#DLUPC

Day 6 Lecture 1

Life-long/incremental Learning

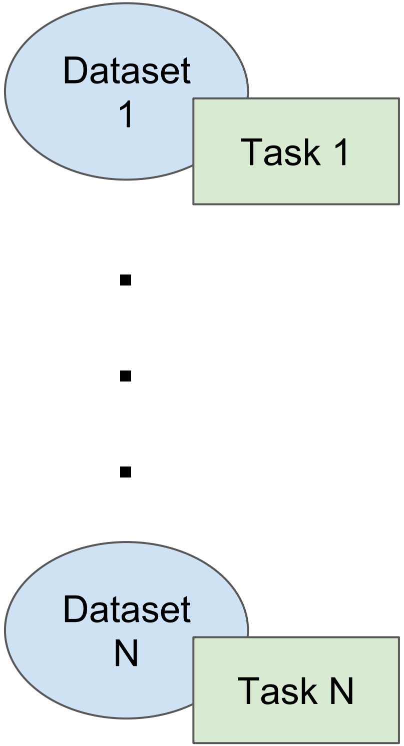
Ramon Morros
ramon.morros@upc.edu

Associate Professor
Universitat Politecnica de Catalunya
Technical University of Catalonia

[course site]

‘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!

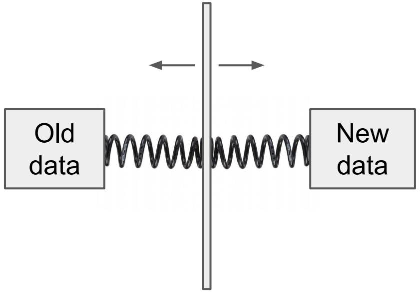
- Adaptive model complexity
  - Impossible to determine model complexity in advance
  - Complexity may be bounded by available resources ➔ intelligent reallocation
  - Meta-parameters such as learning rate or regularization strength can not be determined prior to training ➔ They turn into model parameters!

Challenges

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

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

- 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)

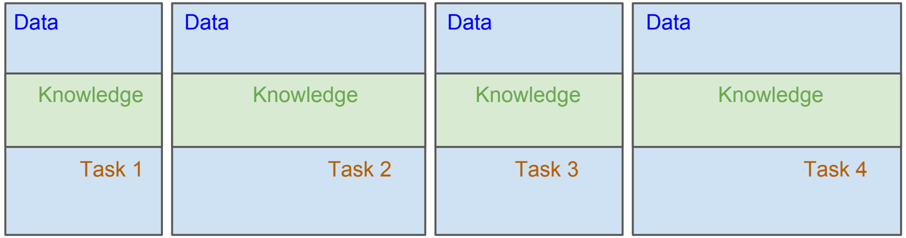
[Silver2013, Gepperth2016, Chen2016b]

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

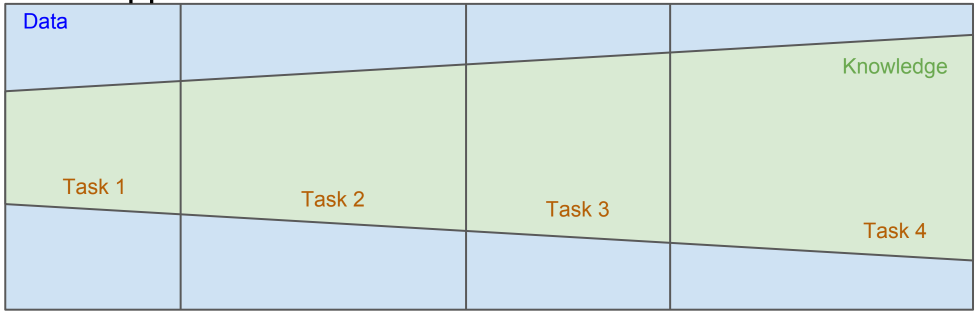
- Data streams, constantly arriving, not static ➜ Incremental learning
- Multiple tasks with multiple learning/mining algorithms
- Retain/accumulate learned knowledge in the past & use it to help future learning
  - Use past knowledge for inductive transfer when learning new tasks
- Mimics human way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

Lifelong Machine Learning (LML)

‘Classical’ approach



LML approach



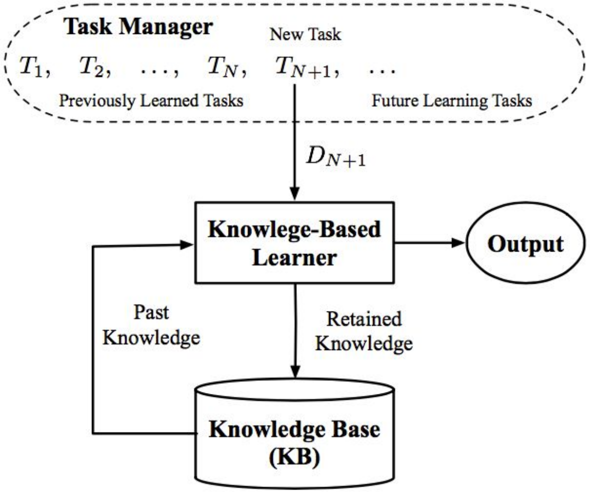


Image from [Chen2016a]

