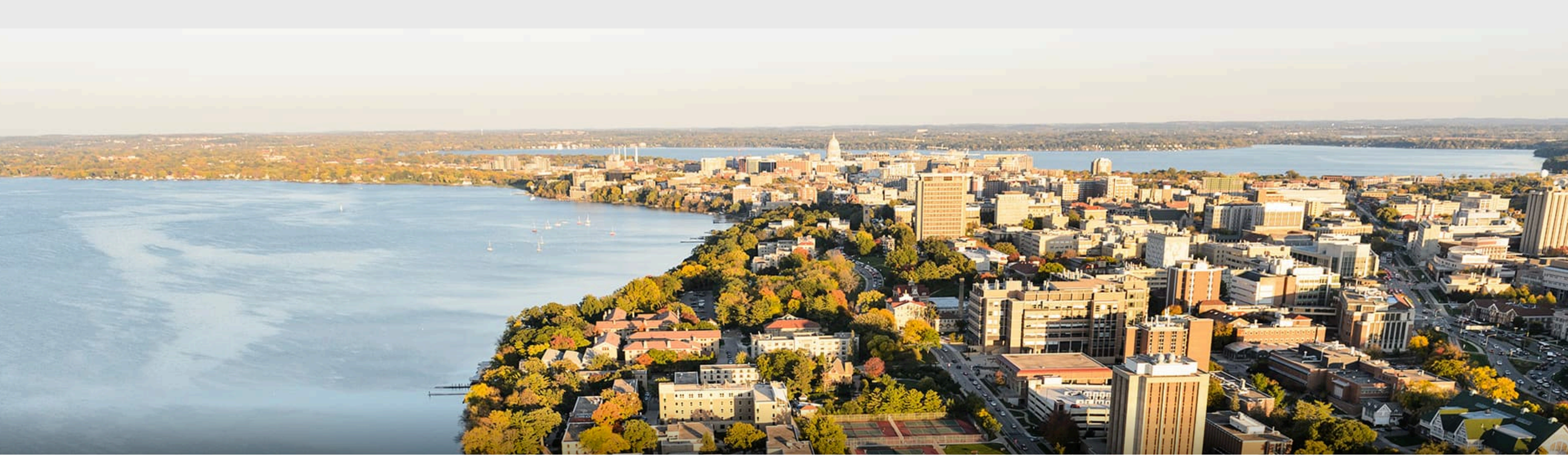


CS839 Special Topics in AI: Deep Learning
Lifelong Machine Learning

Sharon Yixuan Li
University of Wisconsin-Madison

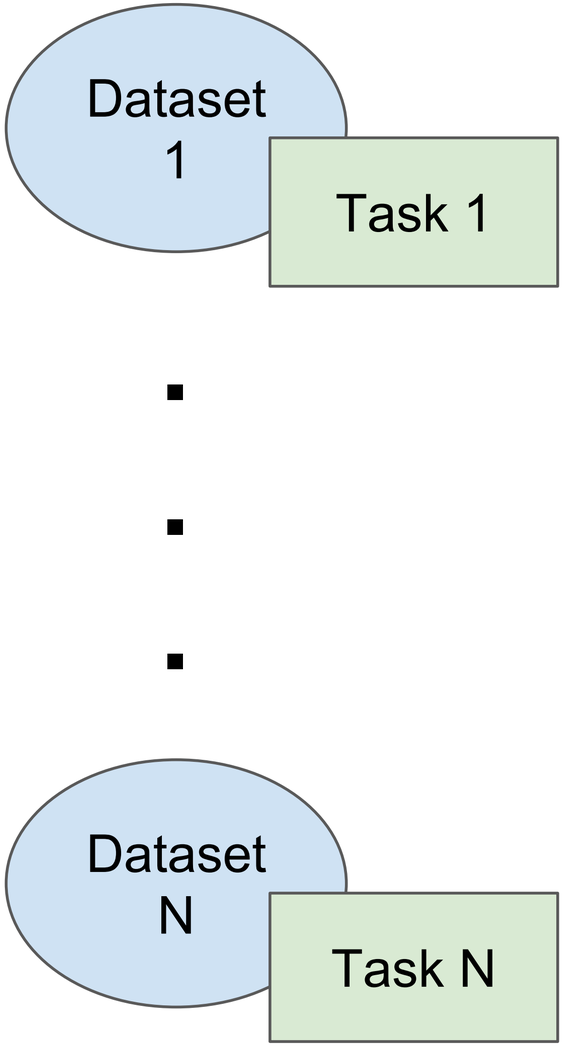
Nov 12, 2020

Overview

- **The need and challenges for Lifelong Machine Learning (LML)**
  - Constant data stream
  - Multiple tasks
  - Knowledge retention
- **LML Methods**
  - Learning without Forgetting (LwF)
  - iCaRL
  - Progressive Neural Networks
- **Growing capacity & discover new classes**

