KNN-Contrastive Learning for Out-of-Domain intent Classification

Yunhua Zhou, Peiju Liu, Xipeng Qiu

School of Computer Science, Fudan University

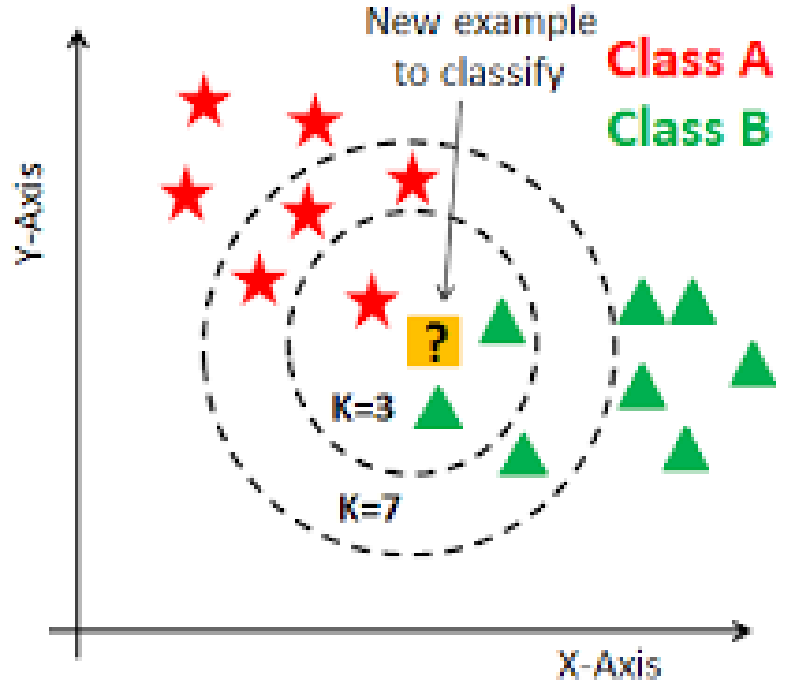
Peng Cheng Laboratory

Shanghai Collaborative Innovation Center of Intelligent Visual Computing

전북대학교 IT정보공학과
202018392
박나현

KNN (K-Nearest Neighbors)

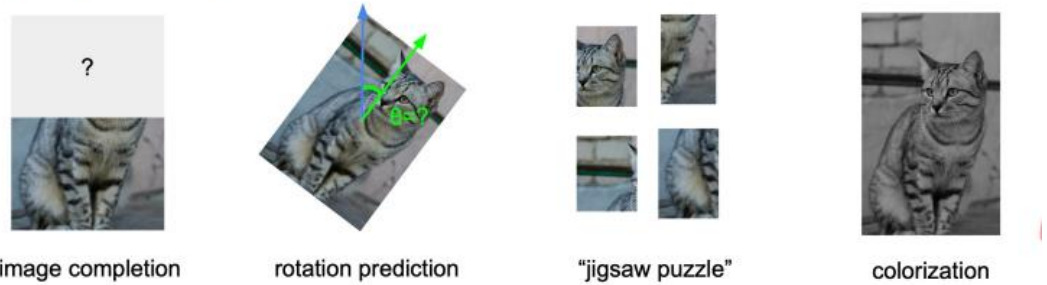
- 모든 training example을 저장
  - memory-based learning
- 데이터 중 자신과 가장 가까운 데이터를 찾아 자신의 label로 결정



Self-supervised learning

- **“Pretext” task**를 해결하는 방법
  - 사전에 정의된 작업을 맞추도록 학습
  - Downstream task를 위한 좋은 feature 를 생성하는 task

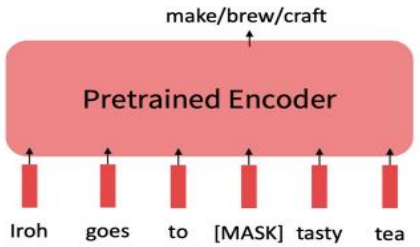
- Example: learn to predict image transformations / complete corrupted images

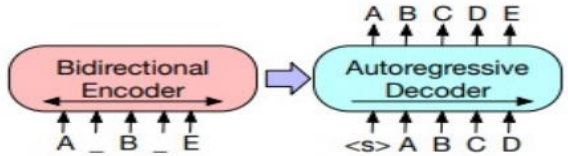


- Solving the pretext tasks allow the model to learn good features
- We can automatically generate labels for the pretext tasks

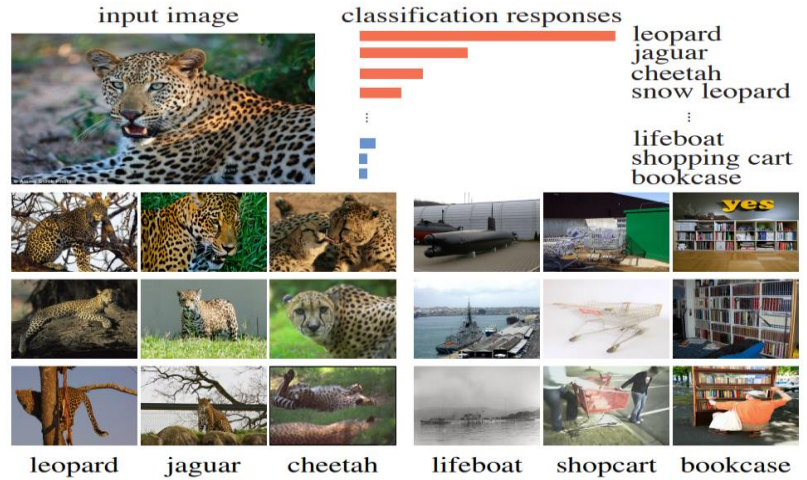
Self-supervised pretext tasks (text)

- Example: Replace some fraction of words in the input with a special [MASK] token and learn to predict these words
  - Masked Language Model, Denoise Autoencoder, …





Contrastive Learning



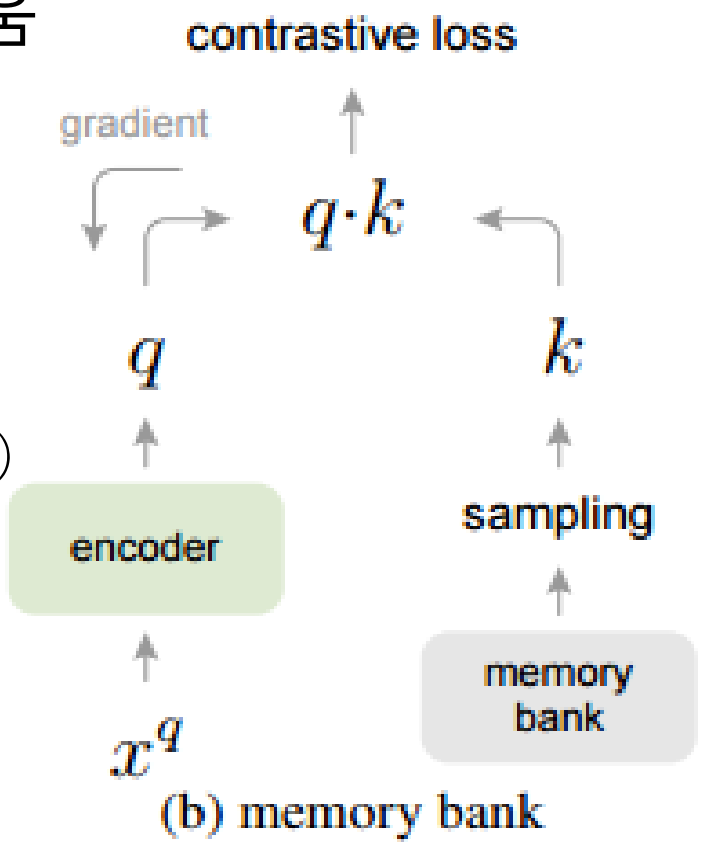
- Motivation
  - 지도학습 기반의 이미지 분류 모델 결과, 비슷한 이미지일 때 확률 값 높음
  - Label 정보 없이 data 사이의 유사성이 존재할 것이라는 가정
- Self-supervised learning 에서 자주 사용
  - **Pretext task 에서 pseudo-label 가져오는 대신**
    similarity 에 따른 multiple input pairs 에 대한 model 학습
- Learning by comparing among different samples
  - **Positive pair 끼리 거리를 좁히고, negative pair 끼리 거리가 멀어지도록 학습**

