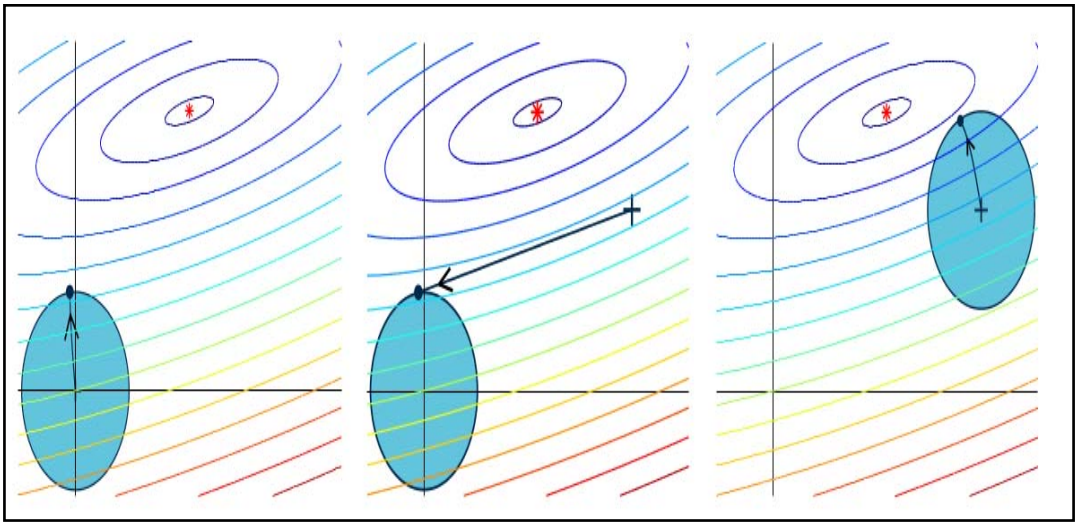
Regularization schemes for transfer learning with convolutional networks

Xuhong Li

Par **Xuhong LI**

Regularization schemes for transfer learning with convolutional networks

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC



Soutenue le 10 septembre 2019
Spécialité : Informatique : Unité de recherche Heudyasic (UMR-7253)

D2497

UNIVERSITÉ DE TECHNOLOGIE DE COMPIÈGNE

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor

Spécialité : Informatique

Area of specialization: Computer Science

by

XUHONG LI

Regularization Schemes for Transfer Learning with Convolutional Networks

10/09/2019

Heudiasyc Laboratory, UMR UTC/CNRS 7253

Thesis Committee:

Rapporteurs	Nicolas Thome	Professor CNAM CEDRIC Lab
	Gilles Gasso	Professor INSA Rouen
Examinateurs	Elisa Fromont	Professor Université Rennes 1
	Alain Rakotomamonjy	Professor Université de Rouen
	Véronique Cherfaoui	Professor Université de technologie de Compiègne
Directeurs	Yves Grandvalet	CNRS senior researcher
	Franck Davoine	CNRS researcher

Acknowledgement

First and foremost, I would like to express my gratitude to my supervisors of the thesis, Yves Grandvalet and Franck Davoine, for their continuous support of my PhD life, for their patience, motivation, and immense knowledge. Franck and Yves have taught me the way of thinking and researching as professional researcher, the tenacity of perfecting a job, as well as the professional knowledge in the field of statistics, machine learning and computer vision. I appreciate every discussion, meeting with them that gave me the motivation to keep working on challenging problems. I am grateful for their wise guidance throughout my thesis, and I feel so lucky to work under their supervision.

I would like to thank the committee members for joining my defense of the thesis. Many thanks to Nicolas Thome and Gilles Gasso for reviewing my manuscript during the summer vacation, and to Elisa Fromont, Alain Rakotomamonjy and Véronique Cherfaoui for all their comments and insightful suggestions for improving the thesis.

I also want to thank the Chinese Scholarship Council (CSC) from Chinese government for the financing during my PhD, and Heudiasyc laboratory, UMR UTC/CNRS 7253 for supporting me to participate in international conferences, and for buying the powerful GPU servers at the beginning of my thesis that makes it possible for me to finish my experiments.

I warmly thank my labmates of Heudiasyc, whose friendships made me have a memorable experience at UTC. I also appreciate my dear Chinese friends for their help and encouragement. I could not have smoothly survived from the PhD thesis without their accompanying. I would like to give my sincere thanks to my parents for their unconditional love and support.

Last but not least, I thank Xiaolei, my best friend and my wife, for understanding me and helping me throughout these years with all her efforts.

Abstract

Transfer learning with deep convolutional neural networks significantly reduces the computation and data overhead of the training process and boosts the performance on the target task, compared to training from scratch. However, transfer learning with a deep network may cause the model to forget the knowledge acquired when learning the source task, leading to the so-called catastrophic forgetting. Since the efficiency of transfer learning derives from the knowledge acquired on the source task, this knowledge should be preserved during transfer. This thesis solves this problem of forgetting by proposing two regularization schemes that preserve the knowledge during transfer. First we investigate several forms of parameter regularization, all of which explicitly promote the similarity of the final solution with the initial model, based on the L^1, L^2, and Group-Lasso penalties. We also propose the variants that use Fisher information as a metric for measuring the importance of parameters. We validate these parameter regularization approaches on various tasks. The second regularization scheme is based on the theory of optimal transport, which enables to estimate the dissimilarity between two distributions. We benefit from optimal transport to penalize the deviations of high-level representations between the source and target task, with the same objective of preserving knowledge during transfer learning. With a mild increase in computation time during training, this novel regularization approach improves the performance of the target tasks, and yields higher accuracy on image classification tasks compared to parameter regularization approaches.

Keywords — transfer learning, regularization, convolutional networks, optimal transport, computer vision.

Résumé

L'apprentissage par transfert de réseaux profonds réduit considérablement les coûts en temps de calcul et en données du processus d'entraînement des réseaux et améliore largement les performances de la tâche cible par rapport à l'apprentissage à partir de zéro. Cependant, l'apprentissage par transfert d'un réseau profond peut provoquer un oubli des connaissances acquises lors de l'apprentissage de la tâche source. Puisque l'efficacité de l'apprentissage par transfert vient des connaissances acquises sur la tâche source, ces connaissances doivent être préservées pendant le transfert. Cette thèse résout ce problème d'oubli en proposant deux schémas de régularisation préservant les connaissances pendant l'apprentissage par transfert. Nous examinons d'abord plusieurs formes de régularisation des paramètres qui favorisent toutes explicitement la similarité de la solution finale avec le modèle initial, par exemple, L^1, L^2, et Group-Lasso. Nous proposons également les variantes qui utilisent l'information de Fisher comme métrique pour mesurer l'importance des paramètres. Nous validons ces approches de régularisation des paramètres sur différentes tâches de segmentation sémantique d'image ou de calcul de flot optique. Le second schéma de régularisation est basé sur la théorie du transport optimal qui permet d'estimer la dissimilarité entre deux distributions. Nous nous appuyons sur la théorie du transport optimal pour pénaliser les déviations des représentations de haut niveau entre la tâche source et la tâche cible, avec le même objectif de préserver les connaissances pendant l'apprentissage par transfert. Au prix d'une légère augmentation du temps de calcul pendant l'apprentissage, cette nouvelle approche de régularisation améliore les performances des tâches cibles et offre une plus grande précision dans les tâches de classification d'images par rapport aux approches de régularisation des paramètres.

Keywords — apprentissage par transfert, régularisation, réseaux de neurones à convolution, transport optimal, vision par ordinateur.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Contents

1.1 Context and Motivation

Deep neural networks have been applied to fields including computer vision, natural language processing, speech recognition, machine translation as well as board game programs, where they have produced results comparable to and in some cases superior to human experts. These deep network models contain millions of parameters to be estimated and require billions of operations for a single forward pass, leading to an obligation of enormous computation resources for fast experimental evaluations, numerous training data for assuring the desired performance, and a delicate design of network structures together with a good initialization of parameters for the gradient-based optimization. Facing these problems, it is not practical for each single task to collect a lot of training data, or spend weeks or months on searching the best training scheme. Few companies and laboratories are capable of launching such costly experiments.

Transfer learning can yet relieve the pain. The idea of transfer learning is to gain prior knowledge from solving a source task, and to exploit the stored knowledge to solve a different but related target task. Fortunately, the open-source environment in

this community encourages the release of large-scale datasets and pre-trained deep network models, which boosts a lot the research in the community. During transfer learning, some form of knowledge that is learned from the source task is then transferred to the target task and exploited to improve the performance. Technically, fine-tuning, a common tool for transferring knowledge, initializes the model with the pre-trained parameter values, and continues training on the target dataset for adaption to an optimal solution for the target task. Through fine-tuning and transfer learning, the training process can be largely shortened to hours or minutes, and less prone to overfitting when there are few data available on the target task. However, transfer learning with deep networks may cause the model to easily forget the knowledge learned from the source, leading to the known *catastrophic forgetting*. Since the efficiency of transfer learning derives from the learned knowledge in the source task, the knowledge should be preserved during transfer learning, especially when the source task is "rich" and contains a large-scale dataset.

