개인연구미팅

2025년 8월 1일 금요일

서울대학교 산업공학과
DSBA 연구실
박사과정 임훈

목차

1 Recap

2 Normalizing Flow(NF)

3 NF into Continual learning

4 Incremental GMM

목차

1 Recap

2 Normalizing Flow(NF)

3 NF into Continual learning

4 Incremental GMM

Sparse Network의 실패

- **기존 연구에 대한 문제점 해결을 위해 각 class별로 별도의 파라미터를 학습하는 Deep Sparse Training을 시도 함**
- 그러나 여러 방법을 시도하였지만 좋은 결과를 얻지 못하였으며, 실패한 방법으로 마무리 하고자 함
- 주요 문제는 다음과 같음

  1. **Task간 지식 단절**
     - Sparse parameter들이 각 task의 지식을 보존한다고 생각했으나, 비활성 파라미터들이 다른 task 학습에 의해 학습되며, 결과적으로 지식은 보존되지 않음
     - 활성 파라미터들이 각 task의 지식을 보존함과 동시에 비활성 파라미터들은 다른 task와의 공통 최적화가 되어야 하지만 그러지 못 함
  2. **높은 학습 비용**
  3. **누더기식 접근 및 시도**
     - 성능 향상을 위해 논리적인 접근이 아닌 무작위적 시도가 다수
     - DST의 구조적 한계를 해결하기 위해 KD를 추가하고, Orthogonal loss를 추가하는 등 의 시도 다수 진행
- **기존에 정의한 문제 및 연구 방향은 이어가되, 접근법 및 구조를 달리하고자 함**

기존 방식의 한계

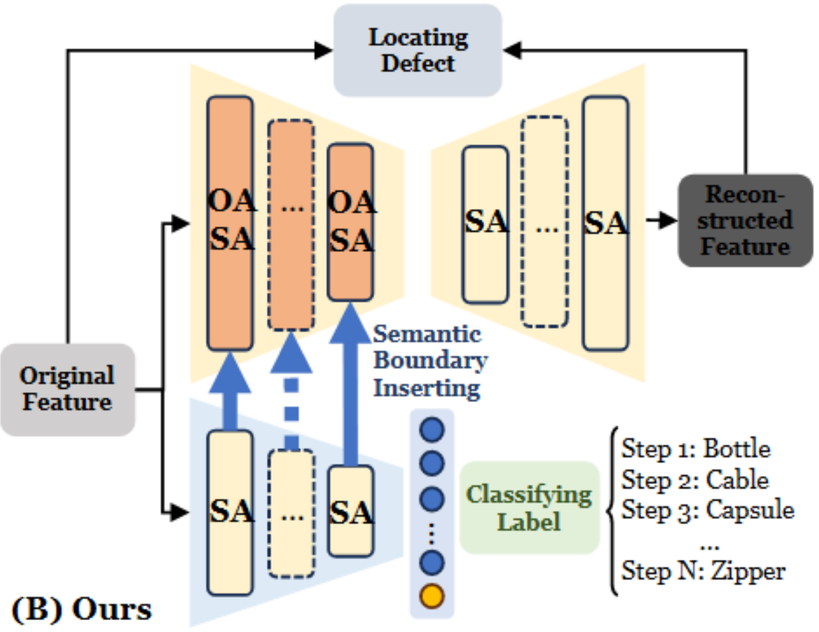
- **기존 연구에 대한 한계와 함께 Reconstruction 방식의 한계를 같이 다루고자 함**

1. 불안정한 학습 및 높은 학습 비용

- 기존 방법론인 IUF는 재구성 네트워크와 판별자를 동시에 학습해야 함
- 판별자와 재구성 네트워크의 동시 최적화를 위해 긴 학습 시간 요구
  - 학습 시 500epoch으로 학습
  - UniAD의 경우 1,000epoch으로 학습

2. 단일 재구성 네트워크의 공유 표현 충돌

- 모든 클래스가 재구성 네트워크의 파라미터를 공유
- 클래스 간 불필요한 간섭과 표현 충돌 발생
- 논문 본문에도 이 문제를 명시적으로 인정
  - 새로운 객체를 학습할 때 이전 객체의 특징 공간을 무차별적으로 업데이트하여 심각한 파국적 망각을 야기

