iCaRL (Incremental Classifier and Representation Learning) 성능 향상 연구

이 름 한지혁 지도교수 황원준

연구 배경

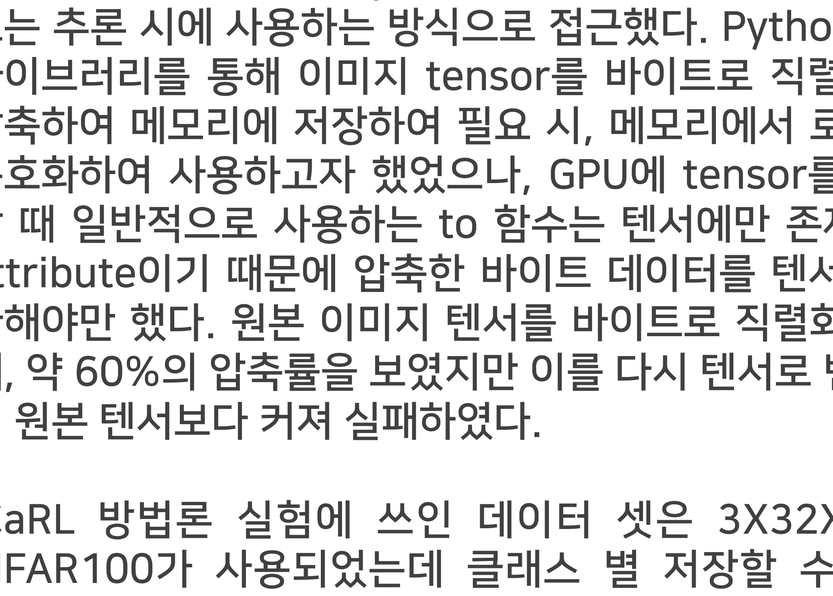
지속적인 학습(Continual Learning)은 인공지능 모델이 새로운 정보를 학습하면서도 기존의 지식을 잃지 않도록 하는 중요한 연구 분야이다. 이 과정에서 '치명적 망각(catastrophic forgetting)' 문제를 해결하는 것이 핵심 과제이다. iCaRL(Incremental Classifier and Representation Learning)은 이러한 문제를 해결하기 위해 제안된 방법으로, 신규 클래스 데이터를 학습하면서도 기존 클래스 데이터를 유지한다. iCaRL 방법론은 치명적 망각과 여러 클래스 분류를 위해, 각 클래스 별로 우수한 이미지인 예제(Exemplar)를 메모리에 저장하여 사용한다. 그러나 iCaRL은 메모리 효율성을 위해 적은 수의 예제를 사용하기 때문에 기존 클래스의 다양성을 충분히 반영하지 못하는 한계가 있으며, 이는 분류 성능 저하로 이어질 수 있다. 본 연구는 iCaRL에서 예제의 개수를 늘려 모델 성능을 향상시키는 방안을 모색하고자 한다.

연구 진행 과정

메모리 자원을 추가하지 않고, Exemplar의 개수를 늘리기 위해서 현재까지 세 가지 방법으로 시도하였다.

1. tensor로 변환된 exemplar를 압축하여 저장했다가 학습 또는 추론 시에 사용하는 방식으로 접근했다. Python zlib 라이브러리를 통해 이미지 tensor를 바이트로 직렬화 후 압축하여 메모리에 저장하여 필요 시, 메모리에서 로드 후 복호화하여 사용하고자 했었으나, GPU에 tensor를 로드할 때 일반적으로 사용하는 to 함수는 텐서에만 존재하는 attribute이기 때문에 압축한 바이트 데이터를 텐서로 변환해야만 했다. 원본 이미지 텐서를 바이트로 직렬화 했을 때, 약 60%의 압축률을 보였지만 이를 다시 텐서로 변환하면 원본 텐서보다 커져 실패하였다.

2. iCaRL 방법론 실험에 쓰인 데이터 셋은 3X32X32인 CIFAR100가 사용되었는데 클래스 별 저장할 수 있는 exemplar의 개수를 늘리기 위해 3X16X16로 크기를 줄여 학습 및 추론을 시도해보았다. 기존 iCaRL 방법론에서는 클래스 별 20개의 exemplar를 보유할 수 있었지만 이미지 크기를 줄임으로써 4배인 80개의 exemplar를 클래스 별로 메모리에 저장할 수 있었다. 학습 후 추론 결과 기존 iCaRL 성능의 약 절반 정도 성능을 보이면서 학습 시, 이미지 해상도의 중요성을 확인할 수 있었다.



