
[course site]

#DLUPC

Day 3 Lecture 2

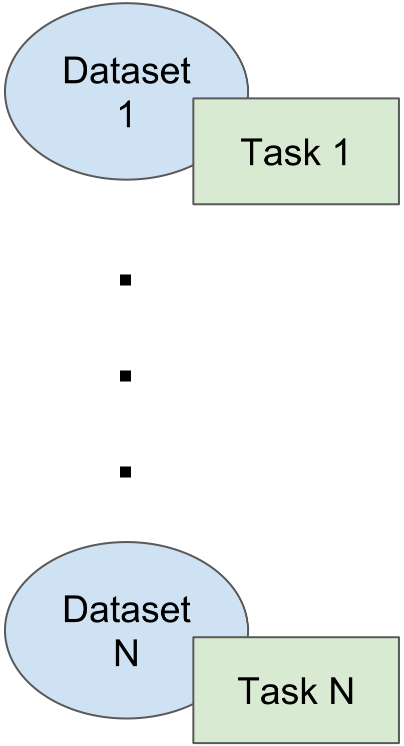
# Life-long/incremental Learning

Ramon Morros
ramon.morros@upc.edu

Associate Professor
Universitat Politecnica de Catalunya
Technical University of Catalonia

# ‘Classical’ approach to ML

- Isolated, single task learning:
  - Well defined tasks.
  - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks
- Data given prior to training
  - Model selection & meta-parameter optimization based on full data set
  - Large number of training data needed
- Batch mode
  - Examples are used at the same time, irrespective of their (temporal) order
- Assumption that data and its underlying structure is static
  - Restricted environment



# Challenges

- Data not available priorly, but exemples arrive over time

- Memory resources may be limited
  - LML has to rely on a compact/implicit representation of the already observed signals
  - NN models provide a good implicit representation!

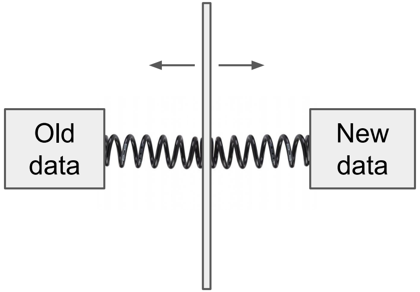
- Adaptive model complexity
  - Impossible to determine model complexity in advance
  - Complexity may be bounded by available resources ➔ intelligent reallocation
  - Meta-parameters such as learning rate or regularization strength can not be determined prior to training ➔ They turn into model parameters!

# Challenges

Source:
https://www.youtube.com/watch?v=HMaWYBlo2Vc

- Concept drift: Changes in data distribution occurs with time
  - For instance, model evolution, changes in appearance, aging, etc.

- Stability -plasticity dilemma: When and how to adapt to the current model
  - Quick update enables rapid adaptation, but old information is forgotten
  - Slower adaptation allows to retain old information but the reactivity of the system is decreased
  - Failure to deal with this dilemma may lead to **catastrophic forgetting**



# Lifelong Machine Learning (LML)

[Silver2013, Gepperth2016, Chen2016b]

## Learn, retain, use knowledge over an extended period of time

- Data streams, constantly arriving, not static ➜ Incremental learning
- Multiple tasks with multiple learning/mining algorithms
- Retain/accumulate learned knowledge in the past & use it to help future learning
    - Use past knowledge for inductive transfer when learning new tasks
- Mimics human way of learning

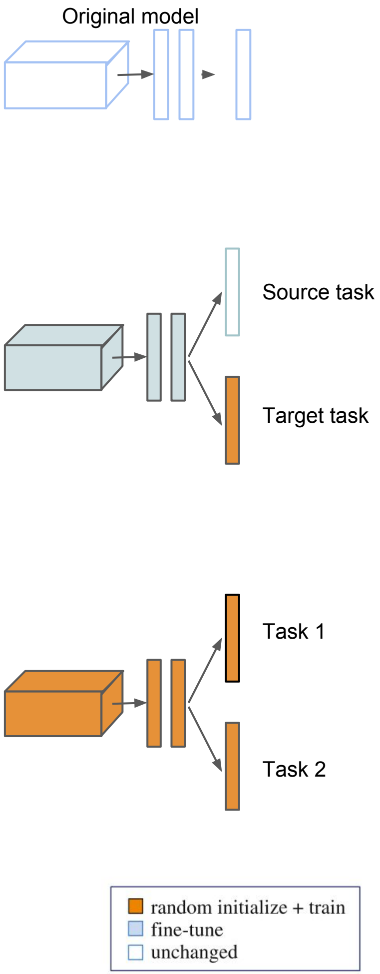
# Related learning approaches

## Transfer learning (finetuning):

- Data in the source domain help learning the target domain
- Less data are needed in the target domain