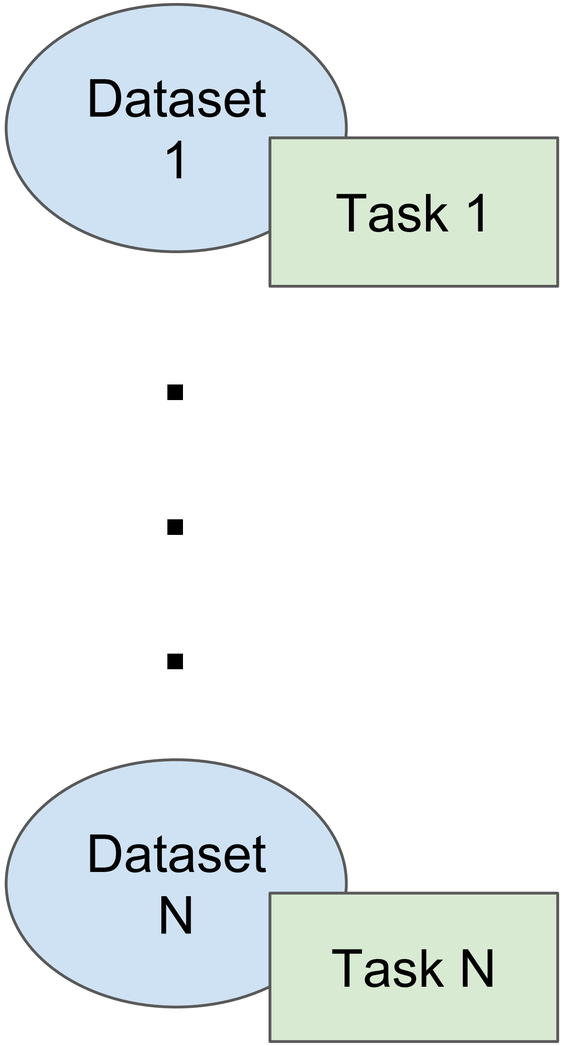
‘Classical’ approach to ML

- 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



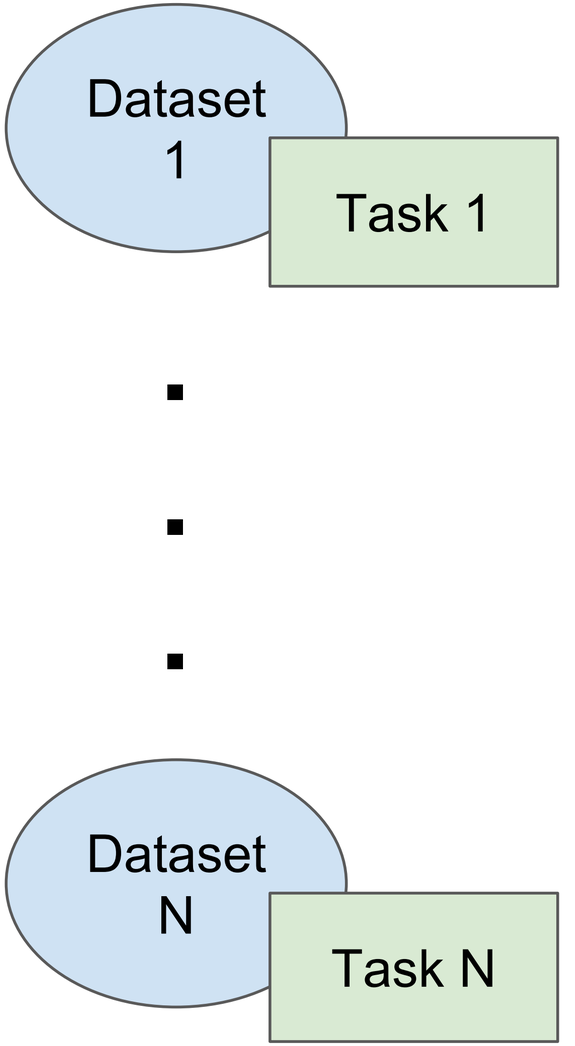
‘Classical’ approach to ML

- 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



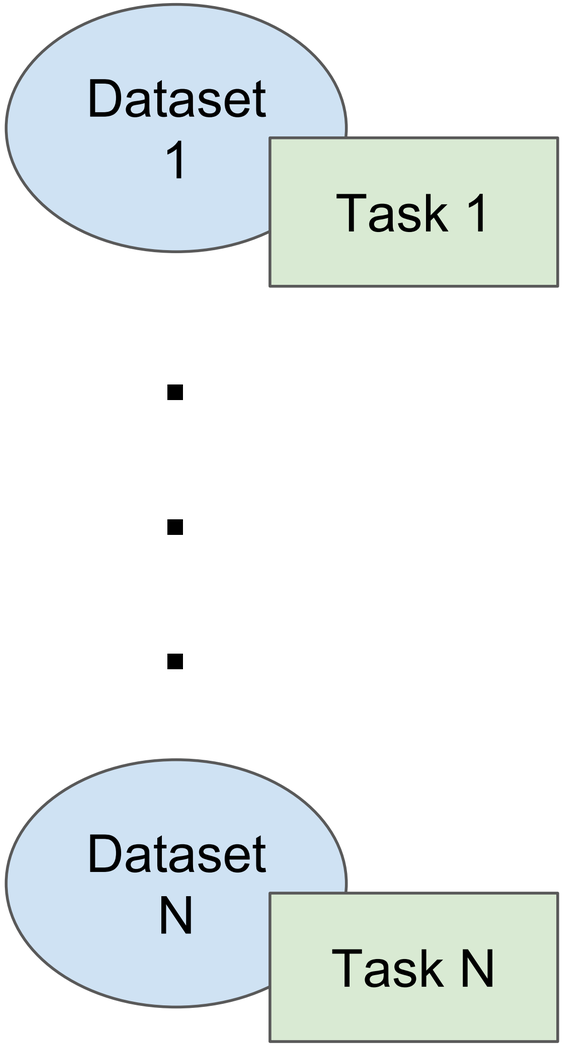
‘Classical’ approach to ML

- 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



‘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

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!

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=HMaWYBlo2

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=HMaWYBlo2

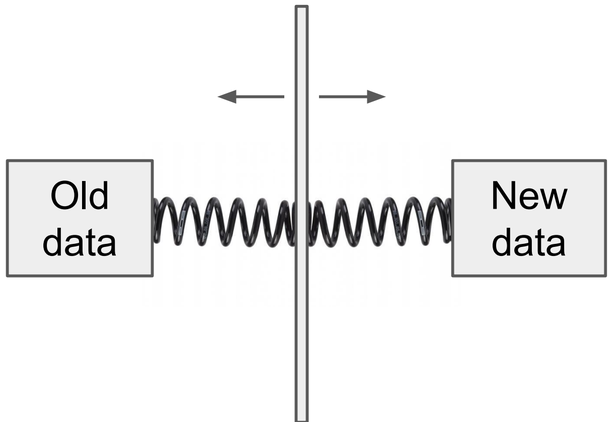
- Stability -plasticity dilemma: When and how to adapt to the current model
  - Quick update enables rapid adaptation, but old information is forgotten

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=HMaWYBlo2

- 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

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

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

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

Lifelong Machine Learning (LML)

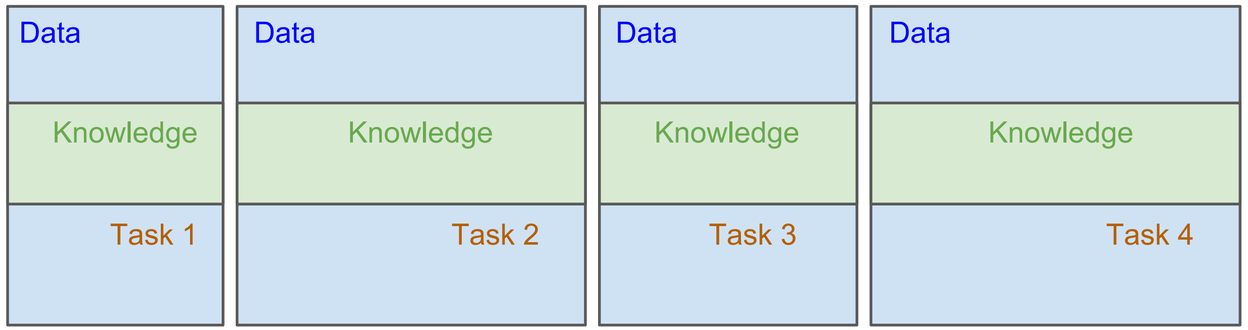
[Silver2013, Gepperth2016, Chen2016b]

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

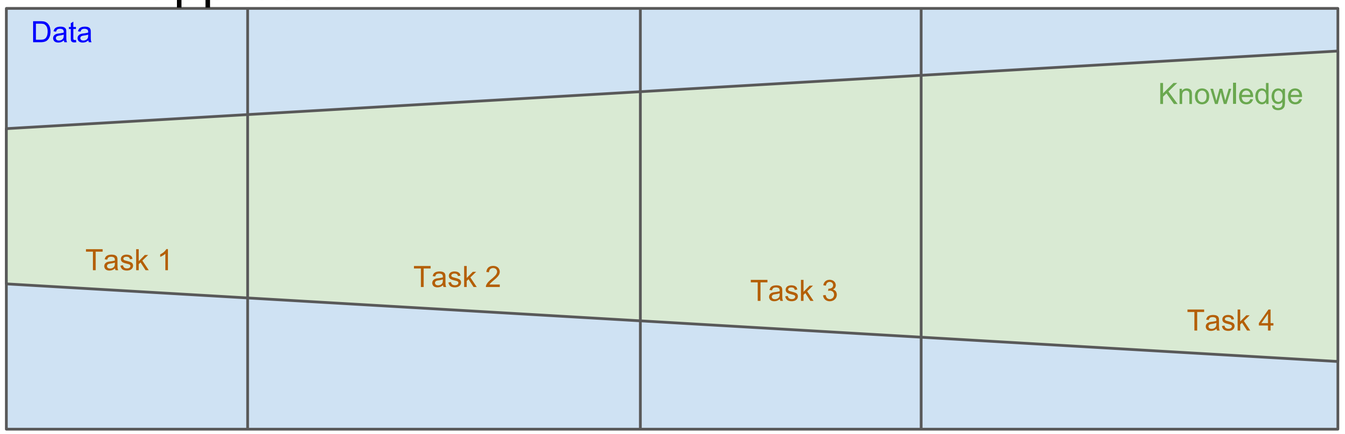
- Data streams, constantly arriving, not static ➔ Incremental learning
- Multiple tasks with multiple learning/mining algorithms
- Retain/accumulate learned knowledge in the past & use it to help future learning
  - Use past knowledge for inductive transfer when learning new tasks
- Mimics human way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

Lifelong Machine Learning (LML)

‘Classical’ approach



LML approach



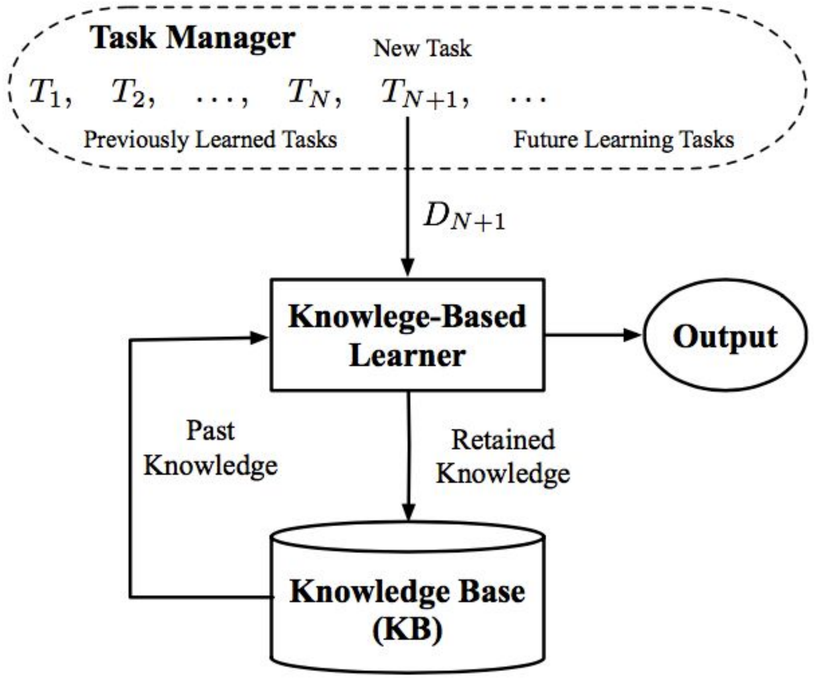
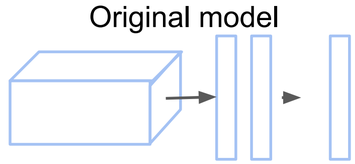


