result

실험 프레임워크

프레임워크: FACIL

데이터셋: CIFAR-100

모델: ResNet32

시나리오:

- (5/20) → 5개의 태스크, 첫 태스크에서 20클래스 학습 후 나머지 80클래스를 4태스크에 걸쳐 분할 학습
- (5/40-15) → 첫 태스크에서 40클래스 학습 후 60클래스를 4태스크에 걸쳐 분할 학습

- 에폭수: 50 (태스크별)
- 배치크기: 128
- 기본 랜덤시드: 0

그룹별 접근법 및 메모리 사용

1) 그룹 A: 메모리 기반 기법 (num_exemplars_per_class = 20)

해당 알고리즘: BiC, iCaRL, EEIL

- 모두 메모리(Replay Buffer)를 사용하는 전형적인 방법.
- 클래스별 20개씩, 총 2000개(100클래스 × 20) 메모리를 운용.

2) 그룹 B: 정규화 / Knowledge Distillation 계열 기법

해당 알고리즘: LwF, EWC, MAS, PathIntegral, R-Walk, LwM

- KD 또는 Regularization로 Catastrophic Forgetting을 줄이는 접근.
- 메모리 설정: 0 또는 20 (클래스 1개당 메모리 20)
- 즉, num_exemplars_per_class=20(메모리 2000개)인 경우와, num_exemplars_per_class=0(메모리 미사용)인 경우 둘 다 진행
- 이를 통해 메모리 사용 유무에 따른 성능 차이를 비교.

3) 그룹 C: 베이스라인 접근법 (메모리 사용 안 함)

해당 알고리즘: Freezing, Finetuning, Joint

- Freezing: 특정 레이어/파라미터를 동결하며 학습
- Finetuning: 순수하게 이전 태스크 파라미터를 이어받아 미세조정만 하는 방식
- Joint: 모든 데이터를 한 번에 학습(Oracle 시나리오)
- num_exemplars_per_class=0으로, 메모리 없이 진행.

다양한 하이퍼파라미터(λ, T, α, $damping$ 등)를 Grid Search 형태로 실험

1) LwF (Learning without Forgetting)

- 탐색 파라미터
  - λ: 0.5, 1, 2, 3, 5
  - T (Temperature): 2, 3, 4

2) EWC (Elastic Weight Consolidation)

- 탐색 파라미터

    - λ: 500, 1000, 5000, 10000

3) MAS (Memory Aware Synapses)

- 탐색 파라미터
    - λ: 0.5, 1, 2, 3, 5

4) R-Walk

- 탐색 파라미터
    - λ: 1, 2, 3, 5
    - α: 0.3, 0.5, 0.6

5) Path Integral

- 탐색 파라미터
    - λ: 0.05, 0.1, 0.15, 0.2
    - $damping$: 0.05, 0.1, 0.2

6) LwM (Learning without Memorizing)

- 탐색 파라미터
    - β: 1, 2, 3
    - γ: 1, 2

7) iCaRL

- 탐색 파라미터
    - λ: 1, 2, 5

8) BiC

- 탐색 파라미터
    - val_exemplar_percentage: 0.1, 0.2, 0.3
    - λ는 -1 (고정), T=2 (고정)

