

MoLE-Flow: Mixture of LoRA Experts with Normalizing Flow for Continual Anomaly Detection

지속적 이상 탐지를 위한 LoRA 전문가 혼합 정규화 흐름

Author Name

Institution Name

January 4, 2026

Contents

- 1 Introduction
- 2 Overall Architecture
- 3 Feature Extraction
- 4 Task Adapters
- 5 MoLE-Flow
- 6 Training & Inference
- 7 Experiments Part 1: Hyperparameter Optimization
- 8 Experiments Part 2: Paper Experiment Design
- 9 Conclusion & Future Work

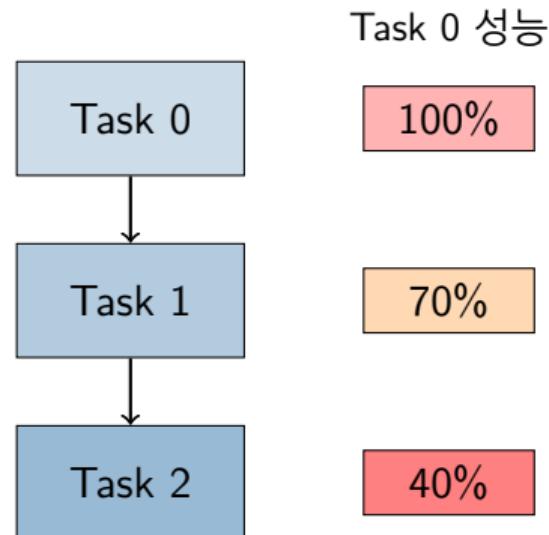
Problem: Continual Anomaly Detection

산업 환경의 현실

- 새로운 제품이 지속적으로 추가됨
- 모든 데이터를 저장/재학습하기 어려움
- 이전 제품에 대한 성능 유지 필요

핵심 문제: Catastrophic Forgetting

- 새 작업 학습 시 이전 작업 성능 급감
- 기존 방법: 전체 모델 복제 → 메모리 비효율



Our Solution: MoLE-Flow

핵심 아이디어: Parameter Isolation

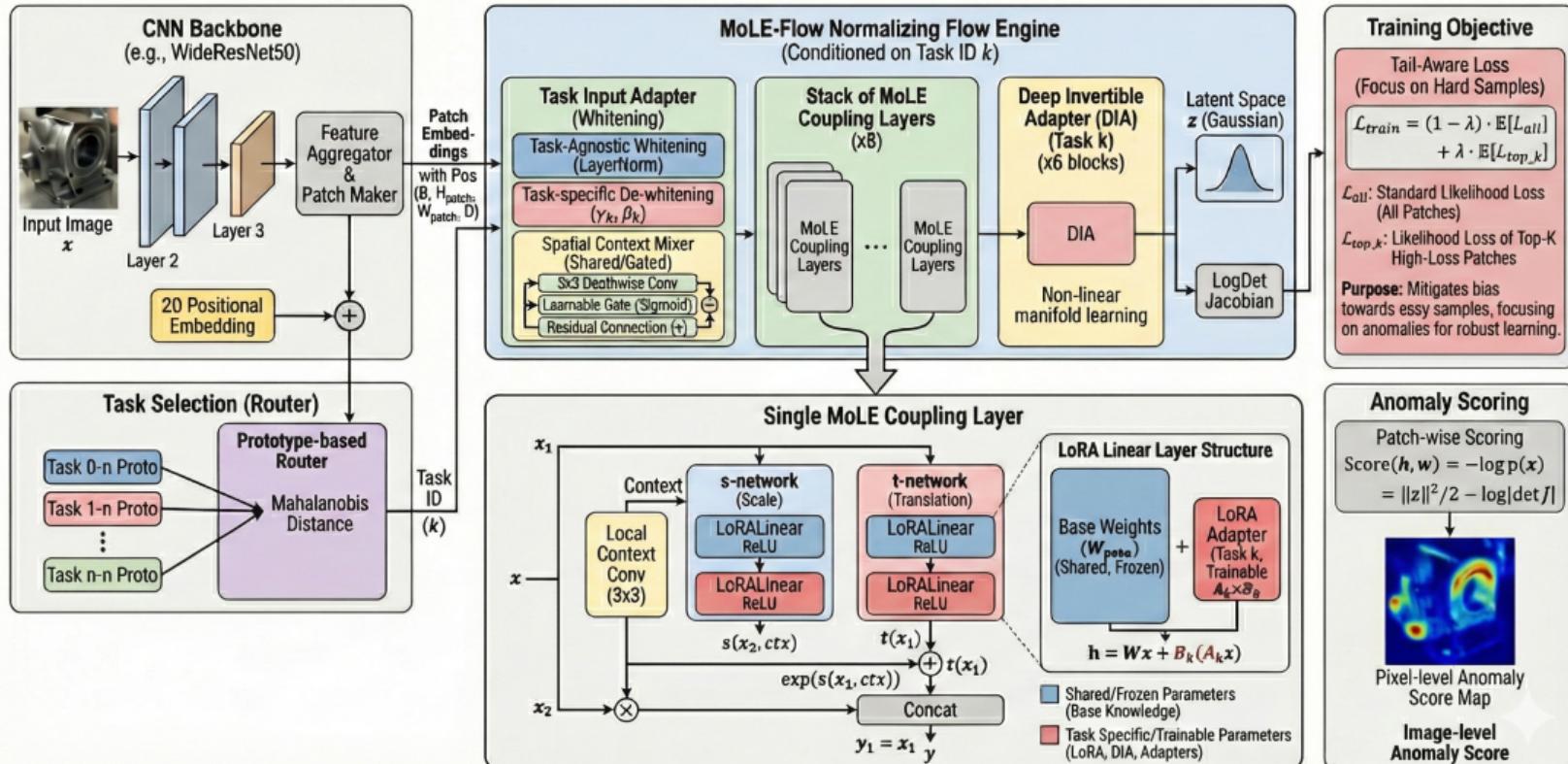
Base 모델은 공유/고정하고, Task-specific 정보는 경량 어댑터에만 저장