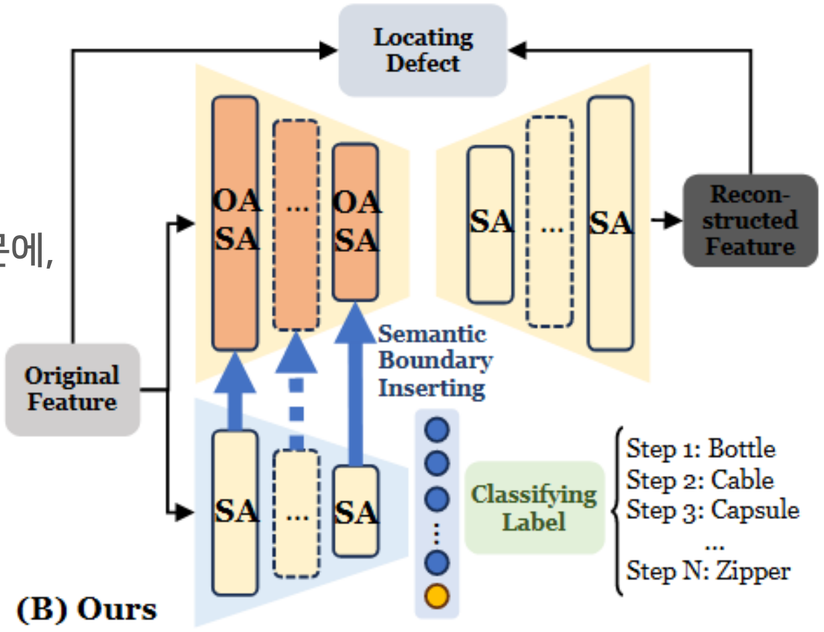


기존 방식의 한계

- **기존 연구에 대한 한계와 함께 Reconstruction 방식의 한계를 같이 다루고자 함**

3. 분리된 모듈의 구조적 비효율성

- 재구성 네트워크 자체는 클래스를 식별하는 능력이 없음
- 그래서 IUF의 경우 클래스 구분을 위해 별도의 분류기를 사용
  - 하지만 이 분류기 마저 전체 클래스의 수를 사전에 알고 있는 것을 가정하고 있기 때문에, 현실에 적합한 CL 방법론이라고 할 수 없음



Reconstruction 기반의 구조가 아닌 다른 방식의 접근이 필요하다고 판단

목차

1 Recap

2 Normalizing Flow(NF)

3 NF into Continual learning

4 Incremental GMM

구조의 전환 : Normalizing Flow

- Reconstruction 방식은 **높은 학습 비용**과 **클래스 간 불필요한 간섭**으로 인한 forgetting이 문제
- **Normalizing Flow(NF)**는 고유한 특성 때문에 좋은 대체재가 될 수 있음

1. 가역성을 통한 강력한 지식 보존 및 간섭 방지

- NF는 가역적 1대1 매핑을 통해 입력 데이터(x)를 잠재 공간(z)에 정보 손실 없이 투영 가능
- 각 클래스가 잠재 공간에서 명확히 분리되므로, 신규 클래스 학습이 기존 클래스의 표현을 침범하는 Task interference를 근본적으로 방지[1]
- 최신 Task로 fine-tuning 하더라도 내부 임베딩/특징 공간의 변형을 직접 규제할 수 있음[2]

2. 내재된 판별 능력과 안정적인 단일 목표 학습[3]

- 클래스별 데이터 분포를 NF를 통해 직접 학습함으로써 입력 데이터에 대한 정확한 우도(likelihood)를 계산할 수 있음
- 이 우도를 기반으로 베이즈 분류 규칙에 따라 입력을 가장 가능성 높은 클래스(및 작업)로 분류 가능
- 별도의 분류기 없이 단일 네트워크로 분류 가능

[1] Pomponi, Jary, Simone Scardapane, and Aurelio Uncini. "Pseudo-rehearsal for continual learning with normalizing flows." arXiv preprint arXiv:2007.02443 (2020).
[2] Pomponi, Jary, Simone Scardapane, and Aurelio Uncini. "Continual learning with invertible generative models." Neural Networks 164 (2023): 606-616.
[3] Kirichenko, Polina, et al. "Task-agnostic continual learning with hybrid probabilistic models." arXiv preprint arXiv:2106.12772 (2021).