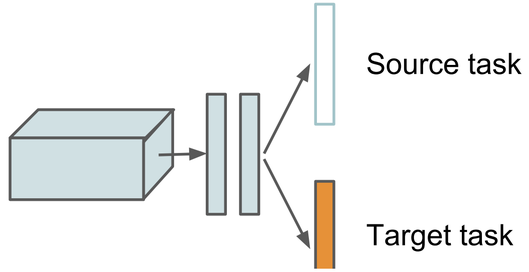
Image from [Chen2016a]

Related learning approaches

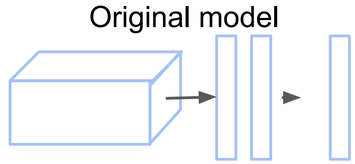


Transfer learning (finetuning):

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

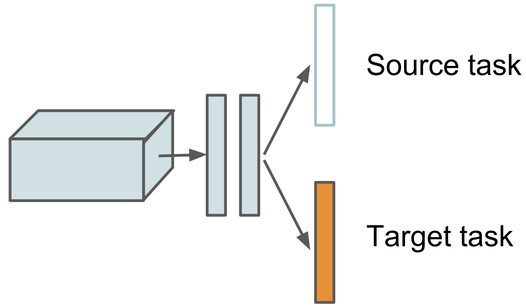


Related learning approaches



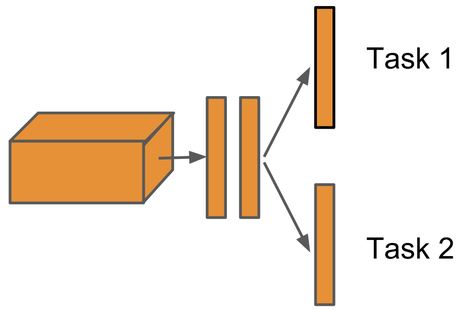
Transfer learning (finetuning):

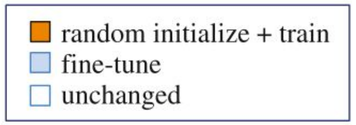
- Data in the source domain helps learning the target domain
- Less data is needed in the target domain
- Tasks must be similar



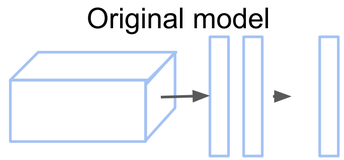
Multi-task learning:

- 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



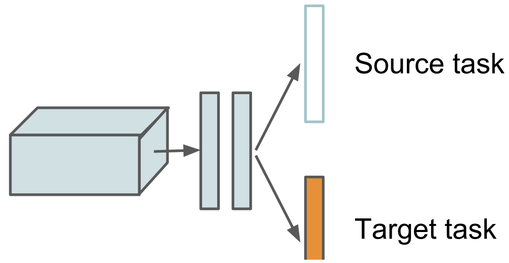


Related learning approaches

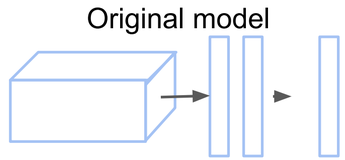


Transfer learning (finetuning):

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

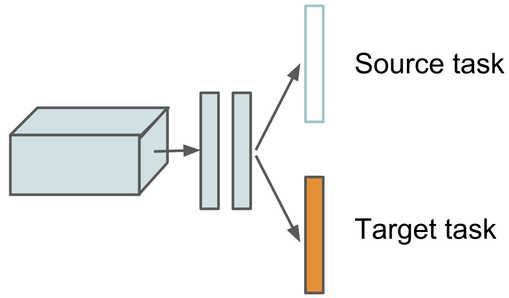


Related learning approaches



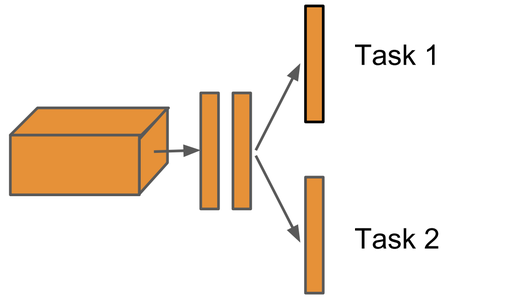
Transfer learning (finetuning):

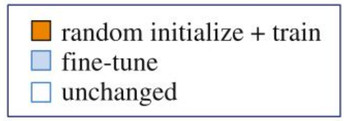
- Unidirectional: source ➜ target
- Not continuous
- No retention/accumulation of knowledge



Multi-task learning:

- Simultaneous learning
- All tasks data is needed for training

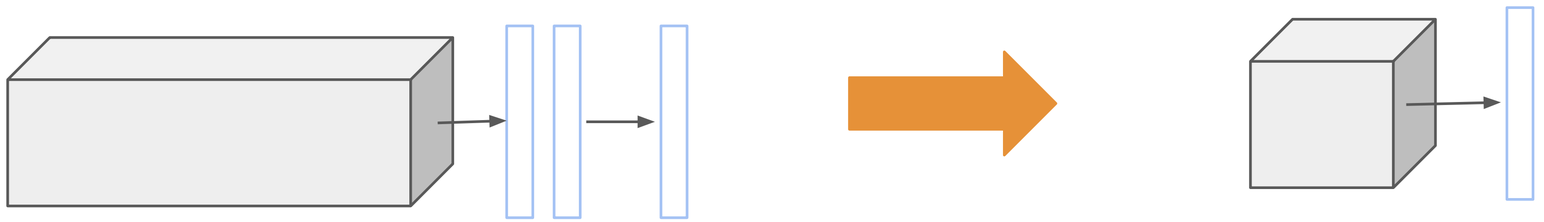




Lifelong Machine Learning Methods (LML)

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

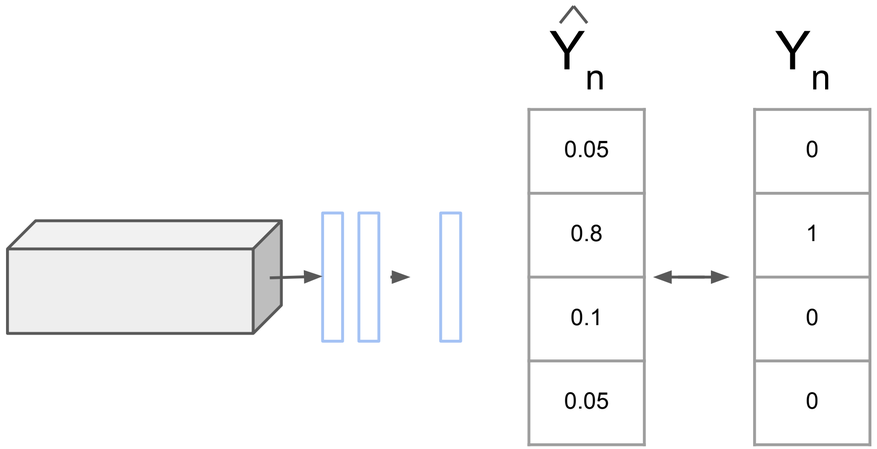
Distillation

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

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

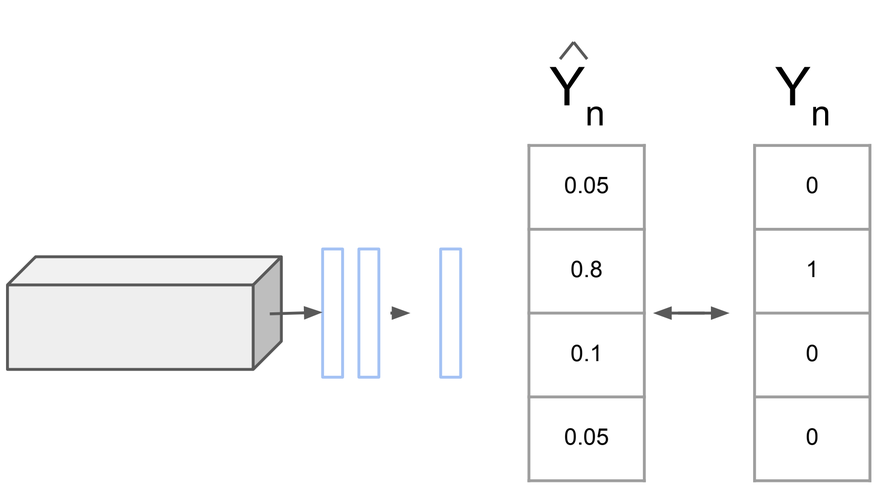
$$\mathcal{L}(\mathbf{y}_n, \hat{\mathbf{y}}_n) = -\mathbf{y}_n \cdot \log \hat{\mathbf{y}}_n$$

