https://arxiv.org/pdf/1611.07725.pdf

iCaRL: Incremental Classifier and Representation Learning

Abstract

only the training data for a small number of classes has to be present at the same time and new classes can be added progressively

Introduction

three properties of an algorithm to qualify as class-incremental

1 it should be trainable from a stream of data in which examples of different classes occur at different times,

2 it should at any time provide a competitive multi-class classifier for the classes observed so far,

3 its computational requirements and memory footprint should remain bounded, or at least grow very slowly, with respect to the number of classes seen so far.

three components of iCarl

classification by a nearest-mean-of-exemplars rule

prioritized exemplar selection based on herding

representation learning using knowledge distillation and prototype rehearsal

Method

Class-Incremental Classifier Learning

Classification

Algorithm 1 iCaRL CLASSIFY

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

Training

Algorithm 2 iCaRL INCREMENTALTRAIN

input $X^s, \ldots, X^t$ // training examples in per-class sets
input K // memory size
require Θ // current model parameters
require $\mathcal{P} = (P_1, \ldots, P_{s-1})$ // current exemplar sets
  $\Theta \leftarrow$ UPDATEREPRESENTATION$(X^s, \ldots, X^t; \mathcal{P}, \Theta)$

$m \leftarrow K/t$ // number of exemplars per class
for $y = 1, \ldots, s-1$ **do**
  $P_y \leftarrow \text{REDUCEEXEMPLARSET}(P_y, m)$
end for
for $y = s, \ldots, t$ **do**
  $P_y \leftarrow \text{CONSTRUCTEXEMPLARSET}(X_y, m, \Theta)$
end for
$\mathcal{P} \leftarrow (P_1, \ldots, P_t)$ // new exemplar sets

Architecture

Note that even though one can interpret these outputs as probabilities, iCaRL uses the network only for representation learning, not for the actual classification step

Resource usage

it can –in theory– run for an unlimited amount of time

If an upper bound on the number of classes is known
simply pre-allocate space for as many weight vectors as required and use all remaining available memory to store exemplars

Without an upper limit
one would actually grow the number of weight vectors over time, and decrease the size of the exemplar set accordingly.
at least one exemplar image and weight vector is required for each classes

Nearest-Mean-of-Exemplars Classification

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

Background

기존 : whenever ϕ changes, all w1, . . . , wt must be updated as well
catastrophic forgetting

iCarl:
The class-prototypes automatically change whenever the feature representation changes, making the classifier robust against changes of the feature representation
use the average over a flexible number of exemplars that are chosen in a way to provide a good approximation to the class mean

Representation Learning

Algorithm 3 iCaRL UPDATEREPRESENTATION

input $X^s, \ldots, X^t$ // training images of classes $s, \ldots, t$

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

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

// store network outputs with pre-update parameters:
for $y = 1, \ldots, s-1$ **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

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

that consists of *classification* and *distillation* terms.

classificaiton loss
distillation loss

background
two modifications to plain finetuning that aim at preventing or at least mitigating catastrophic forgetting
1 the training set is augmented
exemplars are stored as images, not in a feature representation

2 the loss function is augmented
distillation loss - not lost during the new learning step

Exemplar Management
m = K/t exemplars (up to rounding) for each class

two routines are responsible for exemplar management
1 select exemplars for new classes

Algorithm 4 iCaRL CONSTRUCTEXEMPLARSET

input image set $X = \{x_1, \ldots, x_n\}$ of class y
input m target number of exemplars

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

2 reduce the sizes of the exemplar sets

Algorithm 5 iCaRL REDUCEEXEMPLARSET

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

The order of its elements matters, with exemplars earlier in the list being more important.
각 세트별로 결국 끝까지 남는 것은 맨 앞에 있는 요소임 - 그래서 위에서 잘 만들어야함

Background
two objectives
initial exemplar set should approximate the class mean vector well
possible to remove exemplars at any time during the algorithm's runtime without violating this property.

Related Work

Learning with a fixed data representation
nearest neighbor - 모든 data를 메모리 위에 올려야해서 불가능
NCM
NCM represents each class as a prototype vector that is the average feature vector of all examples observed for the class so far.
NCM has been shown to work well and be more robust than standard parametric classifiers in an incremental learning setting
it cannot easily be extended to the situation in which a nonlinear data representation should be learned together with the classifiers

변형
only over a specifically chosen subset, which allows us to keep a small memory footprint and perform all necessary updates with constant computational effort.