구조의 전환 : Normalizing Flow

- Reconstruction 방식은 **높은 학습 비용**과 **클래스 간 불필요한 간섭**으로 인한 forgetting이 문제
- **Normalizing Flow(NF)**는 고유한 특성 때문에 좋은 대체재가 될 수 있음

3. 안정적인 MLE 훈련 [2,3]

- NF는 순수 likelihood 최적화이므로 GAN의 학습 불안정, 모드 붕괴 등에 대한 우려 x
- NF의 높은 학습 안정성은 빠른 학습 수렴과 낮은 forgetting과 직결

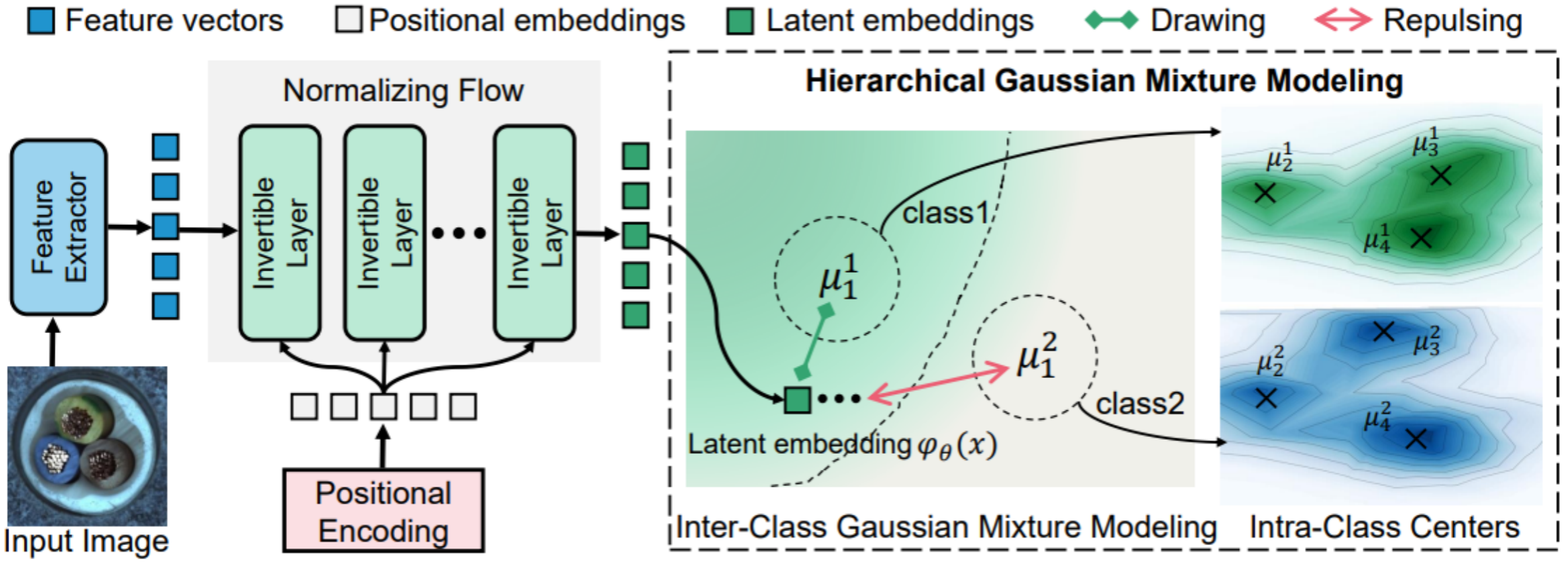
3. 효율적인 연산 및 저비용 구조

- FIM, SVD와 같은 높은 고비용 연산 모듈 불필요

[1] Pomponi, Jary, Simone Scardapane, and Aurelio Uncini. "Pseudo-rehearsal for continual learning with normalizing flows." arXiv preprint arXiv:2007.02443 (2020).
[2] Pomponi, Jary, Simone Scardapane, and Aurelio Uncini. "Continual learning with invertible generative models." Neural Networks 164 (2023): 606-616.
[3] Kirichenko, Polina, et al. "Task-agnostic continual learning with hybrid probabilistic models." arXiv preprint arXiv:2106.12772 (2021).

