CMSC 491/691 Robust Machine Learning

Topic 6: Continual Learning

UMBC

Some slides adapted from Sharon Li and Ramon Morros (2017 tutorial)

Traditional Machine Learning

- Collect once, train once, deploy once
  - Data is collected prior to model training
  - Model is trained prior to deployment
  - Once deployed, no updates
- Assumptions:
  - Data distribution D is static – it doesn't evolve or shift
  - You can always access and sample from D

Do HUMANS (or any natural learning system) learn like this?

- Collect once, train once, deploy once
  - Data is collected prior to model training
  - Model is trained prior to deployment
  - Once deployed, no updates
- Assumptions:
  - Data distribution D is static – it doesn't evolve or shift
  - You can always access and sample from D

Do HUMANS (or any natural learning system) learn like this?

- Collect once, train once, deploy once
  - Data is collected prior to model training

No ... humans learn "on the job"

- Assumptions:
  - Data distribution D is static – it doesn't evolve or shift
  - You can always access and sample from D

Do HUMANS (or any natural learning system) learn like this?

- Collect once, train once, deploy once
  - Data is collected prior to model training

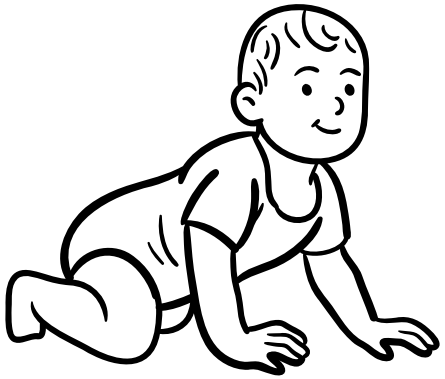
No ... humans learn "on the job"

- Assumptions:
  - Data distribution D is static – it doesn't evolve or shift

Humans aren't just given a million samples a priori ...

Challenges

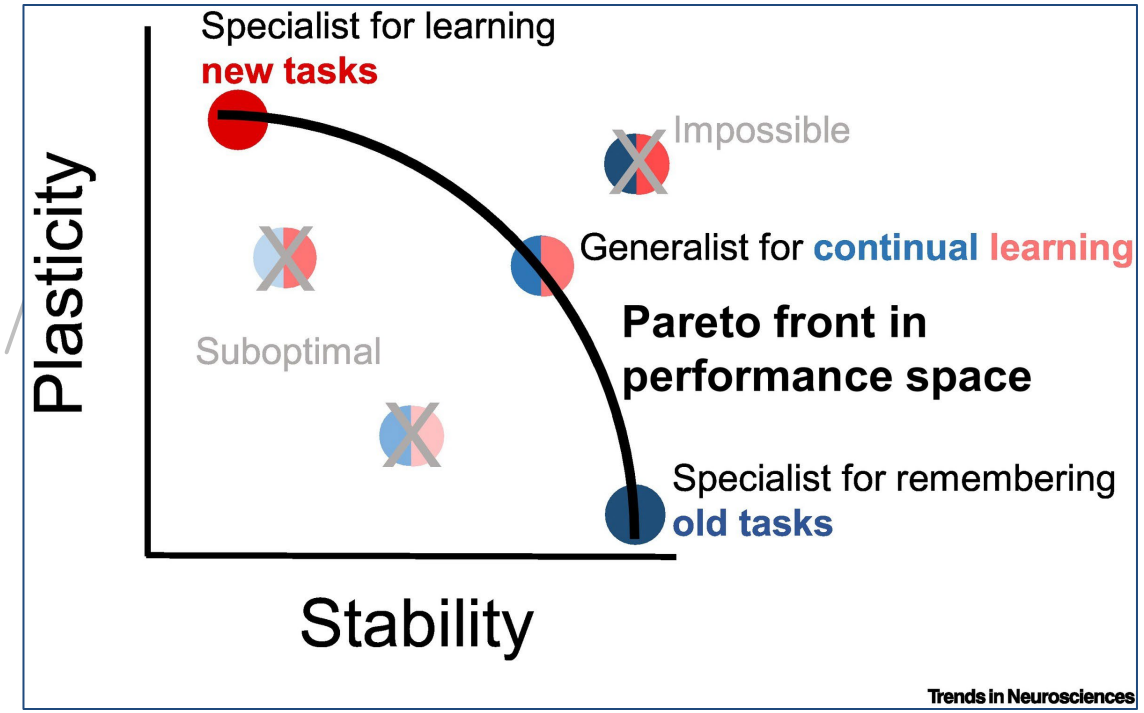
- Data isn't always available a priori
  - Can appear incrementally ...
    - Infants see *mom, dad, siblings, family* first and learn to recognize the differences *(also do "OOD" detection ...)*
    - Then they learn to move around – new objects
    - Then they see outside the window: birds, animals, cars, trees
    - Object recognition builds up over time (vocabulary expands parallelly)



- Concept drift: data distribution may change/evolve over time
  - Changes in appearance, seasons, changes in object frequency *(anyone use a rotary phone recently?)*

Challenges

- Data isn't always available a priori
- Concept drift: data distribution may change/



- Stability-Plasticity Trade-off: when and how to adapt?
  - Rapid adaptation may lead to forgetting old info
  - Slow adaptation may limit the ability to quickly grasp new information

We need a way for models to overcome the challenges of “static” ML ...

Idea:

Can we develop “Lifelong” or “Continual” ML algorithms?

ELSEVIER

Robotics and Autonomous Systems 15 (1995) 25–46

Robotics and Autonomous Systems

Lifelong robot learning ★

Sebastian Thrun [a,*], Tom M. Mitchell [b]

[a] *University of Bonn, Institut für Informatik III, Römerstr. 164, 53117 Bonn, Germany*

[b] *School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA*

Abstract

Learning provides a useful tool for the automatic design of autonomous robots. Recent research on learning robot control has predominantly focused on learning single tasks that were studied in isolation. If robots encounter a multitude of control learning tasks over their entire lifetime there is an opportunity to transfer knowledge between them. In order to do so, robots may learn the invariants and the regularities of the individual tasks and environments. This task-independent knowledge can be employed to bias generalization when learning control, which reduces the need for real-world experimentation. We argue that knowledge transfer is essential if robots are to learn control with moderate learning times in complex scenarios. Two approaches to lifelong robot learning which both capture invariant knowledge about the robot and its environments are presented. Both approaches have been evaluated using a HERO-2000 mobile robot. Learning tasks included navigation in unknown indoor environments and a simple find-and-fetch task.

