Learning to Self-Train for Semi-Supervised Few-Shot Classification

Xinzhe Li, Qianru Sun, Yaoyao Liu
Shibao Zheng, Qin Zhou, Tat-Seng Chua, Bernt Schiele

{lxz1217, sbzh}@sjtu.edu.cn qianrusun@smu.edu.sg liuyaoyao@tju.edu.cn
xining.zq@alibaba-inc.com chuats@comp.nus.edu.sg {qsun, schiele}@mpi-inf.mpg.de

Motivation

- Few-shot classification is challenging due to the scarcity of labeled training data, e.g. only one labeled data point per class.
- Semi-supervised learning is a potential approach to tackling this challenge with low cost.

- **Semi-supervised few-shot classification**
  - how to leverage massive unlabeled data in few-shot learning regimes
  - how to overcome the distracting classes mixed in unlabeled data

Contribution

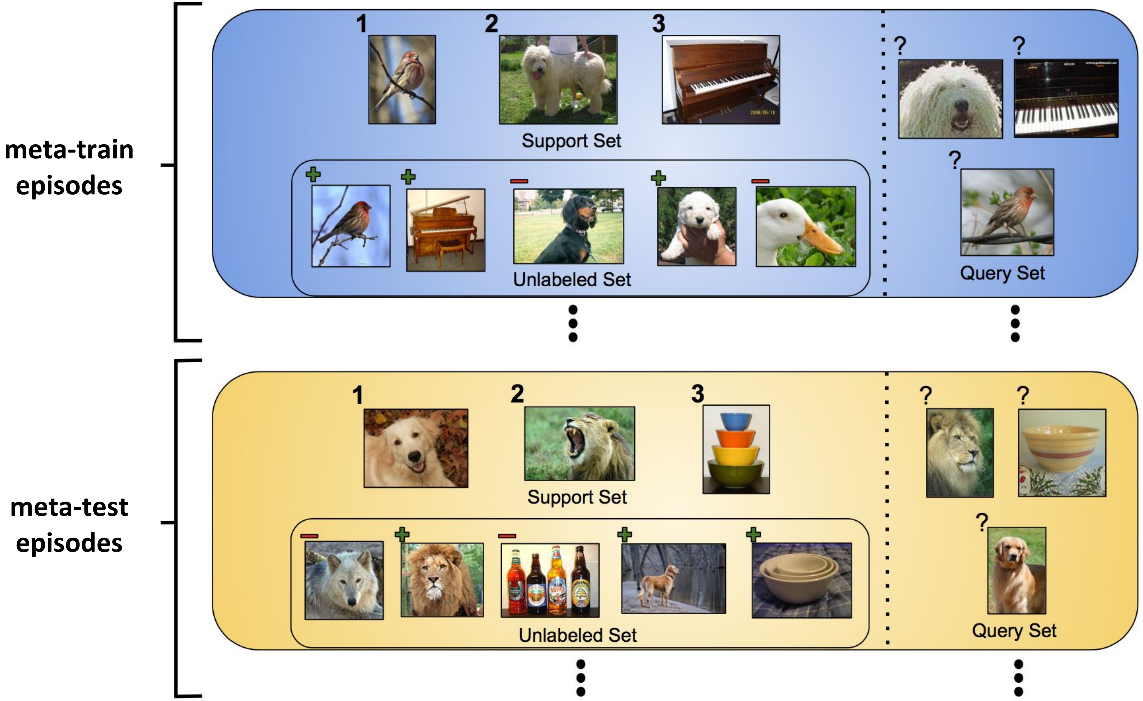
- **A novel self-training strategy** that prevents the model from drifting due to label noise and enables robust recursive training.
- **A novel meta-learned cherry-picking method** that optimizes the weights of pseudo labels particularly for fast and efficient self-training.
- **Extensive experiments on two benchmarks** – miniImageNet and tieredImageNet, in which our method achieves the top performance.