9) EEIL

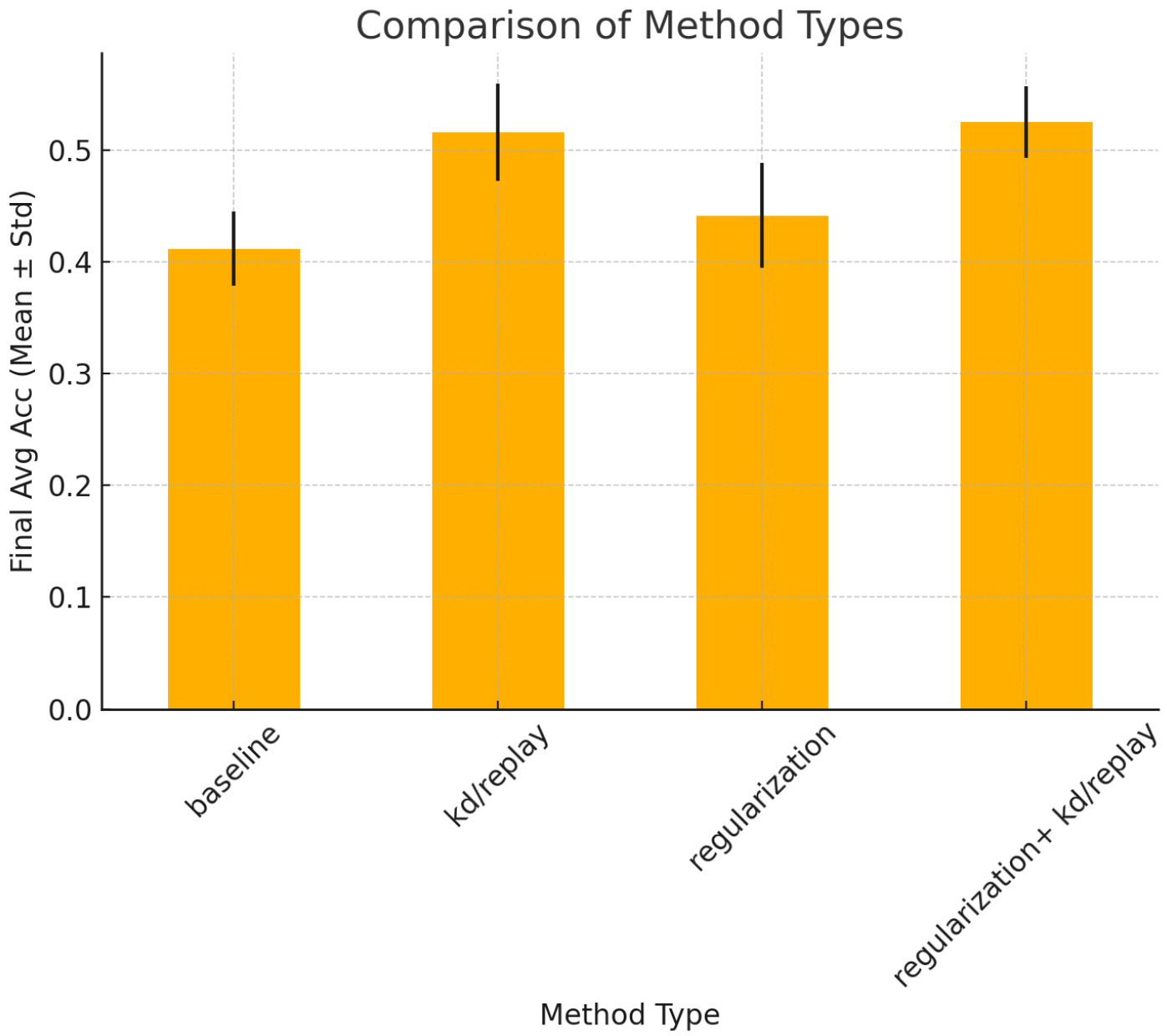
- 탐색 파라미터
    - λ: 1, 2
    - T: 2, 4
    - lr_finetuning_factor: 0.005, 0.01

