Meta-Learning Transferable Representations with a Single Target Domain

Hong Liu, Jeff Z. HaoChen, Colin Wei, Tengyu Ma

https://arxiv.org/pdf/2011.01418.pdf
https://openreview.net/forum?id=apiI1ySCSSR

Slides by: Paul Vicol

Transfer Learning

- **Transfer learning:** transferring knowledge learned from a large-scale source dataset to a small target dataset
  - *Hope* that the source and target tasks are related enough to learn features from the source data that are transferable to the target data
- In this paper, the assumption is that we have access to *both the source and target data at training-time*
  - E.g., both ImageNet (source) and a medical imaging dataset (target)
- They also assume that the target dataset is small
  - So that just training on the target training set would overfit and fail to generalize to the target test set
- Two common approaches: *fine-tuning and joint training*

Fine-Tuning

- Fine-tuning from pre-trained models has been extremely successful across vision and language tasks
  - ImageNet pretrained models are often useful for *object recognition, object detection, segmentation*
  - Pretraining *transformers on large text corpora* and fine-tuning leads to *SOTA results on many NLP tasks*
- However, there are cases where this approach fails:
  - Geirhos et al. found that pre-trained models *learn ImageNet textures, which are biased and not transferable to target tasks*
  - Also, ImageNet pretraining *does not necessarily improve acc on COCO, fine-trained classification, and medical imaging*

(a) Texture image
81.4% **Indian elephant**
10.3% indri
8.2% black swan

(b) Content image
71.1% **tabby cat**
17.3% grey fox
3.3% Siamese cat

(c) Texture-shape cue conflict
63.9% **Indian elephant**
26.4% indri
9.6% black swan

Notation and Formal Definitions

Source dataset: $\hat{\mathcal{D}}_s = \{x_i^s, y_i^s\}_{i=1}^{n_s}$

Target dataset: $\hat{\mathcal{D}}_t = \{x_i^t, y_i^t\}_{i=1}^{n_t}$

Feature extractor: $h_\phi(x)$

Source head: g_{θ_s}

Target head: g_{θ_t}

$$L_{\widehat{D}}(\theta, \phi) := \mathbb{E}_{(x,y)\in\widehat{D}}\ell(g_\theta(h_\phi(x)), y)$$

Target-only is the trivial algorithm that only trains on the target data $\widehat{\mathcal{D}}_t$ with the objective $L_{\widehat{\mathcal{D}}_t}(\theta_t, \phi)$ starting from random initialization. With insufficient target data, target-only is prone to overfitting.

Notation and Formal Definitions

Source dataset: $\hat{\mathcal{D}}_s = \{x_i^s, y_i^s\}_{i=1}^{n_s}$

Target dataset: $\hat{\mathcal{D}}_t = \{x_i^t, y_i^t\}_{i=1}^{n_t}$

Feature extractor: $h_\phi(x)$

Source head: g_{θ_s}

Target head: g_{θ_t}

$$L_{\widehat{D}}(\theta, \phi) := \mathbb{E}_{(x,y)\in\widehat{D}}\ell(g_\theta(h_\phi(x)), y)$$

Target-only is the trivial algorithm that only trains on the target data $\widehat{\mathcal{D}}_t$ with the objective $L_{\widehat{\mathcal{D}}_t}(\theta_t, \phi)$ starting from random initialization. With insufficient target data, target-only is prone to overfitting.

Pre-training starts with random initialization and *pre-trains* on the source dataset with objective function $L_{\widehat{\mathcal{D}}_s}(\theta_s, \phi)$ to obtain the pre-trained feature extractor $\hat{\phi}_{\text{pre}}$ and head $\hat{\theta}_s$.

