SOSELETO: A Unified Approach to Transfer Learning and Training with Noisy Labels

Or Litany

Facebook AI Research

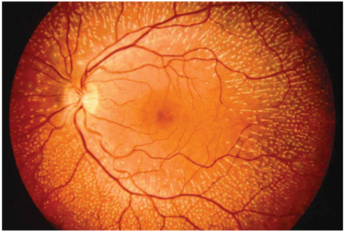
Joint work with Daniel Freedman (Google)

Background

- Deep Learning is data hungry

Background

- Deep Learning is data hungry
- What about data-poor regimes?



Transfer learning

Pass knowledge gleaned from a *source* (data-rich regime) to the *target* (data-poor regime)

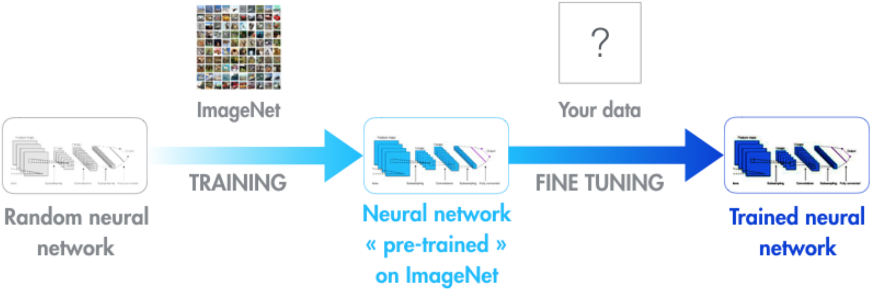


Image credit: Medium.com

Selective Transfer Learning

Observation

Some source examples are more informative than others for the target classification problem.

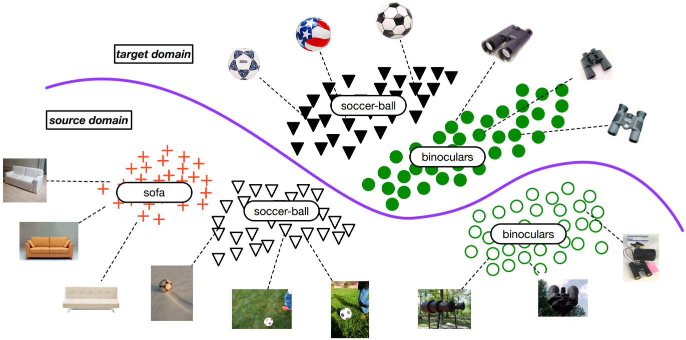
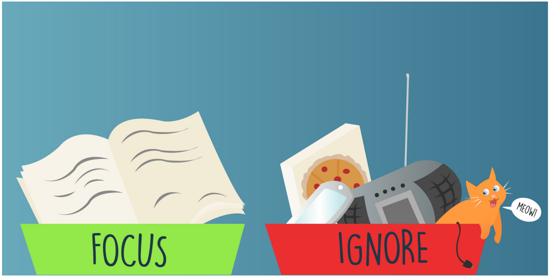


Image credit: Cao et al. 2018

Core idea

Problem

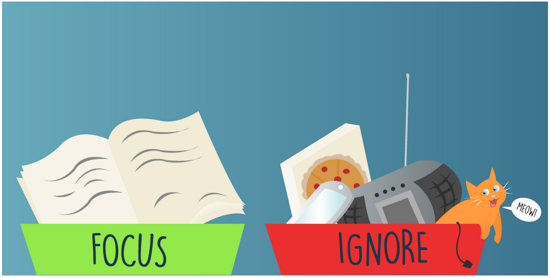
We do not know *a priori* which source examples will be important.



Core idea

Problem

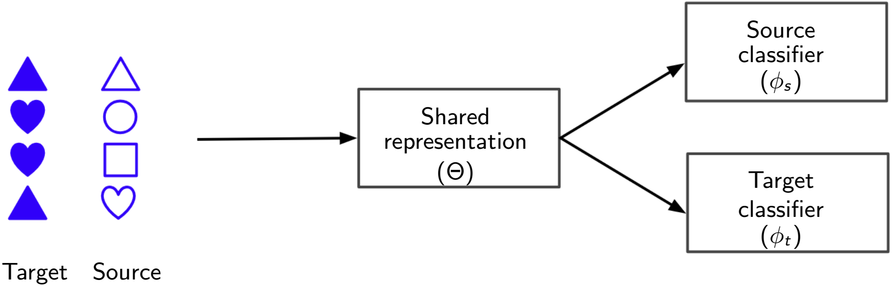
We do not know *a priori* which source examples will be important.



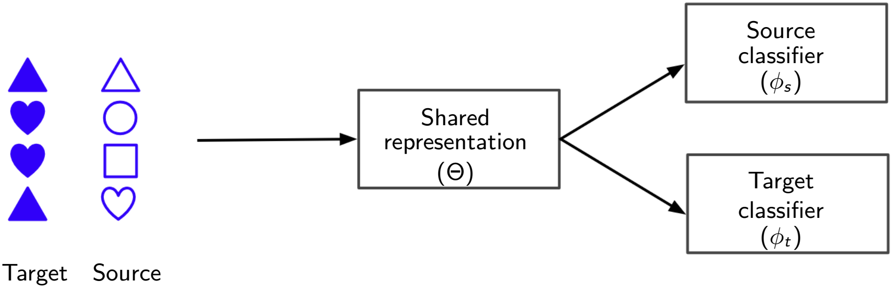
Proposal

What if we let the target decide?

SOSELETO: SOurce SELEction for Target Optimization



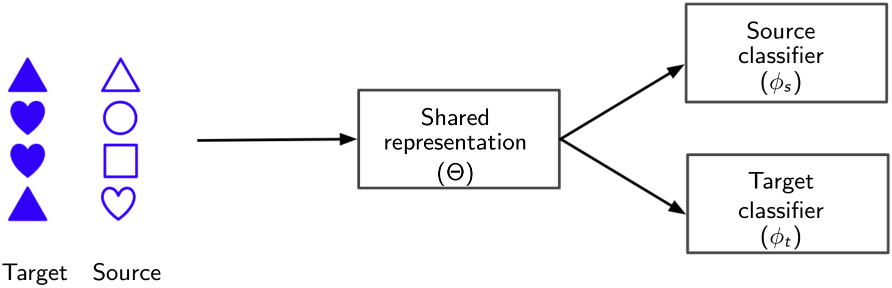
SOSELETO: SOurce SELEction for Target Optimization



$$\theta^*, \phi^{s*} = \arg\min_{\theta, \phi^s} L_s(\theta, \phi^s)$$

$$\phi^{t*} = \arg\min_{\phi^t} L_t(\theta^*, \phi^t)$$

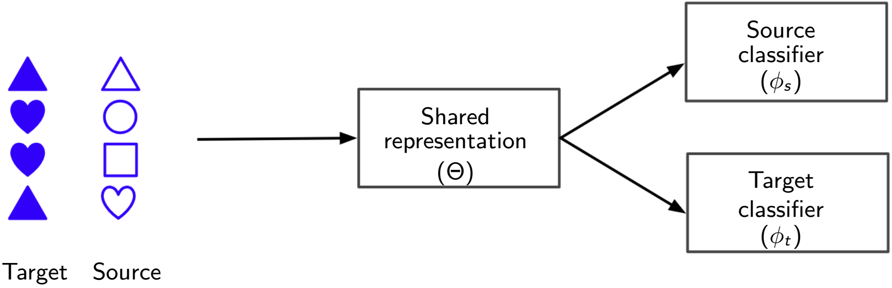
SOSELETO: SOurce SELEction for Target Optimization



$$\theta^*, \phi^{s*} = \arg\min_{\theta, \phi^s} \frac{1}{n^s} \sum_{j=1}^{n^s} \ell(y_j^s, F(x_j^s; \theta, \phi^s))$$