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

Distillation

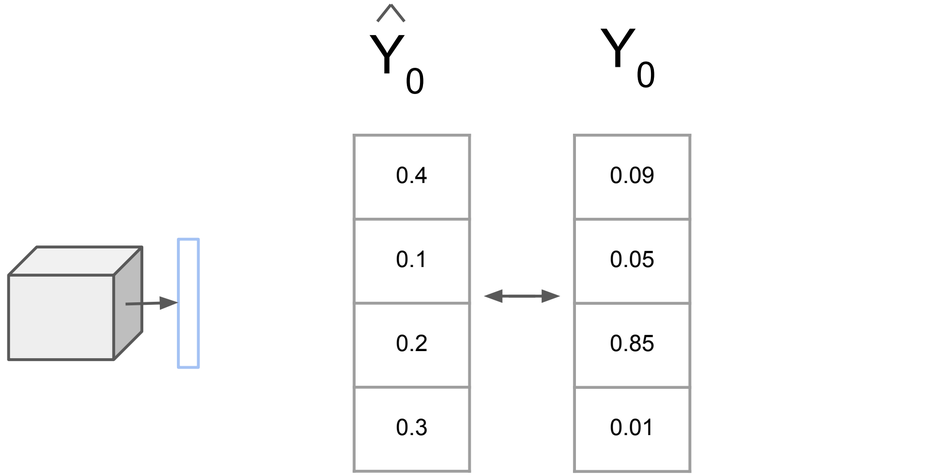
Idea: use the class probabilities produced by the large model as “soft targets” for training the small model

- 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

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

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

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

0.09
0.05
0.85
0.01

T=1

0.15
0.10
0.70
0.05

T>1

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

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

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

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

0.09
0.05
0.85
0.01

T=1

0.15
0.10
0.70
0.05

T>1

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**

LWF: Learning without Forgetting [Li2016]

Goal:

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

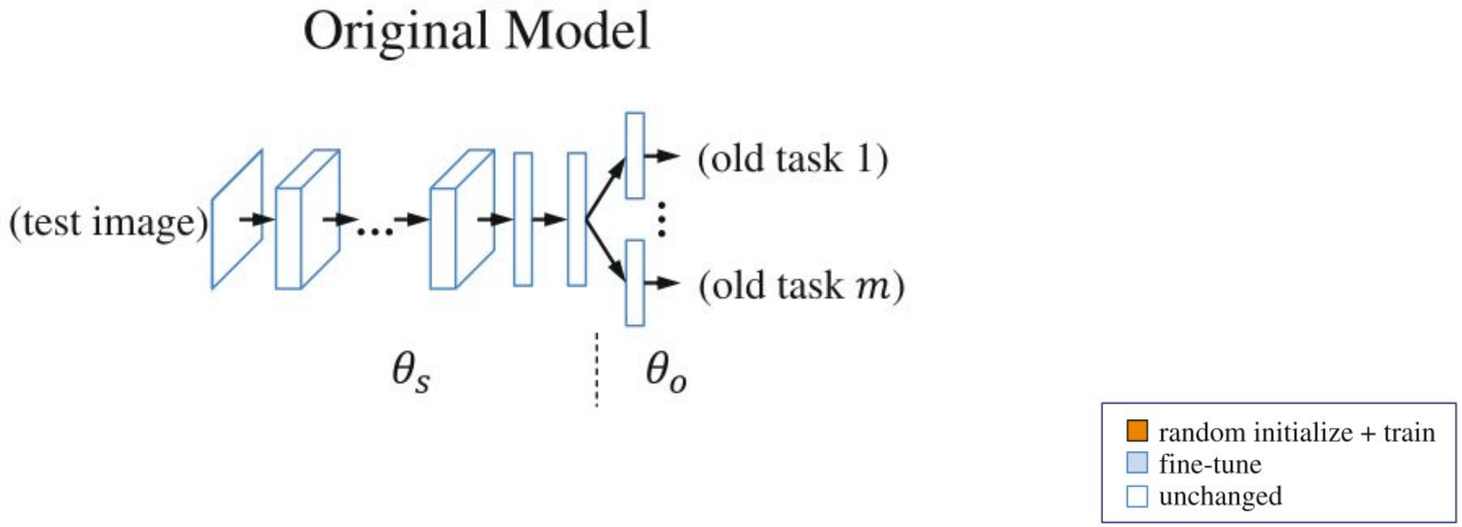
Solution:

Using only examples for the new task, optimize for :

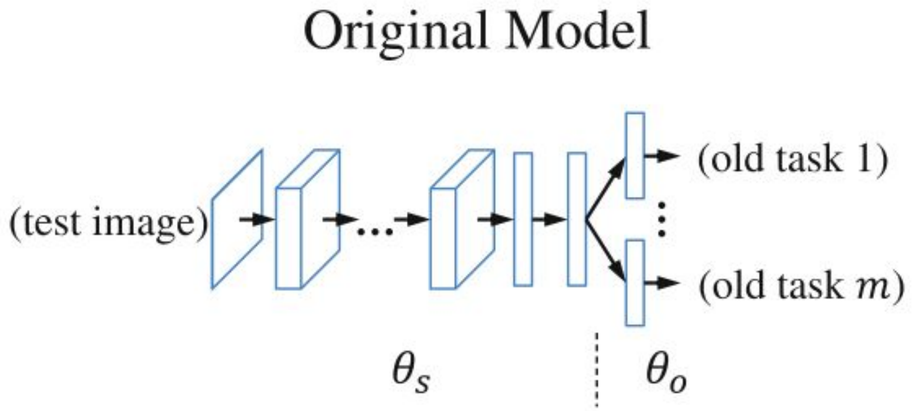
- High accuracy on the new task
- Preservation of responses on existing tasks from the original network (distillation, Hinton2015)
- Storage/complexity does not grow with time. Old samples are not kept

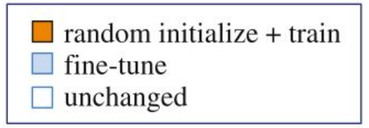
Preserves performance on old task
(even if images in new task provide a poor sampling of old task)

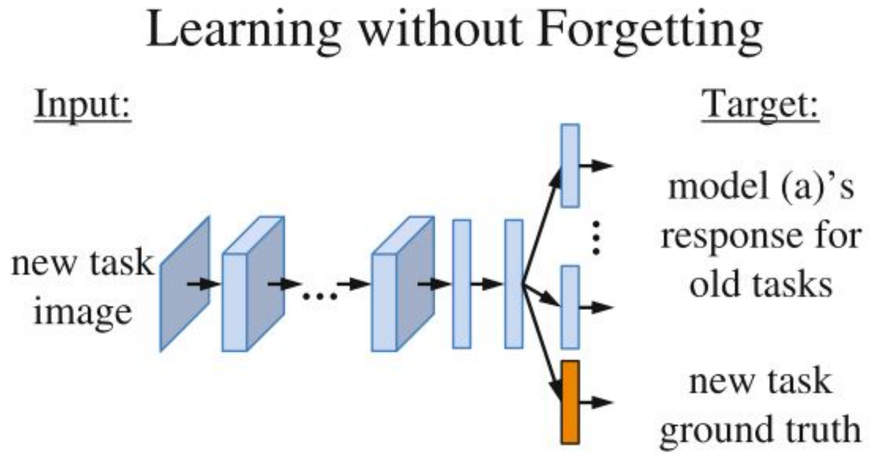
LWF: Learning without Forgetting [Li2016]



LWF: Learning without Forgetting [Li2016]







LWF: Learning without Forgetting [Li2016]

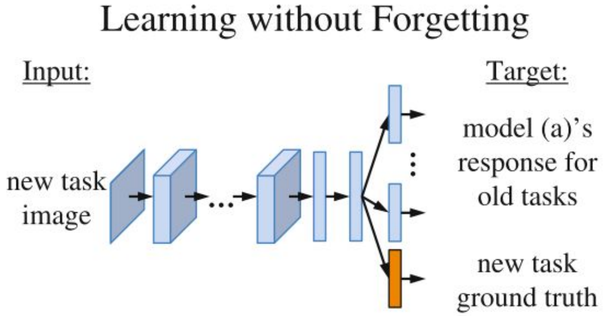
LEARNINGWITHOUTFORGETTING:

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



LWF: Learning without Forgetting [Li2016]

LEARNINGWITHOUTFORGETTING:

Start with:

θ_s: shared parameters

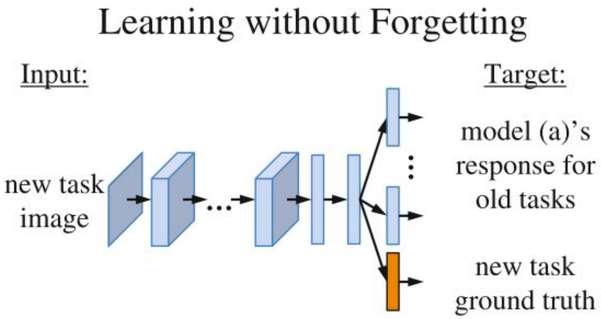
θ_o: task specific parameters for each old task

