EXPLICIT INDUCTION BIAS FOR TRANSFER LEARNING WITH CONVOLUTIONAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

In inductive transfer learning, fine-tuning pre-trained convolutional networks substantially outperforms training from scratch. When using fine-tuning, the underlying assumption is that the pre-trained model extracts generic features, which are at least partially relevant for solving the target task, but would be difficult to extract from the limited amount of data available on the target task. However, besides the initialization with the pre-trained model and the early stopping, there is no mechanism in fine-tuning for retaining the features learned on the source domain. In this paper, we investigate several regularization schemes that explicitly promote the similarity of the final solution with the initial model. We eventually recommend a simple L^2 penalty using the pre-trained model as a reference, and we show that this approach behaves much better than the standard scheme using weight decay on a partially frozen network.

1 INTRODUCTION

It is now well known that modern convolutional neural networks, e.g. Krizhevsky et al. (2012), Simonyan & Zisserman (2015), He et al. (2016), Szegedy et al. (2016), can achieve remarkable performance on large-scale image databases, e.g. ImageNet (Deng et al. 2009) and Places365 (Zhou et al. 2017), but it is really dissatisfying to see the vast amounts of data, computing time and power consumption that are necessary to train deep networks. Fortunately, such convolutional networks, firstly trained on a large database, can further be refined to solve related but different visual tasks by means of transfer learning, like fine-tuning (Yosinski et al. 2014, Simonyan & Zisserman 2015).

Robust information is believed to be learned from the source domain of a large-scale database and transferred to the target domain by initializing the network with the pre-trained parameters. However, the parameters, after fine-tuning, will inevitably be quite different from initial values in all layers of the network due to the difference between the source domain and the target domain. Even worse, during fine-tuning, L^2 regularization forces the parameters towards the origin and expands the distance between the parameters and their initial values. Considering that the size of the target database is moderate or even small compared to the source database plus the effect of L^2 regularization, the changes in the parameters during fine-tuning will negatively affect the information learned from the source domain.

In order to preserve the learned information, we consider using other parameter regularization methods during fine-tuning. L^2 regularization is often preferred to be used because of its efficiency by simply restricting the distance between the parameters and the origin. Nonetheless, for target tasks where initial values of parameters are no longer a random set around the origin, we propose restricting the distance to the initial values instead of the distance to the origin. Thus the regularization can still produce the reducing-overfitting effect, and simultaneously encourage exploiting the learned information by keeping the parameters around the starting point of fine-tuning.

In this paper, we aim at learning from less labeled training data, using networks that were already trained on similar problems. A form of transfer learning is thus considered, where some pieces of information, acquired when solving a previous learning problem, have to be conveyed to another learning problem. Under this setting, we explore several parameter regularization methods that can explicitly retain the learned information. We first investigate a variant of L^2 with the pre-trained model as reference, which we name L^2-*SP* because the pre-trained parameters represent the starting point (-*SP*) of the fine-tuning process. In addition, we evaluate other regularization forms, e.g.

L^1-*SP* and Group-Lasso-*SP* approaches, as they can freeze some individual parameters or groups of parameters at the pre-trained parameters. Fisher information is also taken into account when we test L^2-*SP* and Group-Lasso-*SP* approaches. Our experiments indicate that all tested parameter regularization methods that retain the learned information get an edge over the standard L^2 weight decay approach. We also analyze the effect of L^2-*SP* through theoretical demonstration and experimental evidence and discuss its advantages, as the grounds to recommend using L^2-*SP* for transfer learning tasks.

2 Related Work

We follow here the nomenclature of Pan & Yang (2010), who categorized several types of transfer learning. A domain corresponds to the feature space and its distribution, whereas a task corresponds to the label space and its conditional distribution with respect to features. The initial learning problem is defined on the source domain and source task, whereas the new learning problem is defined on the target domain and the target task.

In the typology of Pan & Yang, we consider the inductive transfer learning setting, where the target task is different from the source task. We furthermore focus on the case where a vast amount of data were available in the source problem, and some limited amount of labeled data are available for solving the target problem. Under this setting, we aim at improving the performance on the target problem through parameter regularization methods that explicitly encourage the similarity of the solutions to the target and source problems.

In this Section, we present improvements in inductive transfer learning on convolutional networks under the same setting as ours. Then we introduce recent parameter regularization methods used in different settings of transfer learning.

