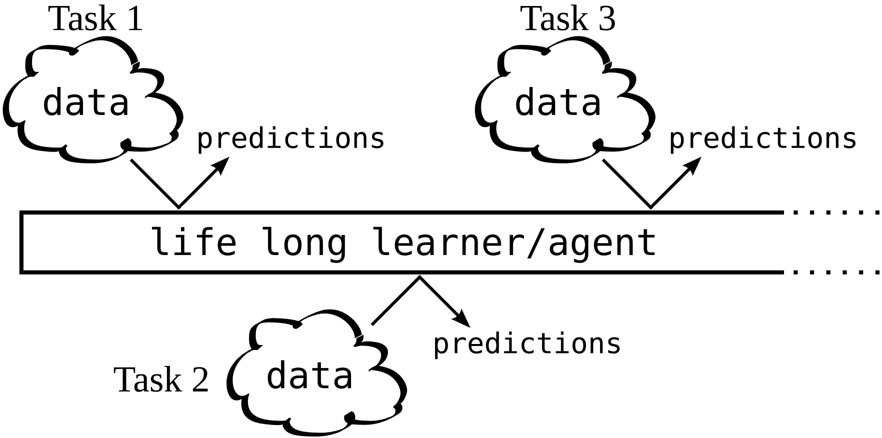
Incremental Classifier and Representation Learning

Christoph Lampert

IST AUSTRIA
Institute of Science and Technology

Workshop at Genova
March 9–10, 2017

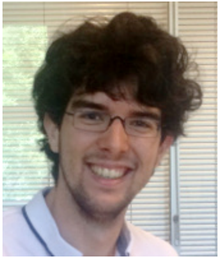
Continuously improving open-ended learning



Lifelong Learning

A few years after the deep learning revolution...

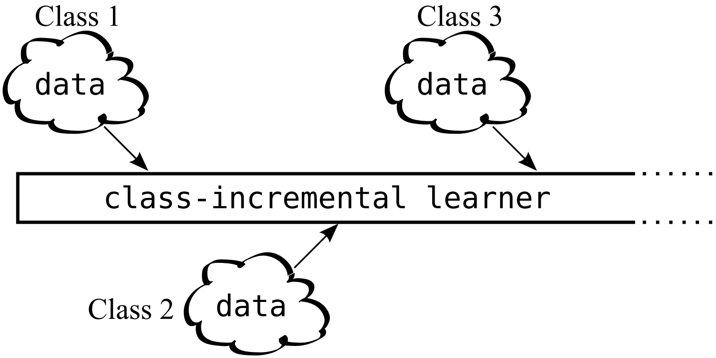
A few years after the deep learning revolution...

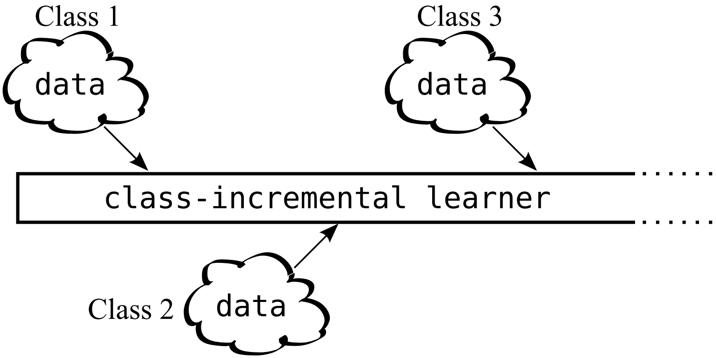


Sylvestre Rebuffi
(U Oxford/IST Austria)

Alex Kolesnikov
IST Austria

Class-Incremental Learning





Input: a stream of data, examples of different classes occur at different times,

Output: at any time, a competitive multi-class classifier for the classes observed so far,

Conditions: computational requirements and memory footprint remain bounded (or at least grow very slowly) with respect to the number of classes seen so far.