Fine-tuning initializes the target head θ_t randomly and initializes the feature extractor ϕ by $\hat{\phi}_{\text{pre}}$ obtained in pre-training, and *fine-tunes* ϕ and θ_t on the target by optimizing $L_{\widehat{\mathcal{D}}_t}(\theta_t, \phi)$ over both θ_t and ϕ. Note that in this paper, fine-tuning refers to fine-tuning all layers by default.

Notation and Formal Definitions

Source dataset: $\hat{\mathcal{D}}_s = \{x_i^s, y_i^s\}_{i=1}^{n_s}$

Target dataset: $\hat{\mathcal{D}}_t = \{x_i^t, y_i^t\}_{i=1}^{n_t}$

Feature extractor: $h_\phi(x)$

Source head: g_{θ_s}

Target head: g_{θ_t}

$$L_{\widehat{D}}(\theta, \phi) := \mathbb{E}_{(x,y)\in\widehat{D}}\ell(g_\theta(h_\phi(x)), y)$$

Target-only is the trivial algorithm that only trains on the target data $\widehat{\mathcal{D}}_t$ with the objective $L_{\widehat{\mathcal{D}}_t}(\theta_t, \phi)$ starting from random initialization. With insufficient target data, target-only is prone to overfitting.

Pre-training starts with random initialization and *pre-trains* on the source dataset with objective function $L_{\widehat{\mathcal{D}}_s}(\theta_s, \phi)$ to obtain the pre-trained feature extractor $\hat{\phi}_{\text{pre}}$ and head $\hat{\theta}_s$.

Fine-tuning initializes the target head θ_t randomly and initializes the feature extractor ϕ by $\hat{\phi}_{\text{pre}}$ obtained in pre-training, and *fine-tunes* ϕ and θ_t on the target by optimizing $L_{\widehat{\mathcal{D}}_t}(\theta_t, \phi)$ over both θ_t and ϕ. Note that in this paper, fine-tuning refers to fine-tuning all layers by default.

Joint training starts with random initialization, and trains on the source and target dataset jointly by optimizing a linear combination of their objectives over the heads θ_s, θ_t and the shared feature extractor ϕ: $\min_{\theta_s,\theta_t,\phi} L_{\text{joint}}(\theta_s, \theta_t, \phi) := (1-\alpha)L_{\widehat{\mathcal{D}}_s}(\theta_s, \phi) + \alpha L_{\widehat{\mathcal{D}}_t}(\theta_t, \phi)$. The hyper-parameter α is used to balance source training and target training. We use cross-validation to select optimal α.

Problems with Fine-Tuning and Joint Training

- Pre-training is *not aware of the downstream (target) task*, so it does not have any incentive to learn transferable features
  - Thus we have to use both source and target data together to learn transferable representations
- Joint training sees both the source and target datasets, but it does not have a mechanism to ensure that the learned feature extractor + classification head actually generalize to the target test set
  - When the source-specific features are the most convenient for the source, *joint training simultaneously learns the source-specific features and memorizes the target dataset*

Contributions of this Paper

- Understand when and *why fine-tuning and joint training can be suboptimal or harmful for transfer learning*
  - *Introduces a synthetic dataset for investigation*
- Propose a new approach to meta-learning useful representations
  - Called *Meta Representation Learning, MeRLin*

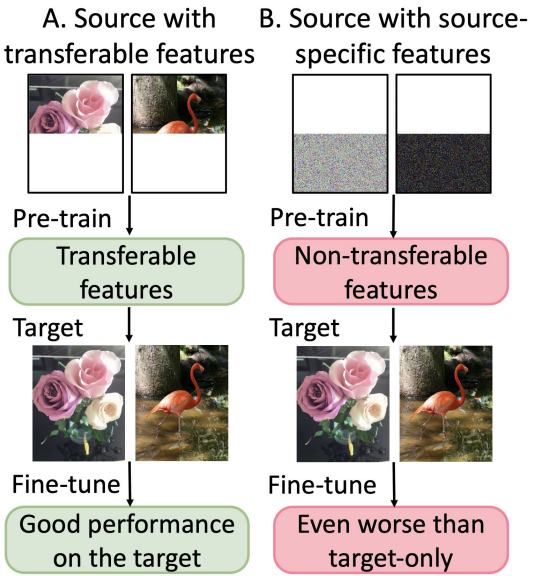
Synthetic Dataset

- Synthetic dataset where we know *which features are source-specific* and *which features are transferable*
- **Target training set:** subset of 500 CIFAR-10 trainset images
  - To simulate a setting with a small target dataset (e.g., medical images)
- **Target test set:** the full original CIFAR-10 test set (10,000 images)
- **Source training set:** the other 49,500 CIFAR-10 trainset images, *modified as follows*:

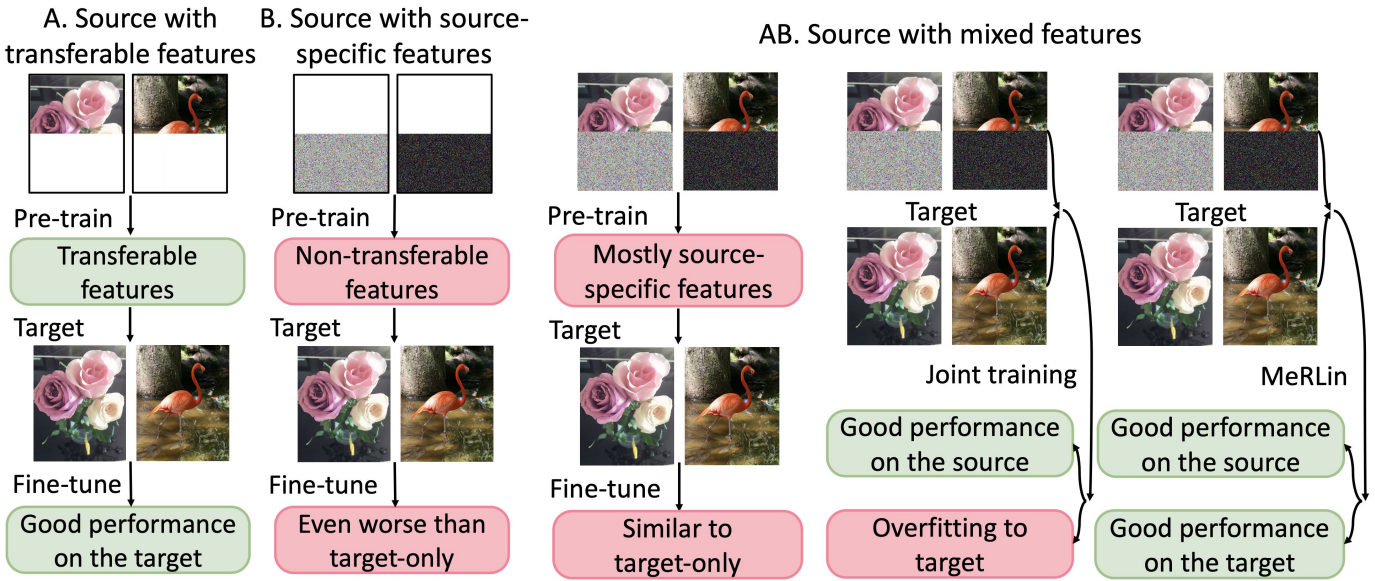
Upper half of each image is kept *unchanged*

Lower half contains a *"signature pattern" that strongly correlated with the class label*: for class c, the pixels are drawn iid from $\mathcal{N}(c/10, 0.2^2)$

