10. Low-Shot Learning with Imprinted Weights

논문 : https://openaccess.thecvf.com/content_cvpr_2018/papers/Qi_Low-Shot_Learning_With_CVPR_2018_paper.pdf

요약

- Metric based Meta-learning
- 8. Low-shot Visual Recognition by Shrinking and Hallucinating Features 과 같이 풍부한 데이터로 pre-train한 뒤 novel class가 있는 적은 양의 테스트셋의 정확도를 향상시킬수 있는 기법 제안
- 마지막에 Fully Connected Layer가 아닌 클래스별 proxy와 비교해서(내적해서) 가장 가까운 proxy의 class로 inference
- novel class의 feature representation을 proxy로 사용해 쉽게 늘리고 줄일수 있는 형태를 제안

제안기법

- Neighbourhood Components Analysis(NCA) 논문에서 제안한 triplet-based embedding training + softmax classification은 식 1과 같음
  - Distance Function **d**는 squred Euclidean distance를 사용
  - Data **x**의 True Label Data Point **y**가 Negative Data Point **Z**보다 가까워지도록 학습됨

$$\mathcal{L}_{\text{NCA}}(x, y, Z) = -\log \frac{\exp(-d(x, y))}{\sum_{z \in Z} \exp(-d(x, z))} \qquad (1)$$

- No Fuss Distance Metric Learning using Proxies 논문이 제안한 Loss Function은 식 2와 같음
  - Data x와 Class 별 대표 Proxy와 비교하여 True Label Proxy Point와 가까워지도록 학습됨
  - 논문은 식 2를 softmax classifier와 묶어서 사용하고자함

$$\begin{aligned}\mathcal{L}_{\text{proxy}}(x) &\triangleq \mathcal{L}_{\text{NCA}}(x, p(x), p(Z)) \\ &= -\log \frac{\exp(-d(x, p(x)))}{\sum_{p(z) \in p(Z)} \exp(-d(x, p(z)))}, \qquad (2)\end{aligned}$$

기호	설명
C	Set of category label function c로 data point x의 true category를 얻음 $c(x) \in C$
P	Set of proxy point $P = \{p_1, p_2, \ldots, p_{\lvert C \rvert}\}$ p는 trainable vector data point x의 true label proxy point는 아래와 같이 표기 $p(x) = p_{c(x)}$

Connections to Softmax Classifier

- category c에 대한 proxy point p_c 가 softmax classifier의 weight w_c 와 유사하다고 함
- 식 3) proxy point p와 data point x가 같은 dimension으로 normalize 되있다고 가정했을 때,
  - proxy point와 data point의 squared Euclidean distance를 최소화
    = proxy point와 data point의 inner-product를 최대화 하는 것
    = proxy point와 data point의 cosine-similarity를 최대화 하는것
  - Euclidean distance를 풀어보면

$u, v \in \mathbb{R}^D$ 일 때, $||u-v||_2^2 = 2-2u^\top v$ 로 표현가능~

$$\min d(x, p(x)) \triangleq \min ||x - p(x)||_2^2 = \max x^\top p(x), \quad (3)$$

- 식 4) 식 2의 distance function 부분을 inner-product 형태로 변경한 식

$$\mathcal{L}(x, c(x)) = -\log \frac{\exp(x^\top p_{c(x)})}{\sum_{c \in C} \exp(x^\top p_c)}, \quad (4)$$

- 식 5) 식 4를 classifier 학습에 사용하는 cross-entropy loss 처럼 표현
  - 초기에 모든 클래스의 bias b는 0으로 설정

$$\mathcal{L}_{\text{softmax}}(x, c(x)) = -\log \frac{\exp(x^\top w_{c(x)} + b_{c(x)})}{\sum_{c \in C} \exp(x^\top w_c + b_c)}, \quad (5)$$

