EEIL (End to End Incremental Learning)

Introduction

- 실제 응용 프로그램을 대상으로 하는 시각적 인식 시스템을 개발할 때 주요 과제 중 하나는 새로운 클래스가 계속해서 학습되는, 즉 분류기를 점진적으로 학습하는 것이다.
  - 예를 들어, 얼굴 인식 시스템은 새로운 사람을 식별하기 위하여 새로운 얼굴을 학습해야 하며, 이러한 task는 이미 학습한 얼굴들을 재학습시킬 필요 없이 해결되어야 한다.
  - 이는 대부분의 사람들에게 사소하지만, 머신 러닝 시스템에서는 그렇지 않다.
- 기존 모델들은 모든 샘플이 훈련 시간에 사용할 수 있어야 하며, 이전 데이터 일부만 사용하고 새로운 데이터만 다루는 경우가 없다.
  - 이상적인 시스템에서, 새로운 클래스는 기존 모델에 이전에 학습된 매개변수를 공유하면서 통합되어야 한다.
  - 이를 해결하기 위한 몇 가지 시도가 있었지만, 대부분의 이전 모델, 특히 딥러닝 접근 방식에서 새로운 정보가 추가될 때 기존 클래스에 대한 성능이 급격히 감소하는 문제를 겪고있다.
  - 본 논문에서는 이미지 분류 과제에서 이 문제를 해결하여 논문의 결과를 설명한다.

- 분류를 위한 진정한 Incremental deep learning 접근 방법은 다음과 같은 특징이 있다.
  1. 임의의 순서 및 시간에 클래스가 추가되는 데이터 흐름에서 훈련될 수 있는 능력
  2. 기존 및 신규 클래스에 대한 우수한 분류 성능
  3. 모델에 대한 합리적인 수의 매개변수 및 메모리 요구
  4. 분류기와 특성 표현을 공동으로 업데이트하기 위한 end-to-end 학습
  - 따라서 가장 이상적인 접근 방식은 무한한 수의 클래스를 마치 처음부터 훈련된 것 처럼, 정확도를 잃지 않고 정확히 동일한 수의 매개변수를 가지면서 점진적으로 훈련시킬

수 있어야 한다.

- 그러나 기존의 접근 방식 중 어느 것도 이러한 제약을 모두 충족하지 않는다.
  - 종종 분류기와 표현 학습 task를 분리하거나, 매우 특수한 상황 (e.g., 새로운 데이터셋에서 학습하지만 기존에 비해 새로운 클래스를 학습하지 않는 경우)에 한정되거나, 특수한 문제(e.g., 객체 탐지)에 한정된다.
  - 그 중 일부는 SVM과 같은 기존의 분류기를 포함하여 딥러닝 아키텍처에 적합하지 않다.
  - 다른 일부는 매개변수 및 레이어 개수를 급격히 증가시켜, 클래스가 늘어남에 따라 메모리 사용량이 매우 커지게 된다.
  - 요약하자면, 진정한 Incremental Learner의 특징을 모두 충족하는 SOTA 방법이 존재하지 않는다.
- 본 논문의 주로 기여한 바는 위의 문제를 incremental learning을 위해 만들어진 논문의 end to end 접근 방법으로 이 문제를 해결한데 있다.

Related Work

- 여기서는 본 논문의 접근 방법과 관련있는 모델을 고정된 특성 세트를 사용하는 traditional approach 와 deep learning을 사용하여 특성을 학습하는 approach 로 분류하여 소개한다.

Traditional approaches

- 초기 방법들은 SVM 분류기의 핵심 컴포넌트인 서포트 벡터와 Karush-Kuhn-Tucker 조건을 활용했다.
  - 그 중 일부는 기존 데이터로부터 학습된 분류기를 인코딩하는 서포트 벡터를 유지하여 새로운 데이터와 함께 새로운 결정 경계를 학습했다.
  - Cauwenberghs와 Poggio는 Karush-Kuhn-Tucker 조건을 이전의 모든 데이터에 유지하면서 새로운 데이터에 따라 해를 업데이트하는 방식으로 앞의 방식에 대한 대안을

제시했다.

    - 이러한 초기 시도는 일부 성공을 보여주었지만, 특정 분류기에 제한되며, 표현과 분류기를 동시에 학습하는 현 패러다임으로 확장되지 않았다.

- 또 다른 접근 방식은 Lifelong이나 never-ending learning의 형태로 시간이 지남에 따라 개념을 학습하는 것이다.
    - Lifelong learning은 기존의 task에서 습득한 knowledge를 새로운 task로 transfer 하는 것과 유사하다.
    - Never-ending-learning은 기존 분류기를 개선하거나 새로운 분류기를 학습하기 위해 지속적으로 데이터를 수집하는데 중점을 둔다.
    - 두 패러다임은 전체 훈련 데이터셋을 필요로 하거나 고정된 표현에 의존한다.
    - 일부 방법들은 완전한 훈련세트 없이 분류기를 학습하여 이 문제를 부분적으로나마 해결하지만, 여전히 고정되거나 관리되는 데이터 표현으로 제한된다.
        - 이는 분류기나 회귀 모델을 제한하거나 (e.g., 선형 분해가 가능한 모델) nearest mean classifier (NMC) 를 사용하거나 , 아니면 random forest 변형을 사용하는 방식으로 이루어진다.
    - 그 다음 기준이나 클래스별 prototype 을 업데이트 함으로 incremental learning이 이루어진다.
- 전반적으로, 이러한 모든 방법의 주요 단점은 task별 데이터 표현이 부족하여 성능이 저하된다는 것이다.
    - 본 논문의 방법은 특성과 분류기의 공동 학습으로 이 문제를 해결한다.

Deep learning approaches

- 이 접근 방법들은 task별 특성과 분류기를 동시에 학습하는 자연스러운 방법을 제공한다.
    - 그러나, 이 패러다임에서 모델의 incremental learning은 catastrophic forgetting 이라는, 기존의 클래스 세트에 대한 성능이 극적으로 저하되는 현상이 발생한다.

