[2025-2 강화학습 프로젝트]
Active Learning for Image Classification with Reinforcement Learning

---

Github link: https://github.com/zoowon/AL_with_RL
Google drive link: https://drive.google.com/drive/folders/1XTuVJQ7raXbRQ53PB5zzwCeDJwUx--2x?usp=drive_link

박윤서 (120250650)
박주원 (120250651)

Department of Computer Science and Engineering
Sogang University

SOGANG UNIVERSITY

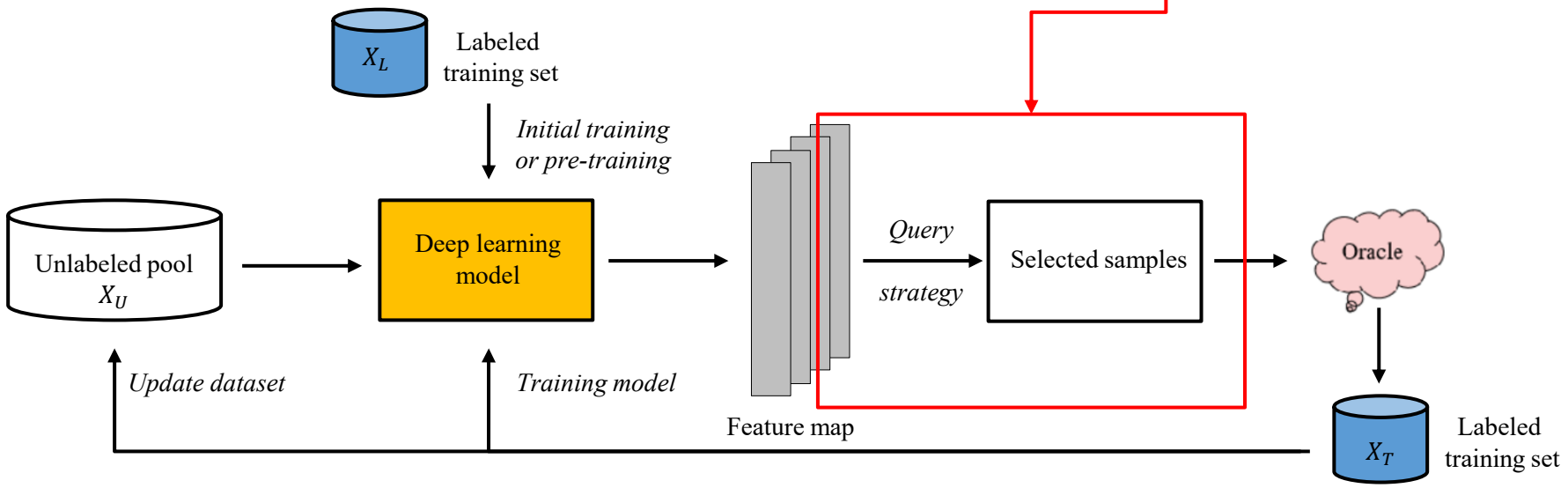
Contents

프로젝트 주제 및 목표

- 주제 : 강화학습을 이용한 이미지 분류에서의 능동학습 (active learning)

1. 주제 선정 과정

  - 팀원들의 주 연구 분야인 computer vision 이라는 큰 틀 안에서, **강화학습의 프레임워크를 적용**할 수 있는 task에 대한 탐색을 진행
  - 기존에 진행했던 연구*인 active learning task에서 사용하는 sampling (querying) 기법을 하나의 action으로 봤을 때, 강화학습 프레임워크를 적용할 수 있다고 판단