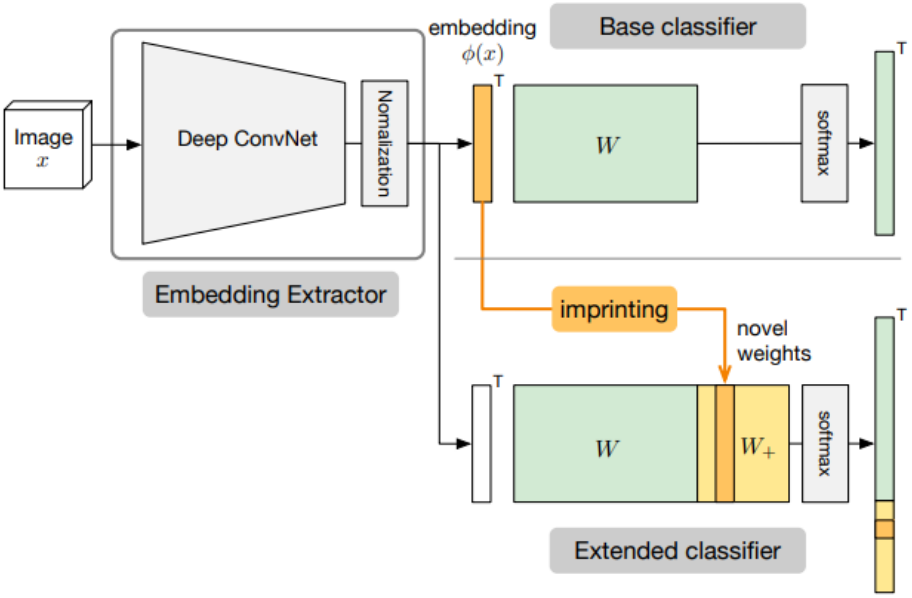


Figure 1. The overall architecture of imprinting. After a base classifier is trained, the embedding vectors of new low-shot examples are used to imprint weights for new classes in the extended classifier.

Imprinting

- 모델 구조 (그림 1)
  - Embedding Extractor
    - 데이터를 입력으로 Feature Representation $\phi(x)$ 연산
    - 논문에서는 ConvNet Architecture를 채용
    - 추가로 마지막 레이어에 L2-Norm을 추가하여 output이 unit length를 가지게함