★ **Code is available at:**

https://github.com/xinzheli1217/learning-to-self-train

Problem definition [2]

- Meta-Learning paradigm
  - meta-train
  - meta-test
- Episodic data splits
  - support set $\mathcal{S}$
  - query set $\mathcal{Q}$
  - unlabeled set $\mathcal{R}$

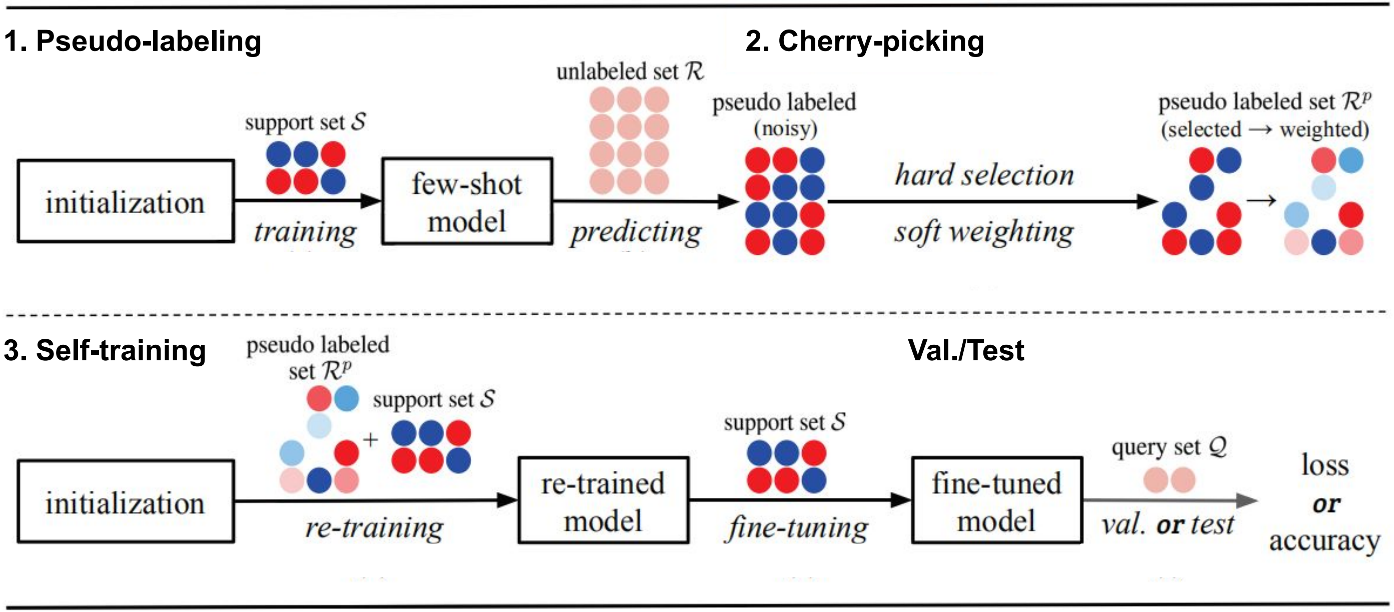


Our approach: learning to self-train (LST)

- Meta-learning based approach: learning to self-train (LST)
- Inner loop (base-learning)
  - pseudo-labeling the unlabeled data
  - cherry-picking the better labeled data
  - self-training the base-learner with cherry-picked data
- Outer loop (meta-learning)
  - meta gradient descent to optimize the meta-learners

