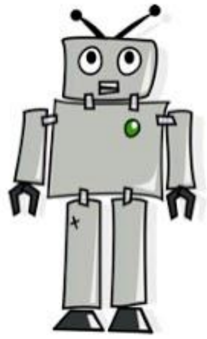
Lifelong Learning towards Machine Intelligence

Jianping Fan
Department of Computer Science
UNC-Charlotte

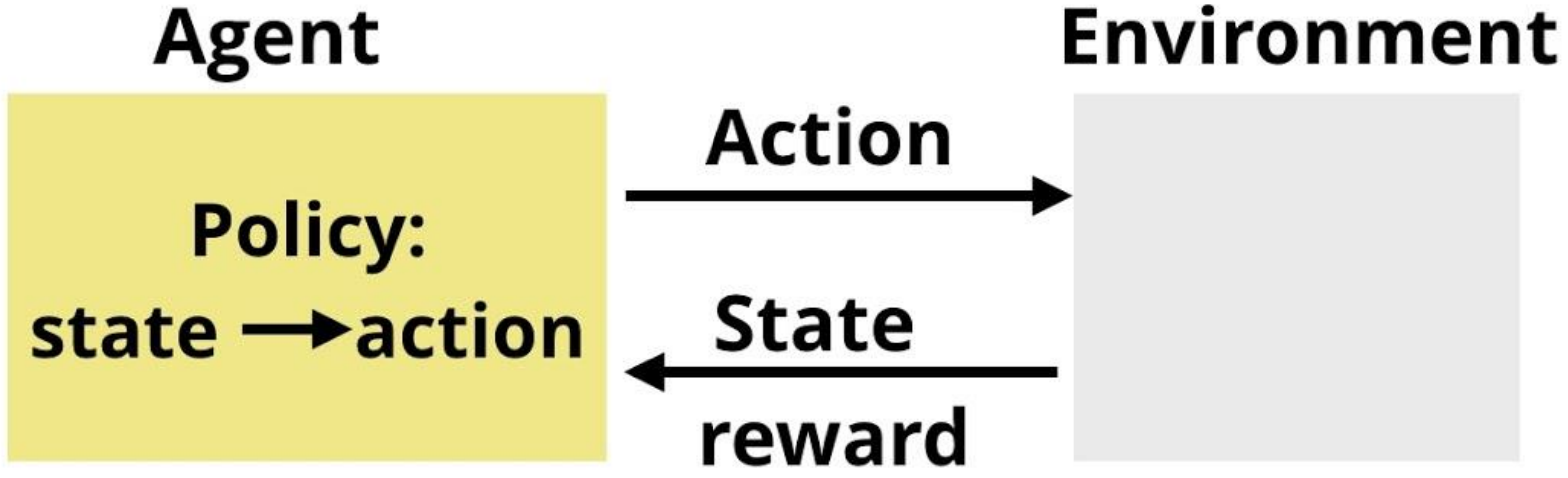
Course Website:
http://webpages.uncc.edu/jfan/itcs5152.html

Reinforcement Learning



Human: *Do you like Microsoft?*
ALICE: *I want to replace Windows with A I*
Human: *Did you chat with Jabberwacky?*
ALICE: *No I didn't chat with Jabberwacky.*
Human: *It is another chat bot*
ALICE: *Really. It is?*
Human: *The news says that you did chat Jabber*
ALICE: *Don't believe everything you see on TV.*
You say:

Get ALICE Silver Edition with talking VHost[tm], winner of the 2004 Loebner Prize



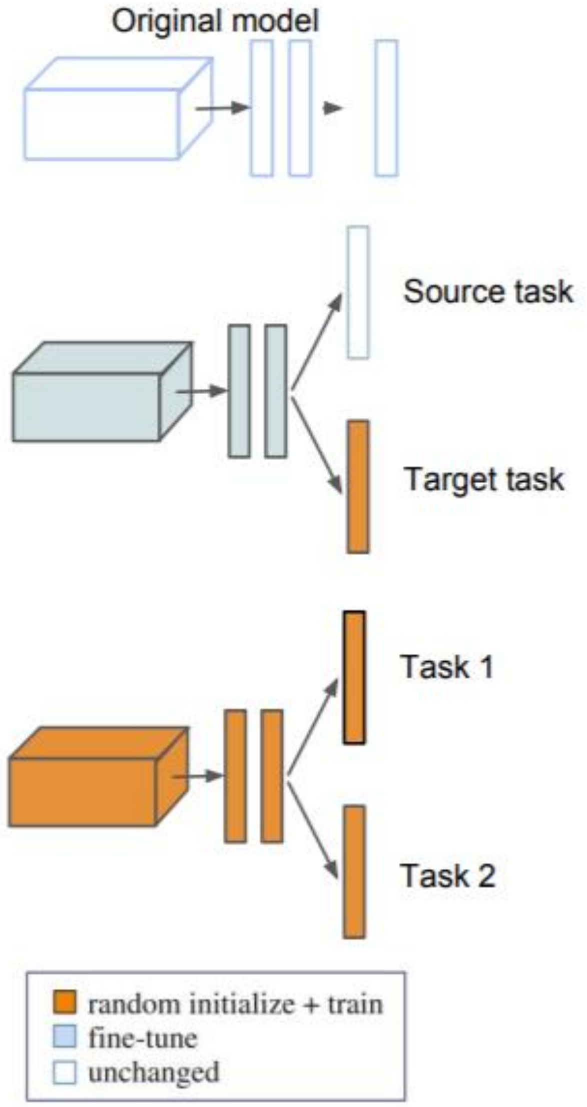
Related learning approaches

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



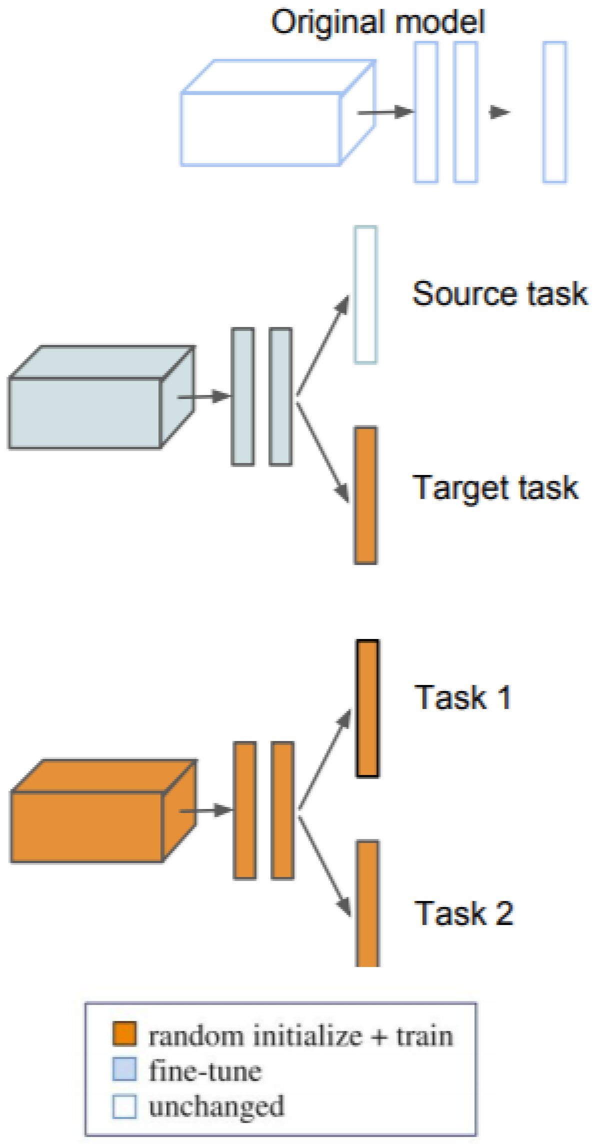
Related learning approaches

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



Biological Aspects of Lifelong Learning

(a) The Stability-Plasticity Dilemma

As humans, we have an astonishing ability to adapt by effectively acquiring knowledge and skills, refining them on the basis of novel experiences, and transferring them across multiple domains. While it is true that we tend to gradually forget previously learned information throughout our lifespan, only rarely does the learning of novel information catastrophically interfere with consolidated knowledge.

Lifelong learning in the brain is mediated by a rich set of neurophysiological principles that regulate the stability-plasticity balance of the different brain areas and that contribute to the development and specialization of our cognitive system on the basis of our sensorimotor experiences.

The stabilityplasticity dilemma regards the extent to which a system must be prone to integrate and adapt to new knowledge and, importantly, how this adaptation process should be compensated by internal mechanisms that stabilize and modulate neural activity to prevent catastrophic forgetting .

Stability -plasticity dilemma: When and how to adapt to the current model

(a) Quick update enables rapid adaptation, but old information is forgotten;
(b) Slower adaptation allows to retain old information but the reactivity of the system is decreased;
(c) Failure to deal with this dilemma may lead to catastrophic forgetting
(d) Data streams, constantly arriving, not static → Incremental learning
(e) Multiple tasks with multiple learning/mining algorithms
(f) Retain/accumulate learned knowledge in the past & use it to help future learning: Use past knowledge for inductive transfer when learning new tasks
(g) Mimics human way of learning

Biological Aspects of Lifelong Learning

(b) Hebbian Plasticity and Stability

The ability of the brain to adapt to changes in its environment provides vital insight into how connectivity and function of the cortex are shaped. It has been shown that while rudimentary patterns of connectivity in the visual system are established in early development, normal visual input is required for the correct development of the visual cortex.