5가지 핵심 기여:

- ① **MoLE (Mixture of LoRA Experts)**: NF coupling 블록에 LoRA 적용
- ② **WhiteningAdapter**: 작업 간 분포 차이 정렬
- ③ **Deep Invertible Adapter (DIA)**: 비선형 매니폴드 적용
- ④ **Prototype-based Router**: Task ID 없이 자동 라우팅
- ⑤ **Tail-Aware Loss**: 분포 경계 학습 강화

Overall Architecture

MoLE-Flow: Continual Learning via Mixture of LoRA Experts & Deep Invertible Adapters for Anomaly Detection



Parameter Isolation 전략:

- 회색 (Shared/Frozen): Backbone, Base NF weights - 모든 작업이 공유
- 빨간색 (Task-specific): LoRA, DIA, Whitening Adapter - 작업별 독립

학습 전략:

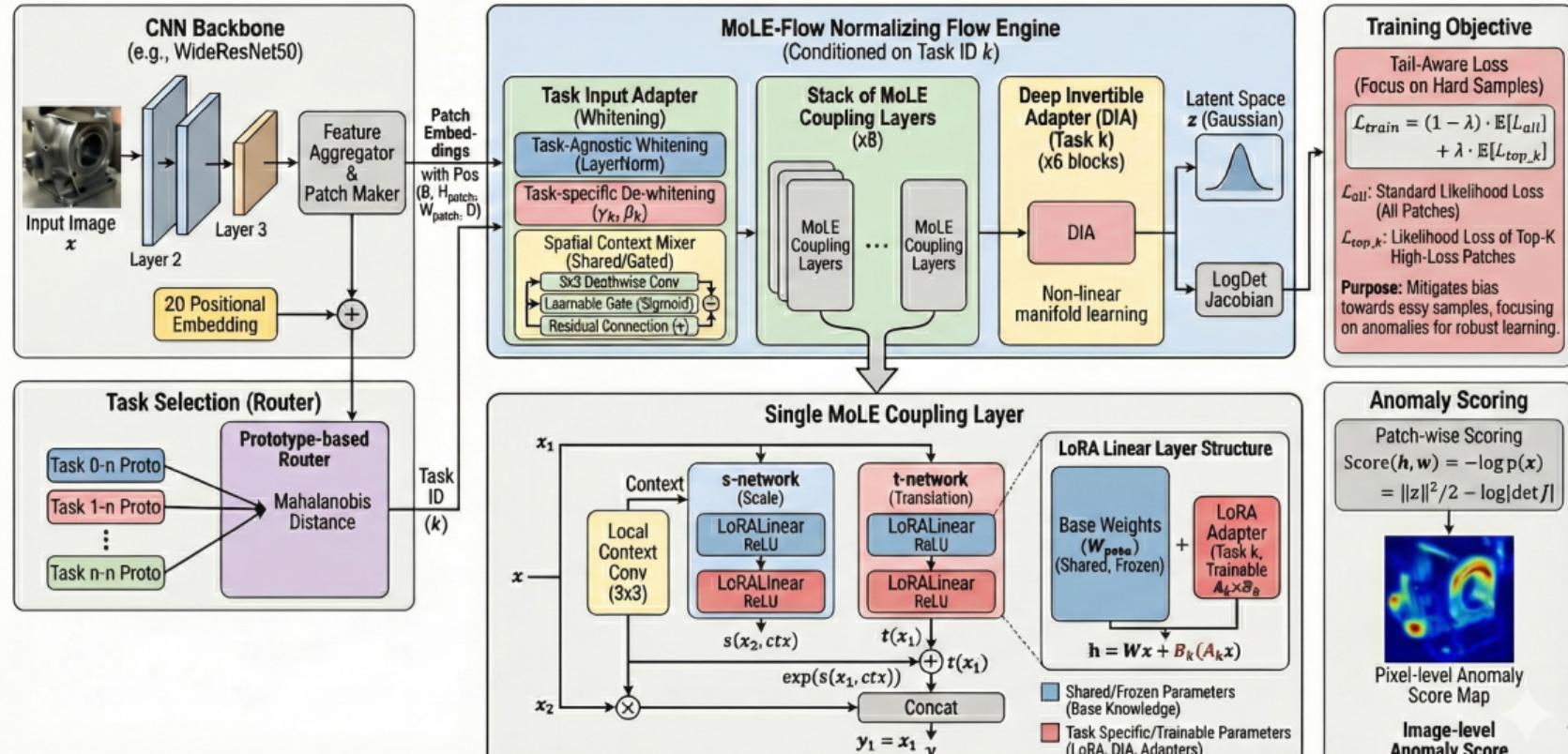
- Task 0: Base NF + 모든 어댑터 학습 → Base 동결
- Task $t > 0$: 새로운 어댑터(LoRA, Whitening, DIA)만 학습

핵심 장점

작업당 **약 10.8%**의 추가 파라미터만으로 완전한 작업 분리 달성

Detailed Architecture

MoLE-Flow: Continual Learning via Mixture of LoRA Experts & Deep Invertible Adapters for Anomaly Detection