AAAI 2015

Never-Ending Learning

T. Mitchell*, W. Cohen*, E. Hruschka†¶, P. Talukdar‡¶, J. Betteridge*, A. Carlson§¶, B. Dalvi*, M. Gardner*, B. Kisiel*, J. Krishnamurthy*, N. Lao§¶, K. Mazaitis*, T. Mohammad¶, N. Nakashole*, E. Platanios*, A. Ritter‖¶, M. Samadi*, B. Settles¶, R. Wang¶, D. Wijaya*, A. Gupta*, X. Chen*, A. Saparov*, M. Greaves††¶, J. Welling‡‡**

tom.mitchell@cs.cmu.edu

NELL: Never-Ending Language Learning

Can computers learn to read? We think so. "Read the Web" is a research project that attempts to create a computer system that learns over time to read the web. Since January 2010, our computer system called NELL (Never-Ending Language Learner) has been running continuously, attempting to perform two tasks each day:

- First, it attempts to "read," or extract facts from text found in hundreds of millions of web pages (e.g., `playsInstrument(George_Harrison, guitar)`).
- Second, it attempts to improve its reading competence, so that tomorrow it can extract more facts from the web, more accurately.

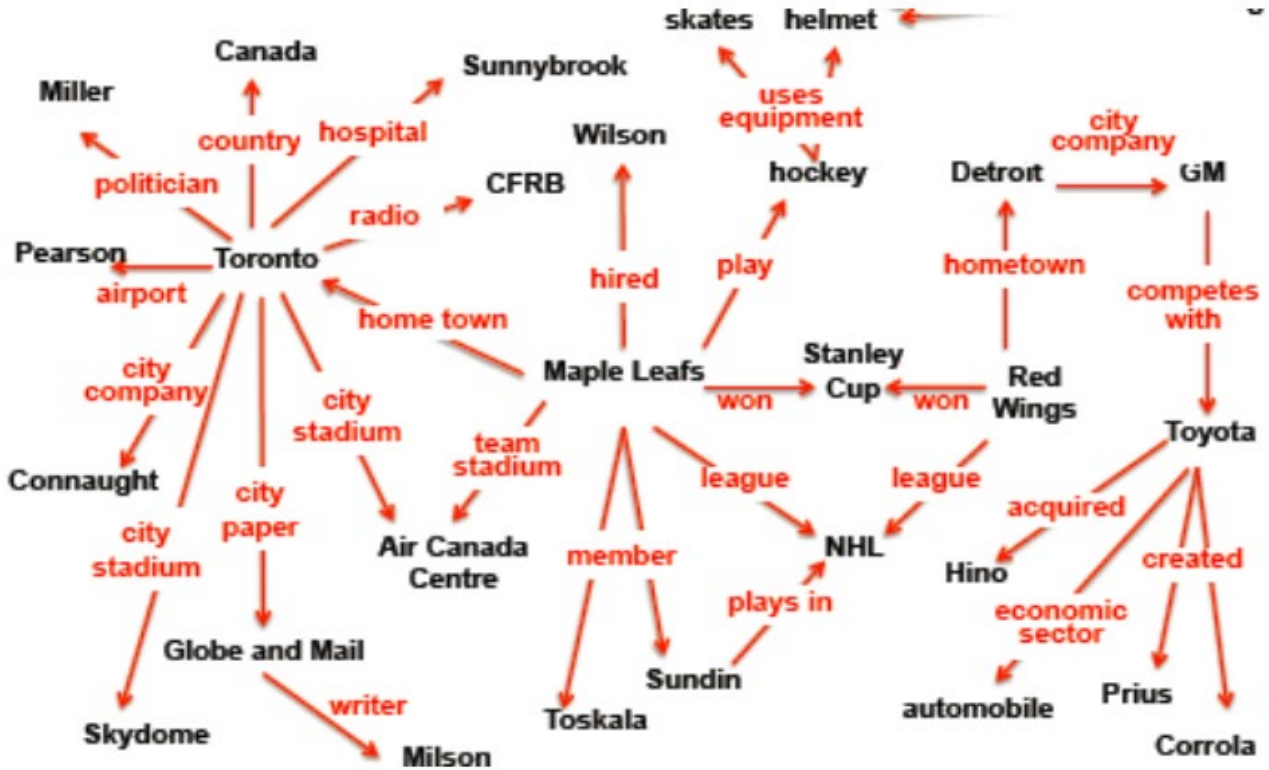
So far, NELL has accumulated over 50 million candidate beliefs by reading the web, and it is considering these at different levels of confidence. NELL has high confidence in 2,810,379 of these beliefs — these are displayed on this website. It is not perfect, but NELL is learning. You can track NELL's progress below or @cmunell on Twitter, browse and download its knowledge base, read more about our technical approach, or join the discussion group.