The most well-known theory describing the mechanisms of synaptic plasticity for the adaptation of neurons to external stimuli was first proposed by Hebb, postulating that when one neuron drives the activity of another neuron, the connection between them is strengthened. More specifically, the Hebb's rule states that the repeated and persistent stimulation of the postsynaptic cell from the presynaptic cell leads to an increased synaptic efficacy. Throughout the process of development, neural systems stabilize to shape optimal functional patterns of neural connectivity.

1. Learning without Forgetting (Regularization Approaches)

Kirkpatrick et al. proposed the elastic weight consolidation (EWC) model in supervised and reinforcement learning scenarios. The approach consists of a quadratic penalty on the difference between the parameters for the old and the new tasks that slows down the learning for task-relevant weights coding for previously learned knowledge. The relevance of the parameter θ with respect to a task's training data D is modelled as the posterior distribution p(θ | D). Assuming a scenario with two independent tasks A with DA and B with DB, the log value of the posterior probability given by the Bayes' rule is:

$$\log p(\theta \mid \mathcal{D}) = \log p(\mathcal{D}_B \mid \theta) + \log p(\theta \mid \mathcal{D}_A) - \log p(\mathcal{D}_B),$$

where the posterior probability logp(θ | DA) embeds all the information about the previous task. However, since this term is intractable, EWC approximates it as a Gaussian distribution with mean given by the parameters θ ∗ A and a diagonal precision given by the diagonal of the fisher information matrix F. Therefore, the loss function of EWC is given by

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

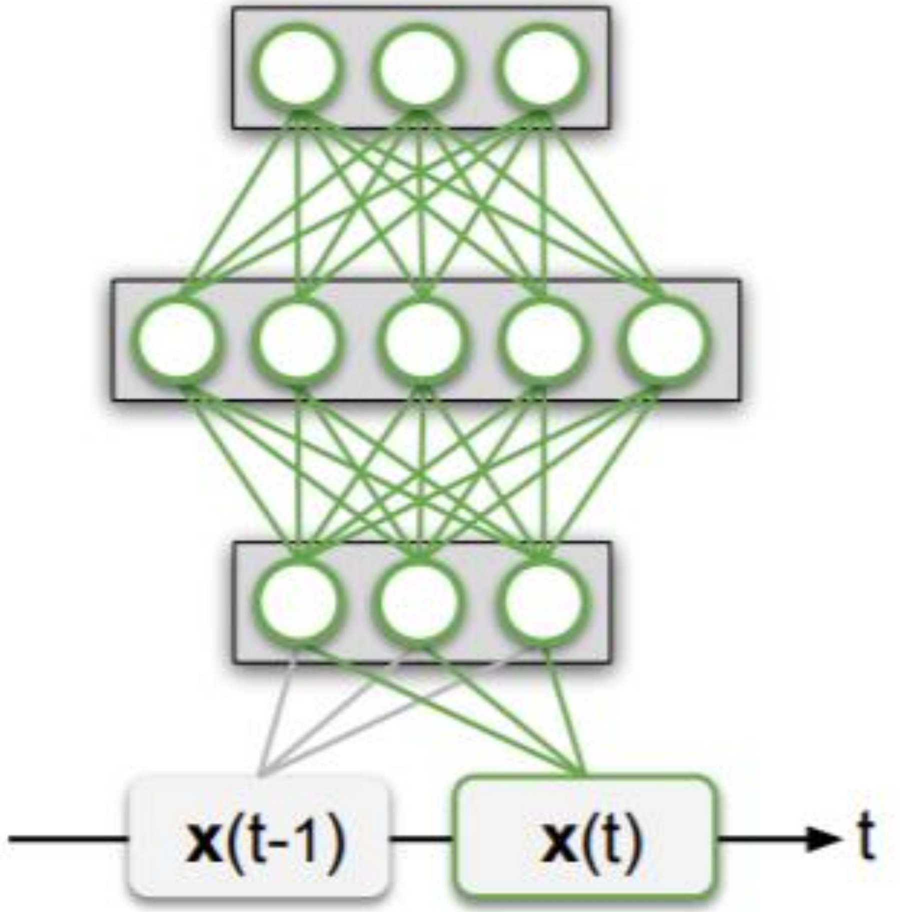
1. Learning without Forgetting (Regularization Approaches)

Zenke, Poole & Ganguli (2017) proposed to alleviate catastrophic forgetting by allowing individual synapses to estimate their importance for solving a learned task. Similar to Kirkpatrick et al. (2017), this approach penalizes changes to the most relevant synapses so that new tasks can be learned with minimal forgetting. To reduce large changes in important parameters θ_k when learning a new task, the authors use a modified cost function $\mathcal{L}_n^*$ with a surrogate loss which approximates the summed loss functions of all previous tasks $\mathcal{L}_o^*$:

$$\mathcal{L}_n^* = \mathcal{L}_n + c \sum_k \Omega_k^n (\theta_k^* - \theta_k)^2,$$

where c is a weighting parameter to balance new and old tasks, θ_k^* are the parameters at the end of the previous task, and Ω_k^n is a per-parameter regulation strength.

1. Learning without Forgetting (Regularization Approaches)



a) Retraining with regularization

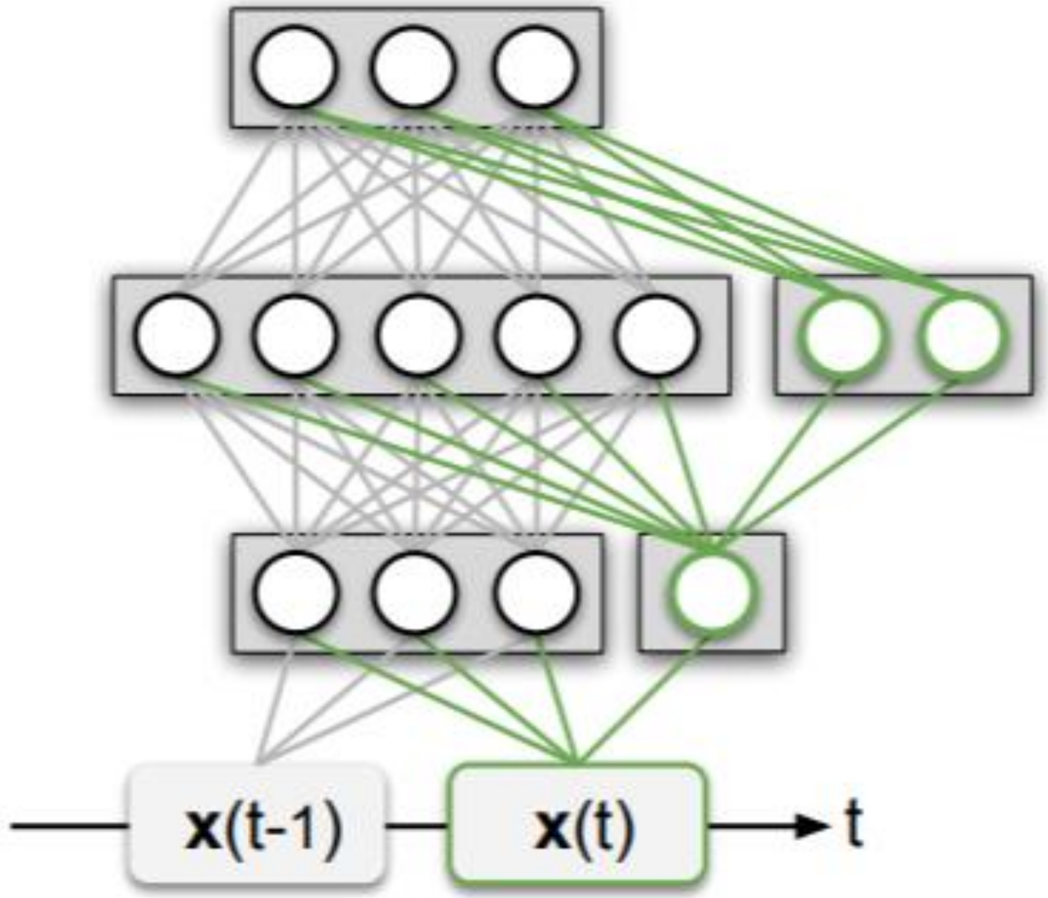
2. Dynamic Architectures

The approaches change architectural properties in response to new information by dynamically accommodating novel neural resources, e.g., re-training with an increased number of neurons or network layers.

Rusu et al. proposed to block any changes to the network trained on previous knowledge and expand the architecture by allocating novel sub-networks with fixed capacity to be trained with the new information.

