교수님 연구미팅 - method

태그	
날짜	@2025년 10월 16일
상태	시작 전
실험 제목	

Tl;dr

기존 연구 + 배경

- 기존 Continual AD 방법론 중 대다수는 reconstruction 기반의 방법론
- 다만 이러한 방법론들은 높은 학습 cost를 가지고 있고, reconstruction 구조에서 continual learning을 가능하게 하기 위해 순차적으로 학습되는 class 별 특징을 식별 및 학습하기 위해 별도의 장치를 사용하곤 했음
    - 대표적으로 IUF가 있으며, 별도의 discriminator를 통해 class 식별 특징을 학습하고자 했으나, 불안정하고 reconstruction network와의 표현 충돌 문제가 존재
- 이러한 문제를 해결하고자 기존과는 다른 방식의 구조가 필요함을 느꼈으며, 대체재로 NF 기반의 이상 탐지 채택

구조의 전환 : NF

- NF는 가역성을 통한 강력한 지식 보존에 강점이 있으며, MLE를 통해 안정적으로 학습할 수 있음
- 이를 위한 베이스라인으로 HGAD를 사용. HGAD는 Multi-class AD를 위한 방법으로, 다중 클래스를 NF와 GMM을 통해 모델링 하는 방식

NFAD + CL 의 문제점

- 실험 결과 continual learning 시 성능이 저하되는 것은 NF가 knowledge를 소실했다기 보다는 GMM과의 alignment가 무너지기 때문
- 새로운 task를 학습하는 과정에서 과거 task에 대한 NF와 GMM 간의 alignment가 어긋남
    - NF의 latent를 통해 t-SNE 시각화를 보면 반면 task가 진행되더라도 클래스 별 분리는 잘 되는 것을 확인할 수 있음 → knowledge는 소실 x
- 기존 CL 방법론들을 사용하더라도 성능은 향상되지 않음 → misalignment를 위한 방법이 아니므로
    - 반면 유일하게 replay를 사용하는 경우 높은 성능이 나오는 것을 확인 → buffer를 통해 간간히 과거 task의 alignment를 복원하기 때문

Pilot Experiments

1. 파라미터 변화 비교

실험 목표

연속 학습(Continual Learning)이 진행됨에 따라 모델의 어떤 파라미터 그룹(Normalizing Flow vs. GMM)이 가장 큰 변화를 보이는지 정량적으로 분석하여, 성능 저하의 주된 원인을 규명하고자 함.

주요 실험 결과

1. **Normalizing Flow (NF) 파라미터 변화 분석**:
    - **변화량의 분포**: NF 내부 파라미터의 변화는 특정 블록(Block 0, 1, 2)이나 특정 레이어에 집중되지 않고, **NF 아키텍처 전반에 걸쳐 비교적 고르게 분산**되어 나타났습니다. (첫 번째 이미지의 히트맵 및 라인 차트 참고)
    - **변화량의 추세**: 직전 Task와 비교했을 때, 파라미터의 변화는 **초기 Task(Task 1→2)에서 가장 크고, 이후 Task가 진행될수록 점차 안정화**되는 양상을 보였습니다. (첫 번째 이미지의 'Average NF Layer Changes' 그래프 참고)
2. **GMM 관련 파라미터 변화 분석**:
    - **핵심 변화 요인 식별**: 여러 파라미터 그룹의 변화량을 정규화하여 비교한 결과, `phi_inters` **(GMM의 클래스 간 관계 파라미터)가 다른 모든 파라미터 그룹에 비해 압도적으로 큰 변화**를 보였습니다.
    - `phi_intras`, `mus` 등 다른 파라미터들은 `phi_inters`에 비해 거의 변화가 없는 수준이었습니다.
    - 이러한 급격한 변화 역시 **Task 초반부(Task 1→2→3)에 집중**되었습니다.

결론

실험 결과를 종합하면, 연속 학습 환경에서 발생하는 **성능 저하("망각")의 주된 원인은 NF의 표현력 자체가 붕괴되기 때문이 아닙니다.** NF 내부 파라미터들은 전반적으로 안정적인 변화를 보였습니다.