Contrastive learning
memory bank

- Data끼리 구분하도록 하였을 경우 일반적인 softmax 사용 시 분모가 너무 커져 확률 값이 작아지므로 학습이 어려움

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i / \tau)}$$

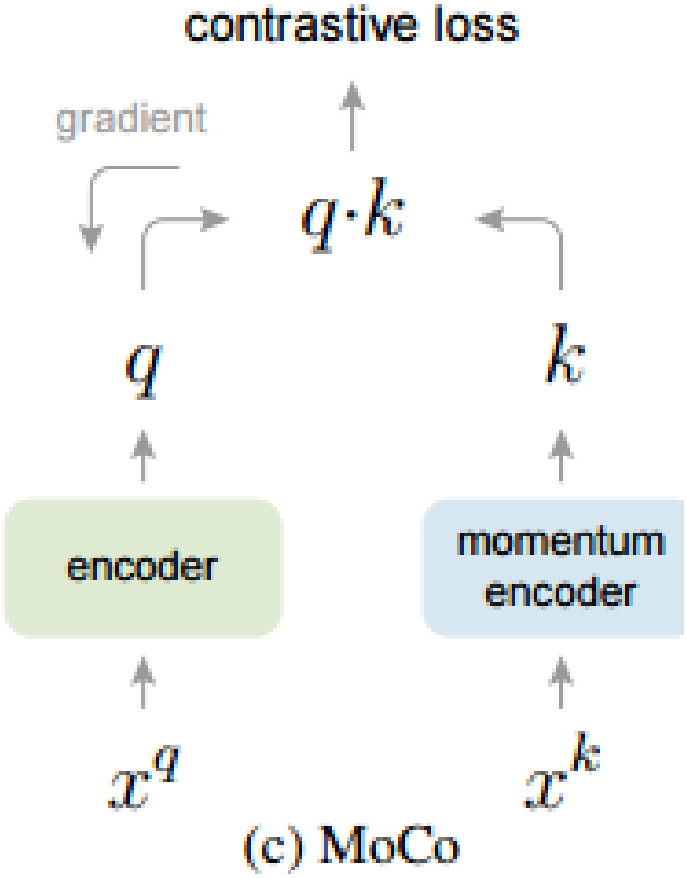
- 계산량이 너무 많아 일부만 **sampling**하여 계산
- Memory Bank 의 단점
  - Data 전체에 대한 embedding 정보를 계속 저장 (memory 문제)
  - Random sampling 을 통해 negative sample 구성
    - Data 별로 학습에 기여하는 정도 다름



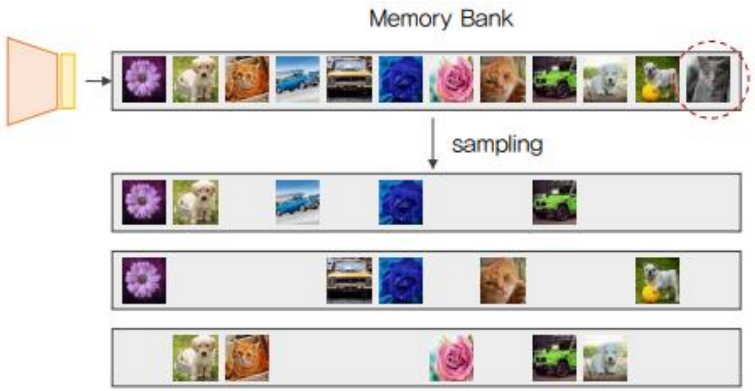
(b) memory bank

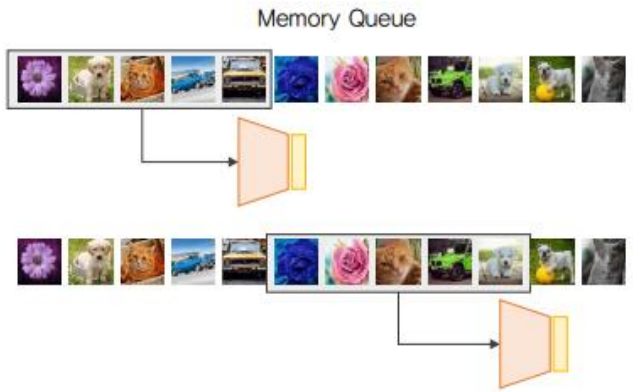
Contrastive learning
MoCo(Momentum Contrast)

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i / \tau)}$$



- Contrastive loss 사용하는 Self-supervised model
- **Memory Queue**



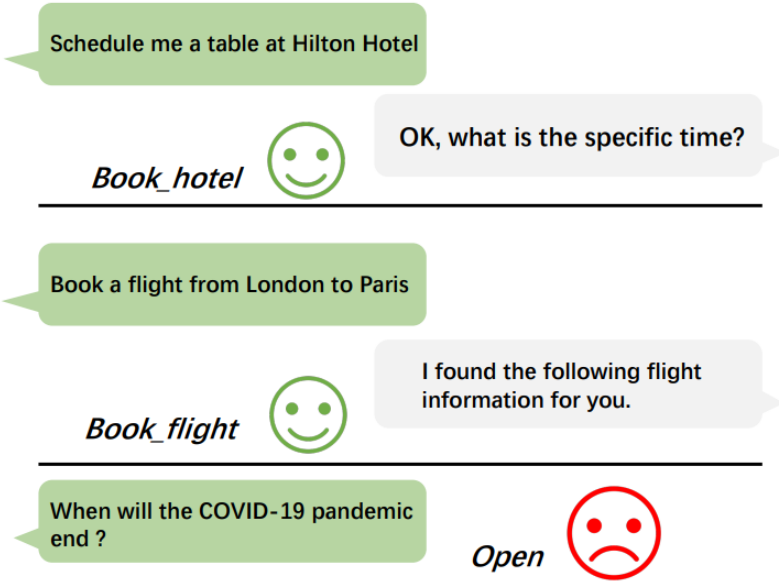


- **Key encoder의 Momentum update**
  - Memory bank 빠르게 변하면 consistency 유지 힘들어 학습이 제대로 이뤄지지 않음

$$\theta_{\mathrm{k}} \leftarrow m\theta_{\mathrm{k}} + (1-m)\theta_{\mathrm{q}}$$

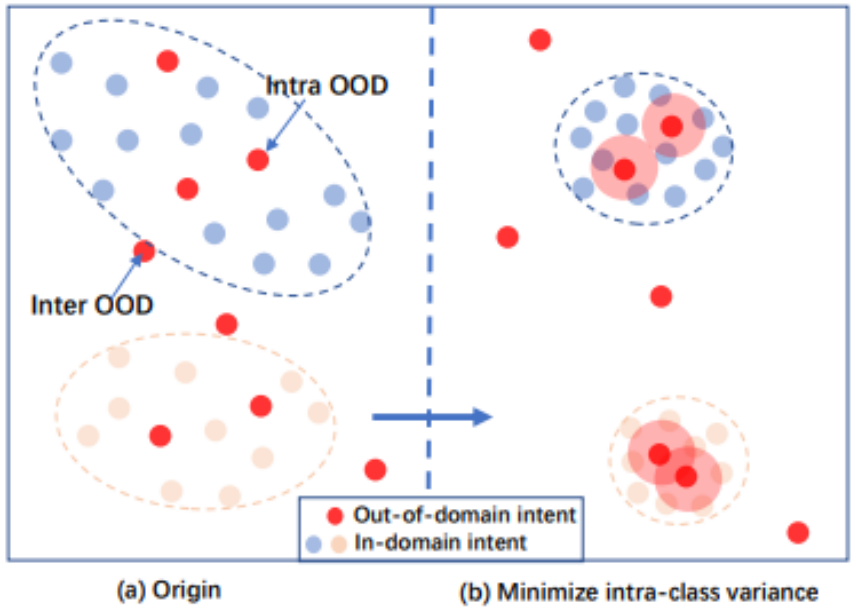
Out-of-Domain Intent Classification

- 대화시스템에서 중요한 문제
  - 기존 intent detection은 학습한 class에 대해서만 출력
  - Closed world 가정
- 2가지 종류
  - Training 동안 OOD sample 사용
  - **Discriminative semantic feature와 OOD detection 두단계로 나누어 training**
    - **IND 와 OOD intent 사이 margin 최대화를 목표**
    - **Gaussian distribution 사용**



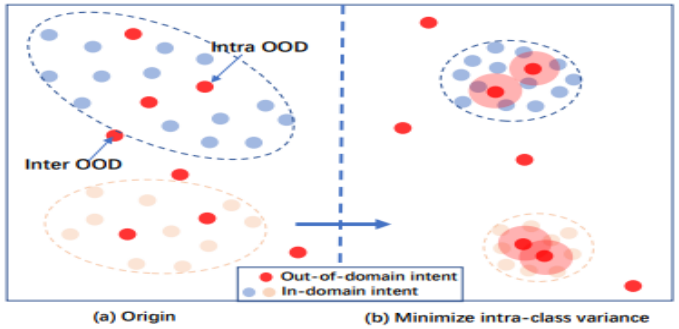
용어 정의

- **Inter OOD intent**
  - 다른 IND class 사이에 분포된 OOD intent
- **Intra OOD intent**
  - IND 분포 내에 있는 OOD intent



Introduction
존재하는 방법들의 제한 요소 분석

Dataset	Classes	*Rate**
CLINC-FULL	150	57%
CLINC-SMALL	150	36%
StackOverflow	20	21%