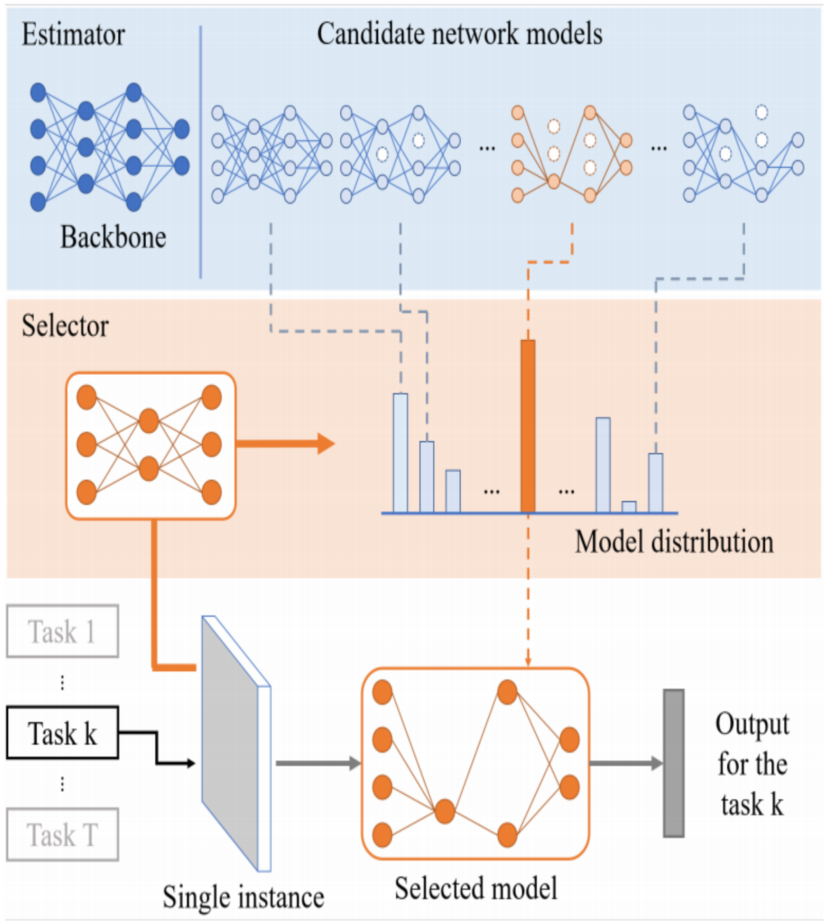
Zhou et al. proposed the incremental training of a denoising autoencoder that adds neurons for samples with high loss and subsequently merges these neurons with existing ones to prevent redundancy. More specifically, the algorithm is composed of two processes for (i) adding new features to minimize the residual of the objective function and (ii) merging similar features to obtain a compact feature representation and in this way prevent overfitting.

2. Dynamic Architectures

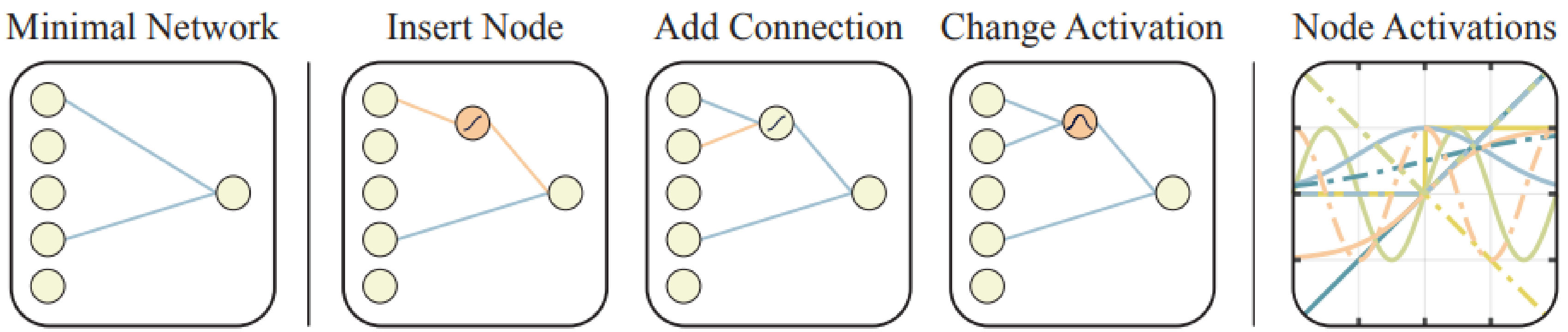


b) Training with network expansion

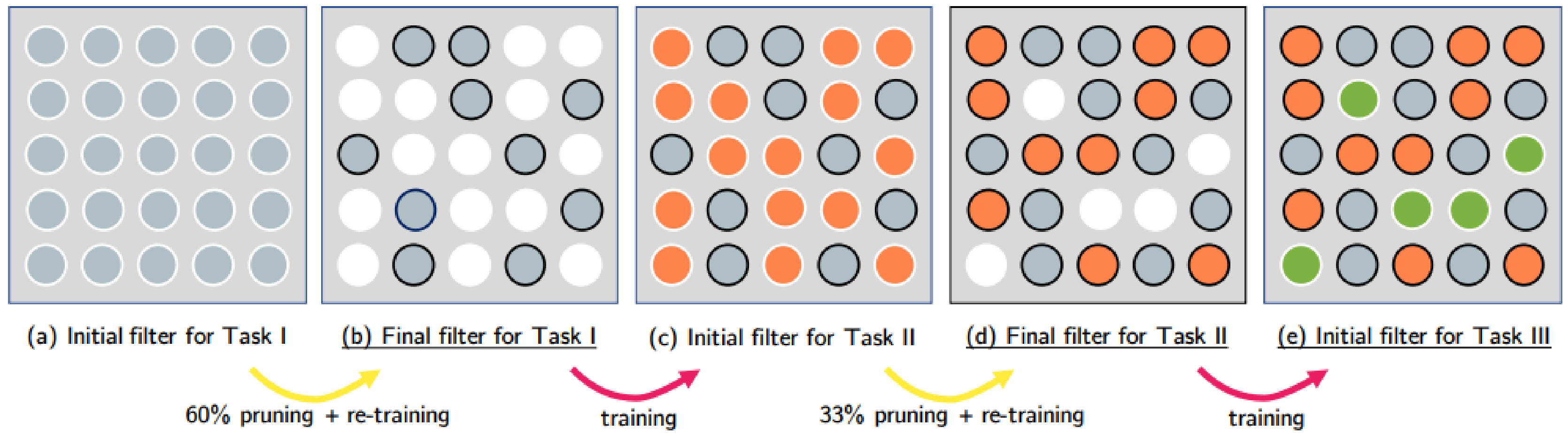
Model Integration



Deep Elastic Networks with Model Selection for Multi-Task Learning



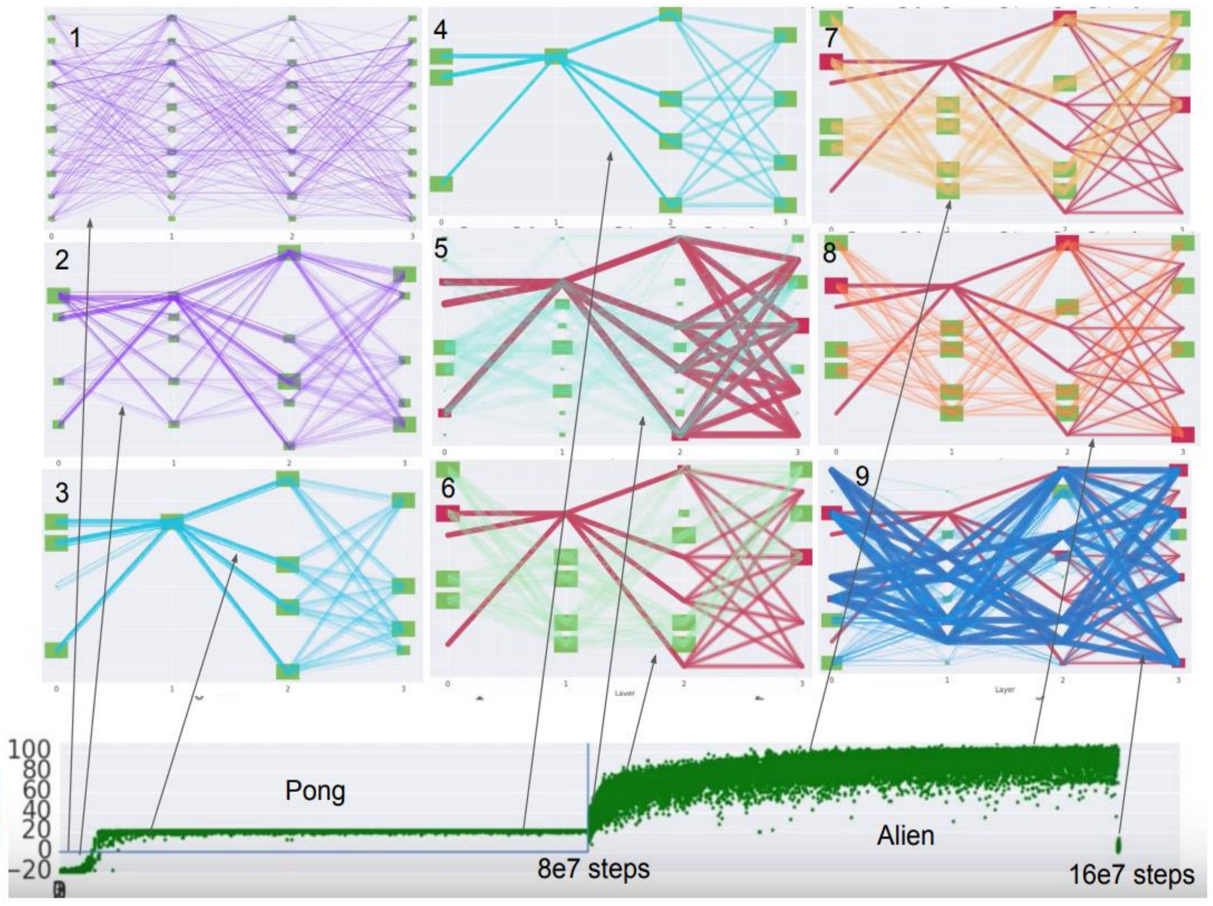
Weight Agnostic Neural Networks


(a) Initial filter for Task I
(b) Final filter for Task I
(c) Initial filter for Task II
(d) Final filter for Task II
(e) Initial filter for Task III
60% pruning + re-training
training
33% pruning + re-training
training