## Multi-task learning:

- Co-learn multiple, related tasks simultaneously
- All tasks have labeled data and are treated equally
- Goal: optimize learning/performance across all tasks through shared knowledge



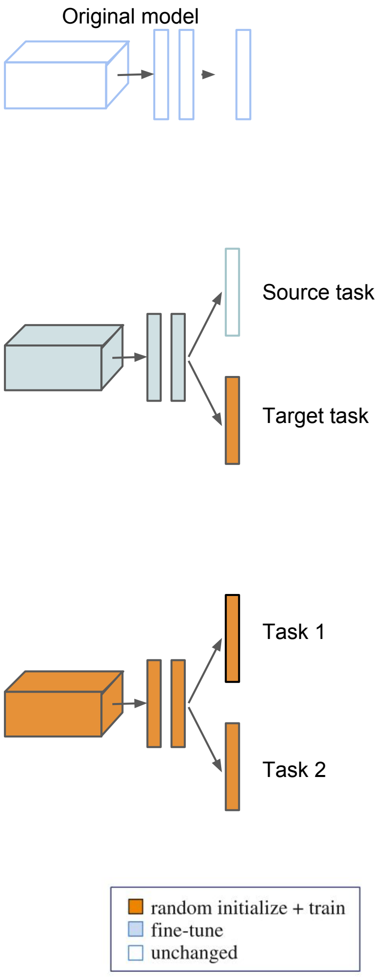
# Related learning approaches

**Transfer learning (finetuning):**

- Unidirectional: source ➔ target
- Not continuous
- No retention/accumulation of knowledge
- Tasks must be similar

**Multi-task learning:**

- Simultaneous learning
- All tasks data are needed for training



# LWF: Learning without Forgetting [Li2016]

**Goal:**

Add **new prediction tasks** based on adapting shared parameters **without access to training data for previously learned tasks**

