南京航空航天大學

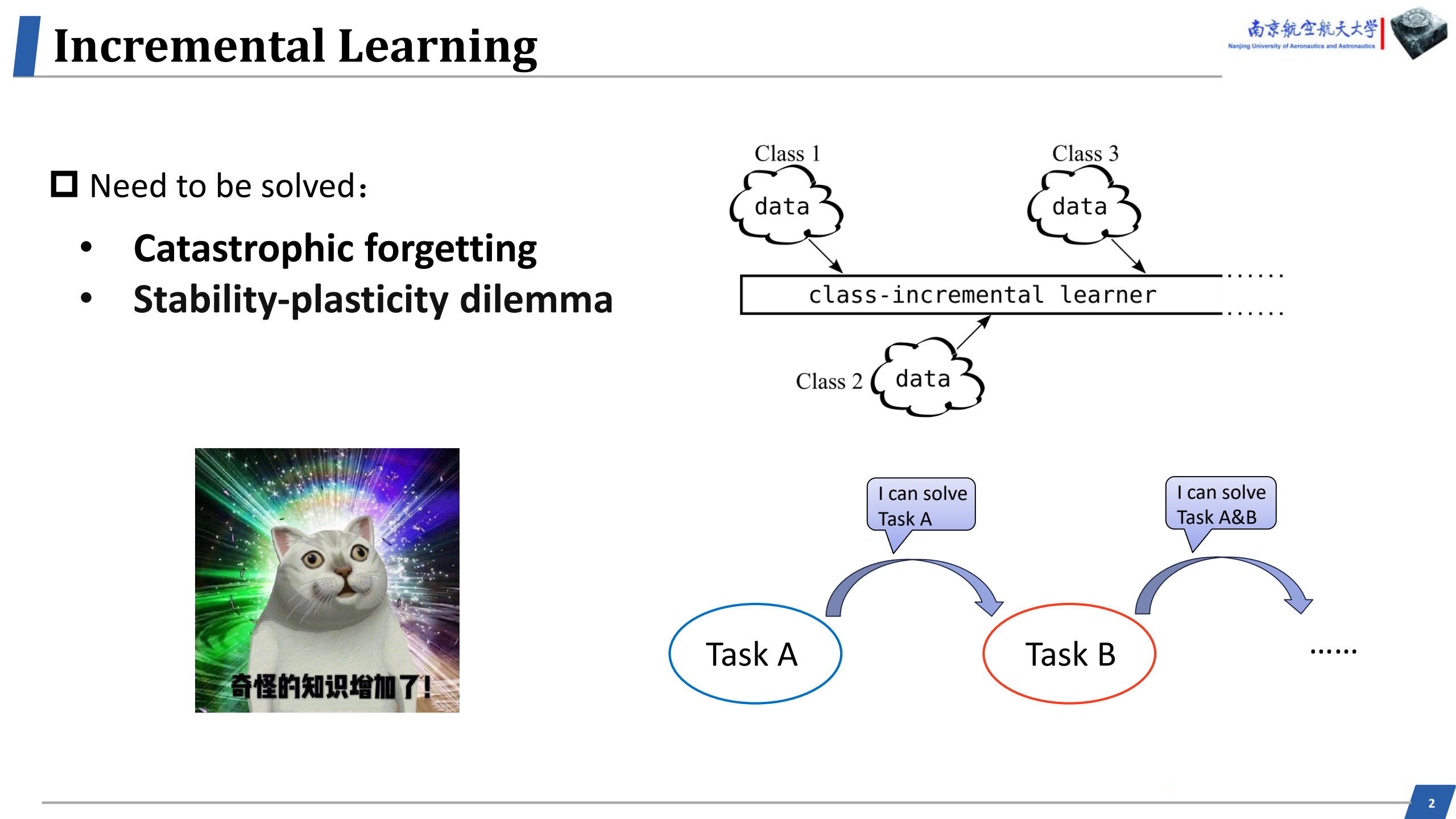
Nanjing University of Aeronautics and Astronautics