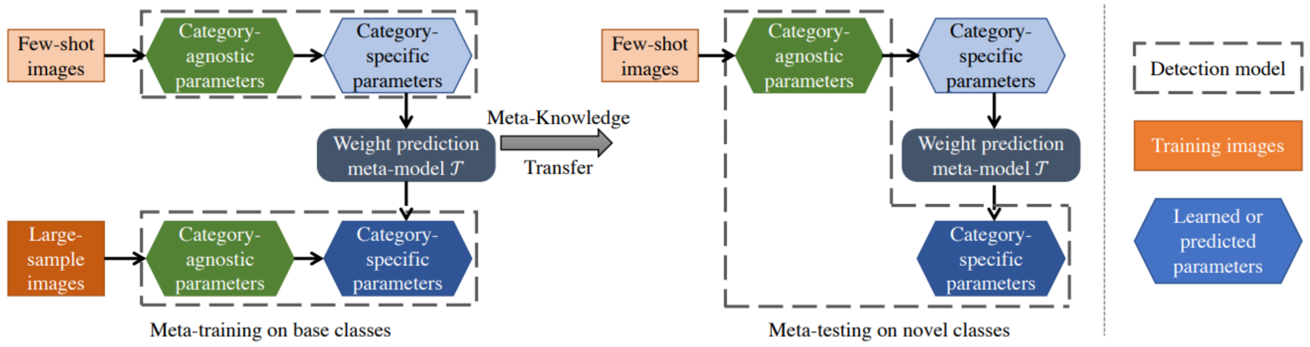
PackNet: Adding Multiple Tasks to a Single Network by Iterative Prun

How to integrate old network model for old tasks with new network model for new task?

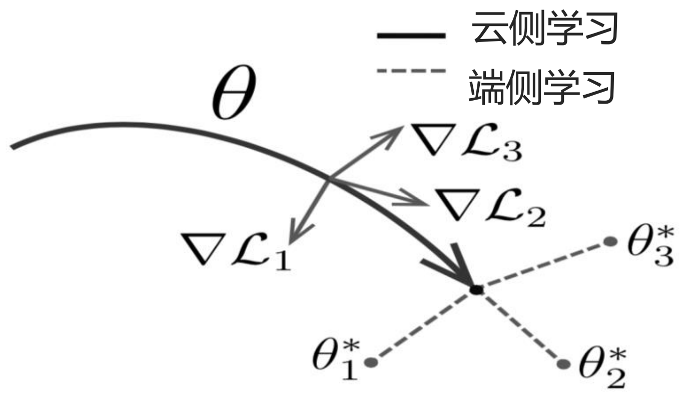
Model Integration



PathNet: Evolution Channels Gradient Descent in Super Neural Networks


Few-shot images
Category-agnostic parameters
Category-specific parameters
Weight prediction meta-model $\mathcal{T}$
Meta-Knowledge
Transfer
Large-sample images
Meta-training on base classes
Meta-testing on novel classes
Detection model
Training images
Learned or predicted parameters

Meta-Learning to Detect Rare Objects

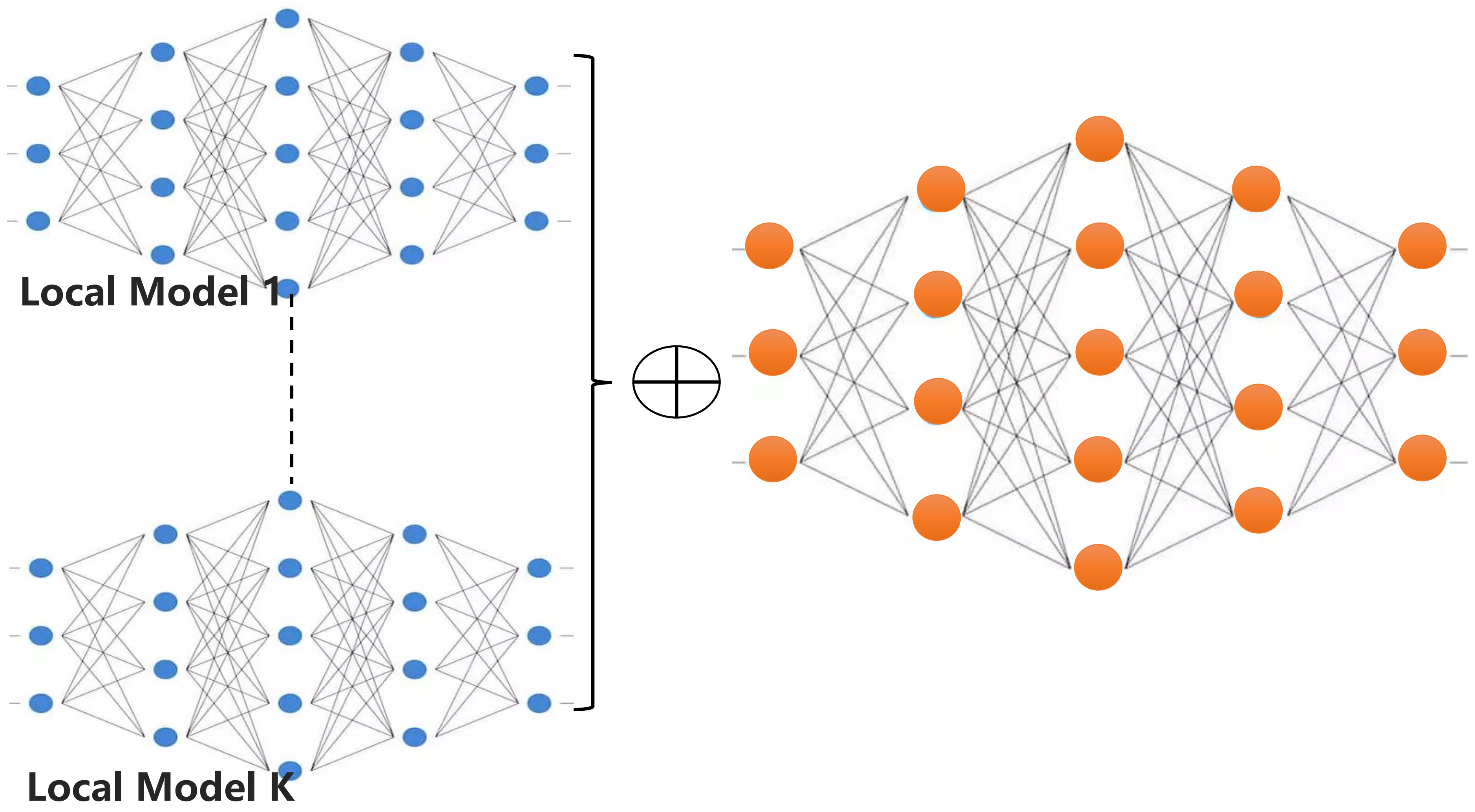


Model-Agnostic Meta-Learning for Fast Adaptation of Deep Net

How to integrate old network model for old tasks with new network model for new task?