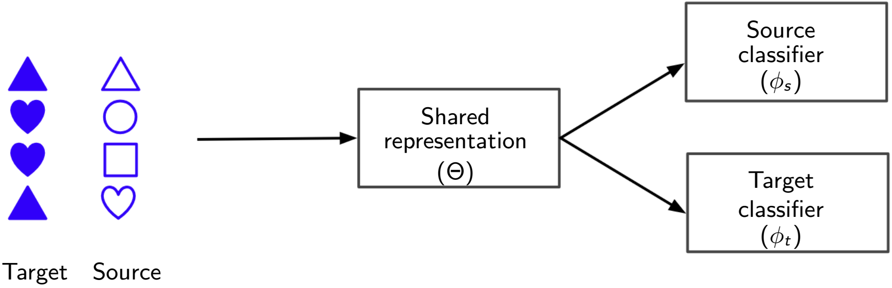
$$\phi^{t*} = \arg\min_{\phi^t} L_t(\theta^*, \phi^t)$$

SOSELETO: SOurce SELEction for Target Optimization



$$\theta^*(\alpha), \phi^{s*}(\alpha) = \arg\min_{\theta, \phi^s} \frac{1}{n^s} \sum_{j=1}^{n^s} \alpha_j \ell(y_j^s, F(x_j^s; \theta, \phi^s))$$

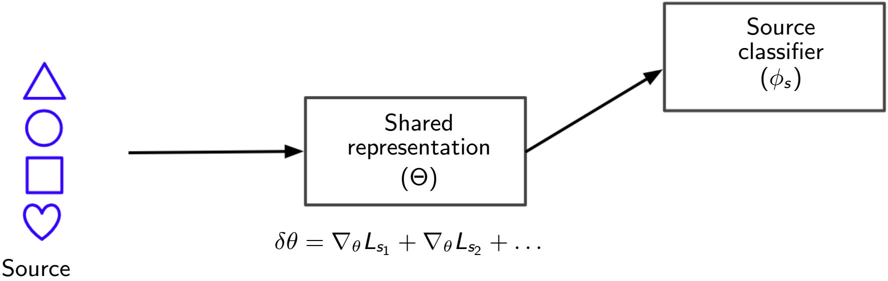
SOSELETO: SOurce SELEction for Target Optimization



$$\theta^*(\alpha), \phi^{s*}(\alpha) = \arg\min_{\theta, \phi^s} \frac{1}{n^s} \sum_{j=1}^{n^s} \alpha_j \ell(y_j^s, F(x_j^s; \theta, \phi^s))$$

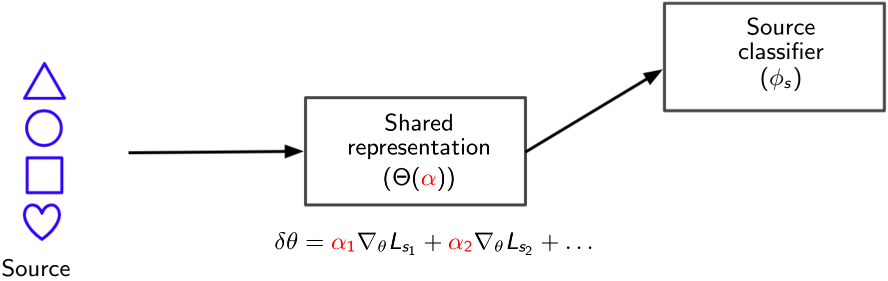
$$\phi^{t*}, \alpha^* = \arg\min_{\alpha, \phi^t} L_t(\theta^*(\alpha), \phi^t)$$

SOSELETO: SOurce SELEction for Target Optimization



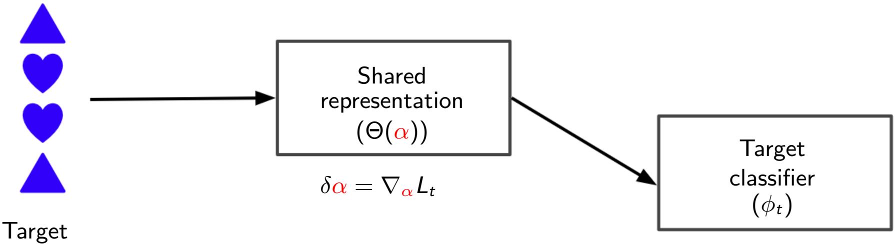
$$\delta\theta = \nabla_\theta L_{s_1} + \nabla_\theta L_{s_2} + \ldots$$

SOSELETO: SOurce SELEction for Target Optimization



$$\delta\theta = \alpha_1 \nabla_\theta L_{s_1} + \alpha_2 \nabla_\theta L_{s_2} + \dots$$

SOSELETO: SOurce SELEction for Target Optimization



Bi-level optimization

Interior level:

$$\theta^*(\alpha), \phi^{s*}(\alpha) = \arg\min_{\theta, \phi^s} L_s(\theta, \phi^s, \alpha)$$

Exterior level:

$$\min_{\alpha, \phi^t} L_t(\theta^*(\alpha), \phi^t)$$

Bi-level optimization

Interior level:

$$\theta^*(\alpha), \phi^{s*}(\alpha) = \arg\min_{\theta, \phi^s} L_s(\theta, \phi^s, \alpha)$$

$$\begin{aligned} \theta_{m+1} &= \theta_m - \lambda_p \frac{\partial L_s}{\partial \theta}(\theta_m, \phi^s_m, \alpha_m) \\ &= \theta_m - \lambda_p Q(\theta_m, \phi^s_m)\alpha_m \end{aligned} \quad (1)$$

Exterior level:

$$\min_{\alpha, \phi^t} L_t(\theta^*(\alpha), \phi^t)$$

Bi-level optimization

Interior level:

$$\theta^*(\alpha), \phi^{s*}(\alpha) = \arg\min_{\theta, \phi^s} L_s(\theta, \phi^s, \alpha)$$

$$\begin{aligned} \theta_{m+1} &= \theta_m - \lambda_p \frac{\partial L_s}{\partial \theta}(\theta_m, \phi^s_m, \alpha_m) \\ &= \theta_m - \lambda_p Q(\theta_m, \phi^s_m)\alpha_m \end{aligned} \tag{1}$$

Exterior level:

$$\min_{\alpha, \phi^t} L_t(\theta_m - \lambda_p Q\alpha, \phi^t)$$

Bi-level optimization

Interior level:

$$\theta^*(\alpha), \phi^{s*}(\alpha) = \arg\min_{\theta, \phi^s} L_s(\theta, \phi^s, \alpha)$$

$$\begin{aligned} \theta_{m+1} &= \theta_m - \lambda_p \frac{\partial L_s}{\partial \theta}(\theta_m, \phi^s_m, \alpha_m) \\ &= \theta_m - \lambda_p Q(\theta_m, \phi^s_m)\alpha_m \end{aligned} \tag{1}$$

Exterior level:

$$\min_{\alpha, \phi^t} L_t(\theta_m - \lambda_p Q\alpha, \phi^t)$$

$$\alpha_{m+1} \approx \alpha_m + \lambda_\alpha \lambda_p Q^T \frac{\partial L_t}{\partial \theta}(\theta_m) \tag{2}$$

Bi-level optimization

Interior level:

$$\theta^*(\alpha), \phi^{s*}(\alpha) = \arg\min_{\theta, \phi^s} L_s(\theta, \phi^s, \alpha)$$

$$\begin{aligned} \theta_{m+1} &= \theta_m - \lambda_p \frac{\partial L_s}{\partial \theta}(\theta_m, \phi^s_m, \alpha_m) \\ &= \theta_m - \lambda_p Q(\theta_m, \phi^s_m)\alpha_m \end{aligned} \tag{1}$$

Exterior level:

$$\min_{\alpha, \phi^t} L_t(\theta_m - \lambda_p Q\alpha, \phi^t)$$

$$\alpha_{m+1} = \mathrm{CLIP}_{[0,1]}\left(\alpha_m + \lambda_\alpha \lambda_p Q^T \frac{\partial L_t}{\partial \theta}(\theta_m)\right) \tag{2}$$