Learning without Forgetting

Zhizhong Li, Derek Hoiem, *Member, IEEE*

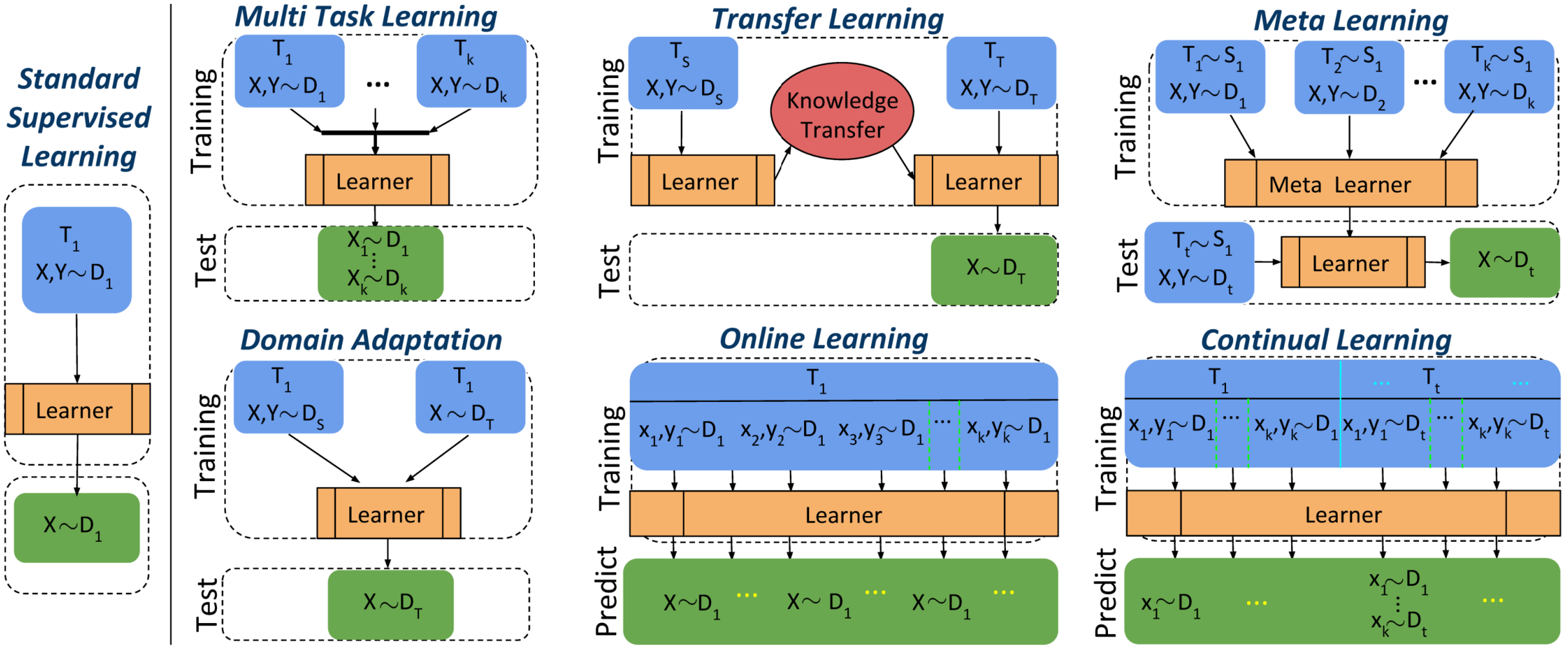
ECCV 2016

Incremental Learning



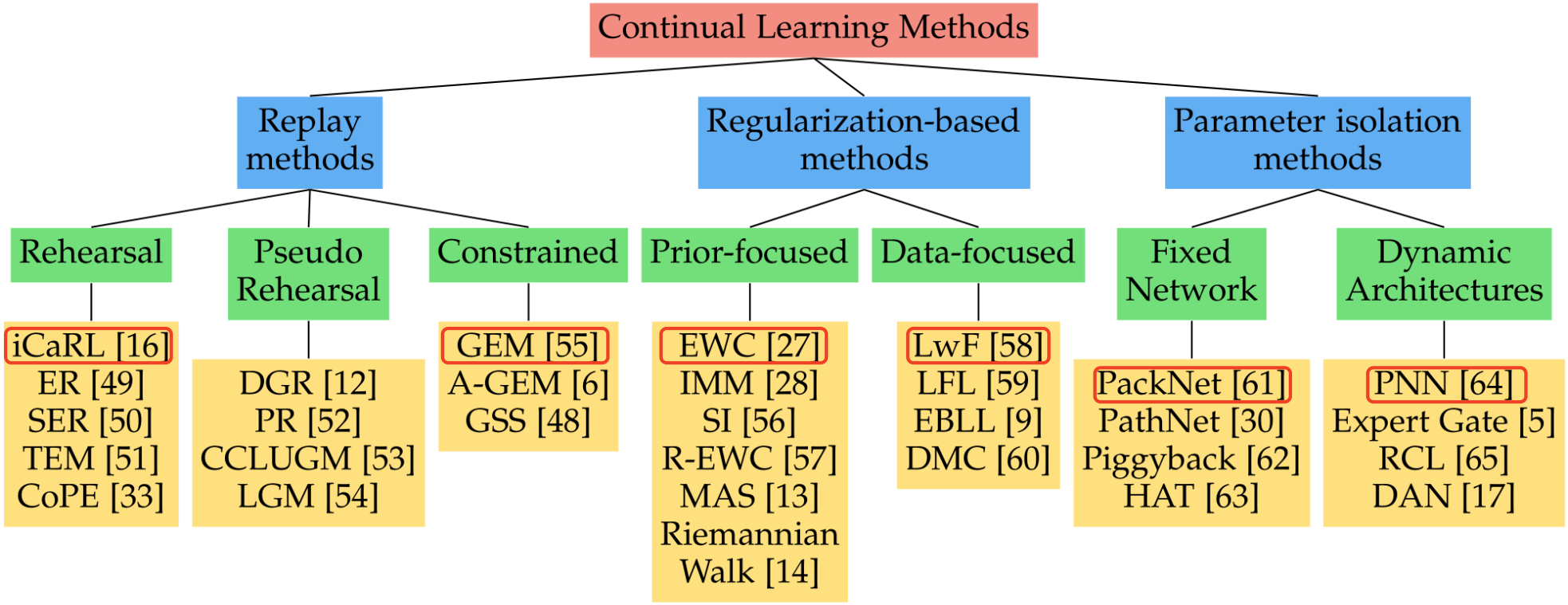
- Need to be solved：
  - **Catastrophic forgetting**
  - **Stability-plasticity dilemma**

Incremental Learning



- Continual Learning
- Incremental Learning
- Life long Learning

Incremental Learning



Learning without Forgetting (LwF)