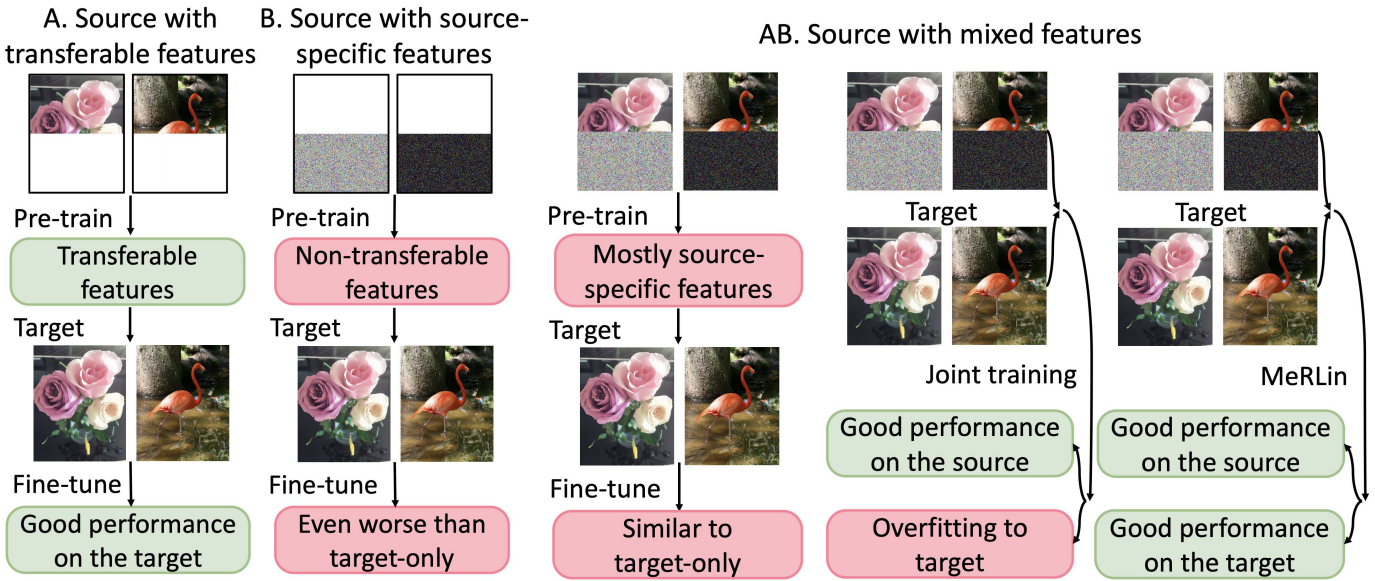
Fine-Tuning, Joint Training, and MeRLin on Synthetic Data

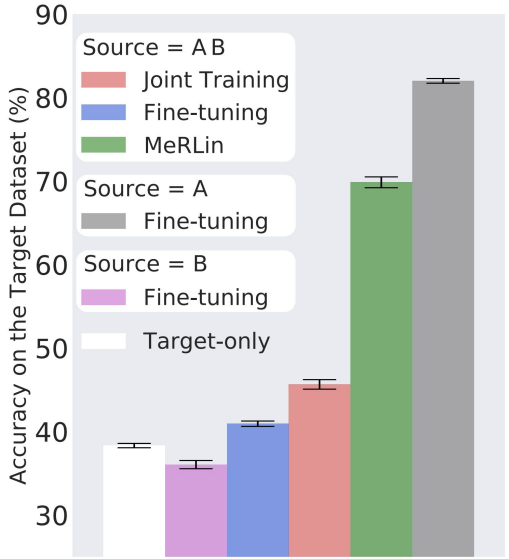


Fine-Tuning, Joint Training, and MeRLin on Synthetic Data



Fine-Tuning, Joint Training, and MeRLin on Synthetic Data





MeRLin: Meta Representation Learning

- **Idea:** Good representations should enable generalization
  - We should not only be able to fit a target head with the representations (as joint training does), but the *learned head should also generalize well to a held-out target validation set*
- **MeRLin:** A meta-representation learning algorithm that encourages the discovery of transferable features
- Split the target training set $\hat{\mathcal{D}}_t$ into *training and val partitions* $\hat{\mathcal{D}}_t^{tr}$, $\hat{\mathcal{D}}_t^{val}$
- **Goal:** Learn a feature extractor $h_\phi(x)$ such that the linear classifier trained on the features of the target training set generalizes the the target validation set
- This naturally leads to a *bilevel problem formulation*

MeRLin: Meta Representation Learning

- **Goal:** Learn a feature extractor $h_\phi(x)$ such that the linear classifier trained on the features of the target training set generalizes the the target validation set

Optimal linear classifier on the target training set, given feature extractor

$$\widehat{\theta}_t(\phi) = \arg\min_\theta L_{\widehat{\mathcal{D}}_t^{\mathrm{tr}}}(\theta, \phi)$$

Meta-objective containing the inner optimization over theta

$$L_{\mathrm{meta},t}(\phi) = L_{\widehat{\mathcal{D}}_t^{\mathrm{val}}}(\widehat{\theta}_t(\phi), \phi) = \mathbb{E}_{(x,y)\in\widehat{\mathcal{D}}_t^{\mathrm{val}}} \ell(g_{\widehat{\theta}_t(\phi)}(h_\phi(x)), y)$$

Overall objective

Learn the feature extractor on the source data as usual

Enforce that the features learned by h are useful s.t. A linear head can generalize to the target val set

$$\underset{\phi\in\mathbf{\Phi},\theta_s\in\mathbf{\Theta}}{\text{minimize}} \; L_{\mathrm{meta}}(\phi, \theta_s) := \overbrace{L_{\widehat{\mathcal{D}}_s}(\theta_s, \phi)} + \overbrace{\rho \cdot L_{\mathrm{meta},t}(\phi)}$$

MeRLin Algorithm

Algorithm 1 Meta Representation Learning (MeRLin).

Input: the source dataset $\widehat{\mathcal{D}}_s$ and the evolving target dataset $\widehat{\mathcal{D}}_t$.
Output: learned representations ϕ.
for $i = 0$ **to** `MaxIter` **do**
  Initialize the target head $\theta_t^{[0]}$.
  Randomly sample target train set $\widehat{\mathcal{D}}_t^{\text{tr}}$ and target validation set $\widehat{\mathcal{D}}_t^{\text{val}}$ from $\widehat{\mathcal{D}}_t$.
  for $k = 0$ **to** $n - 1$ **do**
    Train the target head on $\widehat{\mathcal{D}}_t^{\text{tr}}$:

$$\theta_t^{[k+1]} \leftarrow \theta_t^{[k]} - \eta \nabla_{\theta_t^{[k]}} L_{\widehat{\mathcal{D}}_t^{\text{tr}}}(\theta_t^{[k]}, \phi^{[i]}).$$

  end for
  In the outer loop, update the representation ϕ and the source head θ_s:

$$(\phi^{[i+1]}, \theta_s^{[i+1]}) \leftarrow (\phi^{[i]}, \theta_s^{[i]}) - \eta \nabla_{(\phi^{[i]}, \theta_s^{[i]}} \left[L_{\widehat{\mathcal{D}}_s}(\theta_s^{[i]}, \phi^{[i]}) + \rho L_{\widehat{\mathcal{D}}_t^{\text{val}}}(\theta_t^{[n]}, \phi^{[i]}) \right].$$

end for

MeRLin Algorithm

Algorithm 1 Meta Representation Learning (MeRLin).

