LEARNING TO TRANSFER LEARN

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose learning to transfer learn (L2TL) to improve transfer learning on a target dataset by judicious extraction of information from a source dataset. L2TL considers joint optimization of vastly-shared weights between models for source and target tasks, and employs adaptive weights for scaling of constituent losses. The adaptation of the weights is based on reinforcement learning, guided with a performance metric on the target validation set. We demonstrate state-of-the-art performance of L2TL given fixed models, consistently outperforming fine-tuning baselines on various datasets. In the regimes of small-scale target datasets and significant label mismatch between source and target datasets, L2TL outperforms previous work by an even larger margin.

1 INTRODUCTION

Deep neural networks excel at understanding images (He et al., 2016; Simonyan and Zisserman, 2015; Zagoruyko and Komodakis, 2016), text (Conneau et al., 2017; Devlin et al., 2018; Lai et al., 2015) and audio (van den Oord et al., 2016; Amodei et al., 2016; Chiu et al., 2018). The performance of deep neural networks improves significantly with more training data (Hestness et al., 2017). As the applications of deep neural networks diversify and span use cases with small training datasets, conventional training approaches are often insufficient to yield high performance. It becomes highly beneficial to utilize extra source datasets and "transfer" the relevant information to the target training dataset. Transfer learning, commonly in the form of obtaining a pre-trained model on a large-scale source dataset and then further training it on the target dataset (known as fine-tuning), has become the standard recipe for most real-world artificial intelligence applications. Compared to training from random initialization, fine-tuning yields considerable performance improvements and convergence speedup, as demonstrated for object recognition (Razavian et al., 2014), semantic segmentation (Long et al., 2015), language understanding (Devlin et al., 2018), speech synthesis (Arik et al., 2018), audio-visual recognition (Moon et al., 2014) and language translation (Zoph et al., 2016).

Towards the motivation of pushing the performance of transfer learning, recent studies (Ngiam et al., 2018; Mahajan et al., 2018; Lee et al., 2019) have explored the direction of matching the source and target dataset distributions. Even simple methods to encourage domain similarity, such as prior class distribution matching in Domain Adaptive Transfer Learning (DATL) (Ngiam et al., 2018), are shown to be effective – indeed, in some cases, the distribution similarity is shown to be more important than the scale of the source dataset. In this paper, our goal is to push this direction further by introducing a novel RL-based meta-learning framework. Our framework, learning to transfer learn (L2TL), adaptively infers the beneficial source samples directly from the performance on the target task. There are numerous cases that source samples could have features that are implicitly relevant to the target samples and would benefit the learning process, but they may belong to different classes. For example, consider the classification problem for bird images. The source dataset may not contain bird images, but may have airplane images with similar visual patterns that would aid the training of the bird classifier. L2TL framework is designed to automatically handle such cases with its policy learning, and can push the performance further in ways that manual source dataset selection or fixed domain similarity methods may not be able to. We demonstrate state-of-the-art transfer learning results given fixed models in wide range of scenarios:

- *Source and target datasets from similar domains*: We consistently outperform the fine-tuning baseline with a 0.6%-1.3% relative accuracy gain on five fine-grained datasets, and consistently outperform DATL with a 0.3%-1.5% relative accuracy gain.

- *Low-shot target dataset regime*: We consistently outperform the fine-tuning baseline on five fine-grained datasets, up to 6.5% relative accuracy gain for five samples per class.
- *Source and target datasets from dissimilar domains*: We outperform the fine-tuning baseline, up to 1.7% relative accuracy gain on a texture dataset and 0.7 relative AUC gain on Chest X-Ray dataset.

In addition, L2TL yields ranking of the source data samples according to their contributions to the target task, that open horizons for new forms of interpretable insights for model developers.

2 Related Work

Adaptive transfer learning: There is a long history of transfer learning for neural networks, particularly in the form of fine-tuning (Yosinski et al., 2014) (Girshick et al., 2014). Various directions were recently considered to improve standard fine-tuning. One direction is carefully choosing which portion of the network to adapt while optimizing the information extraction from the source dataset. In (Guo et al., 2019), a policy network is used to make routing decisions on whether to pass the input through the fine-tuned or the pre-trained layers. In (Li et al., 2018), a regularization scheme is proposed to promote the similarity of the fine-tuned model with the pre-trained model as a favorable inductive bias. Another direction is carefully choosing which input samples are relevant to the target task, as in our paper. (Ge and Yu, 2017) uses filter bank responses to select nearest neighbor source samples and demonstrates improved performance. In (Cui et al., 2018), domain similarity between source and target datasets is quantified using Earth Mover's Distance (EMD). Transfer learning is demonstrated to benefit from pre-training on a source domain that is similar in EMD. With a simple greedy subset creation selection criteria, promising results are shown for improving the target test set performance. Domain adaptive transfer learning (DATL) (Ngiam et al., 2018) employs probabilistic shaping, where the value is proportional to the ratios of estimated label prior probabilities. L2TL does not use an indirect similarity metric like proximity of filter bank responses, EMD or prior class probabilities. Instead, it aims to assign weights to optimize the target set metric directly.