대신, **모델의 불안정성을 주도한 것은 GMM의 클래스 중심 관계를 정의하는** `phi_inters` **파라미터의 급격한 변동**이었습니다.

따라서, 우리가 겪고 있는 문제는 NF가 지식을 잃어버리는 전통적인 '파국적 망각'이라기보다, NF의 표현 공간과 GMM의 클래스 구조 간의 **'정렬 이동(Alignment Drift)'**으로 정의할 수 있습니다. 이는 우리가 기존에 세웠던 'Spurious Forgetting' 가설을 강력하게 뒷받침하는 실험적 증거

2. Freeze 후 비교

- NF(Normalizing Flow)와 GMM(Gaussian Mixture Model)의 파라미터를 각각 업데이트하거나 고정(Freeze)하는 조합을 통해, 어떤 요소가 모델의 불안정성을 유발하는지 분석
- **정상 데이터에 대한 확신 증가**: 모든 실험 조건에서 정상 데이터의 엔트로피 변화량(ΔHnormal)은 음수(-) 값을 보였습니다. 이는 모델이 학습을 진행하며 **정상(Normal) 샘플에 대해서는 더 확신을 갖게 됨**을 의미합니다.

- **GMM 업데이트의 영향**: GMM 파라미터를 업데이트한 조건들(`NF_update_GMM_update`, `NF_freeze_GMM_update`)에서 **비정상(Anomaly) 데이터의 엔트로피 변동 폭이 크게 나타났습니다.** 이는 GMM 업데이트가 이상 탐지 경계의 불안정성을 유발하는 주된 원천임을 시사

방법론

- 기존의 HGAD 모델은 CNN encoder로부터 multi-level feature를 추출하고, 각 레벨별 feature를 Normalizing Flow(NF)를 통해 latent space로 사상한 뒤, 해당 latent 분포를 Gaussian Mixture Model(GMM)으로 근사하여 이상(anomaly)을 탐지하는 구조
- 이 접근법은 single-task 환경에서는 유효하지만, continual learning 환경에서는 다음의 근본적 문제가 발생

현재 방법론의 구조적 문제

근본적 문제: "NF–GMM 간 확률 공간 불일치

1. **NF와 GMM은 서로 다른 확률적 가정 하에 작동**
   - **현상**: NF와 GMM이라는 두 개의 독립된 확률 모델을 억지로 결합하면서 발생하는 문제
   - NF는 비선형 bijective mapping을 통해 입력 feature의 밀도를 정규화된 continuous density로 변환
   - GMM은 이 latent를 Euclidean 공간 상의 Gaussian mixture로 근사한다.→ NF의 Jacobian 변환으로 인해 latent space는 비유클리드(Riemannian) metric을 가지게 되는데, GMM은 이를 고려하지 않고 isotropic Gaussian으로 근사

     ⇒ latent geometry와 GMM 가정 간 불일치 발생
2. **Continual learning 시 불일치 심화**
   a. 새로운 task가 도입될 때 NF는 기존 latent manifold를 유지하려 하지만, GMM은 새로운 class를 추가하기 위해 mixture component(μ, Σ)를 재배치
   b. 이 과정에서 기존 class의 gaussian 중심이 변형되어 기존 NF latent 구조와 misaligned 됨
   c. 결과적으로 이전 task의 log-likelihood 분포가 변형되고, normal sample의 uncertainty가 급상승하는 misalignment 문제 발생
3. **확률 공간의 문제: Latent Metric 불일치**
   - **현상**: NF의 변환으로 인해 왜곡된 잠재 공간의 '거리 척도'와 GMM이 가정하는 '거리 척도'가 달라 발생하는 문제
   - **원인**: NF는 **리만 기하학(Riemannian geometry)**을 따르는 잠재 공간을 만들지만, GMM은 **유클리드 기하학(Euclidean geometry)**을 가정하고 확률을 계산
   - **결과**: 확률 계산에 체계적인 오류(systematic error)가 발생하여 클래스 경계가 왜곡되고, 이는 연속 학습이 진행될수록 누적되어 문제를 악화시킴

