

[week2] iCaRL: Incremental Classifier and Representation Learning

iCaRL : Incremental Classifier and Representation Learning



Abstract

class incremental한 방법으로 학습 가능한 모델

→ 적은 class로 학습한 모델이 현재 학습 상태를 유지하면서 새로운 class가 들어왔을 때, 그것에 대해서만 추가적으로 학습할 수 있는 방법이다.

강한 classifier와 data representation을 동시에 학습한다.

긴 시간에 걸쳐 많은 class들을 증가적으로 학습할 수 있는 모델이다.

Introduction

대부분의 인공 object recognition 시스템은 미리 세팅된 class들에 대해서만 학습이 진행된다.

그러나 컴퓨터 비전 필드는 flexible한 방법론을 찾기 바란다.

(객체 분류 시스템이 새로운 class에 대해서도 적용되길 바란다.)

⇒ **class incremental learning!**

- class incremental learning을 위해 취한 알고리즘

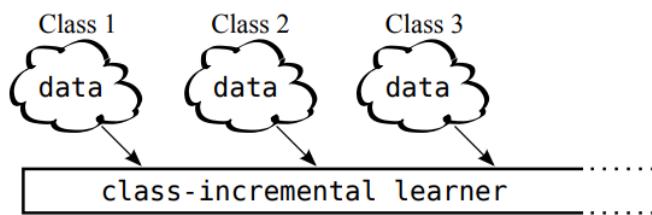


Figure 1: Class-incremental learning: an algorithm learns continuously from a sequential data stream in which new classes occur. At any time, the learner is able to perform multi-class classification for all classes observed so far.

- 1) 다른 시점에 발생한 다른 class의 예시에 대해 학습 가능해야한다.
- 2) 어느 시점이나 지금까지 발견된 class에 대한 multi-class 분류기를 제공해야한다.
- 3) computational requirement와 memory footprint는 지금까지 발견된 class의 수에 비해 제한되거나 매우 천천히 증가해야한다.

→ 현재 존재하는 multi class classifier들은 (1)나 (2) 조건을 위반한다. (그래서 지정된 수자의 class에 대해서만 분류 학습이 가능한 것)

→ 단순하게 class incremental learning을 위해 stochastic gradient descent optimization을 쓴다거나 하면 분류기의 성능이 빠르게 안좋아진다.

- 그래서 본 논문은 **iCaRL: Incremental Classifier and Representation Learning** 제시

: class-incremental setting에서 분류기와 feature representation을 동시에 학습하는 방법론

** main components

- nearest-mean-of-exemplar 규칙을 통한 분류
- herding (군집)을 기반으로 prioritized 된 사례 선택
- knowledge distillation와 prototype rehearsal을 이용한 representation learning

Method

- **Class-Incremental Classifier Learning**

Algorithm 1 iCaRL CLASSIFY

```

input  $x$  // image to be classified
require  $\mathcal{P} = (P_1, \dots, P_t)$  // class exemplar sets
require  $\varphi : \mathcal{X} \rightarrow \mathbb{R}^d$  // feature map
for  $y = 1, \dots, t$  do
     $\mu_y \leftarrow \frac{1}{|P_y|} \sum_{p \in P_y} \varphi(p)$  // mean-of-exemplars
end for
 $y^* \leftarrow \operatorname{argmin}_{y=1, \dots, t} \|\varphi(x) - \mu_y\|$  // nearest prototype
output class label  $y^*$ 

```

Algorithm 2 iCaRL INCREMENTALTRAIN

```

input  $X^s, \dots, X^t$  // training examples in per-class sets
input  $K$  // memory size
require  $\Theta$  // current model parameters
require  $\mathcal{P} = (P_1, \dots, P_{s-1})$  // current exemplar sets
 $\Theta \leftarrow \text{UPDATEREPRESENTATION}(X^s, \dots, X^t; \mathcal{P}, \Theta)$ 
 $m \leftarrow K/t$  // number of exemplars per class
for  $y = 1, \dots, s-1$  do
     $P_y \leftarrow \text{REDUCEEXEMPLARSET}(P_y, m)$ 
end for
for  $y = s, \dots, t$  do
     $P_y \leftarrow \text{CONSTRUCTEXEMPLARSET}(X_y, m, \Theta)$ 
end for
 $\mathcal{P} \leftarrow (P_1, \dots, P_t)$  // new exemplar sets

```

- **classification**

: data stream에서 다이나믹하게 exemplar 이미지셋을 선택한다. (각각의 셋은 지금까지 관찰된 class들을 포함한다)

: 이때 exemplar 이미지의 토탈 개수는 fixed parameter K를 넘지 않도록 한다.