Reweighing training examples: Reweighing of constituent training terms has been considered for various performance goals. (Ren et al., 2019) applies gradient descent-based meta-learning with the goal of providing noise robustness and class balance in learning. Focal loss (Lin et al., 2017) is another soft weighting scheme that emphasizes harder examples. In (Jiang et al., 2018), a student-teacher training framework is utilized such that the teacher model provides a curriculum via a sample weighting scheme for the student model to focus on samples whose labels are likely to be correct. (Ghorbani and Zou, 2019) studies the value of examples by estimating the data Shapley value, and it shows that removing examples with low values would not harm performance. Reweighting of examples is also used in self-paced learning (Kumar et al., 2010) where the weights are optimized to learn easier examples first. Different from these, our paper focuses on optimizing the weights for each class, with the purpose of improving the transfer learning performance.

Meta-learning: Meta-learning broadly refers to learning to learn frameworks (Schmidhuber et al., 1997) whose goal is to improve the adaptation to a new task with the information extracted from other tasks. Meta learners are typically based on inspiration from known learning algorithms like gradient descent (Finn et al., 2017) or derived from black box neural networks (Santoro et al., 2016). In few-shot learning, the use of validation loss as a meta-objective has been explored (Ravi and Larochelle, 2017). However, for optimization problems with non-differentiable objectives like neural architecture search, RL-based meta-learning is shown to be a promising approach (Zoph and Le, 2017; Pham et al., 2018). RL-based optimization has successfully been applied in many search spaces, e.g. learning a data augmentation policy (Cubuk et al., 2018). The specific form of meta-learning application in L2TL is novel – it employs guidance on the source dataset information extraction with the reward from the target validation dataset performance. Different from few-shot learning, we consider a more common real-world scenario where an easily accessible source dataset is integrated.

3 Learning From Source and Target Datasets

We consider a training objective function $\mathcal{L}(\mathbf{\Omega}, \zeta_{\mathbf{S}}, \zeta_{\mathbf{T}}, \lambda, \alpha_s, \alpha_t)$[1] that is jointly optimized for a source dataset D_S and a target dataset D_T in the general form:

$$\mathcal{L} = \alpha_s[i] \cdot \sum_{j=1}^{B_S} \lambda(x_j, y_j; \mathbf{\Phi}) \cdot L_S(f_S(x_j; \mathbf{\Omega}, \zeta_{\mathbf{S}}), y_j) + \alpha_t[i] \cdot \sum_{k=1}^{B_T} L_T(f_T(x'_k; \mathbf{\Omega}, \zeta_{\mathbf{T}}), y'_k), \quad (1)$$

[1]Function arguments are not often shown in the paper for notational convenience.

where (x, y) are the input and output pairs $(x_j, y_j \sim D_S, x'_k, y'_k \sim D_T)$, B_S and B_T are the source and target batch sizes[2], $\alpha_s[i]$ and $\alpha_t[i]$ are the scaling coefficients at i^{th} iteration, λ is the importance weighing function, $f_S(\cdot; \mathbf{\Omega}, \zeta_\mathbf{S})$ and $f_T(\cdot; \mathbf{\Omega}, \zeta_\mathbf{T})$ are encoding functions for the source and the target datasets with trainable parameters $\mathbf{\Omega}$, $\zeta_\mathbf{S}$ and $\zeta_\mathbf{T}$[3]. To maximally benefit from the source dataset, a vast majority of the trainable parameters should be shared. If we consider the decompositions, $f_S(\cdot; \mathbf{\Omega}, \zeta_\mathbf{S}) = h_S(\cdot; \zeta_\mathbf{S}) \circ g(\cdot; \mathbf{\Omega})$ and $f_T(\cdot; \mathbf{\Omega}, \zeta_\mathbf{T}) = h_T(\cdot; \zeta_\mathbf{T}) \circ g(\cdot; \mathbf{\Omega})$, g can be a high capacity function with large number of trainable parameters that can be represented with a deep neural network, and h_T and h_S are low capacity functions with small number of parameters that can be represented with very shallow neural networks.[4] The learning goal of Eq. 1 is to generalize to unseen samples from a held-out target validation dataset, and maximize a target performance metric R on it: $\sum_{x',y' \sim D'_T} R(f_T(x'; \hat{\mathbf{\Omega}}, \hat{\zeta_\mathbf{T}}), y'))$. R may or may not be differentiable with respect to x and y such as the top-1 accuracy or area under the curve (AUC) for classification. $\hat{\mathbf{\Omega}}, \hat{\zeta_\mathbf{T}}$ are the pre-trained weights optimized in Eq. 1. Without transfer learning, i.e., when only the target dataset is considered, $\alpha_s[i] = 0$ and $\alpha_t[i] = 1$ for all i. In fine-tuning, the optimization is first considered for the source dataset for N_S steps with uniform weighing of the samples $\lambda(x, y) = 1$, and then for the target dataset using the pre-trained weights $\hat{\mathbf{\Omega}}, \hat{\zeta_\mathbf{T}}$, i.e.:

$$(\alpha_s[i], \alpha_t[i]) = \begin{cases} (1, 0), i < N_S \\ (0, 1), i > N_S \end{cases} \tag{2}$$

Next, we describe our method towards optimal learning from source and target datasets.