4. **클래스 추가 시 GMM 재정렬 문제:**새로운 class 추가 시 기존 GMM component들의 평균과 분산이 변형되어 기존 클래스의 likelihood 분포가 왜곡된다.
    a. NF와 GMM이 분리되어 학습되는 구조에서는 새로운 task(또는 class)를 학습할 때 NF의 파라미터가 이전 클래스의 latent representation 분포를 변경하게 된다. 그러나 GMM은 과거에 고정된 통계량(mean, covariance)에 의존하므로 이전 클래스의 likelihood 중심이 drift하게 된다. 즉, continual learning에서는 다음과 같은 현상이 반복된다: $p_t(x \mid y_{old}) \neq p_{t-1}(x \mid y_{old})$ p t (x ㅣ y old) ¨I =p t−1 (x ㅣ y old) 이는 catastrophic forgetting의 확률적 형태로, 이전 클래스에 대한 분포 표현이 새로 학습된 클래스의 영향으로 붕괴되어 정상/비정상 구분 경계가 무너진다.
5. **학습 안정성 저하:** NF는 latent space를 “Gaussianize”하려 하고, GMM은 정규분포 mixture로 이를 재근사하려 하여 loss surface가 상충하며 log-determinant(∂f/∂x) 폭발을 유발
    a. NF는 log-determinant term을 통해 density를 정규화하지만, continual learning에서는 task별 feature scale과 분포 범위가 달라진다. 이때 log|det J| 항이 과도하게 커지면 Jacobian 폭발(Jacobian explosion)이 발생하고, 학습이 불안정해진다. 기존 HGAD는 logdet의 magnitude를 제어할 구조적 제약이 없어 task 전환 시마다 NF가 재수렴(convergence)을 반복해야 했다.

방법론 개선 motivation

- 기존 방법론은 두 개의 확률 모델을 이용해 gaussian mixture 조건부 확률 분포를 학습하는 것을 목표로 함
- 그러나 서로 다른 매커니즘 기반의 확률 모델을 연결했기 때문에 continual learning 과정 중 alignment 가 어긋남
- 핵심은 **각 클래스별 데이터 분포를 안정적으로 모델링** 하는 것이며, 이 목표를 달성하기 위해 이중 확률 모델 구조를 버리고 Conditional NF를 활용하는 단일 모델 접근법을 시도하고자 함
- Model requirements
    - NF만을 이용해 조건부 확률 분포 모델링
    - + Continual 하게 학습할 수 있게
- 추가 고려 사항 or Fingdings
    - 클래스 별 균등 학습
    - Task=0 이후 Task들의 빠른 학습
    - 이미지/픽셀 별도 모듈

방법론 개발 방향

- Conditional NF
- + Continual : Head MoE
- 추가 고려 사항 대응

3.3 Class-Conditional Normalizing Flow (CCNF)

3.3.0 전체 구조 개요

- 모델 내에서 클래스별 확률분포를 직접 학습
- 모델은 크게 다음의 세 부분으로 구성

구성 요소	역할	주요 목적
CNN Encoder	멀티스케일 feature 추출	다단계 표현 확보 (hierarchical representation)
Class-Conditional NF (CCNF)	feature 분포를 class 조건부 확률로 모델링	GMM 제거, 단일 확률 구조화
Anomaly Scoring Module	-log p(x	y) + λ H(p(y

3.3.1 Class-Conditioned NF (CCNF)

기존의 NF+GMM 구조를 단일 모델로 통합하여 NF 자체가 클래스별 multi-modal density를 직접 학습하도록 설계했습니다. 이를 통해 확률 공간 불일치 문제를 해결하고, 클래스 조건부 확률 분포를 안정적으로 모델링할 수 있습니다.

