의 test accuracy
   - **Action**을 하며 값들이 **변경**된다.

3.1 강화학습 시스템 정의

✓ **Action**

- **Action space**는 $\boldsymbol{A = \{a \mid a = 0, 0.01, 0.02, \ldots, 0.99\}}$으로 0.01단위로 0~0.99까지 총 **100개**로 나누어져 있다.
- layer하나에 대한 **prune_ratio** a를 **결정**한다. 해당 layer에 있는 가중치 값을 L1 norm을 기준으로 정렬을 한 뒤, 크기가 작은 순으로 $a\%$ 만큼 제거된다.

✓ **Reward**

- Reward는 pruned model의 **Accuracy**가 **높을 때**와 **Sparsity**가 **높을 때**에 주어진다.

$$\boldsymbol{reward = 5\left(\frac{Acc}{Original\_Acc} + \frac{sparsity}{0.8}\right)}$$

- Acc: pruning 이후 정확도
- $Original\_Acc$: pruning을 안 했을 때의 정확도
- $Sparsity$: pruning 이후 sparsity

3.2 심층 신경망 설계

- **Fully-connected layer 3개**와 **Layer Normalization**으로 이루어진 network이다.
  - output_dim은 action space개수인 100이다.

```python
class PolicyNetwork(nn.Module):
    def __init__(self, input_dim=4, hidden_dim=128, output_dim=100):
        super(PolicyNetwork, self).__init__()
        self.fc1 = nn.Linear(input_dim, hidden_dim)
        self.ln1 = nn.LayerNorm(hidden_dim)
        self.fc2 = nn.Linear(hidden_dim, hidden_dim)
        self.ln2 = nn.LayerNorm(hidden_dim)
        self.fc3 = nn.Linear(hidden_dim, output_dim)
        self.softmax = nn.Softmax(dim=-1)

    def forward(self, x):
        x = torch.relu(self.ln1(self.fc1(x)))
        x = torch.relu(self.ln2(self.fc2(x)))
        x = self.softmax(self.fc3(x))
        return x
```

정책기반 강화학습 알고리즘 (pseudo code)

```
1.  DNN, Agent 초기화
2.  While (MAX_EPISODE) do
3.      DNN load
4.      For 각 layer in DNN do
5.          a <- agent에서 action sampling
6.          a를 사용하여 해당 layer prune
7.          DNN test진행 (accuracy를 얻기 위함)
8.          Reward및 new state 계산
9.      If (Prune된 DNN이 best일 시) -> save
10.     EPISODE += 1
```

- 실험

1. 실험 환경
2. 평가 방법
3. 강화학습 학습 로깅 결과
4. 체크포인트 성능 비교
5. 중간결과와의 성능 비교

4.1 실험 환경 및 평가 방법

평가 방법

Baseline_net, Pruned_net, Retrained_net의 **성능**(FLOPs, Accuracy, parameter 수) **비교**해서 평가를 진행한다.

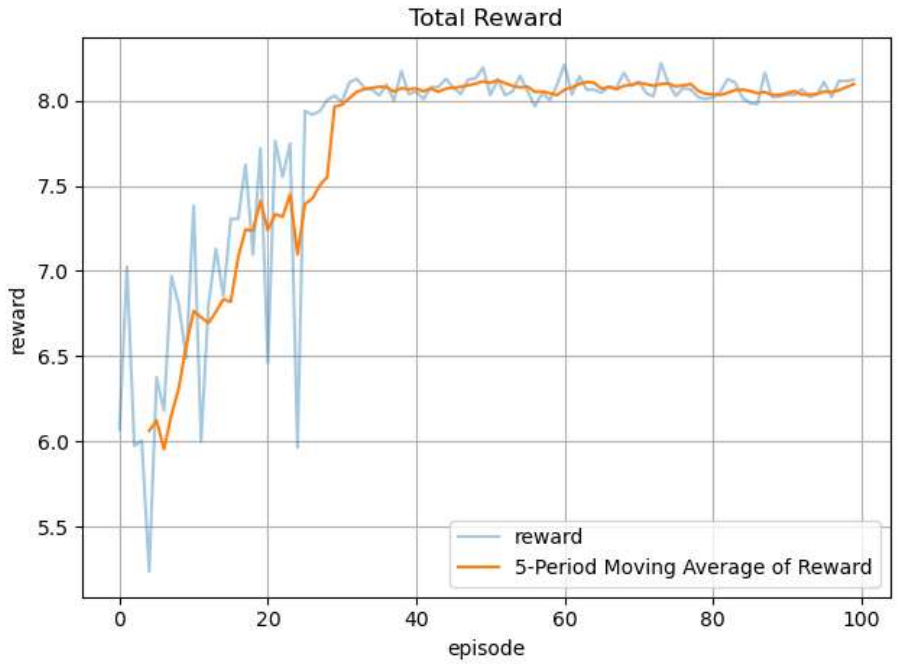
- Baseline_net : pruning을 하기 전 model
- pruned_net : pruning을 한 model
- Retrained_net : pruning 이후 100epochs만큼 재학습한 model