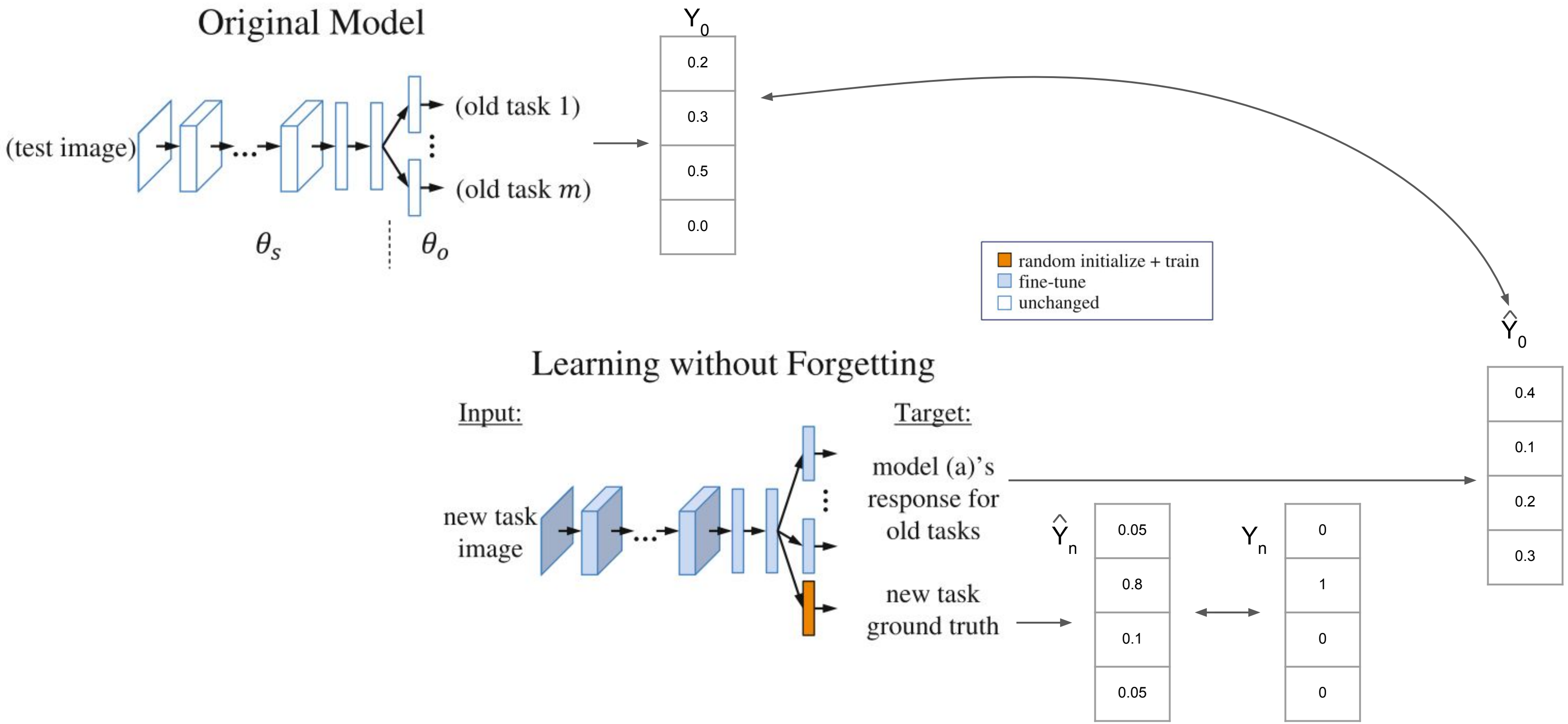
**Solution:**

Using only examples for the new task, optimize for :

- High accuracy on the new task
- Preservation of responses on existing tasks from the original network (distillation, Hinton2015)
- Storage does not grow with time. Old samples are not kept

Preserves performance on old task
(even if images in new task provide a poor sampling of old task)

# LWF: Learning without Forgetting [Li2016]



# LWF: Learning without Forgetting [Li2016]

<u>LearningWithoutForgetting:</u>

<u>Start with:</u>

$\theta_s$: shared parameters

$\theta_o$: task specific parameters for each old task

$X_n$, $Y_n$: training data and ground truth on the new task

<u>Initialize:</u>

$Y_o \leftarrow \text{CNN}(X_n, \theta_s, \theta_o)$ *// compute output of old tasks for new data*

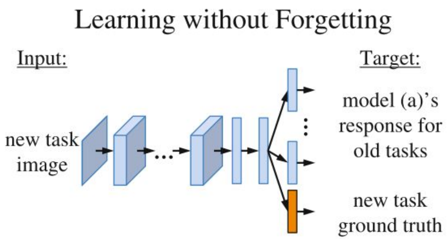
$\theta_n \leftarrow \text{RandInit}(|\theta_n|)$ *// randomly initialize new parameters*

<u>Train:</u>

Define $\hat{Y}_o \equiv \text{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_o)$ *// old task output*

Define $\hat{Y}_n \equiv \text{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_n)$ *// new task output*

$$\theta_s^*,\ \theta_o^*,\ \theta_n^* \leftarrow \operatorname*{argmin}_{\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n} \left( \mathcal{L}_{old}(Y_o, \hat{Y}_o) + \mathcal{L}_{new}(Y_n, \hat{Y}_n) + \mathcal{R}(\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n) \right)$$




Multinomial logistic loss

$$\mathcal{L}_{new}(\mathbf{y}_n, \hat{\mathbf{y}}_n) = -\mathbf{y}_n \cdot \log \hat{\mathbf{y}}_n$$

Weight decay of 0.0005

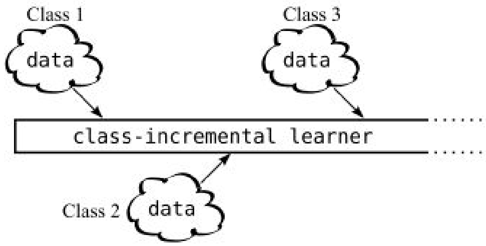
$$\mathcal{L}_{old}(\mathbf{y}_o, \hat{\mathbf{y}}_o) = -H(\mathbf{y}'_o, \hat{\mathbf{y}}'_o) = -\sum_{i=1}^{l} y_o'^{(i)} \log \hat{y}_o'^{(i)} \qquad y_o'^{(i)} = \frac{(y_o^{(i)})^{1/T}}{\sum_j (y_o^{(j)})^{1/T}}, \quad \hat{y}_o'^{(i)} = \frac{(\hat{y}_o^{(i)})^{1/T}}{\sum_j (\hat{y}_o^{(j)})^{1/T}}.$$

Distillation loss

# iCaRL

**Goal:**

Add new classes based on adapting shared parameters **with restricted access** to training data for previously learned classes.