수식으로 표현하면 다음과 같습니다:

$$p(x|y) = p_Z(f_\theta(x; e_y)) \cdot \left| \det \frac{\partial f_\theta(x; e_y)}{\partial x} \right|$$

여기서 각 요소의 의미는 다음과 같습니다:

- f_θ: 가역적 변환 함수 (invertible bijective transform)
- e_y: 클래스 임베딩 (class embedding)
- p_Z: 기본 분포 (standard Gaussian)

CCNF 구성 요소

구성 요소	설명
Class Embedding	각 클래스마다 고정된 embedding 벡터 e_y ∈ ℝ^(E_c)를 생성. 새로운 클래스가 등장하면 embedding table만 확장.
Coupling Layer (Affine)	입력 x를 두 부분 (x_a, x_b)로 분할하고, x_a를 기반으로 x_b를 affine transform: y_b = x_b · e^s(x_a, e_y, p) + t(x_a, e_y, p)
FiLM conditioning (optional)	Class embedding이 scale/translation 네트워크에 feature-wise modulation 형태로 주입.
Spectral Normalization	모든 Linear layer에 적용해 Lipschitz constant ≤1 유지.
Clamping (α=0.5~0.8)	Scale factor를 tanh-α로 제한하여 log-determinant 폭발 방지.

구성 요소	설명
Additive-to-Affine Warm-up	초기에 additive-only로 작동 후 점진적으로 affine으로 전환. Jacobian의 급격한 증가를 방지.

Forward Transform

- 입력 feature x를 latent z로 매핑한 뒤 latent에서 density p(z)와 jacobian determinant의 합으로 log-likelihood를 계산

3.3.2 Continual Learning 구조

- 새로운 task 등장 시 NF의 파라미터는 유지, Class embedding만 확장
- 기존 class의 manifold는 그대로 보존되고 새로운 클래스의 embedding만 초기화되어 빠르게 수렴

1. **첫 Task : NF 학습 및 구조 적응**
   a. 첫 번째 Task에서는 NF 파라미터와 class embedding 모두 학습
   b. 이 단계에서 encoder feature의 분포를 class 조건부 latent manifold로 매핑하도록 NF를 학습
   c. 즉 첫 task는 NF의 구조적 initialization 단계로 작동
   d. NF는 latent 공간의 global geometry를 형성

$$p(x|y) = p_Z(f_\theta(x; e_y)) \cdot |\det J_\theta(x; e_y)|$$

2. **이후 Task: Class embedding 확장만 학습**
   - 이후 task에서는 NF의 core parameter θ는 **동결(freeze)** 상태로 유지된다.
   - 새로운 클래스가 추가되면 embedding table의 weight만 확장되어 학습된다:
   - 기존 latent geometry는 유지되면서, 새로운 class에 해당하는 manifold만 embedding을 통해 추가
   - 즉, NF는 한 번 학습되고 continual 단계에서는 embedding만 학습하는 구조이다. 이는 continual learning에서 catastrophic forgetting을 원천적으로 방지

3.3.3 NF 안정화를 위한 학습 제약

Normalizing Flow는 **invertible bijection (가역 함수)**를 통해 복잡한 분포를 base distribution(예: Gaussian)으로 변환한다. 변환의 local stretching/compression 정도는 **Jacobian determinant**로 표현된다:

$$p(x) = p_Z(f_\theta(x)) \cdot |\det J_\theta(x)|$$

안정적인 학습을 위해 Jacobian의 크기를 제어해야 한다.

---

(1) Jacobian 크기 제어 이유

Jacobian (J) = ∂f/∂x는 입력과 latent 간 local 선형 변환을 나타내며, determinant는 확률 밀도 변화율을 결정한다:

$$|\det J| > 1 \Rightarrow \text{공간 팽창 (density 감소)}, \quad |\det J| < 1 \Rightarrow \text{공간 수축 (density 증가)}$$

|det J|가 커지면 numerical overflow가 발생하고 loss가 발산한다. 따라서 Jacobian 크기 제약은 **확률밀도 보존**과 **가역성 보장**의 핵심이다.