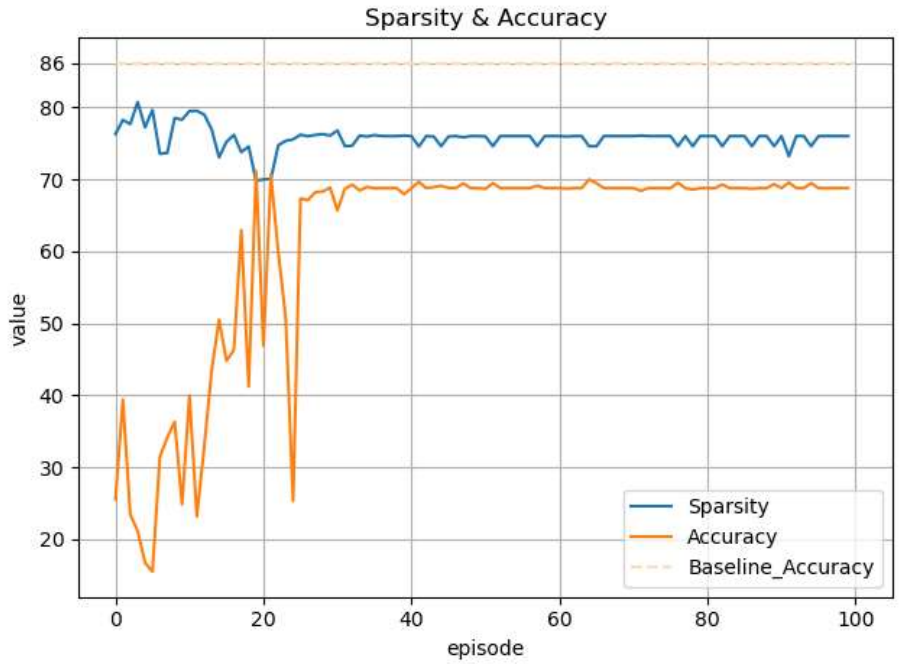
실험환경	
OS	WSL, Ubuntu 22.04.4 LTS
GPU	NVIDIA GeForce RTX 3060 Ti
Library	PyTorch
Dataset	CIFAR-10
Model	Resnet50