- **training**

: update 루틴을 통해서 매번 새로운 class에 접근 가능하다.

→ 접근할 수 있는 새로운 observation에서 사용 가능한 추가 정보를 기반으로 internal knowledge를 조정한다 (새로운 class의 존재를 학습하는 방법)

- **Architecture**

: CNN 구조를 가지고 있다.

: trainable feature extractor으로 네트워크를 해석한다

→ 그 이후에 많은 sigmoid output node (현재까지 관찰된 class 개수만큼)를 가지고 있는 분류 layer가 온다.

: 모든 feature는 L2 정규화를 거친다 (평균도 다시 정규화)

: 네트워크의 파라미터는 theta를 이용해 작성하고, feature extract 부분을 fix 된 매개변수로 나누고, 가중치 vector를 variable한 수로 나눈다.

: 네트워크 output의 결과는 각 class에 대해

$$g_y(x) = \frac{1}{1 + \exp(-a_y(x))} \quad \text{with } a_y(x) = w_y^\top \varphi(x). \quad (1)$$

→ 0| output은 확률로 해석된다.

⇒ iCaRL는 이 네트워크를 representation learning을 위해서만 사용하고 실제 classification step에는 사용하지 않는다.

- **Resource usage**

: class 개수에 대한 upper bound가 없는 경우, 계속 다시 학습하면 시간이 지나면서 가중치 벡터가 엄청나게 증가할 것

→ iCaRL는 증가하는 리소스들을 retrain 없이 처리 가능하므로, 메모리 제한으로 인해 강제로 삭제하지 않는 한 예시들을 삭제하지 않는다.

- **Nearest-Mean-of-Exemplars Classification**

: exemplar의 각 클래스 평균에 대해 가장 가까운 클래스로 구분하겠다.

$$\mu_y = \frac{1}{|P_y|} \sum_{p \in P_y} \varphi(p)$$

$$y^* = \operatorname{argmin}_{y=1,\dots,t} \|\varphi(x) - \mu_y\|.$$

: 이때 softmax를 사용하지 않고 최종 output을 feature에 대한 sigmoid 값으로 출력

: class-prototypes은 feature representation이 변화하면 자동으로 변화
 → feature representation에 대해 robust 해진다.

- **Representation Learning**

Algorithm 3 iCaRL UPDATE REPRESENTATION

```

input  $X^s, \dots, X^t$  // training images of classes  $s, \dots, t$ 
require  $\mathcal{P} = (P_1, \dots, P_{s-1})$  // exemplar sets
require  $\Theta$  // current model parameters
// form combined training set:

$$\mathcal{D} \leftarrow \bigcup_{y=s, \dots, t} \{(x, y) : x \in X^y\} \cup \bigcup_{y=1, \dots, s-1} \{(x, y) : x \in P^y\}$$

// store network outputs with pre-update parameters:
for  $y = 1, \dots, s-1$  do
     $q_i^y \leftarrow g_y(x_i)$  for all  $(x_i, \cdot) \in \mathcal{D}$ 
end for
run network training (e.g. BackProp) with loss function

$$\ell(\Theta) = - \sum_{(x_i, y_i) \in \mathcal{D}} \left[ \sum_{y=s}^t \delta_{y=y_i} \log g_y(x_i) + \delta_{y \neq y_i} \log(1 - g_y(x_i)) \right. \\ \left. + \sum_{y=1}^{s-1} q_i^y \log g_y(x_i) + (1 - q_i^y) \log(1 - g_y(x_i)) \right]$$

that consists of classification and distillation terms.

