Learning Without Forgetting

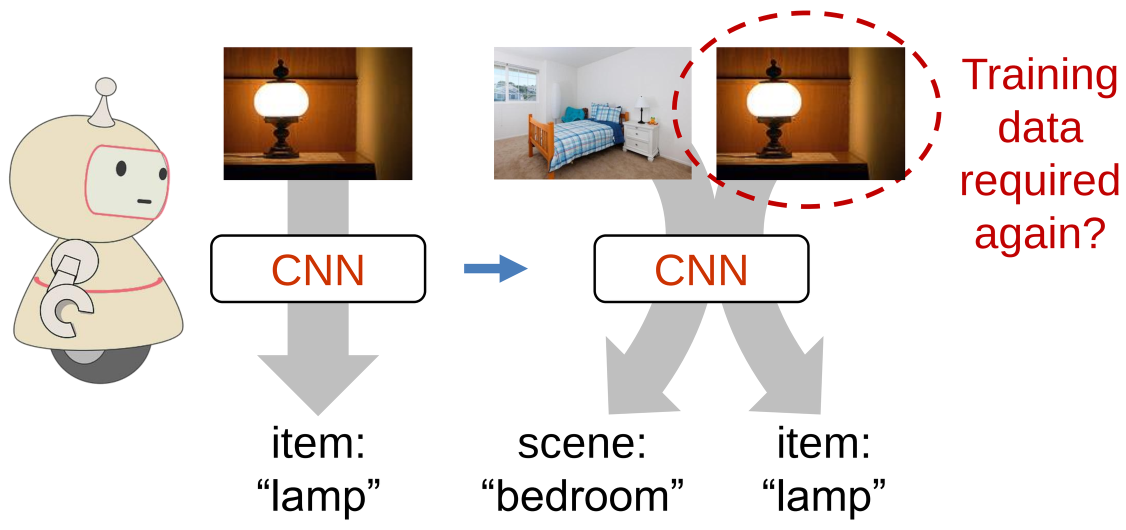
Zhizhong Li, Derek Hoiem {zli115,dhoiem}@Illinois.edu
Department of Computer Science, University of Illinois at Urbana-Champaign

ILLINOIS

http://zli115.web.engr.illinois.edu/learning-without-forgetting/

Motivation

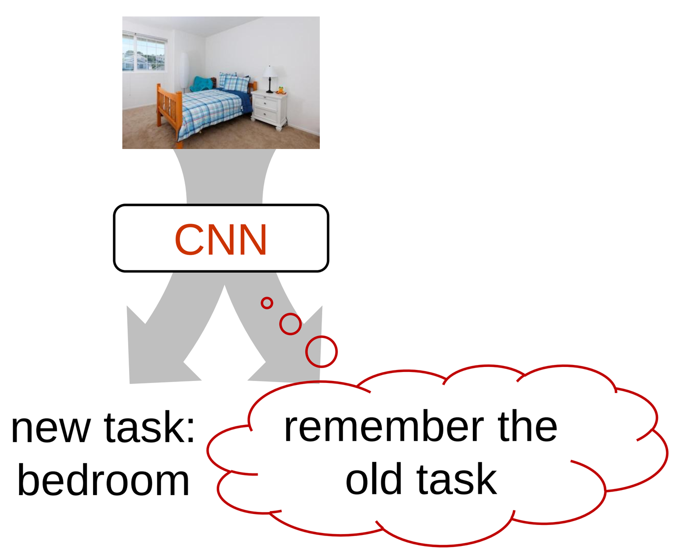
When expanding the capability of a vision system…
- Fine-tuning? (old task suffers)
- Feature extraction? (new task suffers)
- Joint training:



What if the original dataset…
- Is not recorded?
- Is proprietary?
- Is too cumbersome?

But we want…
- Benefit of shared representation
- No or little degradation of the original capability
- Without the need to access original task dataset?

Goal:
Add new capabilities to a CNN-based vision system using *only data from the new task*.