<Active learning의 전체적인 진행 과정>

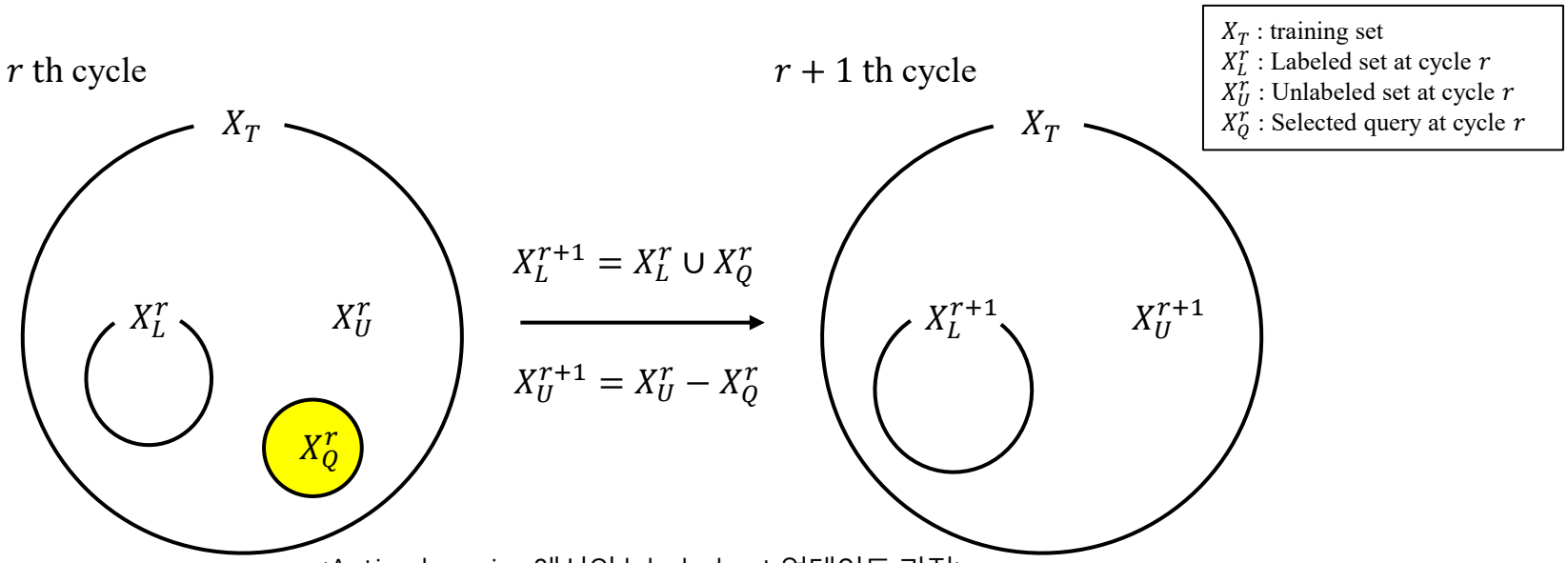
* 박주원, 박재현, 전주현, 조성인, "객체 탐지에서의 액티브 러닝 기술 동향 분석," 대한전자공학회 학술대회, 2024, pp. 1433-1437.

프로젝트 주제 및 목표

- 주제 : 강화학습을 이용한 이미지 분류에서의 능동학습 (active learning)

2. Active learning (AL) : 모델이 학습에 사용할 데이터를 **능동적으로 선택하는** 머신 러닝 기법

- **최소한의 labeling 비용으로 모델 성능을 최대화**하는 것을 목표
  - 기존의 머신 러닝 기법들이 가지는 **labeling 비용 문제를 해결**하기 위함
- 여러 번의 cycle에 걸쳐, unlabeled set으로부터 정보량이 높은 샘플을 **선택 (sampling, querying)**하여 labeled set에 **추가 (업데이트)**하고, 이를 사용해 모델을 다시 학습시킨 뒤 **각 cycle 마다 모델 성능을 평가**
  - 정보량이 많다는 것은 다음 cycle에서 모델 학습에 큰 도움을 준다는 것을 의미함



<Active learning에서의 labeled set 업데이트 과정>

프로젝트 주제 및 목표

■ 목표

1. 같은 양의 labeled 데이터에 대하여, random sampling 기법보다 높은 성능 달성

- Random sampling이란?
  - 매 cycle 마다 labeled set을 업데이트 할 때, unlabeled set에서 무작위로 샘플들을 추가하는 기법
  - Random seed 값에 따라 성능 차이가 존재
- **동일한 random seed 값에 대하여, 같은 cycle에서 더 높은 성능을 발휘하는 모델 학습이 목표**
  - 같은 cycle에서는 같은 양의 labeled 데이터를 활용
  - 양은 같지만 높은 성능을 보이는 것은 **labeled set이 정보량이 많은 데이터들로 구성되어 있음**을 의미