- OOD intent 가 서로 다른 IND class 사이에만 존재한다고 생각
  - **문제점: OOD 위치는 제한되지 않음**
- Inter OOD intent를 위해 intra-class variance를 최소화하고 inter-class variance를 최대화하는 것은 IND를 식별하는 것에 대한 risk 감소 가능
  - **문제점 : intra OOD 가 IND로 식별되는 risk 증가**
- **Gaussian distribution**
  - SCL loss로 학습된 IND semantic feature 분포에 대한 Gaussian Hypothesis Test 수행
    - 57% IND class 만이 Gaussian distribution 따름
    - **Gaussian distribution의 합리성 의심**

Gaussian Hypothesis Test

- NomalTest 사용
  - Dataset의 분포가 정규분포를 따르는 지 검정하는 것
- SCL loss로 학습

$$\mathcal{L}_{SCL} = \sum_{i=1}^{N} -\frac{1}{N_{y_i} - 1} \sum_{j=1}^{N} \mathbf{1}_{i \neq j} \mathbf{1}_{y_i = y_j}$$
$$\log \frac{\exp\left(s_i \cdot s_j / \tau\right)}{\sum_{k=1}^{N} \mathbf{1}_{i \neq k} \exp\left(s_i \cdot s_k / \tau\right)}$$

  - Temperature hyperparameter τ
    - Hard negative sample 의 penalty 의 강도를 조절하는 역할

Proposed Method

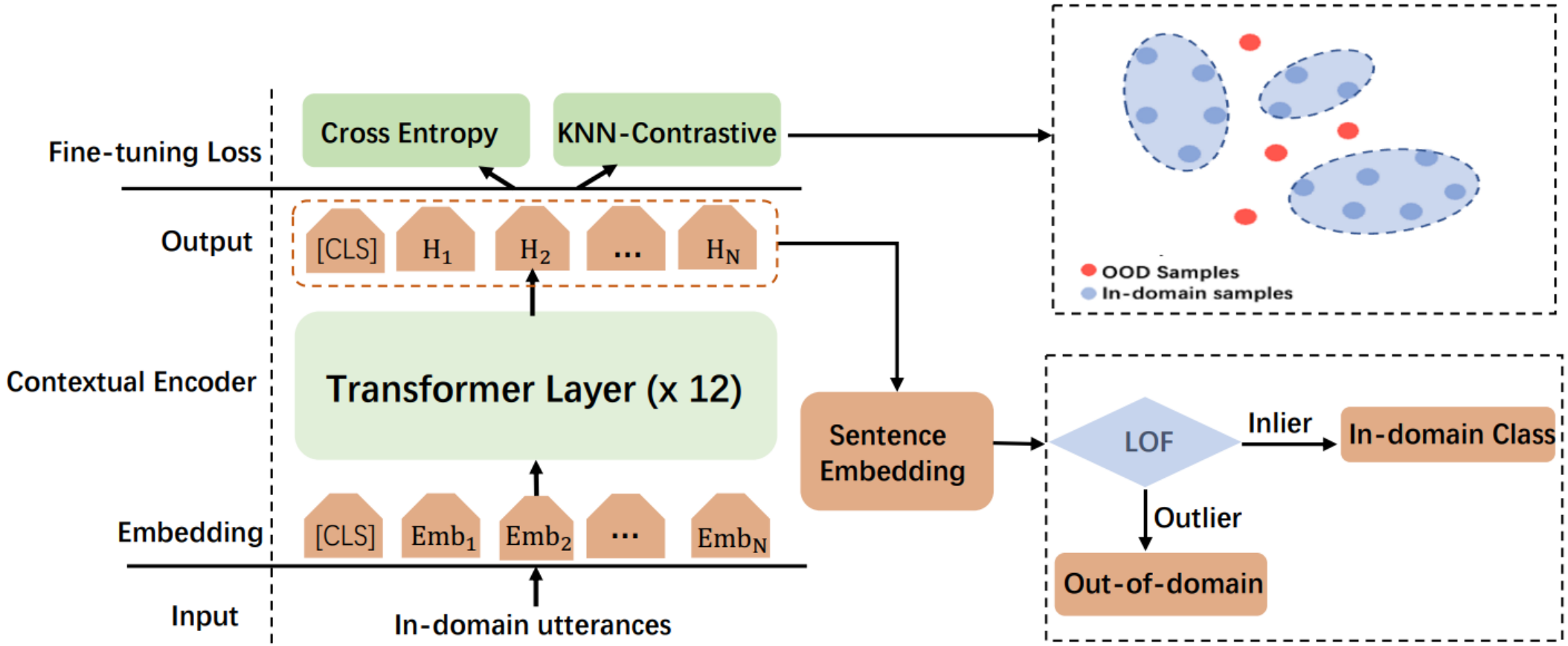


Figure 3: The architecture of our proposed model.

Proposed method

objective of OOD intent classification

- **Open space risk**
  - Open space $\mathcal{O} = \mathcal{S} - \bigcup_{x \in \mathcal{X}} \sigma(x),$ (Local semantic space spanned by x; $\mathcal{X}$: 모든 IND training 샘플 set)
  - Open space risk $\mathcal{R}_{\mathcal{O}}(f) = \frac{\int_{\mathcal{O}} f(x,\theta)dx}{\int_{\mathcal{S}} f(x,\theta)dx}$
- **Objective of OOD intent classification**
  - OOD intent 식별위해 intent representation 학습필요
    - Downstream detection 적용
    - IND 분류 quality 보장
  - **optimization objective** $\arg\min_{f(x,\theta)\in\mathcal{H}} \{(1-\lambda)\cdot\mathcal{R}_{\varepsilon}(f) + \lambda\cdot\mathcal{R}_{\mathcal{O}}(f)\}$

Proposed method

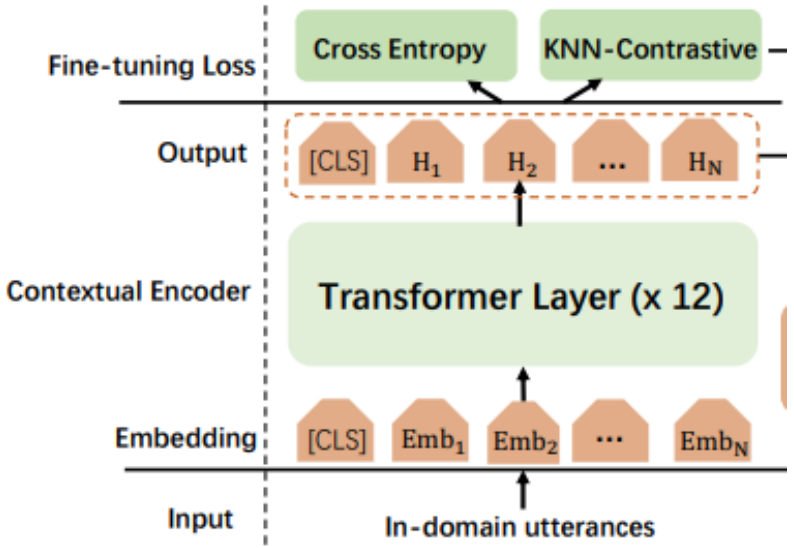
minimize empirical risk

- Intent representation 추출위해 BERT 사용

$$Z_i = \text{Mean-Pooling}([[\text{CLS}], T_1, ...T_N])$$

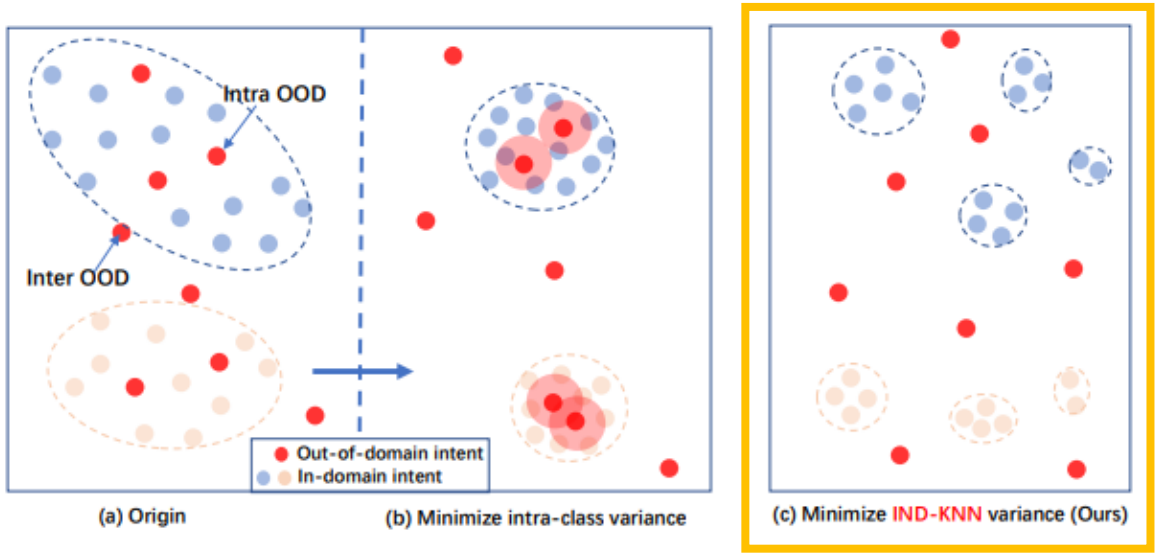
- Optimize the empirical risk

$$\mathcal{L}_{\text{ce}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(\phi_{y_i}(z_i))}{\sum_{j\in[K]}\exp(\phi_j(z_i))}$$