10) Freezing

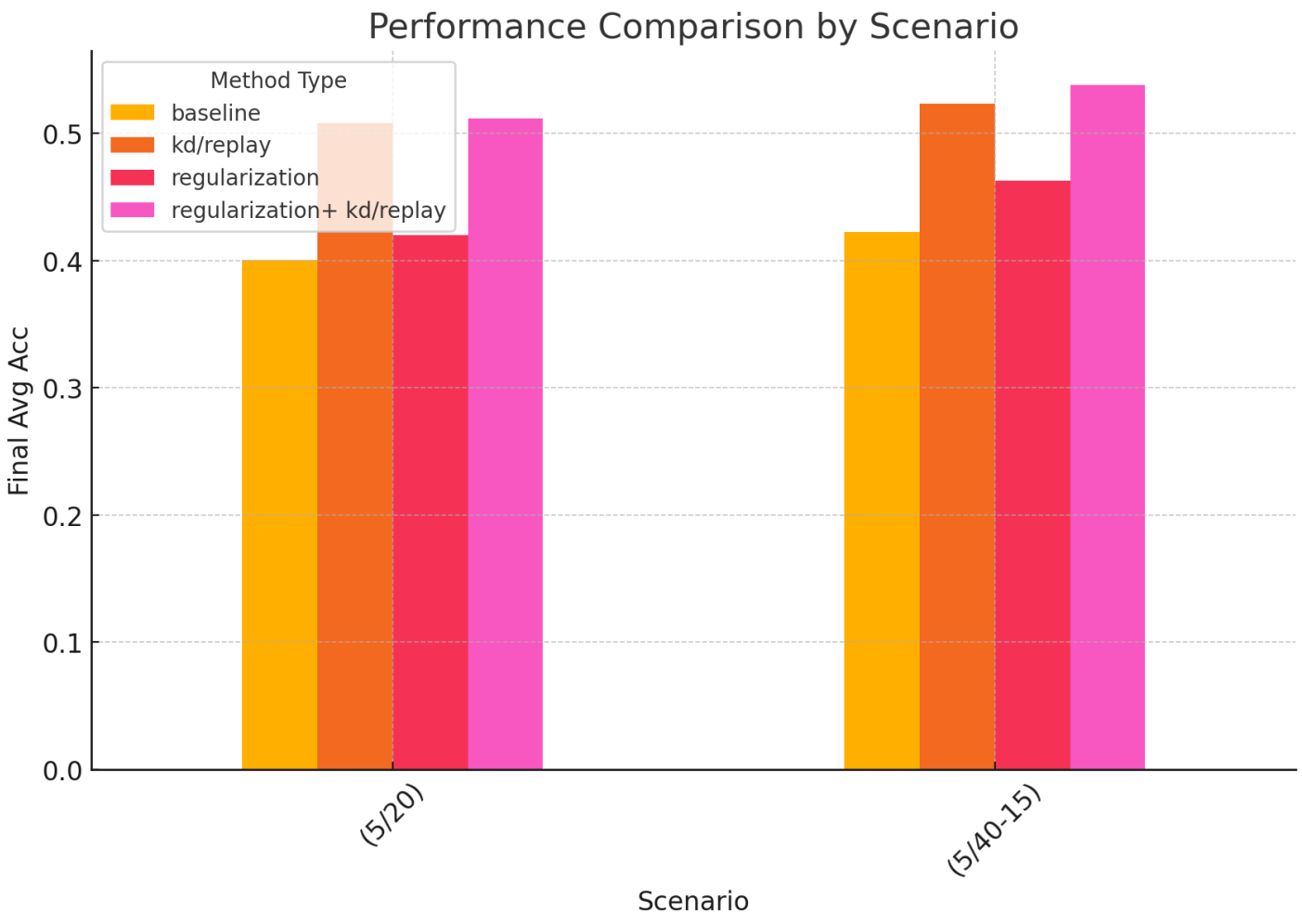
- 탐색 파라미터
    - freeze_after: 1, 2, 3

11) Finetuning

- 탐색 파라미터
    - all_outputs: 0, 1

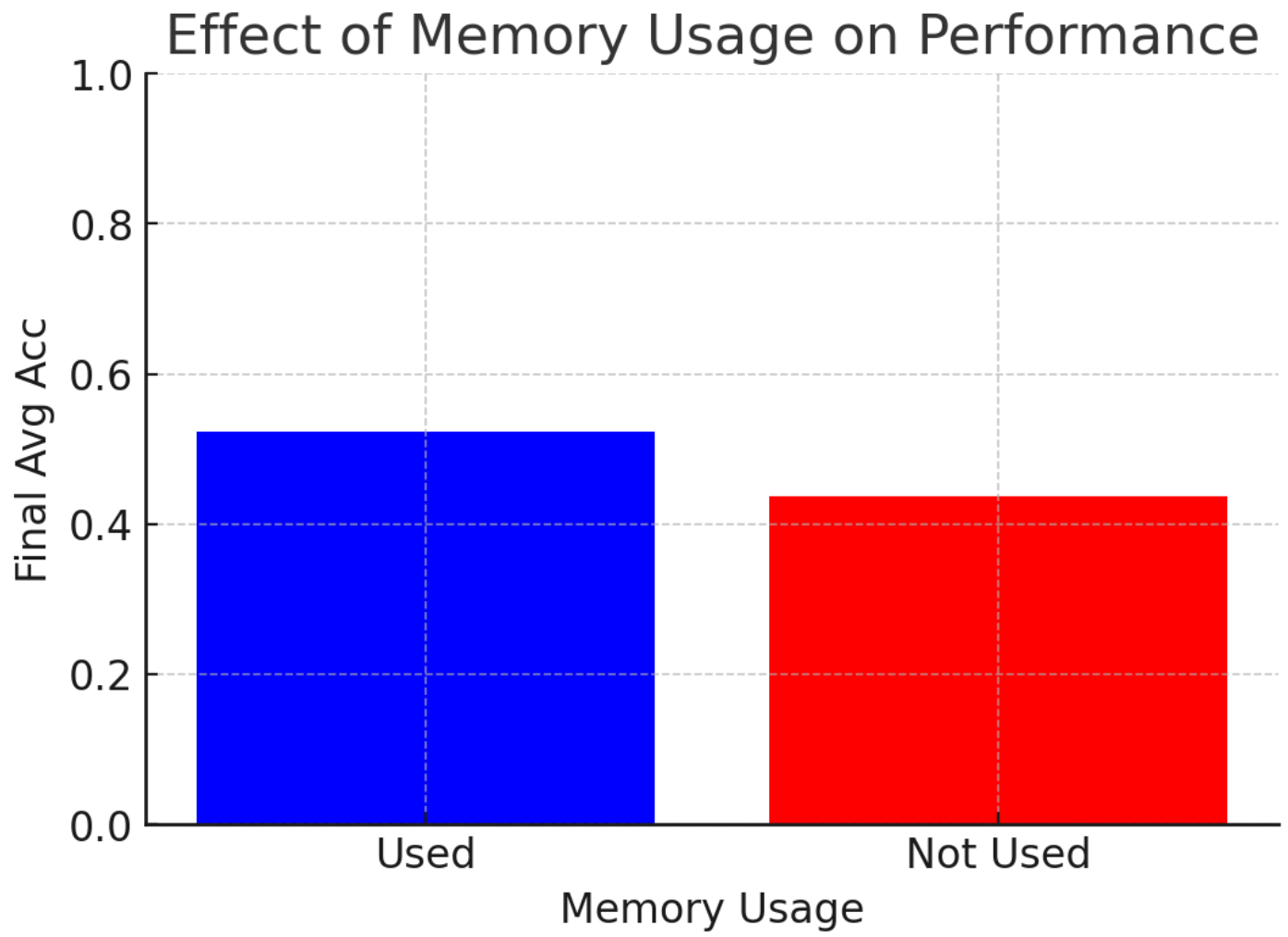


- Baseline: 평균 0.412, 성능이 가장 낮음.
- Regularization: 평균 0.441, baseline보다는 높지만 다른 방법보다 낮음.
- KD/Replay: 평균 0.516, regularization보다 개선된 성능을 보임.
- Regularization + KD/Replay: 평균 0.525, 가장 높은 성능을 보이는 방법.