This thesis thus proposes approaches for preserving the knowledge during transfer learning. We propose to implement this *inductive bias* of preserving the source knowledge for transfer learning problems through *regularization* approaches. In most cases of machine learning, data are split into training set and test set, which are used for training the model and measuring the performance of the learned model respectively. Regularization approaches are designed, via various ways, to reduce the test error, *i.e.* the error that the learned model makes on the test data set, sometimes at the expense of the increased error on the training data set. The regularization approaches are applied during the training phase and hence cause increased computation during training, but no additional operations for inference during testing. In addition, the increased computation during training is usually mild compared to the computation of the input data passing through the deep network.

Many regularization approaches are actually motivated by an inductive bias towards a desired model or a preferred property in the learned model, and, as we discussed before, the desired one in this thesis is to maximally preserve the *knowledge* from the source task. In terms of knowledge, we propose two points of view:

- If two models with the same structure have similar parameters, then they contain similar knowledge.
- If for any input, two models always yield similar outputs, then they contain similar knowledge.

The first point of view may need delicacy on the definition of similar parameters, since each parameter is in different scale for the contribution of the model's discrimination capacity. We tackle this problem via the *Fisher information matrix*, and compare with simply ignoring it, and find that in practice, the different scales for parameters are not so bothersome. The second point view also needs some concrete definitions on the similar outputs. We consider measuring the outputs at the penultimate layer of a deep network, as they can be seen as the most advanced features extracted by the network for the discrimination layer.

Thereby, we propose to transfer the knowledge through parameters and representations, based on which we respectively propose two different families of regularization approaches in this thesis. While the proposed regularization approaches can be applied widely in many transfer learning problems, we evaluate them under a particular and practical setting in transfer learning, where a vast amount of data was available for training on the source problem, and some limited amount of labeled data is available for solving the target problem. Note that different from the setting we consider, many applications like domain adaptation, network compression, lifelong learning, reinforcement learning, where knowledge can be accumulated, may also benefit from our proposed regularization approaches.

1.2 Contributions and Thesis Outline

Following the motivations and the propositions, this thesis is simply organized as follows:

In Chapter 2, we present the background concepts. First we introduce the convolutional neural networks, including the unit operations, optimization algorithms and recent advances in computer vision. Then we recall the transfer learning definition and categorization, and discuss the applications of convolutional networks in transfer learning. After that, we introduce a mathematical tool of measuring the distance between two probability distributions, the optimal transport theory, which is exploited to preserve the source knowledge during transfer learning and based on which we develop the regularization approach on representations. At the end of Chapter 2, we give a general introduction on the regularization approaches in deep learning and some thoughts about using regularization as inductive bias for better learning.

In Chapter 3, we present the regularization approaches that are based on parameters for transfer learning with convolutional networks. We show that they all encode an

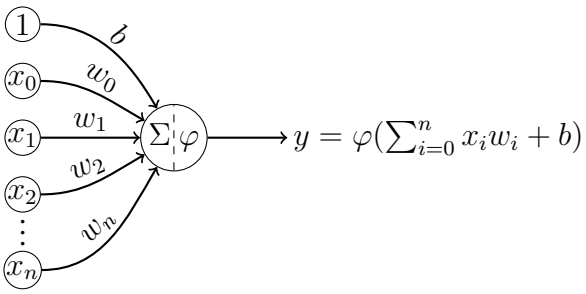


Figure 1.1: Illustration of a neuron and an activation. φ is the activation function, b is the bias, and the output y can be also referred as activation or activity.

explicit bias towards the pre-trained parameters learned on the source task, preserving the source knowledge. We validate the parameter regularization approaches on various experiments in image classification, image segmentation, video segmentation and optical flow estimation. Analyses and theoretical hints are also provided.

In Chapter 4, instead of focusing on the parameters, we propose a novel regularization approach based on the optimal transport theory to penalize the deviations of the output activations on the target task, with the same objective of preserving the knowledge during transfer learning. We show that this novel regularization approach also improves the performance of the target tasks, and yields higher accuracy on image classification tasks than parameter regularization approaches.

In Chapter 5, we conclude the thesis, summarize our contributions and suggest some possible paths to future research.

1.3 Terminology

Before opening this thesis, we would like to clarify some terms in context of machine learning and artificial neural networks.

Neurons, Activations In context of artificial neural networks, an *artificial neuron*, or simply a neuron, is a mathematical function conceived as a model of biological neurons. It consists in two functions, a weighted sum of inputs, and a non-linear *activation function* through which the sum is passed. The output of a neuron is sometimes referred as *activation* or *activity*. Mathematically, a neuron is a function with operations and parameters while an activation is an output value of the neuron, see Figure 1.1. We respect this difference throughout this thesis.

Channels, Kernels, Filters Regarding *convolutional neural networks* where the input is usually an image with RGB color channels, activations at a hidden layer compose a *feature map*, *i.e.* a 3D tensor with the height, the width and the number of *channels*. In computer vision, particularly image processing, a *kernel* is a convolution matrix yet without trainable parameters, used for blurring, sharpening, edge detection and other filters. So sometimes, a kernel can be referred as a *filter*, especially in the field of signal processing. A convolutional kernel in convolutional networks has no difference from the conventional kernel except that the values in the convolutional kernel can be trainable by machine learning algorithms. Usually, the convolution between a kernel and an image produces a channel. So the number of channels at a layer is equal to the number of kernels. Thereby, we do not much distinguish the three notions in this thesis: channels, kernels, filters, regardless of their subtle difference.

Features, Representations, Activations In pattern recognition, extracting pertinent *features* of the data is a crucial step for effective learning algorithms. Traditionally, this step is manually done by feature engineering, but *feature learning* or *representation learning* proposes to replace feature engineering by allowing the machine to both learn the features and use them to perform a specific task. Because of this feature/representation learning scheme, *representations* also refer to features. Features or representations are slightly different from the activations as the activations denote the evaluation of features or representations on particular instances. We respect this difference throughout the thesis.

Parameters, Weights In general, parameters and weights are both trainable variables, embedded in a function. Depending on the variable values, this function can cause different effects and extract different features from the data. Learning algorithms search for the optimal values for these parameters and weights. In neural networks, weights specifically refer to the variables that are multiplied by the inputs of the current layer, and there is an extra variable that is simply multiplied by 1. This extra variable is also trainable and named *bias*. All the weights and biases are the trainable parameters in the neural network.

Chapter 2

Background and Related Work

Contents

Prologue In this chapter, we present the background concepts. First we introduce the convolutional neural networks, including the unit compositions of the network, the evolution of network structures, and recent advances of convolutional networks in computer vision. Then we recall the transfer learning definition and categorization, and discuss the applications of convolutional networks in transfer learning. After that, we introduce a mathematical tool of measuring the distance between two probability distributions, the optimal transport theory, which will be used to develop a regularization approach for preserving the source knowledge during transfer learning. At the end of this chapter, we give a general introduction on the regularization approaches in deep learning and some thoughts about using regularization as inductive bias for better learning.

2.1 Convolutional Neural Networks

Convolutional neural networks have their first success in the 1980s [LeCun et al., 1989] for recognizing handwritten digits from small binarized images. The network used at that time is LeNet [LeCun et al., 1989], a very small convolutional network, but inspires the invention of the first modern convolutional network AlexNet [Krizhevsky et al., 2012], which has won the ImageNet large-scale visual recognition competition [Deng et al., 2009] in 2012 by a wide margin over other models.

2.1.1 Elemental Units in Convolutional Networks

Convolution and Feature Maps The essential operation in convolutional neural network is, without question, the *convolution*. Algorithm 1 presents a direct implementation of convolution for better understanding the convolution operation, without forgetting that it can be largely sped up by parallel computing. With other elemental units, Figure 2.1 shows a simple convolutional neural network for image classification. In this thesis, we consider the notion of layers as the sets of *activations* or the

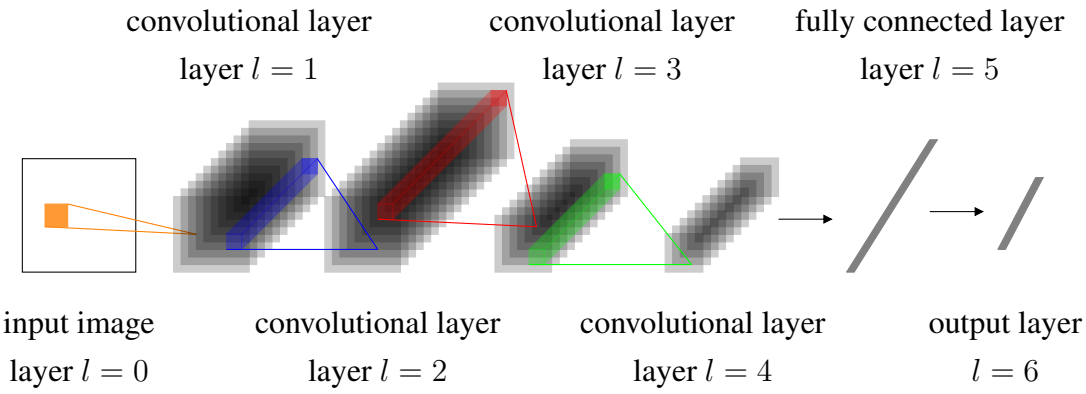


Figure 2.1: An illustration of a simple convolutional network with a single-channel image as input.

outputs of *neurons*, which are determined real values, distinguished from trainable parameters. Note that there are no parameters at layer 0. The slices in black in convolutional layers and the sticks in fully connected layers are the activations or called the feature maps in convolutional networks. The small (sliding) windows with colors present the convolutional *kernels* where the parameters reside. Each of these windows does an element-wise multiplication of kernel parameters and activations within the window, a sum of all the resulted elements, and a non-linear activation function. This operation will be repeated when these windows, with the same parameters, slide onto a new position of the image plane or the feature map plane. As for fully connected layers, all activations are fully connected to those in the previous layer.