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

Initialize:

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

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



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

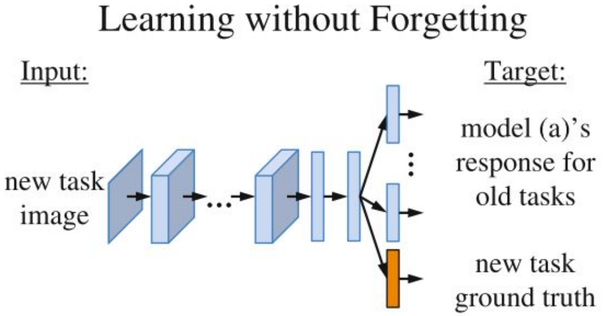
Initialize:

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

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

Train:

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



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:

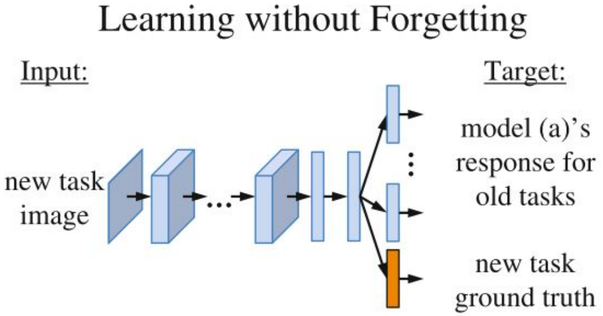
$Y_o \leftarrow \text{CNN}(X_n, \theta_s, \theta_o)$ // *compute output of old tasks for new data*

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

Train:

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*



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:

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

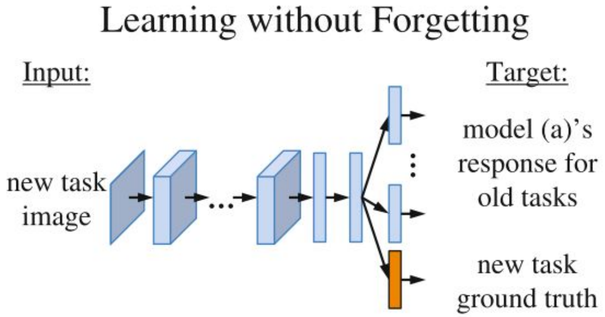
$\theta_n \leftarrow \text{RANDINIT}(|\theta_n|)$ *// randomly initialize new parameters*

Train:

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*

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



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

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

Distillation loss

LWF: Learning without Forgetting [Li2016]

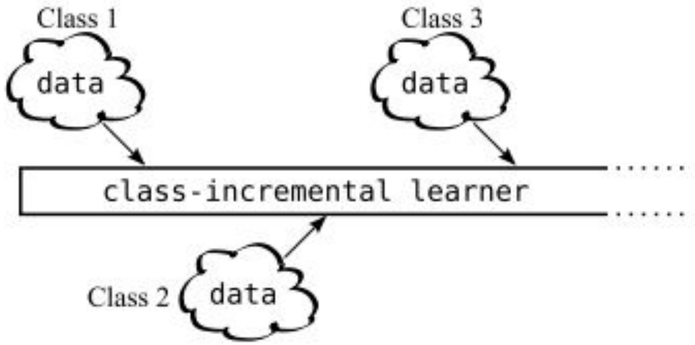
	Places365→VOC	
	old	new
LwF (ours)	50.6	73.7
Fine-tuning	-2.1	0.1
Feat. Extraction	1.3	-2.3
Joint Training	0.9	-0.1

Require access to old data

iCaRL

Goal:

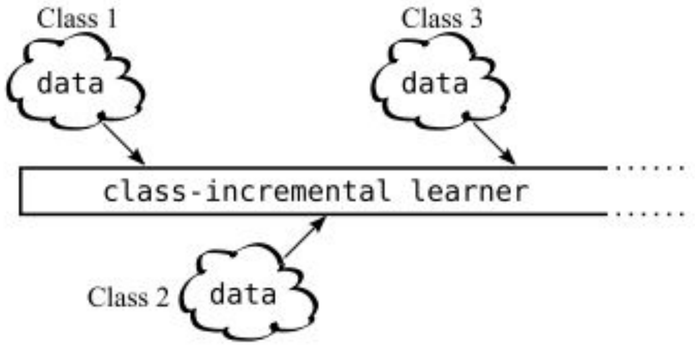
Add new classes based on adapting shared parameters **with restricted access** to training data for previously learned classes.



iCaRL

Goal:

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



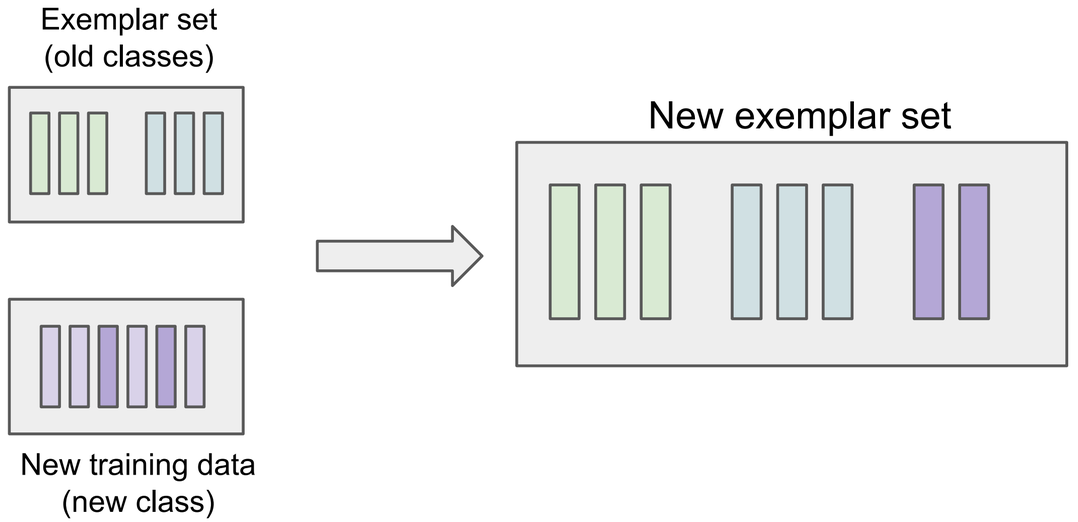
Solution:

- 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

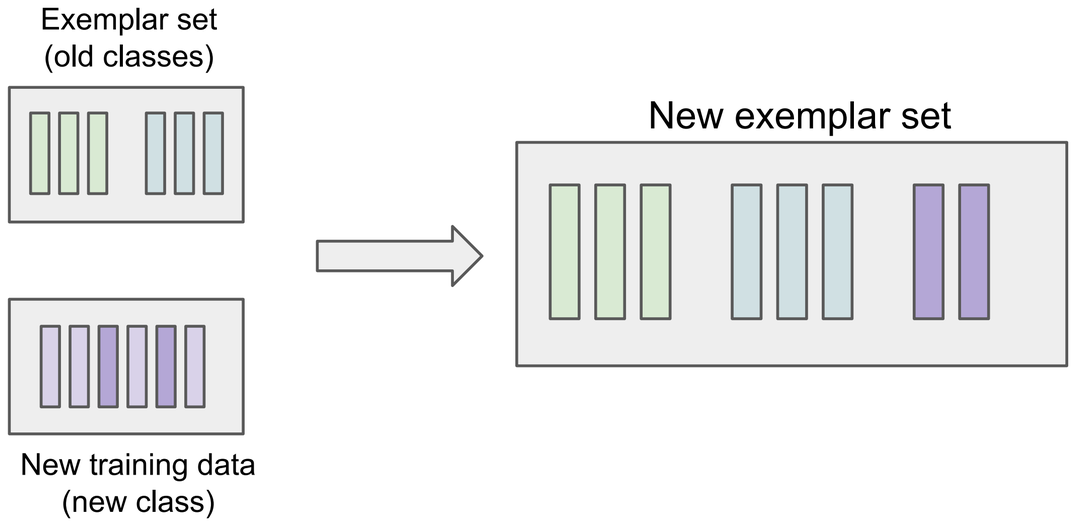
input $X^s, \ldots, X^t$ // training examples in per-class sets
input K // memory size
require Θ // current model parameters
require $\mathcal{P} = (P_1, \ldots, P_{s-1})$ // current exemplar sets