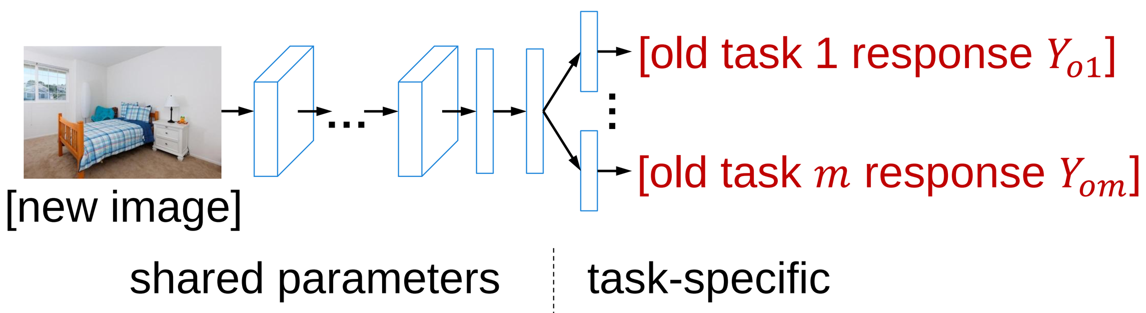
Our strengths:
- Outperforms the widely-used fine-tuning on *both original and new task*.
- Outperforms feature extraction on the new task.
- Simple to implement and deploy
- Training efficiency compared to joint training

Method

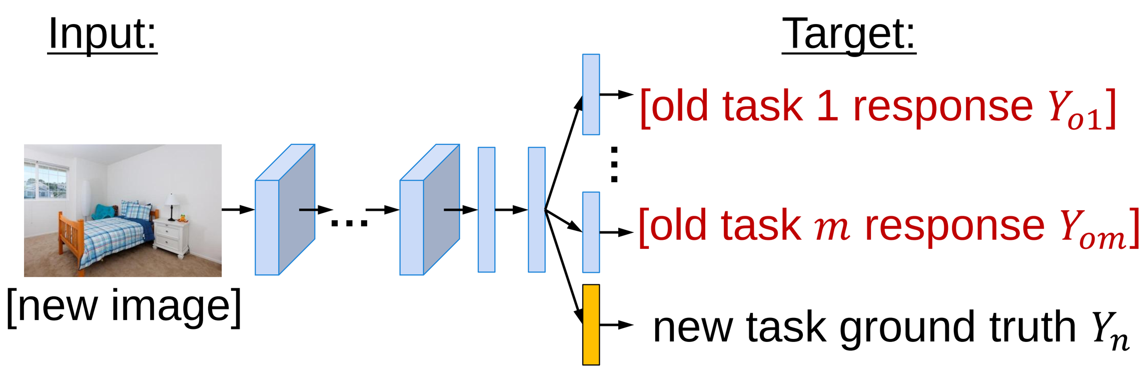
random initialize + train
fine-tune
frozen (original)

Outline

1. Obtain old task responses



2. Train on new images



Training: loss

shared/old/new parameters — old task response preservation loss

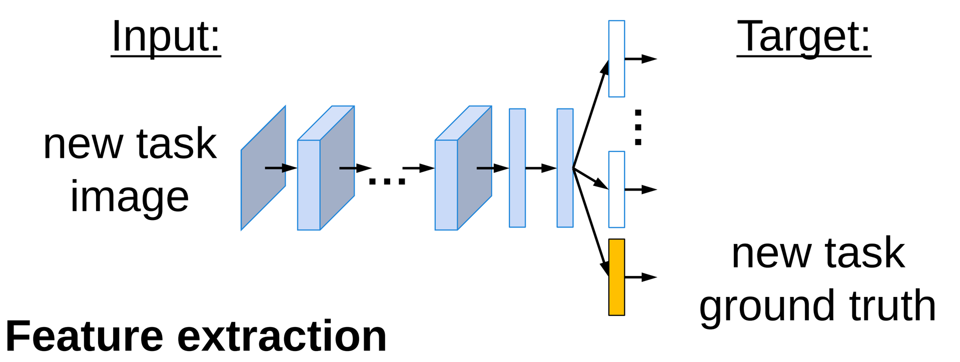
$$\theta_s^*,\ \theta_o^*,\ \theta_n^* \leftarrow \underset{\hat{\theta}_s,\hat{\theta}_o,\hat{\theta}_n}{\operatorname{argmin}} \left(\sum_i^m \mathcal{L}_{old}(Y_o, \hat{Y}_o; \hat{\theta}_s, \hat{\theta}_o) + \mathcal{L}_{new}(Y_n, \hat{Y}_n; \hat{\theta}_s, \hat{\theta}_n) + \mathcal{R}(\hat{\theta}_s,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

