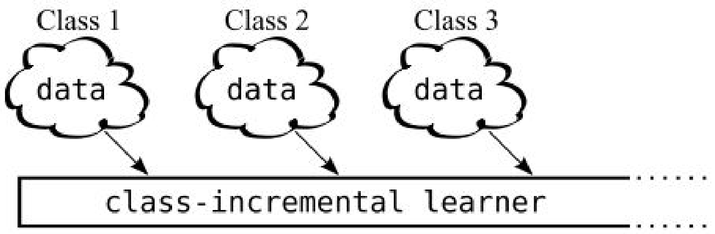
iCarRL: Incremental Classifier and Representation Learning.

Sylvestre-Alvise Rebuffi- University of Oxford/IST Austria

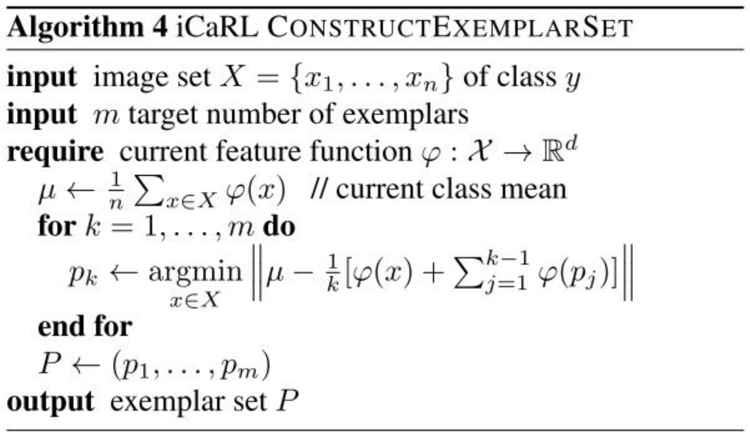
CVPR_2017

This paper proposed a method that learns data representation and Incremental Learning Classifier (using 32-Layer ResNet) simutaneously. The paper experiments on image datasets (CIFAR100, ILSVRC). .

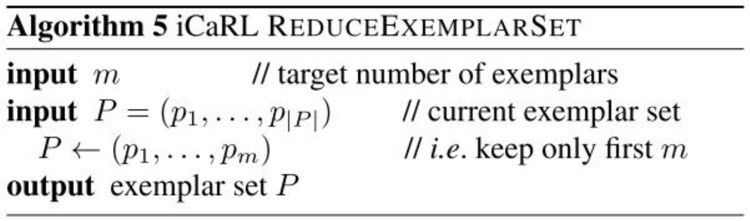


Training/Testing Procudure. The image dataset is split into training/testing sets with a ratio (let's say 90/10). The training is conducted online, where a portion of training sets is fed into the network in each training step. For example, at 1st training step, we can feed 2 classes of dog an cat (600 images/class); at 2nd training step, images of next 2 classes are fed into the network. We can also change the number of classes in each training step to 5,10,...etc. The trained networks in each training steps are tested on testing datasets (10% of data). However, we only test on images of the trained classes.

Data Representation. For every incremental step, the samples (images) of each class for old and new classes are constructed. The total samples must not be excceed K, thus each class will have max $m = K/t$, given t classes are presented.