iCaRL: Incremental Classifier and Representation learning

Algorithm 2 iCaRL INCREMENTALTRAIN

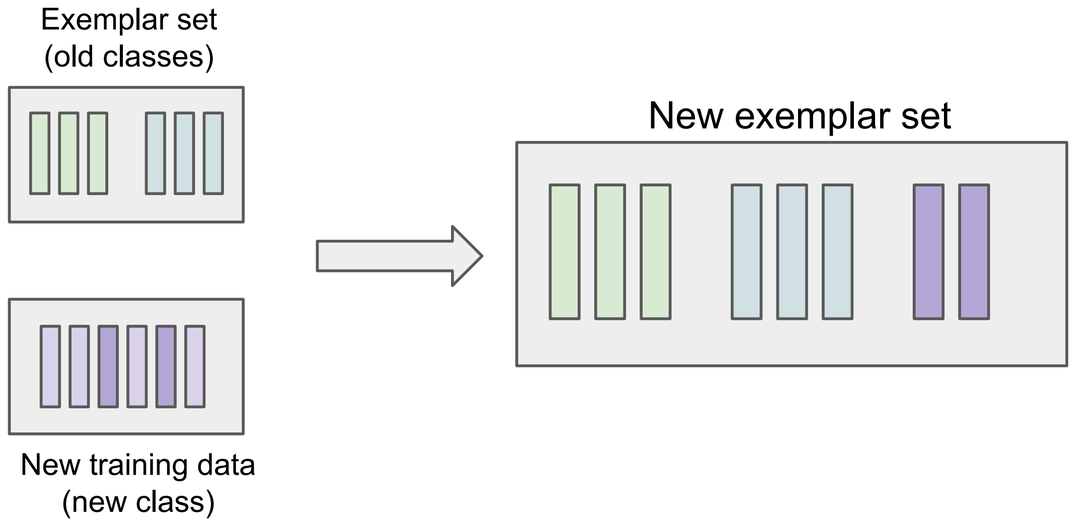
input $X^s, \ldots, X^t$ // training examples in per-class sets
input K // memory size
require Θ // current model parameters
require $\mathcal{P} = (P_1, \ldots, P_{s-1})$ // current exemplar sets
$\Theta \leftarrow$ UPDATEREPRESENTATION$(X^s, \ldots, X^t; \mathcal{P}, \Theta)$
$m \leftarrow K/t$ // number of exemplars per class



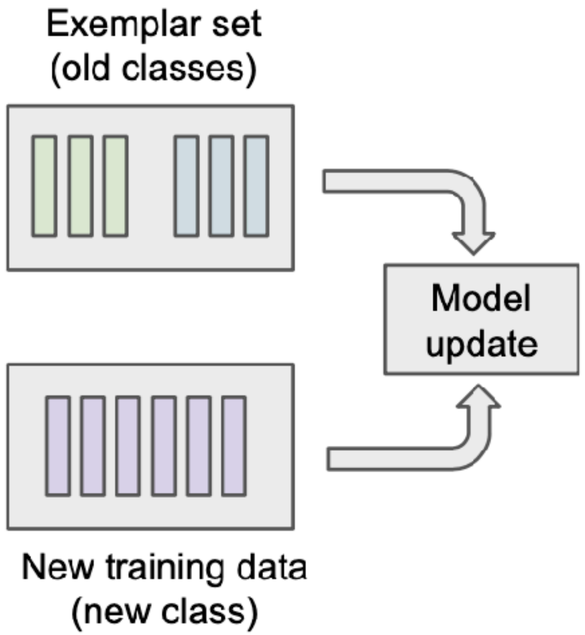
iCaRL: Incremental Classifier and Representation learning

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



iCaRL: Incremental Classifier and Representation learning

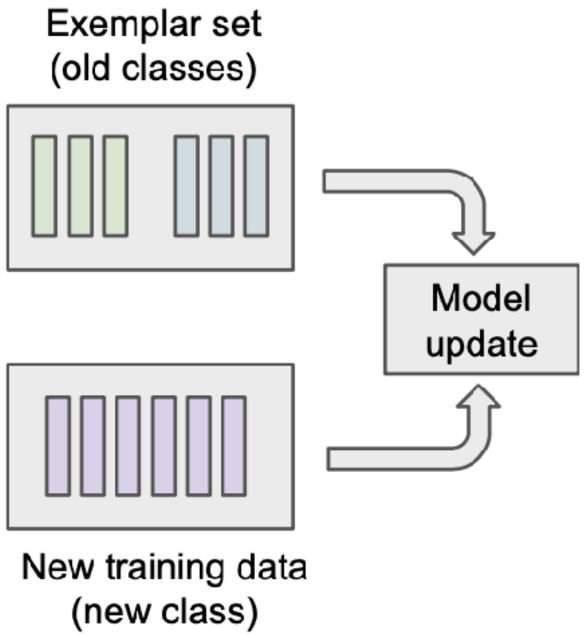


Algorithm 3 iCaRL UPDATEREPRESENTATION

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:

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

iCaRL: Incremental Classifier and Representation learning



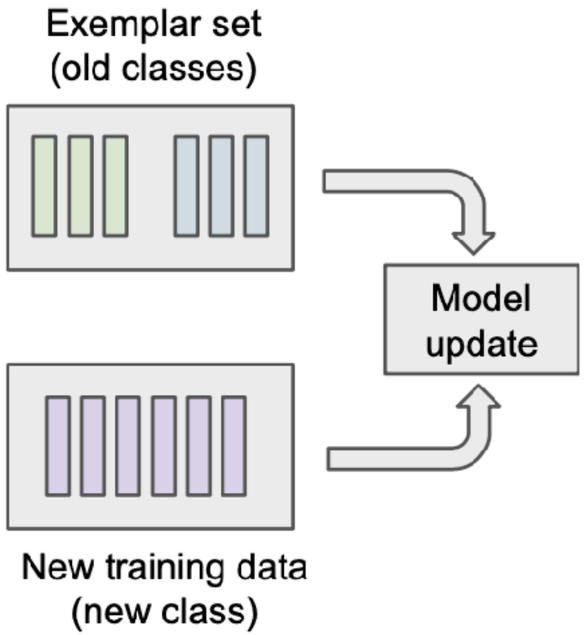
Algorithm 3 iCaRL UPDATEREPRESENTATION

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

iCaRL: Incremental Classifier and Representation learning



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:

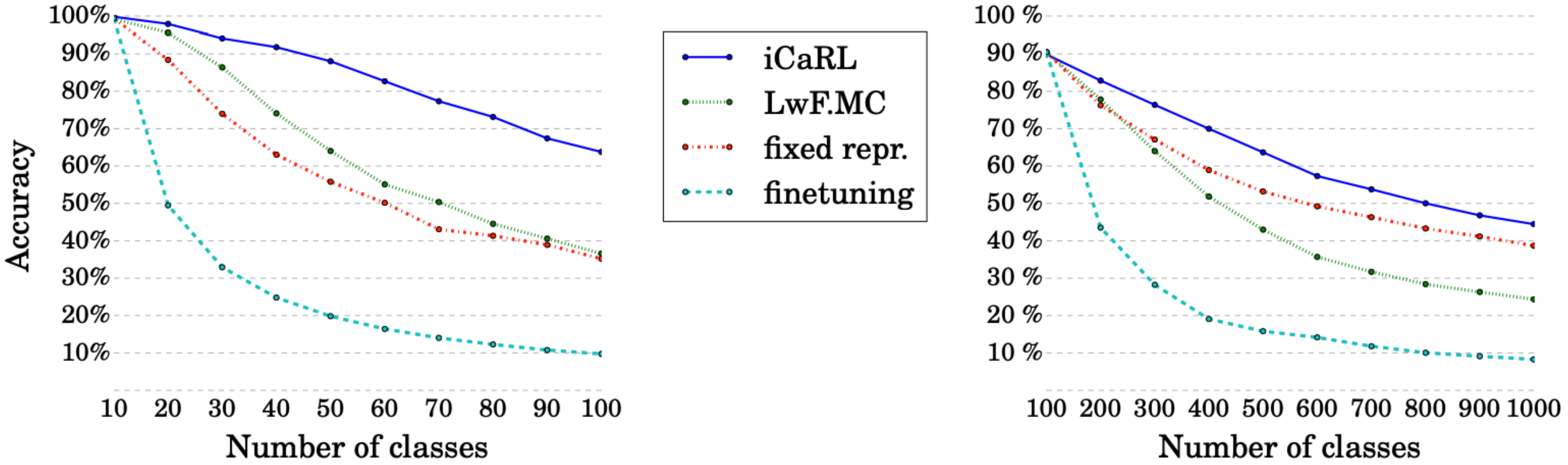
$$\mathcal{D} \leftarrow \bigcup_{y=s,\ldots,t} \{(x,y) : x \in X^y\} \cup \bigcup_{y=1,\ldots,s-1} \{(x,y) : x \in P^y\}$$

// 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

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

that consists of *classification* and *distillation* terms.

iCaRL: Incremental Classifier and Representation learning



(b) Top-5 accuracy on iILSVRC-small (top) and iILSVRC-full (bottom).

