Learning without forgetting

Zhizhong Li and Derek Hoiem

European Conference on Computer Vision (ECCV), 2016

Presented by Boyoung Kim

May 4, 2017

Contents

1. Introduction

- We want to build a unified vision system or gradually add new capabilities to a system.
  (e.g., For construction safety, a system can identify whether a worker is wearing a safety vest or hard hat, but a superintendent may wish to add the ability to detect improper footwear.)

- The usual assumption is that training data for all tasks is always available. However, as the number of tasks grows, storing and retraining on such data becomes infeasible.

⇒ How can we use only new task data to train the network while preserving the original capabilities on image classification problems with Convolutional Neural Network(CNN)?

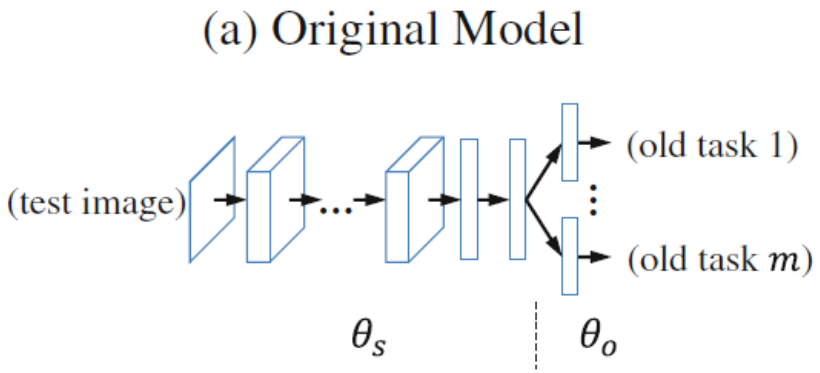
2. Three Common Approaches

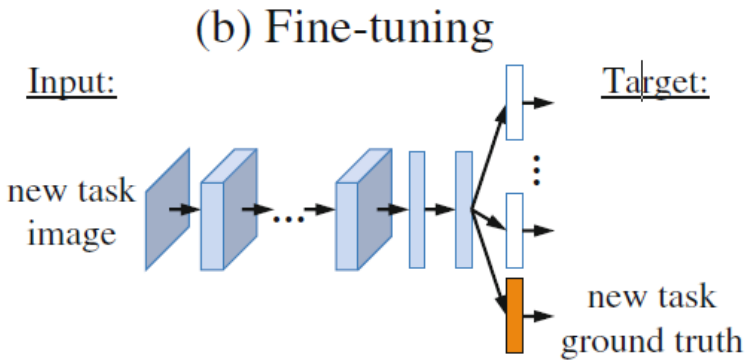
Transfer learning : Storing Knowledge gained solving one problem and applying it to a different but related problems

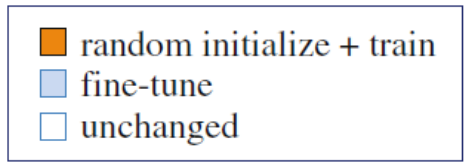
Setting

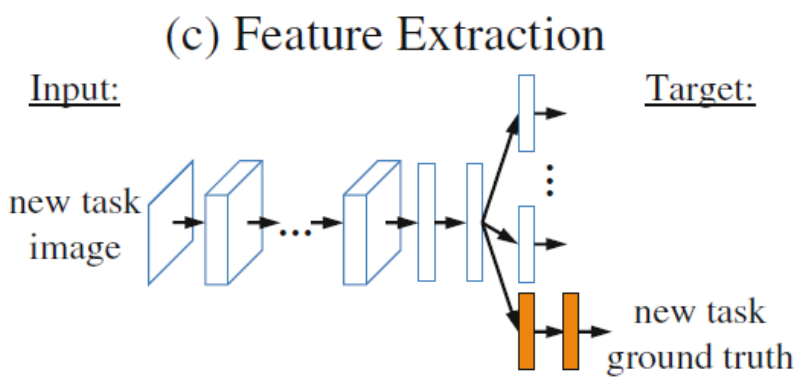
- θ_s : a set of shared parameters
- θ_o : task-specific parameters for previously learned tasks
- θ_n : randomly initialized task-specific parameters for new tasks