Proposed method

KNN-Contrastive learning



- **Optimize the open space risk**
- 이전 방식
  - 같은 class IND sample 당기고 다른 class sample 밀어내는 방식
  - 문제점: intra OOD intent를 IND로 식별하는 risk 증가
- 다른 class 밀어내는 동시에 **KNN만 당김**
  - **KNN-contrastive loss**
    - IND intent sample 의 KNN을 positive sample로 생각

$$\mathcal{L}_{\text{knn-cl}} = \sum_{i=1}^{N} \frac{1}{|\mathcal{X}_k|} \sum_{z_j \in \mathcal{X}_k} -\log \frac{\exp(\frac{z_i \cdot z_j}{\tau})}{\sum_{z_q \in I} \exp(\frac{z_q \cdot z_i}{\tau})}$$

Proposed method

Momentum contrast is All You Need

- KNN-contrastive learning 수행 시 2 가지 문제 존재
  - **Large batch size**
  - **KNN-contrastive train learning은 불안정**
    - KNN은 Training동안 consistency 유지 필요
- Momentum Contrast(MoCo) 에서도 똑같은 문제를 queue 로 해결

Proposed method

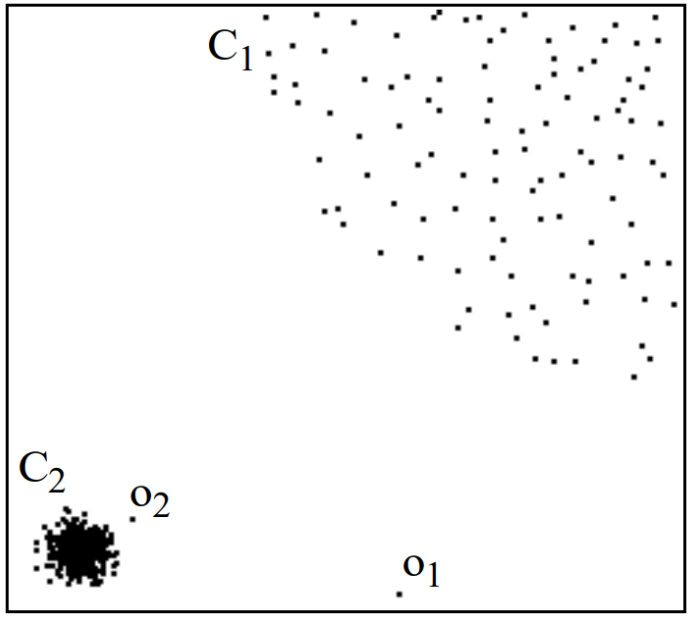
Momentum contrast is All You Need

- **Queue 사용**
  - IND sample 담고 있는 queue 가 현재 batch에서 가장 오래된 feature 빼내어 업데이트
  - Queue 는 batch size 보다 커 더 많은 negative sample 얻는 것 도움
- **Momentum base**
  - Consistency 유지 위해 이전 batch feature 느리게 update(encoder)
- **Final finetune objective**
  - Softmax cross-entropy loss와 KNN-contrastive learning loss 합침

$$\mathcal{L}_{obj} = \lambda \cdot \mathcal{L}_{\text{knn-cl}} + (1 - \lambda) \cdot \mathcal{L}_{\text{ce}}$$

Proposed method

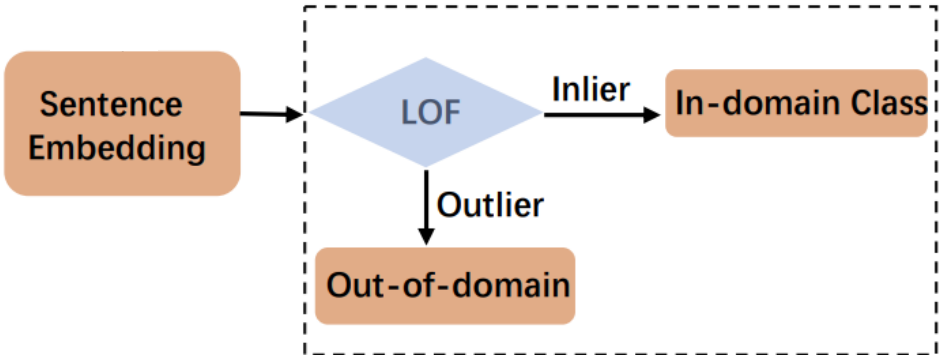
Local Outlier Factor



- **LOF algorithm 사용**
  - 주변 데이터 밀도를 고려한 이상치 탐지 알고리즘
  - 정상 샘플은 주로 주변에 데이터가 많이 존재하면, unknown은 주로 단독으로 존재함을 가정

- KNN 과 비교하여 LOF score 더 낮으면 OOD 높으면 IND

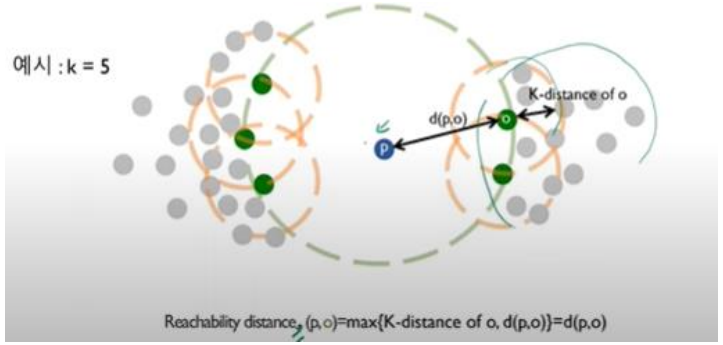
$$\mathrm{LOF}_k(z) = \frac{\sum_{p \in N_k(z)} \frac{lrd(p)}{lrd(z)}}{|N_k(z)|}$$

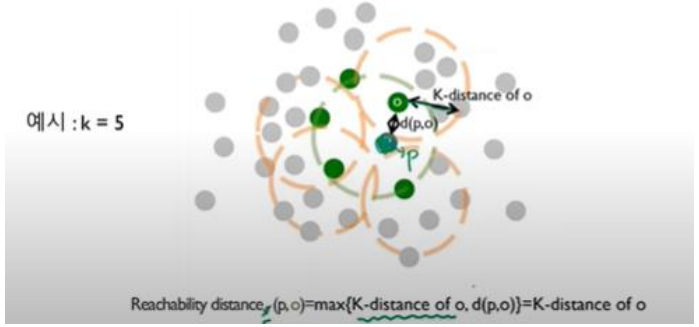


LOF

- K-distance of object p
  - 자기 자신 (p)를 제외하고 k번째로 가까운 이웃과의 거리
- K-distance neighborhood of object p $N_k(p)$
  - K번째로 가까운 이웃과의 거리를 원으로 표현할 때,
    원 안에 포함되는 모든 객체들의 수
- Reachability distance
  - O를 기준으로 k번째 가까운 이웃과의 거리와 o와 p사이 거리 간의 최대 값

$$reach\text{-}dist_k(p, o) = \max \{ k\text{-distance}(o), d(p, o) \}$$





LOF

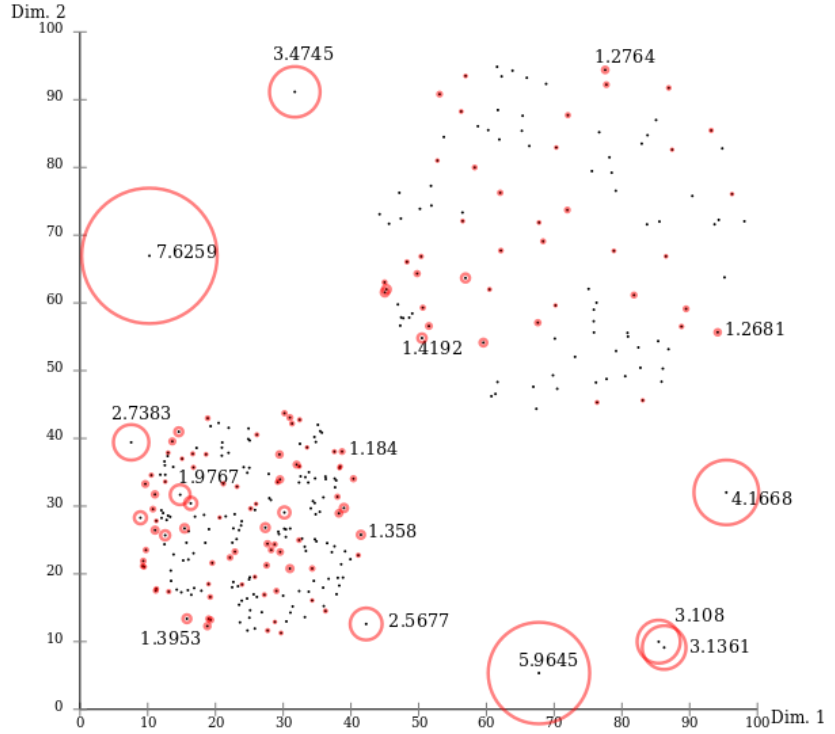
Case	$ldr_k(A)$	$ldr_k(B)$	$LOF_k(A)$
Case 1	Large	Large	Small
Case 2	Small	Large	Large
Case 3	Small	Small	Small