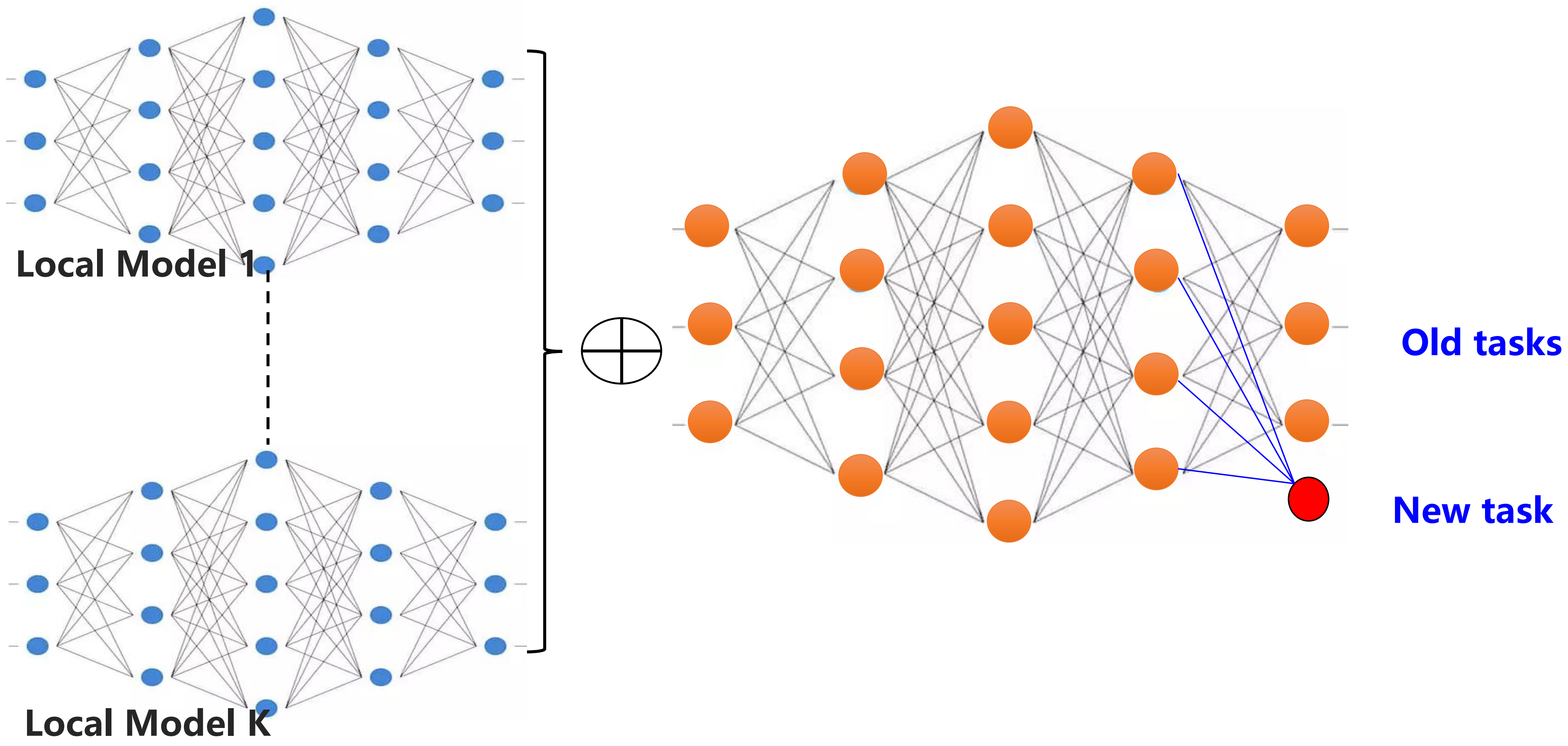
3. Model Integration

- Network Fusion when only few new samples (Concept drift) are added



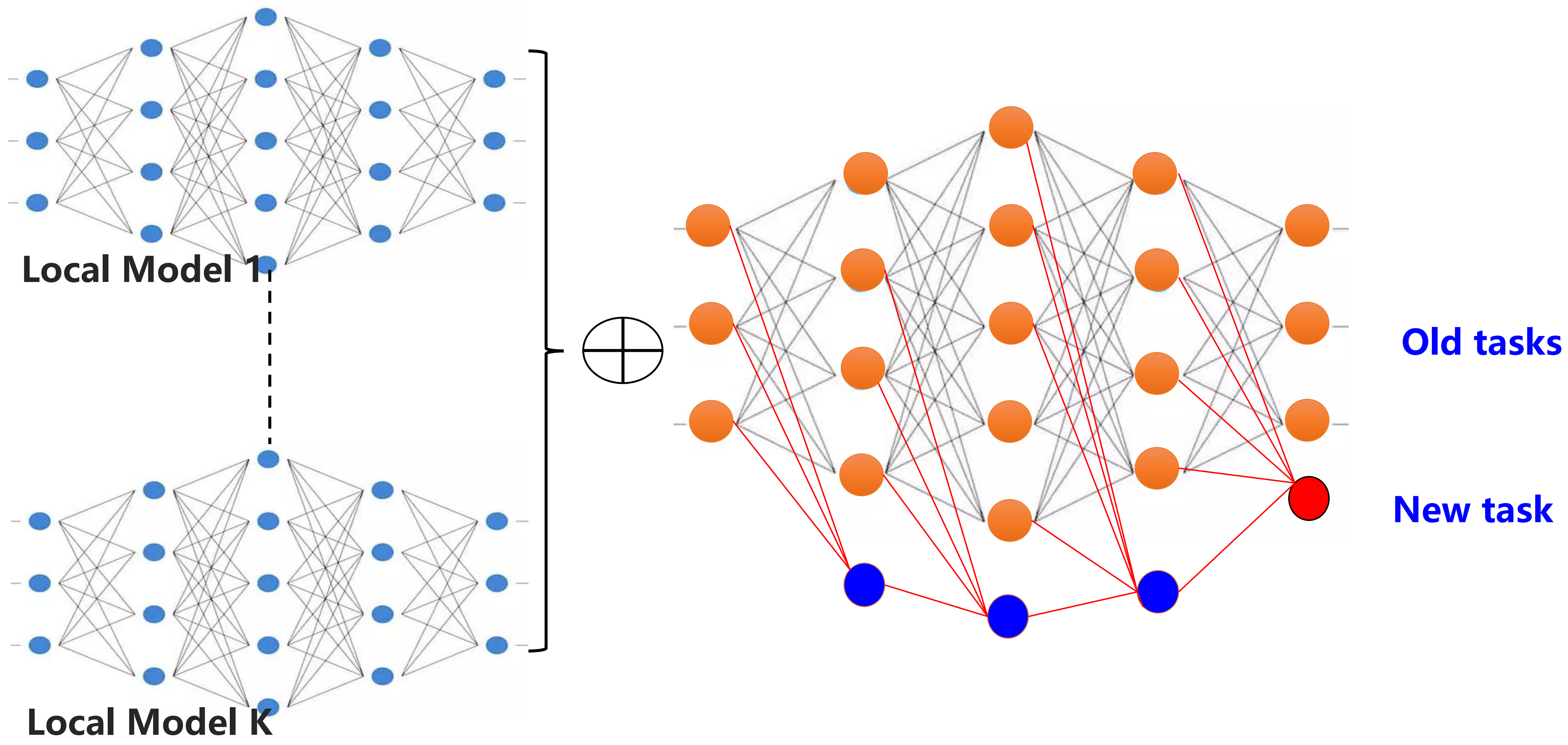
3. Model Integration

- Network Fusion when only few new but similar tasks are added



3. Model Integration

- Network Fusion & Pruning when few new but different tasks are added



LWF: Learning without Forgetting [Li2016]

Goal:

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

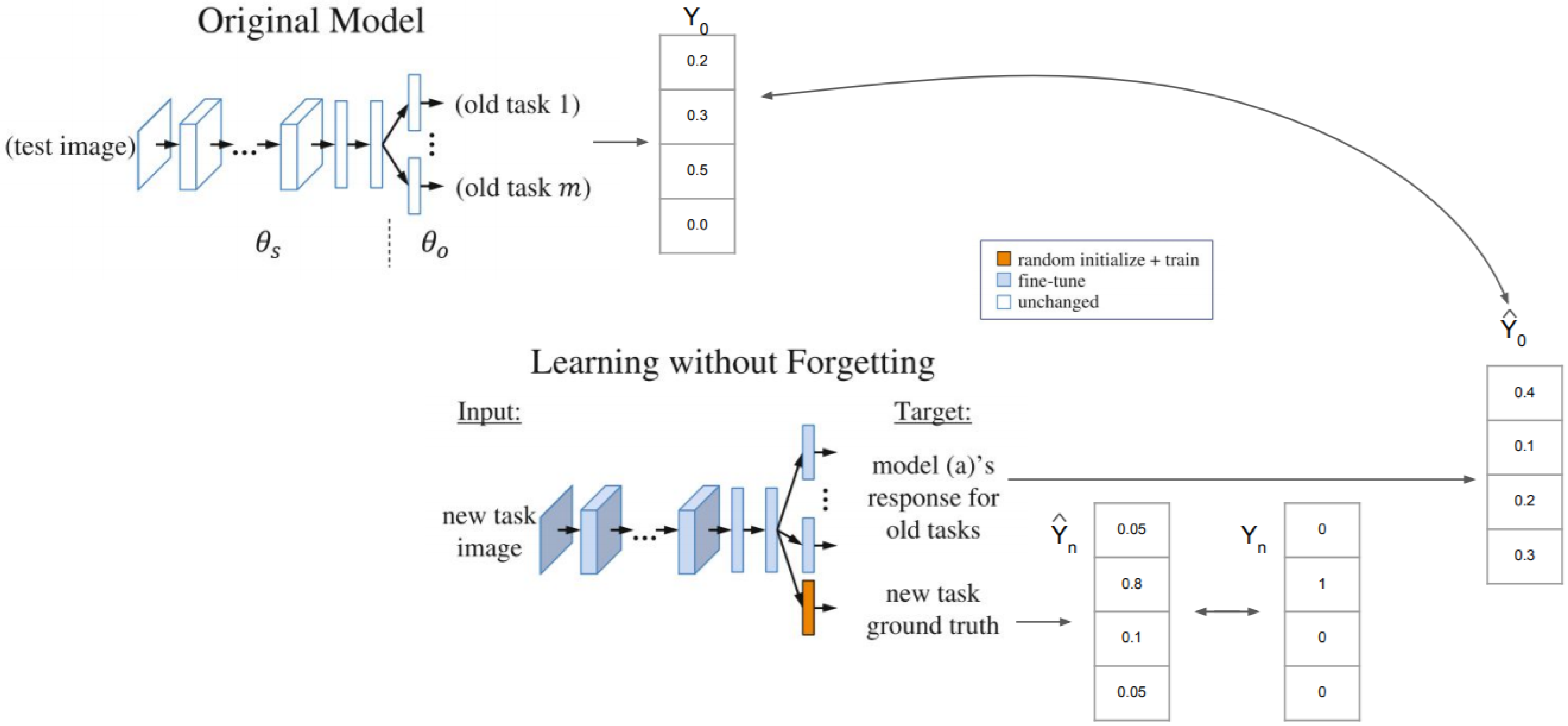
Solution:

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]

```
LEARNINGWITHOUTFORGETTING:
  Start with:
      $\theta_s$: shared parameters
      $\theta_o$: task specific parameters for each old task
      $X_n$, $Y_n$: training data and ground truth on the new task
  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
      $\theta_s^*, \theta_o^*, \theta_n^* \leftarrow \underset{\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n}{\operatorname{argmin}} \left( \mathcal{L}_{old}(Y_o, \hat{Y}_o) + \mathcal{L}_{new}(Y_n, \hat{Y}_n) + \mathcal{R}(\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n) \right)$
```

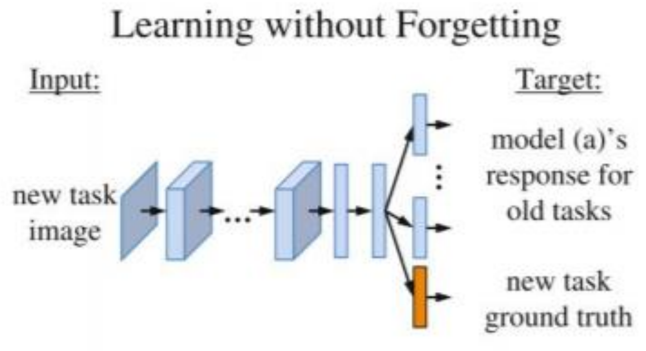
Multinomial logistic loss

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

Weight decay of 0.0005

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

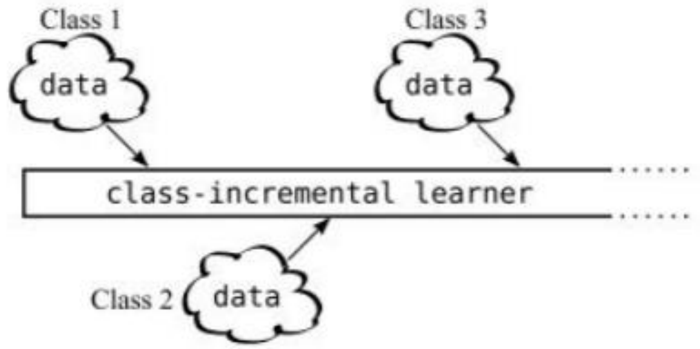
Distillation loss


Learning without Forgetting
Input:
Target:
new task image
model (a)'s response for old tasks
new task ground truth

iCaRL

Goal:

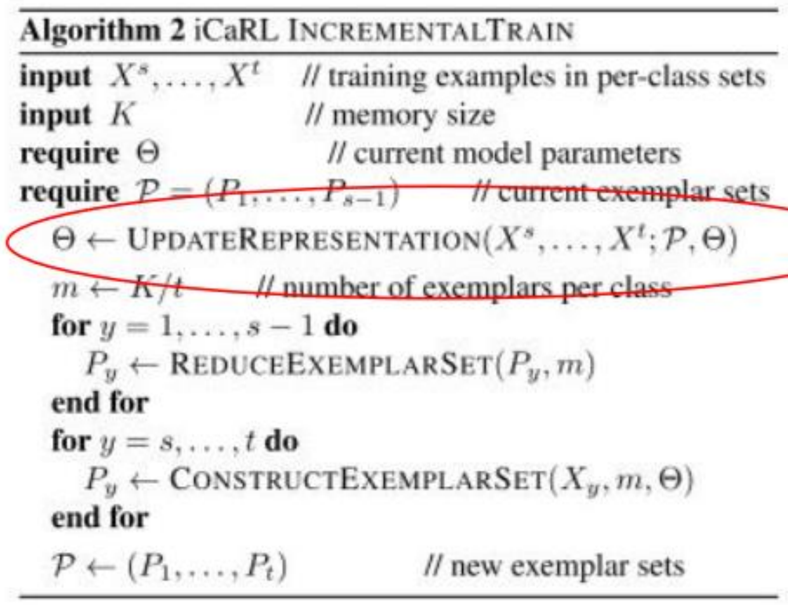
Add new classes based on adapting shared parameters **with restricted access** to training data for previously learned classes.



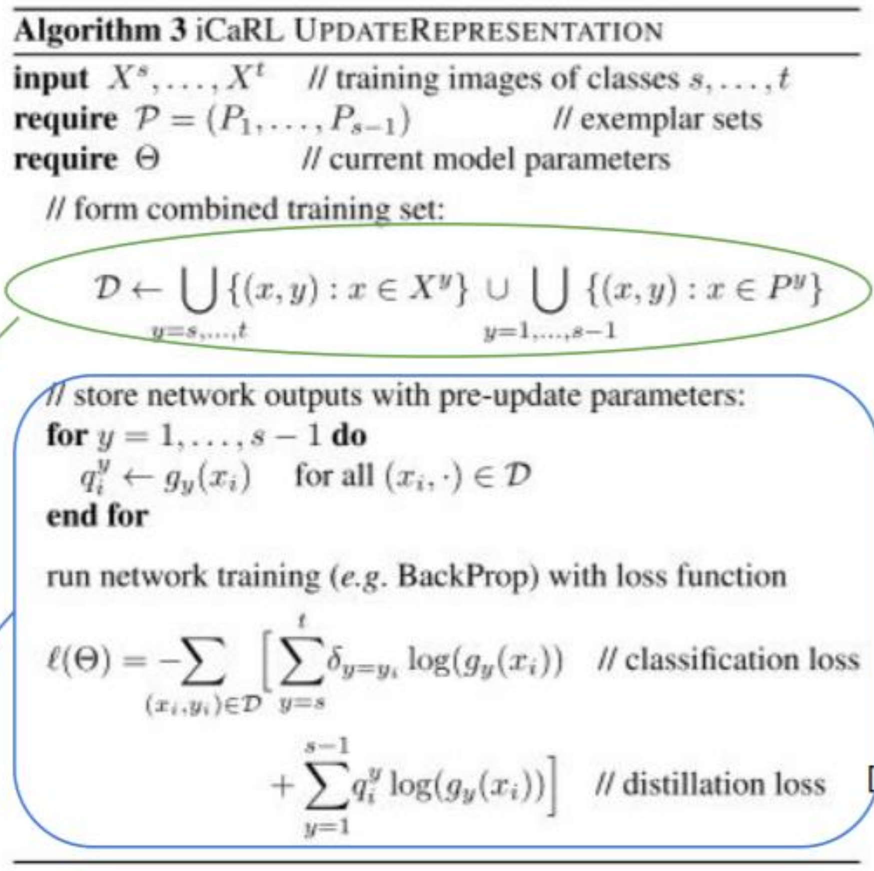
Solution:

- 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

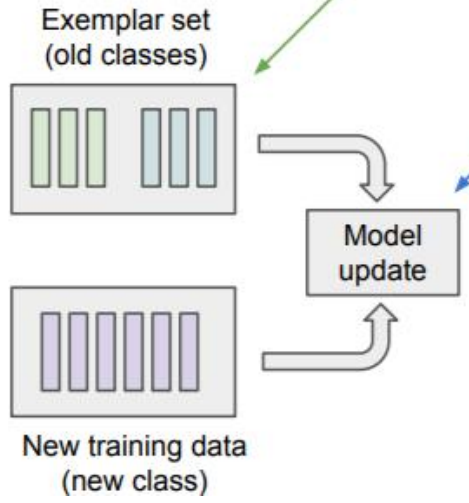


```
Algorithm 2 iCaRL INCREMENTALTRAIN
input $X^s, \dots, X^t$    // training examples in per-class sets
input $K$    // memory size
require $\Theta$    // current model parameters
require $\mathcal{P} = (P_1, \dots, P_{s-1})$    // current exemplar sets
  $\Theta \leftarrow$ UPDATEREPRESENTATION($X^s, \dots, X^t; \mathcal{P}, \Theta$)
  $m \leftarrow K/t$    // number of exemplars per class
  for $y = 1, \dots, s-1$ do
    $P_y \leftarrow$ REDUCEEXEMPLARSET($P_y, m$)
  end for
  for $y = s, \dots, t$ do
    $P_y \leftarrow$ CONSTRUCTEXEMPLARSET($X_y, m, \Theta$)
  end for
  $\mathcal{P} \leftarrow (P_1, \dots, P_t)$    // new exemplar sets
```