    - 이 문제를 극복하기 위한 초기 시도는 connectionist network을 목표로 했기 때문에, 오늘날의 컴퓨터 비전 문제에 대한 심층 아키텍쳐에는 적용할 수 없다.
- 이전 task에 대한 성능을 보존하기 위해 제안된 Li et al. 의 최근 연구는 distillation loss 를 표준 cross-entropy-loss 와 함께 사용하는 것이었다.
    - Distillation loss 는 원래 서로 다른 신경망 사이의 knowledge transfer 를 위해 제안되었으나, 여기에서는 새로운 훈련 샘플로 업데이트 하면서 이전 task에 대한 신경망 응답을 유지하도록 수정되었다.
    - 이러한 접근 방법이 이전 샘플과 새 샘플이 서로 다른 데이터셋에서 왔으며 혼동되지 않는 단순한 시나리오에서의 forgetting을 어느정도 줄였지만, 이상적인 성능을 보여주지는 않았다.
        - 이는 이전 클래스에 대한 knowledge 표현이 약하며, 본 논문의 방법에서 수행된 것 처럼 이를 exemplar set으로 확장하지 않기 때문일지도 모른다.
        - 일부 연구는 새로운 클래스의 샘플들이 지속적으로 추가되는, 특히 새로운 샘플과 이전의 샘플들이 서로 연관된 분포에서 추출되는 순차 학습 시나리오에서 심각한 오류를 보이면서 이러한 약점을 입증했으며, 이는 본 논문에서 다루는 문제이다.

- Distillation loss를 사용하는 다른 방법들은 원 모델의 일부 레이어를 고정하여 새로운 데이터에 대한 적응성을 제한하는 것을 제안했다.
- Triki et al. 은 Li et al. 의 방법을 기반으로 distillation loss 대신 오토 인코더를 사용하여 이전 task의 knowledge 유지하는 방법을 제안했다.
    - 이 방법도 Li et al. 과 비슷하게 이전 신경망과 새 신경망이 서로 다른 데이터셋에 훈련되는 제한적 시나리오에서도 평가되었다.
- Distillation loss는 또한 객체 탐지기의 incremental learning 에도 채택 되었다.
    - 객체 탐지에는 성공했지만, 이 아키텍쳐가 더 범용적인 incremental learning 시나리오에 적용될 수 있는지는 미지수다.
- Catastrophic forgetting을 완화하기 위한 다른 전략에는 신경망의 레이어 수를 증가시켜서 새로운 클래스의 특성을 학습하거나, 매개변수별 정규화를 통해 선택적으로 학습을 낮추는 것이 있다.
    - Xiao et al. 또한 관련한 체계를 따르고, 새로운 클래스를 학습함에 따라 점진적으로 트리 구조 모델을 성장시킨다.

- 이러한 모든 접근 방법의 주요한 단점은 매개변수, 즉 가중치, task, 그리고 새로운 레이어의 전체 수가 급격히 증가한다는 것이다.
- 이에 반해, 본 논문의 모델은 원 모델의 크기에서 최소한의 변화만을 만들어낸다.

- Rebuffi et al. 은 분류기 학습과 데이터 표현 학습이 분리된 Incremental learning 접근 방법인 iCaRL을 제안했다.
  - iCaRL은 테스트 샘플을 분류하기 위해 기존의 NMC를 사용한다. 즉, 기존의 데이터 샘플과 새 데이터 샘플을 포함하는 보조 세트를 유지한다.
  - 데이터 표현 모델인 표준 신경망은 distillation loss 와 classification loss 를 같이 사용하여 새로운 샘플을 사용할 수 있을 때 업데이트 된다.
  - 본 논문의 접근 방법 또한 기존 클래스의 일부 샘플을 representative memory 컴포넌트에서의 exemplar로 사용하지만, ene-to-end 방식으로 분류기와 특성을 동시에 학습한다는 점에서 이 접근 방법의 한계를 극복한다.

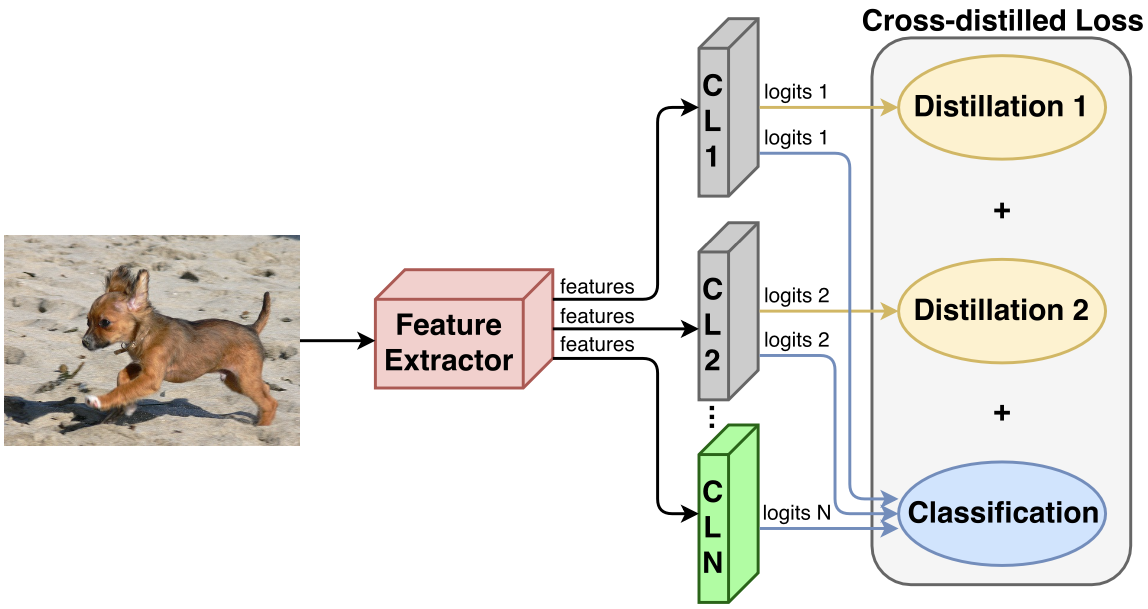

Fig. 1: **Our incremental model**. Given an input image, the feature extractor produces a set of features which are used by the *classification layers* (CL*i* blocks) to generate a set of *logits*. Grey *classification layers* contain old classes and their *logits* are used for distillation and classification. The green *classification layer* (CL*N* block) contains new classes and its *logits* are involved only in classification. (Best viewed in color.)