- Local reachability density(lrd)
  - 여러 reachability distance를 하나의 지표로 계산한 값

$$lrd_k(p) = 1/\frac{\sum_{O \in N_k(p)} \text{reach-dist}_k(p, o)}{|N_k(p)|}$$

- Local outlier factor
  - 자기 자신의 최종 local outlier factor 값 계산

$$\text{LOF}_k(z) = \frac{\sum_{p \in N_k(z)} \frac{lrd(p)}{lrd(z)}}{|N_k(z)|}$$



Experiments

Dataset	Class	#Training	#Validation	#Test	Vocabulary Size	Length (Avg)
CLINC-FULL	150	15100	3100	5500	8288	8.32
CLINC-SMALL	150	7600	3100	5500	7017	8.31
BANKING	77	9003	1000	3080	5028	11.91
StackOverflow	20	12000	2000	6000	17182	9.18

- Dataset
  - **CLINC-FULL**
    - OOD detection 위한 dataset
  - **CLINIC-SMALL**
    - FULL에서 각 IND class 마다 50개만 사용
  - **BANKING**
    - 은행 관련 dataset
  - **StackOverflow**
    - Technical question dataset

- **Experimental settings**
  - Dataset
    - Train, validation, test
  - **The number of known classes**
    - 25%, 50%, 75%
    - 나머지 class 는 open 으로 여기고 train 때는 제거
    - Test 때는 전체 label 사용
  - Model
    - BERT
  - Queue : 6500 7500, 8000
  - Momentum update parameter m = 0.999
  - Optimixer: AdamW
  - Learning rate: 1e-5, 2e-5
  - Batch size : 16, 32

Main Results

	Methods	BANKING				StackOverflow			
		ACC-ALL	F1-ALL	F1-OOD	F1-IND	ACC-ALL	F1-ALL	F1-OOD	F1-IND
25%	MSP†	43.67	50.09	41.43	50.55	28.67	37.85	13.03	42.82
	DOC†	56.99	58.03	61.42	57.85	42.74	47.73	41.25	49.02
	OpenMax†	49.94	54.14	51.32	54.28	40.28	45.98	36.41	47.89
	Softmax∗	57.88	58.32	62.52	58.10	46.17	50.78	42.52	51.83
	LMCL†	64.21	61.36	70.44	60.88	47.84	52.05	49.29	52.60
	SEG∗	51.11	55.68	53.22	55.81	47.00	52.83	46.17	54.16
	ADB†	78.85	71.62	84.56	70.94	86.72	80.83	90.88	78.82
	SCL+GDA‡	83.87	67.94	89.44	66.81	82.29	70.92	88.99	67.44
	SCL+LOF‡	84.05	74.86	89.01	74.12	80.10	78.51	84.45	77.32
	Ours	**85.62**	**77.13**	**90.19**	**76.44**	**89.04**	**81.61**	**92.7**	**79.39**
50%	MSP†	59.73	71.18	41.19	71.97	52.42	63.01	23.99	66.91
	DOC†	64.81	73.12	55.14	73.59	52.53	62.84	25.44	66.58
	OpenMax†	65.31	74.24	54.33	74.76	60.35	68.18	45.00	70.49
	Softmax∗	67.44	74.19	60.28	74.56	65.96	71.94	56.80	73.45
	LMCL†	72.73	77.53	69.53	77.74	58.98	68.01	43.01	70.51
	SEG∗	68.44	76.48	60.42	76.90	68.50	74.18	60.89	75.51
	ADB†	78.86	80.90	78.44	80.96	86.40	85.83	87.34	85.68
	SCL+GDA‡	79.38	79.84	79.97	79.83	82.31	79.54	84.42	79.04
	SCL+LOF‡	80.54	82.4	80.42	82.6	84.47	84.57	85.01	84.53
	Ours	**83.14**	**83.87**	**83.58**	**83.88**	**87.62**	**87.18**	**88.36**	**87.06**
75%	MSP†	75.89	83.60	39.23	84.36	72.17	77.95	33.96	80.88
	DOC†	76.77	83.34	50.60	83.91	68.91	75.06	16.76	78.95
	OpenMax†	77.45	84.07	50.85	84.64	74.42	79.78	44.87	82.11
	Softmax∗	78.20	84.31	56.90	84.78	77.41	82.28	54.07	84.11
	LMCL†	78.52	84.31	58.54	84.75	72.33	78.28	37.59	81.00
	SEG∗	78.87	85.66	54.43	86.20	80.83	84.78	62.30	86.28
	ADB†	81.08	85.96	66.47	86.29	82.78	85.99	73.86	86.80
	SCL+GDA‡	79.86	85.14	64.49	85.5	80.88	84.79	68.83	85.86
	SCL+LOF‡	81.56	86.97	65.05	87.35	80.92	83.98	71.71	84.79
	Ours	**81.77**	**87.07**	**67.66**	**87.41**	**83.85**	**87.06**	**74.20**	**87.92**

Table 1: Results of OOD classificaion with different IND classes rate (25%, 50% and 75%) on BANKING and StackOverflow. The baseline with † are retrieved from (Zhang et al., 2021), results with ∗ are from (Zhan et al., 2021) and ‡ means the results is not provided in the original paper (Zeng et al., 2021a), and we get the results by running its released code.

Main Results

Methods	CLINC-FULL				CLINC-SMALL			
	ACC-ALL	F1-ALL	F1-IND	F1-OOD	ACC-ALL	F1-ALL	F1-IND	F1-OOD
Softmax	-	-	88.98	66.17	-	-	86.20	62.58
LMCL	-	-	89.12	66.80	-	-	86.64	63.11
SCL+GDA	-	-	90.03	68.21	-	-	88.30	65.01
SCL+LOF†	84.87	88.51	88.63	70.05	84.12	87.47	87.58	70.31
Ours	**89.45**	**92.5**	**92.61**	**76.36**	**88.62**	**91.82**	**91.92**	**75.74**

Table 2: Results of OOD classification on CLINC-FULL and CLINC-SMALL. † means the results is not provided in the original paper and we get the results by running its released codes provided by (Zeng et al., 2021a). Other baselines are from (Zeng et al., 2021a).

Main Results

- **Optimize Empirical risk 확인**
  - Whisper-mode(green)
  - Cancel(blue)
- **Optimize open space risk 확인**
  - Vaccines class(brown)