4 Learning to Transfer Learn

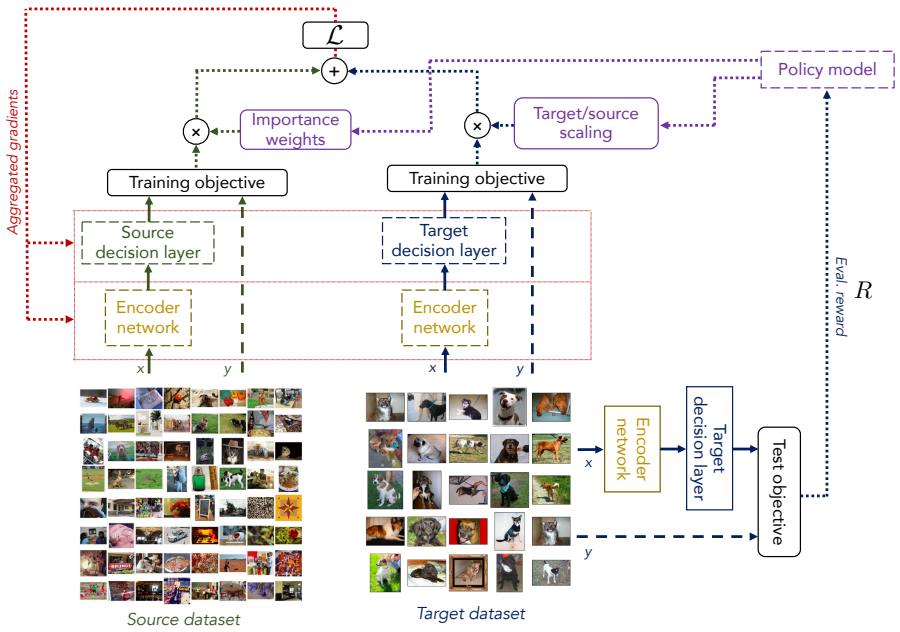

Figure 1: Overall diagram of L2TL. Dashed boxes correspond to trainable functions.

We propose learning to transfer learn (L2TL) framework (shown in Fig. 1) to learn the weight assignment adaptively, rather than using a fixed weight assignment function $\lambda(x, y; \mathbf{\Phi})$ to measure the relatedness between the source domain and the target domain. Learning of the adaptive weights in L2TL is guided by the performance metric R on a held-out target validation dataset. Thus, beyond targeting general relatedness, the framework directly targets relatedness for the specific goal of improvement in target evaluation performance.

[2]Batch approximations may be optimal for different batch sizes for source and target dataset and thus may employ different batch normalization parametrization.

[3]In $f(\cdot; \mathbf{W})$ representation, $\mathbf{W}$ denote the trainable parameters.

[4]Source datasets are typically much larger and contain more classes, hence h_S may have higher number of parameters than h_T.

Algorithm 1: L2TL – Learning to Transfer Learn

$N \leftarrow$ number of training iterations
for $i \leftarrow 1$ *to* N **do**
  $l_s \leftarrow 0, l_t \leftarrow 0$
  for $j \leftarrow 1$ *to* B_S **do**
    Sample x_j, y_j from D_S
    Calculate classification loss $L_S(x_j, y_j; \mathbf{\Omega}, \zeta_\mathbf{S})$
    Calculate example weight $\lambda(x_j, y_j; \mathbf{\Phi})$
    $l_s = l_s + \lambda \cdot L_S$
  $l_s = \alpha_i \cdot l_s$
  for $k \leftarrow 1$ *to* B_T **do**
    Sample x'_k, y'_k from D_T
    Calculate classification loss $L_T(x'_k, y'_k; \mathbf{\Omega}, \zeta_\mathbf{T})$
    $l_t = l_t + L_T$
  Update $\mathbf{\Omega}, \zeta_\mathbf{S}, \zeta_\mathbf{T}$ using stochastic gradient descent with loss $l_s + l_t$
  $r \leftarrow 0$
  for $k \leftarrow 1$ *to* B_P **do**
    Sample x'_k, y'_k from $D_{T'}$
    Calculate reward $R(f_T(x'_k), y'_k)$
    $r = r + R$
  Update $\mathbf{\Phi}$ with reward r using policy gradient

While optimizing for $\lambda(x, y; \mathbf{\Phi})$, one straightforward option for scaling coefficients would be alternating them between $(1, 0)$ and $(0, 1)$ – i.e. training the source dataset until convergence with optimized $\hat{\mathbf{\Phi}}$ and then training the target dataset until convergence with the pre-trained weights from the source dataset. Yet, the approach may potentially require many alternating update steps and the computational cost may become prohibitively high. Instead, we design the policy model in L2TL to output $(\alpha_s[i], \alpha_t[i])$ along with λ.[5] The policy optimization step is decoupled from the gradient-descent based optimization for $\mathbf{\Omega}$, $\zeta_\mathbf{S}$ and $\zeta_\mathbf{T}$. Updates are reflected to the policy model via the information embodied in $\mathbf{\Omega}$ and $\zeta_\mathbf{T}$. Algorithm. 1 overviews the training updates steps.