```

: 기존 데이터에 새로운 class 데이터가 더해진 데이터를 만든다

→ feature extract & exemplar set 만들기

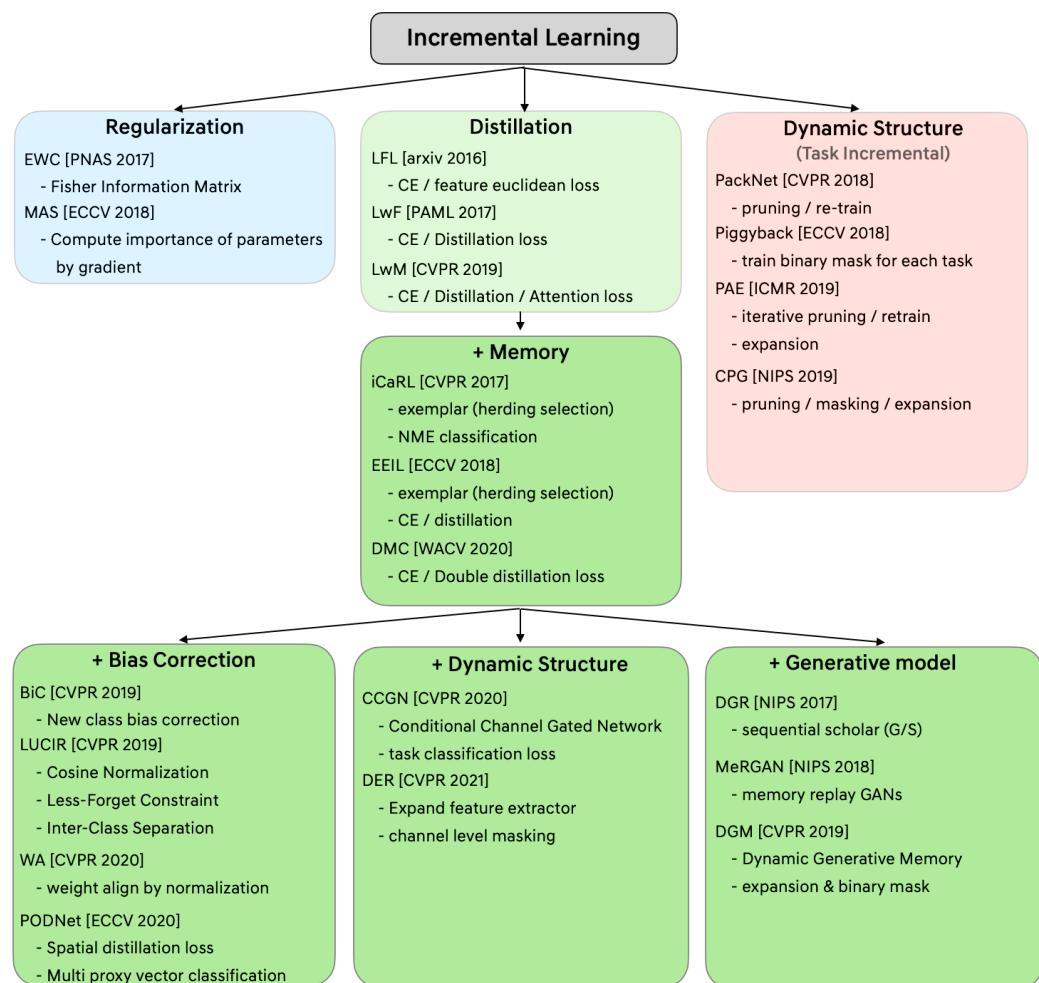
: 새로운 class에 대한 output은 classification loss로 학습하고, 기존 class에 대한 output은 distillation loss로 학습을 진행한다.

+ representation learning step은 일반적인 네트워크의 fine-tuning과 유사하다.

→ 이전의 네트워크 가중치에서 시작해서 train set에 대한 loss를 최소화 시키는 방향으로 학습을 진행하는 것.

Q) distillation?

: 이전 task에서 학습한 파라미터를 새로운 task를 위한 네트워크에 distillation (증류)



>> Incremental Learning의 다양한 방법론 중

iCaRL은 Distillation + Memory에 해당하는 방법 중 하나라고 한다.

(이전 task의 데이터를 소량 메모리로 두고 새로운 task 학습 때 활용)

- **Exemplar Management**

Algorithm 4 iCaRL CONSTRUCTEXEMPLARSET

input image set $X = \{x_1, \dots, x_n\}$ of class y
input m target number of exemplars
require current feature function $\varphi : \mathcal{X} \rightarrow \mathbb{R}^d$
 $\mu \leftarrow \frac{1}{n} \sum_{x \in X} \varphi(x)$ // current class mean
for $k = 1, \dots, m$ **do**
 $p_k \leftarrow \operatorname{argmin}_{x \in X} \left\| \mu - \frac{1}{k} [\varphi(x) + \sum_{j=1}^{k-1} \varphi(p_j)] \right\|$
end for
 $P \leftarrow (p_1, \dots, p_m)$
output exemplar set P

Algorithm 5 iCaRL REDUCEEXEMPLARSET

input m // target number of exemplars
input $P = (p_1, \dots, p_{|P|})$ // current exemplar set
 $P \leftarrow (p_1, \dots, p_m)$ // i.e. keep only first m
output exemplar set P

: 기존의 incremental learning 방법론에서는 new task를 학습할 때 old class data는 사용하지 않았다.