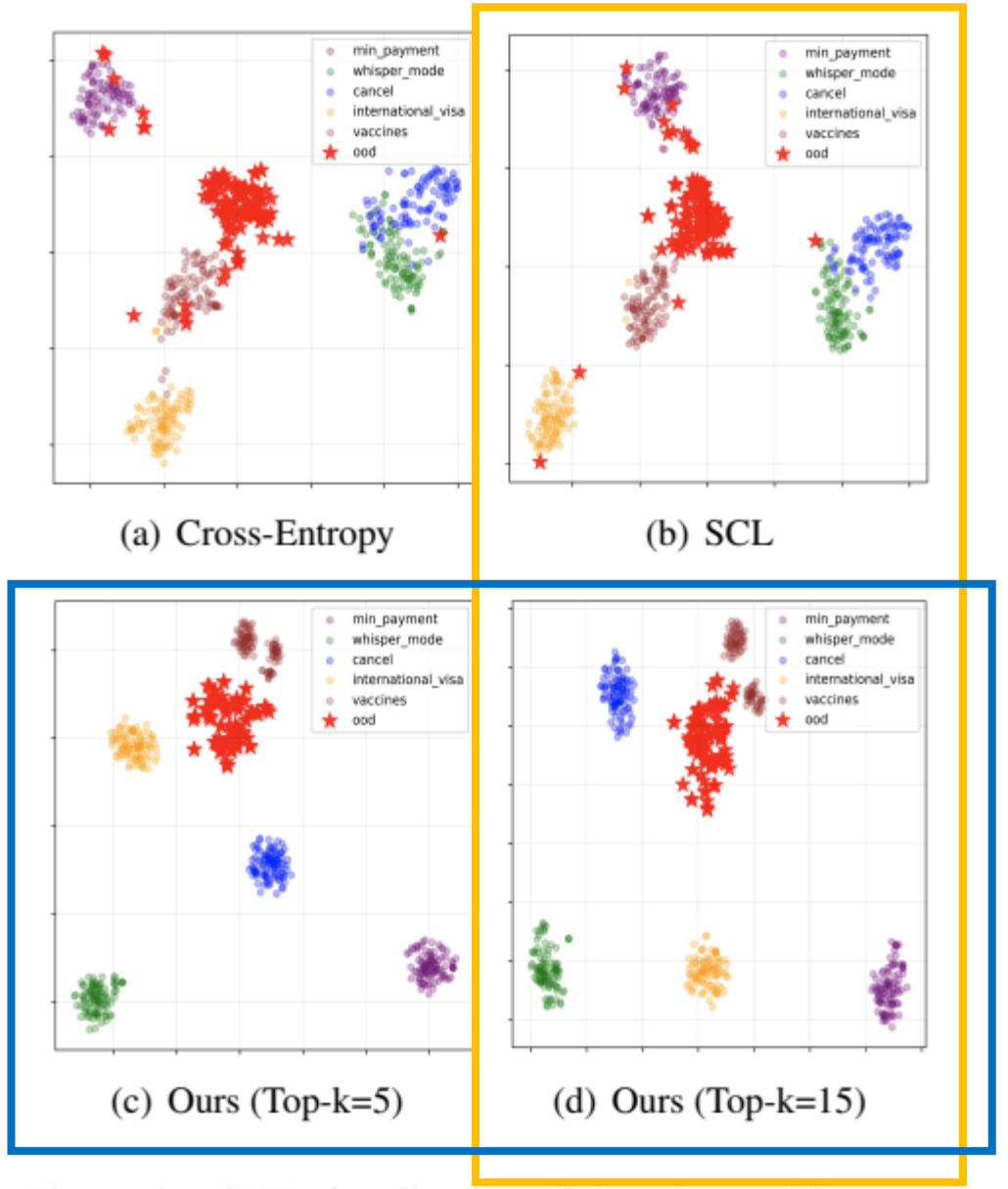


Figure 4: t-SNE visualization of deep learned features of some intent samples in CLINC-FULL. Top-k means we select the k-nearest neighbors of IND samples for constrastive learning.

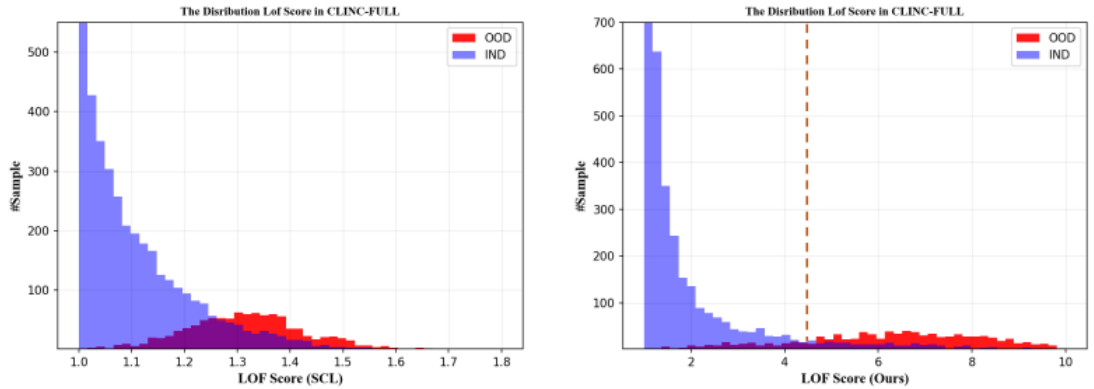
Effect of Distance on Calculation of LOF

- Two distance
  - Cosine
  - Euclidean

	BANKING(75%)		StackOverflow(75%)	
	F1-OOD	F1-ALL	F1-OOD	F1-ALL
Lof Cosine	67.36	86.87	78.12	88.06
Lof Euclidean	65.77	87.46	77.12	87.9

- 분석 결과
  - **Single dataset(같은 data끼리)**
    - 성능 차이 존재(큰 차이는 아님)
    - 원인: threshold – ood sample 정보를 사용할 수 없어 정확한 threshold 얻을 수 없음
  - **같은 방법 다른 data**
    - 더 좋은 방법이 무엇인지 관찰 불가
    - 실험 방법: 다른 random seed 로 3번 test 후 평균값
- CLINC 에서 Euclidean 이 score 대한 threshold 높이면 성능 더 좋음

Visualize the ability to detect OOD



(a) SCL (b) Ours

Figure 5: The LOF score distrubtion in CLINC-FULL. Left: SCL training. Right: Our proposed method training. The X-axis represents the LOF score, and the Y-axis represents the number of samples falling into the corresponding interval.

Conclusion

- Optimization objective of OOD intent classification 제시
- 기존 방법의 한계 분석
- discriminative semantic feature 학습 위한 방법 제안
  - IND intent의 KNN을 끌어당겨 open space risk와 empirical risk 감소
  - feature distribution의 제한 없이 개선 가능

Thank you

Supervised Contrastive Learning

- **Supervised Contrastive Learning (Loss)**

$$L = \sum_{i \in I} L_i^{sup} = \sum_{i \in I} -log \left\{ \frac{1}{|P(i)|} \sum_{p \in P(i)} \frac{exp(z_i \cdot z_p / \tau)}{\sum_{a \in A(i)} exp(z_i \cdot z_a / \tau)} \right\},$$

- i: Anchor (학습 대상 데이터)
- $i \in I \equiv \{1, \ldots, 2N\}$ → 학습 데이터 N개 + 증강 데이터 N개 (multi-viewed data)
- $P(i) \equiv \{p \in A(i) : \tilde{y}_p = \tilde{y}_i\}$ = set of indices of all positive samples
  - Positive sample 이 여러 개 → self-supervised contrastive learning은 positive 가 1개
  - Positive samples $P(i)$:Anchor의 증강 데이터,Anchor와 같은 레이블 데이터,
  - $|P(i)|$ = cardinality (the number of positive samples)
- $a \in A(i) \equiv I \setminus \{i\}$ (anchor 외 모든 데이터 = $|P(i)|$ 개 positives + 2N − $|P(i)|$개 negatives)