Inductive Transfer Learning Donahue et al. (2014) repurposed features extracted from different layers of the pre-trained AlexNet of Krizhevsky et al. (2012) and plugged them into an SVM or a logistic regression classifier. This approach outperformed the state of the art at that time on the Caltech-101 database (Fei-Fei et al. 2006). Yosinski et al. (2014) showed that fine-tuning the whole AlexNet performed better than using the network as a static feature extractor. Fine-tuning pre-trained VGG (Simonyan & Zisserman 2015) on the image classification task of VOC-2012 (Everingham et al. 2010) and Caltech 256 (Griffin et al. 2007) achieved the first place at that moment. Ge & Yu (2017) proposed a scheme for selecting a subset of images from source data that have similar local features to those in target data and then jointly fine-tuned the pre-trained convolutional networks to improve performance on the target task. Besides image classification, a lot of methods for object detection (Girshick et al. 2014, Redmon et al. 2016, Ren et al. 2015) and image segmentation (Long et al. 2015a, Chen et al. 2017, Zhao et al. 2017) have been proposed based on fine-tuning and obtained much better results than training from scratch, in spite of different techniques that were added in their networks.

The success of transfer learning with convolutional networks relies on the generality of the learned information that has been digested from a large database like ImageNet. Yosinski et al. (2014) also quantified the transferability of these pieces of information in different layers, e.g. first layers learn general features, middle layers learn high-level semantic features and the last layer learns very specific features to a particular task. That can be also noticed by the visualization of features (Zeiler & Fergus 2014). Overall, the learned information can be conveyed to related but different domains and the parameters in the network are reusable for different tasks.

Parameter Regularization Parameter regularization is one of various regularization methods in deep learning. L^2 regularization has been used for a long time and is a very simple method for preventing overfitting by limiting the norm of the parameter vector. L^1 prevents overfitting by zeroing out some weights. Max-norm regularization (Srebro & Shraibman 2005) was found especially helpful when using dropout (Srivastava et al. 2014). If the norm of the parameter vector is larger than a threshold, max-norm regularization will project the parameter vector onto the surface of the ball with the threshold as radius. Yuan & Lin (2006) introduced the Group-Lasso penalty in order to allow pre-defined groups of weights to be regularized with the same force. Xie et al. (2017) proposed the orthonormal regularizer that forces each kernel in one convolution layer to have mini-

mum correlation with others in that layer so enhances kernel's expressive ability without increasing parameters.

In lifelong learning (Thrun & Mitchell 1995, Pentina & Lampert 2015) that solves sequentially a quantity of tasks, the information gained from the previous tasks is easily lost as new tasks are learned, which is know as catastrophic forgetting. In order to achieve a good performance on all tasks, Li & Hoiem (2016) proposed to record target examples' outputs on the source tasks from the original network and attempt to preserve these outputs when training on the target task. They also tried to preserve the pre-trained parameters instead of the outputs of examples but they did not obtain satisfying results. Kirkpatrick et al. (2017) computed the Fisher information matrix from source data to measure each parameter's importance to the source tasks. Then they aimed at learning a new task without forgetting previous tasks by elastic weight consolidation, which is similar to L^2-*SP* approach. Fisher information has been used in elastic weight consolidation in order to provide an amount proportional to each parameter's importance on source data. However, when training with target data, Fisher information is not relevant unless the target domain is similar to the source domain and the target task is the same as the source task. Even in their experiments, for each target task, fine-tuning with plain stochastic gradient descent got better performance than using elastic weight consolidation. We argue that for a single target task, elastic weight consolidation is a troubling term for optimization because the target task shuffles data with a random permutation so that the weights in the first layer and Fisher information do not respectively correspond.

In domain adaptation (Long et al. 2015b, Tzeng et al. 2016) where the target task is the same as the source task and no (or a small quantity of) target data are labeled, most approaches are searching for a common representation space for source and target domains to reduce domain shift. Rozantsev et al. (2016) introduced a parameter regularization for keeping the similarity between the pre-trained model and the fine-tuned model. Since domain adaptation needs more effort on reducing domain shift, their regularization was more flexible with the exponential function of a linear transformation. We found in our preliminary experiments that the exponential term was able to improve the results but not as much as L^2-*SP*. The gradient of the exponential term indicates that when the weight goes farther, the force for bringing it back is exponentially stronger.

There are many other parameter regularization methods in machine learning but to the best of our knowledge, besides those mentioned in (Li & Hoiem 2016, Kirkpatrick et al. 2017, Rozantsev et al. 2016), we have not observed other regularization approaches that carry information gained from solving other learning tasks. Despite that, it is encouraging to find similar formulas that help to obtain superior results in both lifelong learning and domain adaptation.

3 Tested Regularization Approaches

We typically deal with relatively small target databases, for which parameter regularization is often critical for the learning process. When learning from scratch, regularization is aimed at facilitating optimization and avoiding overfitting, by implicitly restricting the capacity of the network, that is, the effective size of the search space. In transfer learning, the role of regularization is similar, but the starting point of the fine-tuning process conveys information that pertain to the source problem (domain and task). Hence, the network capacity has not to be restricted blindly: the pre-trained model sets a reference that can be used to define the functional space effectively explored during fine-tuning.