- 본 논문의 end-to-end 접근 방법은 cross-distilled loss function이라는, cross-entropy-loss 와 distillation loss 를 같이 사용하는 함수를 사용하여 훈련된 심층 신경망을 사용한다.
  - 접근 방법이 어떤 특징을 요구하지 않으므로, 본 논문의 신경망은 분류를 위해 설계된 대부분의 심층 모델 아키텍쳐를 기반으로 할 수 있다.
  - 분류를 위한 일반적인 아키텍쳐는 Fig. 1. 에서 하나의 분류 레이어와 분류 손실과 함께 확인할 수 있다.
    - 분류 레이어 (classification layer)는 feature extractor 로 부터 특성을 받아 일련의 logit을 만들어내고, 이는 소프트맥스 레이어로 전달되어 클래스 점수로 변환된다.
  - 모델이 이전 클래스로부터 얻은 knowledge 를 유지하도록 하기 위해, 본 논문에서는 기존 클래스로부터 가장 대표적인 (representative) 샘플을 저장 및 관리하는 representative memory를 사용한다.
  - 추가적으로 data augmentation 과 balanced find tuning을 수행한다.
- 이 모든 컴포넌트를 함께 사용하면 SOTA의 결과를 얻을 수 있다.

Representative memory

- 현재 모델에 새로운 클래스 (또는 클래스 집합)이 추가되면, 이로부터 가장 대표적인 샘플들이 선택되어 representative memory에 저장된다.
- 여기에서는 두 메모리 설정을 조사한다.
  - 첫 번째 설정은 K개 샘플로 한정된 Capacity 의 메모리를 고려한다.
    - 메모리의 capacity가 클래스의 수가 무관하기 때문에, 더 많은 클래스가 저장될 수록 클래스당 유지되는 샘플 수가 줄어든다.
    - 따라서 클래스 별 샘플 수는 $n = [K/c]$이며 c는 메모리에 저장된 클래수의 수이다.
  - 두 번째 설정은 클래스당 일정한 exemplar를 저장한다.
    - 따라서 클래스의 수가 늘어날 수록 메모리 크기도 커진다.

- Representative memory unit 은 저장할 새로운 샘플의 선택, 그리고 나머지 샘플들의 삭제의 두 가지 연산을 수행한다.

Selection of new samples

- 이 연산은 클래스의 평균 샘플에 대한 거리를 기반으로 클래스의 정렬된 샘플 목록을 생상하는 herding selection 을 기반으로 한다.
    - 정렬된 샘플 목록이 주어지면, 첫 n개의 샘플이 선택되며, 평균에 따라 이 샘플들은 클래스를 가장 잘 대표한다.
    - 이 방법은 Sec. 6.3 에서 볼 수 있듯이 무작위 선택이나 각 샘플에서 클래스 평균까지 거리의 히스토그램과 같은 다양한 접근 방법에서 테스트한 결과 선택되었다.
    - 선택은 새로운 클래스가 메모리에 추가될 때 마다 클래스 별로 한 번씩 수행된다.

Removing samples

- 이 연산은 훈련 과정 이후에 수행되어 메모리가 새로운 클래스의 샘플에 할당되도록 한다.
    - 샘플이 정렬된 리스트에 저장되므로 이 연산은 간단하며, 메모리 유닛은 각 클래스 샘플 세트 끝의 샘플들을 제거한다.
    - 이 연산 이후에 삭제된 샘플들은 다시 사용되지 않는다.

Deep network

Architecture

- 신경망은 Fig. 1 과 같이 여러 컴포넌트로 구성된다.
    - 첫 번째 컴포넌트는 feature extractor로 입력 이미지를 특성 벡터로 변환하는 일련의 레이어이다.

    - 그 다음 컴포넌트는 classification layer로, 모델의 마지막 전 연결 레이어이며 클래스 수 만큼의 출력을 가지고 있다.
        - 이 컴포넌트는 특성을 받아 일련의 logit을 만들어 낸다.
    - 훈련 단계에서 신경망을 업데이트하기 위한 gradient는 logit으로 cross-distilled loss function을 통해 계산된다.
    - 테스트 단계는 손실 함수가 소프트맥스 레이어로 대체된다.

- Incremental learning 프레임워크를 구축하기 위해, 첫 번째 클레스에 대해 기존의 분류를 위한 심층 아키텍처로 시작한다.
    - 새로운 클래스가 훈련되면 이 클래스에 관한 새로운 분류 레이어를 추가하고 이를 feature extractor와 cross-distilled loss 컴포넌트에 연결한다.
        - Incremental learning 과정에서 feature extractor의 아키텍쳐는 변하지 않으며, 새로운 분류 레이어만 여기에 연결될 뿐이다.
        - 따라서 단지 필요할 때 incremental classification layer와 cross-distilled loss function을 추가하는 것만으로도, 모든 아키텍쳐를 사용할 수 있다.

Cross-distilled loss function

- 이 함수는 기존 클래스로부터 knowledge를 유지하는 distillation loss 와, 새로운 클래스들을 분류하도록 학습하는 multi-class cross-entropy loss 를 결합한다.
    - Distillation loss는 이전 클래스의 분류 레이어만 적용되는 반면, multi-class cross-entropy loss 는모든 분류 레이어에 적용된다.
    - 이는 모델이 클래스들에 대한 결정 경계를 업데이트하도록 한다.

- 손실 계산은 Fig. 1에서 확인할 수 있으며, cross-distilled loss function $L(w)$는 다음과 같이 정의 된다.