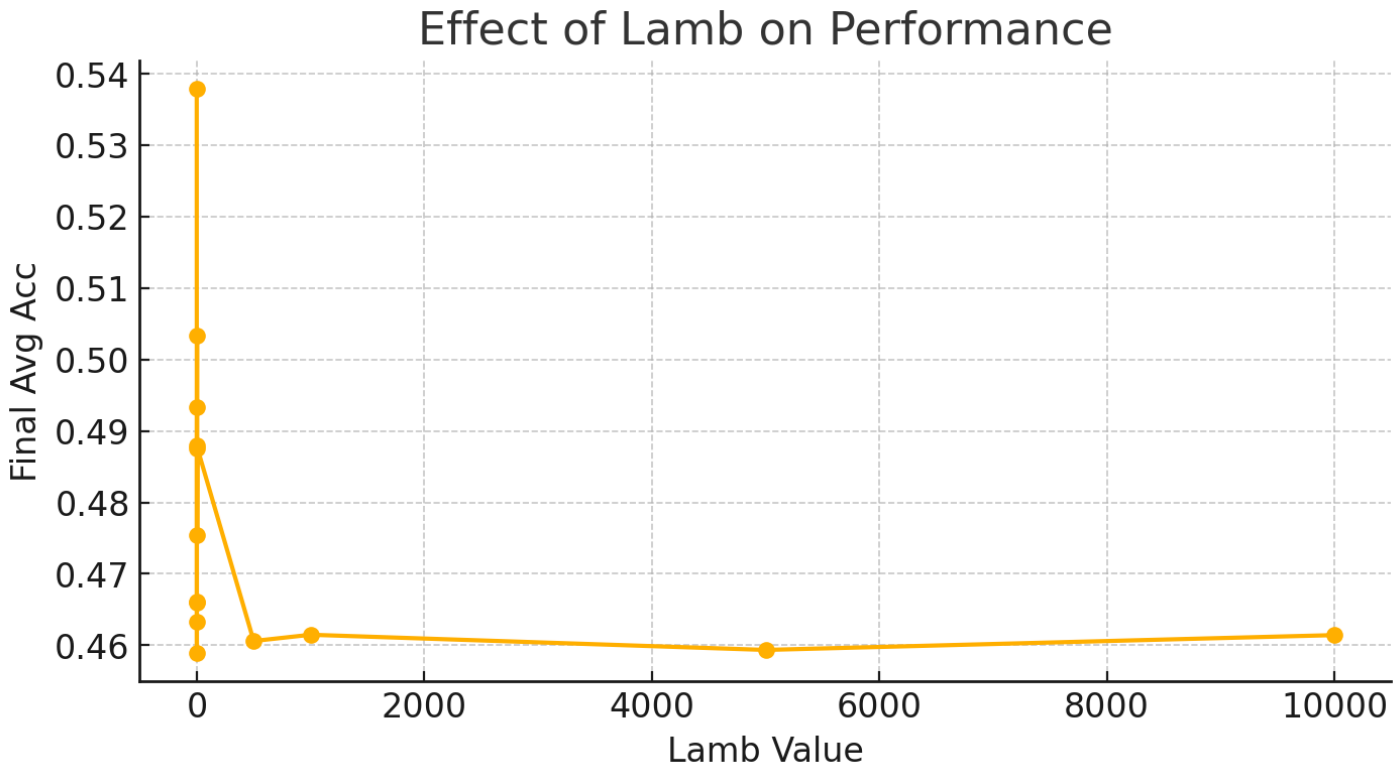


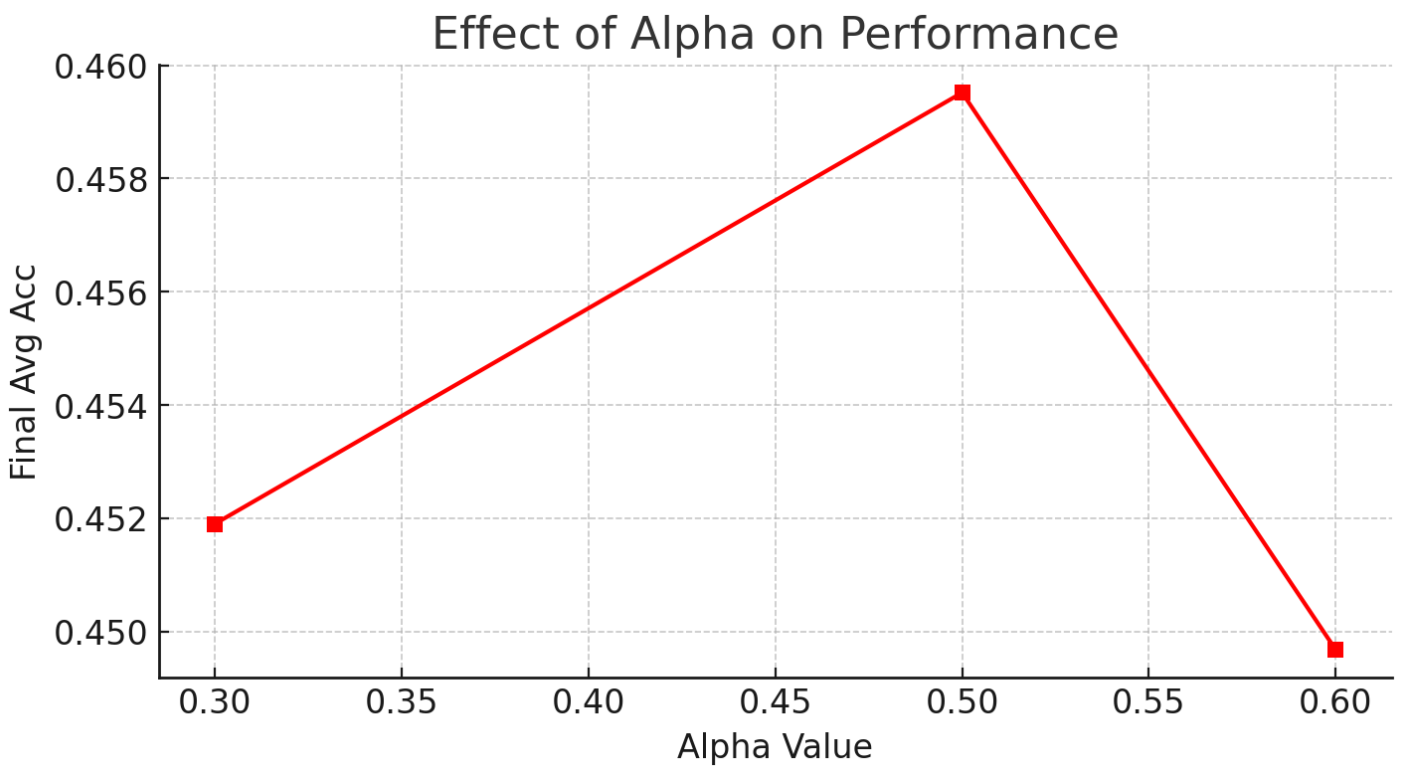
- Baseline: (5/20) 0.400 → (5/40-15) 0.423 (소폭 향상)