---

(2) log-determinant 폭발 억제

NF의 log-likelihood는 다음과 같이 구성된다:

$$\log p(x) = \log p(z) + \log |\det J|$$

두 항의 스케일이 균형을 이루어야 한다. 학습 초기에 log|det J|가 과도하게 커지면 gradient 폭발이 발생한다. 따라서 logdet regularization과 balance loss로 두 항의 평균 크기를 맞춘다:

$$\mathcal{L}_{balance} = |\mathbb{E}[|\log |\det J||] - \mathbb{E}[|\log p(z)|]|$$

---

(3) Lipschitz 안정화 (Spectral Normalization)

Lipschitz 연속성:

$$\|f(x_1) - f(x_2)\| \leq L\|x_1 - x_2\|$$

Spectral Normalization은 각 Linear layer의 weight를 정규화하여 Lipschitz constant ≤ 1을 보장한다:

$$\tilde{W} = \frac{W}{\sigma_{max}(W)}$$

이로써 Jacobian 전체의 spectral radius가 안정되고, NF의 invertibility와 Jacobian 폭발을 동시에 억제한다.

---

(4) Warm-up: Additive → Affine 전환

Coupling layer 변환식:

- **Affine coupling:**

$$y_b = x_b \cdot e^{s(x_a)} + t(x_a)$$

- **Additive coupling:**

$$y_b = x_b + t(x_a)$$

학습 초기에는 scale factor s(x)가 불안정하여 e^s(x)가 급격히 커지면 logdet 폭발이 발생한다. 따라서 **additive-only 모드**로 시작하여 scale factor를 점진적으로 활성화한다:

$$s(x) \leftarrow \tanh(s(x)) \cdot \alpha \cdot \text{scale\_factor}, \quad \text{scale\_factor} = \frac{\text{step}}{\text{warmup\_steps}}$$

Additive는 Jacobian이 항상 1인 안정적 시작점을 제공하고, affine은 표현력을 추가하되 warm-up을 통해 점진적으로 활성화된다.

요약:

- additive는 항상 volume-preserving 변환이므로 학습 초기에 안정적,
- affine은 expressive power를 제공하지만 Jacobian 제어가 필요.
- 현재 코드에서는 **additive → affine** 전환을 warm-up으로 구현하여 두 장점을 결합했다.

구분	Additive	Affine
변환식	(y_b = x_b + t(x_a))	(y_b = x_b \cdot e^{s(x_a)} + t(x_a))
Jacobian	det J = 1	det J = exp(s(x_a))
expressivity	낮음 (비선형 확장 X)	높음 (density scaling 가능)
stability	매우 높음 (no scale)	불안정 가능 (scale 폭발 위험)
초기 학습 시 사용	✅ warm-up 단계	🚫 초기에는 비활성화
목적	identity mapping으로 안정화	density modulation을 통한 표현력 향상

제약 항목	목적	설명
Spectral Normalization	Lipschitz 안정화	모든 Linear layer에 적용, Jacobian 크기 제한
Clamp α	logdet 폭발 억제	s(x) = tanh(s) * α 로 scale 제한
Warm-up (2k steps)	초기 Jacobian 안정화	additive-only → affine 전환
Gradient Clipping	gradient 폭발 방지	norm ≤ 1.0
logdet regularization (λ=2e−4)	Jacobian 크기 제어	
balance loss (λ=1e−2)	log p(z)와 logdet 균형 유지	두 항이 한쪽으로 치우치지 않게
latent variance regularization (λz=0.05)	latent scale normalization	Var(z)≈1 유지, drift 방지
scale penalty (λs=1e−3)	scaling term 억제	∥s∥² 최소화로 안정적 Jacobian 유지

▼ NF 안정화를 위한 제약 근본 원리 (수학적 관점)

NF의 변환은 다음과 같이 표현된다:

$$p(x) = p_Z(f_\theta(x)) \cdot |\det J_\theta(x)|$$

- Jacobian이 너무 크면 density가 급격히 변하며, 확률 질량 보존 ∫p(x)dx=1이 깨진다.
- 따라서 Jacobian의 spectral radius를 제어하는 것은 "확률적 일관성 (probability consistency)"을 유지하기 위한 수학적 조건이다.