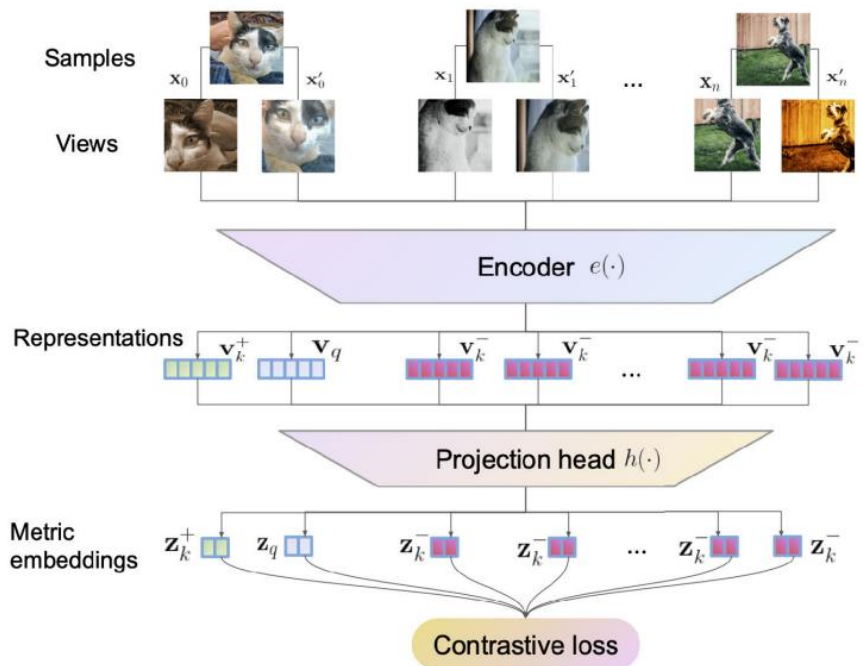
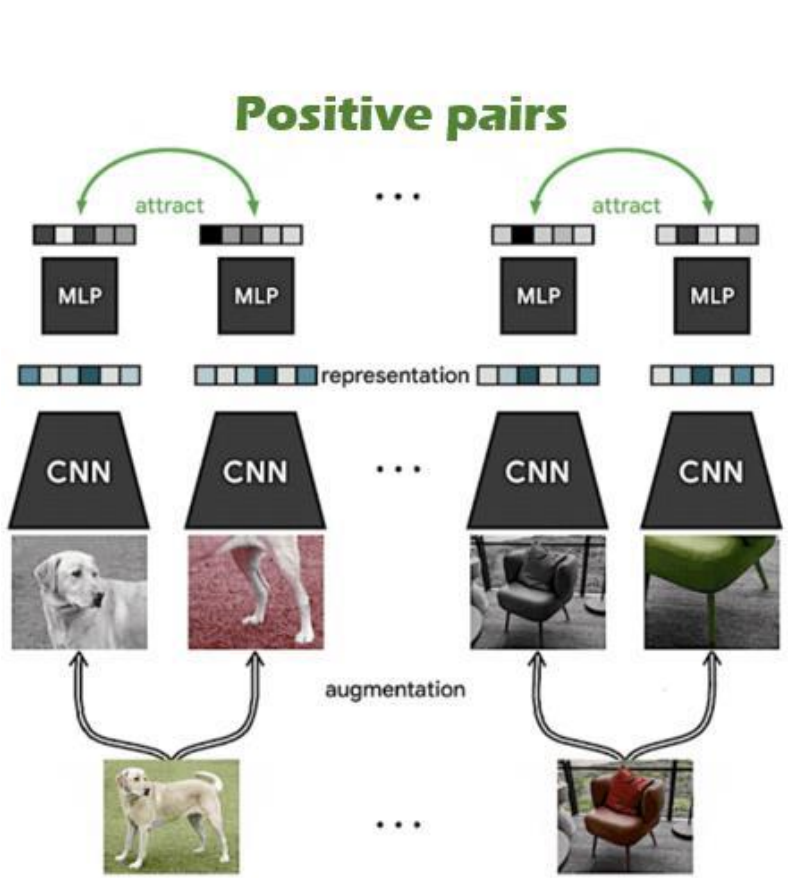
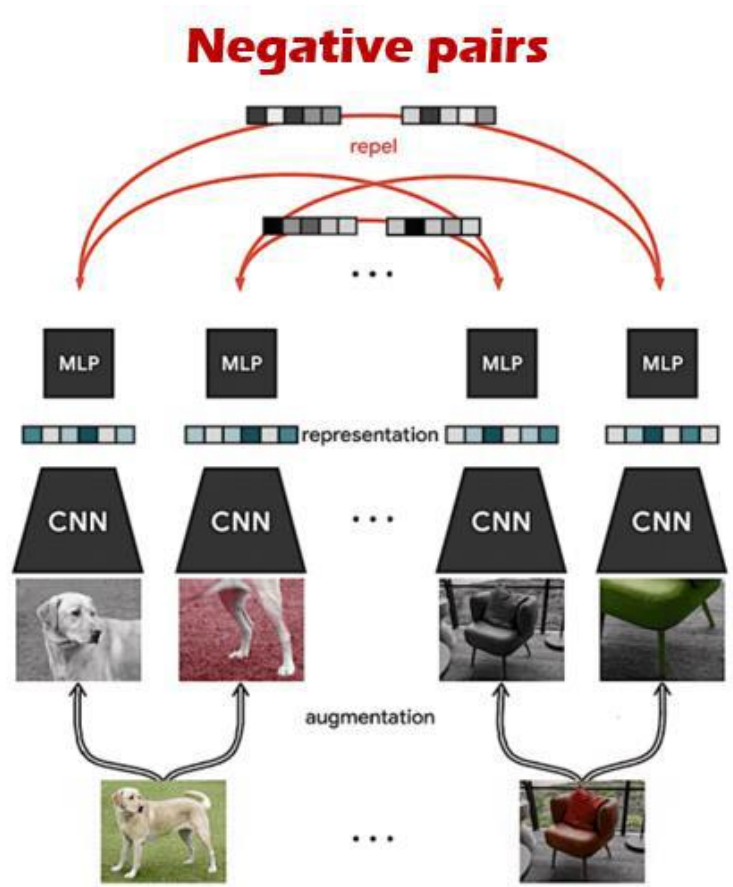
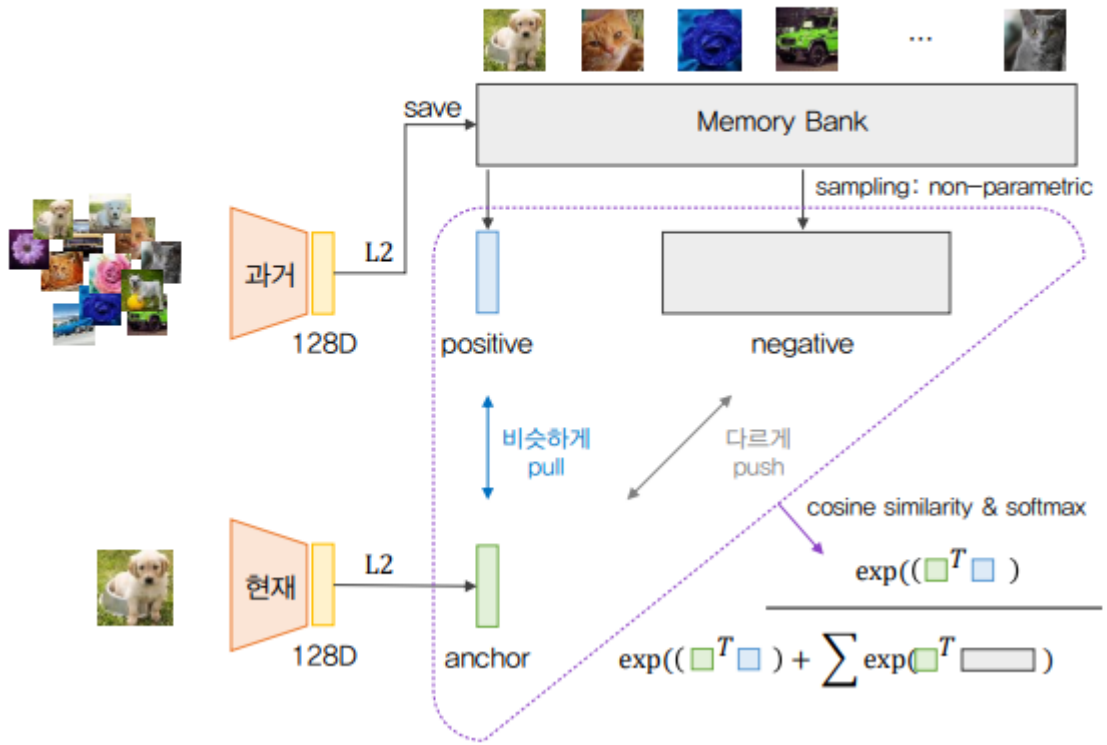


FIGURE 2. **Contrastive learning in the Instance Discrimination pretext task for self-supervised visual representation learning. A positive pair is created from two randomly augmented views of the same image, while negative pairs are created from views of two different images. All views are encoded by the a shared encoder and projection heads before the representations are evaluated by the contrastive loss function.**


Positive pairs
attract
attract
MLP
MLP
MLP
MLP
representation
CNN
CNN
CNN
CNN
augmentation


Negative pairs
repel
MLP
MLP
MLP
MLP
representation
CNN
CNN
CNN
CNN
augmentation

01 | Intuitive Understanding



① Instance discrimination을 하기 위해 모든 데이터의 임베딩 정보를 저장한다. 이때 임베딩 크기를 맞추기 위해 L2-normalization을 수행한다.

② 현재 타겟 이미지의 과거 임베딩 벡터 & 나머지 이미지들의 일부 임베딩 벡터를 샘플링

③ 비슷한 정도를 측정하기 위해 cosine 유사도 측정 (L2 Euclidean – Cosine)

④ 나 자신에 대한 유사도는 높이며, 나머지는 내리고 싶다. Cosine 유사도를 logit으로 사용하고 확률로 표현하기 위해 softmax 적용 (NCE) → $P(i|\boldsymbol{v})$

⑤ Loss: 앞에서 얻은 cosine 유사도를 logit으로 사용하고, 타겟 클래스 레이블을 모두 0으로 cross entropy. (minimize negative log-likelihood)

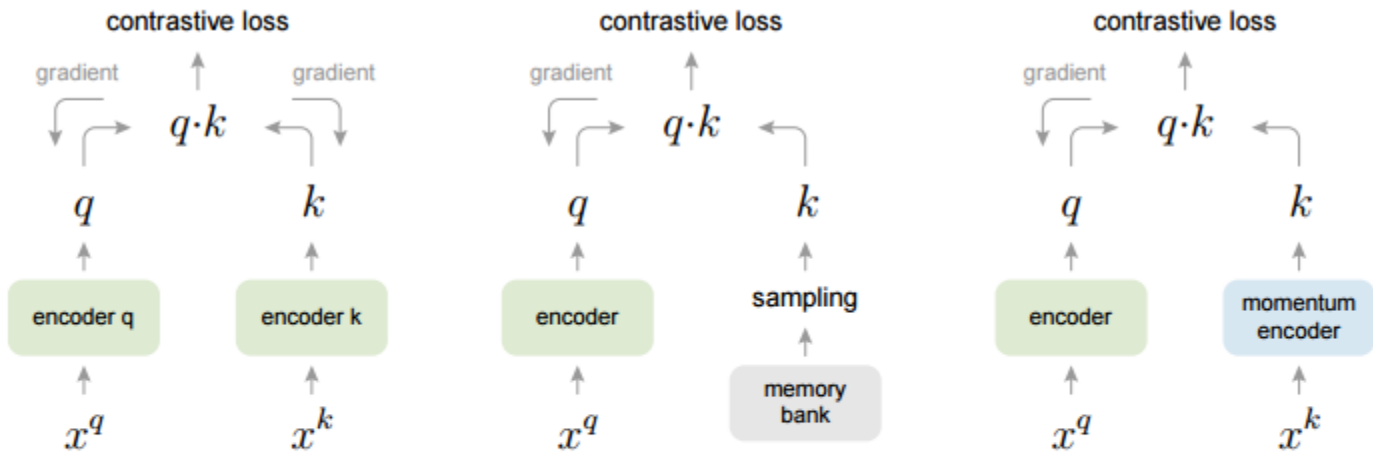


Figure 2. **Conceptual comparison of three contrastive loss mechanisms** (empirical comparisons are in Figure 3 and Table 3). Here we illustrate one pair of query and key. The three mechanisms differ in how the keys are maintained and how the key encoder is updated. **(a)**: The encoders for computing the query and key representations are updated ***end-to-end*** by back-propagation (the two encoders can be different). **(b)**: The key representations are sampled from a ***memory bank*** [61]. **(c)**: ***MoCo*** encodes the new keys on-the-fly by a momentum-updated encoder, and maintains a queue (not illustrated in this figure) of keys.