In this section, we detail the penalties we introduce for fine-tuning. Since we are using early stopping, fine-tuning a pre-trained model is an implicit form of induction bias towards the initial solution. We explore here how an explicit induction bias encoded by a regularization term affects the training process. Section 4 shows that all such schemes get an edge over the standard approaches that either use weight decay or freeze part of the network for preserving the low-level representations that are built in the first layers of the network.

Let $\boldsymbol{w} \in \mathbb{R}^n$ be the parameter vector containing all the network parameters that are to be adapted to the target domain. The regularized objective function $\tilde{J}$ that is to be optimized is the sum of the standard objective function J and the regularizer $\alpha\Omega(\boldsymbol{w})$, where α is the regularization parameter. In our experiments, J is the negative log-likelihood, so that the criterion $\tilde{J}$ could be interpreted in terms of maximum *a posteriori* estimation, where the penalty $\alpha\Omega(\boldsymbol{w})$ would act as the log prior of $\boldsymbol{w}$.

More generally, the minimizer of $\tilde{J}$ is a trade-off between the data-fitting term and the regularization term.

L^2 penalty Our baseline penalty for transfer learning is the usual L^2 penalty, also known as weight decay, since it drives the weights of the network to zero:

$$\Omega(\boldsymbol{w}) = \frac{1}{2} \|\boldsymbol{w}\|_2^2 \quad , \tag{1}$$

where $\|\cdot\|_p$ is the p-norm of a vector.

L^2-SP Let $\boldsymbol{w}^0$ be the parameter vector of the model pre-trained on the source problem, acting as the starting point (*-SP*) in fine-tuning. Using this initial vector as the reference in the L^2 penalty, we get:

$$\Omega(\boldsymbol{w}) = \frac{1}{2} \left\|\boldsymbol{w} - \boldsymbol{w}^0\right\|_2^2 \quad . \tag{2}$$

Typically, the transfer to a target task requires slight modifications of the network architecture used in the source domain, such as on the last layer used for predicting the outputs. Then, there is no one-to-one mapping between $\boldsymbol{w}$ and $\boldsymbol{w}^0$, and we use two penalties: one for the part of the target network that shares the architecture of the source network, denoted $\boldsymbol{w}_{\mathcal{S}}$, the other one for the novel part, denoted $\boldsymbol{w}_{\bar{\mathcal{S}}}$. The compound penalty then becomes:

$$\Omega_\alpha(\boldsymbol{w}) = \frac{1}{2} \left\|\boldsymbol{w}_{\mathcal{S}} - \boldsymbol{w}_{\mathcal{S}}^0\right\|_2^2 + \frac{\beta}{2} \|\boldsymbol{w}_{\bar{\mathcal{S}}}\|_2^2 \quad . \tag{3}$$

***L^2-SP*-Fisher** Elastic weight consolidation (Kirkpatrick et al. 2017) was proposed to avoid catastrophic forgetting in the setup of lifelong learning, where several tasks should be learned sequentially. In addition to preserving the initial parameter vector $\boldsymbol{w}^0$, it consists in using the estimated Fisher information to define the distance between $\boldsymbol{w}_{\mathcal{S}}$ and $\boldsymbol{w}_{\mathcal{S}}^0$. More precisely, it relies on the diagonal of the Fisher information matrix, resulting in the following penalty:

$$\Omega_\alpha(\boldsymbol{w}) = \frac{1}{2} \sum_{j \in \mathcal{S}} \hat{F}_{jj} \left(w_j - w_j^0\right)^2 + \frac{\beta}{2} \|\boldsymbol{w}_{\bar{\mathcal{S}}}\|_2^2 \quad , \tag{4}$$

where $\hat{F}_{jj}$ is the estimate of the jth diagonal element of the Fisher information matrix. It is computed as the average of the squared Fisher's score on the source problem, using the inputs of the source data:

$$\hat{F}_{jj} = \frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} f_k(\boldsymbol{x}^{(i)}; \boldsymbol{w}^0) \left(\frac{\partial}{\partial w_j} \log f_k(\boldsymbol{x}^{(i)}; \boldsymbol{w}^0) \right)^2 \quad ,$$

where the outer average estimates the expectation with respect to inputs $\boldsymbol{x}$ and the inner weighted sum is the estimate of the conditional expectation of outputs given input $\boldsymbol{x}^{(i)}$, with outputs drawn from a categorical distribution of parameters $(f_1(\boldsymbol{x}^{(i)}; \boldsymbol{w}), \ldots, f_k(\boldsymbol{x}^{(i)}; \boldsymbol{w}), \ldots, f_K(\boldsymbol{x}^{(i)}; \boldsymbol{w}))$.