Pipeline 수식

$$\mathbf{x} \xrightarrow{\text{Backbone}} \mathbf{F} \xrightarrow{\text{PE}} \mathbf{F}' \xrightarrow{\text{Whitening}} \hat{\mathbf{F}} \xrightarrow{\text{SCM}} \tilde{\mathbf{F}} \xrightarrow{\text{NF+LoRA}} \mathbf{z}' \xrightarrow{\text{DIA}} (\mathbf{z}, \log |\det \mathbf{J}|)$$

공유/고정 파라미터

- Backbone (ViT-Base)
- Base NF weights (\mathbf{W}_{base})
- Spatial Context Mixer

Task-specific 파라미터

- LoRA: $\mathbf{A}_t, \mathbf{B}_t$
- Whitening: γ_t, β_t
- DIA: 별도 flow 블록

핵심 장점

작업당 **약 10.8%**의 추가 파라미터만으로 완전한 작업 분리 달성

Feature Extraction & Preprocessing

1. Backbone (ViT-Base)

- 사전 학습된 ViT 사용 (Frozen)
- 다중 스케일 특징: 블록 {1, 3, 5, 11}

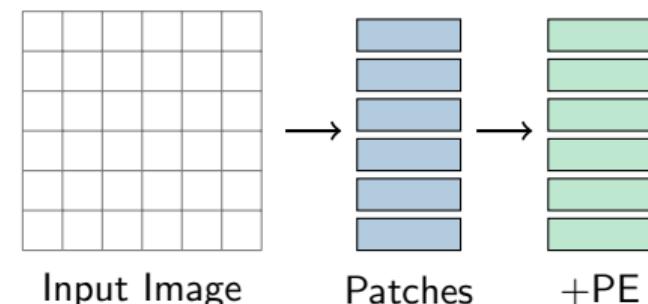
2. Patch Maker

- 특징 맵을 패치 단위로 분할
- 출력: $\mathbf{F} \in \mathbb{R}^{B \times H \times W \times D}$