- $L_C(W)$는 이전 클래스와 새로운 클래스 샘플들에 적용되는 cross-entropy loss 이다.
- $L_{D_f}(w)$는 분류 레이어 f의 distillation loss 이다.
- F는 기존 클래스에 대한 분류 레이어의 총 개수이다.

$$L(\omega) = L_C(\omega) + \sum_{f=1}^{F} L_{D_f}(\omega).$$

- Cross-entropy loss $L_C(w)$는 다음과 같이 정의 된다.
  - q_i는 샘플 i의 분류 레이어의 logit에 소프트 맥스 함수를 적용하여 얻은 점수이다.
  - p_i는 샘플에 대한 ground-truth이다.
  - N과 C는 각각 샘플 및 클래스 개수이다.

$$L_C(\omega) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} p_{ij} \log q_{ij}.$$

- Distillation loss $L_D(w)$ 는 다음과 같이 정의 된다.
  - $pdist_i$와 $qdist_i$는 각각 p_i와 q_i에서 logit을 지수 $1/T$로 올려 얻은 값이며 여기서 T는 distillation 매개 변수이다.
  - $T = 1$이면 점수가 가장 높은 클래스가 손실에 큰 영향을 미치므로, 더 높은 손실값을 최소화되어야 한다.
  - $T > 1$이면 나머지 클래스가 더 큰 영향을 미치므로, 더 높은 손실값을 최소화되어야 한다.
    - 이는 신경망이 클래스 사이보다 세분화된 분리를 학습하도록 하여 결과적으로 신경망으로 클래스에 대한 식별적인 표현을 학습한다.
  - 경험적인 결과에 따라 본 논문에서는 모든 실험에 대해 T = 2로 설정했다.

$$L_D(\omega) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} pdist_{ij} \log qdist_{ij}.$$

Incremental Learning

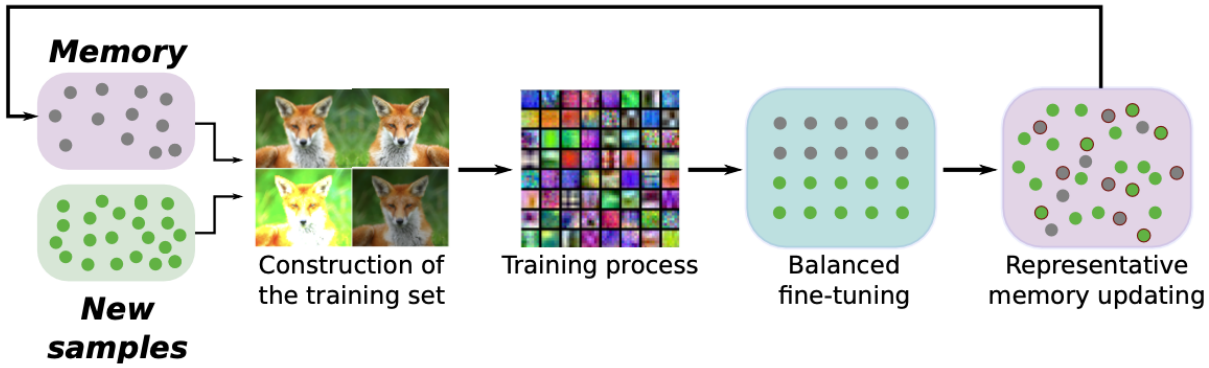

Fig. 2: **Incremental training**. Grey dots correspond to samples stored in the representative memory. Green dots correspond to samples from the new classes. Dots with red border correspond to the selected samples to be stored in the memory. (Best viewed in color.)

- 본 논문의 incremental learning 단계는 Fig. 2 에서 볼 수 있듯이 4가지 주요 단계로 구성된다.
  1. Construction of the training set : 다음 단계에서 사용될 훈련 데이터를 준비한다.
  2. Training process : 모델을 훈련 데이터로 훈련시킨다.
  3. Balanced find-tuning: 훈련 데이터 서브셋으로 모델을 fine-tuning 시킨다.
  4. Representative memory updating : Representative memory를 업데이트하여 새로운 클래스의 샘플을 포함한다.

Construction of the training set

- 본 논문의 훈련 세트는 새로운 클래스의 샘플과, representative memory에 저장된 이전 클래스의 exemplar로 구성된다.
- 접근 방법이 두 가지 손실 함수를 사용하므로, 각 샘플에 두 가지 레이블이 필요한다.

- 분류에서는 이미지의 클래스를 의미하는 one-hot vector를 사용한다.
- Distillation 에서는 이전 클래스의 모든 분류 레이어에서 만들어진 logit들을 레이블로 사용한다.
  - 따라서 이전 클래스의 분류 레이어 수만큼 샘플별 distillation 레이블을 사용한다.
  - 이전 knowledge 를 강화하기 위해 새로운 클래스의 샘플도 distillation에 사용된다.
- 이런 식으로 모든 이미지는 두 가지 손실 모두에 대해 gradient를 만들어낸다.
- 따라서 신경망에서 이미지를 평가할 때, 출력은 레이블과 관계없이 심층 모델의 모든 레이어를 구성하는 가중치 동작을 인코딩한다.
- 훈련 세트의 각 이미지는 분류 레이블과 함께 F개의 distillation 레이블을 가지게 된다.
  - 이 레이블 추출은 각 incremental step에서 수행된다.

Training process

- 본 논문의 cross-distilled loss function은 레이블이 있는 증강 훈련 세트를 사용하고 일련의 gradient를 생성해 심층 모델을 최적화한다.
  - 이때, 훈련 동안 모델의 모든 가중치가 업데이트 된다.
  - 따라서 모든 샘플에 대해 feature extractor에서 얻은 특성은 연속적인 incremental step 사이에 변경될 가능성이 높으며 분류 레이어는 이러한 새로운 특성을 처리하기 위해 가중치를 조정해야 한다.
  - 이는 feature extractor가 고정되고 분류 레이어만 훈련되는 다른 incremental learning 접근 방법과의 중요 차이점이다.

Balanced fine-tuning

- 이전 클래스의 모든 샘플을 저장하지 않기 때문에, 훈련 중 이전 클래스에서 사용할 수 있는 샘플들은 새로운 클래스의 샘플보다 훨씬 적을 수 있다.
  - 이러한 불균형한 훈련 시나리오를 처리하기 위해, 본 논문에서는 낮은 학습률과 샘플의 balanced subset을 사용하여 추가적인 fine-tuning 을 수행한다.