구조의 전환 : Normalizing Flow

- 기존의 NF 기반의 AD를 바로 Continual learning에 접목하기 전에, NF 기반의 Multi-class AD 탐색



- HGAD : Hierarchical gaussian mixture normalizing flow modeling for unified anomaly detection[4]
  - 다중 클래스를 단일 가우시안이 아닌 클래스별 중심을 갖는 가우시안 혼합 모델(GMM)로 모델링
  - 클래스 센터 간의 분별력을 높이기 위해 상호 정보량 최대화(MIM) 손실을 도입하여 잠재 공간을 명확하게 구조화
  - 단일 클래스 내의 다양한 정상 패턴을 포착하고자 클래스별로 여러 개의 하위 중심을 두는 계층적 접근 방식을 사용

Contrastive learning과 유사

[4] Yao, Xincheng, et al. "Hierarchical gaussian mixture normalizing flow modeling for unified anomaly detection." European Conference on Computer Vision. Cham: Springer Nature Switzerland, 2024.

목차

1 Recap

2 Normalizing Flow(NF)

3 NF into Continual learning

4 Incremental GMM

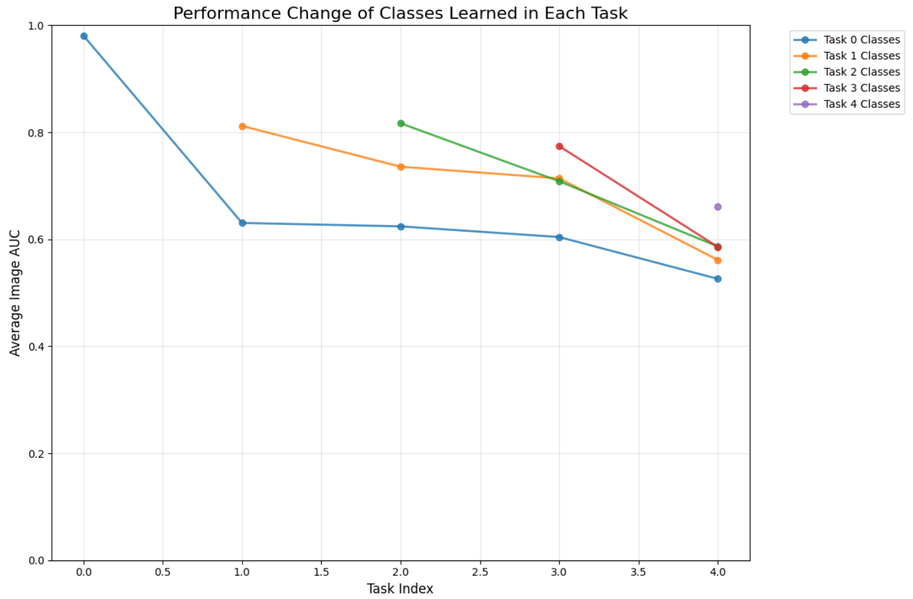
NF into Continual Learning

- **NF 기반의 Multi-class AD의 HGAD를 Continual learning으로 실험 진행**
  - 실험 목적
    - Continual learning으로 학습 시 성능이 저하되는 것은 자명함
    - 그러나 구체적으로 성능이 저하되는 원인을 탐색하고자 함
  - 실험 방법
    - 별도의 장치 x. 각 Task 학습 시 클래스 중심점만 새로 생성하도록 함
    - 시나리오 : 15개의 클래스를 3개씩 5개의 task로 학습

1. Continual learning 학습 결과

Task	Final Image AUROC	Final Pixel AUROC
0	0.44	0.34
1	0.68	0.86
2	0.59	0.83
3	0.54	0.82
4	0.97	0.93
Average	0.644	0.756

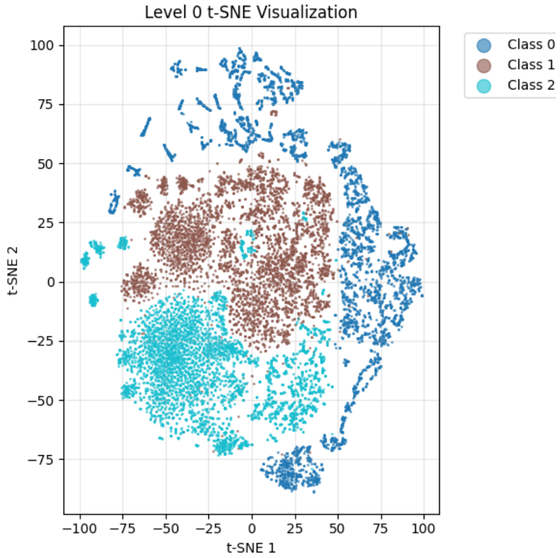
[최종 학습 결과]