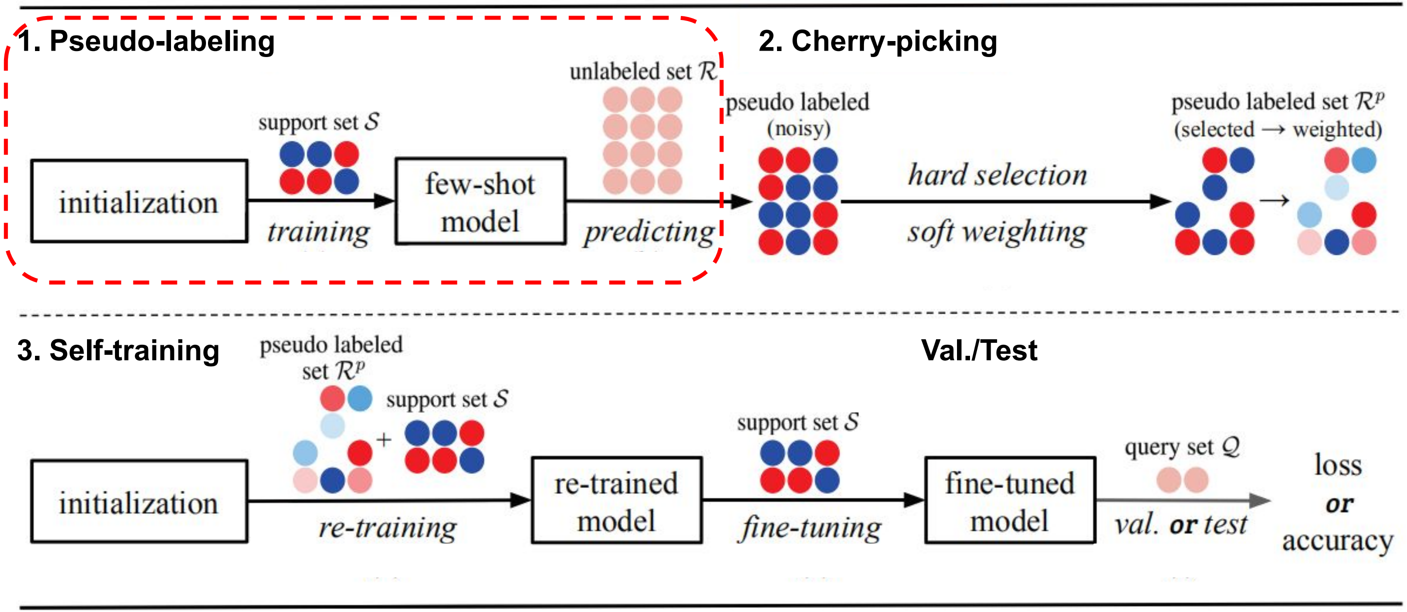
The framework of LST

- Inner loop:



The framework of LST

- Inner loop:



1. Pseudo-labeling

- Initialization to few-shot model: pre-training a few-shot model by MTL[3].

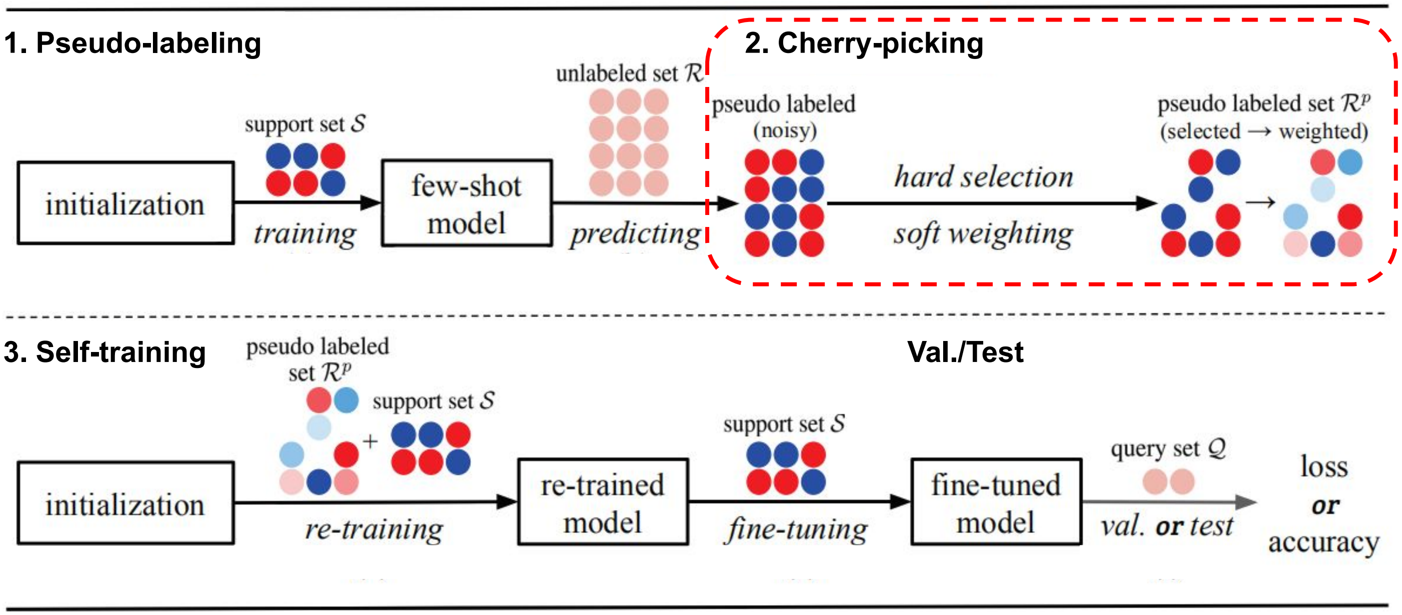
- Given the support set $\mathcal{S}$, we use the cross-entropy loss to optimize the task-specific base-learner θ by gradient descent for T iters:

$$\theta_t \leftarrow \theta_{t-1} - \alpha \nabla_{\theta_{t-1}} L\big(\mathcal{S}; [\Phi_{ss}, \theta_{t-1}]\big)$$

  Once θ_T is trained, we use it to predict the pseudo labels of the unlabeled data $\mathcal{R}$.

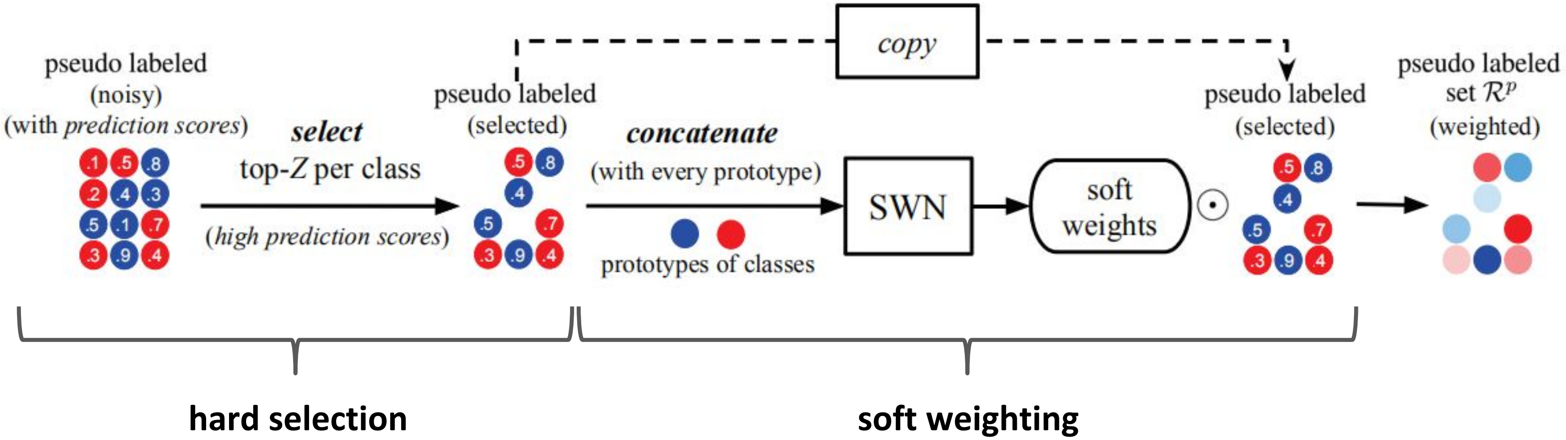
The framework of LST

- Inner loop:



2. Cherry-picking

- Processing the pseudo labels by **hard selection** and **soft weighting**.



2. Cherry-picking

- **Hard selection**: picking up the top Z samples per class, according to the confident scores of pseudo labeled samples.

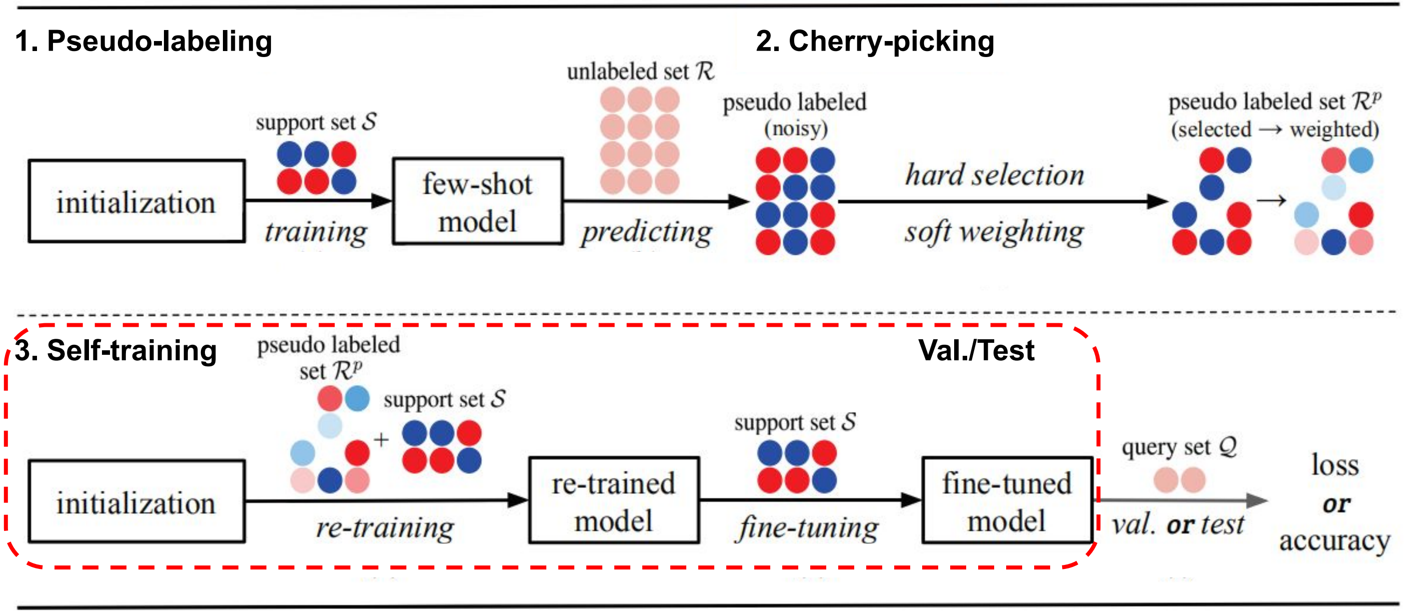
- **Soft weighting**: computing the soft weights of selected samples by a meta-learned soft weighting network (SWN). We refer to RelationNets [5] and compute a sample's weight on the c-th class as:

$$w_{i,c} = f_{\Phi_{swn}}\left(\left[f_{\Phi_{ss}}(x_i); \frac{\sum_k f_{\Phi_{ss}}(x_{c,k})}{K}\right]\right)$$

$f_{\Phi_{ss}}$ is the backbone meta-learner

The framework of LST

- Inner loop:



3. Self-training

- Self-training base-learner contains two stages:
  - **re-training** with cherry-picked data $\mathcal{R}^p$ and support set $\mathcal{S}$
  - **fine-tuning** with only support set $\mathcal{S}$

- An **iterative procedure** can be used in self-training, i.e., recursive training, to enhance the performance.

3. Self-training

- In the first m steps, θ_t is trained as:

$$\theta_t \leftarrow \theta_{t-1} - \alpha \bigtriangledown_{\theta_{t-1}} L\big(\mathcal{S} \cup \mathcal{R}^p; [\Phi_{swn}, \Phi_{ss}, \theta_{t-1}]\big)$$

$$L\big(\mathcal{S} \cup \mathcal{R}^p; [\Phi_{swn}, \Phi_{ss}, \theta_t]\big) = \begin{cases} L_{ce}\big(f_{[\Phi_{swn}, \Phi_{ss}, \theta_t]}(x_i), y_i\big), \text{ if } (x_i, y_i) \in \mathcal{S} \\ L_{ce}\big(\mathbf{w}_i \odot f_{[\Phi_{swn}, \Phi_{ss}, \theta_t]}(x_i), y_i\big), \text{ if } (x_i, y_i) \in \mathcal{R}^p \end{cases}$$

- In the rest T - m steps, θ_t is fine-tuned with $\mathcal{S}$ as:

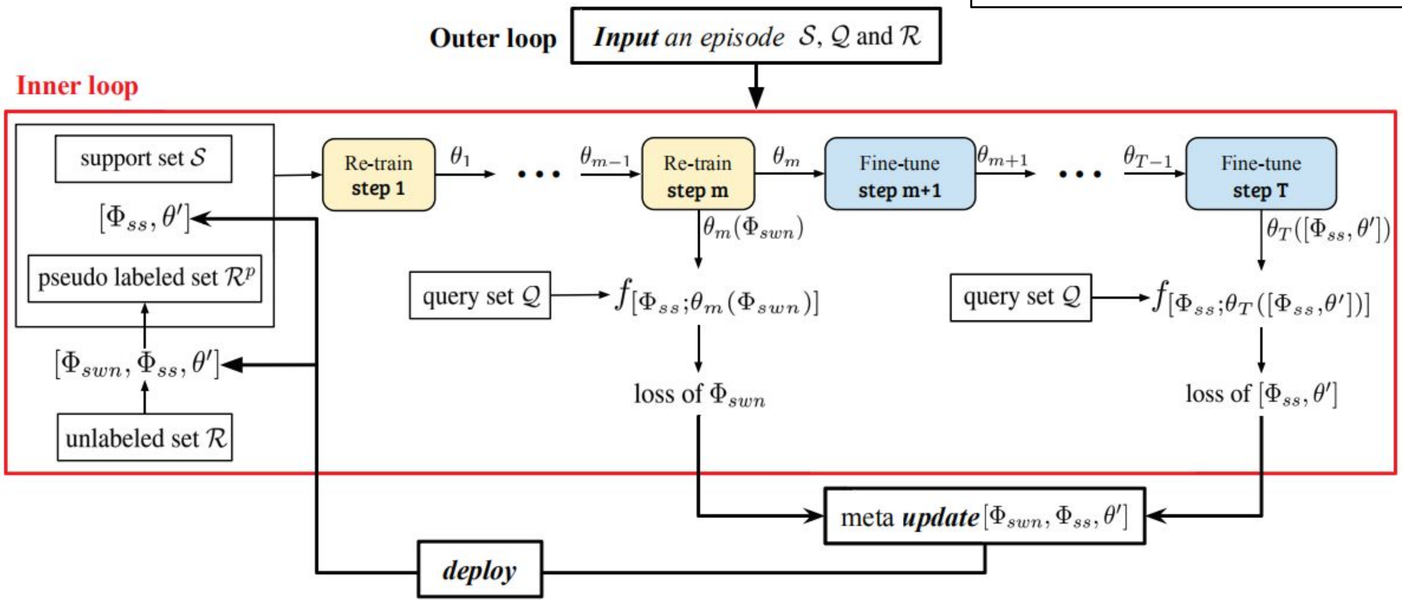
$$\theta_t \leftarrow \theta_{t-1} - \alpha \bigtriangledown_{\theta_{t-1}} L(\mathcal{S}; [\Phi_{swn}, \Phi_{ss}, \theta_{t-1}])$$

The framework of LST

- Outer loop with an inner loop:

After fine-tuning steps, using validation loss (on query set) to update Φ_{ss} and θ'.

After re-training steps, using validation loss (on query set) to update Φ_{swn} .



Experiments

- Comparing with few-shot learning methods, on miniImagenet dataset

Few-shot Learning Method		**Backbone**	**miniImageNet (test)** 1-shot	5-shot
Data augmentation	Adv. ResNet, [13]	WRN-40 (pre)	55.2	69.6
	Delta-encoder, [27]	VGG-16 (pre)	58.7	73.6
Gradient descent	MAML, [3]	4 CONV	48.70 ± 1.75	63.11 ± 0.92
	Meta-LSTM, [21]	4 CONV	43.56 ± 0.84	60.60 ± 0.71
	Bilevel Programming, [5]	ResNet-12$^{\diamond}$	50.54 ± 0.85	64.53 ± 0.68
	MetaGAN, [41]	ResNet-12	52.71 ± 0.64	68.63 ± 0.67
	adaResNet, [17]	ResNet-12‡	56.88 ± 0.62	71.94 ± 0.57
	LEO, [25]	WRN-28-10 (pre)	61.76 ± 0.08	77.59 ± 0.12
	MTL, [30]	ResNet-12 (pre)	61.2 ± 1.8	75.5 ± 0.9
	MetaOpt-SVM, [10]†	ResNet-12	62.64 ± 0.61	78.63 ± 0.46
LST (Ours)	*recursive, hard, soft*	ResNet-12 (pre)	$\mathbf{70.1} \pm 1.9$	$\mathbf{78.7} \pm 0.8$

- Compared to the baseline method MTL [3], LST improves the accuracies by 8.9% and 3.2% respectively for 1-shot and 5-shot, which proves the efficiency of LST using unlabeled data.