In the first phase of a learning iteration, we apply gradient decent-based optimization to learn the encoder weights $\mathbf{\Omega}$, and the classifier layer weights $\zeta_\mathbf{S}, \zeta_\mathbf{T}$ to minimize the loss function $\mathcal{L}$:

$$\hat{\mathbf{\Omega}}, \hat{\zeta_\mathbf{S}}, \hat{\zeta_\mathbf{T}} = \text{argmin}_{\mathbf{\Omega}, \zeta_\mathbf{S}, \zeta_\mathbf{T}} \mathcal{L}(\hat{\mathbf{\Phi}}; \mathbf{\Omega}, \zeta_\mathbf{S}, \zeta_\mathbf{T}). \tag{3}$$

In this learning phase, the policy model is fixed, and its actions are sampled to determine weights. The loss might be skewed when most of source dataset samples in a batch are unrelated, while some batches contain more related examples. To ease this problem, we sample a larger batch and dynamically select more related examples. At each iteration, we sample a training batch of size $M_S \cdot B_S$, and use the top B_S of them with the highest weights for training updates. This approach also yields computational benefits as the gradients would not be computed for most source dataset samples until convergence.

In the second phase of a learning iteration, the goal is to optimize policy weight $\mathbf{\Phi}$ that maximizes the evaluation metric $R_{D'_T}$ on the target validation set with given encoder weights from the first phase:

$$\max_{\mathbf{\Phi}} R_{D'_T}(\hat{\mathbf{\Omega}}, \hat{\zeta_\mathbf{S}}, \hat{\zeta_\mathbf{T}}; \Phi). \tag{4}$$

We treat this step as an RL problem, such that the policy model outputs the actions for $\lambda(x, y; \mathbf{\Phi})$ and α towards optimization of a reward. In its general form $\lambda(x, y; \mathbf{\Phi})$ may yield a very high dimensionality for optimization of $\mathbf{\Phi}$. For simplicity, we consider sample-independent modeling of $\lambda(x, y; \mathbf{\Phi})$, similar to (Ngiam et al., 2018), i.e., $\lambda(x, y; \mathbf{\Phi}) = \lambda(y; \mathbf{\Phi})$. For more efficient optimization, we discretize the possible values of $\lambda(y; \mathbf{\Phi})$ into pre-defined number of actions, in the range $\lambda(y) \in [0, 1]$. We define n actions, such that each action $k \in [0, n-1]$ corresponds to a weight value $k/(n-1)$.

[5] Without loss of generality, we can optimize a single weight $\alpha_s[i]$ (setting $\alpha_t[i] = 1$) as the optimization is scale invariant.

For example, when $n = 11$, the weight values are $[0, 0.1, 0.2, \ldots, 1.0]$. We also discretize the possible values for α, using n' actions. Each action k' corresponds to $\beta k'/(n'-1)$, where β is a hyperparameter to constrain the value range of α. The search space has $n' \times (c_S)^n$ possibilities, where c_S is the number of classes in the source dataset. When training the policy model, we use policy gradient to maximize the reward on the target dataset $D_{T'}$, using a batch size of B_P. We use the evaluation mode for the convolutional encoder. At iteration t, we denote the advantage $A_t = R_t - b_t$, where b_t is the baseline. Following (Pham et al., 2018), we use the moving average baseline to reduce variance, i.e., $b_t = (1-\gamma)b_t + \gamma R_t$, where γ is the decay rate. The policy gradient is computed using REINFORCE (Williams, 1992) and optimized using Adam (Kingma and Ba, 2014).

5 EXPERIMENTS

We demonstrate the performance of L2TL in various scenarios. As the source dataset, we use the ImageNet dataset (Russakovsky et al., 2015) containing 1.28M images from 1K classes, and also a much larger internal dataset, named as ANON dataset, containing ∼300M images from 18,291 classes to demonstrate scalability. Target datasets are chosen based on the scenarios. Hyperparameters of the encoder models are chosen from the published baselines and policy model parameters are cross-validated on a validation set. All hyperparameters are presented in Appendix B. For datasets that the testing accuracy is reported using the model trained on training and validation examples, L2TL is first trained on the training split using the reward from the validation set. Then, the learned control variables are then used to train the joint model on the training and validation examples. We do not use the test set during model training. For the fine-tuning experiments, we also use best set of hyperparameters evaluated on the validation set. The results are averaged over three runs.