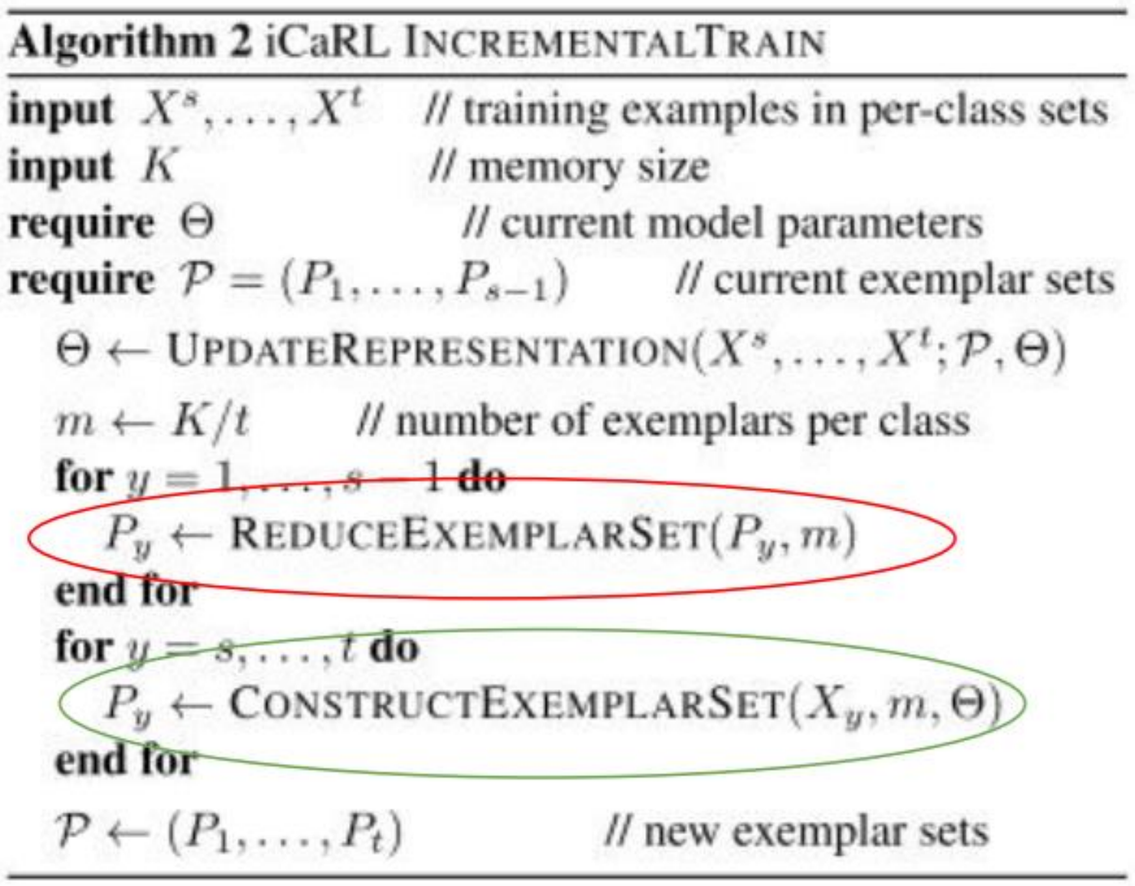


```
Algorithm 3 iCaRL UPDATEREPRESENTATION
input $X^s, \dots, X^t$    // training images of classes $s, \dots, t$
require $\mathcal{P} = (P_1, \dots, P_{s-1})$    // exemplar sets
require $\Theta$    // current model parameters
  // form combined training set:
  $\mathcal{D} \leftarrow \bigcup_{y=s,\dots,t} \{(x,y) : x \in X^y\} \cup \bigcup_{y=1,\dots,s-1} \{(x,y) : x \in P^y\}$
  // 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
  $\ell(\Theta) = -\sum_{(x_i,y_i)\in\mathcal{D}} \Big[ \sum_{y=s}^{t} \delta_{y=y_i} \log(g_y(x_i))$    // classification loss
      $+ \sum_{y=1}^{s-1} q_i^y \log(g_y(x_i)) \Big]$    // distillation loss
```

[Hinton2015]

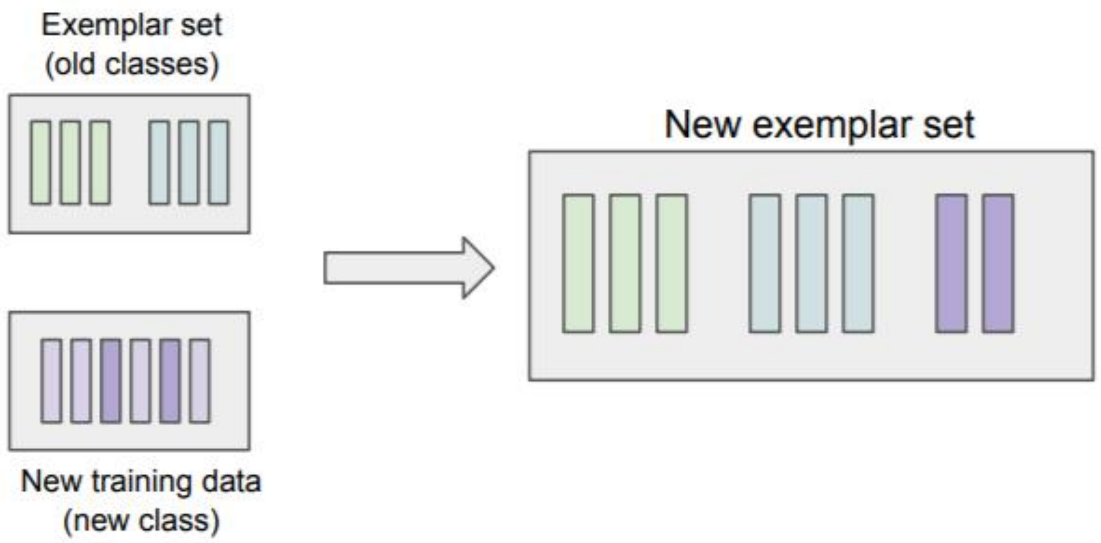


iCaRL: Incremental Classifier and Representation learning



Algorithm 2 iCaRL IncrementalTrain

```
input $X^s, \dots, X^t$   // training examples in per-class sets
input $K$   // memory size
require $\Theta$   // current model parameters
require $\mathcal{P} = (P_1, \dots, P_{s-1})$   // current exemplar sets
  $\Theta \leftarrow$ 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
    $P_y \leftarrow$ ReduceExemplarSet($P_y, m$)
  end for
  for $y = s, \dots, t$ do
    $P_y \leftarrow$ ConstructExemplarSet($X_y, m, \Theta$)
  end for
  $\mathcal{P} \leftarrow (P_1, \dots, P_t)$   // new exemplar sets
```



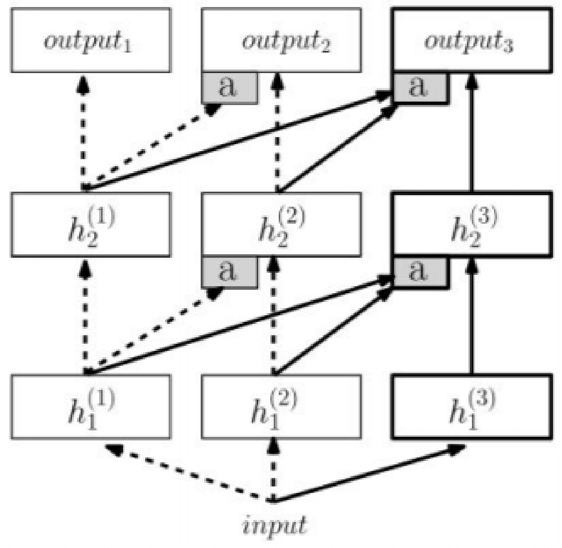
Progressive Neural Networks

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



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

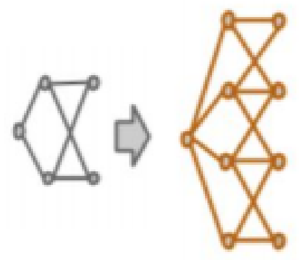
Increasing model capacity (I)

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.