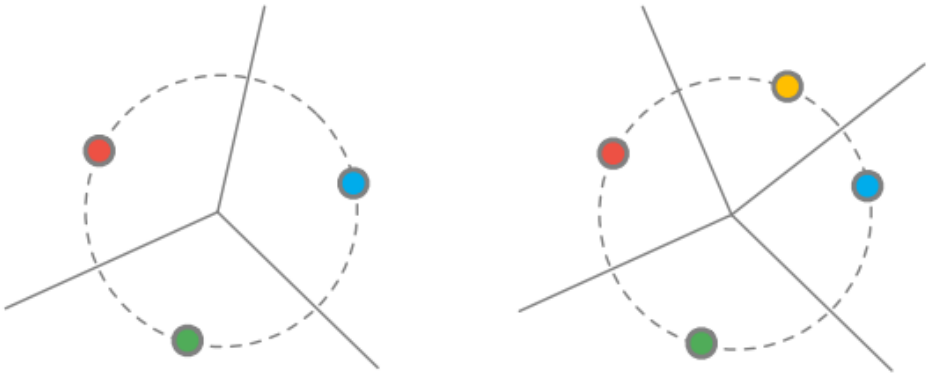
$$||\phi(x)||_2 = 1$$

  - Softmax Classifier
    - unnormalized logt score또한 softmax로 표현할 수 있음 **(식 6)**
      - 이때, weight matrix의 i번째 컬럼 w_i 은 unit length로 normalize 된 상태
        - Feature Representation $\phi(x)$ 도 normalize되있는 상태기 때문에 **식 3**에 의해 내적 값이 큰 것이 euclidean distance 값이 작아지도록하는 것과 동일함! **(식 7)**
      - 해당 레이어에는 bias를 추가하지 않음
      - No Fuss Distance Metric Learning using Proxies는 Proxy를 사용하는 Classifier를 학습할때만 사용하는데 우리 모델은 버리지않고 끝까지 사용!
      - K-NN는 데이터 셋 전체를 사용하는 것에 비해 논문은 대표 데이터만 사용
    - Normalization
      - **제안한 모델에서 중요한 것 : feature representation과 classifier의 weight matrix를 컬럼별로 normalize하는 것**
      - 일반적으로 최적화를 위해 mini-batch나 layer에서 activation을 평균 0 분산 1(unit)이 되도록 권장하지만 neuron activation과 weight 간 규모 차이를 해결하지 않음
      - 논문은 weight matrix의 컬럼과 feature representation을 normalize하기 때문에 규모의 차이가 없고, 내적의 결과가 항상 cosine similiarity 처럼 $w_i^\top \phi(x) \in [-1, 1]$ 를 만족하게됨
    - Scale factor (식 8)
      - normalize로 인해 내적값은 항상 $w_i^\top \phi(x) \in [-1, 1]$ 를 만족하는데 클래스 수가 많아질수록 True-Label의 최대값이 점점 낮아짐
        - ex) 100개의 클래스 중 True-label의 내적이 1, 나머지 99개가 -1로 나왔을 때, True-label에 대한 확률이 0.069이 나오게됨
      - 1에 도달할수 없는 문제가 있으니 내적 결과에 적절한 scaling이 필요
        - 실험적으로 클래스별 scale factor를 달리해봤으나 별로 효과없음

$$f_i(\phi(x)) = \frac{\exp(w_i^\top \phi(x))}{\sum_c \exp(w_c^\top \phi(x))}, \qquad (6)$$

$$\hat{y} = \arg\max_{c \in C} w_c^\top \phi(x) = \arg\min_{c \in C} d(\phi(x), w_c). \qquad (7)$$

$$f_i(\phi(x)) = \frac{\exp(s w_i^\top \phi(x))}{\sum_c \exp(s w_c^\top \phi(x))}. \qquad (8)$$



(a) (b)

Figure 2. Illustration of imprinting in the normalized embedding space. (a) Before imprinting, the decision boundaries are determined by the trained weights. (b) With imprinting, the embedding of an example (the yellow point) from a novel class defines a new region.

- Weight Imprinting
  - 그림 1처럼 novel class의 feature representation을 기존의 weight matrix에 추가하는 방법론 제안
  - 그림 2처럼 novel class(노란색)이 적절하게 떨어져서 배치되는것이 희망
  - Averaging embedding
    - novel class 별 feature representation들을 element-wise average하는 것
    - Unit lenght로 만들기 위해 한번더 re-normalizing
    - 장점 : 재학습이 없어서 계산량이 적음
    - 단점 : base domain에 대해서 학습한 모델의 output을 사용하는 것이기 때문에 정확도는 떨어짐
  - fine-tuning
    - Averaging embedding 이후에 fine-tuning으로 weight matrix를 추가학습
    - 단점 : 재학습의 부담

실험

- CUB-200-2011 (새 사진) dataset으로 실험
  - 200개 class 중 100개는 base class로 활용
  - 남은 100개 class는 novel class로 활용
  - novel class에 N-shot Learning의 top-1 accuracy를 관찰
- 실험 기법
  - without Fine-Tune (w/o FT)
    - Rand-noFT : novel class의 proxy vector를 random init + Fine-tuning하지 않은 상태
      - 학습없이 랜덤값을 썼기때문에 N-shot learning이 아니므로 N을 늘려가면서 테스트하지 않음
    - Imprinting : n개의 novel example의 element-wise average한 값을 활용
    - Imprinting + Aug : Imprinting + data augmentation 기법을 적용
  - with Fine-Tune (w FT)
    - Rand + FT : random init + Fine-tuning
    - Imprinting + FT : n개의 novel example의 element-wise average + Fine-tuning
    - AllClassJoint : base class + novel class 데이터로 fine-tuning
  - Generator + Classifier : 8. Low-shot Visual Recognition by Shrinking and Hallucinating Features
  - Matching Networks : 4. Matching Networks for One Shot Learning