Each *feature map* (or the set of activations in the feature map plan) $\mathbf{A}$ in a convolutional layer is a 3D tensor of dimension $W \times H \times C_{out}$, and computed by a 4D tensor $\boldsymbol{k}$ of $C_{in} \times C_{out} \times k \times k$, which is named as convolutional kernel, where C_{in} is the number of feature map slices in the previous layer, C_{out} is the number of feature map slices in this layer, and $k \times k$ is the window size, varying from 1×1 to 7×7. The size of kernel can be set arbitrarily and it controls principally the size of the network. It can be set very large to have enough capacity of the model being representative and yet be prone to overfitting, or it can be very small for speeding up the training and test phrases. For balancing the representative capacity and the risk of overfitting, and also following the practical engineering advice (related to the GPU memory and computation speed), the sizes of convolution kernels always start with a small value in the first layers and then increase gradually along the network with the dimensions of feature

Algorithm 1 A direct implementation of convolution in the l-*th* layer

for c_{out} in 1...C_{out} **do**
  for w in 1...W **do**
    for h in 1...H **do**
      for x in 1...k **do**
        for y in 1...k **do**
          for c_{in} in 1...C_{in} **do**
            $\mathbf{A}_l[w, h, c_{out}] += \mathbf{A}_{l-1}[w + x, h + y, c_{in}] \times \boldsymbol{k}[c_{in}, c_{out}, x, y]$
          end for
        end for
      end for
    end for
  end for
end for

maps decreasing.

The convolution operation enables the parameter sharing and the translation invariance properties. For each position of the feature map, the same parameters are used for extracting the features, as the kernel window slides on the feature map plan. In a traditional neural network, each element of the parameter matrix is used exactly once when extracting the features. The property of sharing parameter thus reduces largely the number of trainable parameters. As for the translation invariance, the convolution helps to detect the translated features. When the input image is translated, the features related to the objects change positions but will still be captured as the convolution kernel slides on the feature plan, and these important features will be given to the next layer and contribute for the discrimination.

Non-Linear Activation Functions Besides the convolution operation, *non-linear activation functions* are another important component in (convolutional) neural networks, and the stack of activation functions in neural networks can help the network to learn complex data, approximate almost any function representing a feature, and provide accurate predictions. Non-linear activation functions are usually simple and computationally efficient. Several choices are preferred in the community, like $\mathbf{tanh}$, $\mathbf{sigmoid}$, $\mathbf{ReLU}$ (rectified linear unit), *etc*. For deep convolutional neural networks,

ReLU is a desirable choice among many activation functions for its efficacy and not easily falling into the gradient diminishing problem.

Pooling *Pooling* is an operation once frequently used in convolutional neural networks, but for modern convolutional networks, pooling layers are replaced by convolution layers with an equivalent stride, while a pooling layer is applied before the last layer for reducing the feature map spatial dimensions by averaging all outputs of each feature map.

2.1.2 Optimization Algorithms of Deep Networks

A learning problem is different from a pure optimization problem. In most machine learning scenarios, we have no access to the true data distribution noted as p_{data}, causing the pure optimization problem unfeasible. Instead, a learning problem exploits an optimization algorithm to minimize a loss function with respect to the empirical distribution, *i.e.* the training data set, noted as $\hat{p}_{data}$, hoping the optimized model performs well with respect to p_{data}, or a related test data set. Throughout this thesis, we develop the supervised case, where the per-example loss function L is related to the output of the model $f(\boldsymbol{x}; \boldsymbol{w})$ and the label y. Then the generalization loss or error $J^*(\boldsymbol{w})$, which measures the performance of the learned model, is the expectation of the per-example loss over the true underlying distribution p_{data}:

$$J^*(\boldsymbol{w}) = \mathbb{E}_{(\boldsymbol{x},y)\sim p_{data}} L(f(\boldsymbol{x}; \boldsymbol{w}), y). \tag{2.1}$$

Since this data generating distribution p_{data} is difficult to access, we cannot directly compute the generalization error. The simplest solution is to minimize the expected loss on the training set $\hat{p}_{data}$:

$$J(\boldsymbol{w}) = \mathbb{E}_{(\boldsymbol{x},y)\sim \hat{p}_{data}} L(f(\boldsymbol{x}; \boldsymbol{w}), y). \tag{2.2}$$

Specifically, this expectation can be obtained by the average over all training examples:

$$\mathbb{E}_{(\boldsymbol{x},y)\sim \hat{p}_{data}} L(f(\boldsymbol{x}; \boldsymbol{w}), y) = \frac{1}{N} \sum_{i=1}^{N} L(f(\boldsymbol{x}_i; \boldsymbol{w}), y_i), \tag{2.3}$$

where N is the number of training examples. Then the learning problem is to search an optimal solution of Equation 2.3 with an optimization algorithm, expecting the solution, *i.e.* the learned model, to have small generalization error. In practice, the

Algorithm 2 Stochastic gradient descent (SGD) update at training iteration k

Require Learning rate $\epsilon^{(k)}$, parameters $\boldsymbol{w}^{(k)}$, mini-batch size m
while stopping criterion not met **do**
    Sample a mini-batch of m examples from the training set $\{\boldsymbol{x}_i\}$ with corresponding targets $\{y_i\}$.
    Compute gradients: $\boldsymbol{g}^{(k)} \leftarrow \nabla_{\boldsymbol{w}} \frac{1}{m} \sum_{i=1}^{m} L(f(\boldsymbol{x}_i; \boldsymbol{w}^{(k)}), y_i)$
    Update parameters: $\boldsymbol{w}^{(k+1)} \leftarrow \boldsymbol{w}^{(k)} - \epsilon^{(k)} \boldsymbol{g}^{(k)}$
end while

algorithms that are used for deep learning are based on stochastic gradient descent, which is slightly different from solving the Equation 2.3. The SGD algorithm is an iterative gradient-based algorithm. In each iteration, the algorithm randomly selects a mini-batch of training examples, to evaluate the gradients of the loss function with respect to the parameters, then update the parameters by a small step of the gradients for minimizing the loss function, and it stops until the stopping criterion is met. Updating parameters through the SGD algorithm is presented in Algorithm 2. More details about the practical and intuitive views of using SGD for optimizing deep networks can be found in [Goodfellow et al., 2017, Section 8.1.2 and 8.1.3].

The SGD algorithm has many variants, including SGD with momentum, SGD with Nesterov momentum [Nesterov, 1983; Sutskever et al., 2013], Adagrad [Duchi et al., 2011], RMSprop [Tieleman and Hinton, 2012], Adam[Kingma and Ba, 2015], to name a few. They are all based on gradients and the difference is the way of updating parameters. We refer the interested readers to Ruder [2016] for the introduction and the motivation of each SGD variant.

An open question still remains about the SGD algorithms. When optimizing a convex function, a minimum is always good because any local minimum is guaranteed to be a global minimum. However, with non-convex functions, such as neural networks, it is possible to have many local minima and we are not sure about these minima. Whether SGD and its variants encounter these local minima and whether these local minima have high generalization error, it still remains an open question in the community. However, experts now suspect that, for sufficiently large neural networks, most local minima have a low generalization error value, and that it is not important to find a true global minimum rather than to find a point in parameter space that has low but not minimal cost [Goodfellow et al., 2017, Section 8.2.2]. This can be done with the SGD algorithms. So in this thesis, we continue to apply the SGD with momentum to

train the convolutional networks.

2.1.3 Recent Advances in Convolutional Neural Network Structures

LeNet and AlexNet are two networks aforementioned, their structures are very simple, starting with several convolutional layers and ending with two or three fully connected layers. This simple design is yet inefficient. In this subsection, we briefly review the problems of this simple design and discuss the advances in neural network structures, including the general designs and the partial components. Since AlexNet, the research on network structure has been wildly explored.