5.1 SOURCE AND TARGET DATASETS FROM SIMILAR DOMAINS

We initially consider the scenario of target datasets with classes that mostly exist in the source dataset. In this scenario, DATL (Ngiam et al., 2018) was proven to be effective. We evaluate L2TL on five datasets focusing on different subsets. The detailed dataset splits are presented in Appendix A. The reward is measured on the validation set using top-1 accuracy.

Table 1: Transfer learning performance with ImageNet source dataset, compared to MixDCNN (Wang et al., 2015), EMD (Cui et al., 2018), OPAM (Peng et al., 2017) and DATL (Ngiam et al., 2018) (* denotes the results fine-tuning baseline from (Ngiam et al., 2018)).

Method	Target dataset test accuracy (%)				
	Birds	Pets	Cars	Aircraft	Food
MixDCNN	74.8	-	-	82.5	-
EMD	-	-	91.3	85.5	88.7
OPAM	-	93.8	92.2	-	-
Fine-tuning*	77.2	93.3	91.5	88.8	88.7
DATL*	76.6	94.1	92.1	87.8	88.9
Fine-tuning	77.1	93.1	92.0	88.2	88.4
L2TL	**78.1**	**94.4**	**92.6**	**89.1**	**89.2**

Table 2: Results on the test set for Birds using images from ANON.

Method	Accuracy (%)
Fine-tuning	74.9
DATL	81.7
L2TL	**82.4**

The results of L2TL along with fine-tuning and DATL benchmarks are shown in Table 1. As can be seen, L2TL outperforms fine-tuning across all the datasets with 0.6%-1.3% relative accuracy difference, which demonstrates the strength of L2TL in selecting related source examples across various domains. DATL performs worse than fine-tuning on Birdsnap and Aircraft, unlike L2TL. This underlines the importance of leveraging the visual similarity in the ways beyond the label match as in DATL. The results using ANON, shown in Table 2, shows that L2TL preserves its benefits in learning relatedness for much larger scale datasets.

Table 3: Top chosen classes from ImageNet source dataset that are related to the target datasets. For Birds, the top source class is "bea eater" which is one of the bird species in ImageNet. The second top "aepyceros melampus" is an antelope that has narrow mouth, which is similar to some birds with sharp spout. The "valley" also matches the background in some images. For Cars, we interestingly observe the high-weight class "barrel, cask", which indeed include wheels and car-looking body types in many images. "Terrapin" is a reptile that crawls on the ground with four legs, whose shape looks like vehicles in some way. For Food, the high-weight classes are the least relevant intuitively but still contain classes with visually-relevant patterns – e.g., "caldron, cauldron" may contain images with food inside, but most "seashore, coast" are less related to food.

Target dataset	Source classes (weights)
Birds	bee eater (1.0); aepyceros melampus (0.95); sea cradle (0.92); barracouta (0.91) valley (0.90); sombrero (0.86); rosehip (0.84); Scottish deerhound (0.82);
Pets	coral reef (0.88); prayer rug (0.88); koala (0.84); fire salamander (0.81) Irish setter (0.79); Arabian camel (0.78); Irish terrier (0.74); leaf beetle (0.72);
Cars	desktop computer (0.80); butternut squash (0.76); barrel, cask (0.65); weevil (0.60); pool table (0.56); clumber (0.54); passenger car (0.50);
Aircraft	bagel, beigel (0.97); ballplayer, baseball player (0.84); freight car (0.80); teapot (0.83); crate (0.78); velvet (0.74); electric locomotive (0.68);
Food	caldron, cauldron (0.84); menu (0.81); seashore, coast, seacoast (0.74) acorn squash (0.73); dining table, board (0.67); globe artichoke (0.67);

Analysis of high-weight source samples: To build insights on the learned weights, we sample $10k$ actions from the policy and rank the source labels according to their weights. Table 3 shows the classes with the highest weights and Fig. 2 visualize a few representative examples from them. These demonstrate that L2TL can judiciously extract the related classes from the source based on the pattern/shape of the objects, or background scenes.

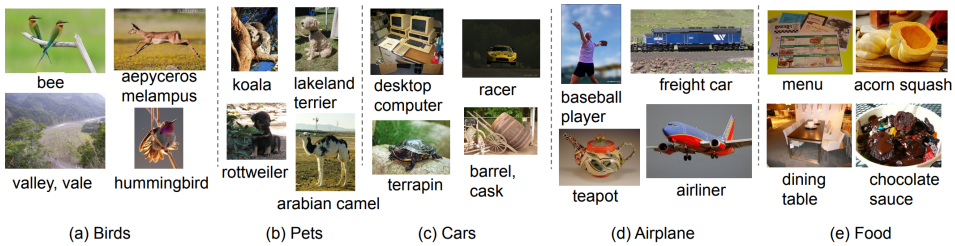

Figure 2: Representative examples from the source datasets with high weight from L2TL for different target datasets. In most cases, we observe them to be highly-related to the target dataset.