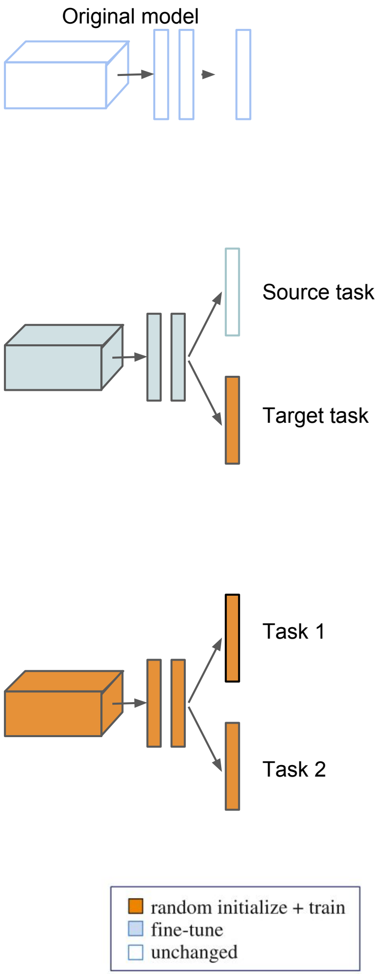
Related learning approaches

Transfer learning (finetuning):

- Data in the source domain helps learning the target domain
- Less data is needed in the target domain
- Tasks must be similar

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



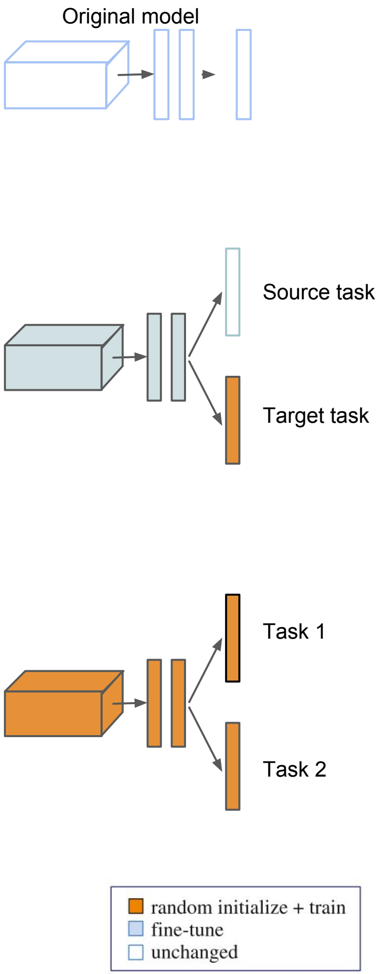
Related learning approaches

Transfer learning (finetuning):

- Unidirectional: source ➔ target
- Not continuous
- No retention/accumulation of knowledge

Multi-task learning:

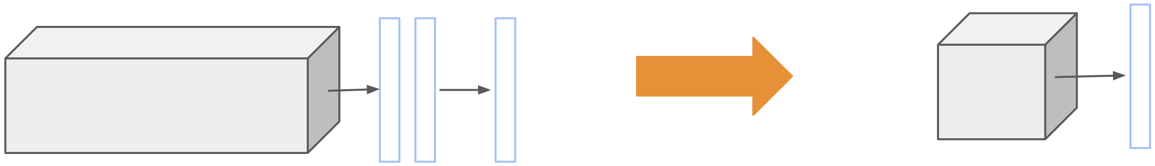
- Simultaneous learning
- All tasks data is needed for training



LML Methods

Distillation

Original application was to transfer the knowledge from a large, easy to train model into a smaller/faster model more suitable for deployment



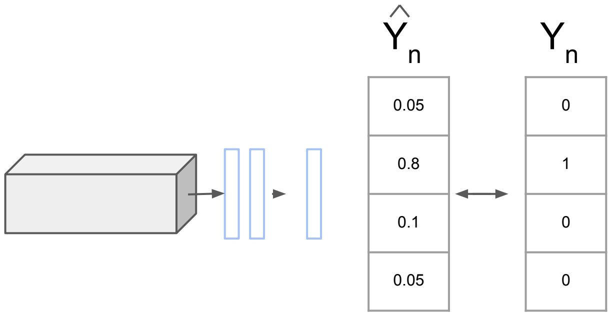
Bucilua[1] demonstrated that this can be done reliably when transferring from a large ensemble of models to a single small model

[1]C.Bucilua, R. Caruana, and A. Niculescu-Mizil. "Model compression". In ACMSIG KDD '06, 2006

Distillation

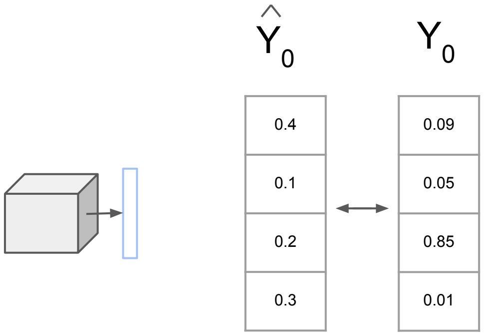
Idea: use the class probabilities produced by the large model as “soft targets” for training the small model

- The ratios of probabilities in the soft targets provide information about the learned function
- These ratios carry information about the structure of the data
- Train by replacing the hard labels with the **softmax activations from the original large model**



Multinomial logistic loss

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



Distillation loss

$$\mathcal{L}(\mathbf{y}_o, \hat{\mathbf{y}}_o) = -H(\mathbf{y}'_o, \hat{\mathbf{y}}'_o) = -\sum_{i=1}^{l} y'^{(i)}_o \log \hat{y}'^{(i)}_o$$