3. Positional Encoding

- NF의 순열 불변성 극복
- 2D Sinusoidal PE 추가

$$\mathbf{F}' = \mathbf{F} + \mathbf{P}$$



Task Adapters: Whitening

문제: 작업 간 특징 분포 차이

- Task 0 이후 Base Flow는 동결됨
- 새로운 작업의 분포가 다르면 성능 저하

해결: 2단계 Whitening 전략

Step 1: Task-Agnostic Whitening

$$\mathbf{x}_{\text{white}} = \frac{\mathbf{F}' - \mathbb{E}[\mathbf{F}']}{\sqrt{\text{Var}[\mathbf{F}']} + \epsilon}$$

- 모든 작업을 동일한 시작점으로
- 학습 파라미터 없음

Step 2: Task-Specific De-whitening

$$\gamma_t \in [0.5, 2.0]$$

$$\beta_t \in [-2.0, 2.0]$$

$$\hat{\mathbf{F}}_t = \gamma_t \odot \mathbf{x}_{\text{white}} + \beta_t$$

- 작업별 학습 파라미터
- 범위 제한으로 안정성 확보

MoLE-Flow: Spatial Context Mixer

문제: 패치 독립 처리의 한계

- 기존 NF는 패치를 독립적으로 처리
- 주변 패치와의 관계 정보 부족

해결: Spatial Context Mixer

$$\mathbf{x}_{\text{context}} = \text{DepthwiseConv}_{3 \times 3}(\mathbf{x})$$

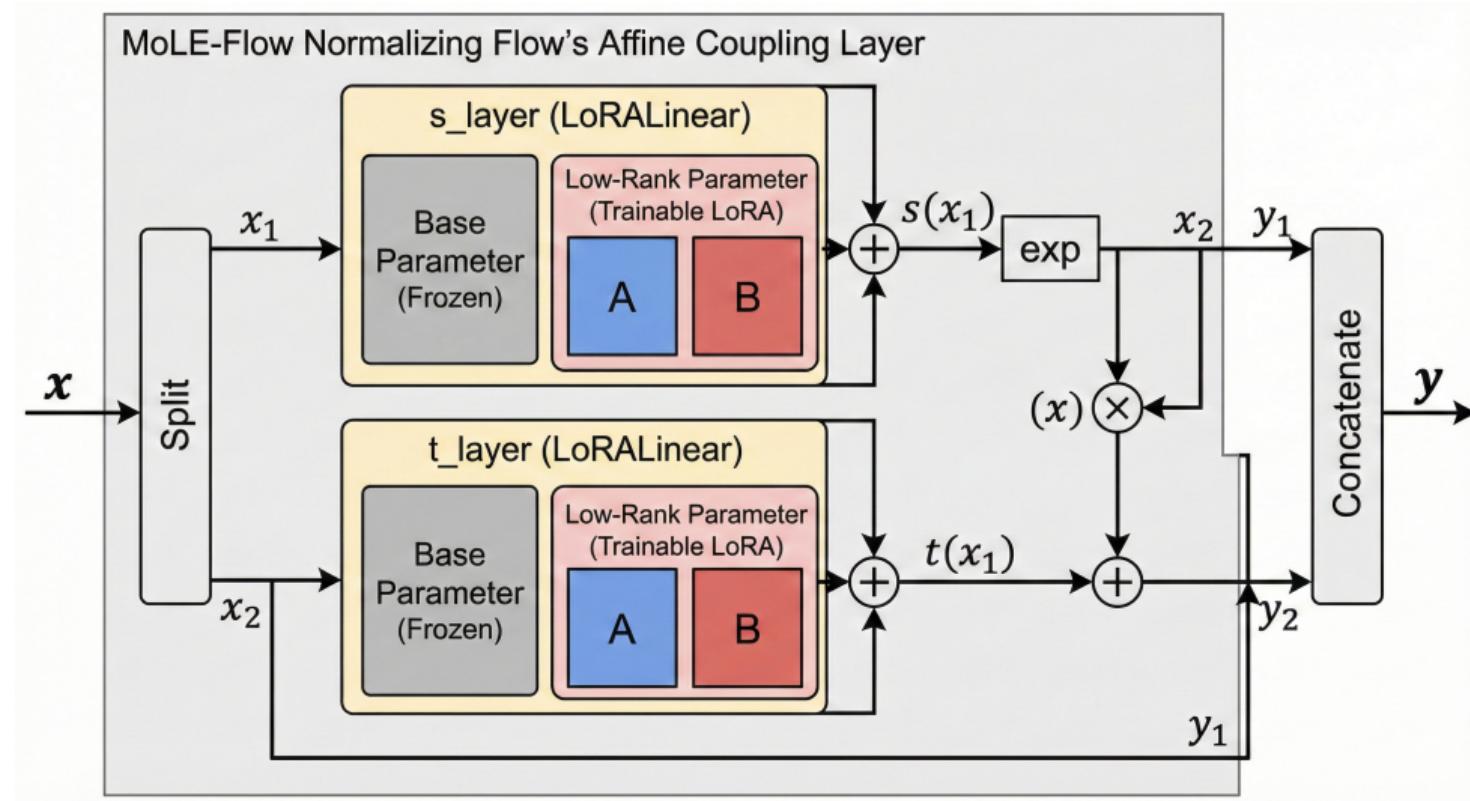
$$g = \sigma(\theta_{\text{gate}})$$

$$\mathbf{x}_{\text{mixed}} = (1 - g) \cdot \mathbf{x} + g \cdot \mathbf{x}_{\text{context}}$$

효과

주변 문맥과의 상대적 불일치를 명시적으로 파악 → 국소적 이상 탐지에 필수적인 **Spatial Inductive Bias** 제공

MoLE-Flow: LoRA-based Coupling Layer



MoLE-Flow: LoRA-based Coupling Layer (상세)

Base Linear (Frozen Anchor)

- Task 0에서 학습 후 고정
- 범용적인 특징 변환 역할
- 모든 작업이 공유

LoRA Linear (Task-Specialist)

- 작업 고유 분포 특성 보정
- 저랭크 행렬 ($r = 64$)
- 작업마다 새로운 **A**, **B** 생성

파라미터 효율성

전체 대비 **약 8.3%**의 파라미터만으로 효과적인 작업 적응