**Solution:**

- A subset of training samples (exemplar set) from previous classes is stored.
- Combination of classification loss for new samples and distillation loss for old samples.
- The size of the exemplar set is kept constant. As new classes arrive, some examples from old classes are removed.

# iCaRL: Incremental Classifier and Representation learning

**Algorithm 2** iCaRL INCREMENTALTRAIN

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

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

**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:

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

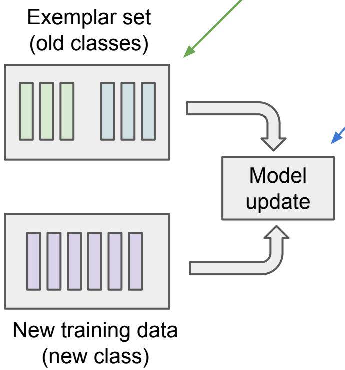
// store network outputs with pre-update parameters:
**for** $y = 1, \dots, s-1$ **do**
  $q_i^y \leftarrow g_y(x_i)$ for all $(x_i, \cdot) \in \mathcal{D}$
**end for**

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

$$\ell(\Theta) = -\sum_{(x_i, y_i) \in \mathcal{D}} \Big[ \sum_{y=s}^{t} \delta_{y=y_i} \log(g_y(x_i)) \quad \text{// classification loss}$$

$$+ \sum_{y=1}^{s-1} q_i^y \log(g_y(x_i)) \Big] \quad \text{// distillation loss}$$

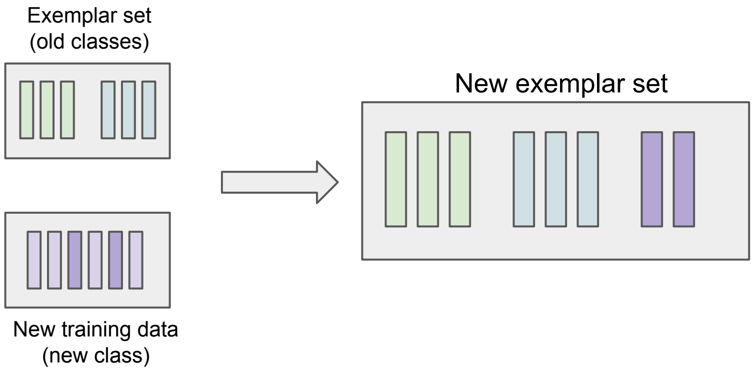
[Hinton2015]



# **iCaRL:** Incremental Classifier and Representation learning

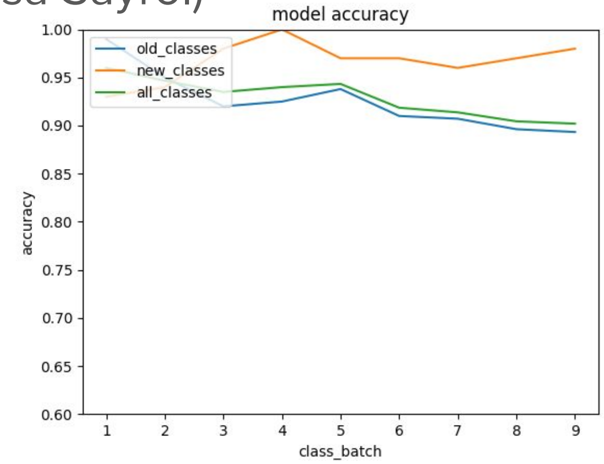
**Algorithm 2** iCaRL INCREMENTALTRAIN

**input** $X^s, \ldots, X^t$ // training examples in per-class sets
**input** $K$ // memory size
**require** $\Theta$ // 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**
    $P_y \leftarrow$ REDUCEEXEMPLARSET$(P_y, m)$
  **end for**
  **for** $y = s, \ldots, t$ **do**
    $P_y \leftarrow$ CONSTRUCTEXEMPLARSET$(X_y, m, \Theta)$
  **end for**
  $\mathcal{P} \leftarrow (P_1, \ldots, P_t)$ // new exemplar sets