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

Distillation

- To increase the influence of non-target class probabilities in the cross entropy, the temperature of the final softmax is raised to "soften" the final probability distribution over classes
- Transfer can be obtained by using the same large model training set or a separate training set
- If the ground-truth labels of the transfer set are known, standard loss and distillation loss can be combined

$$y'^{(i)}_o = \frac{(y_o^{(i)})^{1/T}}{\sum_j (y_o^{(j)})^{1/T}}, \qquad \hat{y}'^{(i)}_o = \frac{(\hat{y}_o^{(i)})^{1/T}}{\sum_j (\hat{y}_o^{(j)})^{1/T}}.$$

T=1	T>1
0.09	0.15
0.05	0.10
0.85	0.70
0.01	0.05

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

LWF: Learning without Forgetting [Li2016]

Goal:

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

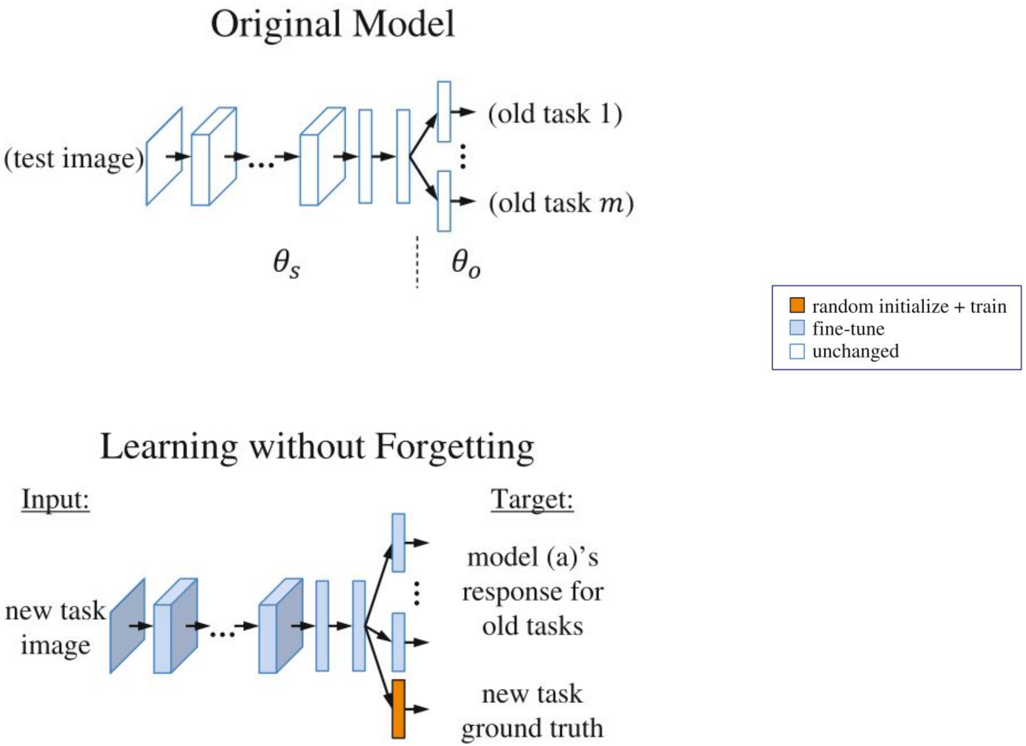
Solution:

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/complexity 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]

LEARNINGWITHOUTFORGETTING:

Start with:

θ_s: shared parameters

θ_o: task specific parameters for each old task

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

Initialize:

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

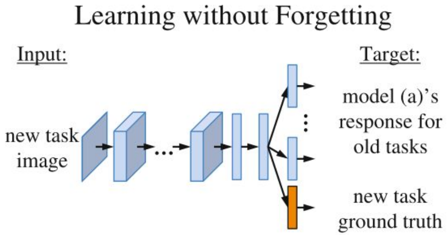
$\theta_n \leftarrow \textsc{RandInit}(|\theta_n|)$ // *randomly initialize new parameters*

Train:

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

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

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



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

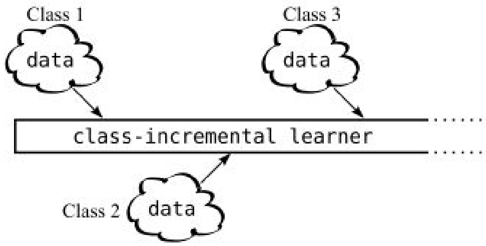
$$\mathcal{L}_{old}(\mathbf{y}_o, \hat{\mathbf{y}}_o) = -H(\mathbf{y}'_o, \hat{\mathbf{y}}'_o) = -\sum_{i=1}^{l} y'^{(i)}_o \log \hat{y}'^{(i)}_o \qquad y'^{(i)}_o = \frac{(y^{(i)}_o)^{1/T}}{\sum_j (y^{(j)}_o)^{1/T}}, \quad \hat{y}'^{(i)}_o = \frac{(\hat{y}^{(i)}_o)^{1/T}}{\sum_j (\hat{y}^{(j)}_o)^{1/T}}.$$

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 $X^s, \ldots, X^t$ // 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**
  $P_y \leftarrow$ REDUCEEXEMPLARSET(P_y, m)
end for
for $y = s, \ldots, t$ **do**
  $P_y \leftarrow$ CONSTRUCTEXEMPLARSET(X_y, m, Θ)
end for
$\mathcal{P} \leftarrow (P_1, \ldots, P_t)$ // new exemplar sets