Representation learning
recent success of (deep) neural networks

suitable data representations

catastrophic forgetting
overwrite
these classical works were mainly in the context of connectionist memory networks, not classifiers
networks used were small and shallow

two main strategies of how catastrophic forgetting can be addressed:
1 by freezing parts of the network weights while at the same time growing the network in order to preserve the ability to learn
2 rehearsal

mainstream - freeze strategy
requires allocating more and more resources to the network over time

icarl - rehearsal
old data
distillation

Experiments

Benchmark protocol
classes are arranged in a fixed random order
After each batch of classes, the resulting classifier is evaluated on the test part data of the dataset, considering only those classes that have already been trained
average incremental accuracy.

iCIFAR-100 benchmark
iILSVRC benchmark

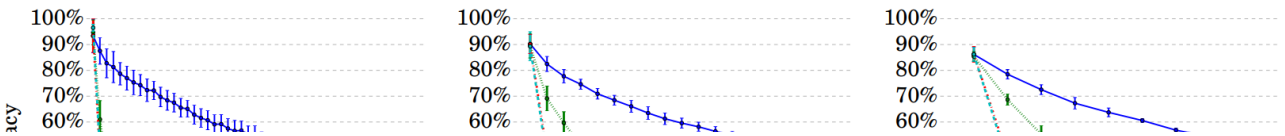
iCarl implementation

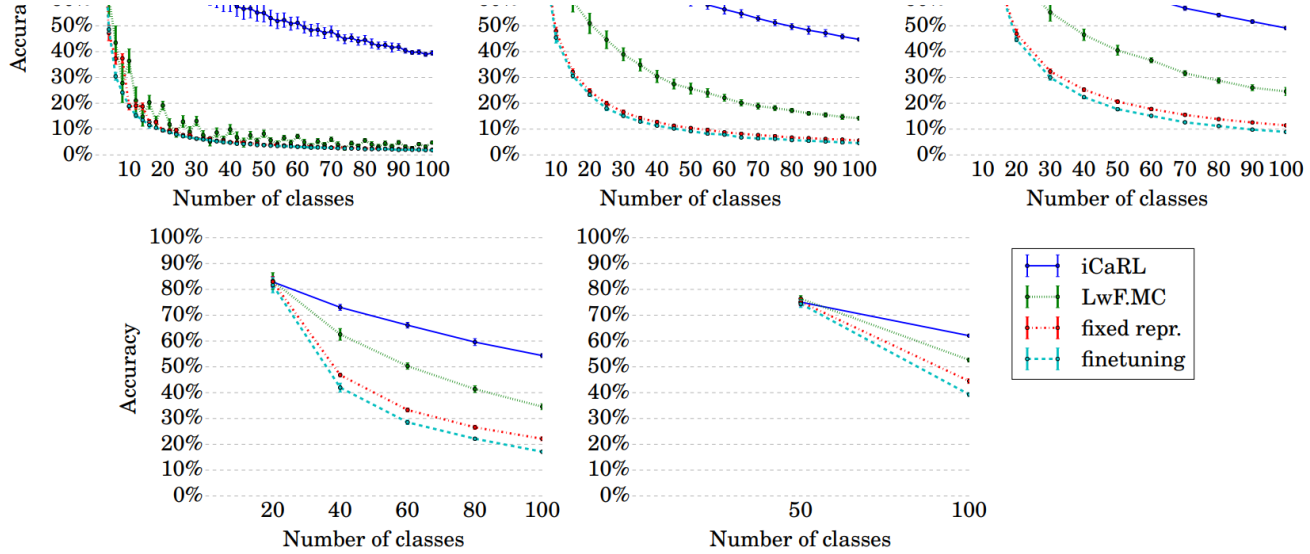
Results

three alternative class-incremental methods
Finetuning

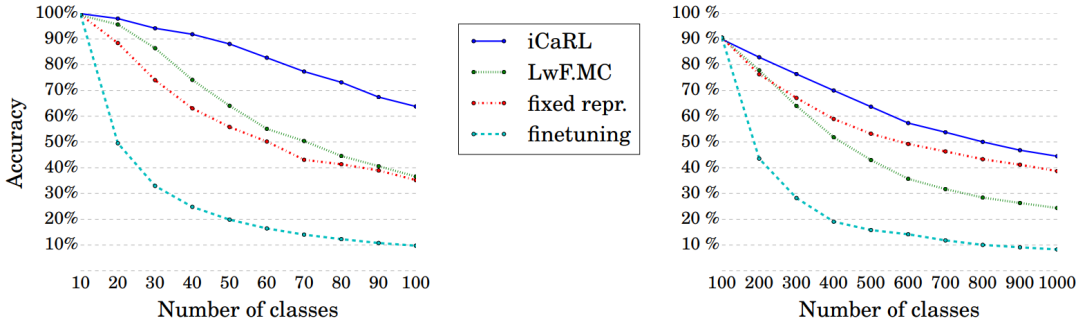
Fixed representation
freezes the feature representation after the first batch of classes has been processed and the weights of the classification layer after the corresponding classes have been processed

distillation loss
like iCaRL does, but that does not use an exemplar set.