표 1, 그림 3: 200-way learning에서 100 종류의 novel class에 대한 top-1 accuracy

	$n =$	1	2	5	10	20
w/o FT	Rand-noFT[2]	0.17	0.17	0.17	0.17	0.17
	Imprinting	21.26	28.69	**39.52**	45.77	49.32
	Imprinting + Aug	**21.40**	**30.03**	39.35	**46.35**	**49.80**
w/ FT	Rand + FT	5.25	13.41	34.95	54.33	65.60
	Imprinting + FT	**18.67**	**30.17**	**46.08**	**59.39**	**68.77**
	AllClassJoint	3.89	10.82	33.00	50.24	64.88
	Generator + Classifier [5]	18.56	19.07	20.00	20.27	20.88
	Matching Networks [22]	13.45	14.75	16.65	18.81	25.77

Table 1. 200-way top-1 accuracy for novel-class examples in CUB-200-2011. Imprinting provides good immediate performance without fine tuning. Adding data augmentation (Imprinting+Aug) does not give significant further benefit. The second block of 3 rows shows the results of fine tuning, for which the imprinting initialization retains an advantage. This remains true even when compared to training all classes from scratch (AllClassJoint). The final 2 rows provide comparisons with previous methods.

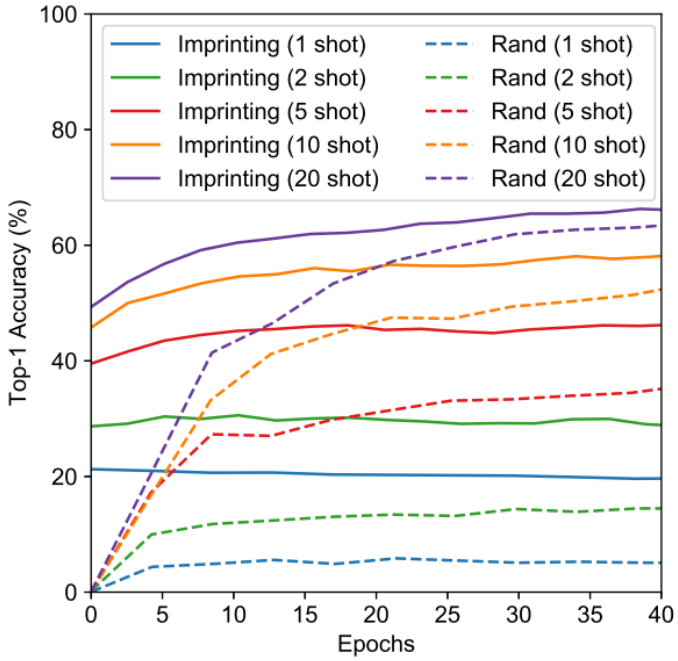


Figure 3. Accuracy of fine-tuned models on novel classes for the first 40 epochs of training. Table 1 lists results after 112 epochs.

표 2: 그림 4 : 200-way learning에서 200 종류(100종의 base class + 100종의 novel class)에 테스트 셋에 대한 top-1 accuracy

	$n =$	1	2	5	10	20
w/o FT	Rand-noFT	37.36	37.36	37.36	37.36	37.36
	Imprinting	**44.75**	48.21	**52.95**	55.99	57.47
	Imprinting + Aug	44.60	**48.48**	52.78	**56.51**	**57.84**
w/ FT	Rand + FT	39.26	43.36	53.69	63.17	**68.75**
	Imprinting + FT	**45.81**	**50.41**	**59.15**	**64.65**	68.73
	AllClassJoint	38.02	41.89	52.24	61.11	68.31
	Generator + Classifier [5]	45.42	46.56	47.79	47.88	48.22
	Matching Networks [22]	41.71	43.15	44.46	45.65	48.63

Table 2. 200-way top-1 accuracy measured across examples in all classes (100 base plus 100 novel classes) of CUB-200-2011. Imprinting retains similar advantages for rapid learning and initialization of fine-tuning as seen in Table 1.