Effectiveness of RL: We study the effectiveness of RL training comparing it to two baselines: (i) random search: where the policy model is not optimized and random actions are chosen as the policy output, and (ii) uniform weights: a constant importance weight is assigned to all training samples. Note that for these baselines, α is still optimized via policy gradient. The hyperparameters are cross-validated on the validation set. We show the best results of the baselines. As shown in Fig. 3, L2TL outperforms both after sufficient number of iterations, demonstrating the importance of reweighting via policy gradient.

Low-shot regime: In the extreme regime of very small number of training examples, generalizing to unseen examples is most challenging as the model can be prone to overfitting. Fig. 4 shows performance in the small data regime for the same datasets. In most cases, we observe significant increase in performance when the number of examples per class is smaller. For five examples per class, the gap is as high as **6.5%** (for Stanford Car). We observe that the gap between the L2TL and the fine-tune baseline often becomes smaller when more examples are used, but still remains as high as 1.5% when 60 examples per class are used (for Birdsnap). The results show that L2TL can yield significant improvements in real-world tasks where the number of training examples are limited.

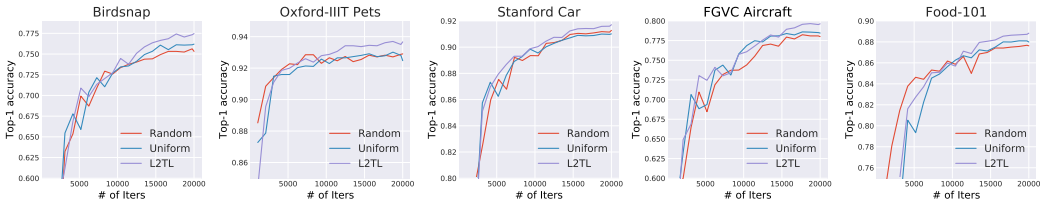

Figure 3: Performance comparison between L2TL, random search and uniform weights. The curves are more oscillatory at the beginning, but become stable later during the training.

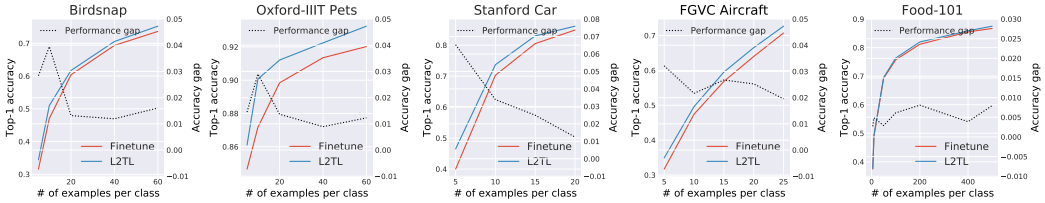

Figure 4: Number of examples per class vs. top-1 accuracy for L2TL and fine-tuning

5.2 Source and target datasets from dissimilar domains

In this section, we consider two target datasets with classes that do not exist in source dataset:

- *Describable Textures Dataset (DTD) dataset (Cimpoi et al., 2014)*: It contains textural images in the wild from 47 classes such as striped and matted. The dataset has 20 splits and we evaluate the testing results on the first split. Each training, validation, and testing split has 1,880 images.
- *Chest X-Ray Dataset CheXpert (Irvin et al., 2019)*: This medical dataset has been recently introduced for chest radiograph interpretation. It consists of 224,316 chest radiographs of 65,240 patients labeled for 14 observations as positive, negative, or uncertain. Following (Irvin et al., 2019), we report AUC on five classes and we regard "uncertain" examples as positive.[6] For L2TL, we use the mean AUC as the reward.

Table 4 shows the results for DTD. We observe that ImageNet fine-tuning greatly improves the classification results. L2TL further improves the fine-tuning baseline by 1.5%, demonstrating the significance of L2TL enabling the use of related source classes instead of using all classes. For the low-shot example case, we observe a more than 5% gain when only 10 examples are provided per class, validating the effectiveness of L2TL in more accurate transfer learning. Fig. 5 shows that L2TL is able to utilize visual similarities between the source and the target classes.

Table 4: Results on the test set for DTD on split 1.

Method	Acc
Random Init	57.4
Fine-tuning	70.3
L2TL	**72.0**
10-shot, Fine-tuning	55.0
10-shot, L2TL	**60.1**

Table 5 shows the results for CheXpert. L2TL performs better than the fine-tuning baseline by an AUC margin of .007. There are not many straightforward visual similarities to humans between ImageNet and CheXpert, but L2TL is still capable of discovering them to improve performance.

[6]Our reproduced results are matched with (Irvin et al., 2019) on mean AUC. However, there are variances as we can see that for some classes, we achieve slightly worse than (Irvin et al., 2019). This may because of the small number of validation examples (200) used.

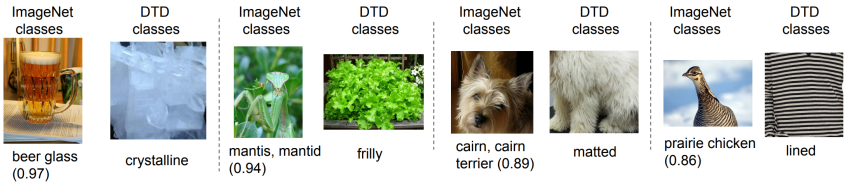

Figure 5: Top source classes with the highest weight from ImageNet while transferring to DTD target. Representative images from each ImageNet class are shown along with related examples from DTD. The similarities occur in the form of texture pattern for most DTD classes. For example, "praire chicken" images from ImageNet typically contain patterns very relevant to "lined" from DTD.

Table 5: AUC comparisons (based on the same evaluation protocol as in (Irvin et al., 2019)) on the CheXpert dataset. * denotes the results reported in (Irvin et al., 2019).

Method	Atelectasis	Cardiomegaly	Consolidation	Edema	Pleural Effusion	Mean
Fine-tuning*	.858	.832	.899	.941	**.934**	.893
Fine-tuning	.852	.838	.900	.945	.928	.893
L2TL	**.861**	**.844**	**.915**	**.948**	.932	**.900**

6 Computational cost of training

Table 6 presents the computational cost for fine-tuning, DATL and L2TL. In DATL, given a new target dataset, a new model has to be trained on the resampled data until convergence. This step is time-consuming for large-scale source datasets. In L2TL, the transfer learning step is more expensive than fine-tuning, as it requires the computation on both source and target datasets. Yet, it only requires a single training pass on the source dataset, and yields much lower training time compared to DATL.

Table 6: Computational cost of training using Inception-V3 on Cloud TPU v2 when the source dataset is ImageNet. The last column of total time assumes availability of a pre-trained source model.

Method	Number of iterations		Time per iterations		Total time	
	Pre-training	Transfer learning	Pre-training	Transfer learning	From scratch	With pre-trained source model
Fine-tuning	213,000	20,000	0.14s	0.21s	9.5h	1.2h
DATL	713,000	20,000	0.14s	0.21s	28.9h	20.6h
L2TL	213,000	20,000	0.14s	0.75s	12.5h	4.2h

7 Conclusions and Future Work

We propose a novel RL-based framework, L2TL, to improve transfer learning on a target dataset by careful extraction of information from a source dataset. L2TL considers joint optimization of models for source and target tasks, while using adaptive weights for scaling of constituent loss terms. We use the performance metric on the target validation set as the reward to train the policy model, which outputs the weights for each source class adaptively. We demonstrate state-of-the-art performance of L2TL for various datasets. The performance benefit of L2TL typically gets more significant as the target dataset size gets smaller. Our framework does not utilize an explicit similarity metric, but learns source class weights to directly optimize the target dataset performance. In cases where source and target datasets come from substantially different domains, L2TL still yields clear improvements. This improvement often comes from utilizing the source dataset classes that have relevant visual patterns despite belonging to a substantially different class.

Our general L2TL framework can be pushed beyond the approximations in this paper. A search space with a higher optimization granularity is expected to improve the results. Modeling the x dependence of the $\lambda(x, y; \mathbf{\Phi})$ function using a policy gradient can be a promising step towards that approach (particularly for the datasets with high intra-class variance, such a model can help to select particular samples within the class) albeit likely accompanied by increased computational complexity.

REFERENCES

D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *ICML*, 2016.

S. Ö. Arik, J. Chen, K. Peng, W. Ping, and Y. Zhou. Neural voice cloning with a few samples. In *NIPS*, 2018.

T. Berg, J. Liu, S. Woo Lee, M. L. Alexander, D. W. Jacobs, and P. N. Belhumeur. Birdsnap: Large-scale fine-grained visual categorization of birds. In *CVPR*, 2014.

L. Bossard, M. Guillaumin, and L. Van Gool. Food-101–mining discriminative components with random forests. In *ECCV*, 2014.

C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, et al. State-of-the-art speech recognition with sequence-to-sequence models. In *ICASSP*, 2018.

M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *CVPR*, 2014.

A. Conneau, H. Schwenk, L. Barrault, and Y. LeCun. Very deep convolutional networks for natural language processing. In *ACL*, 2017.

E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

Y. Cui, Y. Song, C. Sun, A. Howard, and S. Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *CVPR*, 2018.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *arxiv:1810.04805*, 2018.

C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

W. Ge and Y. Yu. Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning. In *CVPR*, 2017.

A. Ghorbani and J. Zou. Data shapley: Equitable valuation of data for machine learning. In *ICML*, 2019.

R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

Y. Guo, H. Shi, A. Kumar, K. Grauman, T. Rosing, and R. S. Feris. Spottune: Transfer learning through adaptive fine-tuning. In *CVPR*, 2019.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

J. Hestness, S. Narang, N. Ardalani, G. F. Diamos, H. Jun, H. Kianinejad, M. M. A. Patwary, Y. Yang, and Y. Zhou. Deep learning scaling is predictable, empirically. *arXiv:1712.00409*, 2017.