Progressive Neural Networks

Goal:

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

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

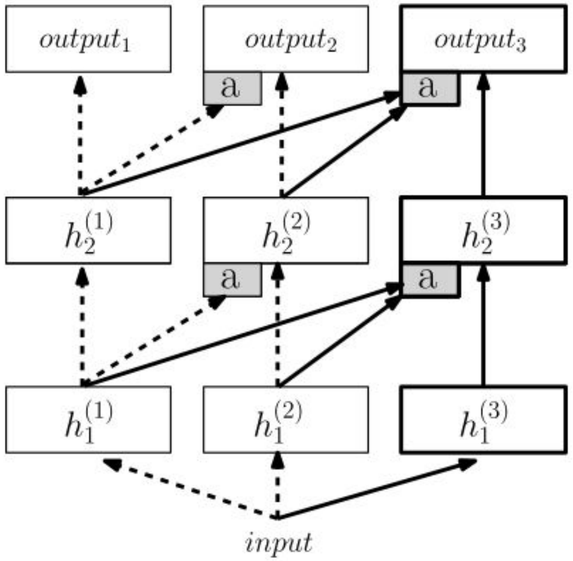
Progressive Neural Networks

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



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

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

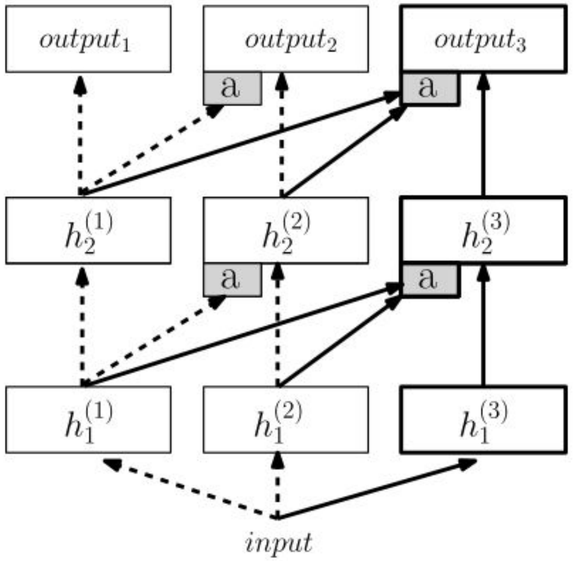
Progressive Neural Networks

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.



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

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

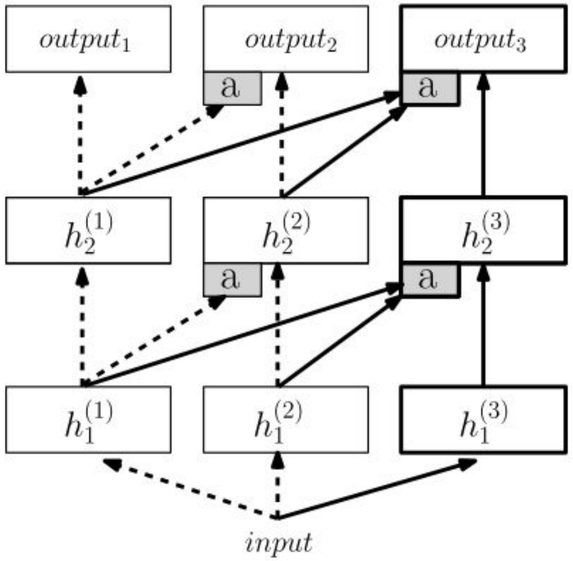
Progressive Neural Networks

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

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

Increasing model capacity (I)

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

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.

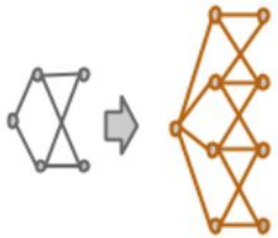
Increasing model capacity (I)

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.

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

Increasing model capacity (II)

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

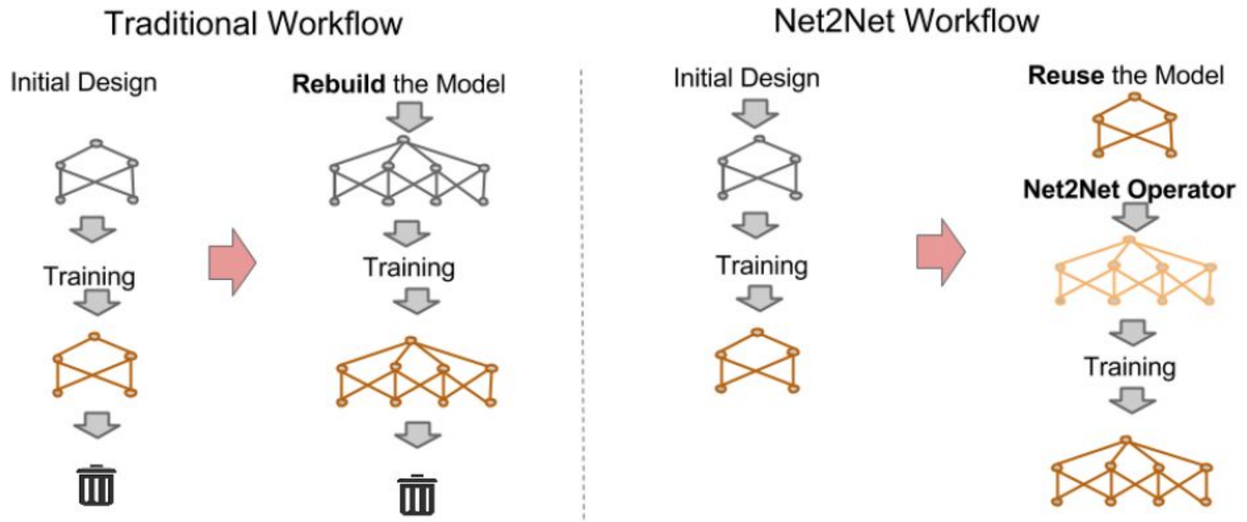
If the capacity of the network has to be incremented we want to avoid retraining the new network from scratch

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)

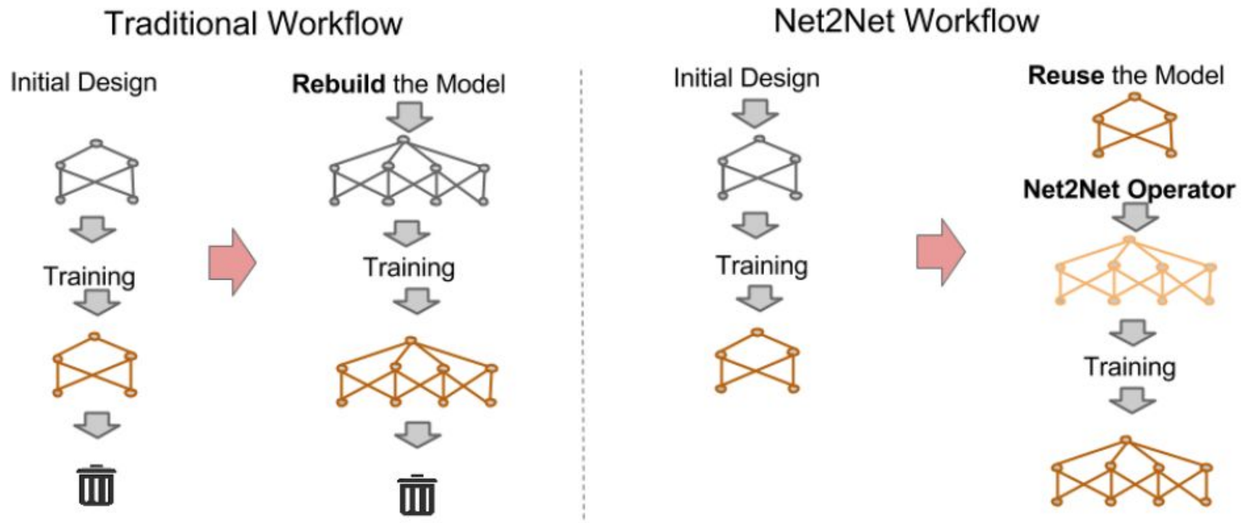
- The new, larger network immediately performs as well as the original network, rather than spending time passing through a period of low performance.



