# Smooth Transfer Learning for Source-to-Target Generalization

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Transfer learning for deep models has shown great success for various recognition tasks. Typically, a backbone network is pre-trained on a source dataset, then fine-tuned on a target dataset. We considered that when both datasets are at hand, learning them simultaneously at least for some period of iterations would yield higher test performance rather than the step-wise optimization. We propose Smooth Transfer Learning, which uses a learnable scheduler function for the loss coefficients so that degrees of contributions from two datasets can be smoothly changed along training time for optimal target performance. The scheduler function is designed so that it can express either pre-training-then-fine-tuning or multi-task learning with fixed weights as special cases. Our method consistently outperforms these special cases in object classification with CIFAR-10 and CIFAR-100, and in digit classification with SVHN and MNIST.

## 1 Introduction

It is widely known that domain gap between the training and test deteriorates generalizability of a machine-learning model. If, however, one has even a small dataset from the target domain at training time, there is a higher chance to greatly improve the model performance. Transfer learning is one of the popular methods [1, 2, 3, 4] for such cases. The popularity stems from the fact that the model fine-tuning on a target dataset likely improves performance compared to direct training on a target dataset from random initialization. The performance gain is reasoned by the generalizability of feature extractors obtained by the pre-training [3], especially when a massive source dataset is used.

However, the performance gain becomes negligible [5] or even negative [6] under certain conditions. He *et al.* have shown evidences [5] that direct training on a target detection dataset with sufficiently many iterations yields no clear performance drop compared to fine-tuning after pre-training on a large dataset such as ImageNet [7]. Fine-tuning can even deteriorate the performance when source and target domains significantly differ [6]. This type of conflicting settings is disregarded in this work.

*Is it best to adopt the step-wise optimization; i.e., fine-tuning on a target dataset after pre-training on a source dataset ("pre-training-then-fine-tuning")?* We hypothesized that when both datasets are at hand, learning both datasets simultaneously for an entire period, or at least for some period of iterations would yield higher test performance rather than the step-wise optimization. Switching datasets in the middle of training almost surely deteriorates performance for the source domain towards the end; therefore, it may limit the generalizability of the feature extractors. When data from both datasets are used for update, the task required by the source dataset likely works as *regularizer*, which is somewhat similar to auxiliary training [8].

A non-triviality about the above idea of simultaneous learning is the asymmetry between the training and test objectives. The training objective now consists of two sub-objectives from the source and

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

target tasks, while the test objective is solely from the target task. Naive minimization of weighted loss functions with either constant weights [9] or with adaptive weights [10, 11] as seen in multi-task learning may not be the best strategy for the test objective. In this work, we employ hyperparameter optmization scheme [12], as it can handle an arbitrary metric for the validation dataset, to optimize a specially-designed, learnable scheduler function for the source and target loss weights.

Contributions of this work are summarized as follows:

- We propose Smooth Transfer Learning that generalizes the popular practice of pre-training-then-fine-tuning by introducing a specially-designed, learnable scheduler function for source and target loss weights. The proposed method can express pre-training-then-fine-tuning and multi-task learning with fixed weights as special cases.
- We demonstrate that the proposed method consistently outperforms pre-training-then-fine-tuning and multi-task learning with fixed weights for object classification task using CIFAR-10 and CIFAR-100 datasets [13] and digit classification task using SVHN [14] and MNIST [15] datasets.
- We report intriguing tendencies found by experiments such that the semi-optimized scheduler function approaches pre-training-then-fine-tuning behavior as the size of the target dataset becomes small, and approaches the behavior of multi-task learning with fixed weights as the size of the target dataset becomes large. In the intermediate region, the semi-optimized scheduler function prefers a transition period that gradually switch from the source to the target dataset.

## 2 Smooth Transfer Learning

We assume that a source dataset $D^{\mathrm{S}}$ and a target dataset for classification are given. The target dataset is splitted into a training set $D^{\mathrm{T}}_{\mathrm{tr}}$ and a validation set $D^{\mathrm{T}}_{\mathrm{va}}$. The objective is to generate a network that yields high test performance on the target domain. To simplify notations, we assume the dimension of an input $x$ and that of a target variable $y$, which is a one-hot vector indicating the true class label, are the same between the target and source datasets, while one can easily extend different dimensionalities.