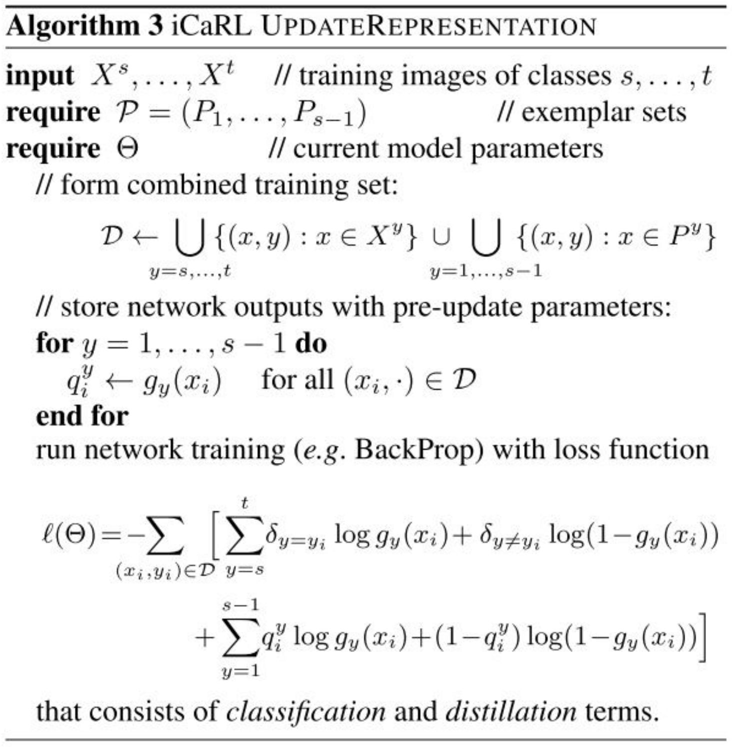
```
Algorithm 4 iCaRL ConstructExemplarSet
input image set $X = \{x_1, \dots, x_n\}$ of class $y$
input $m$ target number of exemplars
require current feature function $\varphi : \mathcal{X} \to \mathbb{R}^d$
  $\mu \leftarrow \frac{1}{n}\sum_{x \in X} \varphi(x)$   // current class mean
  for $k = 1, \dots, m$ do
    $p_k \leftarrow \underset{x \in X}{\operatorname{argmin}} \left\| \mu - \frac{1}{k}[\varphi(x) + \sum_{j=1}^{k-1} \varphi(p_j)] \right\|$
  end for
  $P \leftarrow (p_1, \dots, p_m)$
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 $P$
```

Incremental Learning Given m samples of each class (old and new ones), the steps of incremental learning is as follows:

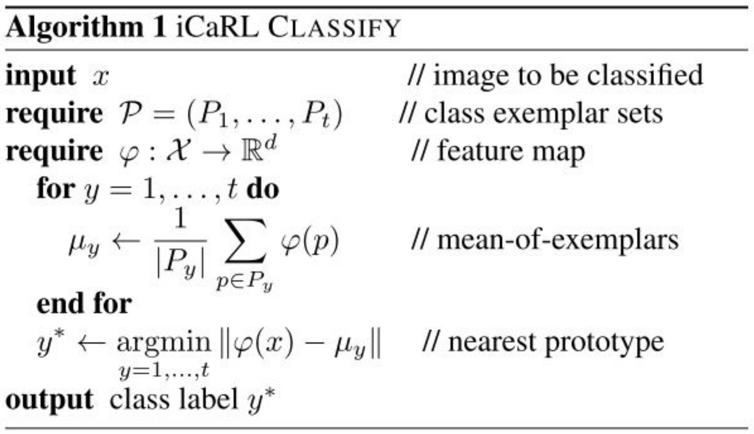
1. All samples are gathered into set D
2. Calculate network outputs with pre-update network parameters. These outputs are used to minimize the distillation loss in next steps.
3. All samples in set D are used to minimize the loss function consisting of **distillation loss** and **classification loss**. The intuition of distillation loss is that we want the new (updated) network weights to generate the same outputs for samples of old classes in step 2. This helps solve the catastrophic forgeting problem. At the same time, the classification loss discriminates samples in entire classes. The algorithm is as followed.



Algorithm 3 iCaRL UPDATEREPRESENTATION

```
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
    $\ell(\Theta) = -\sum_{(x_i,y_i)\in\mathcal{D}} \Big[ \sum_{y=s}^{t} \delta_{y=y_i} \log g_y(x_i) + \delta_{y\neq y_i} \log(1-g_y(x_i))$
        $+ \sum_{y=1}^{s-1} q_i^y \log g_y(x_i) + (1-q_i^y) \log(1-g_y(x_i)) \Big]$
  that consists of classification and distillation terms.
```

Classification: The idea is simmilar to the concept of K-Means. The mean of exemplars (samples that are selected) are calculated using the feature vector of each exemplar. The predicted class of a testing image is the nearest class to that image in feature spaces. The algorithm is below:



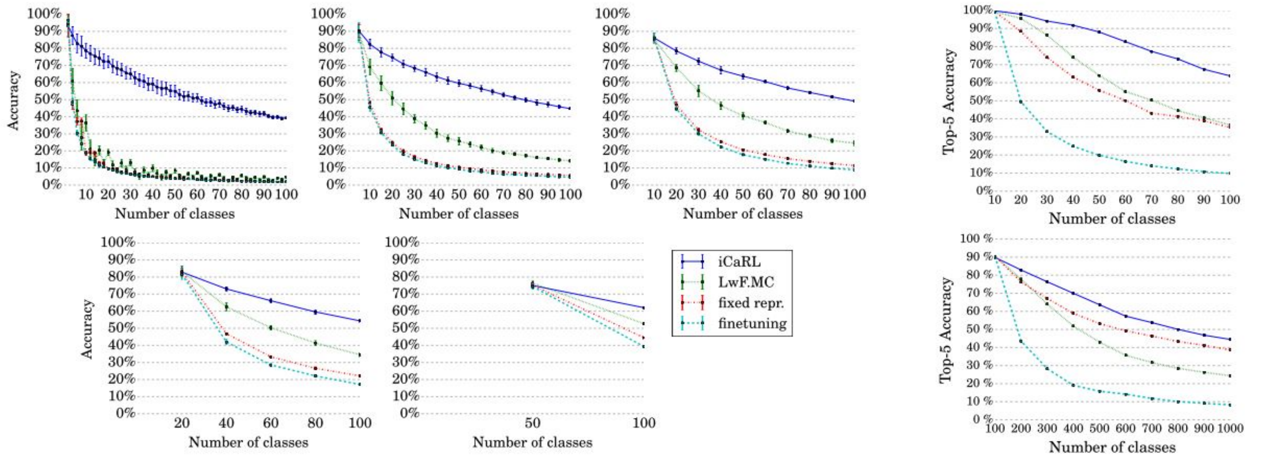
Algorithm 1 iCaRL CLASSIFY

```
input $x$     // image to be classified
require $\mathcal{P} = (P_1, \dots, P_t)$     // class exemplar sets
require $\varphi : \mathcal{X} \to \mathbb{R}^d$     // 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 $y^*$
```

The arguments for using nearest-mean-of-exemplars instead of usual network outputs (softmax) are not clear in this paper. However, the ablation study shows the improved accuracy of using nearest-mean-of-exemplars.

Results.

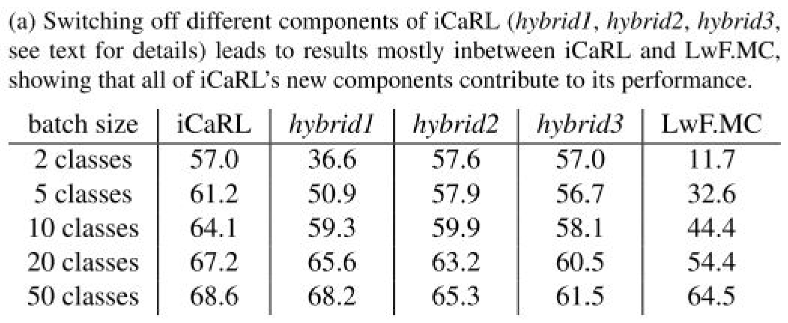
1. Compare with previous methods on CIFAR100 and ILSVRC datasets.



(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).

2. **Confusion matrices**
3. **Ablation study:** hybrid1- use classification outputs. hybrid2- not use distillation loss. hybrid3- not use distillation loss + exemplars for classication, but use exemplars for represetnation learning. lwF.MC- a related method.



(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	*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

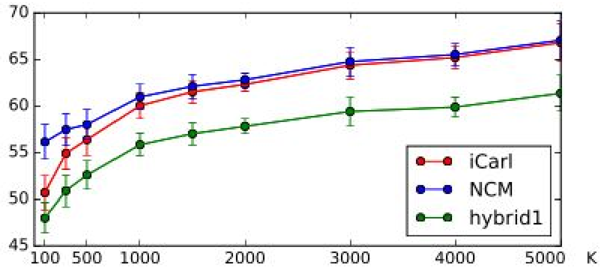


Figure 4: Average incremental accuracy on iCIFAR-100 with 10 classes per batch for different memory budgets K.

In [ ]: