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

{lzx1217, 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** – minilImageNet 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 $S$
  - query set $Q$
  - 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 $\theta$ by gradient descent for $T$ iters:

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

Once $\theta_T$ 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:

1. Pseudo-labeling

   - Initialization
   - Training
   - Few-shot model

2. Cherry-picking

   - Unlabeled set \( \mathcal{R} \)
   - Predicting
   - Pseudo-labeled (noisy)
   - Hard selection
   - Soft weighting
   - Pseudo labeled set \( \mathcal{R}^p \) (selected → weighted)

3. Self-training

   - Initialization
   - Re-training
   - Re-trained model

Val./Test

   - Fine-tuning
   - Fine-tuned model
   - Query set \( Q \)
   - Loss or accuracy
   - Val. or test
3. Self-training

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

- 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, $\theta_t$ is trained as:

$$
\theta_t \leftarrow \theta_{t-1} - \alpha \nabla_{\theta_{t-1}} L(S \cup R^p; \Phi_{sw_m}, \Phi_{ss}, \theta_{t-1})
$$

$$
L(S \cup R^p; \Phi_{sw_m}, \Phi_{ss}, \theta_t) = \begin{cases} 
L_{ce}(f_{[\Phi_{sw_m},\Phi_{ss},\theta_t]}(x_i), y_i), & \text{if } (x_i, y_i) \in S \\
L_{ce}(w_i \odot f_{[\Phi_{sw_m},\Phi_{ss},\theta_t]}(x_i), y_i), & \text{if } (x_i, y_i) \in R^p
\end{cases}
$$

- In the rest $T - m$ steps, $\theta_t$ is fine-tuned with $S$ as:

$$
\theta_t \leftarrow \theta_{t-1} - \alpha \nabla_{\theta_{t-1}} L(S; \Phi_{sw_m}, \Phi_{ss}, \theta_{t-1})
$$
The framework of LST

- Outer loop with an inner loop:

After fine-tuning steps, using validation loss (on query set) to update $\Phi_{ss}$ and $\theta'$.

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

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

<table>
<thead>
<tr>
<th>Few-shot Learning Method</th>
<th>Backbone</th>
<th>miniImageNet (test)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1-shot</td>
</tr>
<tr>
<td><strong>Data augmentation</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delta-encoder, [27]</td>
<td>VGG-16 (pre)</td>
<td>58.7</td>
</tr>
<tr>
<td><strong>Gradient descent</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAML, [3]</td>
<td>4 CONV</td>
<td>48.70 ± 1.75</td>
</tr>
<tr>
<td>Meta-LSTM, [21]</td>
<td>4 CONV</td>
<td>43.56 ± 0.84</td>
</tr>
<tr>
<td>Bilevel Programming, [5]</td>
<td>ResNet-12</td>
<td>50.54 ± 0.85</td>
</tr>
<tr>
<td>MetaGAN, [41]</td>
<td>ResNet-12</td>
<td>52.71 ± 0.64</td>
</tr>
<tr>
<td>adaResNet, [17]</td>
<td>ResNet-12</td>
<td>56.88 ± 0.62</td>
</tr>
<tr>
<td>LEO, [25]</td>
<td>ResNet-12 (pre)</td>
<td>61.76 ± 0.08</td>
</tr>
<tr>
<td>MTL, [30]</td>
<td>ResNet-12 (pre)</td>
<td>61.2 ± 1.8</td>
</tr>
<tr>
<td>MetaOpt-SVM, [10]†</td>
<td>ResNet-12</td>
<td>62.64 ± 0.61</td>
</tr>
<tr>
<td><strong>LST (Ours)</strong></td>
<td>recursive, hard, soft</td>
<td>ResNet-12 (pre)</td>
</tr>
</tbody>
</table>

- 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

<table>
<thead>
<tr>
<th>Few-shot Learning Method</th>
<th>Backbone</th>
<th>tieredImageNet (test)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1-shot</td>
<td>5-shot</td>
</tr>
<tr>
<td>Gradient descent</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAML, [3] (by [13])</td>
<td>ResNet-12</td>
<td>51.67 ± 1.81</td>
<td>70.30 ± 0.08</td>
</tr>
<tr>
<td>LEO, [27]</td>
<td>WRN-28-10 (pre)</td>
<td>66.33 ± 0.05</td>
<td>81.44 ± 0.09</td>
</tr>
<tr>
<td>MTL, [32] (by us)</td>
<td>ResNet-12 (pre)</td>
<td>65.6 ± 1.8</td>
<td>78.6 ± 0.9</td>
</tr>
<tr>
<td>MetaOpt-SVM, [10]</td>
<td>ResNet-12</td>
<td>65.99 ± 0.72</td>
<td>81.56 ± 0.53</td>
</tr>
<tr>
<td>LST (Ours)</td>
<td>recursive, hard, soft</td>
<td>ResNet-12 (pre)</td>
<td><strong>77.7 ± 1.6</strong></td>
</tr>
</tbody>
</table>

- 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

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

*3 w/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):

  ![Graphs showing the effect of number of distracting classes on meta-test accuracy for miniImagenet and tieredImagenet 1-shot tasks.](image)

  - 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