2. 효과적인 sampling을 위한 최적의 DQN 구현

- **Gradient step 변화 실험을 진행**하여, 가장 효과적인 sampling을 위한 최적의 DQN을 구현
  - State, action, reward 세팅 및 random seed 값은 고정
  - **같은 cycle에서 가장 높은 성능을 발휘하는 최적의 gradient step 수를 관찰**

환경 및 데이터셋

- 환경 : Active learning 프레임워크 활용

1. 데이터셋 별로 지정된 양의 샘플을 랜덤하게 선택하여, labeled set으로 사용

2. 총 10회의 cycle로 구성하여, 매 cycle 마다 이전 cycle에서 결정된 labeled set을 활용한 훈련 진행
   - 매 cycle 마다 정해진 labeled set과 이미지 분류기 (ResNet-18)를 활용하여 이미지 분류 훈련 진행
   - 매 cycle 마다 훈련 진행 후, test set에 대한 이미지 분류 성능 평가 및 DQN 기반 sampling 진행
     – DQN 네트워크의 training step (gradient step)은 실험 마다 지정 가능하도록 설계

3. 마지막 cycle 이후, 최종적으로 훈련된 이미지 분류 모델의 checkpoint를 저장

환경 및 데이터셋

■ 데이터셋

1. CIFAR-10
   - 32×32 컬러 이미지, 10개 클래스 (비행기, 자동차, 새, 고양이 등)
   - 총 60,000장 이미지 (Train 50,000 / Test 10,000)
   - 소규모 이미지 분류 및 기본 실험에 널리 사용됨

<CIFAR-10의 예>

2. CIFAR-100
   - 32×32 컬러 이미지, 100개 세부 클래스
   - 총 60,000장 이미지 (Train 50,000 / Test 10,000)
   - CIFAR-10보다 클래스 수가 많아 분류 난이도가 더 높음

3. Fashion MNIST
   - 28×28 흑백 의류 이미지, 10개 클래스 (T-shirt/top, Trouser, Pullover 등)
   - 총 70,000장 이미지 (Train 60,000 / Test 10,000)
   - 전통적인 MNIST (숫자) 대신 사용하는 의류 이미지 분류 벤치마크

<Fashion MNIST의 예>

환경 및 데이터셋

- 데이터 preprocessing

1. CIFAR-10, CIFAR-100

  - Train
    - `RandomHorizontalFlip()` : 이미지를 좌우로 랜덤 뒤집기 (데이터 다양성 증가)
    - `RandomCrop(size=32, padding=4)` : 가장자리에 4픽셀 패딩 후 32×32 랜덤 크롭
    - `ToTensor()` : [0, 1] 범위의 PyTorch 텐서 (tensor)로 변환
    - `Normalize(mean, std)` : 채널별 평균/표준편차로 정규화
  - Test
    - `ToTensor()`
    - `Normalize(mean, std)`

환경 및 데이터셋

- 데이터 preprocessing

2. Fashion MNIST

- Train
  - `Resize(32)` : 원본 28×28 이미지를 32×32로 리사이즈
  - `Grayscale(num_output_channels=3)` : 1채널 이미지를 3채널로 복제 (RGB 모델과 형식 통일)
  - `RandomHorizontalFlip()` : 이미지를 좌우로 랜덤 뒤집기
  - `RandomCrop(size=32, padding=4)` : 4픽셀 패딩 후 32×32 랜덤 크롭
  - `ToTensor()`
  - `Normalize(mean, std)`
- Test
  - `Resize(32)`
  - `Grayscale(num_output_channels=3)`
  - `ToTensor()`
  - `Normalize(mean, std)`

강화학습 알고리즘

- State

1. 한 AL cycle에서 학습된 classifier의 출력 logit들을 활용하여 결정된 $C + 3$ 차원의 벡터
2. 구성 요소

- **전체 softmax 확률** : `probs = softmax(logits)`
  - 현재 모델이 각 클래스를 얼마나 믿고 있는지 반영 (길이 : C … C 는 데이터셋의 클래스 개수)
- **최대 확률** : `max_prob = max(probs)`
  - 모델의 전반적인 확신 정도를 하나의 스칼라 값으로 요약
    - » 1에 가까울수록 매우 확신, $1/C$에 가까울수록 모호
- **Margin**
  - 가장 큰 확률 값과 두 번째로 큰 확률 값의 차이 (top1 - top2)
    - » 값이 크면 1등 클래스가 명확 → 확실한 샘플
    - » 값이 작으면 상위 두 클래스가 비슷 → 경계에 위치한 샘플
- **Entropy (불확실도)**

$$Entropy = -\sum p_i \log p_i$$

  - 예측 확률 분포가 고르게 퍼져 있으면 (모델이 모호하게 판단하면) 값이 커짐
  - 예측 확률 분포가 한 클래스에 쏠려 있으면 (모델이 확신하여 판단하면) 값이 작아짐