Class-Incremental Learning

- feature function $\boldsymbol{\varphi} : \mathcal{X} \to \mathbb{R}^d$
- classifiers: $g_y(x) = \dfrac{1}{1 + e^{-\langle w_y, \boldsymbol{\varphi}(x) \rangle}}$ for each $y \in \mathcal{Y}$ seen so far
- data comes in class batches: $X^s, \dots, X^t$ where all examples in X^y are of class y

- feature function $\varphi : \mathcal{X} \to \mathbb{R}^d$
- classifiers: $g_y(x) = \dfrac{1}{1 + e^{-\langle w_y, \varphi(x) \rangle}}$ for each $y \in \mathcal{Y}$ seen so far
- data comes in class batches: $X^s, \dots, X^t$ where all examples in X^y are of class y

Idea 1: incremental multi-class training, e.g., using stochastic gradient descent:

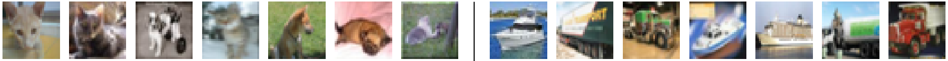
x

w_{cat}	1	1	-1	1	-1	-1	1
w_{dog}	-1	-1	1	-1	1	1	-1

- after being trained on a set of classes, classifier parameters make sense

- feature function $\varphi : \mathcal{X} \to \mathbb{R}^d$
- classifiers: $g_y(x) = \dfrac{1}{1 + e^{-\langle w_y, \varphi(x)\rangle}}$ for each $y \in \mathcal{Y}$ seen so far
- data comes in class batches: $X^s, \dots, X^t$ where all examples in X^y are of class y

Idea 1: incremental multi-class training, e.g., using stochastic gradient descent:

x														
w_{cat}	1	1	-1	1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
w_{dog}	-1	-1	1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
w_{boat}								1	-1	-1	1	1	-1	-1
w_{truck}								-1	1	1	-1	-1	1	1

- after being trained on a set of classes, classifier parameters make sense
- every later sample is a negative example for earlier classes $\to$ parameters w_y deteriorate

- feature function $\varphi : \mathcal{X} \to \mathbb{R}^d$
- classifiers: $g_y(x) = \dfrac{1}{1 + e^{-\langle w_y, \varphi(x) \rangle}}$ for each $y \in \mathcal{Y}$ seen so far
- data comes in class batches: $X^s, \dots, X^t$ where all examples in X^y are of class y

Idea 2: fix classifier parameters after each class batch, train only new ones:

x														
w_{cat}	1	1	-1	1	-1	-1	1							
w_{dog}	-1	-1	1	-1	1	1	-1							
w_{boat}								1	-1	-1	1	1	-1	-1
w_{truck}								-1	1	1	-1	-1	1	1

- feature function $\varphi : \mathcal{X} \to \mathbb{R}^d$
- classifiers: $g_y(x) = \dfrac{1}{1 + e^{-\langle w_y, \varphi(x) \rangle}}$ for each $y \in \mathcal{Y}$ seen so far
- data comes in class batches: $X^s, \dots, X^t$ where all examples in X^y are of class y

Idea 2: fix classifier parameters after each class batch, train only new ones:

x														
w_{cat}	1	1	-1	1	-1	-1	1							
w_{dog}	-1	-1	1	-1	1	1	-1							
w_{boat}								1	-1	-1	1	1	-1	-1
w_{truck}								-1	1	1	-1	-1	1	1

- classifiers of different batches are trained independently $\to$ batches not separated

- feature function $\varphi : \mathcal{X} \to \mathbb{R}^d$
- classifiers: $g_y(x) = \dfrac{1}{1 + e^{-\langle w_y, \varphi(x) \rangle}}$ for each $y \in \mathcal{Y}$ seen so far
- data comes in class batches: $X^s, \dots, X^t$ where all examples in X^y are of class y

Idea 2: fix classifier parameters after each class batch, train only new ones:

x														
w_{cat}	1	1	-1	1	-1	-1	1							
w_{dog}	-1	-1	1	-1	1	1	-1							
w_{boat}								1	-1	-1	1	1	-1	-1
w_{truck}								-1	1	1	-1	-1	1	1

- classifiers of different batches are trained independently $\to$ batches not separated

If data representation is also trained, even fixing old w_y is not enough!