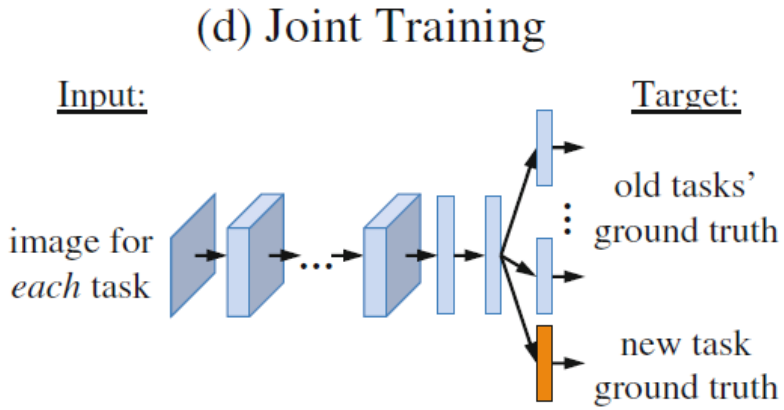
2. Three Common Approaches











2. Three Common Approaches

Each of these strategies has a major drawback.

- Feature extraction typically underperforms on the new task.
- Fine-tuning degrades performance on previously learned tasks
- Joint training becomes increasingly cumbersome in training as more tasks are learned and the training data for previously learned tasks is needed.

⇒ Our goal is to add task-specific parameters θ_n for a new task and to learn parameters that work well on old and new tasks, using images and labels from only the new task.

3. Learning Without Forgetting

Procedure for 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*

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

√ When training, we first freeze θ_s and θ_o and train θ_n to convergence. Then, we jointly train all weights until convergence.

3. Learning Without Forgetting

- For new task, we use common multinomial logistic loss :

$$L_{new}(y_n, \widehat{y_n}) = -y_n \log \widehat{y_n}$$

where $\widehat{y_n}$ is the softmax output of the network and y_n is the one-hot ground truth label vector.

- For each original task, a modified cross-entropy loss (Distillation loss, Hinton et al.[1]) that increases the weight for smaller probabilities for T>1:

$$L_{old}(y_n, \widehat{y_n}) = -\sum_{i=1}^{l} y_o'^{(i)} \log \hat{y}_o'^{(i)}$$

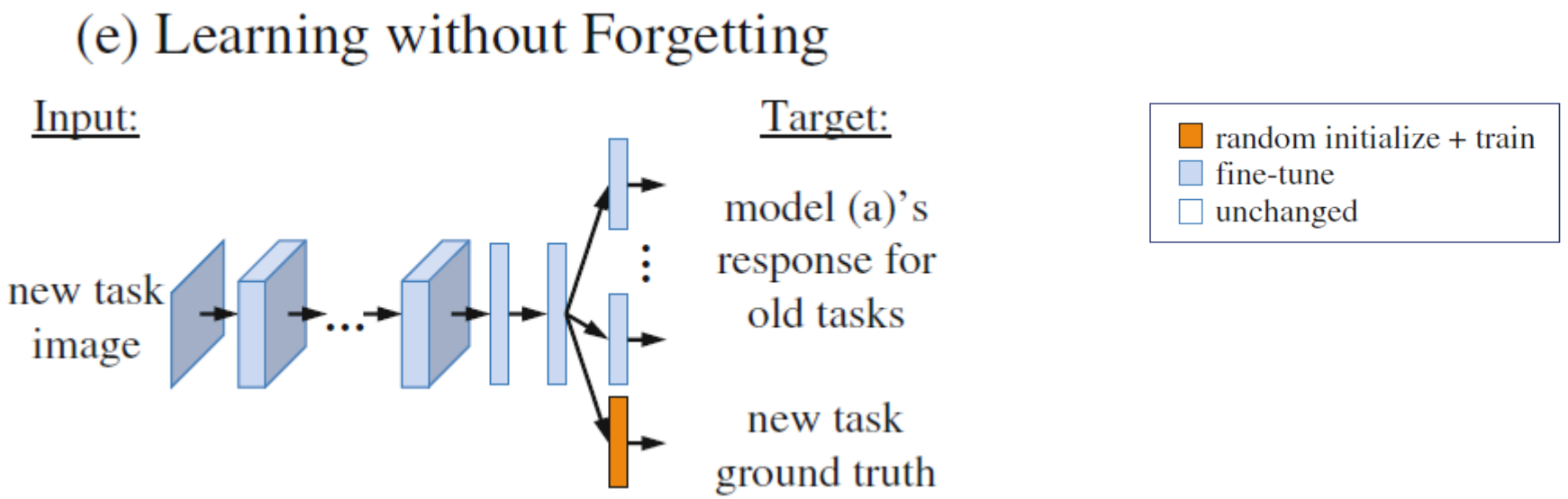
where l is the number of labels and $y_o'^{(i)}$, $\hat{y}_o'^{(i)}$ are the modified versions of recorded and current probabilities $y_o^{(i)}$, $\hat{y}_o^{(i)}$:

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

- The regularization R corresponds to a simple weight decay of 0.0005.

[1] Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. NIPS 2014

3. Learning Without Forgetting



4. Experimental Results

Performance for the single New Task Scenario.

(a) Using AlexNet structure (validation performance for ImageNet/Places2/VOC)

	ImageNet→VOC		ImageNet→CUB		ImageNet→Scenes		Places2→VOC		Places2→CUB		Places2→Scenes		ImageNet→MNIST	
	old	new	old	new	old	new	old	new	old	new	old	new	old	new
LwF (ours)	56.5	75.8	55.1	57.5	55.9	64.5	43.3	72.1	38.4	41.7	43.0	75.3	52.1	99.0
fine-tuning	-1.4	-0.3	-5.1	-1.5	-3.4	-1.0	-1.8	-0.1	-9.1	-0.8	-4.1	-0.8	-4.9	0.2
feat. extraction	0.5	-1.1	2.0	-5.3	1.2	-3.7	-0.2	-3.9	4.7	-19.4	0.2	-0.5	5.0	-0.8
joint training	0.2	0.0	0.5	-0.9	0.5	-0.6	-0.1	0.1	3.3	-0.2	0.2	0.1	4.7	0.2

(b) Test set performance

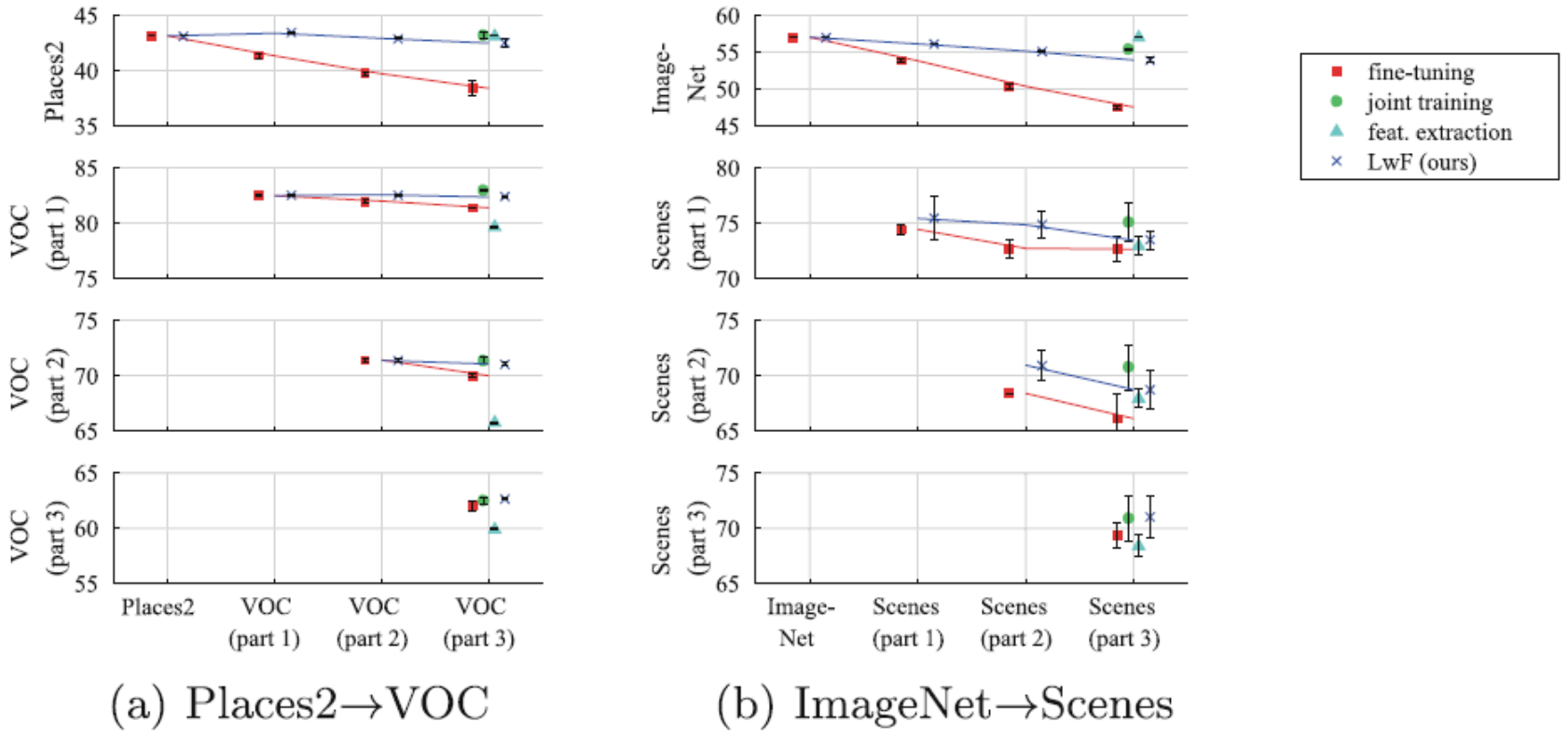
	Places2→VOC	
	old	new
LwF (ours)	41.1	75.2
fine-tuning	-1.9	-0.1
feat. extraction	0.1	-3.5
joint training	0.0	0.0

(c) Using VGGnet structure

	ImageNet→CUB		ImageNet→Scenes	
	old	new	old	new
LwF (ours)	65.6	72.3	68.1	74.7
fine-tuning	-11.0	-0.2	-5.6	-0.7
feat. extraction	3.1	-9.1	0.7	-5.1
joint training	2.5	2.3	2.0	0.8

4. Experimental Results

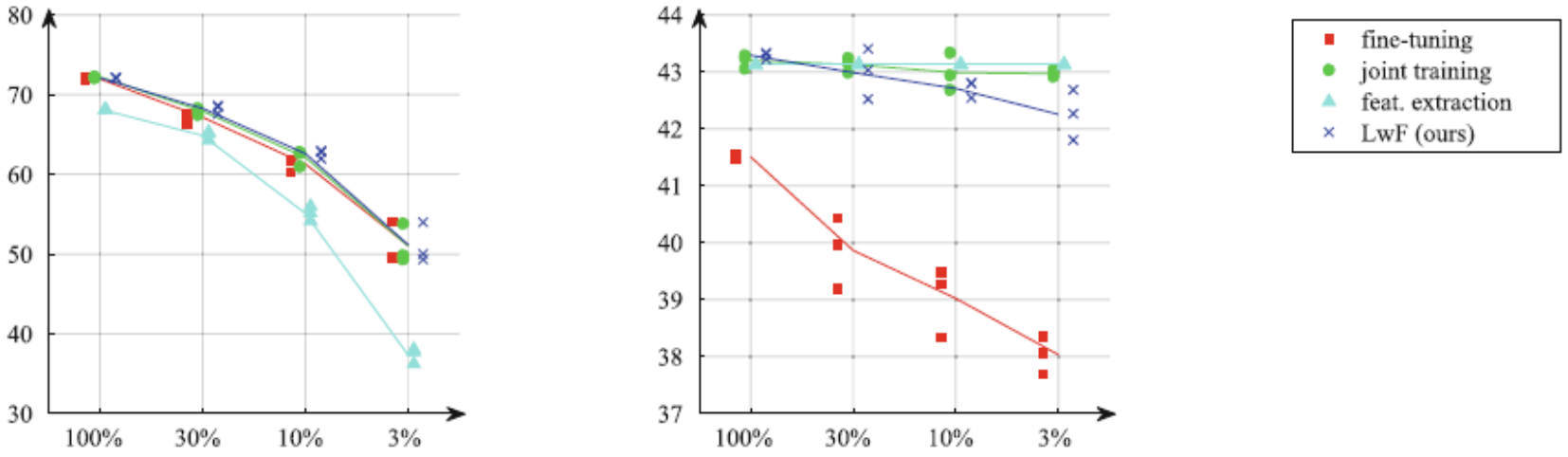
Performance for the multiple New Task Scenario.



(a) Places2→VOC (b) ImageNet→Scenes

4. Experimental Results

Influence of Dataset Size



(a) VOC mAP (new) (b) Places2 accuracy (old)