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i / \tau)}$$

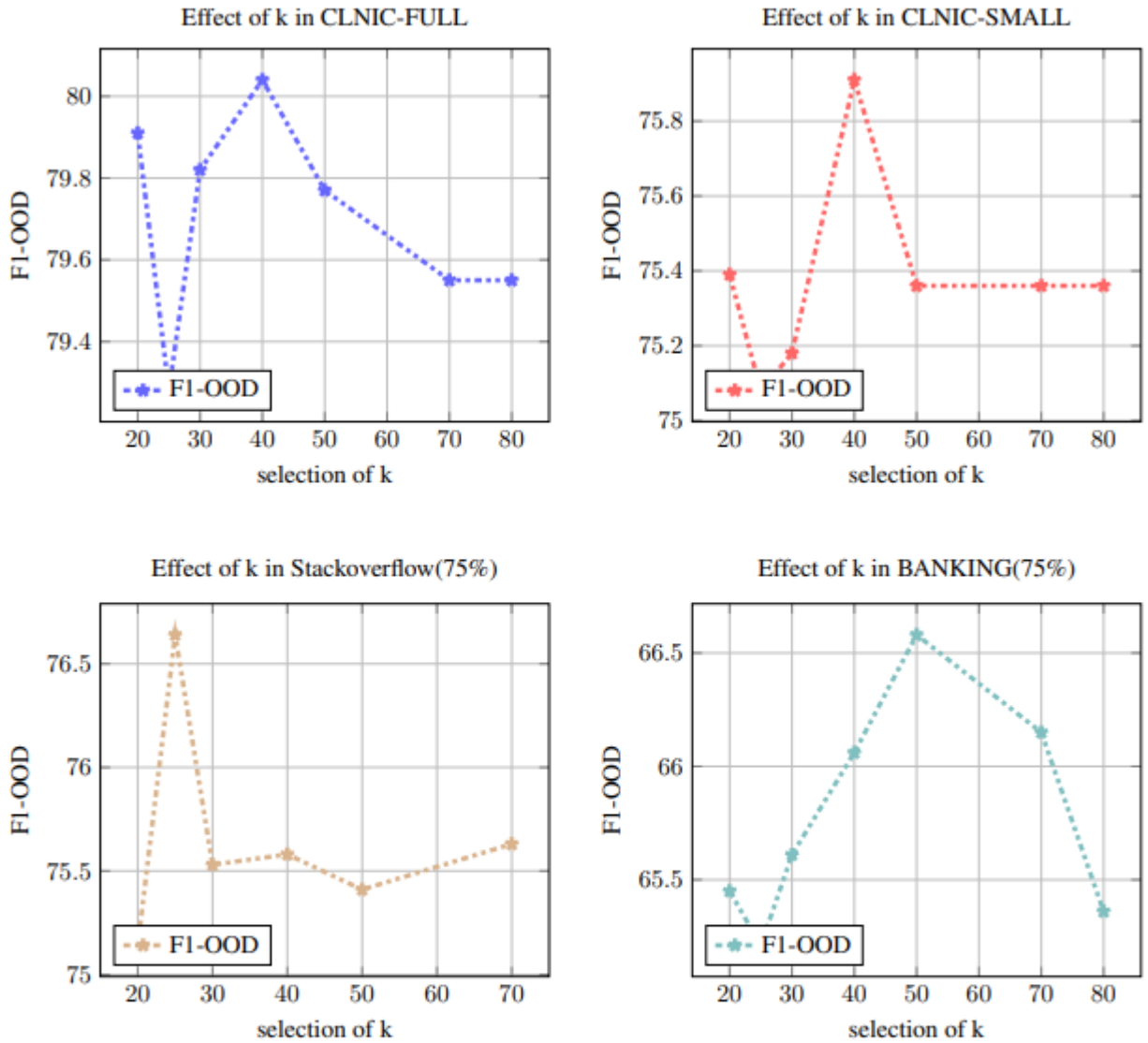


Figure 6: Effect of k. The X-axis represents the value of k, and Y-axis represents F1-score for OOD samples.

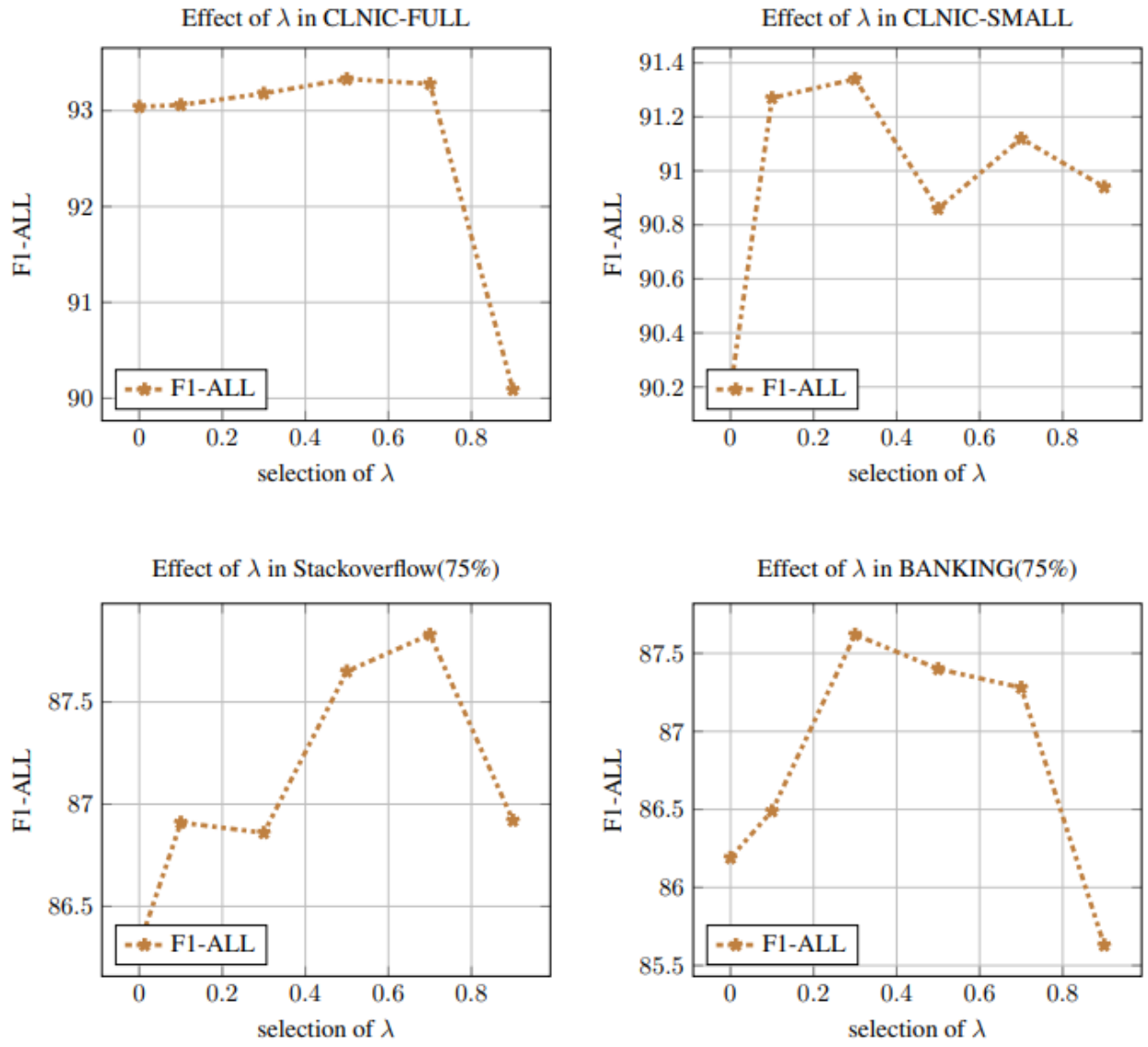


Figure 7: Effect of λ on four datasets. The X-axis represents the λ value, and Y-axis represents F1-ALL for all classes.