new task classification loss — regularization

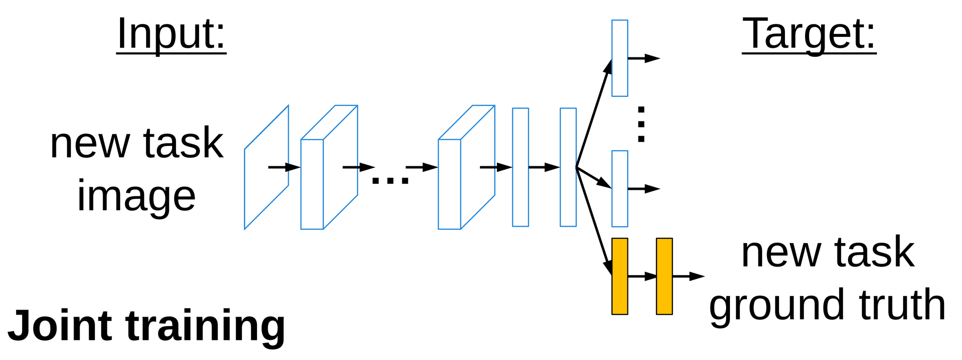
Compared methods

random initialize + train
fine-tune
frozen (original)

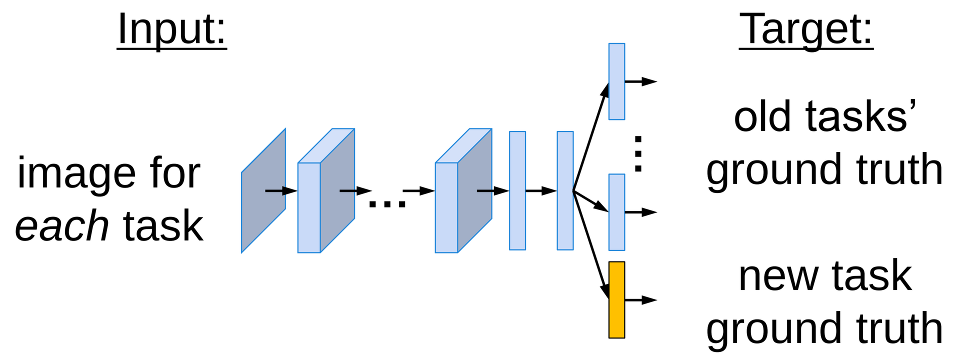
- **Fine-tuning**



- **Feature extraction**



- **Joint training**



Limitations of existing methods

	Fine Tuning	Duplicating and Fine Tuning	Feature Extraction	Joint Training	Learning without Forgetting
new task performance	good	good	X medium	best	✓ best
original task performance	X bad	good	good	good	✓ good
training efficiency	fast	fast	fast	X slow	✓ fast
testing efficiency	fast	X slow	fast	fast	✓ fast
storage requirement	medium	X large	medium	X large	✓ medium
requires old task data	no	no	no	X yes	✓ no

Experiment Settings

Datasets

old task	+	new task
ImageNet 2012* Places2*	+	PASCAL VOC 2012 Caltech-UCSD Birds MIT indoor scenes MNIST

* Pre-trained AlexNet obtained from authors

Efficiency:

- Training: forward-pass shared parameters once. Faster than joint training, similar to fine-tuning
- Test: same as compared methods; more efficient than keeping different networks for each task

Design choices and alternatives

We experimented with some variations:
- Possibly: more layers as task-specific parameters.
- Possibly: add nodes to earlier layers
- Possibly: use alternative loss for $\mathcal{L}_{old}(Y_o, \hat{Y}_o)$
- Possibly: just reduce fine-tuning learning rate

These variations provided insignificant or inconsistent improvements, if any.