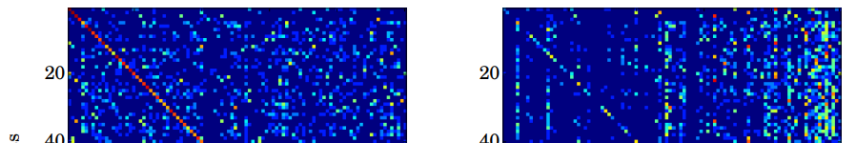


icifar-100 with 2, 5, 10, 20, 50 classes per batch



(b) Top-5 accuracy on iILSVRC-small (top) and iILSVRC-full (bottom).

icarl > distillation > fix > finetuning



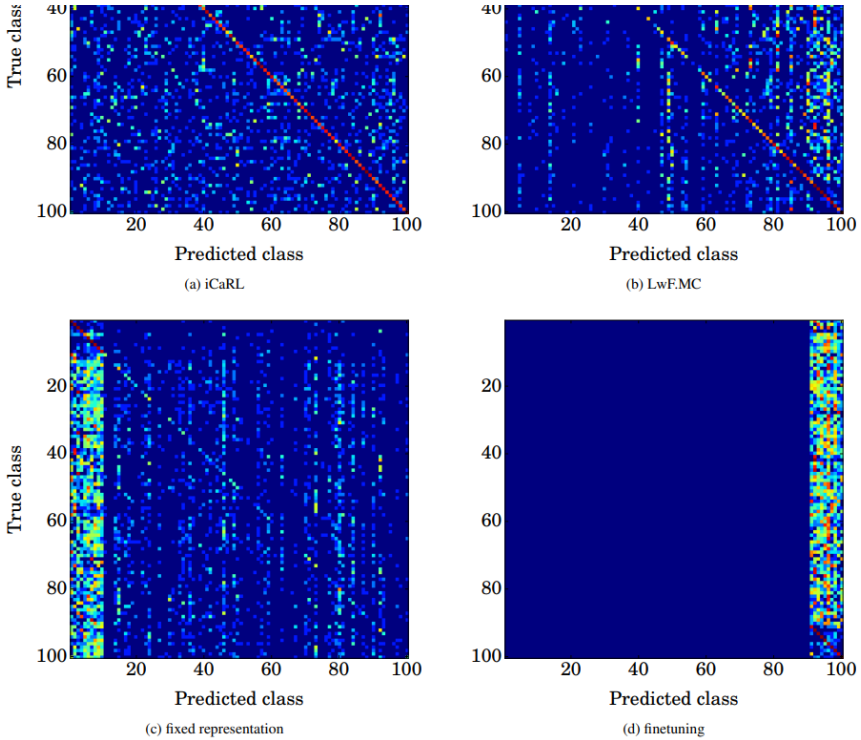


Figure 3: Confusion matrices of different method on iCIFAR-100 (with entries transformed by $\log(1+x)$ for better visibility). iCaRL's predictions are distributed close to uniformly over all classes, whereas LwF.MC tends to predict classes from recent batches more frequently. The classifier with fixed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

The finetuned network simply forgot that earlier classes even exist

Differential Analysis

why exactly iCaRL improves over plain finetuning-based training, from which it differs in three aspects

1 mean-of-exemplars classification rule,

2 exemplars during the representation learning

3 distillation loss

three hybrid setups

learns a representation in the same way as iCaRL, but uses the network's outputs directly for classification, not the mean-of-exemplar classifier

the exemplars for classification, but does not use the distillation loss during training

uses neither the distillation loss nor exemplars for classification, but it makes use of the exemplars during representation learning

LwF.MC again, which uses distillation, but no exemplars at all

batch size	iCaRL	*hybrid1*	*hybrid2*	*hybrid3*	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

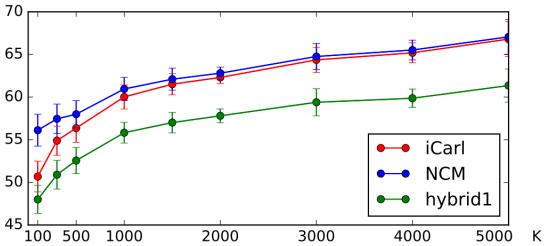


Figure 4: Average incremental accuracy on iCIFAR-100 with 10 classes per batch for different memory budgets K.

All method benefit from a larger memory budget, showing that iCaRL's representation learning step indeed benefits from more prototypes

Conclusion

three main components

1) a nearest-mean-of-exemplars classifier that is robust against changes in the data representation while needing to store only a small number of exemplars per class,

2) a herding-based step for prioritized exemplar selection,

3) a representation learning step that uses the exemplars in combination with distillation to avoid catastrophic forgetting

use of exemplar images