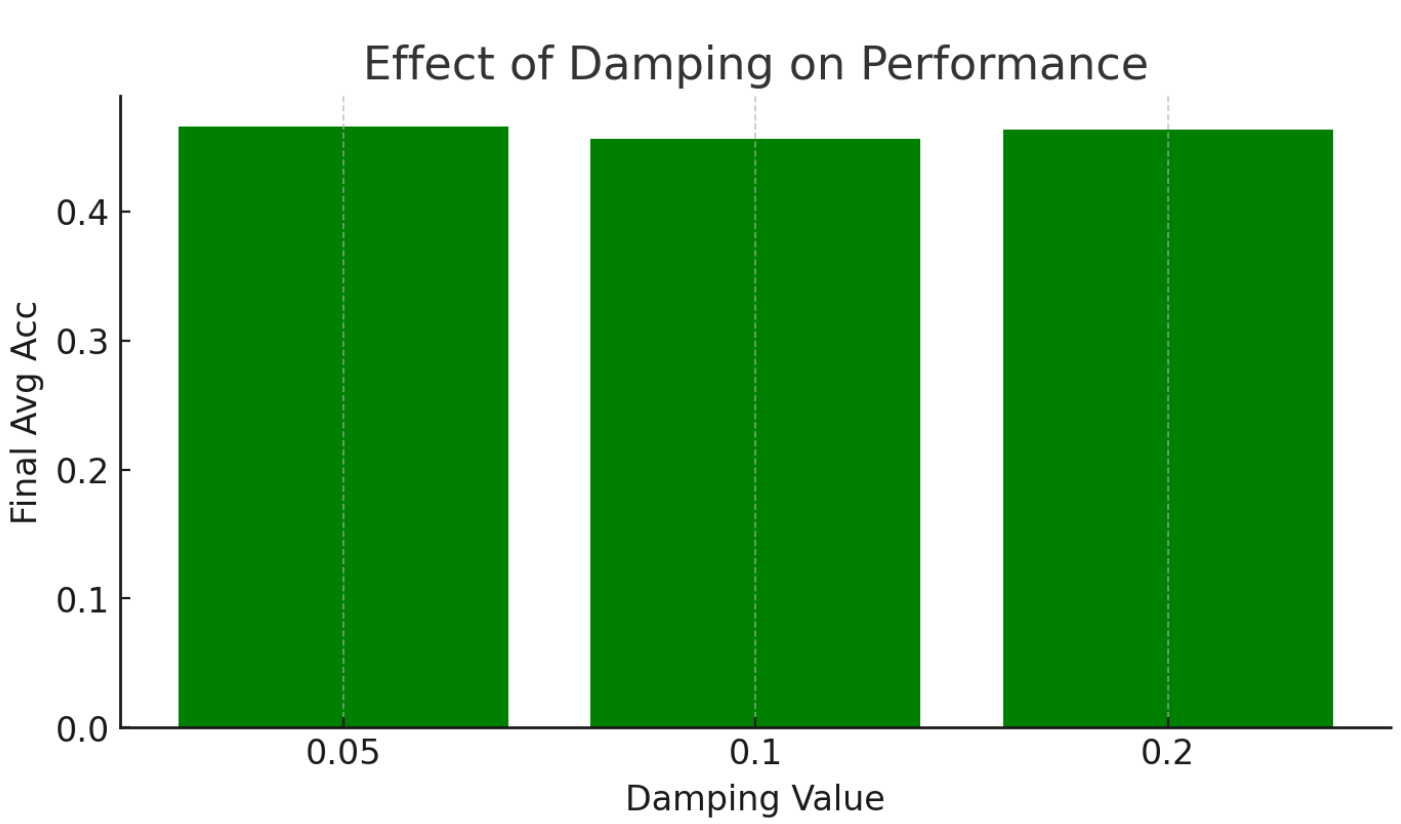
- Regularization: (5/20) 0.420 → (5/40-15) 0.463 (향상)
- KD/Replay: (5/20) 0.508 → (5/40-15) 0.523 (향상)
- Regularization + KD/Replay: (5/20) 0.512 → (5/40-15) 0.538 (가장 높은 성능)