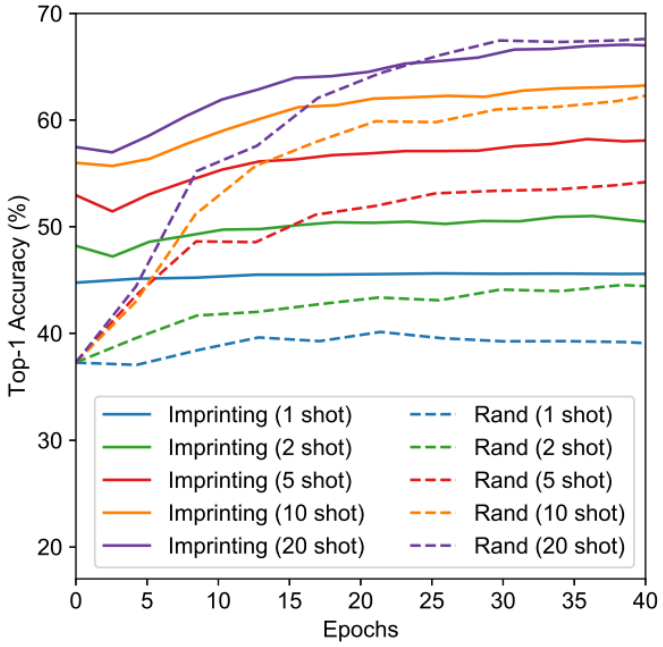


Figure 4. Accuracy of fine-tuned models measured over all classes (100 base plus 100 novel classes) for the first 40 epochs of training. Table 2 lists results after 112 epochs.

표 3, 그림 6 : 100 way learning에서 100 종의 novel class에 대한 Top-1 accuracy

	$n =$	1	2	5	10	20
w/o FT	Rand-noFT	0.85	0.85	0.85	0.85	0.85
	Imprinting	**26.76**	33.11	43.00	48.74	52.25
	Imprinting + Aug	26.08	**34.13**	**43.34**	**48.91**	**52.94**
w/ FT	Rand+FT	15.90	28.84	46.21	61.37	**71.57**
	Imprinting + FT	**26.59**	**34.33**	**49.39**	**61.65**	70.07

Table 3. Top-1 accuracy for transfer learning on CUB-200-2011 using 1–20 examples for computing imprinted weights. The imprinted weights provide good immediate performance while also providing better final classification accuracy for 1 to 5 shot learning following fine tuning.

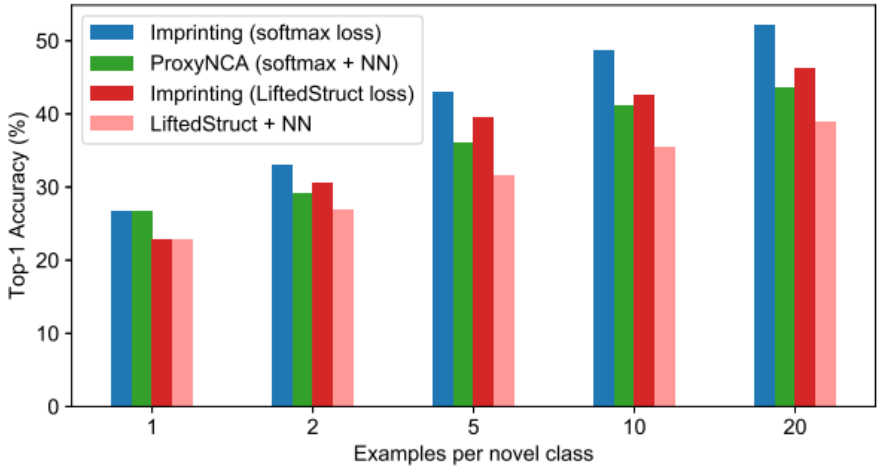


Figure 6. Top-1 accuracy of 100-way classification on novel classes of CUB-200-2011. Imprinting averaged embeddings with a softmax loss (blue bars) outperforms storing all individual embeddings with a nearest-neighbor classifier (green). By comparison, embeddings trained with the lifted structured loss do not perform as well as with the softmax loss (red and pink).