Algorithm 3 iCaRL UPDATEREPRESENTATION

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

// form combined training set:

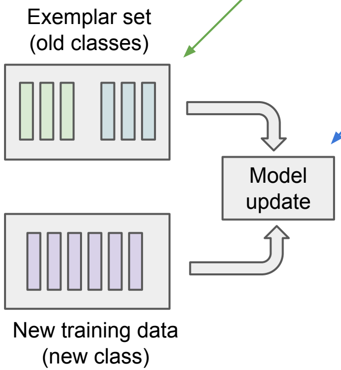
$$\mathcal{D} \leftarrow \bigcup_{y=s,\ldots,t} \{(x, y) : x \in X^y\} \cup \bigcup_{y=1,\ldots,s-1} \{(x, y) : x \in P^y\}$$

// 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

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

[Hinton2015]

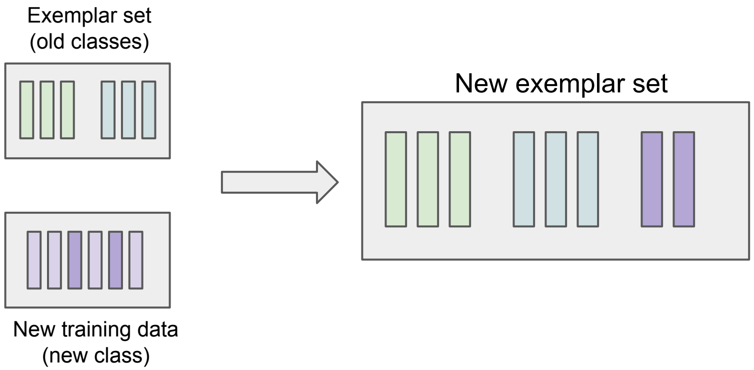


iCaRL: Incremental Classifier and Representation learning

Algorithm 2 iCaRL INCREMENTALTRAIN

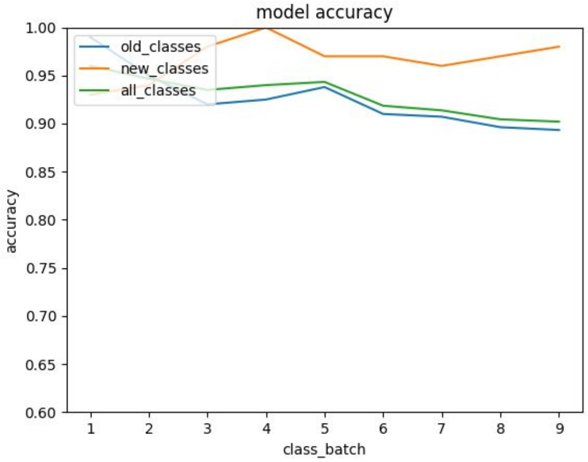
input $X^s, \dots, X^t$ // training examples in per-class sets
input K // memory size
require Θ // current model parameters
require $\mathcal{P} = (P_1, \dots, P_{s-1})$ // current exemplar sets

$\Theta \leftarrow \text{UPDATEREPRESENTATION}(X^s, \dots, X^t; \mathcal{P}, \Theta)$
$m \leftarrow K/t$ // 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

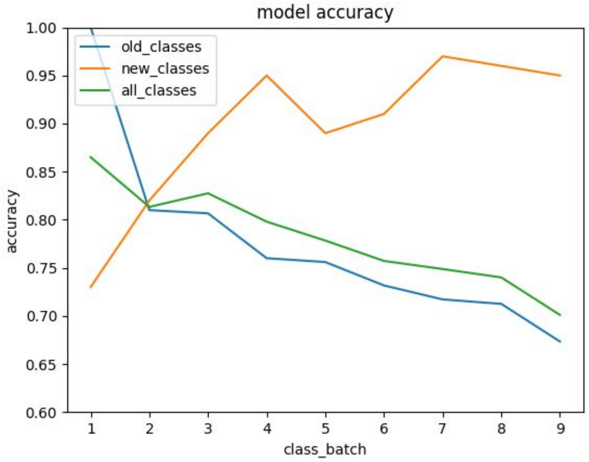


Results on face recognition

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



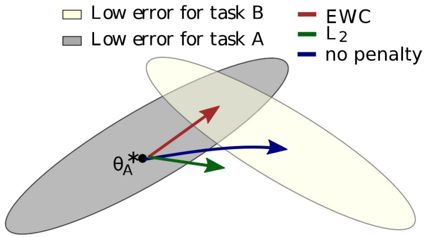
iCaRL



LWF

Elastic Weight Consolidation (EWC)

- Evidence suggests that the mammalian brain may avoid catastrophic forgetting by protecting previously acquired knowledge in neocortical circuits
- Knowledge is durably encoded by rendering a proportion of synapses less plastic (stable over long timescales)
- EWC algorithm slows down learning on certain weights based on how important they are to previously seen tasks
- While learning task B, EWC therefore protects the performance in task A by constraining the parameters to stay in a region of low error for task A centered around θ*
- Constraint implemented as a quadratic penalty. Can be imagined as a spring anchoring the parameters to the previous solution (elastic).
- The stiffness of this spring should not be the same for all parameters; rather, it should be greater for parameters that most affect performance in task A