Receptive Field The *receptive field* of a good network was suggested to be large enough to cover the whole image to capture the global information. Five convolutional layers in AlexNet are far away from enough to get a large receptive filed. Simonyan and Zisserman [2015] showed that the sequence of two convolutional layers of kernel size 3×3 is equal to one layer of 5×5 but has less parameters, so they encouraged to use more 3×3 layers instead of one large layer. Based on this idea, Simonyan and Zisserman [2015] proposed a 19-layer network VGG with the same network design (several straightforward convolutional plus three fully connected layers). The "theoretical" receptive field is not widely applicable and sometimes inexplicable for some problems, *e.g.* image segmentation. Despite that this is still an open question, the network VGG is powerful and popular, also robust when applying on the transfer learning problems.

Deeper or Wider Second, rather than going deeper, Szegedy et al. [2015] concatenated four convolutional layers of different sizes and proposed *Inception network* (also called *GoogLeNet*). Instead of choosing a best kernel size, or stacking 3×3 layers, Szegedy et al. [2015] decided to have them all. Moreover, in each block, to avoid the extra parameters when using large size kernels, Szegedy et al. [2015] followed the suggestion in Lin et al. [2014a], *i.e.*, inserting a layer of 1×1 to reduce the number of slices before the convolution of 3×3 or 5×5. This network has a depth of 22 with some width. Inception networks also follow some engineering insights. If all the layers are sequential, the computation of forward and backward cannot move to the next layer before finishing all the computations at the current layer, wasting time to

wait all threads to be finished in the parallel algorithm. In addition, small convolution operations may not occupy all processing units but large ones may need more, leading some processing units free at some moments. So a good engineer design of the several parallel convolutions is possible to make good use of the powerful parallel resources.

Batch Normalization When there are more layers between the input and output, the *vanishing gradient* problem becomes more critical. Since activation functions like $\mathbf{sigmoid}$, $\mathbf{tanh}$ or $\mathbf{ReLU}$, have the gradients in range between 0 and 1, the gradients computed by the chain rule vanish exponentially backwards. Therefore, it happens easily for deep networks that gradients in the first layers are very small and have little or none effect when updating the parameters. For solving this vanishing gradient problem, Ioffe and Szegedy [2015] proposed to achieve a standard normal distribution for each layer by normalizing the inputs via mini-batch statistics. This new mechanism is called *batch normalization*, and integrated into later versions of Inception networks [Ioffe and Szegedy, 2015; Szegedy et al., 2016, 2017], as well as many other networks. In practice, batch normalization accelerates the training process, makes the parameter initialization less important, and improves the performance. We will continue to discuss batch normalization later in Section 2.4.

Residual Networks Another important advance is the *residual networks* (ResNet) [He et al., 2016a], which were also motivated to solve the gradient vanishing problem. Different from normalizing the activations, He et al. [2016a] introduced a unit for learning the residual representation, where the output of l^{th} layer is $\mathbf{A}_l = \mathbf{A}_{l-1} + g(\mathbf{A}_{l-1}; \boldsymbol{w}_l)$. This forces the model to explicitly learn the residual between layers, instead of learning the whole mapping function. In practice, two or three layers are used for constructing the residual unit, and then a shortcut connection for the identity is added to the learned residual. After residual units, the convolutional networks break the depth record and go to 1,000 layers. Furthermore, the ResNet has proved its good representation capacity and transferability, with robust improvements after transfer learning in image classification, object detection, image segmentation and many other visual recognition problems, see the next subsections. The ResNet is a totally different advance from batch normalization, despite that they focus on the same problem of gradients vanishing, and they are actually compatible to each other.

Global Pooling The last problem exposed in AlexNet and even VGG networks, is that the connection between the last convolutional layer and the first fully connected is very extravagant. The feature map of the last convolutional layer will be rearranged into a 1D vector and fully connected to the following layer. Note that the number of parameters in a fully connected layer is the product of the numbers of activations in the connected two layers, so the extra dimensions from the spatial plan (*i.e.* $W \times H$) will incredibly increase the number of parameters in that layer. Although the generalization ability of large networks does not linearly depend on the number of parameters, extra parameters are always disfavoured. However, this problem has been simply solved by averaging all the spatial activations of the last convolutional layer and reducing the spatial dimension to 1×1, which has been applied in ResNet He et al. [2016a,b] and Inception networks Szegedy et al. [2015]; Ioffe and Szegedy [2015]; Szegedy et al. [2016, 2017].

Other Structures In parallel with ResNet and Inception networks, there are several other popular convolutional network architectures: YOLO [Redmon et al., 2016], MobileNets [Howard et al., 2017], DenseNet [Huang et al., 2017] etc. Different from designing the whole network structure, Hu et al. [2018] introduced a *squeeze-and-excitation* (SE) module that can be easily integrated into any network structure, which achieves better results than vanilla networks. Simultaneously, instead of manually devising the architectures, Zoph and Le [2017]; Zoph et al. [2018] proposed *neural architecture search* (NAS) methods by reinforcement learning, and resulted better performance. However, in this thesis, we use ResNet as the main backbone of the model for tasks because of its popularity and good transferability.

2.1.4 Fully Convolutionalizing the Network

As for many other visual tasks rather than image classification, *e.g.*, object detection and image segmentation, convolutional networks are not directly applicable because a) the fully connected layers lead to an obligation of an input image with a fixed dimension while images' dimensions are variant; b) the output of a network is a vector of probabilities indicating the chance of the image for belonging to certain category, but has no information about the position of regions of interest. These problems have easily been fixed by a simple technique behind the convolution operation: a fully connected layer is equivalent to a corresponding convolutional layer, and thus can be

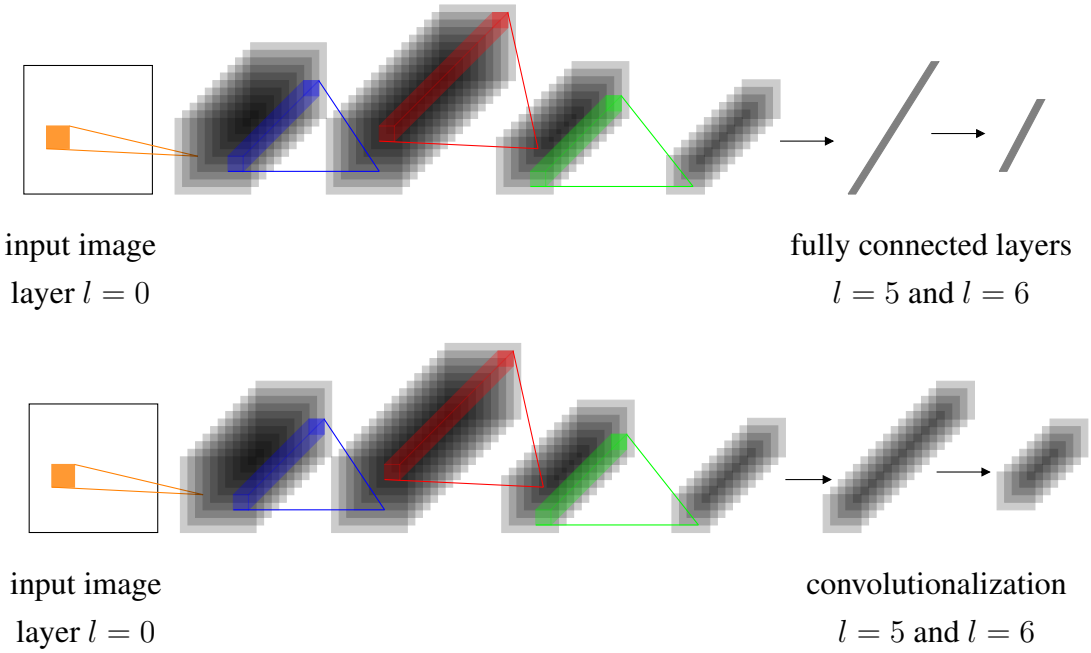


Figure 2.2: An illustration of the comparison between a simple convolutional network (top) and a fully convolutionalized network (bottom) with a single-channel image as input. The last two fully connected layers are replaced by convolutional ones with equivalent operations. The input image for the convolutionalized network is larger for showing the effect of convolutionalized layers, expanding the spatial dimensions and changing the vectors to feature maps. This convolutionalization makes it possible to allow an image in arbitrary dimensions as input for the network.

totally replaced by the convolutional layers, as shown in Figure 2.2.

Convolutionalizing the fully connected layers is, in fact, computing the matrix multiplication in high dimensions. When the image size does not change, there are two situations for the convolutionalization: (1) Between two vectors, the fully connected layer is equivalent to the matrix-vector multiplication. In this situation, the size of parameters matches the size of the input and output vectors. Then for the convolutionalization, we just need to expand the 2D parameter matrix to the 4D convolutional kernel with the kernel size being 1×1, and the 1D activation vector to the 3D feature map with the spatial dimension being 1×1 as the same image size is considered, without changing the effective size of parameters and activations. (2) Between a feature map and a vector, the fully connected layer needs the feature map to be vectorized before doing the matrix-vector multiplication. In this situation, the size of original

parameters matches the size of the vectorized feature map and the output vector. For the convolutionalization, we just need to reshape the 2D parameters to the 4D convolutional kernel with the kernel size being the size of the original feature map, and expand the 1D activation vector to the 3D feature map with the spatial dimension being 1×1 as the same image size is considered, without changing the effective size of parameters and activations. The size of the 2D parameters is equal to the size of the 4D convolutional kernel because it matches the size of the vectorized feature map and the output vector. The convolutionalization is illustrated in Figure 2.3. After the convolutionalization, when larger images come into the network, the output will have larger spatial dimensions.