MoLE-Flow: Context-Aware s/t Networks

Coupling Layer 변환: $y_2 = x_2 \odot \exp(s) + t$

Scale Network (Context-Aware)

$$s = f_s([x; \text{ctx}])$$

Translation Network (Context-Free)

$$t = f_t(x)$$

- 원본 + 문맥 정보 결합
- 스케일 변화는 주변과의 상호작용에 민감

- 원본 특징만 사용
- 분포 중심은 패치 고유 특성에 종속

설계 철학

s 와 t 의 역할 분리 → 각각에 적합한 inductive bias 반영

MoLE-Flow: Deep Invertible Adapter (DIA)

문제: 선형 LoRA의 한계

- LoRA, Whitening은 선형 변환
- 복잡한 비선형 분포 차이 보상 어려움

해결: DIA (비선형 가역 적용)

$$\mathbf{z}_{\text{final}} = f_{\text{DIA}}^{(t)}(\mathbf{z}_{\text{base}})$$

DIA의 장점:

- ① 비선형 매니폴드 적용: 선형 LoRA가 표현 불가능한 변환 학습
- ② 가역성 보장: 밀도 추정 속성 유지
- ③ 완전한 작업 분리: 작업별 독립 파라미터
- ④ 후처리 위치: Base NF 출력에 추가 보정

Training Objective: Likelihood Calculation

Normalizing Flow의 핵심: 변수 변환

$$\log p(\mathbf{x}) = \log p(\mathbf{z}_{\text{final}}) + \log |\det \mathbf{J}_{\text{total}}|$$

잠재 확률

$$\log p(\mathbf{z}) = -\frac{1}{2}\|\mathbf{z}\|_2^2 - \frac{D}{2} \log(2\pi)$$

- 표준 정규분포 가정
- 원점에 가까울수록 높은 확률

Jacobian 누적

$$\log |\det \mathbf{J}_{\text{total}}| = \log |\det \mathbf{J}_{\text{flow}}| + \log |\det \mathbf{J}_{\text{DIA}}|$$

- 각 변환의 부피 변화율
- 밀도 보정 역할

Training Objective: Tail-Aware Loss

문제: 일반 NLL의 한계

- 모든 패치의 평균 우도 최대화
- “쉬운” 정상 패치에만 집중 → 경계 학습 부족

해결: Tail-Aware Loss

$$\mathcal{L}_{\text{train}} = (1 - \lambda) \cdot \mathbb{E}[\mathcal{L}_{\text{all}}] + \lambda \cdot \mathbb{E}[\mathcal{L}_{\text{top-k}}]$$

