LIFELONG-LEARNING
and knowledge distillation from an external memory

Saturday 6\textsuperscript{th} July, 2019

Arthur Douillard
Introduction
General goal of Lifelong-learning

We want to learn **consecutive tasks**, without retraining the model from scratch every time, and without storing all the seen data.

3 scenarios (Lomonaco and Maltoni 2017)

- New **samples** added with potentially new domains (**online learning**)
- New **classes** added (**incremental learning**)
- New **samples & new classes** added
At Heuritech, we need to add every week new garment entity.

Robots in the wild cannot relearn everything due to hardware limitation.

A General Artificial Intelligence should be able to learn continuously like humans do.
Problem definition
Formal definition of Incremental Learning

Let be $T$ tasks $\{D^1, ..., D^T\}$, with $D^i = \{(x^i_1, y^i_1), ..., (x^i_{n_i}, y^i_{n_i})\}$. $x$ being a datum, and $y$ its associated target in a multi-class classification settings.

The classification model at task $t$ is called $\theta^t$.

1. At task $t$, initialize current model: $\theta^t := \theta^{t-1}$.
2. Model $\theta^t$ trains solely on $D^t$.
3. Model $\theta^t$ is evaluated on $\{D^1, ..., D^t\}$ producing accuracy score $a^i$.
4. If there are remaining tasks to learn go to (1.).
5. Compute the Average Incremental Accuracy$^1$: $\frac{1}{T} \sum_{k=1}^{T} a^k$.

$^1$(Rebuffi et al. 2017)
Examples

**iCIFAR100 (Rebuffi et al. 2017)**

- Split CIFAR100 dataset in several tasks
- Ex: 50 tasks of 2 classes each.

**Different datasets**

- Train on different consecutive datasets
- ImageNet -> Birds
(Chaudhry et al. 2018) defines two evaluations settings:

- **Single-head** evaluation: Model is evaluated on all tasks together.

- **Multi-head** evaluation: Model is evaluated on each task separately, knowing beforehand the current task.

Previous slides concerned the single-head evaluation as the vast majority of the literature does. So will we.
Catastrophic forgetting (French 1999)

- Accuracy on previously learned classes is degraded
- Trade-off between **plasticity** (being good on new classes) and **rigidity** (being good on old classes)

Figure 1: Fine-tuning model on iCIFAR100 with 10 tasks of 10 classes
Increment order matter!

The order of the tasks matter a lot. Whether we see *boat & cat* first instead of *plane & car* will change the final results.

Figure 2: Varying models performance depending on the class order

EndToEnd has for results there: 66% (a), 83% (b), 63% (c).
How to solve this problem
Three broad strategies exist in the literature (Parisi et al. 2018):

- **External Memory** storing a limited sample of previous tasks’ data
- **Constraints** making the model more *rigid*
- **Model Plasticity** extending the capacity

They can be used together.
Using an external memory
Strategy 1: External Memory

External memory shows the model previous data & alleviate catastrophic forgetting.

Memory size constraints

- The memory size must be bounded
- The more tasks are added, the less samples can be stored per class

Two variants exist:

- **Reharsal Learning** keeps a subsample of previous data
- **Pseudo-Reharsal Learning** generates data using previous data’s distribution
Constraining the model
Constraints limits the distance between the model at the end of the previous task \( (\theta^{t-1}) \) & the current model \( (\theta^t) \).

Several variants exist, major ones are:

- Constraining the **weights**
- Constraining the **predictions**
- Constraining the **gradients**
Strategy 2: **Constraints on weights**

We add a distance between the weights of the new & old model as a regularization loss:

\[
L_{\text{reg}} = \sum_{i} (\theta_{i}^{t-1} - \theta_{i}^{t})^2
\]

The distance can be weighted by each neuron importance (Kirkpatrick et al. 2017; Aljundi et al. 2018).

\[
L_{\text{reg}} = \sum_{i} \Omega_{i}^{t-1} (\theta_{i}^{t-1} - \theta_{i}^{t})^2
\]

The first (**EWC**) uses the average gradients variance as an importance metric.
Strategy 2: **Constraints** on predictions

**LwF** forces $\theta^t_i$ to have similar predictions with $\theta^{t-1}_i$ for the old targets (Li and Hoiem 2018).

$$L_{\text{distillation}}(y^t_c, y^{t-1}_c) = - \sum_{c=1}^{C} y^{t-1}_c \log(y^t_c)$$

It is similar to the teacher/student of **Knowledge Distillation** (Hinton, Vinyals, and Dean 2015).

A temperature can be used to *soften* the logits:

$$\tilde{y}_i = \frac{y_i^{\frac{1}{\text{Temp}}}}{\sum_{j=1}^{C} y_j^{\frac{1}{\text{Temp}}}}$$
We constrain the loss of $\theta^t$ to lower or equal to the loss of the $\theta^{t-1}$ on the external memory $M$ samples (Lopez-Paz and Ranzato 2017; Aljundi et al. 2019):

$$L(\theta, M) = \frac{1}{|M|} \sum_{(x_i, y_i) \in M} L(\theta(x_i), y_i)$$

$$L(\theta^t, M) \leq L(\theta^{t-1}, M)$$

Note that there is no need to store $\theta^{t-1}$ as long as we ensure iteratively that no update violates the constraint.
Strategy 2: **Constraints on the gradients**

\[ L(\theta^t, M) \leq L(\theta^{t-1}, M) \]

(Lopez-Paz and Ranzato 2017)'s **GEM** rephrased it as an angle constraint on the gradients. We want the gradients "*to go in the same direction*":

\[ \langle \frac{\partial L(\theta(x_i), y_i)}{\partial \theta}, \frac{\partial L(\theta, M)}{\partial \theta} \rangle \geq 0 \]

If this constraint is violated, the gradient \( g \) is projected to its closest valid alternative \( \tilde{g} \):

\[
\text{minimize}_{\tilde{g}} \|g - \tilde{g}\|^2 \\
\text{subject to} \langle g_M, \tilde{g} \rangle \geq 0
\]
Exploiting the model capacity
Strategy 3: Model plasticity

Most algorithms add **new neurons to the classifier** to classify new tasks.