- 그림 7 : n-shot learning에 따라 embedding vector의 dimension에 따른 정확도 변화

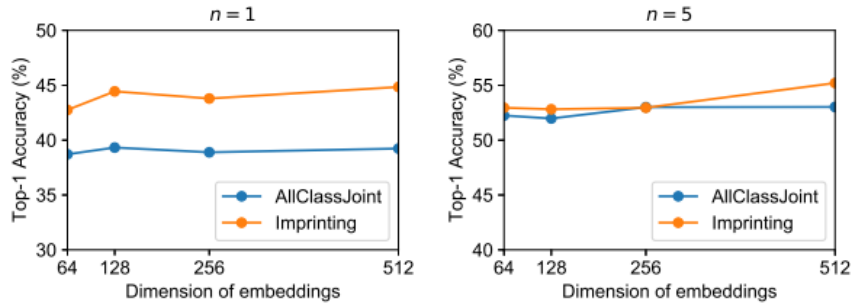


Figure 7. Classification accuracies for Imprinting and AllClassJoint with different embedding dimensionalities under 1-shot and 5-shot settings, respectively.

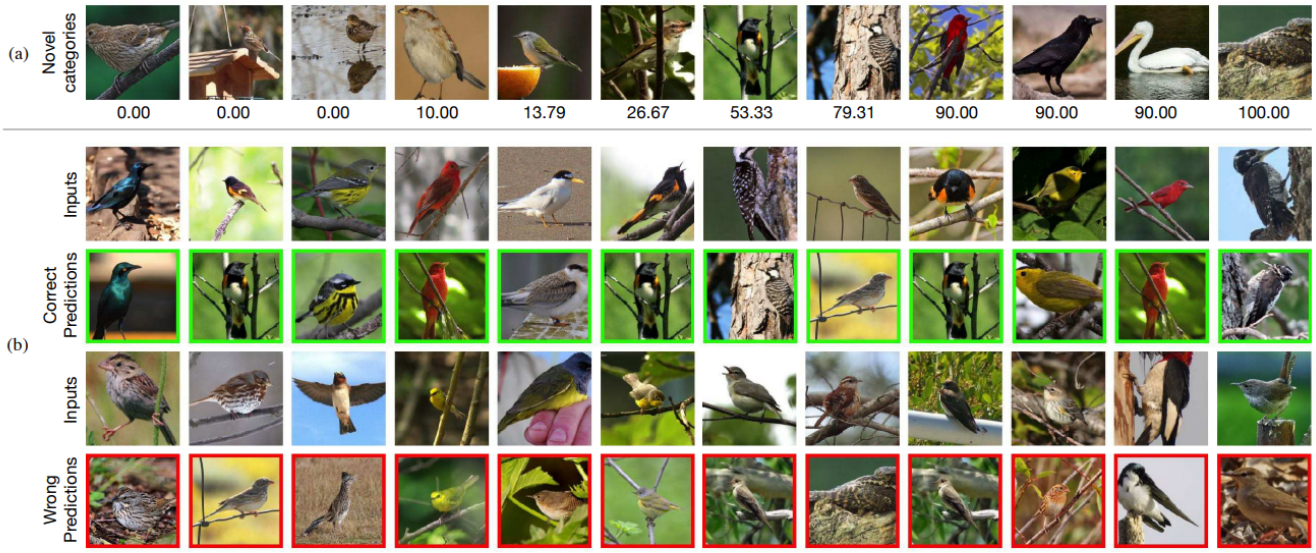


Figure 5. (a) A subset of exemplars used for 1-shot training of novel classes sorted by their recall@1 scores as shown below each exemplar. High-performing categories tend to exhibit more distinctive colors, shapes, and/or textures. (b) Randomly selected success and failure cases predicted by a 1-shot imprinted model on CUB-200-2011. Test images and the 1-shot exemplar whose embedding was used to imprint the predicted class are shown in separate rows. Correct and wrong predictions are marked with green and red borders, respectively.