AAAI 2015

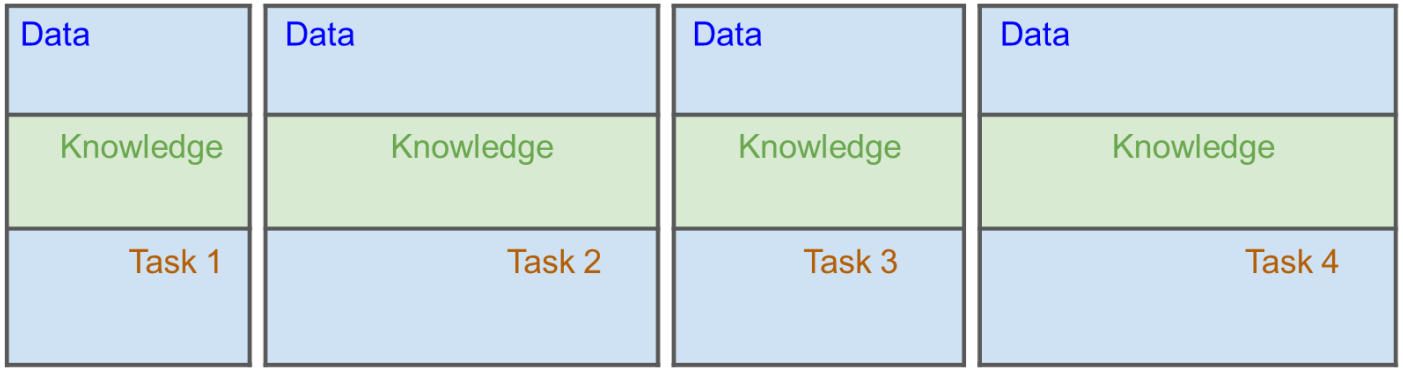
Never-Ending Learning

T. Mitchell*, W. Cohen*, E. Hruschka†¶, P. Talukdar‡¶, J. Betteridge*, A. Carlson§¶, B. Dalvi*, M. Gardner*, B. Kisiel*, J. Krishnamurthy*, N. Lao§¶, K. Mazaitis*, T. Mohammad¶, N. Nakashole*, E. Platanios*, A. Ritter‖¶, M. Samadi*, B. Settles**¶, R. Wang¶, D. Wijaya*, A. Gupta*, X. Chen*, A. Saparov*, M. Greaves††¶, J. Welling‡‡

tom.mitchell@cs.cmu.edu



“Lifelong” / “Continual” ML



- Multiple “episodes” or “experiences” or “tasks”
  - Occur sequentially, not all-at-once
- Broad Goal:
  - Learn new tasks
  - Retain knowledge of old tasks

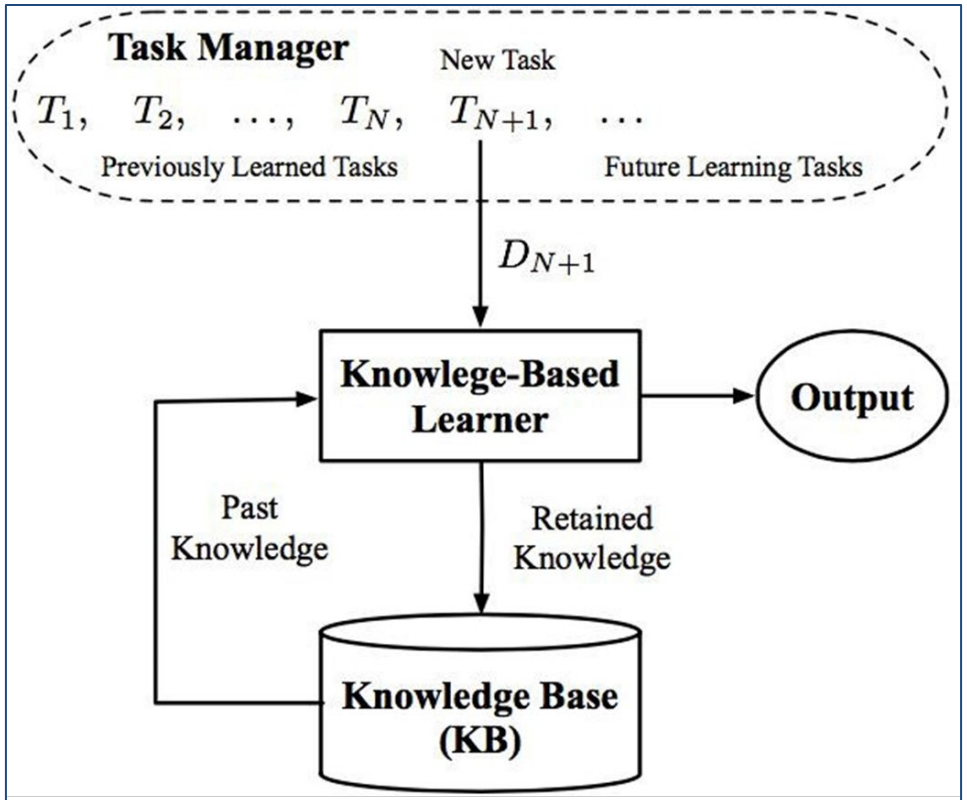


Image from [Chen2016a]

Old Definition of Lifelong Learning

(Thrun 1996, Silver et al 2013; Ruvolo and Eaton, 2013; Chen and Liu, 2014, Chen and Liu, 2016)

- The learner has performed learning on a sequence of tasks from 1 to *N*.
- When faced with the new or (*N*+1)th task, it uses the relevant knowledge in its *knowledge base* (KB) to help learn the (*N*+1)th task.
- After learning (*N*+1)th task, KB is updated with learned results from the (*N*+1)th task.

Thrun. Is learning the n-th thing any easier than learning the first? NIPS-1996.

New definition of lifelong/continual learning

(Chen and Liu, 2014, 2018)

- Learn a sequence of tasks, T_1, T_2, …, T_N, … incrementally. Each task *t* has a training dataset $\mathcal{D}_t = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$.
- Goal: learn each new task T_{N+1} incrementally
  1. **without catastrophic forgetting: Learning of new task T_{N+1} should not result in degradation of accuracy for the previous *N* tasks.**
  2. with knowledge transfer: leveraging the knowledge learned from the previous tasks to learn the new task T_{N+1} better.
- Assumption: Once a task is learned, its data is no longer accessible, at least a majority of it, and both the task T_{N+1}, and
  - its training data D_{N+1} are **given** by the user.

Chen and Liu. Lifelong machine learning. Morgan & Claypool. 2018

Three Main Problem Formulations

1. Class Incremental Learning (CIL)

   - Example: Today we learn to recognize *pig* and *chicken* (one task), and tomorrow, we also learn to recognize *sheep* (another task)

2. Task Incremental Learning (TIL)

   In the TIL setup, each task is a separate classification problem (e.g., one task could be to classify different breeds of dogs and another task could be to classify different types of birds).

3. Domain Incremental Learning

   - Example: One task is to classify car reviews as positive or negative and another task is to classify camera reviews as positive or negative. Car and camera are two domains.

Online Learning

- Training examples come in incrementally (online setting)
  - Computationally infeasible to train over the entire dataset
  - Training data come gradually in a data stream
- Different from CL
  - Online learning still performs the same learning task over time, no data distribution change.
  - CL aims to learn from a sequence of different tasks, retaining and accumulating knowledge

Several Other Problem Formulations

- *Instance-Incremental Learning* (IIL): All training samples belong to the same task and arrive in batches.
- *Domain-Incremental Learning* (DIL): Tasks have the same data label space but different input distributions. Task identities are not required.
- *Task-Incremental Learning* (TIL): Tasks have disjoint data label spaces. Task identities are provided in both training and testing.
- *Class-Incremental Learning* (CIL): Tasks have disjoint data label spaces. Task identities are only provided in training.
- *Task-Free Continual Learning* (TFCL): Tasks have disjoint data label spaces. Task identities are not provided in either training or testing.
- *Online Continual Learning* (OCL): Tasks have disjoint data label spaces. Training samples for each task arrive as a one-pass data stream.
- *Blurred Boundary Continual Learning* (BBCL): Task boundaries are blurred, characterized by distinct but overlapping data label spaces.
- *Continual Pre-training* (CPT): Pre-training data arrives in sequence. The goal is to improve knowledge transfer to downstream tasks.

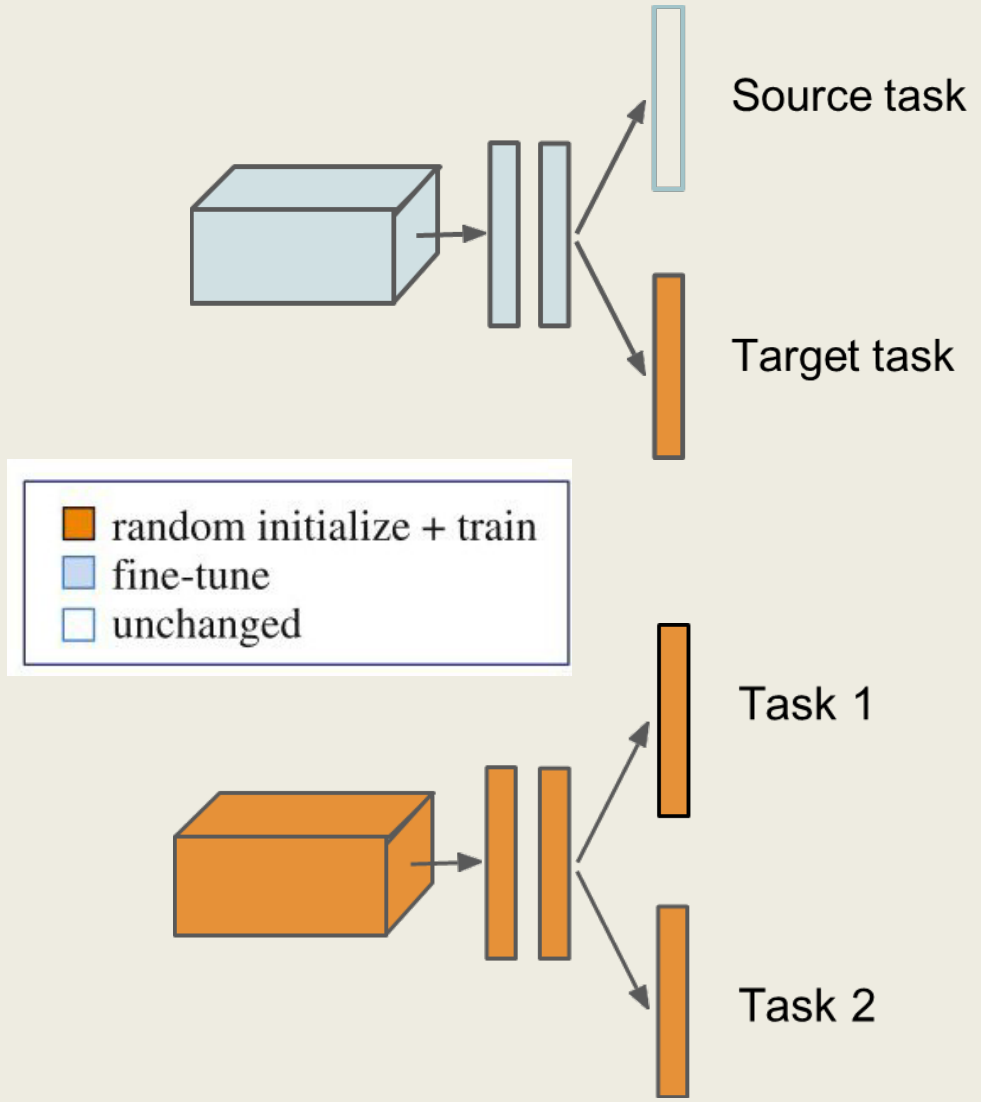
TABLE 1
A formal comparison of typical continual learning scenarios. $\mathcal{D}_{t,b}$: the training samples of task t and batch b. $|b|$: the size of batch b. $\mathcal{B}_t$: the space of incremental batches belonging to task t. $\mathcal{D}_t$: the training set of task t (further specified as $\mathcal{D}_t^{pt}$ for pre-training). $\mathcal{T}$: the space of all incremental tasks (further specified as $\mathcal{T}^{pt}$ for pre-training). $\mathcal{X}_t$: the input data in $\mathcal{D}_t$. $p(\mathcal{X}_t)$: the distribution of $\mathcal{X}_t$. $\mathcal{Y}_t$: the data label of $\mathcal{X}_t$.