G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

J. Irvin, P. Rajpurkar, M. Ko, Y. Yu, S. Ciurea-Ilcus, C. Chute, H. Marklund, B. Haghgoo, R. Ball, K. Shpanskaya, et al. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *AAAI*, 2019.

L. Jiang, Z. Zhou, T. Leung, L. Li, and L. Fei-Fei. Mentornet: Regularizing very deep neural networks on corrupted labels. In *ICML*, 2018.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014.

J. Krause, J. Deng, M. Stark, and L. Fei-Fei. Collecting a large-scale dataset of fine-grained cars. *The Second Workshop on Fine-Grained Visual Categorization, 2013*.

M. P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In *NIPS*, 2010.

S. Lai, L. Xu, K. Liu, and J. Zhao. Recurrent convolutional neural networks for text classification. In *AAAI*, 2015.

J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *arXiv:1901.08746*, 2019.

X. Li, Y. Grandvalet, and F. Davoine. Explicit inductive bias for transfer learning with convolutional networks. In *ICML*, 2018.

T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

D. Mahajan, R. B. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018.

S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *arXiv:1306.5151*, 2013.

S. Moon, S. Kim, and H. Wang. Multimodal transfer deep learning for audio visual recognition. *arXiv:1412.3121*, 2014.

J. Ngiam, D. Peng, V. Vasudevan, S. Kornblith, Q. V. Le, and R. Pang. Domain adaptive transfer learning with specialist models. *arXiv preprint arXiv:1811.07056*, 2018.

O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. Jawahar. Cats and dogs. In *CVPR*, 2012.

Y. Peng, X. He, and J. Zhao. Object-part attention model for fine-grained image classification. *TIP*, 2017.

H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018.

S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *CVPR*, 2014.

M. Ren, W. Zeng, B. Yang, and R. Urtasun. Learning to reweight examples for robust deep learning. In *ICML*, 2019.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016.

J. Schmidhuber, J. Zhao, and M. Wiering. Shifting inductive bias with success-story algorithm, adaptive levin search, and incremental self-improvement. *Machine Learning*, 1997.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, et al. Wavenet: A generative model for raw audio. *arXiv:1609.03499*, 2016.

D. Wang, Z. Shen, J. Shao, W. Zhang, X. Xue, and Z. Zhang. Multiple granularity descriptors for fine-grained categorization. In *ICCV*, 2015.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992.

J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.

S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016.

B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

B. Zoph, D. Yuret, J. May, and K. Knight. Transfer learning for low-resource neural machine translation. In *ACL*, 2016.

A DATASET SPLITS OF THE FINE-GRAINED DATASETS

Table 7: Details for five fine-grained datasets: Birdsnap (Birds) (Berg et al., 2014), Oxford-IIIT Pets (Pets) (Parkhi et al., 2012), Stanford Cars (Cars) (Krause et al.), FGVC Aircraft (Aircraft) (Maji et al., 2013), and Food-101 (Food) (Bossard et al., 2014).

	Birds	Pets	Cars	Aircraft	Food
# of classes	500	37	196	100	101
# of train examples	42,405	2,940	6,494	3,334	68,175
# of valid examples	4,981	740	1,650	3,333	7,575
# of test examples	2,443	3,669	8,041	3,333	25,250

B HYPERPARAMETERS

When the source dataset is ImageNet, we use batch size $B_S = 256$, $B_T = 256$, $B_P = 1024$ and a batch multiplier $M_S = 5$ for all the experiments. For the ANON dataset, to reduce the number of training iterations, we use $B_S = 1,024$. The number of actions n' for α is 100.

We use the Inception-V3 architecture for all the experiments except CheXpert. For target dataset, we searched the initial learning rate from $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.15, 0.2, 0.4\}$, weight decay from $\{0, 4 \times 10^{-5}\}$. All the datasets are optimized by SGD with momentum 0.9. We use the single central crop during evaluation. The learning rate is cosine decayed after first 2,000 iterations warmup. The number of training iterations is 20,000. When optimizing our policy model, we the Adam optimizer with a fixed learning rate 0.0001. As policy model parameters, we set $\beta = 0.5$ and $\gamma = 0.05$ for all the experiments. We followed the standard image preprocessing procedure for Inception-V3 on both the source images and the target images.

For CheXpert, we use the DenseNet-121 architecture Huang et al. (2017) and following the evaluation protocol specified in Irvin et al. (2019), where ten crops are used for evaluation and 30 checkpoints are ensemble to obtain the final results. We cross validate weight decay and initial learning rate, where the weight decay is searched in [0, 0.0001] and the learning rate searched in range $[0.5, 0.8, 1.0, 1.3, 1.5, 2.0]$. All other hyperparameters are same as above. We use the same input preprocessing as described in `https://github.com/zoogzog/chexnet`.