강화학습 알고리즘

- Action (a)

1. binary action ($a \in \{0,1\}$)으로 구성
   - Action 1 ($a = 1$) : 해당 샘플을 이번 cycle에서 labeling할 샘플로 선택 (select)
   - Action 0 ($a = 0$) : 해당 샘플을 선택하지 않음 (skip)

- Reward (r)

1. 선택된 샘플에 대해 현재 classifier의 예측과 정답 라벨을 비교
   - 오분류인 경우 : $r = 1$ (학습 가치가 큰 샘플)
   - 정분류인 경우 : $r = 0$
2. 미선택 샘플 일부에 대해서는 $a = 0,\ r = 0$ 으로 버퍼에 저장
3. 에피소드 길이가 1인 contextual bandit으로 가정 (discount factor : $\gamma = 0$)
   - $Q(s,a) \approx \mathbb{E}[r|s,a]$ 를 학습

강화학습 알고리즘

- Q-Network

1. 상태 (state)를 입력으로 받아 각 행동 (action)에 대한 Q값 ($Q(s,a)$)을 출력하는 함수 근사기
   - 입력 : 분류기 logit에서 추출한 불확실도 특징 (확률, margin, entropy 등)
   - 출력 : $[Q(s,0), Q(s,1)] \in \mathbb{R}^2$
     - $Q(s,0)$: 이 샘플을 선택하지 않았을 때의 기대 보상
     - $Q(s,1)$: 이 샘플을 선택했을 때의 기대 보상
   - DQN 에이전트가 Q-Network를 이용해 지금 이 샘플을 선택하면 얼마나 이득인지를 수치로 평가
   - $Q(s,1)$ 만을 score로 사용하여, score가 큰 상위 K개 샘플을 다음 cycle에 labeled dataset에 추가할 샘플로 선택

2. 손실 계산
   - 평균 제곱 오차 (Mean Squared Error, MSE) 사용

$$\mathcal{L} = \frac{1}{B}\sum_{i=1}^{B}\left(Q_\theta(s_i, a_i) - r_i\right)^2$$

$Q_\theta(s,a)$: 예측 값
r : 정답 (reward)

강화학습 알고리즘

- Hyperparameter 설명

1. Dimension of hidden layer : `hidden_dim = 128`
   - Q-network의 은닉층 (hidden layer) 크기
   - State 차원이 비교적 작은 task … 너무 깊은 네트워크는 과적합 및 학습 불안정 문제 발생

   → 중간 정도의 크기 선택

2. Learning rate : `lr = 1e-3`
   - Adam optimizer의 학습률 (learning rate)
   - 값이 너무 크면 Q값이 발산하고, 너무 작으면 정책 변화가 매우 느림

   → 경험적으로 안정적인 범위인 1e-3 사용

3. Buffer capacity : `buffer_capacity = 10000`
   - Replay buffer에 저장할 수 있는 최대 transition 수
   - 메모리 사용량과 다양성 사이의 타협

   → 여러 AL cycle에서 다양한 샘플을 축적하기 위한 충분한 크기로 지정

강화학습 알고리즘

- Hyperparameter 설명

4. Batch size : `batch_size = 64`
   - DQN 업데이트 시 한 번에 사용할 transition 수

   → 일반적인 분류/강화학습 설정에서 자주 쓰는 값 사용

5. Gradient step : `train_steps = 200`
   - 한 Cycle 당 DQN 업데이트 스텝 수
   - Replay buffer를 기반으로 미니 배치 업데이트를 몇 번 반복할지 결정

   → **실험을 통해 DQN이 새로 추가된 경험에 충분히 적응하도록 하기 위한 최적의 값을 관찰**

실험 셋업

▪ 실험 환경

1. 서버 스펙

- CPU : AMD Ryzen Threadriper PRO 3975WX 32-Cores
- GPU : NVIDIA RTX A6000
- CUDA : v11.5