Although it is an inherent property of convolution, it was firstly applied recently on object detection (classification and bounding-box localization) by Sermanet et al. [2014]. They combined this convolutionalization technique with multi-scale predictions and obtained a good performance on both object classification and localization. However, in the work of Sermanet et al. [2014], the convolutionalization was only used in the inference step; when training, the images were cropped to a fixed dimension and the network was trained to output a classifier for predicting the category of the cropped image and a regression for localizing the bounding box. Briefly speaking, the convolutionalization helps for some object detection tasks, and yet it helps a lot for image segmentation, which does not need to predict the coordinates of bounding boxes because its output corresponds the positions of image pixels. Long et al. [2015a] proposed to train a fully convolutional VGG network end-to-end and pixels-to-pixels for image segmentation, applied bilinear upsampling and skip connections from intermediate layers, and achieved state-of-the-art performance.

2.1.5 Structure Modifications for Image Segmentation

Based on fully convolutional networks, many modifications have been proposed to improve the performance. One problem in the AlexNet and VGG nets is that operations like large stridden convolutions or poolings, reduce significantly the spatial resolution and lead to a fuzzy prediction at local pixels. We introduce three approaches that address this problem.

Fully Convolutional Networks Long et al. [2015a] coped with this problem by restoring the spatial resolution via bilinear upsampling and shortcut from intermediate

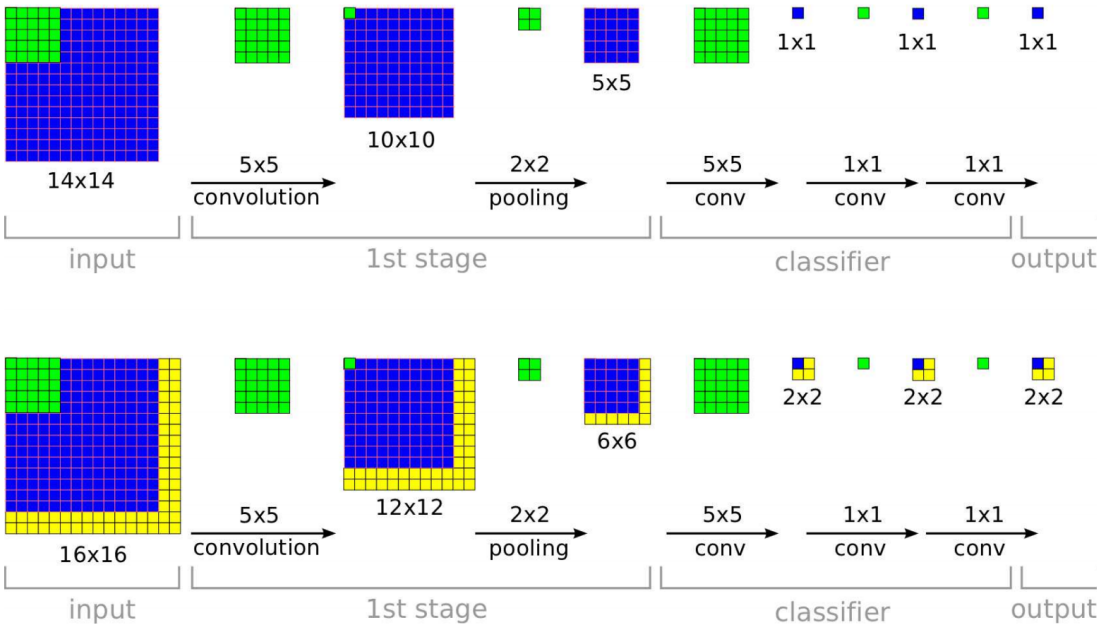


Figure 2.3: Illustration of a fully convolutionalized network. A simple convolutional network with two fully connected layers at the end is fully convolutionalized. This diagram omits the feature dimension for simplicity. A 14×14 image (top) or a 16×16 image is used as input. The first stage reduces the 14×14 image to a 5×5 feature map, and a 5×5 convolutional kernel squeezes all the spatial dimension of the 5×5 feature map. Then the classifier layers follow. However, when the image size is 16×16, the feature map after the first stage is 6×6 and the 5×5 convolutional kernel strides on the feature map, getting a 2×2 feature map. Those in yellow in the bottom sub-figure are the additional compared with the top sub-figure. Then the classifier layers of 1×1 convolutional kernel still cannot squeeze the spatial dimension, and output a 2×2 prediction. The copyright of this figure belongs to Sermanet et al. [2014].

layers that have more detailed position information, see Figure 2.4. ResNet can also be used for image segmentation by removing the global average pooling layer and performs better than VGG nets, but it still suffers from that problem[1].

DeepLab Chen et al. [2018a] proposed that the spatial resolution can be retained by applying *dilated convolution* (or convolution à trous) on some convolutional layers. The dilated convolution multiplies the convolution kernel with dilated elements in the feature maps and explicitly enables the large resolution for computing the features. This operation was originally developed for the efficient computation of the wavelet transform [Holschneider et al., 1990], and in the context of convolutional networks, it was also used in Papandreou et al. [2015]. The dilated convolution is effective for image segmentation, and since Chen et al. [2018a], it is used in every convolutional network for image segmentation. Furthermore, inspired by [He et al., 2015] from object detection, Chen et al. [2018a] also introduced an *atrous spatial pyramid pooling* (ASPP) module that is a concatenation of multiple parallel dilated convolutional layers with different dilatation rates. This is an explicit increase in spatial resolution that successfully improves the detection of small objects without losing the precision on large ones. The ensemble of these approaches used in Chen et al. [2018a] is referred as *DeepLab*. Later Chen et al. [2017, 2018b] improved the DeepLab by absorbing other advantageous components and obtained better results on different databases.

PSPNet Instead of using ASPP, Zhao et al. [2017] proposed *PSPNet*, which is endowed with another pooling pyramid module that gives information of different scales. Information at each scale comes from a branch, which reduces the same input feature map to different scales, extracts features with new convolutional layers (and batch normalization layers), and finally re-upsamples the obtained new features to the original spatial dimensions. All branches and the input feature maps are then concatenated for the next layer. This pooling pyramid module combines the global information with different level local information, and yields a more precise prediction.

[1]The "theoretical" wide receptive field is an open question as mentioned before. It causes a contradiction here: the wide receptive field of ResNet is able to cover the entire input image, and expected to capture enough global information. However, when predicting the spatial information, it turns out that Resnet does not have enough global information, because of large gains from the approaches that explicitly extract global information.

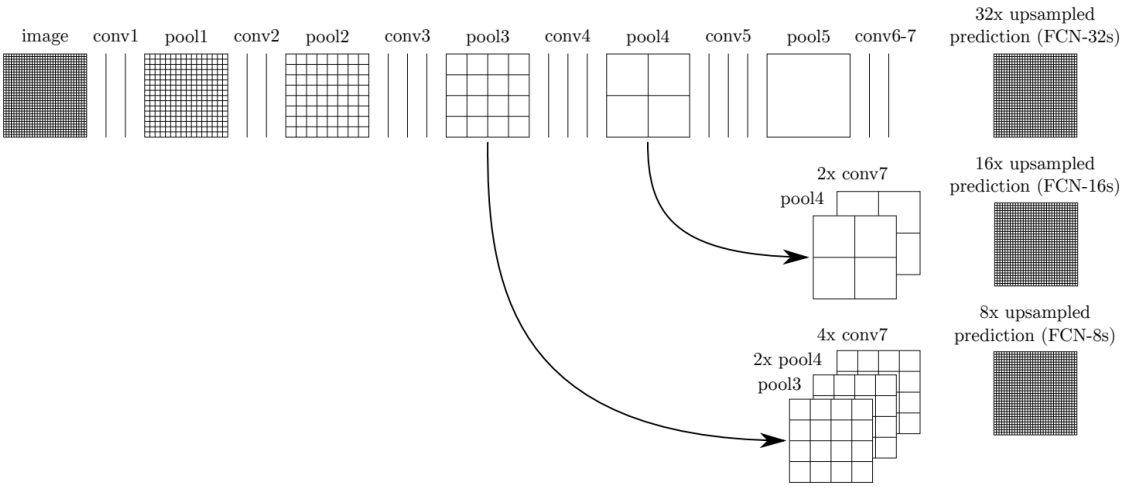


Figure 2.4: Illustration of a fully convolutional network proposed by Long et al. [2015a]. Input, pooled and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. First row is the single-stream network that upsamples the final predictions, which are downsampled to 1/32 by 5 pooling layers, back to pixels. Second row (FCN-16s) combines predictions from both the final layer and the pool4 layer that is after 4 pooling layers, for predicting finer details and retaining high-level semantic information. Third row repeats the similar operation with additional predictions from pool3 and provides further precision. The copyright of this figure belongs to Long et al. [2015a].