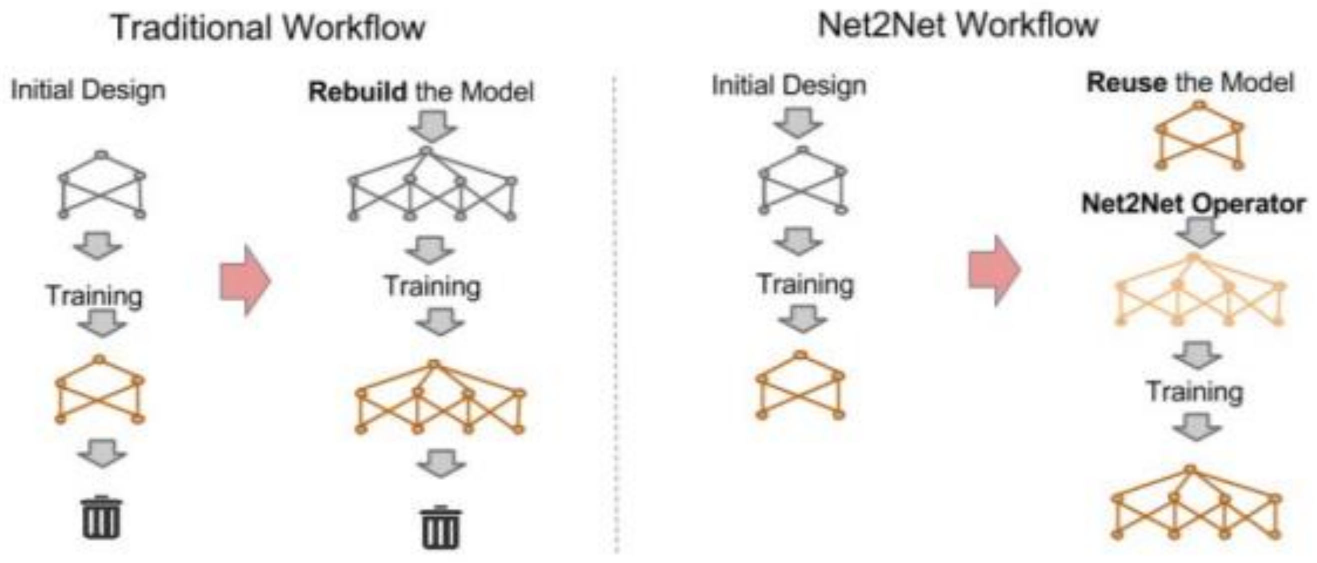
If the capacity of the network has to be incremented we want to avoid retraining the new network from scratch

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)

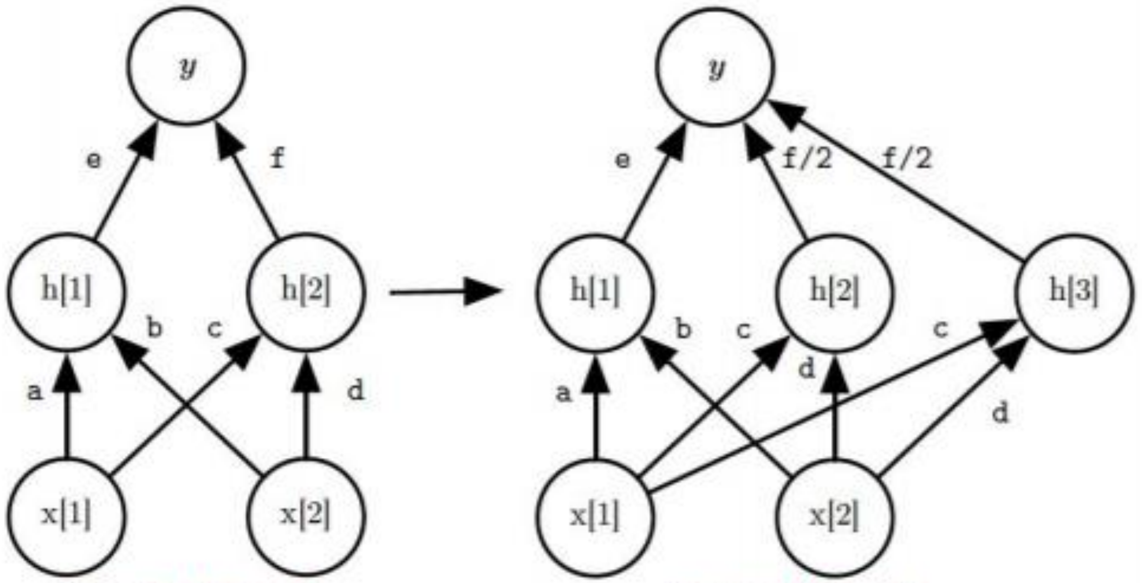
Net2WiderNet:

- 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

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

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

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

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.
```

$$\forall x, f(x;\theta) = g(x;\theta')$$
$$g : \{1, 2, \cdots, q\} \rightarrow \{1, 2, \cdots, n\} \quad q > n$$
$$g(j) = \begin{cases} j & j \leq n \\ \text{random sample from } \{1, 2, \cdots, n\} & j > n \end{cases}$$

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

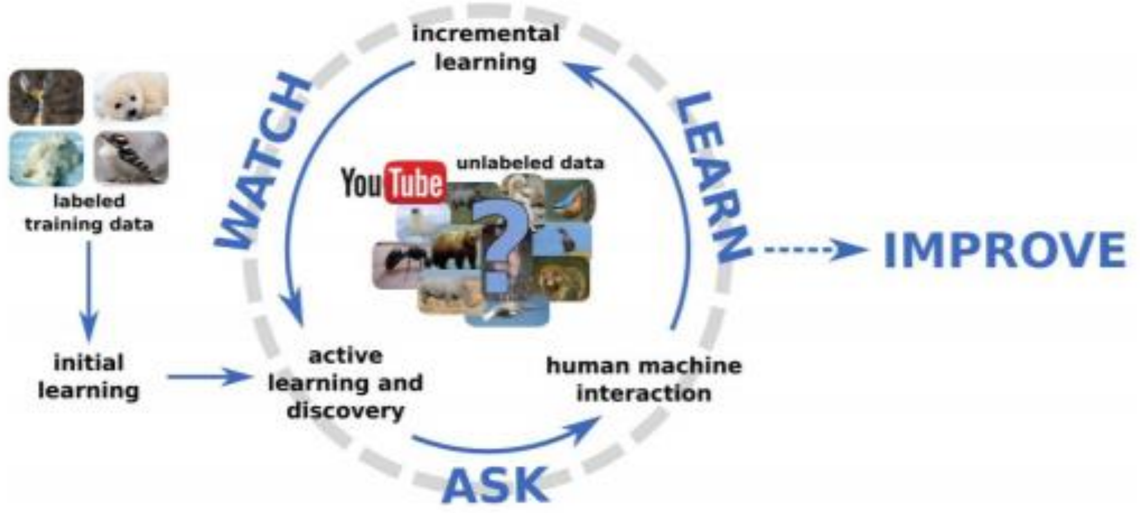
Chen, T., Goodfellow, I., & Shlens, J. (2016). Net2Net: Accelerating Learning via Knowledge Transfer. In *ICLR 2016*

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



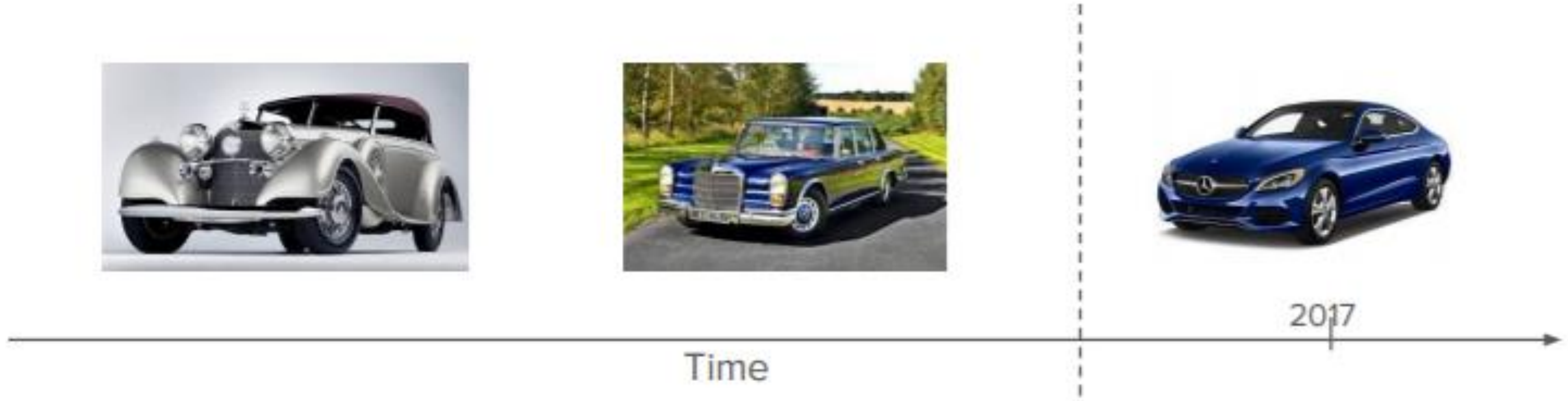
Käding, C., Rodner, E., Freytag, A., & Denzler, J. (2016). Watch, Ask, Learn, and Improve: a lifelong learning cycle for visual recognition. *European Symposium on Artificial NN.*

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