2. 가상 환경 (Github repository에 requirements.txt 파일 저장 및 README 내 설치 방법 안내 완료)

- Python : v3.8.20
- PyTorch : v1.10.2
- Torch vision : v1.21.6
- CUDA toolkit : v11.3.1
- Numpy : v1.21.6
- Tqdm : v4.66.2

실험 셋업

- 평가 지표 (Evaluation metric)

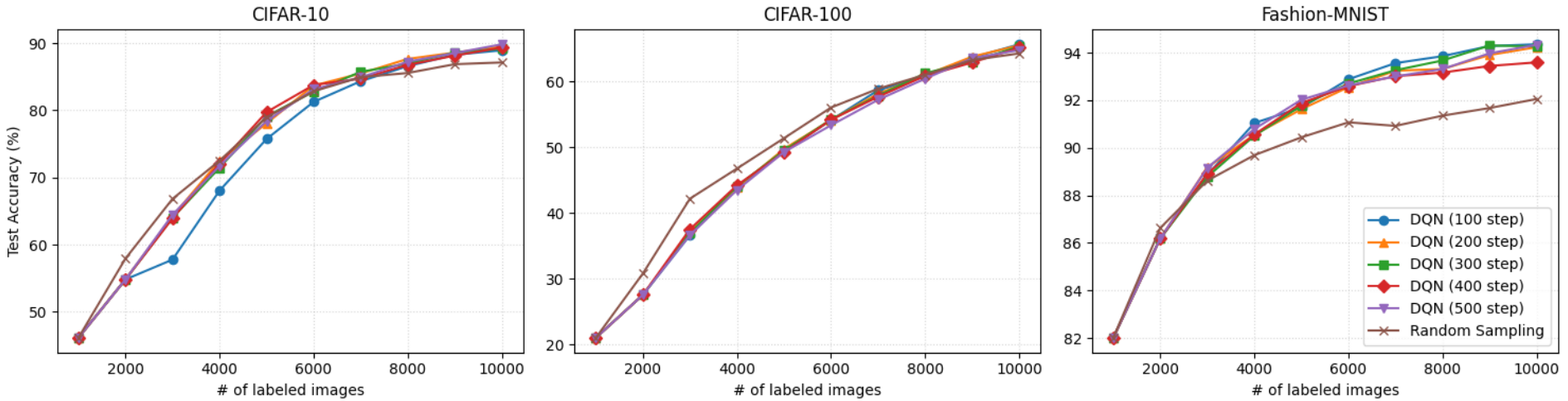
1. Accuracy ↑ (%)
   - 모델이 예측한 label과 실제 label 간의 비교를 통해 정확도 (정답 비율)를 계산
   - 값이 높을수록 좋은 성능을 발휘하는 모델