Scenario	Training	Testing
IIL [279]	$\{\{\mathcal{D}_{t,b}, t\}_{b\in\mathcal{B}_t}\}_{t=j}$	$\{p(\mathcal{X}_t)\}_{t=j}$; t is not required
DIL [167], [423]	$\{\mathcal{D}_t, t\}_{t\in\mathcal{T}}$; $p(\mathcal{X}_i) \neq p(\mathcal{X}_j)$ and $\mathcal{Y}_i = \mathcal{Y}_j$ for $i \neq j$	$\{p(\mathcal{X}_t)\}_{t\in\mathcal{T}}$, t is not required
TIL [167], [423]	$\{\mathcal{D}_t, t\}_{t\in\mathcal{T}}$; $p(\mathcal{X}_i) \neq p(\mathcal{X}_j)$ and $\mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset$ for $i \neq j$	$\{p(\mathcal{X}_t)\}_{t\in\mathcal{T}}$; t is available
CIL [167], [423]	$\{\mathcal{D}_t, t\}_{t\in\mathcal{T}}$; $p(\mathcal{X}_i) \neq p(\mathcal{X}_j)$ and $\mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset$ for $i \neq j$	$\{p(\mathcal{X}_t)\}_{t\in\mathcal{T}}$; t is unavailable
TFCL [15]	$\{\{\mathcal{D}_{t,b}\}_{b\in\mathcal{B}_t}\}_{t\in\mathcal{T}}$; $p(\mathcal{X}_i) \neq p(\mathcal{X}_j)$ and $\mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset$ for $i \neq j$	$\{p(\mathcal{X}_t)\}_{t\in\mathcal{T}}$; t is optionally available
OCL [16]	$\{\{\mathcal{D}_{t,b}\}_{b\in\mathcal{B}_t}\}_{t\in\mathcal{T}}$, $\|b\| = 1$; $p(\mathcal{X}_i) \neq p(\mathcal{X}_j)$ and $\mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset$ for $i \neq j$	$\{p(\mathcal{X}_t)\}_{t\in\mathcal{T}}$; t is optionally available
BBCL [27], [46]	$\{\mathcal{D}_t, t\}_{t\in\mathcal{T}}$; $p(\mathcal{X}_i) \neq p(\mathcal{X}_j)$, $\mathcal{Y}_i \neq \mathcal{Y}_j$ and $\mathcal{Y}_i \cap \mathcal{Y}_j \neq \emptyset$ for $i \neq j$	$\{p(\mathcal{X}_t)\}_{t\in\mathcal{T}}$; t is unavailable
CPT [409]	$\{\mathcal{D}_t^{pt}, t\}_{t\in\mathcal{T}^{pt}}$, followed by a downstream task j	$\{p(\mathcal{X}_t)\}_{t=j}$; t is not required

Table from Wang, Zhang, Su, Zhu. T-PAMI 2023

Related Learning Approaches

- Transfer Learning (finetuning)
  - Model trained on task 1; finetune it on task 2 (but use limited task 2 data)
  - *Not continual (just one step finetuning)*
  - *No retention/accumulation of knowledge*



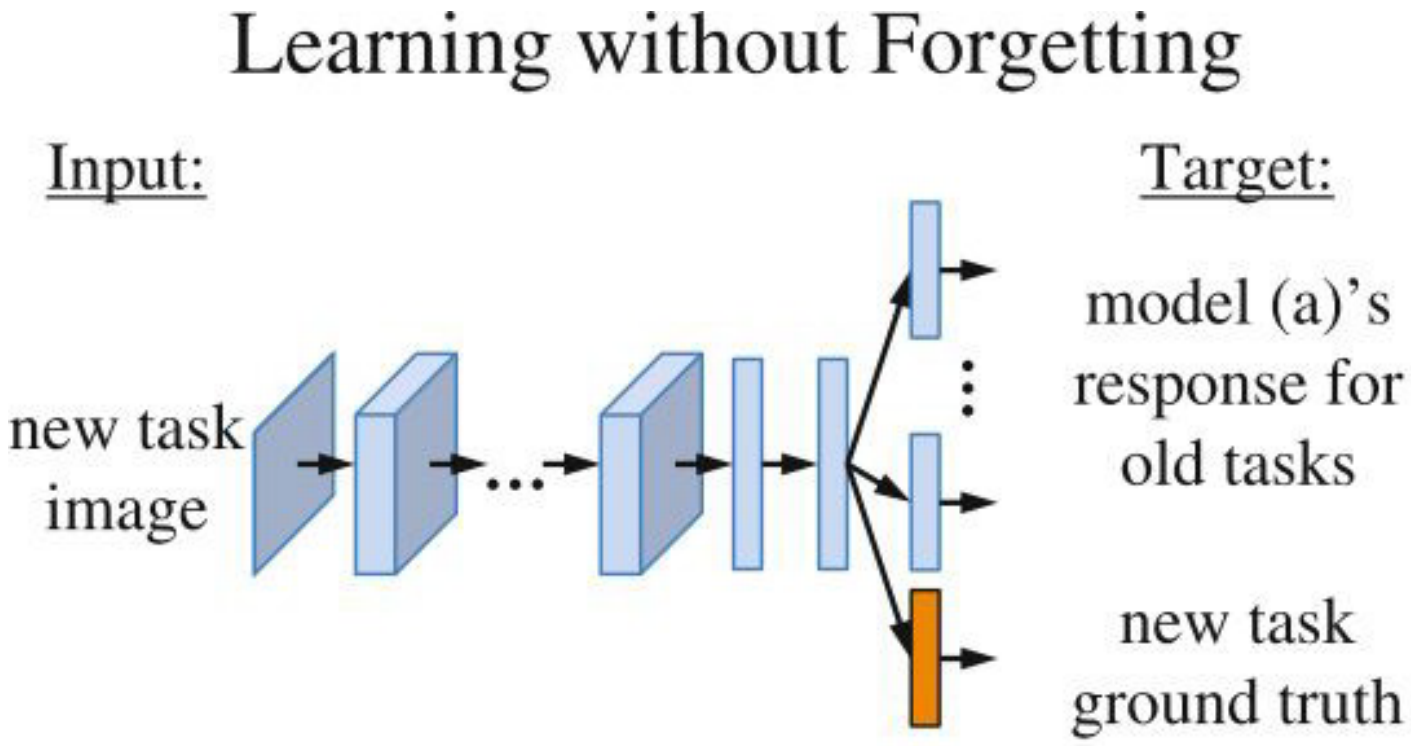
- Multi-task Learning
  - Jointly learn multiple related tasks simultaneously
  - *Not continual (simultaneous learning on all tasks)*

illustration from Sharon Li

Methods for Continual / Lifelong Learning

LwF: Learning without Forgetting

Only use examples from new task (no access to previous task data) and optimize for:
(1) high accuracy on new task (2) preservation of responses on old tasks