    - 새로운 훈련 서브셋은 이전 클래스 또는 새 클래스에 속하는지에 관계없이 클래스별로 동일한 수의 샘플을 포함한다.
        - 이 서브셋은 Sec 3.1 에 설명된 선택 알로리즘에 따라 각 클래스의 가장 대표적인 샘플만을 유지하여, 새로운 클래스의 샘플 수를 줄임으로서 만들어진다.
    - 새로운 클래스의 샘플을 제거하면, 모델은 이전 훈련 단계에서 얻은 knowledge를 잠재적으로 잊을 수 있다.
        - 본 논문에서는 새로운 클래스의 분류 레이어의 임시 distillation loss를 추가하여 이를 방지한다.

Representative memory updating

- Balanced fine-tuning 이후 새 클래스로부터 exemplar를 포함하도록 representative memory를 업데이트 해아한다.
    - 이는 Sec 3.1에서 설명한 선택 및 제거 연산을 통해 수행된다.
        - 먼저 메모리 유닛은 저장된 클래스의 샘플들을 제거해 새로운 클래스의 샘플을 위한 공간을 할당한다.
        - 그 다음 선택 알고리즘에 따라 새로운 클래스에서 가장 대표적인 샘플을 선택하여 메모리 유닛에 저장한다.

Implementation Details

- 본 논문의 모델은 MatConvet에서 구현된다.
- 각 incremental step 마다 40 epoch의 훈련을 수행하고, 30 epoch의 balanced fine-tuning 을 추가로 수행한다.
- 학습률은 훈련에서 0.1에서 시작하고, 그 이후 10epoch마다 10배 감소한다.
    - Fine-tuning에서는 0.01로 시작하고, 마찬가지로 매 10 epoch 마다 10배 감소한다.

- 128 샘플의 미니배치, 가중치 감소 0.0001 그리고 momentem 0.9의 표준 확률 경사하강법 SGD으로 신경망을 훈련시킨다.

- 과적합을 최소화하기 위하여 그래디언트 L^2 정규화와 무작위 노이즈 (η =0.3,γ = 0.55)를 적용한다.
- He et al.(ResNet)의 설정을 따라, 데이터셋 별 CNN/deep 모델을 사용한다.
  - 이를 통해 신경망 아키텍쳐는 데이터셋 마다의 특징에 맞도록 조정될 수 있다.
    - 심층 모델로는 CIFAR-100에는 32-layer ResNet을, 그리고 ImageNet에는 18-layer ResNet을 사용한다.
- Representative memory 는 CIFAR-100에서 K = 2000개, ImageNet에서는 K = 20000 개의 distillation sample을 저장한다.
- CIFAR-100 에서 모델을 훈련시킬 때, 픽셀값으로 255로 나누고 훈련세트의 평균 이미지를 빼서 입력 데이터를 정규화한다.
- ImageNet에서는 He et al. 에서와 같이 픽셀값 정규화 없이 훈련 세트의 평균 이미지를 빼기만 하면 된다.
- 간단히 사용할 수 있는 class-incremental learning 벤치마크가 없으므로, 기존의 다중 클래스 데이터셋의 클래스를 incremental batch로 분할하는 표준 설정을 따른다.
- 실험에서 사용한 모델의 설명은 다음과 같다.
  - iCaRL-Hybrid1 은 iCaRL에서 NCM 대신 CNN 분류기를 사용하는 변형이다.
  - LwF.MC는 LwF의 다중 클래스 구현이다.

- 각 방법의 결과를 모든 incremental batch 에서의 평균 정확도로 기록한다.
  - 이 평균에서 첫 번째 batch의 정확도는 incremental learning 과 상관이 없으므로 고려하지 않는다.
  - 이는 iCaRL 논문의 평가 방식과 다르며, 본 논문의 결과와 다른 결과를 보이는 이유이다.

Data augmentation

- 접근 방법의 2,3 번째 단계는 훈련 단계 이전에 데이터 증강을 수행하며 수행되는 연산들은 구체적으로 다음과 같다.

1. Brightness : 원 이미지의 intensity에 [-63, 63]의 무작위 intensity를 추가한다.
2. Contrast normalization: 원 이미지의 대비를 [0.2, 1.8]의 무작위 값으로 대체된다. 계산 식은 $im_{altered} = (im \text{ x } mean) \text{ x } (im \text{ x } mean) \text{ x } contrast + mean$이며 im은 원의 이미지, $mean$은 채널별 평균 픽셀, $contrast$는 무작위 상수이다.
3. Random cropping. 1 ,2 연살 결과를 포함한 모든 이미지를 무작위로 crop 한다.
4. Mirroring. 1, 2, 3 연산 결과를 포함한 모든 이미지에 대한 거울상 이미지를 계산한다.

Evaluation on CIFAR-100

- CIFAR-100 데이터셋에서는 세 가지의 실험을 수행한다.
  - 첫 번째 실험은 iCaRL의 실험 프로토콜을 따라 representative memory unit의 최대 저장 capacity를 설정한다.
  - 두 번째 실험에서는 메모리 크기가 고정되지 않고 대신 각 기존 클래스 별로 고정된 수의 샘플을 사용한다.
    - 각 incremental step마다 representative memory unit 에 새로운 클래스가 저장되므로 메모리 크기가 중요하다.
    - 마지막 실험에서는 ablation을 진행해 정확도 측면에서 접근 방법의 다양한 컴포넌트의 영향을 분석한다.

Dataset

- CIFAR-100 데이터셋은 100가지 클래스의 32 x 32 RGB 이미지의 60,000장으로 이루어져 있으며, 클래스 별로 600장의 이미지를 가지고 있다.
  - 각 클래스는 훈련 이미지 500장과 테스트 이미지 100장으로 이루어진다.
  - 본 논문에서는 100클래스를 무작위 순서로, 2, 5, 10, 20, 50 클래스 분할로 나눈다.
    - 이에 따라 각각 50, 20, 10, 5, 2개의 incremental step을 가지게 된다.
  - 각 incremental step 이후 모델은 모든 훈련된 클래스에서의 테스트 데이터로 평가된다.
    - 각 incremental step의 평가 metric은 표준 다중 클래스 정확도이다.