4.3 강화학습 학습 로깅 결과

학습결과 **Sparsity: 76.00**, **Accuracy: 68.77**로 **수렴**하는 모습을 보임

Trade off 관계인 Sparsity와 Accuracy의 **균형**을 찾는 **적절점**을 잘 찾은 것으로 보임

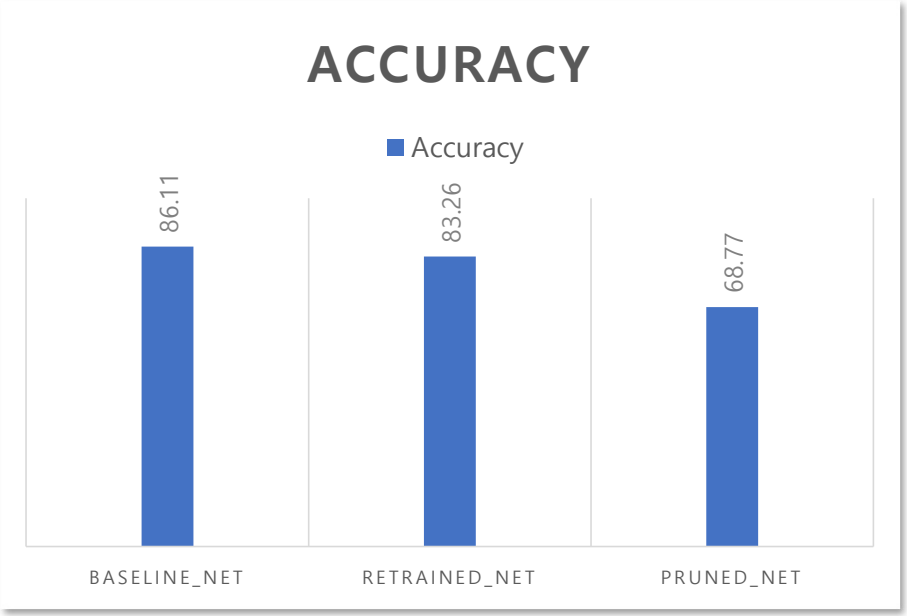


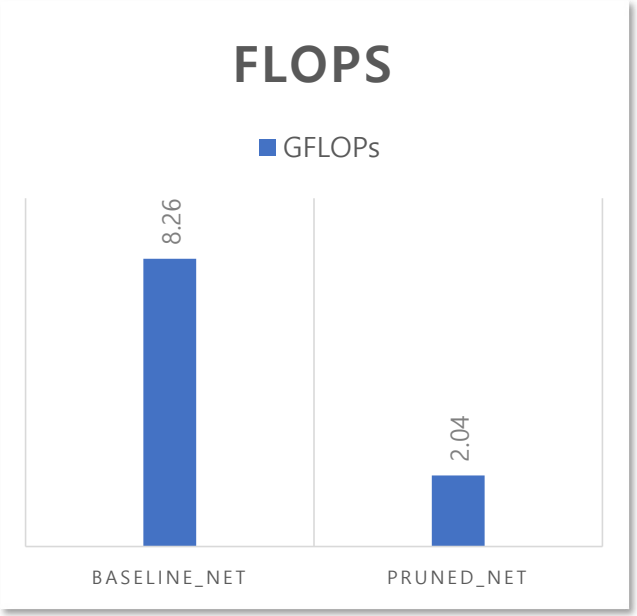


4.3 강화학습 학습 로깅 결과

Baseline_net, Pruned_net, Retrained_net의 성능비교

- **parameter 수**의 경우 23,528,522개에서 5,687,226개로 **75% 감소**하였다.
- **FLOPs**를 보면 pruning이후 **4배** 가량 **빨라진 모습**을 보인다.