→ iCaRL에서는 기존 지식을 잊어버리는 현상을 방지하기 위해 기존 데이터를 조금씩 sampling하여 exemplar라는 작은 data set을 구축한 뒤 이를 활용하는 방안을 사용했다.

** 중요 사항

→ 모든 class의 데이터 개수를 동일하게

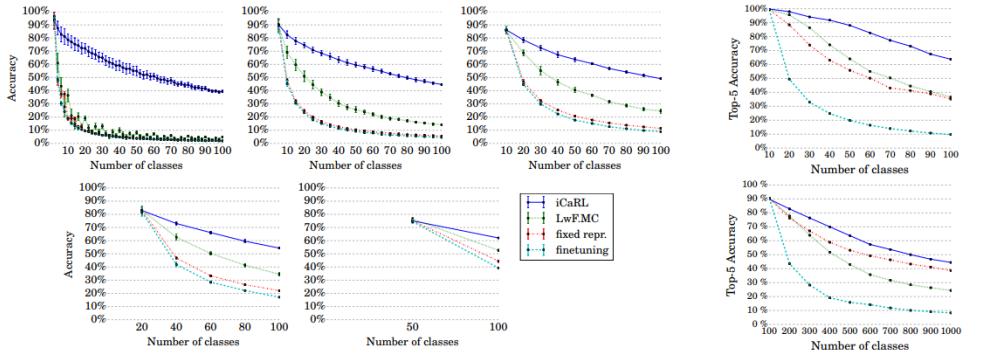
→ exemplar에 포함되는 데이터는 각 class를 가장 잘 표현하는 데이터여야 한다.

→ exemplar 구성 시, 각 class 별로 해당 class를 잘 표현하는 순서대로 구성되어야 한다.

⇒ 구현을 위해 herding의 prioritized construction 방식 이용

Experiments

- iCIFAR100, ImageNet에 대한 실험 결과



(a) Multi-class accuracy (averages and standard deviations over 10 repeats) on iCIFAR-100 with 2 (top left), 5 (top middle), 10 (top right), 20 (bottom left) or 50 (bottom right) classes per batch.
(b) Top-5 accuracy on iILSVRC-small (top) and iILSVRC-full (bottom).

Figure 2: Experimental results of class-incremental training on iCIFAR-100 and iILSVRC: reported are multi-class accuracies across all classes observed up to a certain time point. iCaRL clearly outperforms the other methods in this setting. Fixing the data representation after having trained on the first batch (*fixed repr.*) performs worse than distillation-based LwFMC, except for *iILSVRC-full*. Finetuning the network without preventing catastrophic forgetting (*finetuning*) achieves the worst results. For comparison, the same network trained with all data available achieves 68.6% multi-class accuracy.

- 실험 방식

1)

CIFAR100 데이터셋을 각각 2개 클래스 * 50번 / 5개 클래스 * 20번 / 10개 클래스 * 10번 / 20개 클래스 * 5번 / 50개 클래스 * 2번으로 나누기

→ incremental learning 으로 학습

→ 모든 클래스의 classification acc 평균, 표준편차를 측정

2)

ImageNet 데이터를 100개 클래스, 모든 클래스(1000개)로 나누기

→ incremental learning 방식으로 학습, classification acc 측정

- result

- 실험 종류

: iCaRL, fine-tuning, fixed representation, LwF

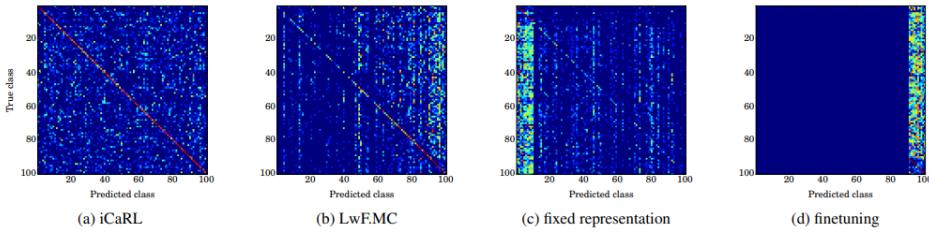


Figure 3: Confusion matrices of different method on iCIFAR-100 (with entries transformed by $\log(1+x)$ for better visibility). iCaRL's predictions are distributed close to uniformly over all classes, whereas LwF.MC tends to predict classes from recent batches more frequently. The classifier with fixed representation has a bias towards classes from the first batch, while the network trained by finetuning predicts exclusively classes labels from the last batch.