- 결과 분석 기준

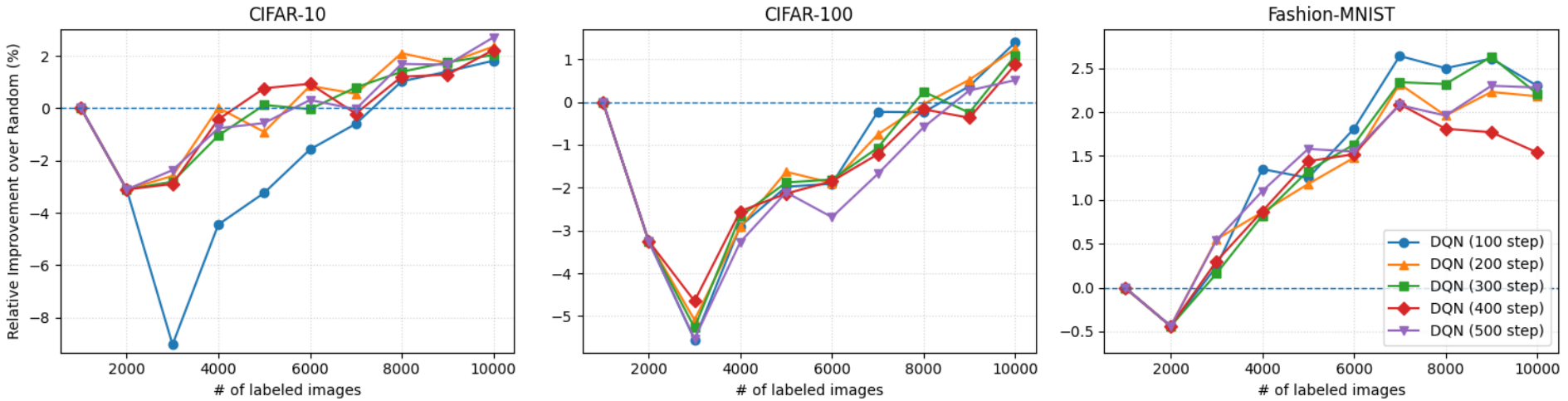
1. 같은 cycle에서의 accuracy 차이 비교
   - 같은 양의 labeled set을 활용 했을 때, 어떤 sampling 기법을 활용한 모델이 더 좋은 성능을 보여주는지 비교 가능
2. 각 cycle 사이의 accuracy 증가 폭 비교
   - 어떤 sampling 기법을 활용했을 때 더 효과적으로 정보량이 많은 데이터를 sampling하는지 상세한 비교 가능
3. 첫 번째 cycle과 마지막 cycle 사이의 증가 폭 비교
   - 어떤 sampling 기법을 활용했을 때 더 효과적으로 정보량이 많은 데이터를 sampling하는지 전반적인 비교 가능

실험 결과

- Summary (`RANDOM_SEED = 0`)



<Visualization of test accuracy >



<Visualization of relative improvement over random sampling>

실험 결과

- 데이터셋 별 실험 결과 (RANDOM_SEED = 0)

1. CIFAR-10

Method	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6	Cycle 7	Cycle 8	Cycle 9	Cycle 10
Random Sampling	46.15	57.94	66.82	72.50	78.97	82.83	84.93	85.55	86.86	87.12
DQN (100 step)		54.83	57.78	68.05	75.73	81.25	84.33	86.57	88.26	88.93
DQN (200 step)			64.23	72.51	78.06	83.68	85.48	87.65	88.58	89.47
DQN (300 step)			64.01	71.45	79.09	82.79	85.72	86.94	88.61	89.15
DQN (400 step)			63.92	72.07	79.73	83.76	84.72	86.75	88.14	89.34
DQN (500 step)			64.45	71.73	78.40	83.14	84.89	87.24	88.52	89.82

실험 결과

- 데이터셋 별 실험 결과 (RANDOM_SEED = 0)

2. CIFAR-100

Method	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6	Cycle 7	Cycle 8	Cycle 9	Cycle 10
Random Sampling	21.03	30.81	42.19	46.77	51.39	56.05	58.91	61.03	63.27	64.28
DQN (100 step)		27.56	36.64	43.87	49.41	54.14	58.68	60.79	63.64	65.67
DQN (200 step)			37.10	43.85	49.76	54.16	58.15	60.98	63.79	65.52
DQN (300 step)			36.92	44.10	49.51	54.24	57.84	61.27	63.02	65.36
DQN (400 step)			37.54	44.22	49.25	54.20	57.69	60.86	62.90	65.15
DQN (500 step)			36.66	43.50	49.28	53.36	57.23	60.44	63.54	64.79

실험 결과

- 데이터셋 별 실험 결과 (RANDOM_SEED = 0)

3. Fashion MNIST

Method	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6	Cycle 7	Cycle 8	Cycle 9	Cycle 10
Random Sampling	82.01	86.63	88.62	89.69	90.44	91.07	90.92	91.35	91.67	92.05
DQN (100 step)		86.19	88.86	91.04	91.69	92.88	93.56	93.85	94.28	94.35
DQN (200 step)			89.17	90.55	91.62	92.55	93.24	93.31	93.90	94.23
DQN (300 step)			88.78	90.51	91.77	92.70	93.26	93.67	94.30	94.26
DQN (400 step)			88.92	90.56	91.88	92.59	93.01	93.16	93.44	93.59
DQN (500 step)			89.16	90.79	92.02	92.62	93.00	93.31	93.97	94.33