L^1-SP We also experiment the L^1 variant of L^2*-SP*:

$$\Omega_\alpha(\boldsymbol{w}) = \left\|\boldsymbol{w}_{\mathcal{S}} - \boldsymbol{w}_{\mathcal{S}}^0\right\|_1 + \frac{\beta}{2} \|\boldsymbol{w}_{\bar{\mathcal{S}}}\|_2^2 \quad . \tag{5}$$

The usual L^1 penalty encourages sparsity; here, by using $\boldsymbol{w}_{\mathcal{S}}^0$ as a reference in the penalty, L^1-SP encourages some components of the parameter vector to be frozen, equal to the pre-trained initial values. The penalty can thus be thought as intermediate between L^2*-SP* (3) and the strategies consisting in freezing a part of the initial network. We explore below other ways of doing so.

Group-Lasso-*SP* (GL-*SP*) Instead of freezing some individual parameters, we may encourage freezing some groups of parameters corresponding to channels of convolution kernels. Formally, we endow the set of parameters with a group structure, defined by a fixed partition of the index set $\mathcal{I} = \{1, \ldots, p\}$, that is, $\mathcal{I} = \bigcup_{g=0}^{G} \mathcal{G}_g$, with $\mathcal{G}_g \cap \mathcal{G}_h = \emptyset$ for $g \neq h$. In our setup, $\mathcal{G}_0 = \bar{\mathcal{S}}$, and for

$g > 0$, $\mathcal{G}_g$ is the set of fan-in parameters of channel g. Let p_g denote the cardinality of group g, and $\boldsymbol{w}_{\mathcal{G}_g} \in \mathbb{R}^{p_g}$ be the vector $(w_j)_{j \in \mathcal{G}_g}$. Then, the GL-*SP* penalty is:

$$\Omega(\boldsymbol{w}) = \sum_{g=0}^{G} \alpha_g \left\| \boldsymbol{w}_{\mathcal{G}_g} - \boldsymbol{w}^0_{\mathcal{G}_g} \right\|_2 \quad , \tag{6}$$

where $\boldsymbol{w}^0_{\mathcal{G}_0} = \boldsymbol{w}^0_{\bar{\mathcal{S}}} \triangleq \mathbf{0}$, and, for $g > 0$, α_g is a predefined constant that may be used to balance the different cardinalities of groups. In our experiments, we used $\alpha_g = p_g^{1/2}$.

Our implementation of Group-Lasso-*SP* can freeze feature extractors at any depth of the convolutional network, to preserve the pre-trained feature extractors as a whole instead of isolated pre-trained parameters. The group $\mathcal{G}_g$ of size $p_g = h_g \times \mathrm{w}_g \times d_g$ gathers all the parameters of a convolution kernel of height h_g, width w_g, and depth d_g. This grouping is done at each layer of the network, for each output channel, so that the group index g corresponds to two indexes in the network architecture: the layer index l and the output channel index at layer l. If we have c_l such channels at layer l, we have a total of $G = \sum_l c_l$ groups.

Group-Lasso-*SP*-Fisher (GL-*SP*-Fisher) Following the idea of L^2-*SP*-Fisher, the Fisher version of GL-*SP* is:

$$\Omega(\boldsymbol{w}) = \alpha_0 \left\| \boldsymbol{w}_{\bar{\mathcal{S}}} \right\|_2 + \sum_{g=1}^{G} \alpha_g \left(\sum_{j \in \mathcal{G}_g} \hat{F}_{jj} \left(w_j - w_j^0 \right)^2 \right)^{1/2} \quad . \tag{7}$$

4 Experiments

We evaluate the aforementioned parameter regularization approaches in Section 3 on different target tasks by fine-tuning pre-trained models. Fine-tuning is the most favorable technique to resolve these tasks because it can maximally exploit the capacity of convolutional network and the information gained in the source database while limiting risks of overfitting. We use ResNet (He et al. 2016), which has proven its good generality on many transfer learning tasks, as the base network. Conventionally, if the target task is also a classification task, the training process starts from replacing the last layer of classification with a new layer whose size depends on the number of classes in the target task. All mentioned parameter regularization approaches can be applied to all layers except new layers, and parameters in new layers are regularized by L^2 penalty in our experiments.

4.1 Source and Target Databases

For comparing the effect of similarity between source databases and target databases on transfer learning, ImageNet (Deng et al. 2009) for generic object recognition and Places365 (Zhou et al. 2017) for scene classification have been chosen as source databases. Two groups of experiments are performed. One uses ImageNet as the source database and the other uses Places365. Both of them will be transferred to the target tasks described below, with different regularization approaches.

We select three different databases for three different target tasks: Caltech 256 (Griffin et al. 2007) contains different objects for generic object recognition, similar to ImageNet; Stanford Dogs 120 (Khosla et al. 2011) contains images of 120 breeds of dogs; MIT Indoors 67 (Quattoni & Torralba 2009) consists of 67 indoor scene categories. See Table 1 for details.