- $\lambda = 0.5$: 가중치 비율 (최적화된 값)
- $\mathcal{L}_{\text{top-k}}$: 상위 5% 고손실 패치의 평균 NLL

효과

정상 분포의 경계(Boundary)를 더 명확하게 학습

Inference: Task Routing

핵심 문제: 추론 시 Task ID가 주어지지 않음

해결: Prototype-based Mahalanobis Router

$$t^* = \arg \min_t D_M(\mathbf{f}, t), \quad D_M(\mathbf{f}, t) = \sqrt{(\mathbf{f} - \boldsymbol{\mu}_t)^\top \boldsymbol{\Sigma}_t^{-1} (\mathbf{f} - \boldsymbol{\mu}_t)}$$

장점:

- **One-stage**: 별도의 라우팅 추론 불필요
- **공분산 고려**: 분포 형태를 반영한 거리 측정
- **스케일 불변**: 특징 차원 간 스케일 차이 자동 보정

기존 방법 대비 장점

기존: Two-stage (라우팅 → 탐지)

Ours: **One-stage** (라우팅 + 탐지 동시)

Inference: Anomaly Scoring

Step 1: Patch-wise Scoring

$$\text{Score}_{(h,w)} = \underbrace{\frac{1}{2} \|\mathbf{z}_{(h,w)}\|^2}_{\text{Distance Score}} - \underbrace{\log |\det \mathbf{J}|_{(h,w)}}_{\text{Distortion Score}}$$

Step 2: Top-K Mean Aggregation

$$\text{Final Score} = \frac{1}{K} \sum_{i=1}^K S_{\text{top}}^{(i)}$$

vs Max

- 단일 노이즈에 의한 오탐 방지

- 이상 신호 희석 방지
- 결함은 보통 군집 형성

Hyperparameter Optimization Overview

실험 규모

78개 이상의 실험을 수행하여 최적의 하이퍼파라미터 조합 탐색

Baseline 성능 (MVTec AD, WRN50-80ep):

Metric	Baseline	Target
Image AUC	0.9796	≥ 0.98
Pixel AP	0.4640	0.54 - 0.60
Routing Acc.	100%	100%

최종 달성:

- Pixel AP: 0.5350 (+15.3% 개선)
- Image AUC: 0.9824 (유지/향상)

Individual Component Effects

각 하이퍼파라미터의 개별 효과 (vs Baseline 0.4640):

Component	Pixel AP	Delta	Impact
LogdetReg 1e-4	0.5055	+4.15%	★★★★★
ScaleCtxK5	0.4870	+2.30%	★★★★
TopK5-TailW0.5	0.4866	+2.26%	★★★★
lr 3e-4	0.4718	+0.78%	★★
DIA6	0.4606	-0.34%	★

핵심 발견

Log-determinant Regularization (1e-4)이 단일 최대 개선 효과 (+4.15%)

Tail Weight Effect Analysis

TailWeight: 어려운 패치에 집중하는 정도

TailW	Pixel AP	Image AUC	최적 TailTopK
0.50	0.5221	0.983	5%
0.55	0.5256	0.983	5%
0.60	0.5290	0.983	3%
0.65	0.5324	0.983	1%
0.75	0.5449	0.981	2%
0.80	0.5447	0.981	3%

Trade-off 발견:

- TailW 증가 → Pixel AP 향상
- TailW > 0.75 → Image AUC 약간 하락 ($0.983 \rightarrow 0.981$)

Optimization Journey

Baseline에서 최적 설정까지의 개선 과정:



최적 설정:

- TailW=0.55, TopK=5, LogdetReg=1e-4, ScaleCtxK=5, lr=3e-4

Top 10 Experiments by Pixel AP

Rank	Configuration	Image AUC	Pixel AP	Routing
1	TailW0.55-TopK5-LogdetReg1e-4-ScaleCtxK5-lr3e-4	0.9824	0.5350	100%
2	TailW0.65-TailTopK3-TopK5-LogdetReg1e-4	0.9827	0.5324	100%
3	TopK3-TailW0.5-LogdetReg1e-4-ScaleCtxK5	0.9802	0.5317	100%
4	TopK5-TailW0.5-LogdetReg1e-4-ScaleCtxK5	0.9809	0.5317	100%
5	TailW0.6-TailTopK3-TopK5-LogdetReg1e-4-ScaleCtxK5-80ep	0.9826	0.5310	100%
6	TailW0.6-TopK5-LogdetReg1e-4	0.9827	0.5290	100%
7	TailW0.55-TopK5-LogdetReg1e-4	0.9827	0.5256	100%
8	TailW0.5-TailTopK3-TopK5-LogdetReg1e-4	0.9830	0.5242	100%
9	FullBest-80ep-lr3e-4-LoRA128-C10-DIA5-TailW0.55	0.9836	0.5242	100%
10	TopK5-TailW0.5-LogdetReg1e-4	0.9826	0.5221	100%

Observation: 모든 상위 설정에서 **Routing Accuracy 100% 유지**

Per-Class Performance Improvement

Class	Baseline	Top Config	Improvement
carpet	0.3601	0.6167	+0.2566
bottle	0.4551	0.6774	+0.2223
leather	0.2292	0.3970	+0.1678
toothbrush	0.4028	0.5619	+0.1591
wood	0.3546	0.4453	+0.0907
hazelnut	0.5110	0.5798	+0.0688
metal_nut	0.8491	0.7776	-0.0715
cable	0.6575	0.6339	-0.0236
Mean	0.4640	0.5350	+0.0710

Key Observations:

- **Textured classes** (carpet, leather): 가장 큰 개선
- **Fine-grained objects** (metal_nut, cable): 약간의 성능 저하

Hyperparameter Sensitivity Summary

DIA Blocks

Blocks	Image AUC	Pixel AP
4	0.9793	0.4735
6	0.9820	0.4606
8	0.9825	0.4546

↑ Blocks → ↑ Image AUC

↓ Pixel AP (trade-off)

LoRA Rank (Minimal Effect)

Rank	Image AUC	Pixel AP
32	0.9794	0.4737
64	0.9793	0.4735
128	0.9794	0.4736
256	0.9796	0.4741

- Rank 64로 충분
- 파라미터 효율성 확인

Warning: Coupling Layers

Coupling16: 학습 불안정 발생 (Image AUC 0.73으로 급락). 8-12 권장.

Optimal Configuration Recommendations

Balanced Setting (권장)

- tail_weight = 0.55
- topk = 5
- logdet_reg = 1e-4
- scale_context_k = 5
- lr = 3e-4
- epochs = 60

→ Image AUC ≈ **0.982**, Pixel AP ≈ **0.535**

Max Image AUC Setting

- tail_weight = 0.55
- lora_rank = 128
- dia_n_blocks = 5
- coupling = 10
- lr = 3e-4
- epochs = 80

→ Image AUC ≈ **0.984**, Pixel AP ≈ **0.524**

Experimental Setup

Datasets

- **MVTec AD**: 15 classes, 5,354 images
- **VisA**: 12 classes, 10,821 images
- **MPDD**: 6 classes (metal parts)

Evaluation Metrics

- Image AUROC / AP (primary)
- Pixel AUROC / AP / PRO
- Forgetting (F), BWT
- Routing Accuracy

Implementation

- Backbone: ViT-Base-16 (frozen)
- NF: 8 Coupling Layers
- LoRA rank: 64
- DIA: 4 blocks
- Epochs: 60 per task
- LR: 2×10^{-4}
- Input: 224×224

Hardware: NVIDIA RTX 4090

4가지 시나리오로 지속 학습 능력 검증:

① Scenario A: Standard 5-Task Protocol

- MVTec 15 classes → 5 tasks (각 3 classes)
- Task 0: leather, grid, transistor (텍스처 + 객체 혼합)

② Scenario B: Long Sequence (15-Task)

- 15개 클래스를 각각 독립 작업으로 구성
- 장기 망각(Long-term Forgetting) 분석

③ Scenario C: Class Order Sensitivity

- 5개 무작위 순서로 학습
- Parameter Isolation의 순서 견고성 검증

④ Scenario D: Task 0 Dependency Analysis

- Texture-first vs Object-first vs Mixed-first
- Base weights 학습이 후속 작업에 미치는 영향

Comparison with SOTA Methods

비교 대상:

Continual AD Methods

- **DNE**: 분포 저장 기반
- **UCAD**: Prompt tuning 기반 SOTA

CL + AD Adaptation

- EWC: 중요 파라미터 정규화
- LwF: Knowledge Distillation
- Replay (5%): 데이터 저장

Bounds

- **Joint Training**: Upper bound
- **Task-Separated**: 망각 없음, 비효율
- **Fine-tuning**: Lower bound

검증 목표:

- 이상 탐지 성능 우수성
- Forgetting ≈ 0 (Parameter Isolation)
- Routing Accuracy 100%

Ablation Study Design (7 Components)

Removed Component	Config Flag	Δ Img AUC	Δ Pix AP	Importance
<i>Core Adapters</i>				
w/o DIA	use_dia=False	-3.14%	-1.49%	★★★★★
w/o Whitening	use_task_adapter=False	-1.89%	-2.74%	★★★★
w/o LoRA	use_lora=False	+0.04%	+0.18%	★
<i>Context Modules</i>				
w/o Spatial Context		-0.21%	-0.76%	★★
w/o Scale Context		-0.18%	-0.41%	★★
<i>Loss Components</i>				
w/o Tail-Aware Loss		-	-	★★★
w/o LogdetReg	logdet_reg=0	-	-4.15%	★★★★★

Key Insight

DIA가 Image AUC에, LogdetReg가 Pixel AP에 가장 중요

Cross-Dataset Generalization

MVTec → VisA → MPDD 일반화 평가:

Dataset	Classes	Image AUC	Pixel AP	Routing Acc.
MVTec AD	15	0.9824	0.5350	100%
VisA	12	0.8566	0.2878	100%
MPDD	6	0.9019	0.2890	98.12%

VisA Analysis:

- 복잡한 텍스처와 미세 결함 → 더 도전적
- 병목 클래스: macaroni1/2 (Pixel AP < 0.1)
- ViT backbone: Image AUC 0.88 (WRN50 대비 +4.2%)

MPDD:

- 금속 표면의 유사성 → Routing 오류 (98.12%)

Continual Learning Analysis

Forgetting 분석:

- Parameter Isolation (LoRA + DIA) → **Forgetting** ≈ 0
- vs Fine-tuning: 급격한 성능 하락
- vs EWC/LwF: 완만한 하락

Router Performance:

- 5-Task: 100% Routing Accuracy
- 15-Task: Oracle과의 Gap 분석
- Task 수 증가에 따른 Scalability 검증

Storage Efficiency:

- MoLE-Flow: 약 2MB/Task (LoRA + DIA)
- vs Replay (5%): 50MB/Task → **25x 절약**

Summary

MoLE-Flow: 지속적 이상 탐지를 위한 프레임워크

- ① **Parameter Isolation**: Base 공유 + Task-specific 어댑터
- ② **MoLE**: LoRA 기반 경량 작업 적응
- ③ **Whitening Adapter**: 분포 정렬을 통한 안정적 학습
- ④ **DIA**: 비선형 매니폴드 적응
- ⑤ **Prototype Router**: One-stage 추론
- ⑥ **Tail-Aware Loss**: 경계 학습 강화

핵심 장점

- 파멸적 망각 방지 ($\text{Forgetting} \approx 0$)
- 파라미터 효율성 (작업당 $\sim 10.8\%$)
- Task ID 없이 추론 가능 (Routing 100%)

Key Results

MVTec AD (Best Performance)

- Image AUC: **0.9836**
- Pixel AP: **0.5350**
- Routing Acc.: **100%**

개선 효과

- Pixel AP: +15.3% (Baseline 대비)
- 78개 실험을 통한 최적화

Cross-Dataset

- VisA: Image AUC 0.8566
- MPDD: Image AUC 0.9019

핵심 하이퍼파라미터

- LogdetReg 1e-4 (최대 효과)
- TailW 0.55-0.75
- ScaleCtxK 5

Future Work

1. 성능 향상 방향

- Pixel AP 0.6+ 달성을 위한 추가 최적화
- 고해상도 입력 (448×448) 적용
- DINOv2 ViT-L/H 등 강력한 backbone 탐색

2. 일반화 강화

- VisA 병목 클래스 (macaroni) 특화 전략
- Multi-scale 평가 양상을
- Class-specific hyperparameter tuning

3. 실용화 연구

- 실시간 추론 최적화
- 경량화 (Quantization, Pruning)
- 온라인 학습 시나리오 확장

Thank You!

Questions?

GitHub: github.com/your-repo/moleflow
Contact: author@example.com