실험 결과

- Random seed 변경을 통한 실험 (`RANDOM_SEED = time_seed`)

1. 실험 규칙
   - Random seed를 실행 시간 (time 라이브러리 활용)으로 지정
   - 각 데이터셋 별로 찾아낸 최적의 gradient step*을 활용하여, 각각 5번의 실험 진행 후 평균 값 관찰

2. 실험 결과

Dataset	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6	Cycle 7	Cycle 8	Cycle 9	Cycle 10
CIFAR-10 (200 step)	46.378	56.352	66.186	74.178	80.788	84.486	86.074	87.538	88.854	89.632
CIFAR-100 (100 step)	19.820	28.390	38.708	45.446	51.328	55.796	59.000	61.956	64.278	66.124
Fashion MNIST (100 step)	81.898	85.786	88.302	89.816	90.958	91.812	92.650	93.208	93.474	93.844

* 다음 슬라이드부터 서술되는 데이터셋 별 실험 결과 분석을 통하여 결정

토의 및 결론

■ 데이터셋 별 실험 결과 분석

1. CIFAR-10

- Random sampling과의 결과 비교
  - DQN이 제대로 학습되지 않은 cycle 2~3까지는 random sampling을 활용한 결과가 가장 우수함
  - Cycle 4 이후로는 DQN을 활용한 sampling 기법이 거의 대부분 더 우수한 성능을 보임
- 최적의 DQN gradient step 분석
  - 200 step에서 cycle 별로 우수한 성능을 기록한 횟수가 가장 많음
    - » 가장 안정적으로 우수한 성능을 계속 발휘한 200 step을 최적의 gradient step으로 판단

2. CIFAR-100

- Random sampling과의 결과 비교
  - 정답 label이 다양한 어려운 데이터셋 … DQN이 학습되기까지 많은 cycle이 필요
  - Cycle 9 이후에는 DQN을 활용한 sampling 기법이 더 우수한 성능을 보임
- 최적의 DQN gradient step 분석
  - 전체적으로 유사한 성능 분포 … 100 step에서 cycle 별로 우수한 성능을 기록한 횟수가 가장 많음
    - » 가장 가벼운 모델임에도 불구하고 안정적으로 좋은 성능을 발휘하는 100 step을 최적의 gradient step으로 판단

토의 및 결론

- 데이터셋 별 실험 결과 분석

3. Fashion MNIST

- Random sampling과의 결과 비교
  - DQN이 거의 학습되지 않은 cycle 2 이후로는, train step 수에 관계 없이 DQN을 활용한 sampling 기법이 더 우수한 성능을 보임
- 최적의 DQN gradient step 분석
  - 100 step에서 cycle 별 최고 성능을 기록한 횟수가 가장 많음
    - » 가장 가벼운 모델임에도 불구하고 좋은 성능을 발휘하는 100 step을 최적의 gradient step으로 판단

토의 및 결론

- 보완 및 개선사항

1. DQN 기반 sampling의 문제점

- Sampling을 위한 모델 (DQN) 자체가 학습이 필요한 설계

→ 초반 sampling 시 성능 향상폭이 random sampling보다도 떨어지는 문제가 발생

2. 해결 방안

- Unlabeled set과 labeled set 사이의 정보량 차이에 집중하는 기존의 uncertainty-based heuristic AL* 기법들과 접목 시, 초반 cycle에서 나타나는 성능 향상 폭 저하 문제를 해결 가능할 것으로 예상

→ Entropy 외에 추가적으로 계산된 불확실성 (uncertainty)을 활용한 state 정의 가능

* 전주현, 박재현, 박주원, 조성인, "이미지 분류에서의 액티브 러닝 기술의 동향 분석," 인공지능신호처리 학술대회, 2024, pp. 19-22.

Thank you !

Appendix

- 팀원 별 기여 사항

1. 박윤서
   - DQN 알고리즘 구현
   - 실험 결과 정리 및 그래프 제작
   - PPT 틀 제작 및 README 개선
   - DQN 알고리즘 설명 자료 작성

2. 박주원
   - 주제 제시 및 기본 AL 알고리즘 구현
   - DQN 알고리즘 구현
   - 실험 결과 정리 및 분석
   - README 틀 제작 및 PPT 개선
   - AL 알고리즘 설명 자료 작성