- 실험은 서로 다른 무작위의 클래스 순서로 다섯 번 진행되어, 평균 정확도와 표준편차를 계산한다.
  - 앞서 언급했듯이 첫 번째 step의 정확도는 incremental learning을 의미하지 않기 때문에 평균에 추가하지 않는다.

- CIFAR에서는 데이터 증강 단계를 따르고, 매 훈련 샘플마다 11개의 새로운 샘플을 생성한다.
  - Bright normalization 1장, contrast normalization 1장, random crop 3장, mirror 6장이다.

Fixed memory size

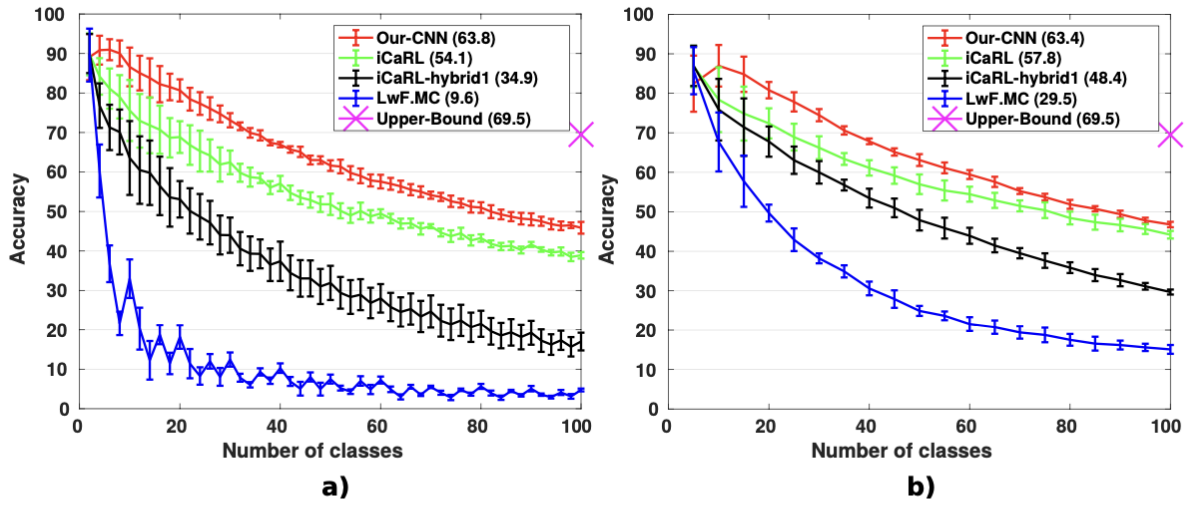

Fig. 3: **Accuracy on CIFAR-100.** Average and standard deviation of 5 executions with (a) 2 and (b) 5 classes per incremental step. Average of the incremental steps is shown in parentheses for each method. (Best viewed in pdf.)

# classes	2	5	10	20	50
Our-CNN	**63.8 ± 1.9**	**63.4 ± 1.6**	**63.6 ± 1.3**	**63.7 ± 1.1**	60.8 ± 0.3
iCaRL	54.1 ± 2.5	57.8 ± 2.6	60.5 ± 1.6	62.0 ± 1.2	**61.8 ± 0.4**
Hybrid1	34.9 ± 4.5	48.4 ± 2.6	55.8 ± 1.8	60.4 ± 1.0	60.8 ± 0.7
LwF.MC	9.6 ± 1.5	29.5 ± 2.2	40.4 ± 2.0	47.6 ± 1.5	52.9 ± 0.6

(a) CIFAR-100

# classes	10	100
Our-CNN	**90.4**	**69.4**
iCaRL	85.0	62.5
Hybrid1	83.5	46.1
LwF.MC	79.1	43.8

(b) ImageNet

Table 1: **Fixed memory size: accuracy on CIFAR-100 and ImageNet.** Each column represents a different number of classes per incremental step. Each row represents a different approach. The best results are marked in bold.

- 서로 다른 클래스 순서와 2, 5, 10 ,20 ,50 클래스의 incremental step으로 이루어진 5가지 분할에서 실험을 진행한다.
    - 결과를 서로 비교할 수 있도록 모든 실험된 방법에서는 클래스 순서가 동일하다.
    - Tab. 1(a)는 실험 결과를 요약하고 있으며, Fig.3 에서는 2, 5 클래스의 incremental step을 보여주고 있다.
        - 나머지 step들은 Appendix에서 확인할 수 있다.
    - Fig. 3의 Upper-Bound는 모든 클래스와 훈련 샘플을 사용해 훈련시킨 non-incremental model이다.
- 실험 결과, 본 논문의 end-to-end 접근 방법이 2, 5, 10, 20 클래스에서 최고의 성능을 보여주었다.
    - 50 클래스에서는 Hybrid1 과 같은 점수를 달성했지만, iCaRL보다 1% 낮은 성능을 보여주었다.
        - 이는 메모리 크기가 제한되어, 이전 클래스보다 새로운 샘플이 12.5배 더 많이 포함된 불균형한 훈련세트를 가지기 때문이다.
    - iCaRL과 비교하여 본 논문 방법 성능의 통계적 유의성을 강조하기 위해 CIFAR-100에서 얻은 결과에 paired-test를 수행했다.
        - 2, 5, 10, 20, 50 각각에 대한 p-값은 0.00005, 0.0005, 0.003, 0.0077, 0.9886 이며, 이는 비슷한 성능을 보인 50클래스를 제외하고 모든 경우에서 iCaRL에 대한 본 논문 방법의 성능 향상이 통계적으로 유의함을 보여준다.
- 또한 Tab.1(a) 에서 볼수 있듯이 incremental step의 크기에 대해 본 논문의 접근 방법은 (각 step에 추가된 클래스의 수에 의존하는) 다른 방법들과는 다르게 안정적으로 성능을 유지한다.