LwF: Learning without Forgetting

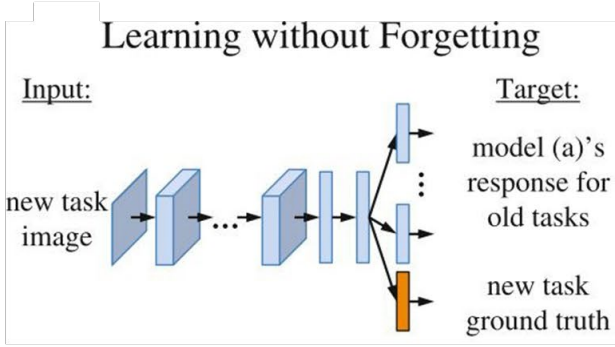
LEARNINGWITHOUTFORGETTING:

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

LEARNINGWITHOUTFORGETTING:

Start with:

θ_s: shared parameters

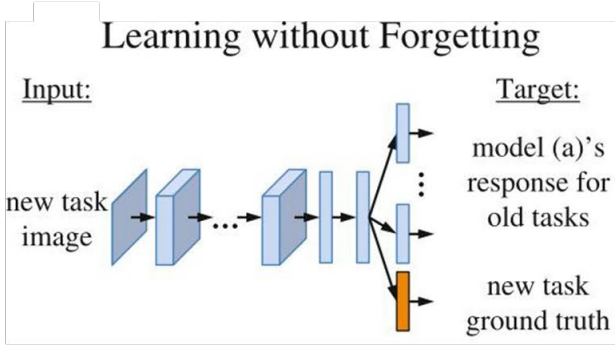
θ_o: task specific parameters for each old task

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

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:

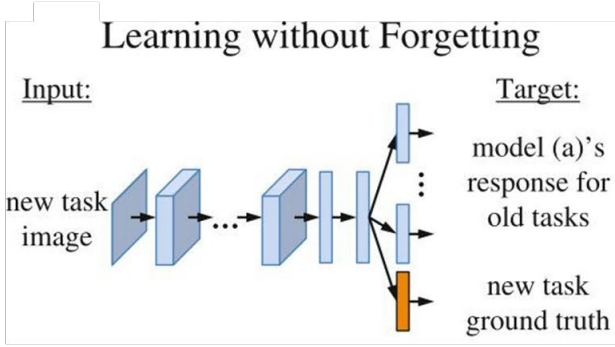
$Y_o \leftarrow \text{CNN}(X_n, \theta_s, \theta_o)$ // *compute output of old tasks for new data*

$\theta_n \leftarrow \text{RANDINIT}(|\theta_n|)$ // *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

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

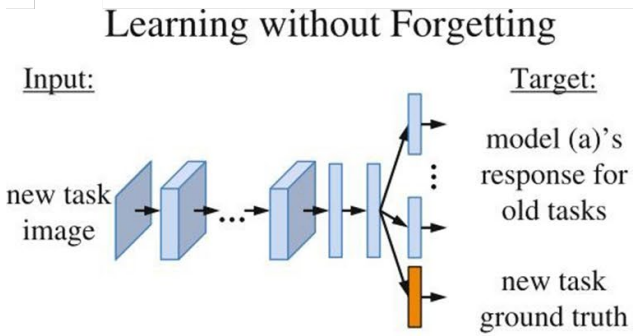
$$\mathcal{L}_{new}(\mathbf{y}_n, \hat{\mathbf{y}}_n) = -\mathbf{y}_n \cdot \log \hat{\mathbf{y}}_n$$

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

Distillation loss

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

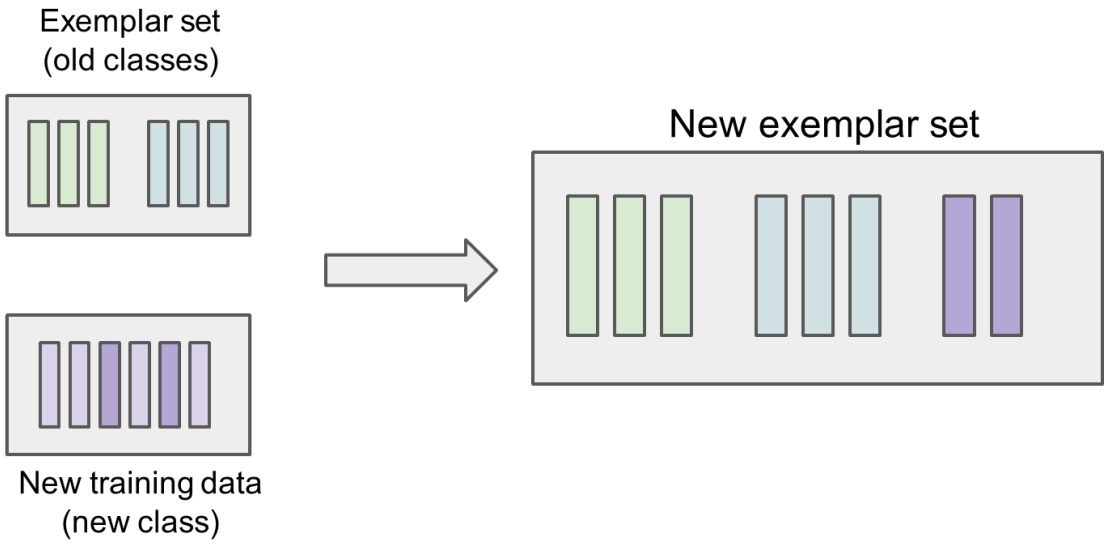
Weight decay of 0.0005



iCaRL: Incremental Classifier and Representation Learning

Add new classes, with restricted access to previous task data, to optimize for:
(1) high accuracy on new task (2) preservation of responses on old tasks