When φ changes but w_y does not, outputs deteriorate $\to$ *"catastrophic forgetting"*

[McCloskey, Cohen. *"Catastrophic interference in connectionist networks: The sequential learning problem"*, Psychology of learning and motivation, 1989]

Idea 3: Nearest-Class-Mean (NCM) classifier [Mensink *et al.* 2013]

$$y^* = \operatorname*{\mathbf{argmin}}_{y \in \mathcal{Y}} \|\boldsymbol{\varphi}(x) - \mu_y\|^2$$

$$\text{for} \qquad \mu_y = \frac{1}{|\{i : y_i = y\}|} \sum_{\{i:y_i=y\}} \boldsymbol{\varphi}(x_i)$$

Advantage:

- class mean μ_y does not deteriorate when new samples come in
- even classes within different batches 'compete'

Problem:

- when φ changes, we have to recompute μ_y
  $\rightarrow$ we need to store all training examples
  $\rightarrow$ does not fulfill condition for 'class-incremental'

[Mensink, Verbeek, Perronnin, Csurka. "Distance-based image classification: generalizing to new classes at near-zero cost.", TPAMI 2013]

Proposal:

iCaRL (Incremental Class and Representation Learning) [Rebuffi *et al.*, CVPR 2017]

Internal representation:

- feature function $\varphi : \mathcal{X} \to \mathbb{R}^d$, weight vectors w_y for $y \in \mathcal{Y}$
- for each seen class, y, a set of exemplar samples, P_y, (in total up to K samples)

For representation learning:

- probabilistic outputs: $g_y(x) = \dfrac{1}{1 + e^{-\langle w_y, \varphi(x) \rangle}}$ for each $y \in \mathcal{Y}$ seen so far

For classification:

- classify samples by their distance to a class prototype (like NCM does), but using the mean of exemplars, not the mean of all training examples

[S. Rebuffi, A. Kolesnikov, CHL. *"iCaRL: Incremental Class and Representation Learning"*, CVPR 2017]

$y^* \leftarrow \textsc{NearestMeanOfExemplars}(x)$

input x // sample to be classified
require $\mathcal{P} = (P_1, \dots, P_t)$ // class exemplar sets of classes $1, \dots, t$ seen so far
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^*

$\textsc{IncrementalTrain}(X^s, \dots, X^t, K)$

input $X^s, \dots, X^t$ // new training examples in per-class sets (all $x \in X^y$ are of class y)
input K // maximum number of exemplars
require Θ // current model parameters
require $\mathcal{P} = (P_1, \dots, P_{s-1})$ // current exemplar sets

$\Theta \leftarrow \textsc{UpdateRepresentation}(X^s, \dots, X^t; \mathcal{P}, \Theta)$
$m \leftarrow \lfloor K/t \rfloor$ // number of exemplars per class
for $y = 1, \dots, s-1$ **do**
  $P_y \leftarrow \textsc{ReduceExemplarSet}(P_y, m)$
end for
for $y = s, \dots, t$ **do**
  $P_y \leftarrow \textsc{ConstructExemplarSet}(X_y, m, \Theta)$
end for
$\mathcal{P} \leftarrow (P_1, \dots, P_t)$ // new exemplar sets

UPDATEREPRESENTATION($X^s, \ldots, X^t$)

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

require $\mathcal{P} = (P_1, \ldots, P_{s-1})$ // exemplar sets

require Θ // current model parameters

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

for $y = 1, \ldots, s-1$ **do**

$q_i^y \leftarrow g_y(x_i)$ for all $(x_i, \cdot) \in \mathcal{D}$ // store outputs of pre-update network

end for

run network training (e.g. BackProp) with loss function

$$\mathcal{L}(\Theta) = -\sum_{(x_i,y_i)\in\mathcal{D}} \Big[\underbrace{\sum_{y=s}^{t} \ell(\delta_{y=y_i}, g_y(x_i))}_{\text{classification loss}} + \underbrace{\sum_{y=1}^{s-1} \ell(q_i^y, g_y(x_i))}_{\text{distillation loss}} \Big]$$

Two difference to ordinary network learning/finetuning:

- training set

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

consists of samples of new classes, but also exemplars of old classes
$\rightarrow$ representation is 'reminded' of old classes regularly