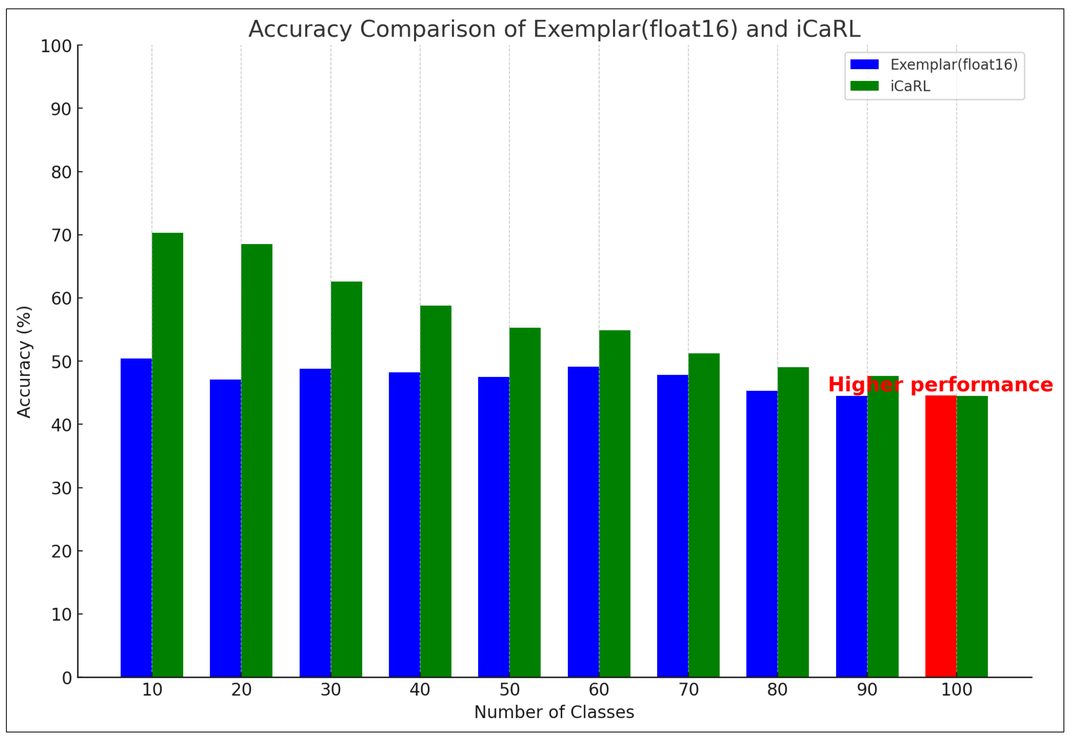
3. Exemplar 이미지 tensor의 데이터 타입은 float32이다. 메모리 공간을 확보하기 위해서 float32를 float16으로 변환한 뒤, 연산 시에 다시 float32로 변환하는 실험을 해보았다. Float32에서 float16으로 변환함으로써 추가 확보할 수 있는 exempla의 개수는 기존 개수의 2배인 40개가 된다. 아래 그림은 iCaRL 방법론에서 여러 클래스를 어떻게 점진적으로 학습하는 지 나타내는 pseudo code인데. 해당 과정에서 사용되는 exemplar 연산 때마다 float16으로 메모리에 저장되어 있는 exemplar를 float32로 변환하여 사용하게 되는 것이다. 해당 방법을 테스트할 때, 모델 학습은 기존 iCaRL과 동일하게 진행하고, exemplar 관련 부분만 수정하였다.

Algorithm 2 iCaRL INCREMENTALTRAIN

input $X^s, \ldots, X^t$ // training examples in per-class sets
input K // memory size
require Θ // current model parameters
require $\mathcal{P} = (P_1, \ldots, P_{s-1})$ // current exemplar sets
$\Theta \leftarrow$ UPDATEREPRESENTATION$(X^s, \ldots, X^t; \mathcal{P}, \Theta)$
$m \leftarrow K/t$ // number of exemplars per class
for $y = 1, \ldots, s-1$ **do**
  $P_y \leftarrow$ REDUCEEXEMPLARSET(P_y, m)
end for
for $y = s, \ldots, t$ **do**
  $P_y \leftarrow$ CONSTRUCTEXEMPLARSET(X_y, m, Θ)
end for
$\mathcal{P} \leftarrow (P_1, \ldots, P_t)$ // new exemplar sets

결과 및 분석

아래 그림은 iCaRL (초록)과 float16(파랑)의 정확도 비교 차트다.
그림과 같이 float16은 기존 iCaRL 보다 10~90개의 클래스를 구별 능력은 낮았으나, 100개의 클래스를 구별할 때, 0.1로 미세하지만 기존 iCaRL 방법론보다 더 나은 성능을 보였다. 또한 지속적인 학습을 통해 구별해야 하는 클래스 개수가 많아지면서 정확도의 감소폭이 큰 iCaRL과는 달리 float16은 상대적으로 완만한 감소율을 보였다. 위와 같은 결과를 통해, 100개의 클래스 보다 더 많은 클래스를 구별할 경우 기존 iCaRL보다 더 나은 성능을 보일 수 있다는 가능성이 존재한다고 생각할 수 있다.



오픈소스 URL

Icarl-pytorch:
https://github.com/lrzpellegrini/icarl-pytorch