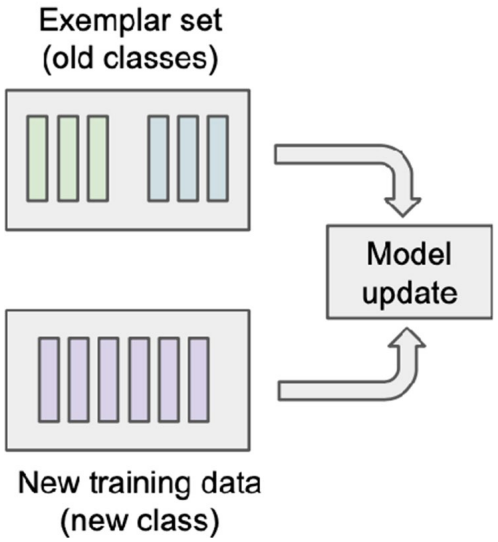
- A subset of previous task data is stored ("exemplars")
  - Size of exemplar set is constant
  - As new classes arrive, some exemplars are removed/replaced



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

Algorithm 3 iCaRL UpdateRepresentation

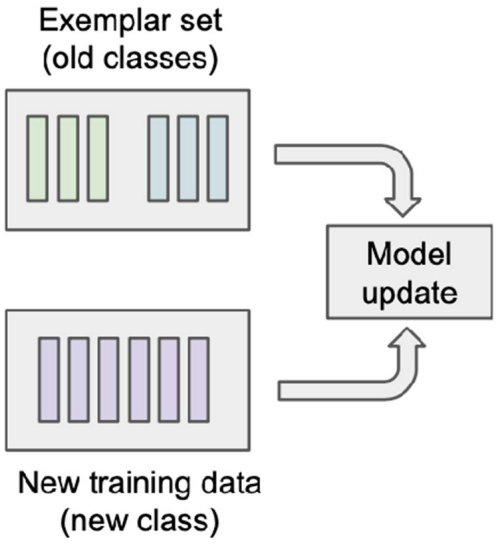
input $X^s, \ldots, X^t$ // training images of classes $s, \ldots, t$
require $\mathcal{P} = (P_1, \ldots, P_{s-1})$ // exemplar sets
require Θ // current model parameters
// form combined training set:

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

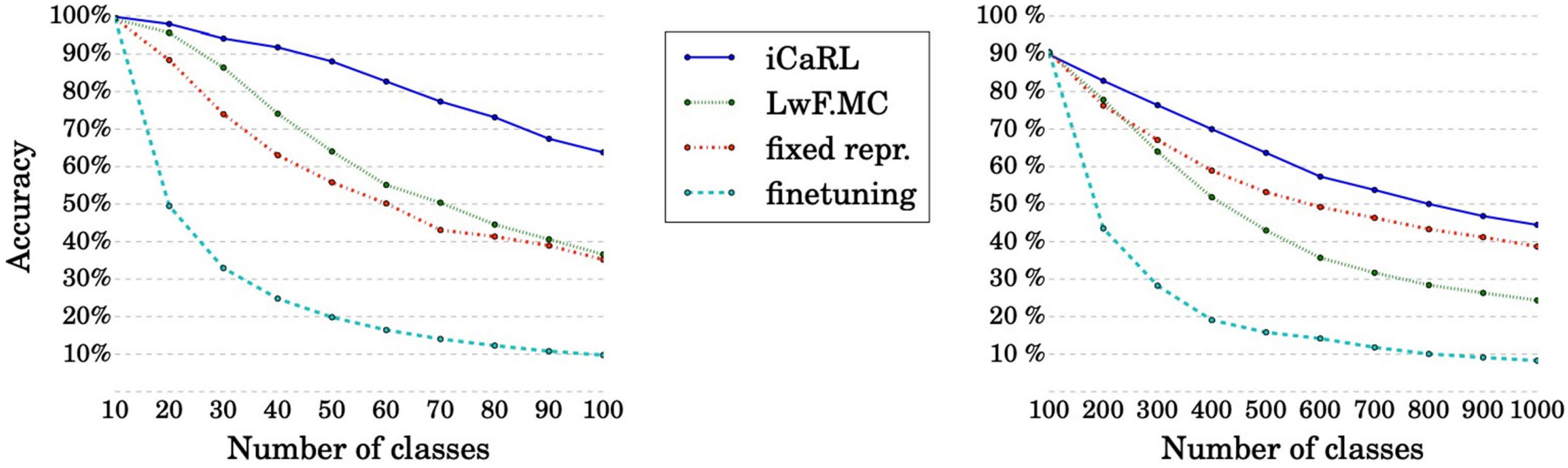
// store network outputs with pre-update parameters:
for $y = 1, \ldots, s-1$ **do**
$q_i^y \leftarrow g_y(x_i)$ for all $(x_i, \cdot) \in \mathcal{D}$
end for
run network training (*e.g.* BackProp) with loss function