Experiments

- Comparing with few-shot learning methods, on tieredImageNet dataset

Few-shot Learning Method		Backbone	tieredImageNet (test) 1-shot	tieredImageNet (test) 5-shot
Gradient descent	MAML, [3] (by [13])	ResNet-12	51.67 ± 1.81	70.30 ± 0.08
	LEO, [27]	WRN-28-10 (pre)	66.33 ± 0.05	81.44 ± 0.09
	MTL, [32] (by us)	ResNet-12 (pre)	65.6 ± 1.8	78.6 ± 0.9
	MetaOpt-SVM, [10]†	ResNet-12	65.99 ± 0.72	81.56 ± 0.53
LST (Ours)	*recursive, hard, soft*	ResNet-12 (pre)	$\mathbf{77.7} \pm 1.6$	$\mathbf{85.2} \pm 0.8$

  - Compared to the baseline method MTL [3], LST improves the results by 12.1% and 6.6% respectively for 1-shot and 5-shot.

Experiments

- Comparing with semi-supervised few-shot learning methods on two datasets

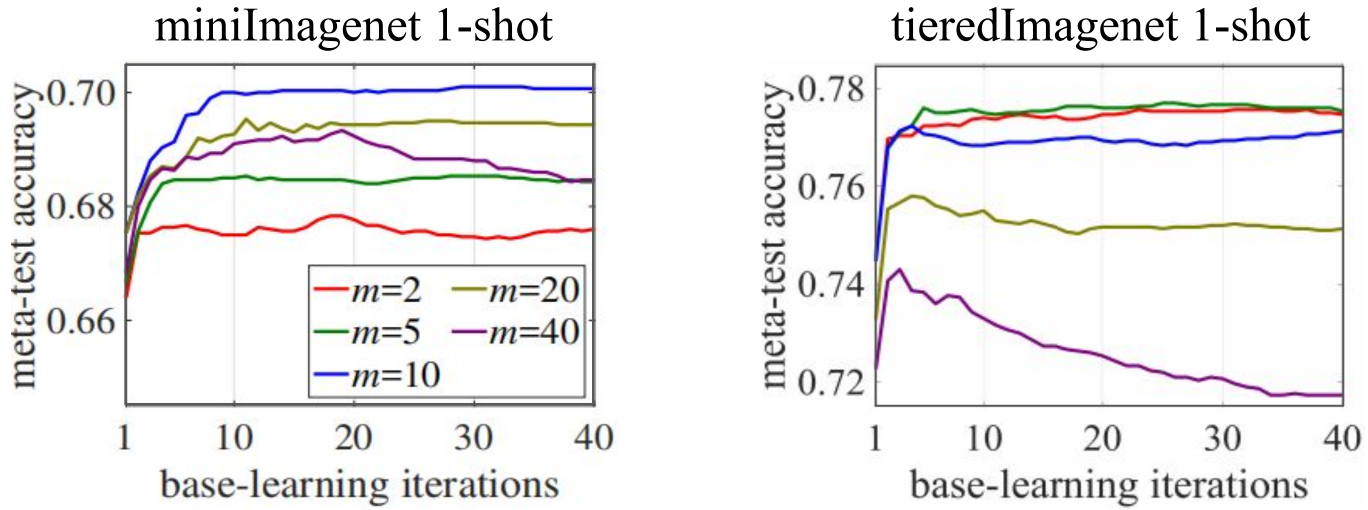
		mini		tiered		mini w/$\mathcal{D}$		tiered w/$\mathcal{D}$	
		1(shot)	5	1	5	1	5	1	5
fully supervised (upper bound)		80.4	83.3	86.5	88.7	-	-	-	-
no meta	*no selection*	59.7	75.2	67.4	81.1	54.4	73.3	66.1	79.4
	hard	63.0	76.3	69.8	81.5	61.6	75.3	68.8	81.1
	recursive,hard	64.6	77.2	72.1	82.4	61.2	75.7	68.3	81.1
meta	*hard* (Φ_{ss}, θ')	64.1	76.9	74.7	83.2	62.9	75.4	73.4	82.5
	soft	62.8	75.9	73.1	82.8	61.1	74.6	72.1	81.7
	hard,soft	65.0	77.8	75.4	83.4	63.7	76.2	**74.1**	82.9
	recursive,hard,soft	**70.1**	**78.7**	**77.7**	**85.2**	64.1	**77.4**	73.5	83.4
	mixing,hard,soft	66.2	77.9	75.6	84.6	**64.5**	76.5	73.6	**83.8**
Masked Soft k-Means *with* MTL		62.1	73.6	68.6	81.0	61.0	72.0	66.9	80.2
TPN *with* MTL		62.7	74.2	72.1	83.3	61.3	72.4	71.5	82.7
Masked Soft k-Means [24]		50.4	64.4	52.4	69.9	49.0	63.0	51.4	69.1
TPN [13]		52.8	66.4	55.7	71.0	50.4	64.9	53.5	69.9