$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2$$

F: Fisher information matrix
(https://en.wikipedia.org/wiki/Fisher_information#Matrix_form)

Kirkpatrick *et al*. Overcoming catastrophic forgetting in neural networks. *Proc. of the National Academy of Sciences*, *114*(13), 2017

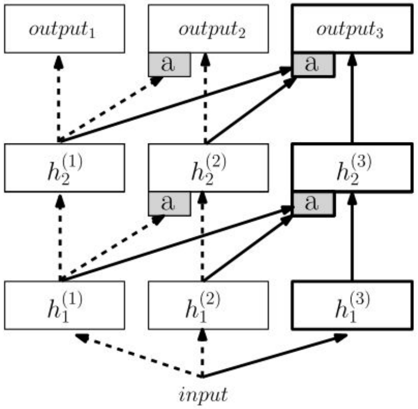
Progressive Neural Networks

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



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

Rusu *et al* (2016). *Progressive Neural Networks*. *CoRR*. *arXiv:1606.04671*. Retrieved from http://arxiv.org/abs/1606.04671

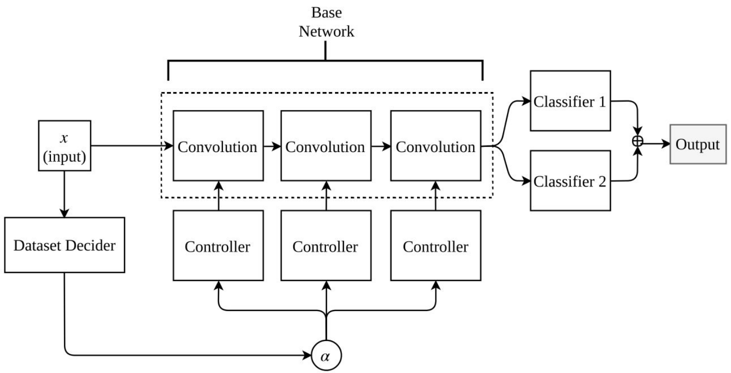
Deep adaptation (I)

In Progressive NN, the number of parameters is duplicated for each task

In iCaRL, LWF and EWC, the performance in older tasks can decrease because weights are shared between tasks

Idea: Augmenting a network learned for one task with controller modules which utilize already learned representations for another

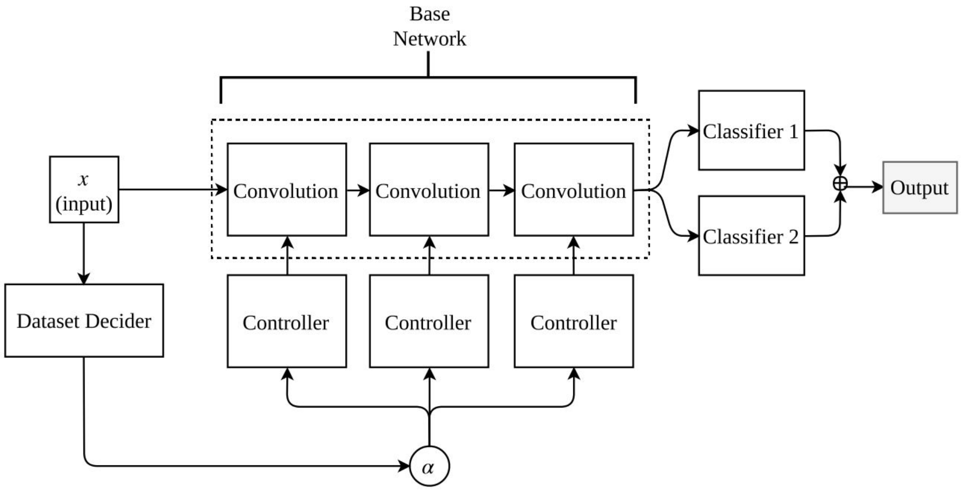
- Parameters of the controller modules are optimized to minimize a loss on a new task.
- The training data for the original task is not required for successive tasks.
- The network's output on the original task data stays exactly as it was
- Any number of controller modules may be added so that a single network can simultaneously encode multiple distinct tasks



Rosenfeld, A., & Tsotsos, J. K. (2017). Incremental Learning Through Deep Adaptation. *arXiv*. Retrieved from http://arxiv.org/abs/1705.04228

Deep adaptation (II)

- Each controller module uses the existing weights of the corresponding layer of N to create new convolutional filters adapted to the new task T2
- Throughout training & testing, the weights of the base network are fixed and only used as basis functions.

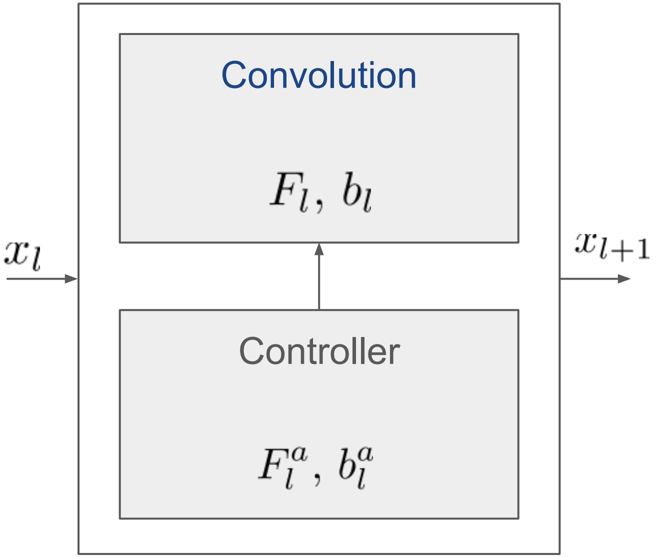


Rosenfeld, A., & Tsotsos, J. K. (2017). Incremental Learning Through Deep Adaptation. *arXiv*. Retrieved from http://arxiv.org/abs/1705.04228

Deep adaptation (III)

- Each controller module uses the existing weights of the corresponding layer of N to create new convolutional filters adapted to the new task T2
- Throughout training & testing, the weights of the base network are fixed and only used as basis functions.