- 메모리 사용 여부(num_ex_per_class > 0 vs num_ex_per_class = 0)에 따른 성능 비교
- 메모리 사용(O): 평균 성능 0.523
- 메모리 미사용(X): 평균 성능 0.437
  - 결과 요약:
    - 메모리를 사용할 경우, 성능이 약 8.5%p 더 높음







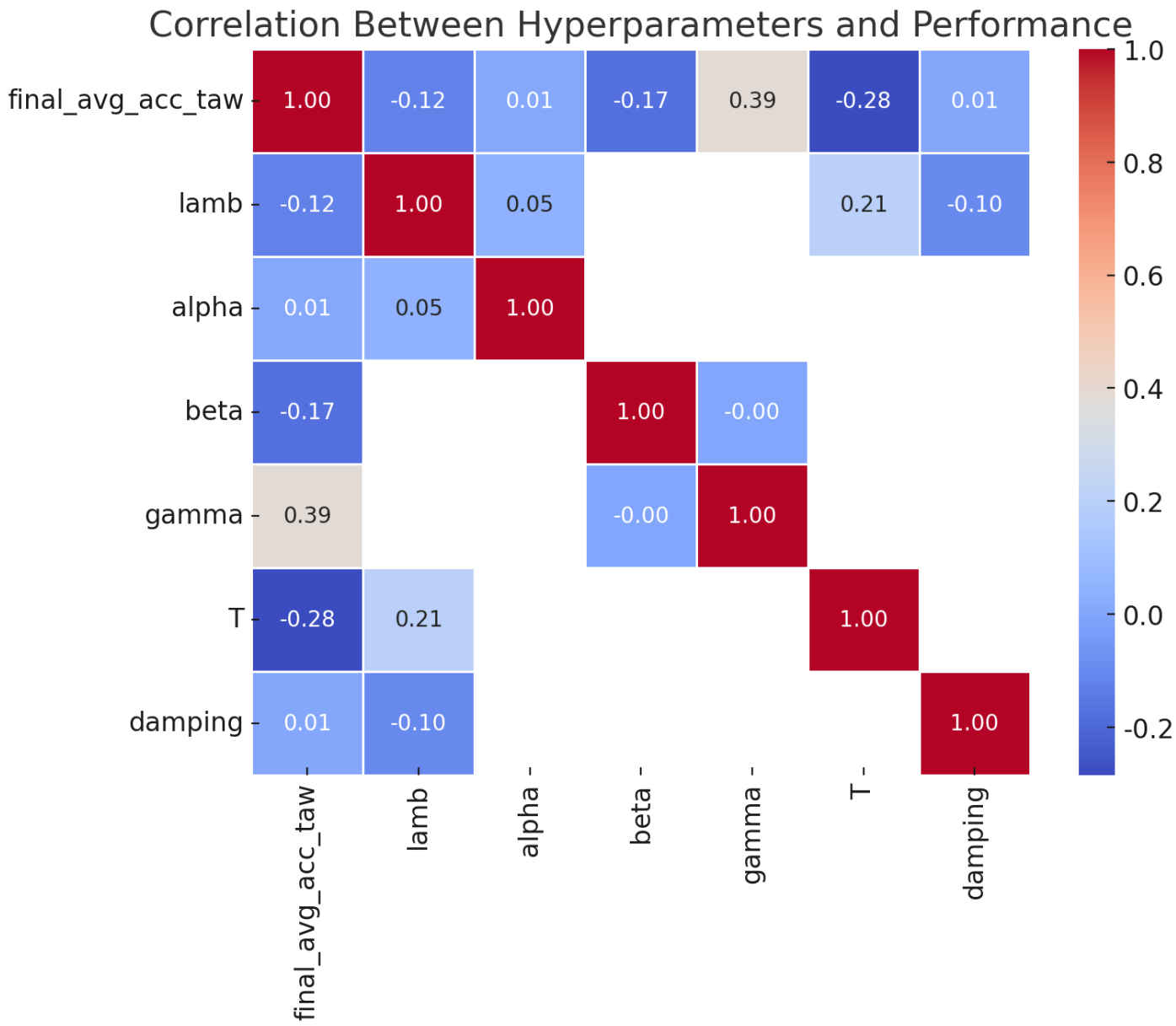
- Lamb 값에 따른 성능 변화
  - Lamb이 -1.0일 때 성능이 가장 높음 (0.538).
  - Lamb이 0.05~0.20일 때는 성능이 비슷한 수준 (0.46~0.47).
  - Lamb 값이 커질수록 성능이 뚜렷하게 증가하거나 감소하는 경향은 보이지 않음.