NF의 log-likelihood는 다음과 같이 구성된다:

$$\log p(x) = \log p(z) + \log |\det J|$$

- log|det J|는 공간의 volume scaling을 의미한다. 너무 커지면 NF가 log p(z)의 부족한 표현력을 보상하려 함 → 학습 불안정.
- Balance loss는 이 두 항의 균형을 맞춰 "stable density transformation"을 유도한다.

Lipschitz 연속성 조건:

$$\sigma_{max}(W) \leq 1 \Rightarrow \|f(x_1) - f(x_2)\| \leq \|x_1 - x_2\|$$

- 각 layer의 Lipschitz bound를 1 이하로 제한하면, 전체 flow가 contraction mapping이 되어 역함수 존재 및 수렴 보장.

- 초기 additive coupling (det J = 1)은 항상 안정적이며 identity 근사.
- 이후 scale term e^{s(x)}를 점진적으로 활성화하여 expressive power 확보. 즉, "identity 시작 → volume-preserving → adaptive scaling"의 점진적 전이.

3.4 학습 목적 및 Loss 구성

모델은 log-likelihood 최대화 기반의 비지도 밀도 학습을 수행한다.

전체 loss는 다음과 같이 구성된다:

$$\mathcal{L} = \underbrace{\mathcal{L}_g}_{\text{Main likelihood loss}} + \lambda_{det}\mathcal{L}_{logdet} + \lambda_b\mathcal{L}_{balance} + \lambda_z\mathcal{L}_{z\text{-}reg} + \lambda_s\mathcal{L}_{scale}$$

(1) Main Loss: Negative Log-Likelihood (L_g)

$$\mathcal{L}_g = -\frac{1}{D}\log p(x|y)$$

- 모델이 입력 x를 주어진 클래스 y의 확률 공간에 최대한 높은 likelihood로 mapping하도록 학습
- normalization by feature dim D으로 레벨별 scale 균형 유지

(2) Log-determinant Regularization (L_logdet)

$$\mathcal{L}_{logdet} = \frac{1}{D}\mathbb{E}\left[|\log|\det J||\right]$$

- Jacobian 항의 절댓값이 커지면 NF가 지나치게 stretching하게 되므로 이를 억제해 numerical stability 확보

(3) **Balance Loss (L_balance)**

$$\mathcal{L}_{balance} = \frac{1}{D} \left| \mathbb{E}\left[\left| \log \left| \det J \right| \right| \right] - \mathbb{E}\left[\left| \log p(z) \right| \right] \right|$$

- Jacobian term이 과도하게 커지면 latent likelihood가 collapse되므로 두 항의 균형을 맞추기 위한 penalty
- empirical ratio(log_pz/logdet) ≈ 1~3 범위 내 유지

---

(4) **Latent Regularization (L_z_reg)**

$$\mathcal{L}_{z\text{-}reg} = \left(\mathrm{std}(z) - 1 \right)^2$$

- latent representation의 분산을 1에 가깝게 유지하여 continual stage에서 drift 방지 및 density scale 안정화

---

(5) **Scale Penalty (L_scale)**

$$\mathcal{L}_{scale} = \|s\|^2$$

- scale term의 값이 커질수록 Jacobian이 폭발하므로 ‖s‖²를 최소화하여 안정화

---

(6) **Discriminative Contrastive Losses (L_mi, L_e)**

class 간 명시적 구분 신호를 제공하기 위해 contrastive-style 보조 loss 사용

이름	목적	수식	역할
L_mi (Mutual Information)	positive/negative class 구분	\mathcal{L}_{mi} = -(\log p(x|y_{true}) - \log p(x|y_{neg}))	true class와 negative class 간 likelihood 차이 극대화
L_e (Entropy Loss)	예측 확신도 조절	\mathcal{L}_e = -p\log p - (1-p)\log(1-p)	predictive entropy 조절 (confidence calibration)

3.5 Anomaly Scoring

- **Inference 과정**
  - 각 클래스에 대해 log p(x|y)를 계산하여 normality를 평가한다.
  - Posterior 확률은 p(y|x) = Softmax(log p(x|y))로 계산한다.
  - 모델의 불확실성은 entropy H(p(y|x))로 측정한다.
- **Anomaly Score**

$$S(x) = -\log p(x|y) + \lambda_H H(p(y|x))$$

    - 첫 번째 항: likelihood 기반 anomaly (데이터가 학습 분포 밖에 있을수록 높아짐)
    - 두 번째 항: uncertainty 기반 보정 (모델 확신이 낮을수록 높아짐)
- **Multi-level Fusion**
    - 각 NF level에서 계산된 score map을 평균 앙상블하여 최종 anomaly score를 생성한다.