→ iCaRL이 구역 별 차이 없이 고른 성능 분포를 보이면서도 상대적으로 좋은 성능을 보임을 확인 가능하다.

+) fine tuning은 앞 지식은 잊고 뒷 부분에 대해서만 좋은 성능

• Differential Analysis

iCaRL이 기본적인 파인튜닝 모델보다 성능이 뛰어난 이유를 3가지 측면에서 분석하고자 했다.

(a)

(3개의 하이브리드 setup을 만들어서 확인해봄)

(a) Switching off different components of iCaRL (*hybrid1*, *hybrid2*, *hybrid3*, see text for details) leads to results mostly inbetween iCaRL and LwF.MC, showing that all of iCaRL's new components contribute to its performance.

batch size	iCaRL	<i>hybrid1</i>	<i>hybrid2</i>	<i>hybrid3</i>	LwF.MC
2 classes	57.0	36.6	57.6	57.0	11.7
5 classes	61.2	50.9	57.9	56.7	32.6
10 classes	64.1	59.3	59.9	58.1	44.4
20 classes	67.2	65.6	63.2	60.5	54.4
50 classes	68.6	68.2	65.3	61.5	64.5

- *hybrid1* : mean of exemplar classifier 대신 direct output classification 사용
- *hybrid2* : distillation loss 사용 안 함
- *hybrid3* : distillation loss와 exemplar 모두 사용 안 함

- LwF : classification + distillation loss를 사용했지만 exemplar을 사용하지 않은 방식

** 결과

- iCaRL / hybrid1 : mean of exemplar classifier는 smaller batch size에서 더 중요
- iCaRL / hybrid2 : 매우 작은 batch size에서는 distillation loss가 오히려 안 좋음
- hybrid3 / LwF : exemplar는 중요한 역할을 함

(b)

(b) Replacing iCaRL's mean-of-exemplars by a nearest-class-mean classifier (NCM) has only a small positive effect on the classification accuracy, showing that iCaRL's strategy for selecting exemplars is effective.

batch size	iCaRL	NCM
2 classes	57.0	59.3
5 classes	61.2	62.1
10 classes	64.1	64.5
20 classes	67.2	67.5
50 classes	68.6	68.7

⇒ NCM: 모든 데이터셋으로 exemplar을 구성하는 방식 (사실상 불가능) ...upper bound

⇒ iCaRL은 데이터를 많이 사용하지 않았음에도 불구하고 NCM 대비 크게 뒤지지 않는 성능 확인 가능

(c)

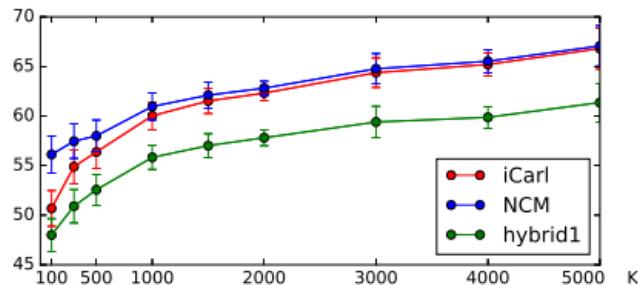


Figure 4: Average incremental accuracy on iCIFAR-100 with 10 classes per batch for different memory budgets K .

- 메모리 사용량에 대한 성능

⇒ 메모리 사용량이 커질수록 NCM에 근접하는 성능을 보여준다.

⇒ 하지만 메모리 사용량을 아무리 늘려도 direct classification 방식인 hybrid1은 성능이 크게 좋지 않은 것도 확인 가능하다.

Conclusion

: class incremental learning을 위한 방법

→ 분류기와 feature representation을 동시에 학습한다.

: 3가지의 main component를 가진다

- class 별로 적은 수의 데이터를 가지면서도 변화하는 data representation에 대해 robust한 nearest-mean-of-exemplars classifier
- 군집 기반의 prioritized exemplar 선택
- 과거 정보 망각을 방지하기 위해 representation learning 과정에서 distillation 사용

: exemplar image의 사용을 통해 강력한 분류 결과를 얻어냈다.

+) 그러나 여전히 성능이 뛰어난 것은 아니기 때문에 해당 task 해결에는 더 연구가 필요하다.