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

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



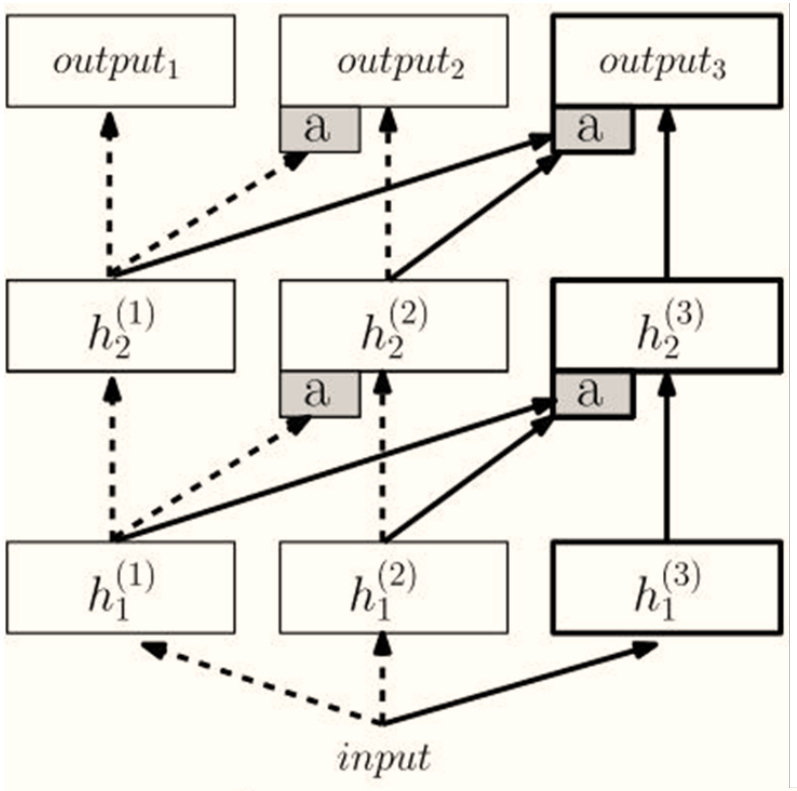
iCaRL vs LwF Results



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

Progressive Neural Networks

- Instantiate a new NN for each new task
  - Lateral connections to features of previously learned data
- Previous task data is NOT stored
  - NN weights implicitly encode knowledge of previous tasks
- Complexity of model grows with each task
- Task labels are needed at test time
  - To decide which subset of weights to use for prediction



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

Comparison of iCaRL vs LwF vs PNN

	Task labels needed?	Old training data needed?	Constant data size	Constant model complexity	Type
iCaRL	No	Yes	Yes	Yes	Class incremental
LFW	Yes	No	Yes	Yes	Task incremental
PNN	Yes	No	Yes	No (doubling per each new task)	Task incremental

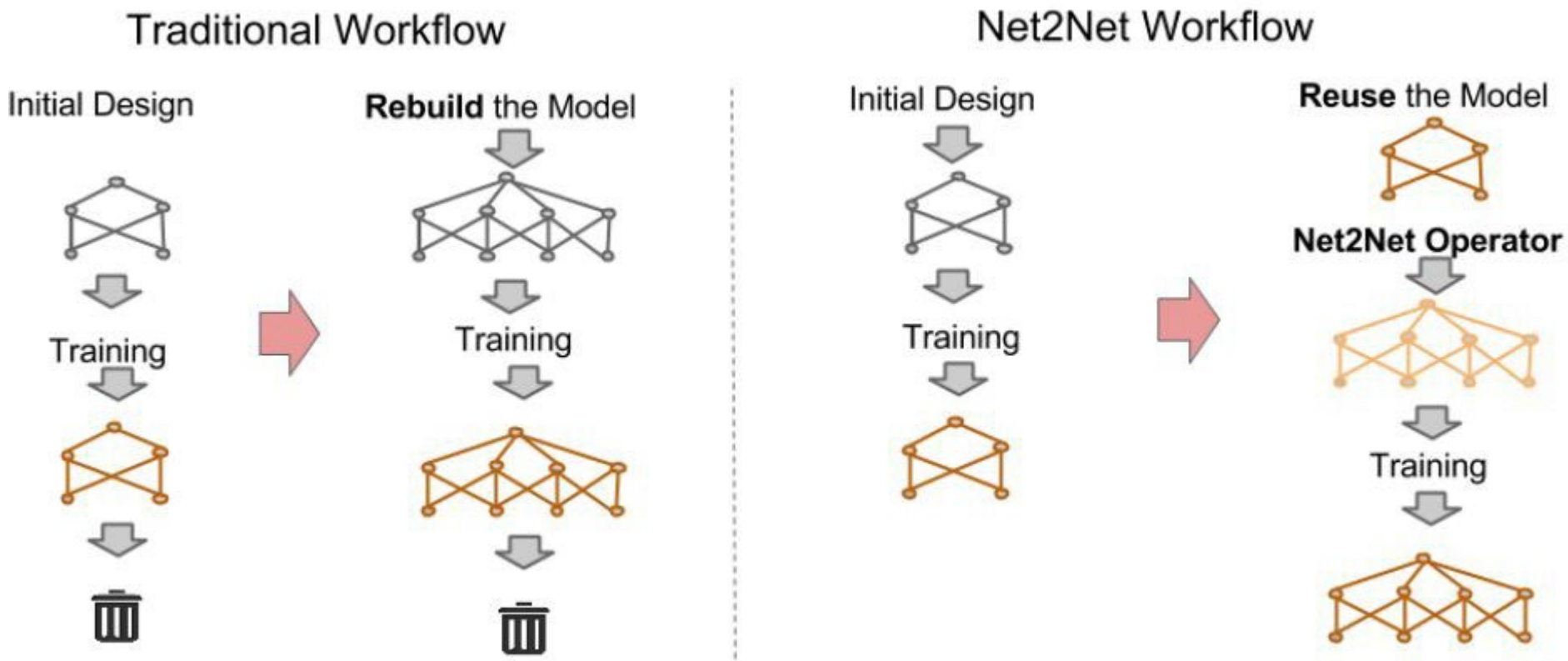
borrowed from Sharon Li

Thoughts on Increasing Model Complexity

- Lifelong/Continual Learning ~ progressively growing training set
- New knowledge acquired (new classes, new domains, ...) over time may saturate network capacity
  - As data complexity increases, model capacity (# parameters) should also increase?
  - Initially a small model could work (to prevent overfitting)
- If model capacity needs to be incremented, we need to avoid retraining the new (bigger) network from scratch every time.
  - The main idea behind "Student-Teacher" networks

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

Increasing Model Complexity: Net2Net



Resources

- Bing Liu's Webpage (Bing Liu is a pioneer in continual/open world ML)
  https://www.cs.uic.edu/~liub/lifelong-learning.html

- Book: https://link.springer.com/book/10.1007/978-3-031-01581-6
  (send me an email if you don't have PDF access)

- Podcast: https://worldofpiggy.com/podcast/2017/03/28/lifelong-machine-learning/
  (with one of the authors of the above book)