We propose Smooth Transfer Learning, which uses a specially-designed, learnable scheduler function for the loss weights such that degrees of contributions from two datasets can be smoothly changed along training time. The form of the scheduler function is designed with only a few learnable parameters so that it can express either naive pre-training-then-fine-tuning or multi-task learning with fixed weights; therefore, the proposed method is a generalization of these extreme cases. The difficulty on optimizing such a scheduler lies in the asymmetry between the training objective for the network (source and target) and the test objective (target). To address this asymmetry, we employ a hyperparameter search to optimize the parameters of the scheduler function for the *target* domain.

The network architecture considered in this work has a domain-generic backbone $g_\theta(\cdot)$ with a parameter set $\theta$ and two parallel domain-specific heads $h^{\mathrm{T}}_{\phi^{\mathrm{T}}}(\cdot), h^{\mathrm{S}}_{\phi^{\mathrm{S}}}(\cdot)$ with target-specific parameter set $\phi^{\mathrm{T}}$ and source-specific parameter set $\phi^{\mathrm{S}}$. The sample-wise loss functions are defined as

$$\mathcal{L}^{\mathrm{T}}_{\theta,\phi^{\mathrm{T}}}(x,y) = L\left(s(h^{\mathrm{T}}_{\phi^{\mathrm{T}}}(g_\theta(x))), y\right), (x,y) \in D^{\mathrm{T}}_{\mathrm{tr}}, \tag{1}$$

$$\mathcal{L}^{\mathrm{S}}_{\theta,\phi^{\mathrm{S}}}(x,y) = L\left(s(h^{\mathrm{S}}_{\phi^{\mathrm{S}}}(g_\theta(x))), y\right), (x,y) \in D^{\mathrm{S}}, \tag{2}$$

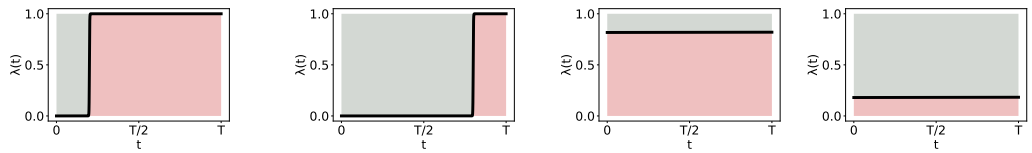
where $s(\cdot)$ denotes the softmax function and $L(\cdot)$ denotes the cross-entropy loss function. The mini-batch loss function is then defined as

$$\mathcal{L}_{\Theta,\lambda_0}(B^{\mathrm{T}}, B^{\mathrm{S}}) = \frac{\lambda_0}{|B^{\mathrm{T}}|} \sum_{(x,y)\in B^{\mathrm{T}}} \mathcal{L}^{\mathrm{T}}_{\theta,\phi^{\mathrm{T}}}(x,y) + \frac{1-\lambda_0}{|B^{\mathrm{S}}|} \sum_{(x,y)\in B^{\mathrm{S}}} \mathcal{L}^{\mathrm{S}}_{\theta,\phi^{\mathrm{S}}}(x,y). \tag{3}$$

Here, a mini-batch consists of a sub-batch $B^{\mathrm{T}}$ from the target dataset $D^{\mathrm{T}}_{\mathrm{tr}}$ and a sub-batch $B^{\mathrm{S}}$ from the source dataset $D^{\mathrm{S}}$. The set $\Theta$ consists of $\theta, \phi^{\mathrm{T}}$ and $\phi^{\mathrm{S}}$. The degrees of contributions from two sub-batches are controlled by the loss weight $\lambda_0 \in [0,1]$, which we will extend to a function of training time $t$ (in the units of iterations) as discussed below.

We define the scheduler function for the loss weight as the sigmoid function of the form

$$\lambda_\omega(t) = \left[1 + \exp\left(-\frac{t - T/2 - t_0}{\tau} + \alpha\right)\right]^{-1}, \tag{4}$$




(a) $|\tau| \ll 1, t_0 = -0.3T$ (b) $|\tau| \ll 1, t_0 = 0.3T$ (c) $\tau \gg 1, \alpha = -1.5$ (d) $\tau \gg 1, \alpha = 1.5$

Figure 1: Extreme cases of the scheduler function $\lambda_\omega(t)$. (a,b) Pre-training-then-fine-tuning with different switching timings. (c,d) Multi-task learning with different fixed weights.

where $T$ is the total number of iterations and $\omega = \{t_0, \tau, \alpha\}$ denotes a set of tunable scalars. Now, we illustrate that this scheduler function can realize the following extreme cases.

**Pre-training-then-fine-tuning:** In the limit of $\tau \to 0$ with a finite $\alpha$,

$$\lambda_\omega(t) = \begin{cases} 0, & t < T/2 + t_0 \\ 1, & t > T/2 + t_0. \end{cases} \tag{5}$$

In this extreme case, $\lambda_\omega(t)$ essentially behaves as a step function that changes the value at $T/2 + t_0$. In other words, the training process boils down to the conventional pre-training-then-fine-tuning, where the transfer timing is controlled by $t_0$, as illustrated in Fig. 1 (a) and (b).

**Multi-task learning with fixed weights:** In the limit of $\tau \to \infty$ with a finite $t_0$,

$$\lambda_\omega(t) = (1 + e^{\alpha})^{-1}, \tag{6}$$

which is just a constant function, whose value is controlled by $\alpha$. This tells that multi-task learning with fixed weights can be expressed in this limit, as illustrated in Fig. 1 (c) and (d).

It is worth mentioning about the parametrization of $\lambda_\omega(t)$. A sigmoid function is typically expressed with two parameters of gain and offset. The reason why we introduced the third one is to characterize each parameter to carry a specific meaning. Namely, the gain $\tau$ is responsible for the maximum slope, the gain-independent offset $t_0$ is responsible for the transfer timing, and the gain-dependent offset $\alpha$ is responsible for the fixed weight. We could eliminate, say $\alpha$, but it requires a very complex parameter tuning for both $\tau$ and $t_0$ such that $t_0 = \alpha'\tau \to \infty$ to express a constant function.

We use a hyperparameter search to optimize the scheduler parameters $\omega$ for the validation dataset of the target domain. With a trial $\omega$, we apply the following update rule or its variant to the parameter $\Theta$

$$\textbf{For } t = 1, \cdots, T: \quad \Theta \leftarrow \Theta - \eta \frac{\partial}{\partial \Theta} \mathcal{L}_{\Theta, \lambda_\omega(t)}(B_t^{\mathrm{T}}, B_t^{\mathrm{S}}), \tag{7}$$

where $B_t^{\mathrm{T}}$ and $B_t^{\mathrm{S}}$ are uniformly sampled sub-batches from $D_{\mathrm{tr}}^{\mathrm{T}}$ and $D^{\mathrm{S}}$, respectively. Temporal model generation is repeated many times with different trials of $\omega$ to search for the best $\omega$ and $\Theta$ in terms of validation accuracy in the target domain. We employ Tree-structured Parzen Estimator (TPE) [16] for the search. The gain $\tau$ is searched in log domain, and other two offsets are searched in linear domain so that parameter setting close to the two extreme cases can be sufficiently searched.

# 3 Evaluation

The experimental purpose is to compare the test performance of models trained by Smooth Transfer Learning and simple baselines including pre-training-then-fine-tuning and multi-task learning with fixed weights. We are also interested in how a semi-optimal scheduler behaves for $|D_{\mathrm{tr}}^{\mathrm{T}}| \ll |D^{\mathrm{S}}|$ and $|D_{\mathrm{tr}}^{\mathrm{T}}| \simeq |D^{\mathrm{S}}|$. In the former case where the target dataset is much smaller than the source dataset, we expect that it approaches pre-training-then-fine-tuning behavior, since fine-tuning has been found to be beneficial for this case. In the latter case, the maximum slope would be much more moderate.

We conducted two sets of experiments. One is for object classification using CIFAR-10 [13] as the target dataset and CIFAR-100 [13] as the source dataset. The other is for digit classification using SVHN [14] as the target and MNIST [15] as the source. Training details are given in Appendix.

Quantitative results for generalization abilities are summarized in Table 1. Recalling that the proposed method is a generalization of the baselines and the hyperparameter search yields semi-optimal

Table 1: Test accuracies (%) on the target domain for different target dataset size $N = |D^{\mathrm{T}}_{\mathrm{tr}}|$. Top row is the case of $N \simeq |D^{\mathrm{S}}|$. 'Target' means models trained only on target dataset $D^{\mathrm{T}}_{\mathrm{tr}}$. 'Fine-tune' means models produced by pre-training-then-fine-tuning. 'Multi-task' means models trained with the target and source losses with fixed weights. 'Ours' means the proposed Smooth Transfer Learning.

[Target: CIFAR-10, Source: CIFAR-100]

| $N$ | Target | Fine-tune | Multi-task | Ours |
|---|---|---|---|---|
| $45,000$ | 91.9 | 88.7 | 90.3 | **92.2** |
| $14,230$ | 87.0 | 85.3 | 87.0 | **89.1** |
| $4,500$ | 79.0 | 81.1 | 82.1 | **85.4** |
| $1,420$ | 66.0 | 75.7 | 75.1 | **80.4** |
| $450$ | 43.9 | 69.9 | 67.4 | **72.8** |
| $140$ | 36.7 | 61.1 | 61.0 | **65.3** |

[Target: SVHN, Source: MNIST]

| $N$ | Target | Fine-tune | Multi-task | Ours |
|---|---|---|---|---|
| $65,931$ | 95.8 | 95.3 | **96.1** | 96.0 |
| $20,849$ | 93.8 | 94.2 | 94.4 | **94.5** |
| $6,593$ | 91.5 | 92.1 | 92.1 | **92.8** |
| $2,085$ | 84.8 | 88.5 | 87.8 | **90.5** |
| $659$ | 47.2 | 75.0 | 63.0 | **85.0** |
| $208$ | 18.4 | 45.3 | 22.2 | **66.0** |

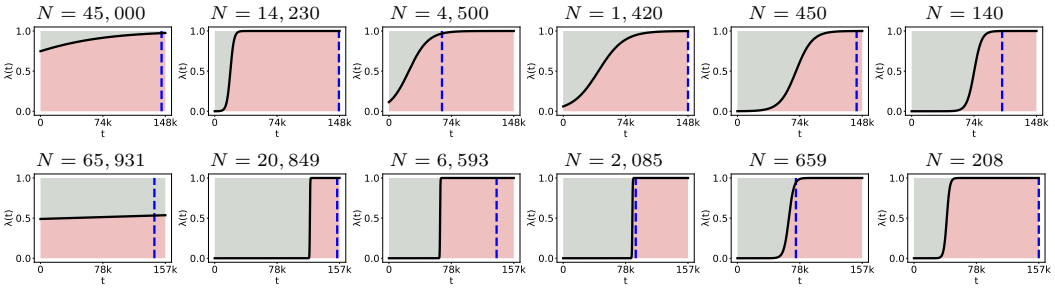



Figure 2: The semi-optimal scheduler functions $\lambda_\omega(t)$ obtained for the object classification task (top) and for the digit classification task (bottom). The dashed blue lines indicate the early-stopped timings. Roughly speaking, the scheduler approaches pre-training-then-fine-tuning (multi-task learning with fixed weights) when $N$ is small (large).

hyperparameters for the validation set, it is no surprise that ours outperforms the baselines. One can observe that 'Fine-tuning' ('Multi-task') tends to perform better than 'Multi-task' ('Fine-tune') when the target dataset size is much smaller than (similar to) the source dataset size. When the target dataset size is relatively small, having many iterations with the target samples would easily cause overfitting. Thus, switching the datasets in the middle of the training would be a better strategy than to keep using the target samples from the beginning with a fixed loss weight. In contrast, when the target dataset is as large as the source dataset, target samples can be trained longer without experiencing overfitting. Dataset switching would not be ideal in this case because forgetting of the source dataset likely happens.

Obtained scheduler functions are shown in Fig. 2. Although the number of trials by TPE is not large enough in these preliminary experiments[1] to yield truly the best validation performance, we still observe following tendencies. Roughly speaking, the semi-optimal scheduler approaches pre-training-then-fine-tuning (multi-task learning with fixed weights) when the target dataset size is much smaller than (similar to) the source dataset size. In the object classification task with non-extreme $N$ (say $N = 1,420$), the transition is quite smooth; $\lambda_\omega(t)$ changes from a value $\simeq 0$ to a value $\simeq 1$ slowly over the entire period of training.

## 4 Conclusion

We proposed Smooth Transfer Learning, which uses a learnable scheduler function for the loss weight. The scheduler function is designed to generalize the pre-training-then-fine-tuning and multi-task learning with fixed weights. We confirmed that our method outperforms these special cases in image classification tasks. Obtained scheduler functions nearly behave as the pre-training-then-fine-tuning (multi-task learning with fixed weights) when the target dataset size is much smaller than (similar to) the source dataset size.

[1]The average number of replacing the best performing parameter set $\omega$ by TPE was 5.8.