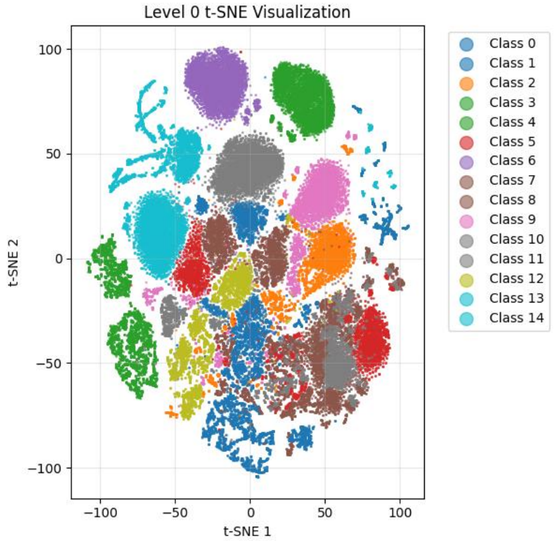
[Task에 따른 성능 변화]

- Multi-class AD임에도 불구하고 NF 기반의 방법론을 바로 Continual learning으로 학습 시 성능은 저하 됨
- Task 별 성능 변화 그래프를 통해 Catastrophic forgetting이 발생되는 것 확인
- MIM 손실은 현재 클래스 데이터만으로 '밀고 당기기'를 계산하므로 **과거 클래스의 잠재 공간 표현을 유지시킬 힘이 없음**.
- 이로 인해 새로운 클래스 학습 과정에서 **과거 클래스의 중심점이 의도치 않게 밀려나거나 겹치며** 성능이 저하

2. 임베딩 및 클래스 중심점 간 거리 시각화

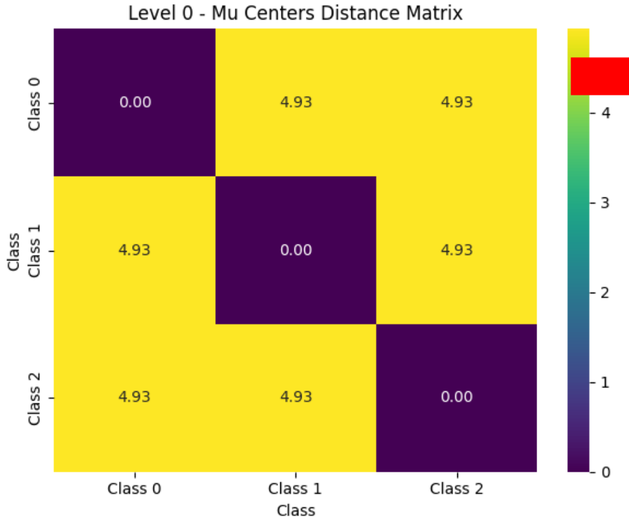


[Task 1 학습 후]

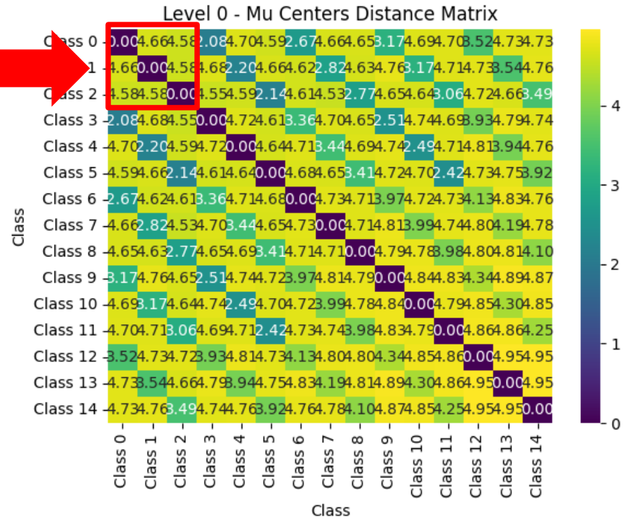


[Task 5 학습 후]