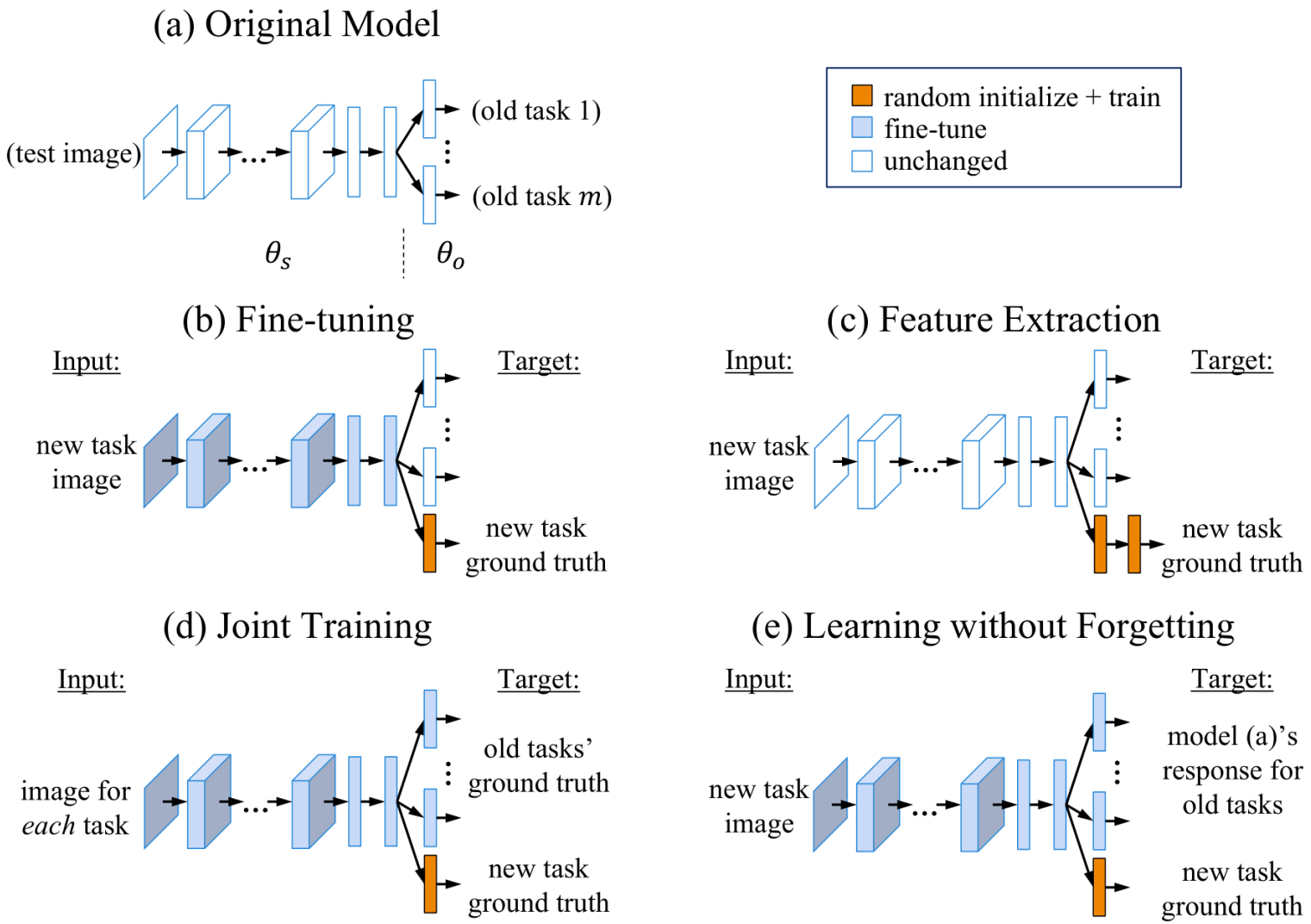


Fig. 2. Illustration for our method (e) and methods we compare to (b-d). Images and labels used in training are shown. Data for different tasks are used in alternation in joint training.

Learning without Forgetting (LwF)

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*

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

$$\mathcal{L}_{new}(\mathbf{y}_n, \hat{\mathbf{y}}_n) = -\mathbf{y}_n \cdot \log \hat{\mathbf{y}}_n \quad (1)$$

$$\mathcal{L}_{old}(\mathbf{y}_o, \hat{\mathbf{y}}_o) = -H(\mathbf{y}'_o, \hat{\mathbf{y}}'_o) \quad (2)$$

$$= -\sum_{i=1}^{l} y_o'^{(i)} \log \hat{y}_o'^{(i)} \quad (3)$$

$$y_o'^{(i)} = \frac{(y_o^{(i)})^{1/T}}{\sum_j (y_o^{(j)})^{1/T}}, \quad \hat{y}_o'^{(i)} = \frac{(\hat{y}_o^{(i)})^{1/T}}{\sum_j (\hat{y}_o^{(j)})^{1/T}}. \quad (4)$$

Experiments

(a) Using AlexNet structure (validation performance for ImageNet/Places365/VOC)

	ImageNet→VOC		ImageNet→CUB		ImageNet→Scenes		Places365→VOC		Places365→CUB		Places365→Scenes		ImageNet→MNIST	
	old	new	old	new	old	new	old	new	old	new	old	new	old	new
LwF (ours)	56.2	76.1	54.7	57.7	55.9	64.5	50.6	70.2	47.9	34.8	50.9	75.2	49.8	99.3
Fine-tuning	-0.9	-0.3	-3.8	-0.7	-2.0	-0.8	-2.2	0.1	-4.6	1.0	-2.1	-1.7	-2.8	0.0
LFL	0.0	-0.4	-1.9	-2.6	-0.3	-0.9	0.2	-0.7	0.7	-1.7	-0.2	-0.5	-2.9	-0.6
Fine-tune FC	0.5	-0.7	0.2	-3.9	0.6	-2.1	0.5	-1.3	1.8	-4.9	0.3	-1.1	7.0	-0.2
Feat. Extraction	0.8	-0.5	2.3	-5.2	1.2	-3.3	1.1	-1.4	3.8	-12.3	0.8	-1.7	7.3	-0.8
Joint Training	0.7	-0.2	0.6	-1.1	0.5	-0.6	0.7	-0.0	2.3	1.5	0.3	-0.3	7.2	-0.0

(b) Test set performance

	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

(c) Using VGGnet structure

	ImageNet→CUB		ImageNet→Scenes	
	old	new	old	new
LwF (ours)	60.6	72.5	66.8	74.9
Fine-tuning	-9.9	0.6	-4.1	-0.3
LFL	0.3	-2.8	-0.0	-2.1
Fine-tune FC	3.2	-6.7	1.4	-2.4
Feat. Extraction	8.2	-8.6	1.9	-5.1
Joint Training	8.0	2.5	4.1	1.5

南京航空航天大學
Nanjing University of Aeronautics and Astronautics

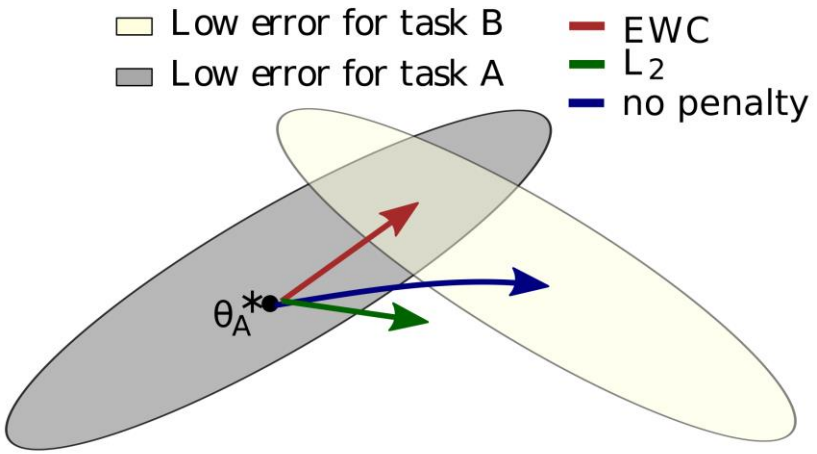
Overcoming catastrophic forgetting in neural networks

James Kirkpatrick[a], Razvan Pascanu[a], Neil Rabinowitz[a], Joel Veness[a], Guillaume Desjardins[a], Andrei A. Rusu[a], Kieran Milan[a], John Quan[a], Tiago Ramalho[a], Agnieszka Grabska-Barwinska [a], Demis Hassabis[a], Claudia Clopath[b], Dharshan Kumaran[a], and Raia Hadsell[a]

[a]DeepMind, London, N1C 4AG, United Kingdom
[b]Bioengineering department, Imperial College London, SW7 2AZ, London, United Kingdom

PNAS 2017

Elastic weight consolidation(EWC)



$$\log p(\theta|\mathcal{D}) = \log p(\mathcal{D}|\theta) + \log p(\theta) - \log p(\mathcal{D}) \tag{1}$$

$$\log p(\theta|\mathcal{D}) = \log p(\mathcal{D}_B|\theta) + \log p(\theta|\mathcal{D}_A) - \log p(\mathcal{D}_B) \tag{2}$$

$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2 \tag{3}$$

Elastic weight consolidation(EWC)

$$\log p(\theta|\mathcal{D}) = \log p(\mathcal{D}|\theta) + \log p(\theta) - \log p(\mathcal{D}) \tag{1}$$

$$p(\theta|D) = p(\theta|D_A, D_B)$$

$$p(\theta|D_A, D_B) = \frac{p(D_A, D_B|\theta)p(\theta)}{p(D_A, D_B)} = \frac{p(D_A|\theta)p(D_B|\theta)p(\theta)}{p(D_A)p(D_B)} = \frac{p(\theta|D_A)p(D_B|\theta)}{p(D_B)}$$

$$\log p(\theta|\mathcal{D}) = \log p(\mathcal{D}_B|\theta) + \boxed{\log p(\theta|\mathcal{D}_A)} - \log p(\mathcal{D}_B) \tag{2}$$

B Likelihood *Hard to solve!* *Constant*

Elastic weight consolidation(EWC)

$$\log p(\theta|\mathcal{D}) = \log p(\mathcal{D}_B|\theta) + \log p(\theta|\mathcal{D}_A) - \log p(\mathcal{D}_B) \qquad (2)$$

Laplace Approximation

$$p(\theta|D_A) \sim \mathcal{N}(\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\theta-\mu)^2}{2\sigma^2}}$$

$$f(\theta) = \log p(\theta \mid D_A) = \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{(\theta-\mu)^2}{2\sigma^2} \approx f(\theta_A^*) + \frac{1}{2}(\theta-\theta_A^*)^2 f''(\theta_A^*)$$

Taylor expansion to 2 order

$$-\frac{(\theta-\mu)^2}{2\sigma^2} = \frac{1}{2} f''(\theta_A^*)(\theta-\theta_A^*)^2 \longrightarrow \mu = \theta_A^*, \sigma^2 = -\frac{1}{f''(\theta_A^*)} \qquad p(\theta|D_A) \sim \mathcal{N}(\theta_A^*, -\frac{1}{f''(\theta_A^*)})$$

$$\arg\min_\theta L_B(\theta) - \frac{1}{2}(\theta-\theta_A^*)^2 f''(\theta_A^*)$$

Hessian Matrix

$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i (\theta_i - \theta_{A,i}^*)^2 \qquad (3)$$

Fisher Information Matrix

(a) It is equivalent to the second derivative of the loss near a minimum.
(b) It can be computed from first-order derivatives alone and is thus easy to calculate even for large models.
(c) It is guaranteed to be positive semi-definite.

Experiments

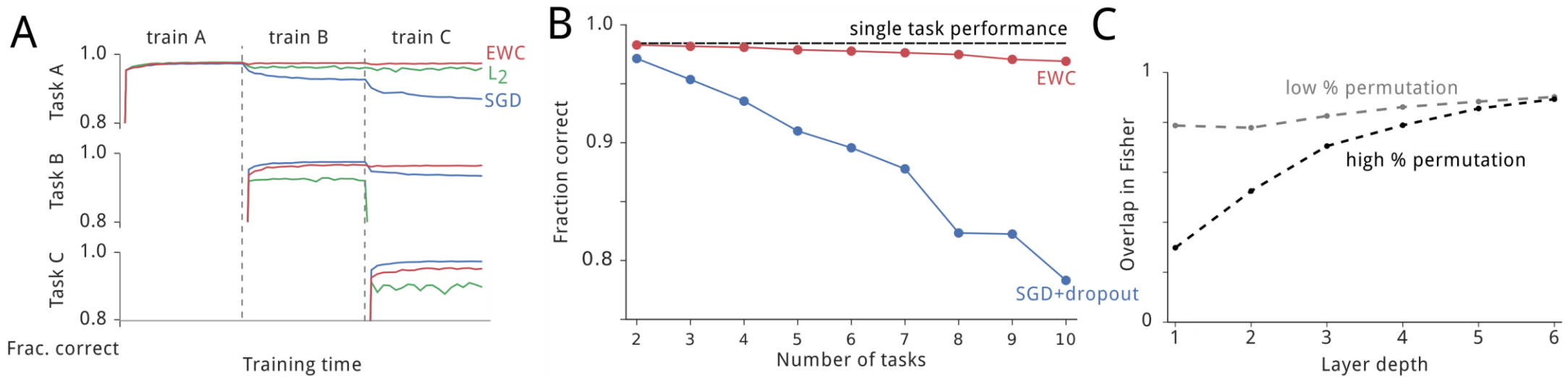


Figure 2: Results on the permuted MNIST task. A: Training curves for three random permutations A, B and C using EWC(red), L_2 regularization (green) and plain SGD(blue). Note that only EWC is capable of mantaining a high performance on old tasks, while retaining the ability to learn new tasks. B: Average performance across all tasks using EWC (red) or SGD with dropout regularization (blue). The dashed line shows the performance on a single task only. C: Similarity between the Fisher information matrices as a function of network depth for two different amounts of permutation. Either a small square of 8x8 pixels in the middle of the image is permuted (grey) or a large square of 26x26 pixels is permuted (black). Note how the more different the tasks are, the smaller the overlap in Fisher information matrices in early layers.

南京航空航天大學

Nanjing University of Aeronautics and Astronautics

iCaRL: Incremental Classifier and Representation Learning

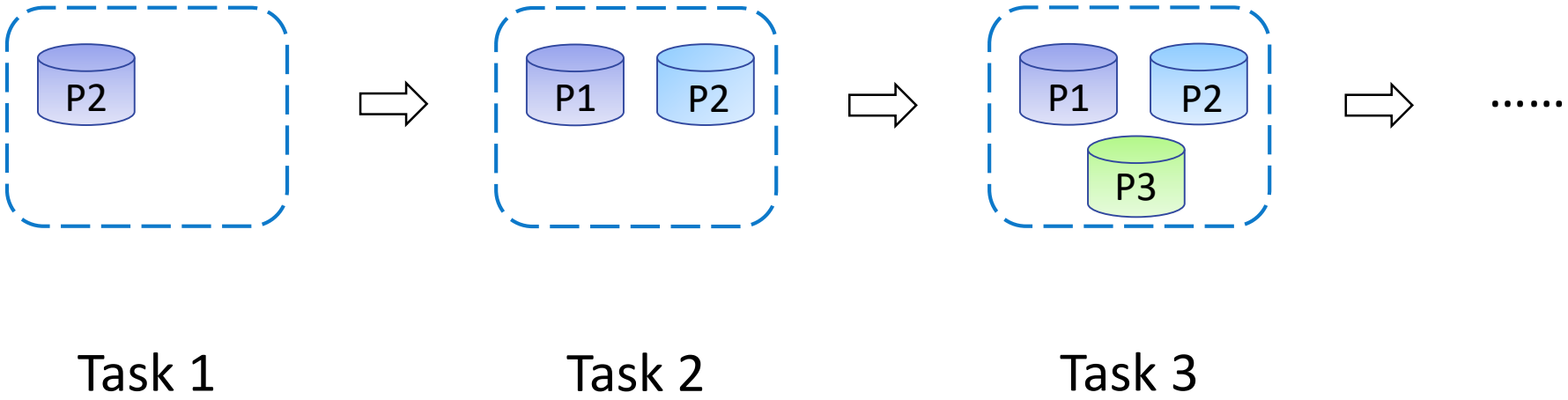
Sylvestre-Alvise Rebuffi
University of Oxford/IST Austria

Alexander Kolesnikov, Georg Sperl, Christoph H. Lampert
IST Austria

CVPR 2017

iCaRL

- *Training data for a small number of classes has to be present at the same time and new classes can be added progressively.*



iCaRL

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

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

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

iCaRL

Algorithm 4 iCaRL CONSTRUCTEXEMPLARSET

input image set $X = \{x_1, \ldots, x_n\}$ of class y
input m target number of exemplars
require current feature function $\varphi : \mathcal{X} \to \mathbb{R}^d$
  $\mu \leftarrow \frac{1}{n} \sum_{x \in X} \varphi(x)$ // current class mean
  for $k = 1, \ldots, m$ **do**
    $p_k \leftarrow \underset{x \in X}{\operatorname{argmin}} \left\| \mu - \frac{1}{k} [\varphi(x) + \sum_{j=1}^{k-1} \varphi(p_j)] \right\|$
  end for
  $P \leftarrow (p_1, \ldots, p_m)$
output exemplar set P

Algorithm 1 iCaRL CLASSIFY

input x // image to be classified
require $\mathcal{P} = (P_1, \ldots, P_t)$ // class exemplar sets
require $\varphi : \mathcal{X} \to \mathbb{R}^d$ // feature map
  for $y = 1, \ldots, t$ **do**
    $\mu_y \leftarrow \frac{1}{|P_y|} \sum_{p \in P_y} \varphi(p)$ // mean-of-exemplars
  end for
  $y^* \leftarrow \underset{y=1,\ldots,t}{\operatorname{argmin}} \|\varphi(x) - \mu_y\|$ // nearest prototype
output class label y^*

南京航空航天大學

Nanjing University of Aeronautics and Astronautics

Gradient Episodic Memory for Continual Learning

David Lopez-Paz and Marc'Aurelio Ranzato
Facebook Artificial Intelligence Research
{dlp,ranzato}@fb.com

NIPS 2017

Gradient Episodic Memory (GEM)

$$\ell(f_\theta, \mathcal{M}_k) = \frac{1}{|\mathcal{M}_k|} \sum_{(x_i,k,y_i)\in\mathcal{M}_k} \ell(f_\theta(x_i, k), y_i). \tag{5}$$

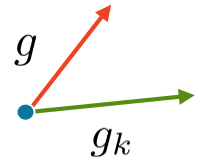
Constrained optimization:

$$\begin{aligned} \text{minimize}_\theta \quad & \ell(f_\theta(x,t),y) \\ \text{subject to} \quad & \ell(f_\theta, \mathcal{M}_k) \le \ell(f_\theta^{t-1}, \mathcal{M}_k) \text{ for all } k < t, \end{aligned} \tag{6}$$

Not saving θ

Angle < 90° :

$$\langle g, g_k \rangle := \left\langle \frac{\partial \ell(f_\theta(x,t),y)}{\partial\theta}, \frac{\partial \ell(f_\theta, \mathcal{M}_k)}{\partial\theta} \right\rangle \ge 0, \text{ for all } k < t. \tag{7}$$



Angle > 90° ? :

$$\begin{aligned} \text{minimize}_{\tilde{g}} \frac{1}{2} \quad & \|g - \tilde{g}\|_2^2 \\ \text{subject to} \quad & \langle \tilde{g}, g_k \rangle \ge 0 \text{ for all } k < t. \end{aligned} \tag{8}$$

南京航空航天大學

Nanjing University of Aeronautics and Astronautics

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

Arun Mallya and Svetlana Lazebnik
University of Illinois at Urbana-Champaign
{amallya2,slazebni}@illinois.edu