- Alpha 값에 따른 성능 변화
  - 데이터가 부족하여 alpha에 따른 뚜렷한 패턴은 분석하기 어려움.
  - 추가 데이터가 필요함.

- Beta, Gamma 값에 따른 성능 변화
  - 마찬가지로 데이터가 충분하지 않아 뚜렷한 경향을 파악하기 어려움.

- Damping 값에 따른 성능 변화 (path_integral, r_walk 등)
  - Damping=0.05 → 성능 0.466
  - Damping=0.10 → 성능 0.457

- Damping=0.20 → 성능 0.463
- 즉, damping=0.05가 가장 좋은 성능을 보이며, damping이 증가한다고 해서 성능이 반드시 향상되지는 않음.



- Gamma (γ)와 Final Avg Acc의 상관관계: 0.386
  - Gamma 값이 클수록 성능이 증가하는 경향이 있음.
- Lamb (λ)와 Final Avg Acc의 상관관계: -0.120
  - Lamb이 증가할수록 성능이 약간 감소하는 경향이 있음.
- Beta (β)와 Final Avg Acc의 상관관계: -0.174
  - Beta가 클수록 성능이 다소 감소하는 경향이 있음.
- T와 Final Avg Acc의 상관관계: -0.284
  - T가 증가할수록 성능이 낮아지는 경향을 보임.

결론적으로, 본 실험에서 각 하이퍼파라미터와 최종 성능 간의 상관관계를 살펴본 결과:

1. γ
   - 성능과 뚜렷한 양의 상관이 관찰되어, 값을 높였을 때 더욱 향상된 결과를 기대할 수 있음.
   - 이는 γ가 과거 지식 유지나 KD/Replay 손실을 강조하기 때문
2. λ, β, T
   - 음의 상관을 보여, 값을 지나치게 크게 설정하면 오히려 성능 저하로 이어질 가능성이 높음.
   - Catastrophic Forgetting을 막기 위해 정규화·온도 등 파라미터를 활용하되, 과도한 설정은 새로운 태스크 학습을 제한하거나 분류 경계를 흐리게 만들 수 있음.
3. Damping, α

- 이번 실험 범위 내에서 성능에 유의미한 영향이 확인되지 않았음.
- 즉, Damping이나 α 값을 큰 폭으로 바꿔도 평균 정확도에 미치는 영향이 크지 않았으며, 어느 수준에서든 큰 편차 없이 비슷한 성능을 유지함.

이러한 결과를 종합해 볼 때,

- γ값은 좀 더 높게 설정해 망각 방지 및 모델 성능 개선을 설정할수있으며,
- λ, β, T 등은 적절한 수준에서 조절하여 과도한 규제로 인한 성능 저하를 피하는 것이 좋음
- Damping, α는 실험 환경에 따라 중요도가 다를 수 있으나, 이번 연구 조건에서는 크게 영향을 미치지 않음

- 추가 분석이 필요한 부분
  - Gamma(γ) 값을 높였을 때 정확히 얼마나 성능이 개선되는지 실험적으로 검증.
  - Lambda(λ)와 Beta(β) 값을 적절하게 조정하는 최적의 조합 탐색.

—> 실험 진행중.

1. γ 세분 탐색: γ를 넓은 범위에서 (ex. 1~15) 랜덤/그리드 탐색하며, (10/10) 같은 더 많은 스텝 시나리오로도 검증.
2. $\lambda \times \beta$ 동시 최적화: 2D Heatmap 실험을 통해 어느 조합이 가장 이상적인지 찾기.
3. Bayesian Optimization: 파라미터가 많아지는 만큼 효율적 튜닝 접근이 권장됨.
4. 새로운 시나리오(10태스크, 2~3태스크)로 결과 비교, Class Imbalance나 Task Order 변화도 시도.
5. 추가 지표 도입: 정확도뿐 아니라 BWT, FWT(Forward Transfer), 클래스별 편차 등을 FACIL 로깅에 추가해서 세밀한 성능 분석 진행.