C_o is the number of output features, C_i the number of inputs, k size of the conv. filters



$F_l \in \mathcal{R}^{C_o \times C_i \times k \times k}, \ b_l \in \mathcal{R}^{C_o}$ parameters of conv. layer l

$F_l^a = W_l \otimes F_l, \quad b_l^a$ parameters of the controller

$W_l \in \mathcal{R}^{C_o \times C_o}$

$a \otimes b$: flatten b $\rightarrow$ matrix multiply by a $\rightarrow$ unflatten

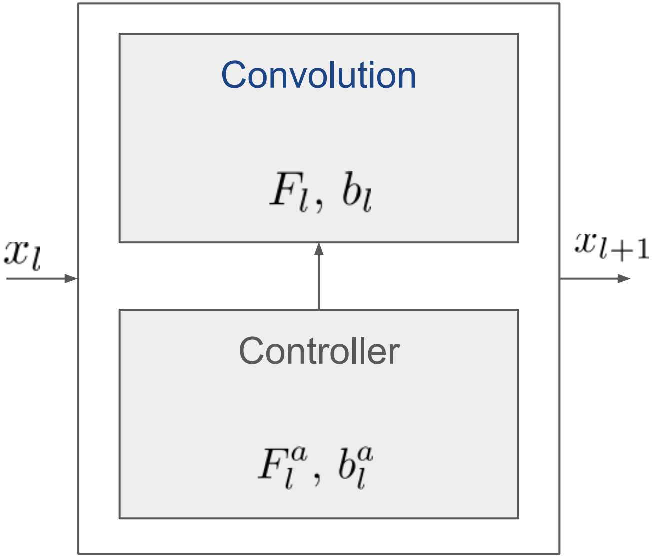
$$x_{l+1} = [\alpha F_l^a) + (1-\alpha)F_l] * x_l + \alpha b_l^a + (1-\alpha)b_l$$

$$\alpha \in \{0, 1\}$$

Rosenfeld, A., & Tsotsos, J. K. (2017). Incremental Learning Through Deep Adaptation. *arXiv*. Retrieved from http://arxiv.org/abs/1705.04228

Deep adaptation (IV)

- Fully connected layers are not reused
- The weights of the controller modules are learned via back-propagation given the loss function
- The number of new of parameters added for each task is moderate



Ratio of new parameters to old ones (per layer):

$$\frac{C_o+1}{C_i\times k\times k+1}\approx\frac{C_o}{C_i\times k\times k}$$

$C_o = C_i = 256$, $k=5 \rightarrow r = 0.04$

C_o is the number of output features, C_i the number of inputs, k size of the conv. filters

For a complete network, typically : 20 ~ 30%

Rosenfeld, A., & Tsotsos, J. K. (2017). Incremental Learning Through Deep Adaptation. *arXiv*. Retrieved from http://arxiv.org/abs/1705.04228

Summary

	Task labels needed?	Old training data needed?	Constant data size	Constant model complexity	Type	Mechanism
iCaRL	No	Yes	Yes	Yes	Class incremental	Distillation
LFW	Yes	No	Yes	Yes	Task incremental	Distillation
PNN	Yes	No	Yes	No (doubling per each new task)	Task incremental	New network with lateral connections to old ones
EWC	No	No	Yes	Yes	Task incremental	Preserve important weights
DA	Yes ()	No	Yes	No (20~30% increment per new task)	Task incremental	Add controller modules

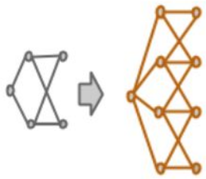
Increasing model capacity (I)

New knowledge acquired (new classes, new domains) over time may saturate network capacity

We can think of a lifelong learning system as experiencing a continually growing training set.

The optimal model complexity changes as training set size changes over time.

- Initially, a small model may be preferred, in order to prevent overfitting and to reduce the computational cost of using the model.
- Later, a large model may be necessary to fully utilize the large dataset.



Increasing model capacity (II)

Some LML methods already add capacity for each task (PNN, DA) but others do not.