3.5 방법론 정리

항목	내용
기본 구조	Encoder + Multi-level CCNF (Class-Conditional NF)
확률적 역할	NF가 직접 class-conditional density (p(x
GMM 대체	GMM 제거, NF 내부에서 multi-modal 구조 표현
안정화 기법	Spectral norm, clamping, warm-up, logdet & balance loss
학습 목적	Log-likelihood 최대화 + Jacobian 안정화 + latent normalization
Anomaly score	(S(x) = -\log p(x
Continual 대응	Class embedding 확장만으로 new task 적응

3.6 기존 연구와 비교

👉 **Ardizzone et al., "Conditional Invertible Neural Networks for Diverse Image-to-Image Translation (cINN, CVPR 2019)** 입니다.

이 논문이 바로 FiLM 스타일 조건부 Normalizing Flow의 "원형 구조"에 해당합니다.

당신의 **CCNF (Class-Conditional Normalizing Flow)** 와 이 논문의 **cINN** 은 핵심 철학이 유사하지만, **목적과 구조 설계의 관점**이 완전히 다릅니다.

🔷 1. 공통점 — "Conditional Invertible Flow"의 뿌리 공유

항목	공통점 요약
기반 수식	둘 다 change-of-variable 공식을 이용한 maximum-likelihood 학습 (log p(x
조건화 방식	class/condition embedding을 coupling layer의 scale s, translation t network에 주입
FiLM-like 구조	cINN의 *Conditional Coupling Block (CCB)* 과 당신의 *CCNFCoupling (FiLM)* 은 동일 원리 — 조건벡터가 feature를 affine하게 변조
학습 목적	GAN 없이 안정적 MLE 학습 / mode collapse 없음 / invertibility 유지
적용 가능성	모두 class or image condition에 따라 확률 분포 p(x
추론 방식	Forward : x→z→p(z)로 likelihood 평가 / Inverse : z→x로 생성 or 재구성

즉, **당신의 모델(CCNF)** 은 이 논문에서 제안한 **Conditional Coupling Block (CCB)** 개념을 class-level conditioning으로 변형한 구조입니다.

🔷 2. 구조적 차이 — “condition의 성격”과 “학습 방식”

구분	cINN (Ardizzone et al.)	당신의 CCNF
condition 입력	“이미지 Y” (예: day → night) — high-dimensional condition	“클래스 ID y” — discrete categorical condition
conditioning network	별도 CNN ϕ(y) 로 feature pyramid c = {c^(k)} 추출 후 각 coupling layer에 주입	단순 Embedding table → vector e_y + optional positional embedding
학습 목적	**다양한 output 생성 (diverse image-to-image translation)**, 즉 p(X	Y) 를 다모달하게 학습
latent z의 역할	이미지 생성 다양성 확보 (sampling z → 다른 스타일)	정상/비정상 likelihood 평가 (z score 중심으로 이상 탐지)
architecture depth	U-Net형 다해상 coupling + Haar wavelet downsampling (CCB + conditioning pyramid)	single-scale feature NF (multi-level encoder feature NF ensemble)
training strategy	full MLE, joint INN + conditioning network ϕ end-to-end	per-task continual learning with NF freeze + embedding update
continual adaptation	고려 X (single dataset translation task)	핵심 : 새 task → embedding 확장만 학습 (NF 고정) → catastrophic forgetting 방지
stability trick	wavelet downsampling + tanh scale clamp (γ tanh(r))	spectral norm + warm-up (additive→affine) + balance loss + scale penalty

🔶 **요약하면:**

> cINN은 “image → image” conditional generation 문제에 초점,
>
> 당신의 CCNF는 “class → density” conditional likelihood 학습 + continual adaptation에 초점.