Results

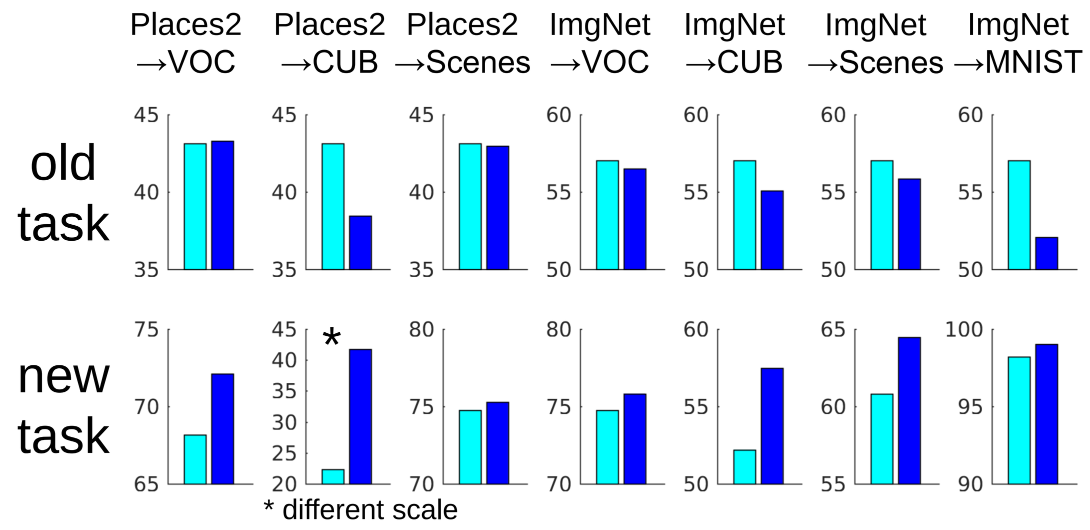
Single new task scenario

Feature Extraction
Fine-tuning
Joint Training
LwF (ours)

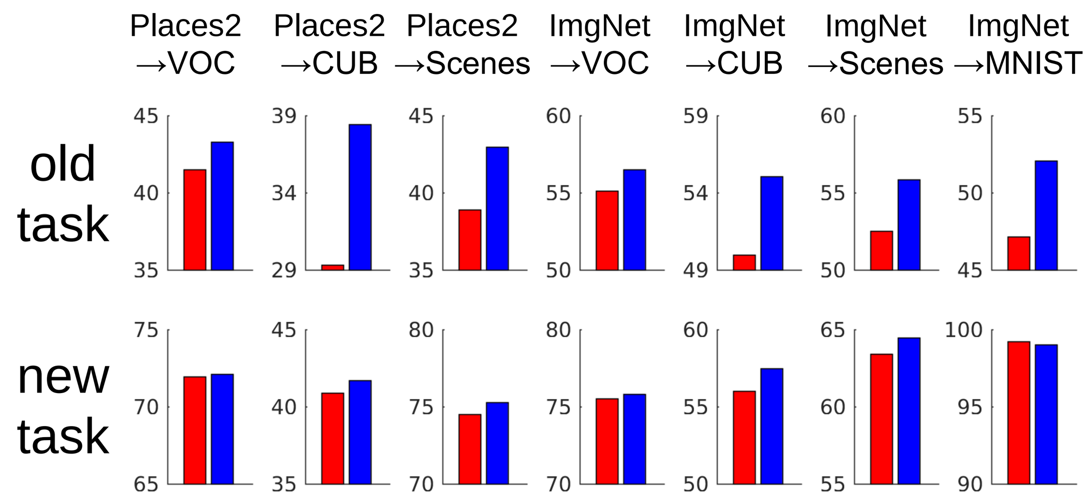
1 old → 1 old + 1 new

* Accuracy (average precision for VOC)
* Using AlexNet

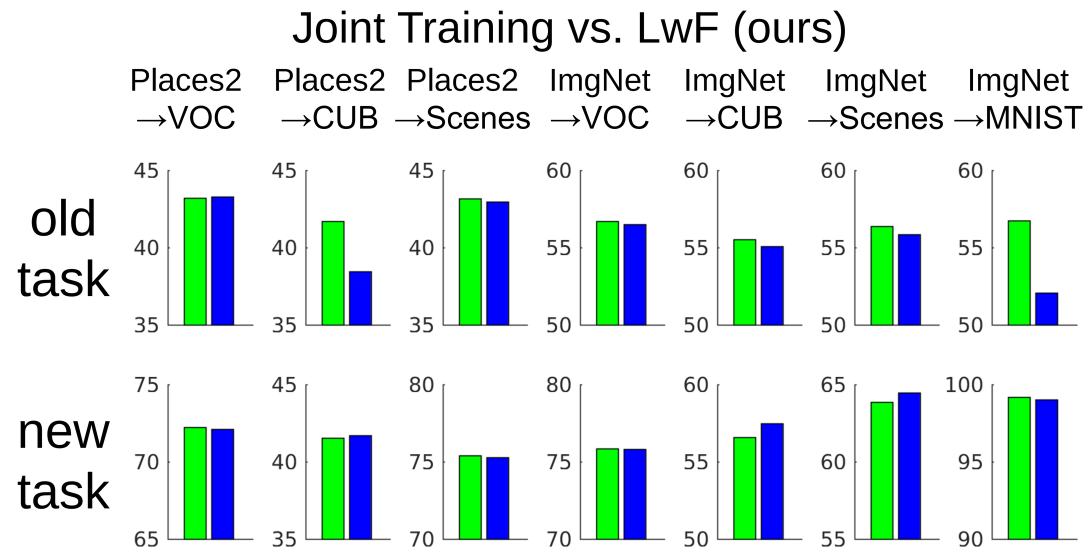
Feature Extraction vs. LwF (ours)