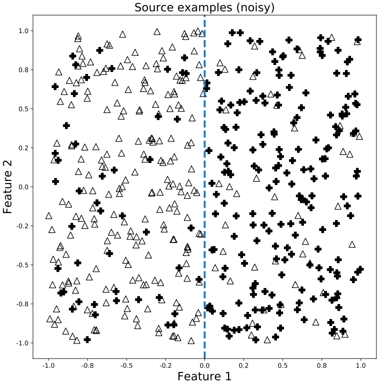
SOSELETO: SOurce SELEction for Target Optimization

1. Weigh source instances.
2. Train a shared representation, as a bi-level optimization:
   - *Interior level*: minimize source loss wrt representation parameters.
   - *Exterior level:* minimize target loss wrt source weights.

Intuition

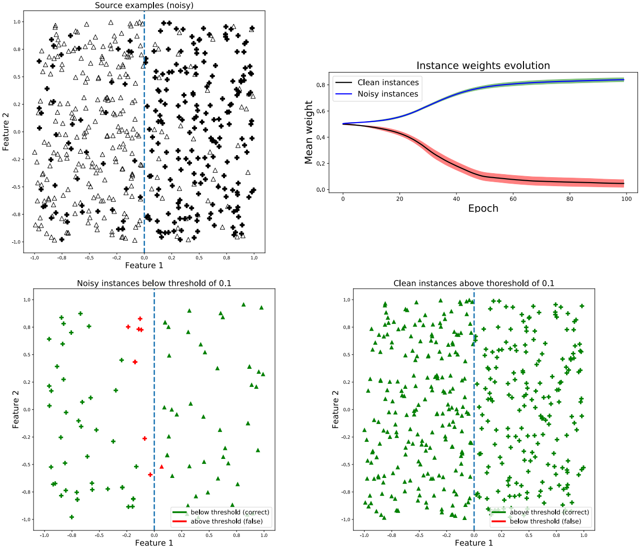
- The target set "chooses" source samples which are informative for its own classification task.
- bi-level optimization mitigates overfitting: target samples do not control the representation parameter directly.

Noisy Labels: synthetic experiment



- Source: 500 points with 20% noisy labels.
- Target: 50 points with clean labels

Noisy Labels: synthetic experiment



Results: Noisy labels (CIFAR-10)

- $60,000$ images of 10 categories (airplane, automobile, bird, etc.)
- Noise was added uniformly (unstructured)

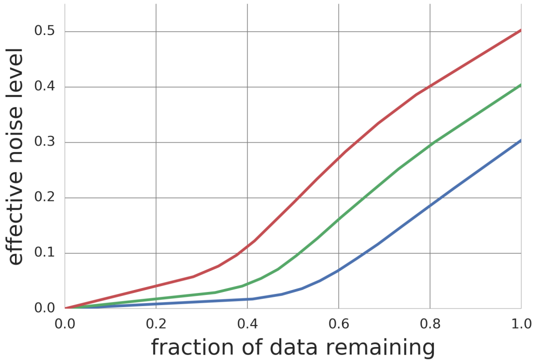
Results: Noisy labels (CIFAR-10)

- $60,000$ images of 10 categories (airplane, automobile, bird, etc.)
- Noise was added uniformly (unstructured)

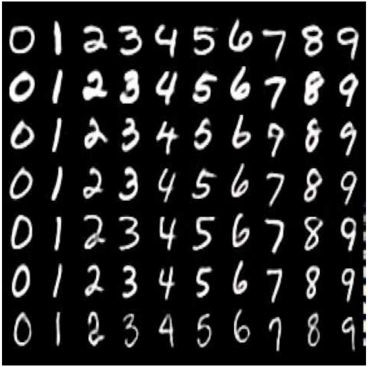
Noise Level	CIFAR-10 Quick	Sukhbaatar *et al.* 10K clean examples	Xiao *et al.* 10K clean examples	Ours 5K clean examples
30%	65.57	69.73	69.81	**72.41**
40%	62.38	66.66	66.76	**69.98**
50%	57.36	63.39	63.00	**66.33**

Results: Noisy labels (CIFAR-10)