🔷 3. 학습 철학의 차이

항목	cINN	CCNF
joint training	condition CNN ϕ 와 INN 을 joint MLE 학습 → feature pyramid 가 mutual information 최대화	첫 task에서만 joint 학습 → 이후 NF freeze 하고 class embedding 만 update (continual setup)
mutual information view	ϕ(y) 가 x와 MI 최대화하도록 자동 학습 (논문 Proposition 1 ~ 2)	class embedding 은 명시적 MI 최대화 아님 — log p(x
expressivity	conditioning network 자체가 CNN이므로 high-level feature 학습 가능	embedding 벡터로 condition 표현 → low dimensional 하지만 stable / expandable

🔷 4. Continual Learning 관점에서의 가장 큰 차별점

항목	cINN	CCNF
task	single domain translation	continual multi-class anomaly detection
forgetting 문제	고려 안함	NF freeze + embedding expansion으로 망각 방지
확률 공간 misalignment 대응	필요 없음 (한 도메인 안)	기존 NF-GMM 불일치 문제 해결 — 단일 확률 모델로 통합
architecture objective	expressivity ↑	stability + expandability ↑

🔷 5. 수식 수준 비교

모델	likelihood 식
cINN	(\log p(x
CCNF (당신)	(\log p(x

두 식 모두 동일한 flow likelihood 구조지만,

CCNF는 **정규화 term과 안정화 term**을 추가하고,

condition embedding e_y 를 discrete task별로 관리한다는 점이 가장 큰 차이입니다.

🔷 6. 시각적 비교 (논문 Fig. 2 vs. 당신의 구조)

- **cINN (Fig. 2 in paper)** → "Conditioning CNN ϕ(y)" → feature pyramid → INN의 모든 coupling layer에 multi-scale feature 주입.
- **CCNF** → "Encoder feature → Class embedding e_y (혹은 FiLM)" → NF per feature level 적용 (멀티레벨 NF 병렬).

즉,

- cINN은 **condition을 multi-scale CNN feature로 만드는 전방 네트워크형 conditioning**,
- CCNF는 **NF 자체를 여러 스케일로 분리하고 각각을 class embedding 으로 condition**.

✅ 요약: 당신의 CCNF vs cINN 핵심 비교표

항목	cINN (CVPR 2019)	CCNF (당신 모델)
목적	conditional generation (image→image)	continual anomaly detection (class conditional density)
condition type	high-dim image	discrete class ID
condition encoder	CNN ϕ(y) (feature pyramid)	embedding table (+ FiLM)

항목	cINN (CVPR 2019)	CCNF (당신 모델)
coupling layer	CCB (conditional affine block)	CCNFCoupling + FiLM + spectral norm
downsampling	Haar wavelet	없음 / encoder multi-level feature 대신
training	joint MLE end-to-end	continual per-task MLE with NF freeze
stability 기법	tanh scale clamp	spectral norm, warm-up, balance, z-reg
outcome	diverse sample generation	stable likelihood estimation + alignment 보존

🔶 결론

> CCNF는 cINN의 아이디어(conditional coupling block)를 “class condition” + “continual learning” 맥락으로 재구성한 확장형입니다.
>
> 즉,
>
> - **cINN → “조건부 생성 모델링”**,
> - **CCNF → “조건부 확률 밀도 + continual 적응”**
>
> 구조적으로는 cINN에서의 **conditioning network ϕ(y)** 를 **class embedding + FiLM** 으로 바꾸고,
>
> 학습을 **continual task별 freeze/expand** 형태로 바꾼 것이 핵심 차이입니다.

원하신다면,

제가 cINN 논문 Figure 2 구조와 당신의 CCNF 구조를 나란히 정리한

도식 비교(architecture diagram) 도 그려드릴 수 있어요 —

“cINN의 conditioning CNN → CCNF의 class embedding & FiLM” 대응관계를 시각적으로 보여주는 형태로요.

그림으로 비교해드릴까요?