Each target database is split into training and testing sets following the suggestion of their creators. We keep two configurations for Caltech 256: randomly choose 30 examples or 60 examples per category for training, then test on the remaining examples. Stanford Dogs 120 has exactly 100 examples per category for training and its test set contains 8580 examples in total. As for MIT Indoors 67, there are exactly 80 examples per category for training and 20 examples per category for testing.

Table 1: Characteristics of the target databases: name and type, numbers of training and test images per class, and number of classes

Database	task category	# training images	# test images	# classes
Caltech 256 – 30	generic object recog.	30	> 20	257
Caltech 256 – 60	generic object recog.	60	> 20	257
Stanford Dogs 120	specific object recog.	100	~ 72	120
MIT Indoors 67	scene classification	80	20	67

Table 2: Average classification accuracies of L^2 and L^2-*SP* on different tasks, on 5 different runs. The first line is the reference of selective joint fine-tuning (Ge & Yu 2017) that selects 200,000 images from the source database during transfer learning.

	MIT Indoors 67 [1]	Stanford Dogs 120	Caltech 256 - 30	Caltech 256 - 60
Ge & Yu (2017)	85.8	90.2	83.8	89.1
L^2	79.28$\pm$0.88	81.77$\pm$0.08	82.15$\pm$0.27	86.22$\pm$0.07
L^2-*SP*	84.18$\pm$0.41	84.84$\pm$0.06	**85.31**$\pm$0.19	**89.66**$\pm$0.09

4.2 Training Details

Most images in those databases are color images. If not, we create a three-channel image by duplicating the gray-scale data. All images are pre-processed: we resize images to 256$\times$256 and subtract the mean activity computed over the training set from each channel, then we adopt random blur, random mirror and random crop to 224$\times$224 for data augmentation. Parameters in the last layer are regularized by L^2 and other parameters are regularized by the methods described in Section 3. These two kinds of parameters receive different regularization strengths. The Fisher information matrix, used in in L^2-*SP*-Fisher and GL-*SP*-Fisher, is estimated on the source database. Different source databases (ImageNet or Places365) yield different estimates. Regarding testing, we use central crops as inputs to compute the classification accuracy. Stochastic gradient descent with momentum 0.9 is used for optimization. We run 9000 iterations and divide the learning rate by 10 after 6000 iterations. The initial learning rates are 0.005, 0.01 or 0.02, depending on the tasks. Batch size is 64. Cross validation is used for searching the best regularization hyperparameters. 0.01 is found as the best $\alpha\beta$ for regularizing $\boldsymbol{w}_{\bar{\mathcal{S}}}$, while α differs a lot depending on the regularization approach. Then under the best configuration, we repeat five times learning process to obtain an average classification precision and standard deviation. All the experiments are performed with Tensorflow (Abadi et al. 2015).

4.3 Results

4.3.1 Fine-Tuning with L^2-*SP* *vs.* L^2

Table 2 displays results of fine-tuning with L^2-*SP*, which is compared to the baseline of fine-tuning with L^2, and the state-of-the-art reference of selective joint fine-tuning (Ge & Yu 2017). Note that we use the same experimental protocol, except that Ge & Yu (2017) allow 200,000 additional images from the source problem to be used during transfer learning, whereas we did not use any.

We report the average accuracies and their standard deviations on 5 different runs. Since we use the same data and start from the same starting point, runs differ only due to the randomness of stochastic gradient descent and to the weight initialization of the last layer. Our results with L^2 penalty are consistent with Ge & Yu (2017).

In all experiments reported in Table 2 and later in Figure 1, the worst run of fine-tuning with L^2-*SP* is significantly better than the best run of the standard L^2 fine-tuning according to classical pairwise tests at the 5% level. Furthermore, in spite of its simplicity, the worst run of L^2-*SP* fine-tuning outperforms the state-of-the-art results of Ge & Yu (2017) on the two Caltech 256 setups at the 5% level.