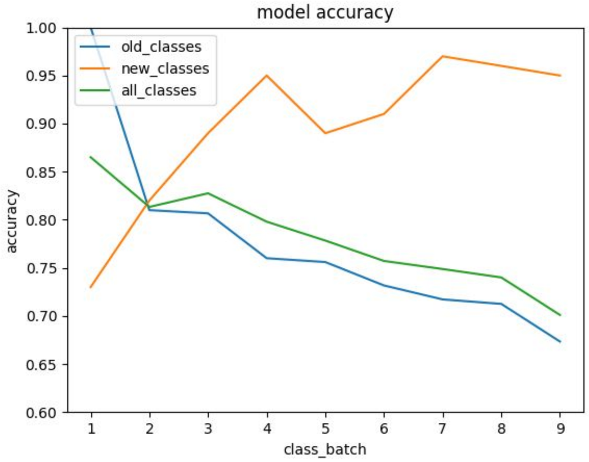


# Results on face recognition

- Preliminary results on face recognition from Eric Presas TFG (co-directed with Elisa Sayrol)




iCaRL




LWF

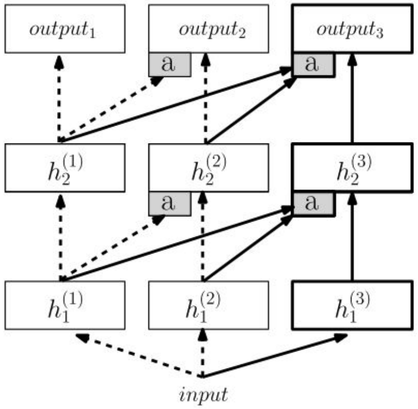
# Progressive Neural Networks

**Goal:**

Learn a series of tasks in sequence, using knowledge from previous tasks to improve convergence speed

**Solution:**

- Instantiate a new NN for each task being solved, with lateral connections to features of previously learned columns
- Previous tasks training data is not stored. Implicit representation as NN weights.
- Complexity of the model grows with each task
- Task labels needed at test time

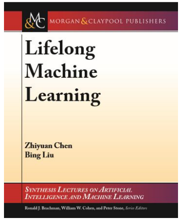



$$h_i^{(k)} = f\left(W_i^{(k)} h_{i-1}^{(k)} + \sum_{j<k} U_i^{(k:j)} h_{i-1}^{(j)}\right)$$

# Summary

| | Task labels needed? | Old training data needed? | Constant data size | Increase in model complexity | Type | Mechanism |
|---|---|---|---|---|---|---|
| **iCaRL** | No | Yes | Yes | Very small<br>(neurons in output layer) | Class incremental | Distillation |
| **LFW** | Yes | No | Yes | Small (output layer) | Task incremental | Distillation |
| **PNN** | Yes | No | Yes | Linear (new network) | Task incremental | New network with lateral connections to old ones |

# References




[Chen2016a] Z. Chen, Google, B. Liu, “Lifelong Machine Learning for Natural Language Processing”, *EMNLP-2016 Tutorial*, 2016

[Chen2016b] Z. Chen and B. Liu, “Lifelong Machine Learning”, Morgan & Claypool Publishers, November 2016.

[Gepperth2016] A. Gepperth, B. Hammer, “Incremental learning algorithms and applications”, ESANN 2016

[Hinton2015] Hinton, G., Vinyals, O., & Dean, J. “Distilling the Knowledge in a Neural Network”. *NIPS 2014 DL Workshop*, 1–9.

[Li2016] Li, Z., & Hoiem, D. “Learning without forgetting”. In vol. 9908 LNCS, 2016.

[Rebuffi2016] Rebuffi, S.-A., Kolesnikov, A., & Lampert, C. H. “iCaRL: Incremental Classifier and Representation Learning”. 2016 *arXiv:1611.07725*

[Rusu2016] Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., … Hadsell, R. “*Progressive Neural Networks”*. 2016 *CoRR. arXiv:1606.04671*.

[Silver2013] D.L.Silver, et al, “Lifelong machine learning systems: Beyond learning algorithms”, 2013 AAAI Spring Symposium

# Questions?