- loss function

$$\mathcal{L}(\Theta) = -\sum_{(x_i,y_i)\in\mathcal{D}} \Big[\underbrace{\sum_{y=s}^{t} \ell(\delta_{y=y_i}, g_y(x_i))}_{\text{classification (cross-entropy) loss}} + \underbrace{\sum_{y=1}^{s-1} \ell(q_i^y, g_y(x_i))}_{\text{distillation loss}} \Big]$$

contains not just ordinary classification term, but also *distillation* term [Hinton *et al.*, 2014]
$\rightarrow$ encourage network to preserve its output values across training steps [Li, Hoiem. 2016]

[G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In NIPS Workshop on Deep Learning, 2014]
[Z. Li and D. Hoiem. Learning without forgetting. In European Conference on Computer Vision (ECCV), 2016]

iCaRL: Exemplar Management

$P \leftarrow \textsc{ReduceExemplarSet}(P, m)$

input $P = (p_1, \ldots, p_{|P|})$ // current exemplar set
input m // target number of exemplars

output exemplar set $P = (p_1, \ldots, p_m)$ // keep only first m exemplars

$P \leftarrow \textsc{ConstructExemplarSet}(X, m)$

input sample set $X = \{x_1, \ldots, x_n\}$ 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 $k = 1, \ldots, m$ **do**
  $p_k \leftarrow \underset{x \in X}{\mathbf{argmin}} \left\| \mu - \frac{1}{k}[\varphi(x) + \sum_{j=1}^{k-1} \varphi(p_j)] \right\|$ // next exemplar
end for

output exemplar set $P = (p_1, \ldots, p_m)$

CIFAR-100 dataset:

- 60K low-resolution images (50K train, 10K test)
- 100 classes

iCaRL setup:

- 32-layer ResNet [He *et al*. 2015]
- $K = 2000$ exemplars
- $2, 5, 10, 20$ or 50 classes per batch

[K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv:1512.03385 (cs.CV)]

iCaRL: Experiments

ImageNet ILSVRC-2012 dataset:

- $>$1.2M high-resolution images (1.2M train, 50K val)
- 1000 classes

iCaRL setup:

- 18-layer ResNet [He *et al*. 2015]
- $K = 20000$ exemplars
- 10 or 100 classes per batch

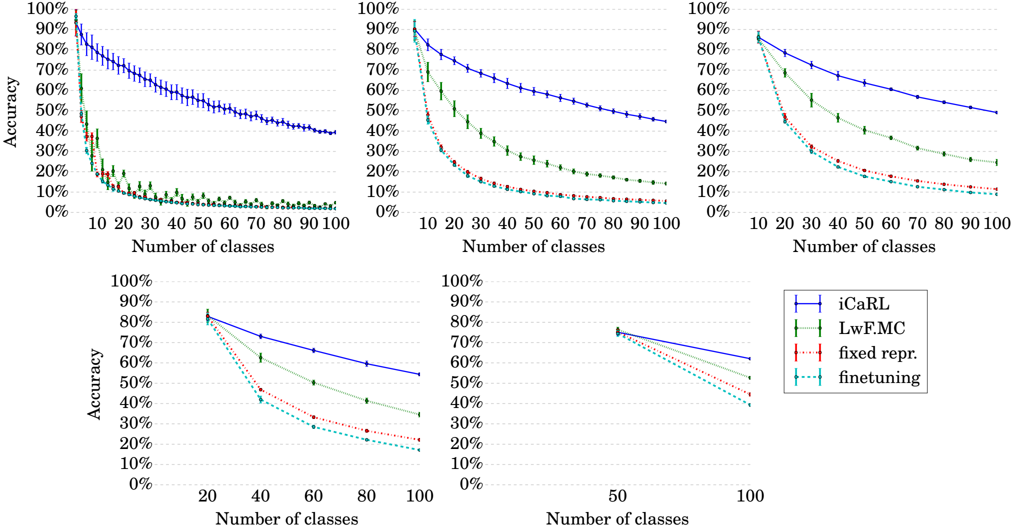
[K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv:1512.03385 (cs.CV)]

Alternative methods:

- finetuning:
  - train network using stochastic gradient descent
  - no measures to prevent catastrophic forgetting
- fixed representation:
  - on the first batch of classes, train the complete network,
  - then, freeze the data representation (all network layers except the last one),
  - for subsequence batches of classes, train only new classifiers
- LwF.MC:
  - train network including distillation loss, but do not use exemplars anywhere
  - resembles multi-class version of *"Learning with Forgetting" (LwF)* [Li and Hoiem, 2016]

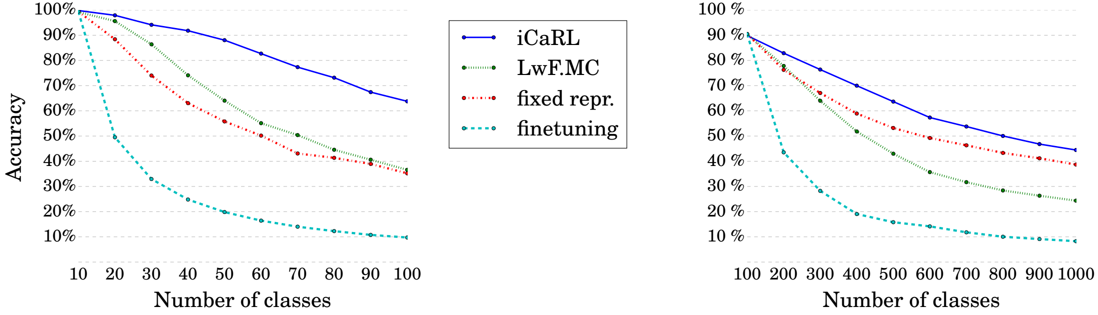
[Z. Li and D. Hoiem. Learning without forgetting. In European Conference on Computer Vision (ECCV), 2016]

Class-Incremental Learning: Results



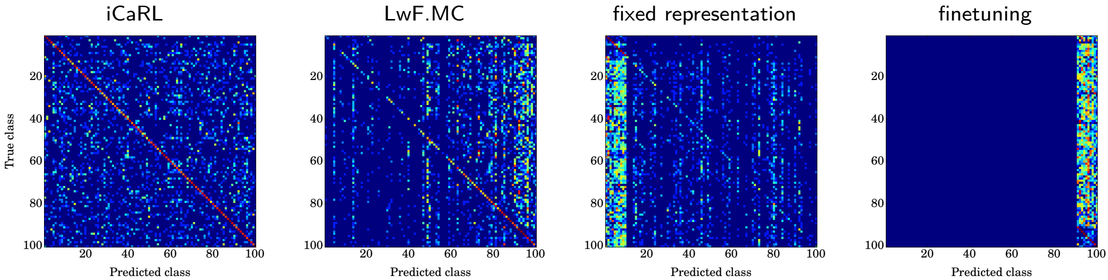
Multi-class accuracies over 10 repeats (average and standard deviation) for class-incremental training on CIFAR-100 with 2 (top left), 5 (top middle), 10 (top right), 20 (bottom left) or 50 (bottom right) classes per batch.

Class-Incremental Learning: Results



Top-5 accuracies for class-incremental training on ILSVRC 2012 with 10 (left) or 100 (right) classes per batch.

Class-Incremental Learning: Results



Confusion matrices of different method on CIFAR-100 after training for 100 classes with 10 classes per batch (entries are transformed by $\log(1+x)$ for better visibility).

- iCaRL: predictions spread homogeneously over all classes
- LwF.MC: prefer recently seen classes $\rightarrow$ some long-term memory loss
- fixed representation: prefer batch of classes seen first $\rightarrow$ lack of neural plasticity
- finetuning: predict only most recently seen classes $\rightarrow$ catastrophic forgetting

Summary and Conclusion

Class-incremental learning is very reasonable, but it is far from solved:

- how to learn a multi-classifier and a representation jointly?
- how to avoid catastrophic forgetting?

iCaRL is our proposal:

- keep a small set of exemplars for each class
- use a mean-of-exemplars classifier rule instead of network outputs
- using distillation during representation learning to avoid catastrophic forgetting

Open questions:

- can other learning strategies be made class-incremental?
- will exemplars be as beneficial for other problem settings?
- what if we cannot store exemplars, e.g. due to privacy/copyright?