[Task1의 embedding 시각화 결과]



[Task 1 학습 후]



[Task 5 학습 후]

[클래스 중심점 간 거리 행렬]

- Task 1 학습 직후 Task1의 임베딩들은 잘 분리 되어 있음. 그러나 Task 5 학습 한 후에는 다른 class들과 뒤섞임
- 거리 행렬에서 보이듯 시각적으로 유사한 클래스(예: 1과 2, 3과 4)의 클래스 중심점은 서로 이끌려 변화 함
- 새로운 클래스 학습 과정에서 과거 클래스의 중심점이 의도치 않게 밀려나는 현상을 간접적으로 보여줌

3. 새로운 Task를 위한 확장 전후 Task1 성능 비교

- Task 1의 데이터[Bottle, Cable, Capsule]들에 대해 학습 직후, 그리고 Task2를 위한 확장 직후 결과를 비교
- NF는 전혀 변하지 않았지만, 새로운 중심점이 생성된 것 만으로도 성능이 저하 됨

	Image AUROC			Pixel AUROC		
	Bottle	**Cable**	**Capsule**	**Bottle**	**Cable**	**Capsule**
Task 1 학습 직후	1	0.946	0.979	0.985	0.859	0.99
Task 2 확장 직후	0.998	0.719	0.837	0.956	0.722	0.950

- NF는 가역성과 1대1 매핑이라는 특성 덕분에 forgetting에 매우 강력함
- 따라서 성능이 저하되는 것은 실제 지식이 손실된 것이 아니라 **GMM의 정렬이 잘못된 것은 아닌가 추측**

4. Few-shot training

- Task 1의 데이터들에 대해 Task 1, Task 2 학습이 끝난 뒤 각각 성능을 평가
- 그리고 Task1의 데이터에 대해 추가적으로 1 step 학습 후 평가 진행

	Image AUROC			Pixel AUROC		
	Bottle	**Cable**	**Capsule**	**Bottle**	**Cable**	**Capsule**
Task 1 학습 후	1	0.946	0.979	0.985	0.859	0.99
Task 2 학습 후	**0.654**	**0.519**	**0.491**	**0.927**	**0.586**	**0.657**
Few-shot training	**0.998**	**0.719**	**0.837**	**0.956**	**0.722**	**0.950**

- Task 2 학습 후 Task 1의 성능은 매우 낮게 나타남
- 그러나 Task 1의 데이터로 약간의 학습을 해주게 되면 바로 학습 직후에 준하는 성능으로 복구 됨
- 실제 지식이 forgetting이 된 것이 아닌, GMM의 정렬이 잘못되어 성능이 저하되는 것으로 추측
- 일종의 spurious forgetting[5]이라고 볼 수 있음
  - 실제 지식이 소실된 것이 아니라, 과거 task에 대한 정렬이 손실되었기 때문에 성능이 저하되는 것

[5] Zheng, Junhao, et al. "Spurious forgetting in continual learning of language models." arXiv preprint arXiv:2501.13453 (2025).

5. 기존 CL 방법론 적용 실험 결과

- HGAD에 기존의 Continual learning 방법론을 적용하여 실험 진행

	Image-AUROC	Pixel-AUROC	Image-Forgetting	Pixel-Forgetting
HGAD	0.646	0.753	-0.334	-0.207
+EWC	0.520	0.797	-0.167	-0.066
+LWF	0.646	0.750	-0.333	-0.209
+PackNet	0.647	0.757	-0.332	-0.206
+Replay	**0.952**	**0.975**	**-0.030**	**-0.002**

- 대부분의 방법은 NF 자체에 Regularization을 가하는 방법이기 때문에 성능 차이가 크게 발생하지 않음
- 그러나 Replay의 경우 과거 Task의 데이터를 저장한 뒤 다시 학습에 사용하는 방법이기 때문에 GMM의 클래스 중심에 영향을 줌
  - 따라서 최종 결과가 multi-class에 준하는 성능이 나옴

Summary

- ✓ Reconstruction 기반의 구조적 문제를 극복하고자 Normalizing Flow 기반의 방법을 채택
- ✓ Normalizing Flow 기반의 AD를 Continual Learning에 적용하기 위해 multi-class AD 방법론인 HGAD를 베이스라인으로 채택

1. Continual learning 학습 결과

- Continual learning으로 학습 시 성능 저하
- 과거 클래스의 잠재 공간 표현을 유지시킬 힘이 없어, 새로운 클래스 학습 과정에서 과거 클래스의 중심점이 의도치 않게 밀려나며 성능이 저하

2. 임베딩 및 클래스 중심점 간 거리 시각화

- 새로운 클래스 학습 과정에서 과거 클래스의 중심점과 임베딩이 변하는 것을 간접적으로 확인

3. 새로운 Task를 위한 확장 전후 Task1 성능 비교

- 학습 없이 단순히 클래스 중심 점 확장 후 성능 저하 발생
- 따라서 성능 저하는 실제 지식이 손실되는 것이 아닌, GMM의 정렬이 잘못된 것으로 추측

4. Few-shot training

- Task2 학습 후 Task 1에 대해 few-shot 학습 시 성능이 복원 되는 것 확인
- 이는 실제 지식 손실이 아닌 GMM 정렬 문제로 인한 성능 저하라는 주장을 더욱 확실하게 뒷받침

5. 기존 CL 방법론 적용 실험 결과

- NF 자체를 규제하는 다른 방법들은 성능이 낮았던 반면, GMM에 영향을 줄 수 있는 replay는 multi-class에 준하는 높은 성능이 나옴

✓ **Continual learning 시 성능이 저하되는 것은 NF에서 실제 지식이 손실된 것이 아닌 GMM의 정렬이 잘못되어 나타나는 spurious forgetting 때문**

✓ **GMM의 구조적 개선을 통해 GMM을 Incremental 하게 학습 후 Multi-class GMM에 근사되도록 하는 방법 필요**

목차

1 Recap

2 Normalizing Flow(NF)

3 NF into Continual learning

4 Incremental GMM

Motivation

- ✓ NF의 구조는 유지한 채 GMM의 구조적 개선이 필요
- ✓ Incremental하게 GMM을 학습하는 방법을 활용해보자

- **Incremental Gaussian Mixture Model**
  - ✓ 순차적으로 입력되는 데이터들에 대해 학습하고 최종적으로 GMM으로 근사하는 방법
  - ✓ 관련 연구
    - ✓ Ahmad, Waseem. "Incremental learning of Gaussian mixture models." (2006).
    - ✓ Cappé, Olivier, and Eric Moulines. "On-line expectation–maximization algorithm for latent data models." Journal of the Royal Statistical Society Series B: Statistical Methodology 71.3 (2009): 593-613.
    - ✓ Engel P, Heinen M. Incremental learning of multivariate gaussian mixture models. Advances in Artificial IntelligenceSBIA 2010. 2011;p. 82-91.
    - ✓ Heinen, Milton Roberto, Paulo Martins Engel, and Rafael C. Pinto. "IGMN: An incremental gaussian mixture network that learns instantaneously from data flows." Proc VIII Encontro Nacional de Inteligência Artificial (ENIA2011) 14 (2011).
    - ✓ Heinen, Milton Roberto, Paulo Martins Engel, and Rafael C. Pinto. "Using a Gaussian mixture neural network for incremental learning and robotics." The 2012 international joint conference on neural networks (IJCNN). IEEE, 2012.
    - ✓ Pinto, Rafael, and Paulo Engel. "Scalable and incremental learning of gaussian mixture models." arXiv preprint arXiv:1701.03940 (2017).

Ideation

✓ 하지만 이러한 방법을 HGAD에 적용하는 것은 한계가 있음

- data sample 단위로 업데이트 하며, mini-batch 업데이트가 아님
- 통계 기반 방법으로, Gradient 기반으로 업데이트가 불가 함
  - 기존의 Hierarchical GMM은 NF와 각 클래스의 중심 양방향으로 gradient가 흐름

[5] Ahmad, Waseem. "Incremental learning of Gaussian mixture models." (2006).
[6] Pinto, Rafael, and Paulo Engel. "Scalable and incremental learning of gaussian mixture models." arXiv preprint arXiv:1701.03940 (2017).