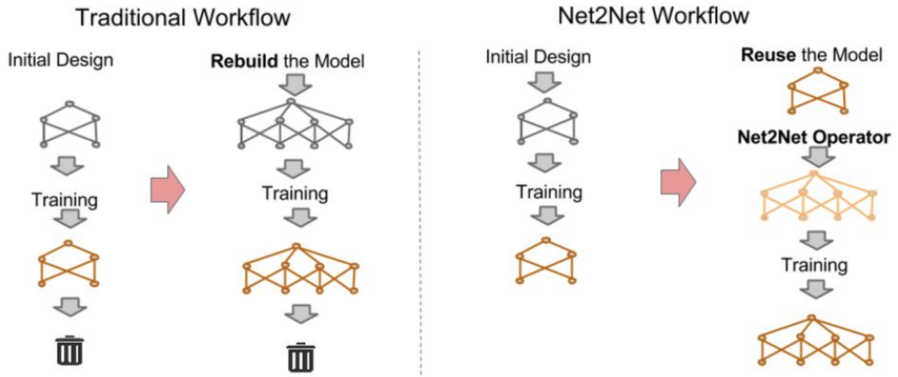
If the capacity of the network has to be incremented we want to avoid retraining the new network from scratch

It is possible to transfer knowledge from a **teacher** network to a ‘bigger’ **student** network in an efficient way

Chen, T., Goodfellow, I., & Shlens, J. (2016). **Net2Net: Accelerating Learning via Knowledge Transfer**. In *ICLR 2016*

Increasing model capacity: Net2Net (I)

- The new, larger network immediately performs as well as the original network, rather than spending time passing through a period of low performance.
- Any change made to the network after initialization is guaranteed to be an improvement, so long as each local step is an improvement.
- It is always "safe" to optimize all parameters in the network.



Chen, T., Goodfellow, I., & Shlens, J. (2016). Net2Net: Accelerating Learning via Knowledge Transfer. In *ICLR 2016*

Increasing model capacity: Net2Net (II)

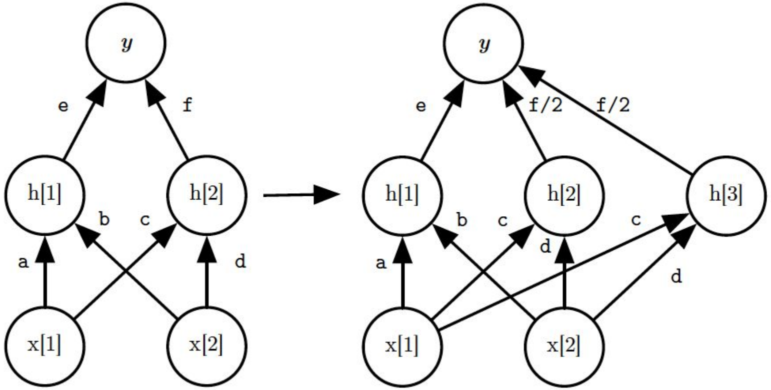
Net2WiderNet:

- Allows a layer to be replaced with a wider layer (a layer that has more units)
- For convolution architectures, this means more convolution channels

(Biases are omitted for simplicity)

$$W^{i+1} \in \mathcal{R}^{n \times p}$$

$$W^{i} \in \mathcal{R}^{m \times n}$$



Teacher Network   Student Network

$$U^{i+1} \in \mathcal{R}^{q \times p}$$

$$U^{i} \in \mathcal{R}^{m \times q}, \quad q > n$$

Chen, T., Goodfellow, I., & Shlens, J. (2016). Net2Net: Accelerating Learning via Knowledge Transfer. In *ICLR 2016*

Increasing model capacity: Net2Net (III)

A random mapping g(·) is used to build U from W:

- The first n columns of $W^{(i)}$ are copied directly into $U^{(i)}$
- Columns n+1 through q of $U^{(i)}$ are created by choosing at random (with replacement) as defined in g.
- For weights in $U^{(i+1)}$, we must account for the replication by dividing the weight by a replication factor, so all the units have the same value as the unit in the original net
- This can be generalized to making multiple layers wider

Algorithm 1: Net2WiderNet

Input: $\{W^{(i)} | i = 1, 2, \cdots n\}$, the weight matrix of teacher net
Use forward inference to generate a consistent random mapping $\{g^{(i)}\}$
for $i \in 1, 2, \cdots n$ **do**
  $c_j \leftarrow 0$ **for** $j \in 1, 2, \cdots q$ **do**
    $c_{g^{(i-1)}(j)} \leftarrow c_{g^{(i-1)}(j)} + 1$
  end
  for $j \in 1, 2, \cdots q$ **do**
    $U^{(i)}_{k,j} \leftarrow \frac{1}{c_j} W^{(i)}_{g^{(i-1)}(k), g^{(i)}(j)}$
  end
end
Output: $\{U^{(i)} | i = 1, 2, \cdots n\}$: the transformed weight matrix for wider net.

$$\forall x, f(x;\theta) = g(x;\theta')$$

$$g : \{1, 2, \cdots, q\} \rightarrow \{1, 2, \cdots, n\} \quad q > n$$

$$g(j) = \begin{cases} j & j \leq n \\ \text{random sample from } \{1, 2, \cdots, n\} & j > n \end{cases}$$

$$U^{(i)}_{k,j} = W^{(i)}_{k,g(j)}, \quad U^{(i+1)}_{j,h} = \frac{1}{|\{x | g(x) = g(j)\}|} W^{(i+1)}_{g(j),h}$$

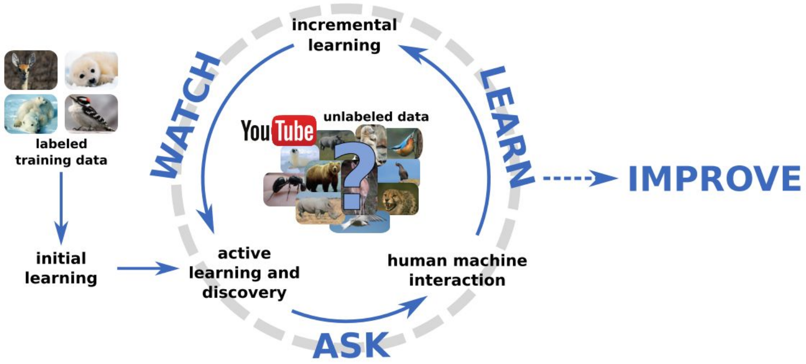
Chen, T., Goodfellow, I., & Shlens, J. (2016). Net2Net: Accelerating Learning via Knowledge Transfer. In *ICLR 2016*