CVPR 2018

PackNet

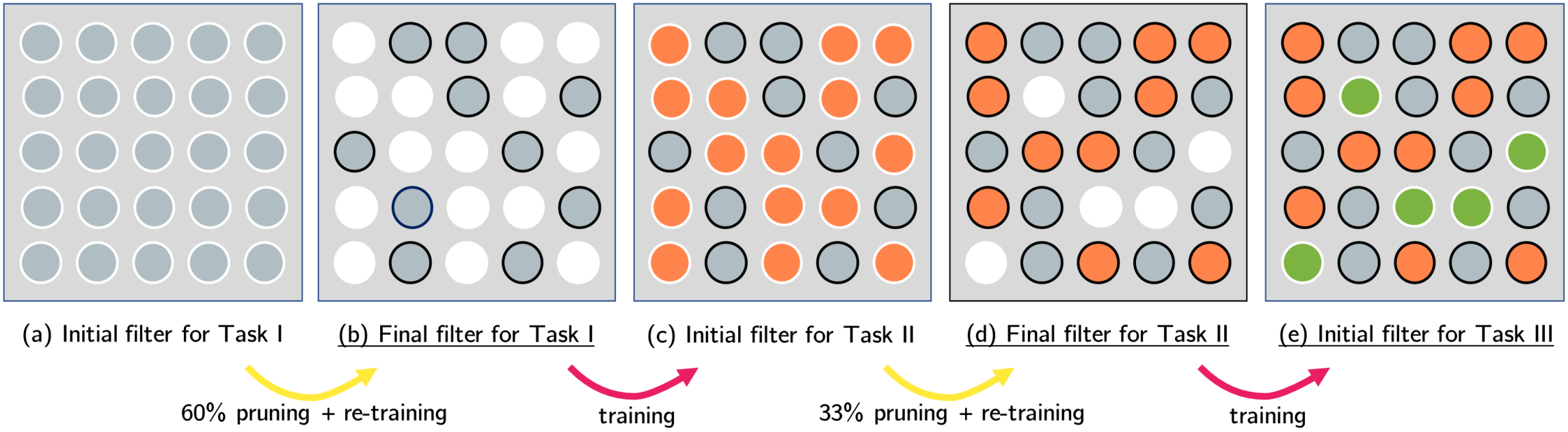


Figure 1: Illustration of the evolution of a 5×5 filter with steps of training. Initial training of the network for Task I learns a dense filter as illustrated in (a). After pruning by 60% (15/25) and re-training, we obtain a sparse filter for Task I, as depicted in (b), where white circles denote 0 valued weights. Weights retained for Task I are kept fixed for the remainder of the method, and are not eligible for further pruning. We allow the pruned weights to be updated for Task II, leading to filter (c), which shares weights learned for Task I. Another round of pruning by 33% (5/15) and re-training leads to filter (d), which is the filter used for evaluating on task II (Note that weights for Task I, in gray, are not considered for pruning). Hereafter, weights for Task II, depicted in orange, are kept fixed. This process is completed until desired, or we run out of pruned weights, as shown in filter (e). The final filter (e) for task III shares weights learned for tasks I and II. At test time, appropriate masks are applied depending on the selected task so as to replicate filters learned for the respective tasks.

南京航空航天大學

Nanjing University of Aeronautics and Astronautics

Progressive Neural Networks

Andrei A. Rusu*, Neil C. Rabinowitz*, Guillaume Desjardins, Hubert Soyer,
James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, Raia Hadsell

* These authors contributed equally to this work

Google DeepMind
London, UK

{andreirusu, ncr, gdesjardins, soyer, kirkpatrick, korayk, razp, raia}@google.com

NIPS 2016

Progressive Neural Networks (PNN)

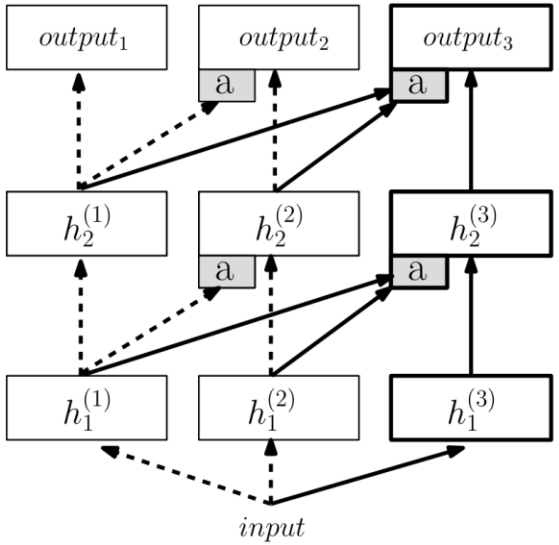


Figure 1: Depiction of a three column progressive network. The first two columns on the left (dashed arrows) were trained on task 1 and 2 respectively. The grey box labelled a represent the adapter layers (see text). A third column is added for the final task having access to all previously learned features.

Conclusion

Regularization: Need related tasks.

Replay: Need extra memory.

Parameter isolation: Need more parameters and computation.

THANKS