Input: the source dataset $\widehat{\mathcal{D}}_s$ and the evolving target dataset $\widehat{\mathcal{D}}_t$.
Output: learned representations ϕ.
for $i = 0$ **to** `MaxIter` **do**
  Initialize the target head $\theta_t^{[0]}$.
  Randomly sample target train set $\widehat{\mathcal{D}}_t^{\text{tr}}$ and target validation set $\widehat{\mathcal{D}}_t^{\text{val}}$ from $\widehat{\mathcal{D}}_t$.
  for $k = 0$ **to** $n - 1$ **do**
    Train the target head on $\widehat{\mathcal{D}}_t^{\text{tr}}$:

$$\theta_t^{[k+1]} \leftarrow \theta_t^{[k]} - \eta \nabla_{\theta_t^{[k]}} L_{\widehat{\mathcal{D}}_t^{\text{tr}}}(\theta_t^{[k]}, \phi^{[i]}).$$

  end for
  In the outer loop, update the representation ϕ and the source head θ_s:

$$(\phi^{[i+1]}, \theta_s^{[i+1]}) \leftarrow (\phi^{[i]}, \theta_s^{[i]}) - \eta \nabla_{(\phi^{[i]}, \theta_s^{[i]})} \left[L_{\widehat{\mathcal{D}}_s}(\theta_s^{[i]}, \phi^{[i]}) + \rho L_{\widehat{\mathcal{D}}_t^{\text{val}}}(\theta_t^{[n]}, \phi^{[i]}) \right].$$

end for

They resample the train and val splits of the target dataset each iteration

Inner optimization

$$\hat{\theta}_t(\phi) = \arg\min_{\theta} L_{\hat{\mathcal{D}}_t^{tr}}(\theta, \phi)$$

Experiments: Vision Tasks

- *Object recognition problems with different label sets* (and different amounts of dissimilarity between domains)
- Using the *full objective* $\underset{\phi\in\mathbf{\Phi},\theta_s\in\mathbf{\Theta}}{\text{minimize}}\ L_{\text{meta}}(\phi,\theta_s) := L_{\widehat{\mathcal{D}}_s}(\theta_s,\phi) + \rho\cdot L_{\text{meta},t}(\phi)$

Table 1: **Accuracy** (%) on computer vision tasks.

Source	Fashion	SVHN	ImageNet			Food-101
Target	USPS (600)		CUB-200	Caltech-256	Stanford Cars	CUB-200
Target-only	91.07 ± 0.45	91.07 ± 0.45	32.05 ± 0.67	45.63 ± 1.26	23.22 ± 1.02	32.13 ± 0.64
Joint training	89.59 ± 0.56	91.54 ± 0.32	55.81 ± 1.36	78.20 ± 0.50	63.25 ± 0.72	42.08 ± 0.59
Fine-tuning	90.80 ± 0.20	92.12 ± 0.39	72.52 ± 0.51	81.12 ± 0.27	81.59 ± 0.49	52.30 ± 0.51
L2-sp	89.74 ± 0.41	91.86 ± 0.27	73.20 ± 0.38	82.31 ± 0.22	81.26 ± 0.27	53.84 ± 0.37
MeRLin	**93.34** ± 0.41	**93.10** ± 0.38	**75.42** ± 0.47	**82.45** ± 0.26	**83.68** ± 0.57	**58.68** ± 0.43

Experiments: NLP Tasks

- The source is a *language modeling task* and *targets are classification problems*
- Here, they didn't train BERT from scratch
  - Instead, they started from pre-trained BERT and only meta-learned the representation on the target domain $\phi = \arg\min_{\phi} L_{meta,t}(\phi) = \arg\min_{\phi} L_{\hat{\mathcal{D}}_t^{tr}}(\hat{\theta}_t(\phi), \phi)$
  - This is neat because it shows that you *don't actually have to train from scratch on the source + target datasets*

Table 2: **Accuracy** (%) of BERT-base on GLUE sub-tasks `dev` set.

Target	MRPC	RTE	QNLI
Fine-tuning	83.74 ± 0.93	68.35 ± 0.86	91.54 ± 0.25
L2-sp	84.31 ± 0.37	67.50 ± 0.62	91.29 ± 0.36
MeRLin-ft	**86.03** ± 0.25	**70.22** ± 0.86	**92.10** ± 0.27

Thank you!