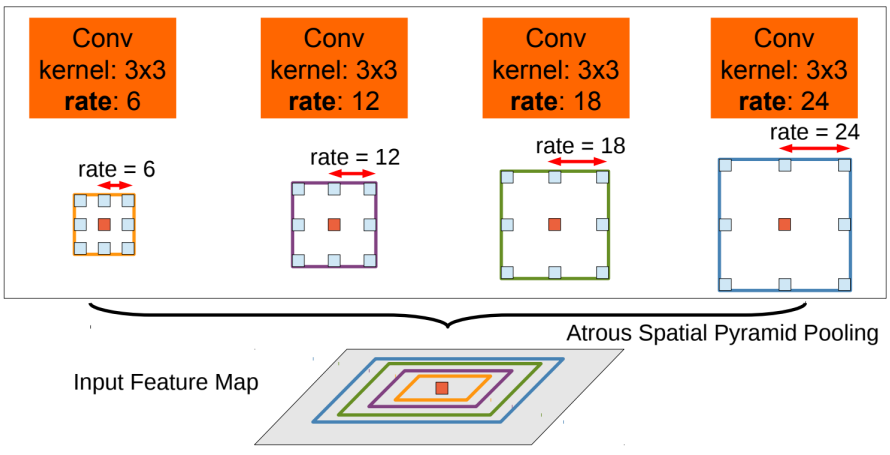


Figure 2.5: Illustration of the atrous spatial pyramid pooling (ASPP) module proposed by Chen et al. [2018a]. To classify the center pixel (orange), ASPP exploits multi-scale features by employing multiple parallel dilated convolution with different rates. The copyright of this figure belongs to Chen et al. [2018a].

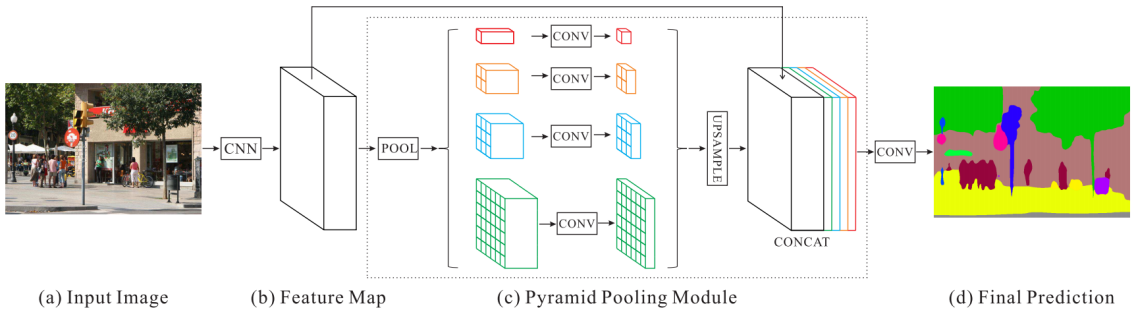


Figure 2.6: Illustration of the pooling pyramid module proposed by Zhao et al. [2017]. Given an input image (a), a convolutional network extracts features(b). Then a pyramid parsing module is applied to harvest different sub-region representations by different pooling layers, followed by convolution and batch normalization layers. After that, each branch is upsampled to the original scale and concatenated to form the final feature representations, which carry both local and global context information in (c). Finally, the representations are fed into a convolution layer to get the final per-pixel prediction (d). The copyright of this figure belongs to Zhao et al. [2017].

Other Networks Regarding image segmentation, there are other interesting networks which should be mentioned, *e.g.*, U-Net [Ronneberger et al., 2015], DeconvNet [Noh et al., 2015] and SegNet [Badrinarayanan et al., 2017], to name a few. They resolve the image segmentation problem under a different scheme where several *deconvolutional* layers (similar to upsampling, but the coefficients are trainable parameters) are after the convolutional layers, which is also called a encoder-decoder network.

2.1.6 Structure Modifications for Other Vision Tasks

Besides image segmentation, many computer vision tasks benefit from convolutional networks. Here, we briefly recall the recent advances in object detection and optical flow estimation, under the notion of deep learning. Experiments in object detection and optical flow estimation are conducted in this thesis for evaluating the proposed regularization approaches.

Object Detection In respect to *object detection*, before fully convolutionalizing the network [Sermanet et al., 2014; Long et al., 2015a], Girshick et al. [2014] already

offered a solution for object detection, which is based on region proposal methods for proposing category-independent bounding boxes, then each of the proposed regions, which is already localized, then given to a pre-trained convolutional network, and classified to a category if it contains an object. The property of convolution saves repeated computations for each region [He et al., 2015; Girshick, 2015] because the feature plan in convolutional networks matches the image pixel plan. Furthermore, a *region proposal network* (RPN) tackles the region proposal task with a neural network and is integrated into one single convolutional network [Ren et al., 2015]. However, a bounding-box localization is not satisfactory to He et al. [2017] and they proposed to predict each pixel in each candidate region instead of the coordinates of the bounding box. In parallel to the R-CNN family, Redmon et al. [2016]; Redmon and Farhadi [2017, 2018] proposed an extremely fast structure, which have equal performance to other approaches. They excluded the region proposal methods and designed a unified elaborate network through pre-defining several anchor boxes for object detection. Liu et al. [2016] shared similar idea and expanded to use multi-scale features to do the detection.

Optical Flow Estimation As for *optical flow estimation*, Weinzaepfel et al. [2013] stacked multi-layer feature maps from a convolutional network and built a matching algorithm. Dosovitskiy et al. [2015] created a synthetic database for optical flow estimation and trained a convolutional network to estimate optical flow directly. Then Ilg et al. [2017] improved the structure and greatly reduced the estimation error.

More Applications in Computer Vision Convolutional networks can also be trained to estimate the stereo matching cost [Zbontar et al., 2016] for computing the disparity map or depth [Zagoruyko and Komodakis, 2015; Luo et al., 2016]. More applications with convolutional networks follow: face recognition [Lawrence et al., 1997], simultaneous localization and mapping (SLAM) [McCormac et al., 2017], detection in point clouds [Qi et al., 2017], human action recognition [Ji et al., 2013], etc. We limit our references in computer vision tasks, without forgetting that convolutional networks are applicable in natural language processing and speech recognition.

2.2 Transfer Learning

Transfer learning is biologically motivated by the way that humans apply learned knowledge to solve new problems, and consists in exploiting knowledge learned in one problem and searching a good protocol of transferring to a new problem. We follow the nomenclature of transfer learning from Pan and Yang [2010]: 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* as the *source task*, whereas the new learning problem is defined on the *target domain* as the *target task*.

Pan and Yang [2010] categorized transfer learning under three sub-settings, *unsupervised transductive*, and *inductive*. In unsupervised transfer learning, data are not labeled in both source and target problems. The transductive transfer learning in Pan and Yang [2010], emphasizes only that the source and target tasks are the same, not strictly limited under the traditional transductive learning, where the learned model depends on all seen data (including test data) and is not applicable for new data. Neither the unsupervised nor the transductive transfer learning is considered in this thesis, as we need source labels for a good pre-trained model and we have no access to the test data during training. So we particularly consider the last transfer learning setting. The inductive transfer learning learns a predictive model or general principals from specific observations, and infers the decision for the target problem. In this setting, the source and target tasks are different but related. In practice, in inductive transfer learning problems, a parametric model is trained in the source problem and transferred to the target problem in a special way, like transferring parameters, or considering the relations between problems.

According to domain and task settings during the transfer, Pan and Yang [2010] introduced several types of transfer learning problems, and in this section, we discus about a few types that matter in this thesis, like domain adaptation and multi-task learning. We also refer to works on new specific problems that were formalized or popularized after Pan and Yang [2010], such as lifelong learning, but their typology remains valid. Finally, we investigate the inductive transfer learning, with convolutional networks as the predictive model.

2.2.1 Domain Adaptation

Despite that domain adaptation is a branch of transductive transfer learning, many approaches for solving the domain adaptation problems are inspiring or directly applicable for solving other transfer learning problems. In domain adaptation, the target domain differs from the source domain whereas the target task is identical to the source task and no (or few) target examples are labeled. The difference between distributions of data in domains is called the *domain shift*, sometimes referred to *covariate shift*. For instance, a spam-filtering model has been learned from a lot of collected emails (as the source domain). When the learned model comes to a new user who receives significantly different emails (as the target domain), the model needs to be adapted because of the domain shift between the source and target domains. So most approaches in domain adaptation are searching for a common feature space for source and target domains to reduce domain shift.

Before deep networks, Pan et al. [2011] reduced the domain shift by minimizing the *maximum mean discrepancy* (MMD) between the empirical means of two domains, after features passing through a mapping function in the *reproducing kernel Hilbert space* (RKHS). Courty et al. [2017] proposed to solved the domain adaptation problems by optimal transport. In the context of convolutional networks, Long et al. [2015b] utilized multi-kernel MMD to regularize the feature maps in the fully connected layers between domains. Meanwhile, Tzeng et al. [2015] proposed a domain confusion loss, trying to learn domain invariant features that a domain classifier is confused and not able to recognize which domain those features come from. Another contribution of Tzeng et al. [2015] is that they also gave "soft" labels [Hinton et al., 2015], computed from source labeled samples, to target unlabeled data as to consider an inter-category relationship. Rozantsev et al. [2019] integrated the MMD, domain confusion and a parameter regularizer into a single framework to reduce the domain shift.

In many databases, a test set is prepared, which has roughly the same distribution as the training set, but in reality, their distributions can be very different. A good model from manually created databases usually performs poorly in a different environment. Domain adaptation becomes more and more attractive because it can improve the performance on a set of unlabeled examples that are different from the source data.

2.2.2 Multi-Task Learning

Multi-task learning is a branch of the inductive transfer learning, and it focuses on the performance of all tasks. In multi-task learning, multiple tasks are learned jointly by a single model. Each task can improve the performance of others by using the knowledge on the problem as a regularization term. All tasks share the common representations and what is learned for each task can help other tasks to be better solved [Caruana, 1997]. While the intuition seems sound, in practice, it is not always a free lunch.

Multi-task learning with neural networks exists since Baxter [1997] which shares low dimensional representations (the first layers) and separately outputs a branch for each task. This kind of structure is still being used nowadays for contemporary convolutional networks, and much advanced. For instance, Yang and Hospedales [2017] utilized a tensor factorization approach to recognize parameters that can be shared and ones that are task-dependent. Misra et al. [2016] proposed a cross-stitch unit that is a linear combination between shared feature maps for one pair of tasks, where the linear weights are manually controlled by the similarity between each pair of tasks. Kendall et al. [2018] modeled each output as a Gaussian distribution and learned to weight the tasks by minimizing the log likelihood. Similarly, Chen et al. [2018d] learned the rates of tasks by constraining the norm of gradients with respect to each task to be close.

Multi-task learning with convolutional networks in computer vision is widely applied, like surface normal maps and depth estimation [Qi et al., 2018], optical flow estimation and video segmentation [Cheng et al., 2017], or everything [Sharif Razavian et al., 2014]. This is because many related tasks can be solved simultaneously, and the representations are transferable among tasks.

2.2.3 Lifelong Learning (Continual Learning)

Lifelong learning [Thrun and Mitchell, 1995] or continual learning copes with a sequential set of tasks usually by a single model. Lifelong learning is crucial for computational systems and autonomous agents interacting in the real world and processing continuous streams of information, and contributes a lot towards the strong artificial intelligence. However, along the lifelong learning, the knowledge extracted from the previous tasks can be easily lost as new tasks are learned, resulting in what is known as *catastrophic forgetting*.

Pentina and Lampert [2015] transformed a multi-task problem to a lifelong learning problem. They argued that learning sequentially multiple tasks can be more effective than learning them jointly. Their solution is to establish an order of tasks to be learned and apply the adaptive SVM (A-SVM) [Yang et al., 2007] to each task with the parameters in the previous task as a reference. Rusu et al. [2016] stored one model for each learned task and transferred the knowledge to the new task through the connections with previous models. Li and Hoiem [2017] proposed to use the outputs of the target examples, computed by the original network on the source task, to define a distillation learning scheme [Hinton et al., 2015] preserving the memory of the source tasks when training on the target task. Jung et al. [2018] regularized the representations in the penultimate layer to be similar to the ones computed by the source network. Different from keeping the similar representations, Kirkpatrick et al. [2017] search solutions with similar parameters for preserving the knowledge and solving the target task jointly. They get sensible improvements by measuring the sensitivity of the parameters of the network learned on the source data thanks to the Fisher information. The Fisher information matrix defines a metric in parameter space that is used in their regularizer to preserve the representations learned on the source data, thereby retaining the knowledge acquired on the previous tasks.

Lifelong learning is very attractive for designing an autonomous agent dealing with many (new) tasks. The key to lifelong learning is to maximally preserve the knowledge that was already learned to solve all the inductive transfer learning problems.

2.2.4 Inductive Transfer Learning with CNN

Convolutional networks are broadly applicable in the fields mentioned before, and they are even more attractive in the *inductive transfer learning* setting, where the target domain is identical to the source domain, and the target task is different from the source task. All the applications introduced in 2.1.5 and 2.1.6 can be framed in the field of inductive transfer learning.

From the view of feature extractors, convolutional networks are extremely powerful compared to other models. Donahue et al. [2014] selected the features computed at different layers of the pre-trained AlexNet [Krizhevsky et al., 2012] and plugged them into an SVM or a logistic regression classifier for learning a new task. This approach outperformed the state of the art at that time on the Caltech-101 database [Fei-Fei

et al., 2006]. Similar conclusions were found by Oquab et al. [2014]; Sharif Razavian et al. [2014]. Later, Yosinski et al. [2014] showed that fine-tuning the whole AlexNet resulted in better performances than using the network as a static feature extractor. Furthermore, fine-tuning pre-trained VGG [Simonyan and Zisserman, 2015] on the image classification task of VOC-2012 [Everingham et al., 2010] and Caltech 256 [Griffin et al., 2007] achieved the best results at that time. Since that, fine-tuning becomes a very practical way of benefitting from knowledge learned on a large database, and a useful technique for inductive transfer learning tasks.

After that, Ge and Yu [2017] proposed a scheme for selecting a subset of images from the source problem that have similar local features to those in the target problem and then fine-tuned a pre-trained convolutional network for image classification tasks. Similarly, Cui et al. [2018] selected an optimal source domain for a target domain by computing the similarity between each source class and each target class and performed the transfer. In fact, all the applications introduced in 2.1.5 and 2.1.6 that tackled complex computer vision tasks, also benefited a lot from fine-tuning and transfer learning. All these approaches showed promising results in a challenging transfer learning setup, as going from classification to object detection or image segmentation requires rather heavy modifications of the architecture of the network.

The success of transfer learning with convolutional networks relies on the generality of the learned representations that have been constructed from a large database like ImageNet. Yosinski et al. [2014] quantified the transferability of these pieces of information in different layers, e.g. the first layers learn general features, the middle layers learn high-level semantic features and the last layers learn the features that are very specific to a particular task. Zeiler and Fergus [2014] also visualized the features in the intermediate layers, demonstrating, with images, that convolutional networks learn features from general level to task-specific level. Overall, the learned representations can be conveyed to related but different domains and the parameters in the network are reusable for different tasks.

2.3 Optimal Transport

The *optimal transport* (OT) theory is a mathematical tool of estimating the distance between probability distributions with consideration of the geometric structure of the support space on which the distributions are defined. We would like to benefit from the OT theory to access the distance between the distributions from source and target

domains and minimize it in order to preserve the source knowledge.

The optimal transport problem was firstly described by the French mathematician Gaspard Monge [Monge, 1781] as an assignment problem, and later relaxed to a probabilistic transport by Kantorovich [1942]. The problem was inspired by a minimization task about a worker transporting a pile of sand to another specific-shape pile with minimal efforts. For instance, the sand was a shape of cone and the worker would like to reshape it to a castle. The efforts are defined as the total distance of moving each small heap of sand from original place to the target position, and recall that the task is to minimize the effort, or equivalently, to find the optimal target position for each heap of sand. The OT problem is interesting because it can be cast as a measure between distributions, either or both discrete and continuous. We can, of course, compare distributions using alternative measures, like classic Kullback-Leibler divergence, Jensen-Shannon divergence, Hellinger distance or others, but the OT problem entails the geometric structure of the support space on which the distributions are defined. That is the cost of transporting the mass, which differs the optimal transport from other measures.

Many great books introduce mathematical and algorithmic foundations of the optimal transport theory, like Villani [2008]; Santambrogio [2015]; Peyré and Cuturi [2018]. We refer readers to these books for detailed introduction of the OT theory. Here we make a brief reminder on the OT problem and its efficient solutions, and discuss the applications in machine/deep learning.