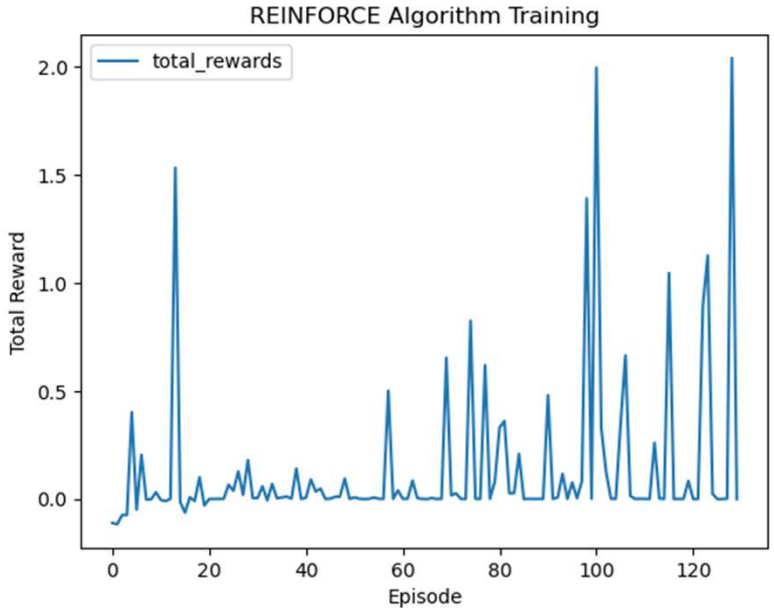




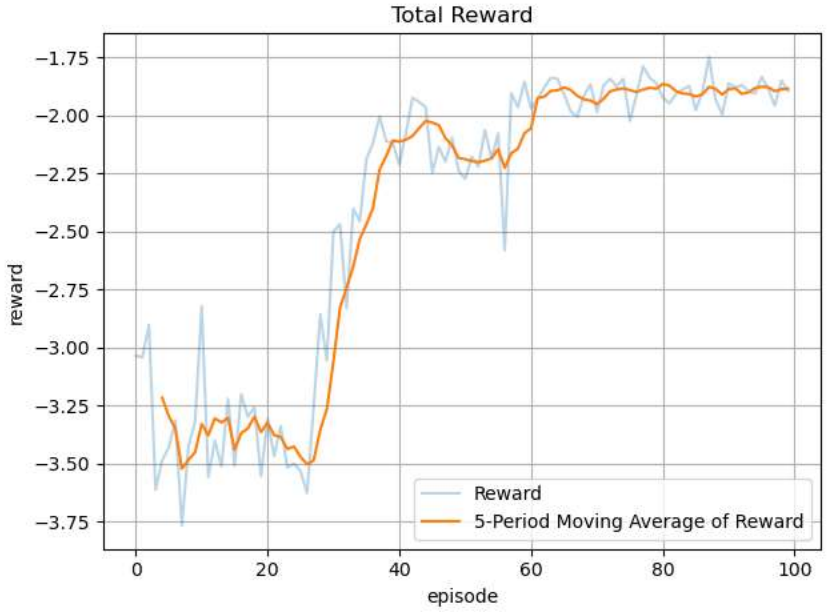
4.4 체크포인트 성능 비교

Discount factor에 따라 성능 차이가 심함을 경험하였다. 여러 차례 경험에 따라 현재는 0.75를 사용한다.

[discount factor: 0.99 사용 시 reward]



[discount factor: 0.75 사용 시 reward]



4.5 중간결과와의 성능비교

- **중간 결과**에선 **학습 중**에 pruning된 모델들은 별도로 **저장하지 않았다**.
  - 학습이 끝난 후 inference를 돌렸다.

- **학습 도중** pruning된 모델의 **성능이 좋을 경우** 저장하여 더 다양한 상황에 맞춰 사용할 수 있다.
  - **Sparsity**가 더 높은 걸 원하는 경우
  - **Accuracy**가 더 높은 걸 원하는 경우

[학습중 저장된 pruned model들 sparsity와 accuracy를 확인할 수 있다.]

4.5 중간결과와의 성능비교

- **중간 결과**에선 **학습 중**에 pruning된 모델들은 별도로 **저장하지 않았다**.
  - 학습이 끝난 후 inference를 돌렸다.

- **학습 도중** pruning된 모델의 **성능이 좋을 경우** 저장하여 더 다양한 상황에 맞춰 사용할 수 있다.
  - **Sparsity**가 더 높은 걸 원하는 경우
  - **Accuracy**가 더 높은 걸 원하는 경우

중간 결과 Retrained_net

- sparsity 71%
- accuracy 83.79%

최종 결과 Retrained_net

- sparsity 76% (+5)
- accuracy 83.26% (-0.53)

- 토의

1. 현 강화학습 시스템의 한계점
2. 개선사항 제안

5.1 현 시스템의 한계점 및 개선사항 제안

한계점

- 현재 학습 시 Pruning할 **layer**를 **random sampling**을 해주고 있습니다.
- **Layer마다** pruning을 하는 정도가 **다름**
  - **초기 단계 layer**들을 pruning하면 **전반적인 성능**에 큰 영향을 미칠 수 있습니다.
  - **기본 특징**을 잃게 되어 **후속 layer**들이 제대로 **동작하지 않을 수 있기** 때문입니다.

개선방안

Model의 상황(layer별 sparsity, sensitivity 등)을 고려하여 **pruning을 할 layer를 직접 골라 pruning하는 RL**을 설계하여 해결할 수 있다.

- 기대효과

1. 프로젝트 결과물의 활용 예

6.1 기대효과

- **전문가의 시행착오가 없어도 경량화 할 수 있습니다.**
  - Model의 배포 과정에서 비용을 절약하게 할 수 있다.

- **모델을 배포할 때 발생하는 계산 및 메모리 부담을 줄일 수 있습니다.**
  - 특히 리소스가 제한된 환경에서의 성능을 향상시킬 수 있다.
  - 모바일 기기, IoT 장치 등 리소스가 제한된 환경에서 효율적으로 동작한다.
  - 낮은 전력 소비량으로 빠른 inference time을 기대할 수 있다.

감사합니다.

REF

The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks: https://arxiv.org/abs/1803.03635

AMC: AutoML for Model Compression and Acceleration on Mobile Devices: https://arxiv.org/abs/1802.03494

Learning both Weights and Connections for Efficient Neural Networks: https://arxiv.org/abs/1506.02626