    - 이는 incremental learning 초기 단계에서 각 incremental step 에서 각 클래스의 수가 적으면 적은 수의 클래스만 분류하게 되므로 정확도에 이점이 있기 때문이다. (하지만 더 많은 step 이 진행되어 모든 클래스에 대해 훈련시키면 최종 단계의 정확도는 떨어진다.)

- 각 incremental step에서 많은 수의 클래스가 추가된다면 반대로 동작한다.
    - 초기 단계에서 낮은 정확도를 보여주나, 최종 단계에서 더 높은 값으로 바뀐다.
    - Fig. 3은 또한 incremental step에서 적은 수의 클래스를 사용할 경우 본 논문의 접근 방법이 iCaRL보다 훨씬 더 우수함을 보여준다.
        - 각 step별 클래스 수가 더 커지면 iCaRL 방법은 본 논문의 성능에 근접하나 전반적으로 낮은 성능을 보여준다.
        - 본 논문의 방법은 모든 경우에서 LwF.MC를 명백히 능가하며 이는 representative memory의 중요성을 강조한다.

Fixed number of samples

- 이 실험에서는 기존 클래스별 훈련 샘플 수를 고정하여 모델을 훈련시킨다.
    - 따라서 메모리 사이즈가 고정되는 앞선 실험에 비해, 클래스 수에 비례하여 메모리가 증가한다.
    - 추가로, 정확도 측면에서 샘플 수의 영향을 측정하기 위해, 클래스별 샘플 수를 50, 75, 100으로 평가한다.
    - 5, 10, 20 의 incremental step에서 실험을 진행한다.
    - 결과를 비교할 수 있도록 iCaRL과 본 논문의 방법에서 동일한 클래스 순서를 사용한다.
    - LwF.MC가 낮은 성능을 보이므로, iCaRL 및 Hybrid1과 성능을 비교한다.

# classes	5			10			20		
# img / cls	50	75	100	50	75	100	50	75	100
Our-CNN	**62.4**	**66.9**	**68.6**	**62.7**	**65.7**	**68.5**	**63.3**	**65.4**	**67.3**
iCaRL	56.5	59.9	62.2	60.0	62.3	63.7	61.9	63.0	64.0
Hybrid1	45.7	49.2	50.9	55.3	56.5	57.4	60.4	61.5	62.2

(a) Fixed number of samples

# classes	5	10	20
Our-CNN-Base	57.0	53.7	50.1
Our-CNN-DA	59.2	57.9	57.2
Our-CNN-BF	57.9	58.1	57.1
Our-CNN-Full	**63.8**	**64.0**	**63.2**
iCaRL	58.8	60.9	61.2
Hybrid1	48.7	55.1	59.8

(b) Ablation study

Table 2: **Accuracy on CIFAR-100.** Each row represents a different approach. The best results are marked in bold. See the main text for more details.

- Tab. 2(a)는 실험의 결과를 요약하고 있다.
  - Table의 첫 번째 행은 각 incremental step별 클래스의 수이며, 두 번째 행은 훈련 중에 사용한 기존 클래스별 exemplar의 개수이고, 나머지 행은 평가한 방법들의 결과이다.
  - Our-CNN과 iCaRL류 방법들을 비교한 결과, 모든 시나리오에서 본 논문의 방법이 더 좋은 성능을 보여줌을 알 수 있다.
  - Sec. 6.1 의 fixed memory size 실험과 같이, 논문의 방법은 5, 10, 20 클래스의 incremental step크기에 대해 비슷한 평균 정확도를 달성하며, 그 안정성을 확인할 수 있다.

- 훈련에서 클래스별 exemplar 수에 따른 영향을 측정하기 위해, Tab. 1(a) 의 결과를 Tab. 2(a)의 결과와 비교한다.
- 50 exemplar의 경우, Tab. 1의 성능보다 좋지 못하며 그 이유는 초기 incremental step 에서 사용할 수 있는 exemplar의 수가 더 적고 초기 모델이 충분히 훈련되지 않기 때문이다.
  - 이로 인해 연쇄 효과가 발생하여 더 많은 exemplar를 사용할 수 있는 경우에도 최종 단계에서 얻은 모델이 예상보다 좋지 못한 성능을 보여준다.

Ablation studies

- 여기에서는 본 논문의 접근 방법의 컴포넌트를 분석하고 최종 성능에서 이 컴포넌트들이 미치는 영향을 보여준다.
  - 모든 ablation은 fixed memory 설정에서 진행된다.

- 먼저 10개의 클래스의 incremental step을 사용한 실험에서 herding, random, histogram 의 세가지 sample selection 전략을 평가한다.
  - Herding: Sec. 3.1 에서 언급한 본 논문의 선택 방법이다.
  - Random: 샘플이 메모리에 무작위로 저장되도록 하는 방법이다.
  - Histogram: 샘플이 클래스 평균으로부터의 거리에 따라 선택된다. 먼저 10개 bin으로 거리의 히스토그램을 계산하고 각 샘플을 이 bin 중 하나에 할당한다. 그 다음 포함된 샘플의 비율에 따라 각 bin에서 샘플을 선택한다.

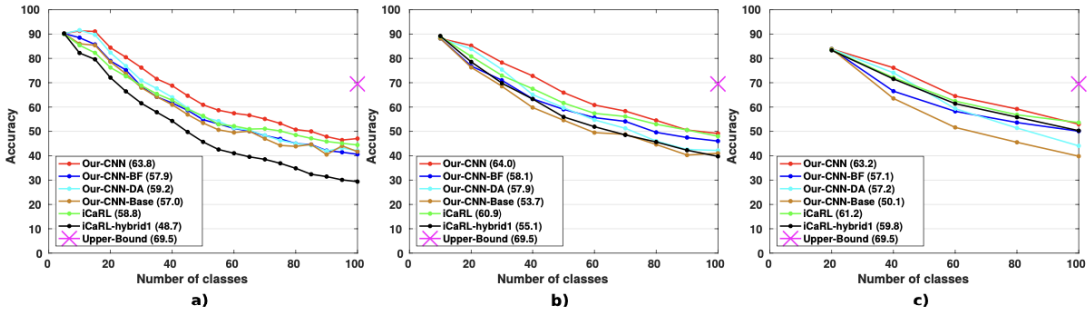

Fig. 4: **Ablation study with CIFAR-100.** Results for (a) 5, (b) 10, and (c) 20 classes. The average over all the incremental steps is shown in parentheses for each method. (Best viewed in pdf.)