4.3.2 A CONSISTENT BEHAVIOR ACROSS PENALTIES, SOURCE AND TARGET DATABASES

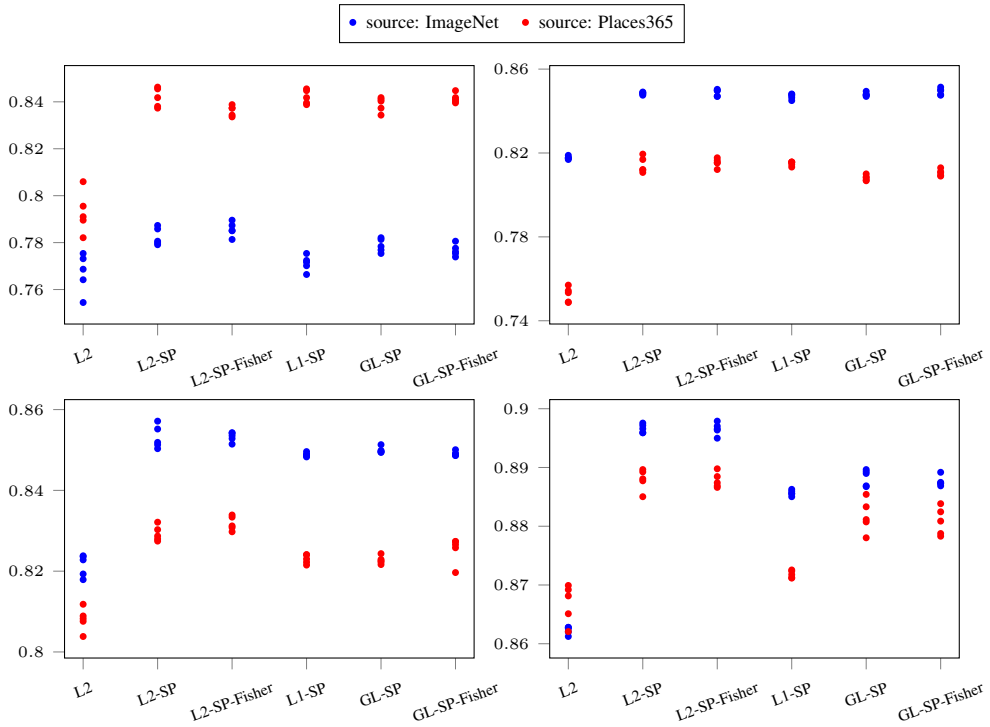

Figure 1: Classification accuracies of all tested approaches using ImageNet or Places365 as source databases. Top left: MIT Indoor 67; top right: Stanford Dogs 120; bottom left: Caltech 256 with 30 examples per class; bottom right: Caltech 256 with 60 examples per class.

A comprehensive view of our experimental results is given in Figure 1. Each plot corresponds to one of the four target databases listed in Table 1. The light red points mark the accuracies of transfer learning when using Places365 as the source database, whereas the dark blue points correspond to the results obtained with ImageNet. As expected, the results of transfer learning are much better when source and target are alike: the scene classification target task MIT Indoor 67 (top left) is better transferred from the scene classification source task Places365, whereas the object recognition target tasks benefit more from the object recognition source task ImageNet. It is however interesting to note that the trends are alike for the two source databases: all the fine-tuning strategies based on penalties using the starting point -*SP* as a reference perform consistently better than standard fine-tuning (L^2). There is thus a benefit in having an explicit bias towards the starting point, even when the target domain and task that are not too close to the source domain and task. This benefit tends to be comparable for all -*SP* penalties, although the strategies based on L^1 and Group-Lasso penalties tend to behave poorly in comparison to the simple L^2-*SP* penalty. We suspect that the standard stochastic gradient descent optimization algorithm that we used throughout all experiments is not well suited to these penalties: these penalties have a discontinuity at the starting point where the optimization starts. We implemented a classical smoothing technique to avoid these discontinuities, but it did not help. Finally, the variants using the Fisher information matrix behave like the simpler variants with a Euclidean metric on parameters. We believe that this is due to the fact that, contrary to lifelong learning, our objective does not favor solutions that keep the memory of the source task. The metric defined by the Fisher information matrix may thus be irrelevant for our actual objective that only relates to the target database.

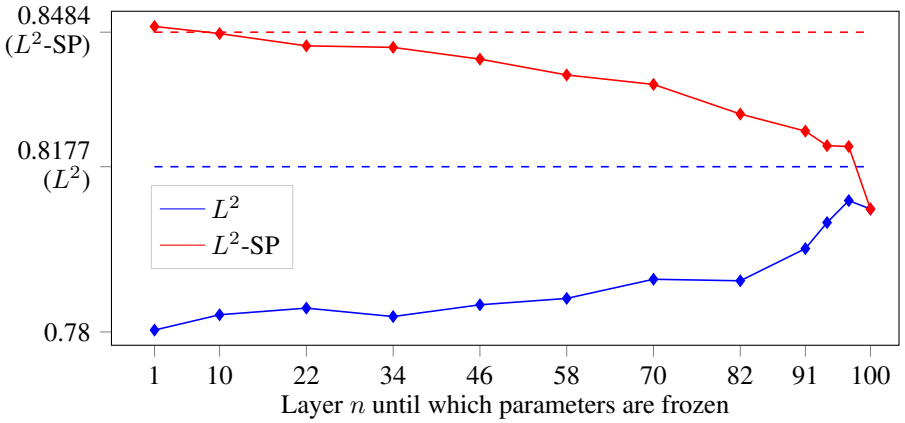

Figure 2: Classification accuracies of fine-tuning with L^2 and L^2-*SP* on Stanford Dogs 120 when freezing the first n layers of ResNet-101. The dashed lines correspond to the results in Table 2 where no layers are frozen. Note that ResNet-101 begins with one convolutional layer, then stacks 3-layer blocks. The three layers in one block are either frozen or trained altogether.

4.3.3 Fine-Tuning with L^2-*SP* *vs.* Freezing the Network

Freezing the first layers of a network during transfer learning is another way to ensure a very strong induction bias, letting only few degrees of freedom to transfer learning in the last layers. Figure 2 shows that this strategy, which is costly to implement if one looks for the optimal number of layers to be frozen, can dramatically help L^2 fine-tuning, but does not bring any improvements to L^2-*SP* fine-tuning, which allows small modifications at all levels of the network.

4.4 Analysis and Discussion

Among all -*SP* methods, while the results of different methods are similar, L^2-*SP* and L^2-*SP*-Fisher always have marginal lead to others. We expected L^2-*SP*-Fisher to outperform L^2-*SP*, since Fisher information helps in continual learning, but the objective here is different. Since L^2-*SP* is simpler than L^2-*SP*-Fisher, we recommended the former, and we focus on the analysis of L^2-*SP*, although most of the analysis and the discussion would also apply to the other regularization approaches.

Theoretical Explanation Under some simplifying assumptions, we show in Appendix A that the optimum of the regularized objective function with L^2-*SP* is a compromise between the optimum of the unregularized objective function and the pre-trained parameter vector, precisely an affine combination along the directions of eigenvectors of Hessian matrix of the unregularized objective function. This contrasts with L^2 that leads to a compromise between the optimum of the unregularized objective function and the origin. Clearly, searching a solution around the pre-trained parameter vector is intuitively much more appealing.

Linear Dependence To further demonstrate the effect of L^2-*SP* on training phase, we measured the linear dependence between the activations using the pre-trained parameters and the fine-tuned parameters. We considered feature-wise R^2 coefficients of the activations before and after fine-tuning to measure their dependence. The box plots are displayed in Figure 3 versus depth. We see that according to this measure, based on activations, the L^2-*SP* fine-tuning retains more of the pretrained network at all depths.

Stronger Regularization Stronger regularization of L^2 will result in the weights being too small to be effective feature detectors. As for L^2-*SP*, stronger regularization will not lose the capacity of network but only involve more strength to avoid overfitting, especially when the target database is small. Even when the target database is moderate or relatively large, L^2-*SP* is still a better choice

[1]The results of MIT Indoors 67 are obtained using Places365 as source database.

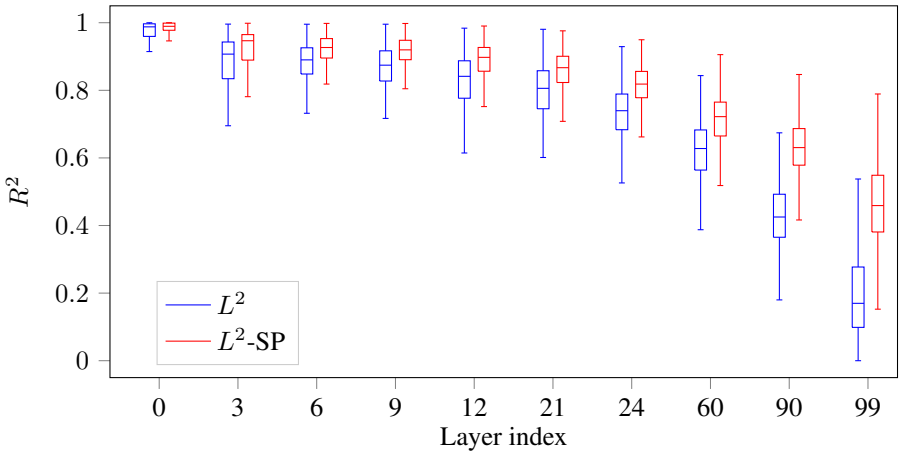

Figure 3: R^2 coefficients of determination with L^2 and L^2-*SP* regularizations. For each feature in one layer, using training examples of Stanford Dogs 120, we computed the R^2 coefficients of the activations before and after fine-tuning and obtained a distribution for this layer. For each layer, we traced the distribution using box-and-whisker diagrams. ResNet-101 begins with one convolutional layer, then stacks 3-layer blocks and ends with one classifier layer. We investigate here only the first layer and the outputs of some 3-layer blocks, noting that the layer index starts from 0.

than L^2. For the tasks that are different from the source task, the strength of regularization should be appropriately reduced.

Computational Efficiency All -*SP* methods have no extra parameters, only increasing little computation. L^2-*SP* augments floating point operations of ResNet-101 by less than 1%. At little cost of computation, we can obtain 3∼4% performance increase in classification accuracy. Furthermore, no cost is experienced at test time.

5 Conclusion

We proposed simple regularization techniques in inductive transfer learning, to encode an explicit bias towards the solution learned on the source domain. Most of the regularizers evaluated here have been already used for other purposes, but we demonstrate here their relevance for deep convolutional networks in inductive transfer learning.