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

Increasing model capacity: Net2Net (I)

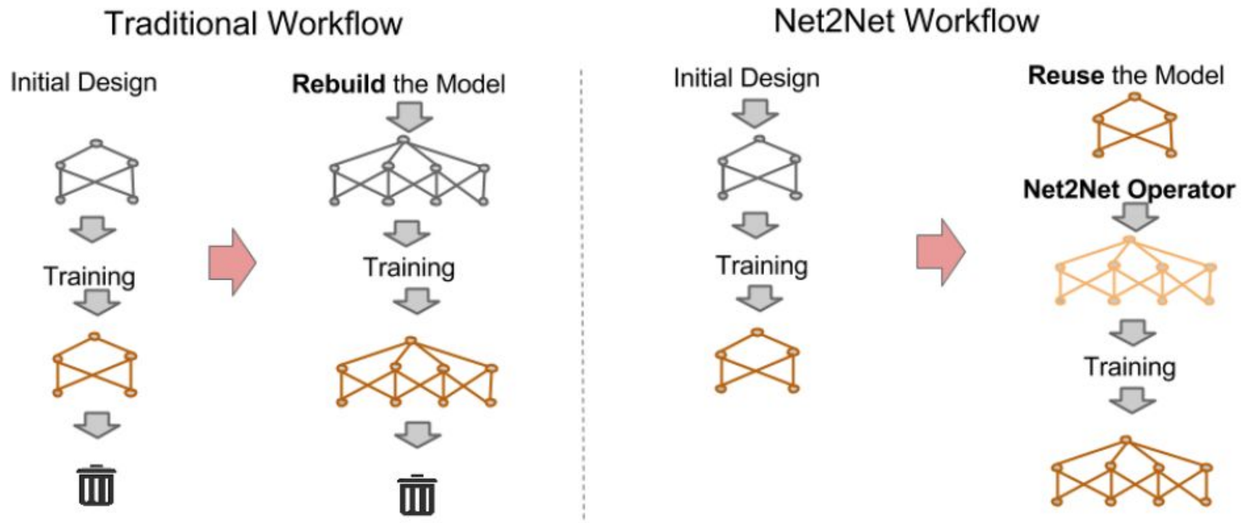
- The new, larger network immediately performs as well as the original network, rather than spending time passing through a period of low performance.
- Any change made to the network after initialization is guaranteed to be an improvement, so long as each local step is an improvement.



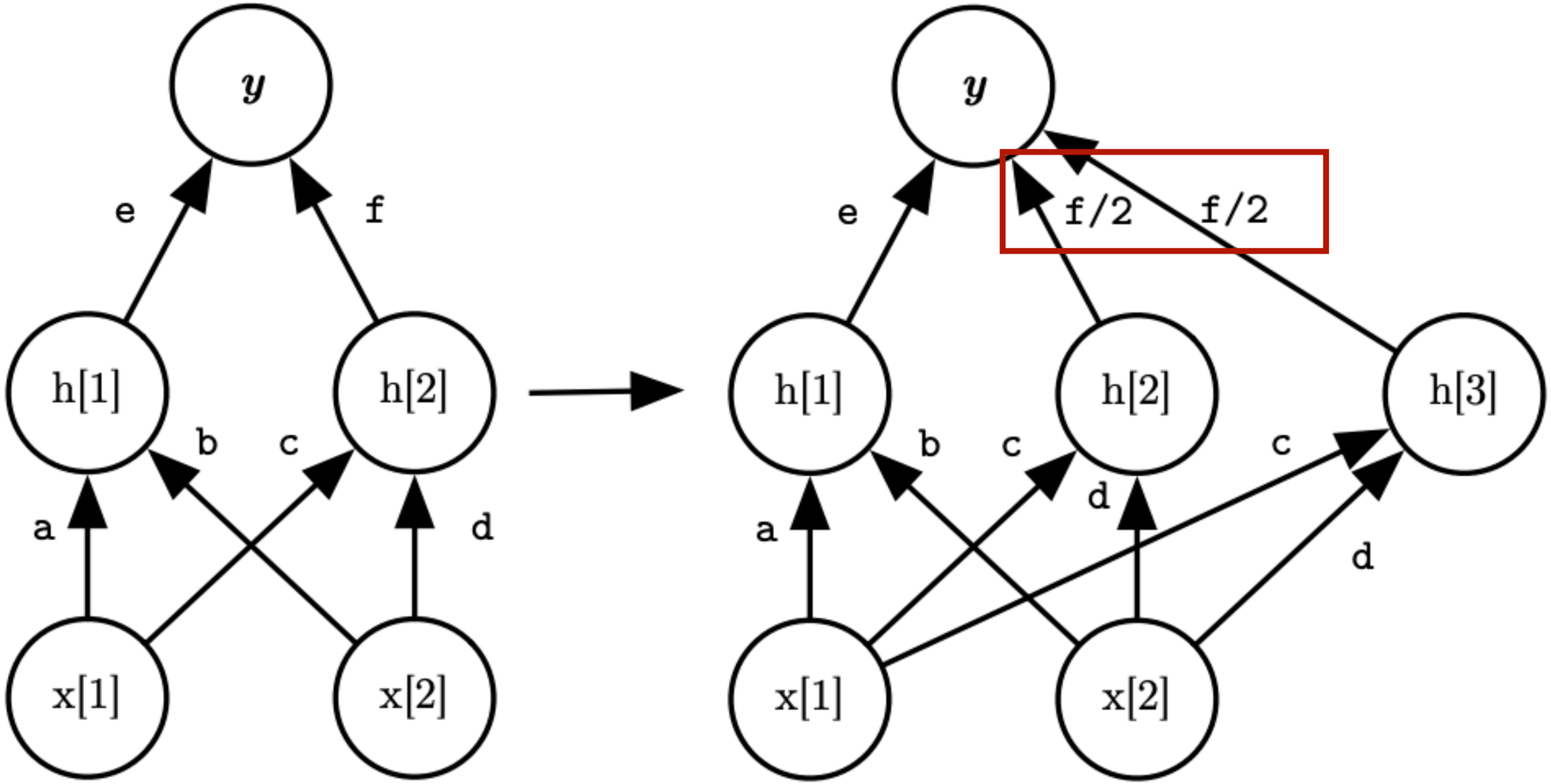
Chen, T., Goodfellow, I., & Shlens, J. (2016). Net2Net: Accelerating Learning via Knowledge Transfer. In *ICLR 2016*

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*


y
e
f
h[1]
h[2]
b
c
a
d
x[1]
x[2]
y
e
f/2
f/2
h[1]
h[2]
h[3]
b
c
c
d
a
d
x[1]
x[2]

Discovering new classes

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

Discovering new classes

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

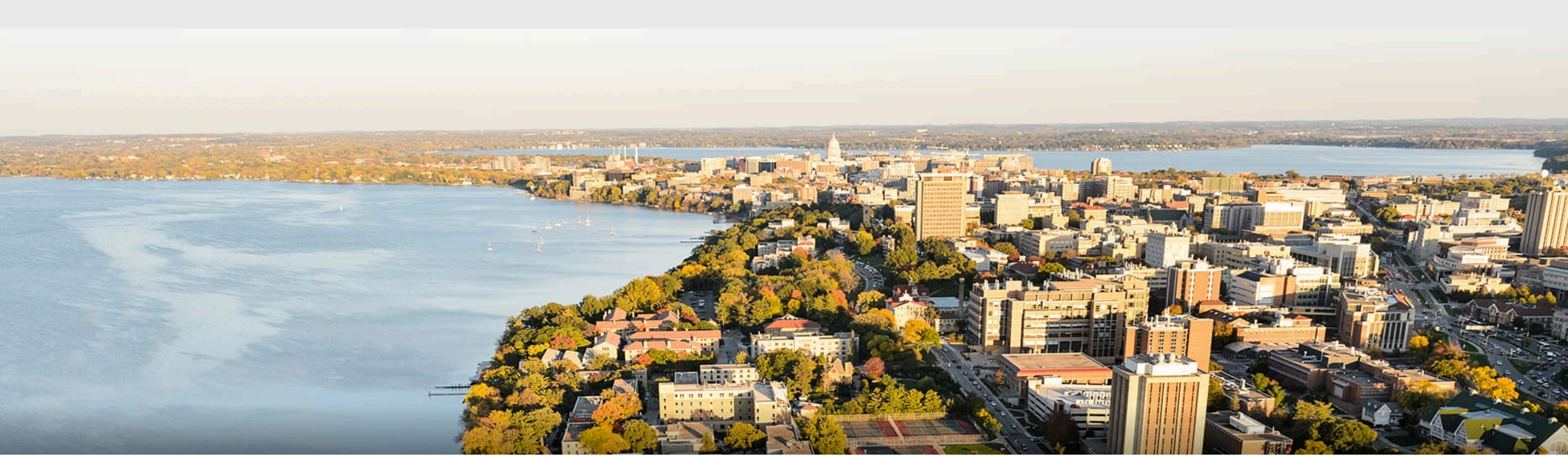
New classes may appear over time. Systems need a way to detect them and to introduce them in the learning process

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

Summary

- **The need and challenges for LML**
  - Constant data stream
  - Multiple tasks
  - Knowledge retention
- **LML Methods**
  - Learning without Forgetting (LwF)
  - iCaRL
  - Progressive Neural Networks
- **Growing capacity & discover new classes**
  - Net2Net
  - OOD detection (from Theme 2)



Questions?

Acknowledgement: slides are adapted from Ramon Morros's tutorial at https://telecombcn-dl.github.io/2017-dlai/