A denoising effect:



SVHN 0-4 $\rightarrow$ MNIST 5-9



SVHN 0-4 $\rightarrow$ MNIST 5-9

Uses Unlabelled Data?	Method	$n^t = 20$	$n^t = 25$
No	Target only	80.1	84.0
No	Fine-tuning	80.2	83.0
No	SOSELETO	**83.2**	**87.9**
Yes	Matching Nets[1]	56.6	51.3
Yes	Fine-tuned Matching Nets	79.3	82.7
Yes	Fine-tune domain adversarial[2]	80.4	83.1
Yes	Label Efficient [2]	**94.2**	**95.0**

[1] Vinyals *et al.*, 2016

[2] Luo *et al.*, 2017

SVHN 0-4 $\rightarrow$ MNIST 5-9

Uses Unlabelled Data?	Method	$n^t = 20$	$n^t = 25$
No	Target only	80.1	84.0
No	Fine-tuning	80.2	83.0
No	SOSELETO	**83.2**	**87.9**
Yes	Matching Nets[1]	56.6	51.3
Yes	Fine-tuned Matching Nets	79.3	82.7
Yes	Fine-tune domain adversarial[2]	80.4	83.1
Yes	Label Efficient [2]	**94.2**	**95.0**

- Leveraging unlabeled MNIST: $\approx 92\%$

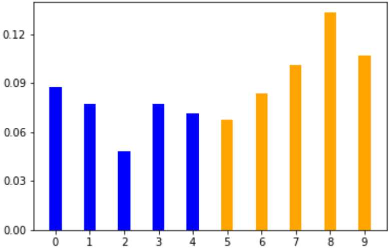
[1] Vinyals *et al.*, 2016
[2] Luo *et al.*, 2017

Partial overlap: SVHN 0-9 $\rightarrow$ MNIST 5-9

- Boosts performance to above 90%

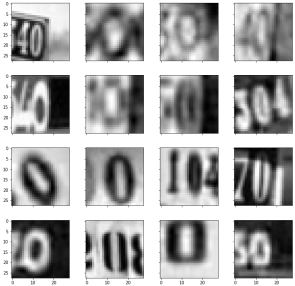
Partial overlap: SVHN 0-9 $\rightarrow$ MNIST 5-9

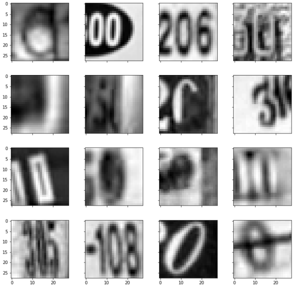
- Boosts performance to above 90%
- Distribution across classes: does not correlate with labels



Partial overlap: SVHN 0-9 → MNIST 5-9

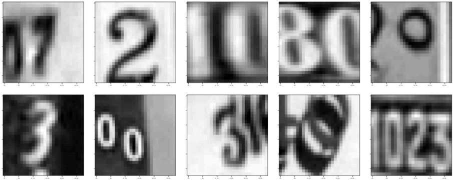
- Boosts performance to above 90%
- Distribution across classes: does not correlate with labels
- Some correlation to appearance





Partial overlap: SVHN 0-9 → MNIST 5-9

- Boosts performance to above 90%
- Distribution across classes: does not correlate with labels
- Some correlation to appearance
- Noisy lable detector



Summary

- Datasets and tasks share information (don't reinvent the mechanical turk)
- SOSELETO: Simple to implement, can be used with any architecture

Summary

- Datasets and tasks share information (don't reinvent the mechanical turk)
- SOSELETO: Simple to implement, can be used with any architecture
- Limitations:
  - Requires more memory (and adds variables)
  - Updates a sample once per epoch

Summary

- Datasets and tasks share information (don‘t reinvent the mechanical turk)
- SOSELETO: Simple to implement, can be used with any architecture
- Limitations:
  - Requires more memory (and adds variables)
  - Updates a sample once per epoch
- Follow ups:
  - Group weighting
  - Domain adaptation
  - Task weighting

Thanks!

github.com/orlitany