We show that a simple L^2 penalty using the starting point as a reference is useful, even if early stopping is used. This penalty is much more efficient that the standard L^2 penalty that is commonly used in fine-tuning. It is also more efficient and simpler to implement than the strategy consisting in freezing the first layers of a network. We also provide theoretical arguments and experimental evidence showing that L^2-*SP* retains the memory of the features learned on the source database.

Besides, we tested the effect of more elaborate penalties, based on L^1 or Group-L^1 norms, or based on Fisher information. None of these options seem to be valuable in the context of inductive transfer learning that we considered here. However, different approaches, which implement an implicit bias at the functional level, alike (Li & Hoiem 2016), remain to be tested: being based on a different principle, their value should be assessed in the framework of inductive transfer learning.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vin-

cent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *PAMI*, 2017.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *PAMI*, 2006.

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

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Adaptive Computation and Machine Learning. MIT Press, 2017. URL `http://www.deeplearningbook.org`.

Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. Technical report, California Institute of Technology, 2007.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*, 2011.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *ECCV*, 2016.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015a.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *ICML*, 2015b.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

Anastasia Pentina and Christoph H Lampert. Lifelong learning with non-iid tasks. In *NIPS*, 2015.

Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *CVPR*, 2009.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

Artem Rozantsev, Mathieu Salzmann, and Pascal Fua. Beyond sharing weights for deep domain adaptation. *arXiv preprint arXiv:1603.06432*, 2016.

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

Nathan Srebro and Adi Shraibman. Rank, trace-norm and max-norm. In *COLT*, volume 5, pp. 545–560. Springer, 2005.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

Sebastian Thrun and Tom M Mitchell. Lifelong robot learning. *Robotics and Autonomous Systems*, 15(1-2):25–46, 1995.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *NIPS Workshop on Adversarial Training (WAT)*, 2016.

Di Xie, Jiang Xiong, and Shiliang Pu. All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation. *arXiv preprint arXiv:1703.01827*, 2017.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *PAMI*, 2017.

A EFFECT OF L^2-*SP* REGULARIZATION ON OPTIMIZATION

The effect of L^2 regularization can be analyzed by doing a quadratic approximation of the objective function around the optimum (see, e.g. Goodfellow et al. 2017, Section 7.1.1). This analysis shows that L^2 regularization rescales the parameters along the directions defined by the eigenvectors of the Hessian matrix. This scaling is equal to $\frac{\lambda_i}{\lambda_i+\alpha}$ for the i-th eigenvector of eigenvalue λ_i. A similar analysis can be used for the L^2-*SP* regularization.

We recall that $J(\boldsymbol{w})$ is the unregularized objective function, and $\tilde{J}(\boldsymbol{w}) = J(\boldsymbol{w}) + \alpha \left\|\boldsymbol{w} - \boldsymbol{w}^0\right\|_2^2$ is the regularized objective function. Let $\boldsymbol{w}^* = \operatorname{argmin}_{\boldsymbol{w}} J(\boldsymbol{w})$ and $\tilde{\boldsymbol{w}} = \operatorname{argmin}_{\boldsymbol{w}} \tilde{J}$ be their respective minima. The quadratic approximation of $J(\boldsymbol{w}^*)$ gives

$$\mathbf{H}(\tilde{\boldsymbol{w}} - \boldsymbol{w}^*) + \alpha(\tilde{\boldsymbol{w}} - \boldsymbol{w}^0) = 0 \ , \tag{8}$$

where $\mathbf{H}$ is the Hessian matrix of J w.r.t. $\boldsymbol{w}$, evaluated at $\boldsymbol{w}^*$. Since $\mathbf{H}$ is positive semidefinite, it can be decomposed as $\mathbf{H} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T$. Applying the decomposition to Equation (8), we obtain the following relationship between $\tilde{\boldsymbol{w}}$ and $\boldsymbol{w}^*$:

$$\mathbf{Q}^T\tilde{\boldsymbol{w}} = (\boldsymbol{\Lambda} + \alpha\mathbf{I})^{-1}\boldsymbol{\Lambda}\mathbf{Q}^T\boldsymbol{w}^* + \alpha(\boldsymbol{\Lambda} + \alpha\mathbf{I})^{-1}\mathbf{Q}^T\boldsymbol{w}^0 \ . \tag{9}$$

We can see that with L^2-*SP* regularization, in the direction defined by the i-th eigenvector of $\mathbf{H}$, $\tilde{\boldsymbol{w}}$ is a convex combination of $\boldsymbol{w}^*$ and $\boldsymbol{w}^0$ in that direction since $\frac{\lambda_i}{\lambda_i+\alpha}$ and $\frac{\alpha}{\lambda_i+\alpha}$ sum to 1.