(Yoon et al. 2018)’s **Dynamically Expandable Networks** increases model capacity by adding new neurons to all layers if the model cannot generalize well enough on the new task.

(Fernando et al. 2017; Golkar, Kagan, and Cho 2019)’s **PathNet** and **Neural Pruning** want to exploit better the existing capacity: they use the fact that networks are over-parametrized\(^2\) to uncover sub-networks for each tasks.

---

\(^2\)See Lottery Ticket Hypothesis (Frankle and Carbin 2019)
We will focus on **incremental learning** with the **iCIFAR100** benchmark.

**iCIFAR100** (Rebuffi et al. 2017)

- Split CIFAR100 dataset in several tasks
- Tested with 50 tasks of 2 classes, 20 of 2, 10 of 10, and 2 of 50.

We will base our work on **iCaRL** (Rebuffi et al. 2017) and **End-to-End Incremental Learning** (Castro et al. 2018).
iCaRL & EndToEnd
Common attributes

- Fixed-size memory with an exemplars selection
- Constraints on the predictions consistency
Fixed-size memory

- Memory size fixed to $K = 2000$ images ("examplars").
- The number of images per class in the memory decreases as the number of tasks grows.
- Model is trained on whole new data + memory data.
Memory selection for iCaRL

Iterative selection where an image is selected if its mean with all the already selected examplars is closest to class mean:

$$\mu \leftarrow \frac{1}{n} \sum_{x \in X} \varphi(x) \quad \text{// current class mean}$$

$$p_k \leftarrow \arg\min_{x \in X} \left\| \mu - \frac{1}{k} [\varphi(x) + \sum_{j=1}^{k-1} \varphi(p_j)] \right\|$$

Algorithm 4 iCaRL CONSTRUCT_EXEMPLAR_SET

**Algorithm 4 iCaRL CONSTRUCT_EXEMPLAR_SET**

**input** image set $X = \{x_1, \ldots, x_n\}$ of class $y$

**input** $m$ target number of examplars

**require** current feature function $\varphi : \mathcal{X} \rightarrow \mathbb{R}^d$

$$\mu \leftarrow \frac{1}{n} \sum_{x \in X} \varphi(x) \quad \text{// current class mean}$$

**for** $k = 1, \ldots, m$ **do**

$$p_k \leftarrow \arg\min_{x \in X} \left\| \mu - \frac{1}{k} [\varphi(x) + \sum_{j=1}^{k-1} \varphi(p_j)] \right\|$$

**end for**

$P \leftarrow (p_1, \ldots, p_m)$

**output** exemplar set $P$

Figure 4: iCaRL’s memory selection

(Javed and Shafait 2018) claims that this selection method is as good as a random selection.
Memory selection for EndToEnd

The closest images to their class mean are selected:

$$\text{minimize}_x \| \mu - x \|^2$$

Authors note that their selection is only a minor improvement over a random selection (63.6% vs 63.1%).
iCaRL’s last activation is a multi-sigmoid.

It has:

- One classification loss which is a binary cross-entropy with the new targets.
- One distillation loss which is a binary cross-entropy between the current and previous model old targets predictions.

Both are applied on new data and memory data.
Predictions consistency for iCaRL

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

// store network outputs with pre-update parameters:
for \( y = 1, \ldots, s - 1 \) do
\( q^y_i \leftarrow g_y(x_i) \) for all \((x_i, \cdot) \in \mathcal{D}\)
end for

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

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

that consists of classification and distillation terms.

Figure 5: iCaRL’s losses: classification & distillation
Predictions consistency for EndToEnd

Figure 6: EndToEnd’s losses: classification & distillations
Predictions consistency for EndToEnd

\[ L(\theta^t) = \sum_{i=1}^{t-1} L_{\text{distill}}(\theta^t) \]

Classification is a softmax + cross-entropy applied on task's data & memory data for all the targets.

\[ L_{\text{cls}} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} p_{ij} \log(q_{ij}) \]

Distillation is like classification but both the old and new predictions targets are smoothed as in LwF.

\[ \tilde{y}_i = \frac{y_i}{\sum_{j=1}^{C} y_j} \]
Training scheduling for iCaRL

Scheduling

- Train for 70 epochs.
- Initial learning rate of 2.0 decayed throughout the training.
Training scheduling for EndToEnd

Scheduling

1. Train for 40 epochs with a low learning rate that is decayed.
2. Reduce the new dataset using their examplar selection to balance the classes.
3. Add a distillation loss to the new classes.
4. Fine-tuning for 30 epochs with a very low learning rate.
Augmentations & regularizations

iCaRL

- **Augmentation:** Horizontal flip
- **Regularization:** L2 Weight decay

End-to-End Incremental Learning

- **Augmentation:** Horizontal flip + random cropping + brightness + contrast.
- **Regularization:** L2 weight decay, gradient noise, and gradient L2 regularization.
Inference

iCaRL
- For each class, computes the mean of examplars.
- Uses a nearest-neighbours classifier with all the examplars means

End-to-End Incremental Learning
- Classify with its fully-connected weights + a softmax/argmax
References
References I


Yoon, Jaehong et al. (2018). Lifelong Learning with Dynamically Expandable Networks. (Cit. on p. 22).