- 다음 ablation에서는 augmentation 과 fine-tuning이 미치는 영향을 분석한다.
  - Our-CNN-DA: 데이터 증강을 사용하나 balanced fine-tuning을 사용하지 않는 모델.
  - Our-CNN-BF : Balanced find-tuning을 사용하나 데이터 증강을 사용하지 않는 모델.
  - Our-CNN-Base: 데이터 증강과 balanced fine-tuning 모두 사용하지 않는 모델.
  - 여기에서는 incremental step이 5, 10, 20 클래스인 상황에서 실험을 진행한다.
  - 이전 실험과 같이 비교를 위해 첫 번째 분할은 똑같은 클래스 순서로 실행된다.
- Tab. 2(b)와 Fig. 4에서 연구 결과를 확인 할 수 있으며 Baseline인 Our-CNN-Base은 모든 경우에서 가장 나쁜 성능을 보인다.
- Our-CNN-DA의 경우에는 데이터 증강을 추가하여 5개 클래스 크기에서 좋은 성능을 보여준다.
  - 하지만 이전 클래스와 새 클래스 사이의 샘플 수가 불균형하므로, 더 큰 incremental step에서 balanced fine-tuning 을 추가해야한다. (Our-CNN-BF)

- Our-CNN-BF의 경우 모든 경우에서 향상된 결과를 보여주며 특히 큰 incremental step에서 그 효과가 커 balanced training set의 중요성을 강조하고 있다.
- 마지막으로, 두 컴포넌트를 사용한 경우 (Our-CNN-Full), 최고의 결과를 확인할 수 있으며, 본 논문의 모델이 incremental learning 에서 이 데이터셋으로 SOTA를 달성함을 알 수 있다.

Evaluation on ImageNet

Dataset

- ImageNet Large-Scale Visual Recognition Challenge 2012는 ImageNet의 서브셋을 사용하는 대회이다.
  - 이 서브셋은 클래스별로 1000장 이상의 이미지를 담고 있는 1000가지 클래스로 구성된다.
  - 총 약 120만 장의 훈련 이미지와 5만 장의 검증 이미지, 그리고 15만장의 테스트 이미지가 있다.
- 여기에서는 이 데이터셋으로 두 가지 실험을 진행한다.
  - 첫 번째 실험에서는 무작위로 100가지 클래스를 선택하고 이를 무작위로 10클래스 분할로 나눈다.
  - 두 번째 실험에서는 1000클래스를 무작위로 선택된 100가지 클래스 분할로 나눈다.
  - 결과를 비교할 수 있도록 모든 접근 방법에 대해 동일한 클래스 집합을 사용한다.
  - 각 incremental step 이후 만들어진 모델은 훈련된 모든 클래스로 구성된 테스트 데이터로 평가된다.
  - 실험은 한 번 진행하며 각incremental step 이후 top -5 정확도를 기록한다.
  - Sec. 6에서 설명한 평균 incremental accuracy 또한 기록한다.
- Sec. 5에서 설명한 데이터 증강을 사용하며, 각 훈련 샘플마다 거울 이미지를 생성한 이후 무작위로 모든 이미지에 변환을 가한다.
  - 따라서 데이터 증강 이후 훈련 샘플의 수는 두 배가 된다.

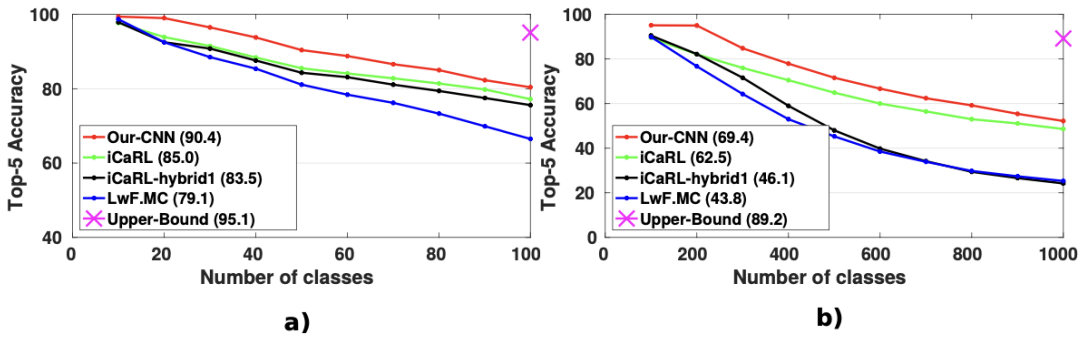

Fig. 5: **Accuracy on ImageNet.** One execution with (a) 10 and (b) 100 classes per incremental step. Average of the incremental steps is shown in parentheses for each method. (Best viewed in pdf.)

- 결과를 비교할 수 있도록 평가된 모든 방법에 대해 동일한 클래스 순서를 유지한다.
  - 또한 iCaRL 및 Hybrid1과의 공정한 비교를 위해 iCaRL 논문의 프로토콜을 따른다.

- Tab. 1(b)는 fixed memory size 의 결과를 요약하며, Fig. 5는 10, 100 클래스의 incremental step을 보여준다.
  - Upper-Bound 는 모든 클래스 훈련 샘플을 사용한 non-incremental model이다.
  - 실험 결과, 본 논문의 방법은 두 가지 경우 모두 기존 평균 결과로부터 5%나 향상시킨 새로운 SOTA가 되었다.
  - 이를 통해 논문의 방법이 또한 많은 클래스가 포함된 커다란 데이터셋에서도 적합하다는 것을 알 수 있다.
  - 게다가 CIFAR-100보다 신규 및 기존 클래스의 샘플 수가 더 균형을 이루기 때문에 incremental step이 100클래스로 크더라도 좋은 정확도를 보여준다.