Fine-tuning vs. LwF (ours)



Joint Training vs. LwF (ours)

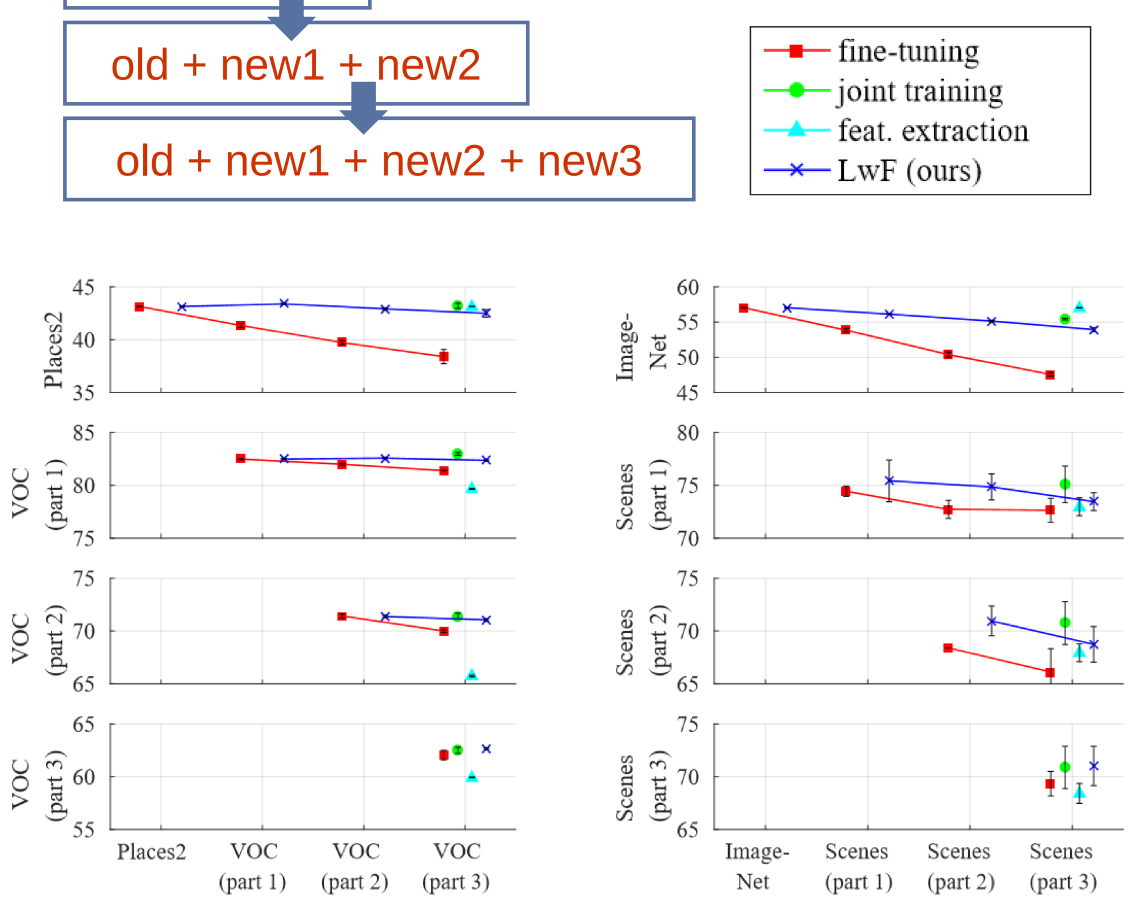


* Validation set results shown. Test set results similar
* VGG-16 network results are mostly similar, however Joint Training outperforms our method more on both tasks (0.8%~2.5%)

Multiple new task scenario

old → old + new1 → old + new1 + new2 → old + new1 + new2 + new3

The new dataset is split into three parts. Record Y_{oi} again each time.



(a) Places2→VOC (b) ImageNet→Scenes

Conclusions

- Vs. Feature Extraction: LwF outperforms on new task; underperforms on old task
- Vs. Fine-tuning: LwF outperforms on both tasks, as keeping old responses regularizes model
- Vs. Joint Training: LwF performs nearly as well as joint training
- Dissimilar new tasks degrade old task performance
- Similar results and same observations for adding multiple new tasks

Acknowledgments This material is based upon work supported in part by NSF Awards 14-46765 and 10-53768 and ONR MURI N000014-16-1-2007.