# References


[1] Sinno Jialin Pan and Qiang Yang. A Survey on Transfer Learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[2] Karl Weiss, Taghi M. Khoshgoftaar, and Ding Ding Wang. A survey of transfer learning. *Journal of Big Data*, 3(9), 2016.

[3] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International Conference on Artificial Neural Networks (ICANN)*, 2018.

[4] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE*, 109(1):43–76, 2021.

[5] Kaiming He, Ross Girshick, and Piotr Dollar. Rethinking imageNet pre-training. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-October(i):4917–4926, 2019.

[6] Wen Zhang, Lingfei Deng, and Dongrui Wu. Overcoming negative transfer: A survey. *CoRR*, abs/2009.00909, 2020.

[7] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[8] Linfeng Zhang, Muzhou Yu, Tong Chen, Zuoqiang Shi, Chenglong Bao, and Kaisheng Ma. Auxiliary training: Towards accurate and robust models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[9] Iasonas Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5454–5463, 2017.

[10] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[11] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, pages 794–803. PMLR, 2018.

[12] Tong Yu and Hong Zhu. Hyper-parameter optimization: A review of algorithms and applications. *CoRR*, abs/2003.05689, 2020.

[13] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto, Technical Report*, 2009.

[14] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[15] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.

[16] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, volume 24, 2011.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[18] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *The British Machine Vision Conference*, 2016.

# A Training details

## A.1 Training details specific to the object classification task

We use ResNet20 [17] architecture in all experiments of the object classification task.

Among the target CIFAR-10 samples, we randomly split 50,000 default training samples into 45,000 training samples and 5,000 validation samples. From the 45,000 training samples, we further construct partial datasets of different sizes $N$ by random sampling to examine how performance differs for different training dataset sizes.

We use the same sub-batch size 128 for the source and target in the multi-task learning and Smooth Transfer Learning. Other than those, we use the mini-batch size of 128, except for the single dataset training with $N = 140$ and the fine-tuning with $N = 140$, where the mini-batch size 16 is used.

We run the single dataset training and the source dataset training in the conventional transfer learning for 200 epochs. Duration of the multi-task learning and that of Smooth Transfer Learning is 380 epochs with respect to the source dataset.

The learning rate is scheduled by the cosine annealing rule. The peak and final learning rates are 0.1 and 1e-3, respectively. Weight decay rate is 1e-4. We do not use dropout in the ResNet20.

We used batch normalization in all experiments of the object classification task. In the multi-task learning and Smooth Transfer Learning, batch statistics are shared between the source and target datasets.

TPE setting is as follows. We use 11 workers that run in parallel. Each worker is responsible for training one model at a time. We repeat the parallel training 9 times, and at each time a worker generates a trial parameter set $\omega$ which stems from the one that performs best so far on the validation dataset.

## A.2 Training details specific to the digit classification task

We use WideResNet-10-2 [18] architecture in all experiments of the digit classification task.

Among the target SVHN samples, we split 73,257 default training samples into 65,931 training samples (90%) and 7,326 validation samples (10%). From the 65,931 training samples, we further construct partial datasets of different sizes $N$ by random sampling to examine how performance differs for different training dataset sizes.

We use the same sub-batch size 128 for the source and target in the multi-task learning and Smooth Transfer Learning. Other than those, we use the mini-batch size of 128.

We run the single dataset training and the source dataset training in the conventional transfer learning for 160 epochs. Duration of the multi-task learning and that of Smooth Transfer Learning is 336 epochs with respect to the source dataset.

The learning rate is scheduled by the cosine annealing rule. The peak and final learning rates are 1e-2 and 1e-4, respectively. Weight decay rate is 5e-4. Dropout ratio used in the WideResNet-10-2 is 0.4.

We used batch normalization in all experiments of the digit classification task. In the multi-task learning and Smooth Transfer Learning, dataset-specific batch statistics are used; *i.e.*, the source and target datasets have different sets of batch statistics. We checked that use of shared batch statistics brings noticeable drops in test accuracies.

As for TPE, we used 9 workers which sequentially train models 11 times.

## A.3 Training details common in all experiments

We adopt cosine annealing for learning rate scheduling. Linear warmup is adopted in the beginning part of training for a period of 2.5% of total duration. We used momentum SGD with momentum rate of 0.9.

Early stopping is applied on each experiment. Model parameters are saved as certain intervals during total training epochs mentioned above and a model that yields best validation accuracy is tested.