Three LST models

w/$\mathcal{D}$ means using unlabeled data from 3 distracting classes.

Experiments

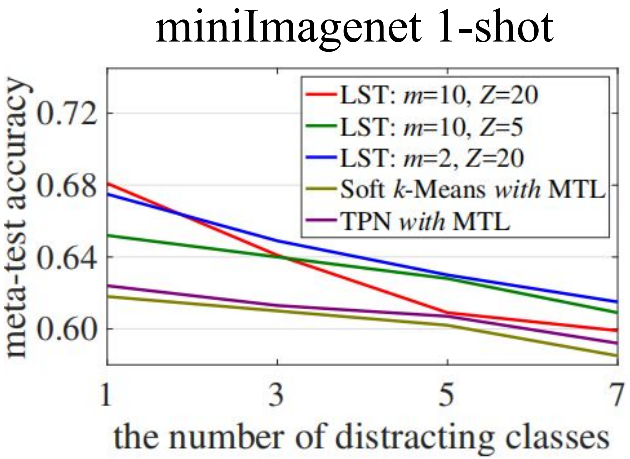
- The effect of the number of re-training steps m:

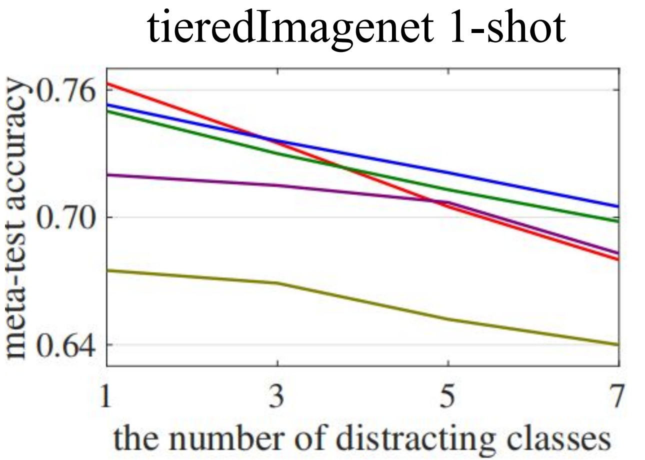


- Too many re-training steps, e.g. m=40, may lead to drifting problems and cause side effects on performance.

Experiments

- The effect of the number of distracting classes (1~7):





- LST achieves the top performance, especially more than 2% higher than TPN in the hardest case with 7 distracting classes.
- Among different settings, LST with less re-training steps, i.e., a smaller m value, works better for reducing the effect from a larger number of distracting classes.

References

[1] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[2] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. Meta-learning for semi-supervised few-shot classification. In *ICLR*, 2018.

[3] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *CVPR*, 2019.

[4] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NIPS*, 2016.

[5] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. Learning to compare: relation network for few-shot learning. In *CVPR*, 2018.