Discovering new classes

Most learning systems follow a closed world assumption (the number of categories is predetermined at training time)

New classes may appear over time. Systems need a way to detect them and to introduce them in the learning process

The method in [Kading2016] inspires in the way humans (children) learn over time



Käding, C., Rodner, E., Freytag, A., & Denzler, J. (2016). Watch, Ask, Learn, and Improve: a lifelong learning cycle for visual recognition. *European Symposium on Artificial NN*.

Discovering new classes

Most learning systems follow a closed world assumption (the number of categories is predetermined at training time)

New classes may appear over time. Systems need a way to detect them and to introduce them in the learning process

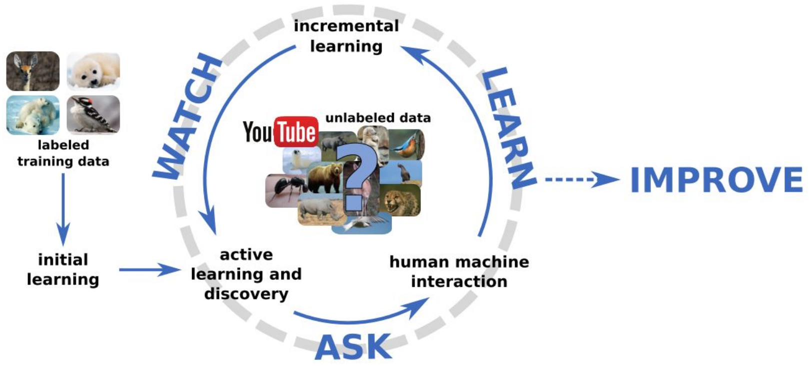
The method in [Kading2016] inspires in the way humans (children) learn over time

Käding, C., Rodner, E., Freytag, A., & Denzler, J. Watch, Ask, Learn, and Improve: a lifelong learning cycle for visual recognition. *European Symposium on Artificial NN*. 2016

WALI (I)

The system incorporates four phases:

- Watch: the system is feed with continuous streams of youtube video
- Ask: The system actively selects few examples for manual annotations
- Learn: Obtained feedback is used to update the current model
- Improve: This never-ending cycle allows to adapt to new scenarios



Käding, C., Rodner, E., Freytag, A., & Denzler, J. Watch, Ask, Learn, and Improve: a lifelong learning cycle for visual recognition. *European Symposium on Artificial NN*. 2016

WALI (II)

Watch

- Continuous stream of unlabeled images
- Obtained by automatically downloading videos from youtube using the official API
- A given youtube category is used (animal documentary)
- Images are sampled every 10^{th} frame to reduce redundancy
- Visual descriptors are extracted using using pre-trained CNN activations (relu7 of AlexNet trained on ImageNet)

ASK

- A key feature is to select images to be labeled by human annotators.
- Images that will lead to an information gain must be selected
- Active learning: unlabeled samples are evaluated whether they likely result in an increase on the classifier performance once labeled and added to the training

Käding, C., Rodner, E., Freytag, A., & Denzler, J. Watch, Ask, Learn, and Improve: a lifelong learning cycle for visual recognition. *European Symposium on Artificial NN*. 2016

WALI (III)

ASK (cont.)

- Query images are selected according to the best vs. second-best strategy as proposed in [Ajay2009]
  - One-vs-all classifier for each class
  - The example with the smallest q(x) score is selected for labeling

$$q(x) = score_{best\_class} - score_{second\_best\_class}$$

- A rejection category is added (not all frames can be associated with a semantic category or maybe some categories are not important).

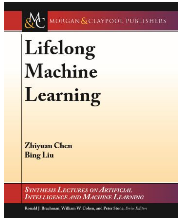
$$q^*(x) = (1 - p(rejection \mid x)) \cdot q(x)$$

Learn

- Use an incremental learning to retrain the classifiers with the new samples

Käding, C., Rodner, E., Freytag, A., & Denzler, J. Watch, Ask, Learn, and Improve: a lifelong learning cycle for visual recognition. *European Symposium on Artificial NN*. 2016

References



[Ajay2009] Joshi, A. J., Porikli, F., & Papanikolopoulos, N.. Multi-class active learning for image classification. In *2009 CVPR*

[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

[ChenT2016] Chen, T., Goodfellow, I., & Shlens, J. (2016). Net2Net: Accelerating Learning via Knowledge Transfer. In *ICLR 2016*

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

[Kading2016] Käding, C., Rodner, E., Freytag, A., & Denzler, J. Watch, Ask, Learn, and Improve: a lifelong learning cycle for visual recognition. *European Symposium on Artificial Neural Networks*. 2016

[Kirkpatrick2017] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., … Hadsell, R. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, *114*(13), 2017.

[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*

[Rosenfeld2017] Rosenfeld, A., & Tsotsos, J. K. (2017). Incremental Learning Through Deep Adaptation. *arXiv*. http://arxiv.org/abs/1705.04228

[Rusu2016] Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., … 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?