2.3.1 Mathematical Definition

Let $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ be two discrete probability distributions: $\boldsymbol{\mu} = \sum_{i=1}^{n} \mu_i \delta_{\boldsymbol{x}_i}$ and $\boldsymbol{\nu} = \sum_{i=1}^{m} \nu_i \delta_{\boldsymbol{y}_i}$, where $\sum_{i=1}^{n} \mu_i = \sum_{i=1}^{m} \nu_i = 1$, and $\delta_{\boldsymbol{x}}$ is the Dirac delta function at position $\boldsymbol{x}$. Then with a defined cost function d and the cost matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$, where $\mathbf{M}_{ij} = d(\boldsymbol{x}_i, \boldsymbol{y}_j)$, we can give the optimal transport cost in the Kantorovich-relaxed OT problem:

$$L_{\mathbf{M}}(\boldsymbol{\mu}, \boldsymbol{\nu}) = \min_{\mathbf{P} \in U(\boldsymbol{\mu}, \boldsymbol{\nu})} \langle \mathbf{P}, \mathbf{M} \rangle_F, \tag{2.4}$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product, $U(\boldsymbol{\mu}, \boldsymbol{\nu})$ is the set of all possible joint distributions of $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$, equivalent to $U(\boldsymbol{\mu}, \boldsymbol{\nu}) = \{\mathbf{P} \in \mathbb{R}_{+}^{n \times m} \mid \mathbf{P}\mathbf{1}_m = \boldsymbol{\mu}, \mathbf{P}^T \mathbf{1}_n = \boldsymbol{\nu}\}$. The optimal joint distribution that minimizes the transport cost is the optimal

transport plan $\mathbf{P}_0$:

$$\mathbf{P}_0 = \underset{\mathbf{P} \in U(\boldsymbol{\mu}, \boldsymbol{\nu})}{\operatorname{argmin}} \langle \mathbf{P}, \mathbf{M} \rangle_F. \tag{2.5}$$

The Equation 2.4 is thus equal to

$$L_{\mathbf{M}}(\boldsymbol{\mu}, \boldsymbol{\nu}) = \langle \mathbf{P}_0, \mathbf{M} \rangle_F. \tag{2.6}$$

This optimum is a distance between $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ if $\mathbf{M}$ is a metric matrix, namely d is a distance between $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ [Villani, 2008, Chapter 6].

We reconsider the initial problem of transporting sand within the mathematical definition. The cost function d can be simply the Euclidean distance between two positions at a 2D plan, as the effort of moving a small heap of sand is proportional to the distance. Then in this 2D plan, a source distribution of sand is given, the worker would like to change to a target distribution with minimal effort. Although the transport happens in the same space, this problem does not lose the generality. The cost d can be zero when the sand does not move, and it can be large if the worker moves some sand from a corner to another. So a minimal transport plan should be decided in advance, and that is $\mathbf{P}_0$. Each row of $\mathbf{P}_0$ tells the worker how much sand to move from each source position to each target position. Each column of $\mathbf{P}_0$ shows how much sand each target position is received from each source position. Following the optimal transport plan $\mathbf{P}_0$, the effort is minimal.

2.3.2 Entropic Solvers

Linear programming is a solver for the OT problem because of the linear objective function and linear constraints, however, its computational budget increases in a cubic rate or more [Pele and Werman, 2009] with the n or m increasing. It is difficult to make it applicable in practice with a large model or a large dataset.

Recently, Cuturi [2013] proposed to search for an approximate solution to the OT problem in a set of entropic-constrained joint distributions:

$$L^{\alpha}_{\mathbf{M}}(\boldsymbol{\mu}, \boldsymbol{\nu}) = \min_{\mathbf{P} \in U_{\alpha}(\boldsymbol{\mu}, \boldsymbol{\nu})} \langle \mathbf{P}, \mathbf{M} \rangle_F, \tag{2.7}$$

where $U_{\alpha}(\boldsymbol{\mu}, \boldsymbol{\nu}) := \{\mathbf{P} \in U(\boldsymbol{\mu}, \boldsymbol{\nu}) \mid h(\boldsymbol{\mu}) + h(\boldsymbol{\nu}) - h(\mathbf{P}) \leq \alpha)\} \subset U(\boldsymbol{\mu}, \boldsymbol{\nu})$, and h is the entropy, specifically,

$$h(\boldsymbol{\mu}) = -\sum_i \mu_i \log \mu_i, \quad h(\boldsymbol{\nu}) = -\sum_i \nu_i \log \nu_i, \quad h(\mathbf{P}) = -\sum_{i,j} p_{ij} \log p_{ij}, \tag{2.8}$$

noting that both $h(\boldsymbol{\mu})$ and $h(\boldsymbol{\nu})$ are constant. In fact, $h(\boldsymbol{\mu}) + h(\boldsymbol{\nu}) - h(\mathbf{P})$ is the Kullback-Leibler divergence between $\mathbf{P}$ and $\boldsymbol{\mu}\boldsymbol{\nu}^T$, remembering that $KL(\boldsymbol{a}, \boldsymbol{b}) = \sum_i a_i \log \frac{a_i}{b_i}$. In order to connect the transport plan with $U_\alpha(\boldsymbol{\mu}, \boldsymbol{\nu})$, Cuturi [2013] proposed to compute

$$\mathbf{P}_0^\lambda = \underset{\mathbf{P} \in U(\boldsymbol{\mu}, \boldsymbol{\nu})}{\operatorname{argmin}} \langle \mathbf{P}, \mathbf{M} \rangle_F - \frac{1}{\lambda} h(\mathbf{P}), \tag{2.9}$$

where a $\lambda \in (0, +\infty)$ can be always found for each α in $L_{\mathbf{M}}^\alpha(\boldsymbol{\mu}, \boldsymbol{\nu})$, such that $L_{\mathbf{M}}^\alpha(\boldsymbol{\mu}, \boldsymbol{\nu}) = \langle \mathbf{P}_0^\lambda, \mathbf{M} \rangle_F$.

The rest is simple. Solving Equation 2.9 needs nothing but applying the method of Lagrange multipliers and the Sinkhorn-Knopp algorithm. With the Lagrangian, we cannot get the closed-form solution of $\mathbf{P}_0^\lambda$ directly, but we can obtain that the solution has a special form $\operatorname{diag}(\boldsymbol{a})\mathbf{K}\operatorname{diag}(\boldsymbol{b})$, where $\mathbf{K} = e^{-\lambda \mathbf{M}}$ is the element-wise exponential of $-\lambda\mathbf{M}$, $\operatorname{diag}(\boldsymbol{a})$ is a diagonal matrix with the elements of the vector $\boldsymbol{a}$ as its diagonal, and $\boldsymbol{a}$ and $\boldsymbol{b}$ depend on the two Lagrangian multipliers respectively. Then, Sinkhorn's theorem [Sinkhorn, 1964] states that for a strictly positive matrix $\mathbf{K}$, there exists unique diagonal matrices $\mathbf{D}_1$ and $\mathbf{D}_2$ with strictly positive diagonal elements, such that $\mathbf{D}_1\mathbf{K}\mathbf{D}_2$ is a doubly stochastic matrix. Thus the solution $\mathbf{P}_0^\lambda$ exists and is unique. Furthermore, a simple iterative method Sinkhorn-Knopp algorithm can converge to the solution, by alternately rescaling rows and columns of $\mathbf{K}$ to sum to 1.

The Sinkhorn iterations are sometimes numerically unstable, and our experiments often encounter that instability, but it can be relieved by the proximal point algorithm with any Bregman divergence, suffering from slightly more computation cost, see [Peyré and Cuturi, 2018, Remark 4.9] and Xie et al. [2018].

2.3.3 Optimal Transport Applications in Deep Learning

In terms of deep networks, the optimal transport is quite appealing for many applications, especially for training generative models [Arjovsky et al., 2017; Gulrajani et al., 2017]. In the following, We discuss briefly about the generative models with optimal transport, and some other applications of the optimal transport.

The stochastic process of generating natural images or language texts is difficult to accurately describe. Goodfellow et al. [2014] proposed the *generative adversarial networks* (GANs) to appaximate this process with deep networks. The GAN structure composes two networks: one generator for generating the examples, one discriminator for distinguishing generated examples from natural ones. The loss function in Goodfellow et al. [2014] is equivalent to a Jensen-Shannon divergence between the under-

Algorithm 3 The Sinkhorn-Knopp algorithm

Input two distributions $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ with length n and m respectively, cost matrix $\mathbf{M}$ with shape (n, m), regularizer term scalar λ

$\mathbf{K} = e^{-\lambda \mathbf{M}}$: element-wise exponentiation

$\boldsymbol{a} = \mathbf{1}_n/n, \boldsymbol{b} = \mathbf{1}_m/m$: initialization of the two scaling diagonal matrices in vector form

while $\boldsymbol{a}$ changes or not exceed the preset maximum iteration step **do**

$\boldsymbol{a} = \frac{\boldsymbol{\mu}}{\mathbf{K}\boldsymbol{b}}$: update $\boldsymbol{a}$, element-wise division, scaling the sum of cols to $\boldsymbol{\mu}$

$\boldsymbol{b} = \frac{\boldsymbol{\nu}}{\mathbf{K}^T\boldsymbol{a}}$: update $\boldsymbol{b}$, element-wise division, scaling the sum of cols to $\boldsymbol{\nu}$

end while

$\mathbf{P}_0^\lambda = \text{diag}(\boldsymbol{a})\mathbf{K}\text{diag}(\boldsymbol{b})$: $\boldsymbol{a}$ is converged as well as $\boldsymbol{b}$ and $\mathbf{P}_0^\lambda$

output optimal transport plan $\mathbf{P}_0^\lambda$ with shape (n, m)

lying data distribution and the model distribution. Arjovsky et al. [2017] showed that the optimal transport costs might have nicer properties over the Jensen-Shannon divergence when learning distributions supported by low dimensional manifolds, since the OT costs leverage the geometry of the underlying space when measuring the difference between distributions. With a better measurement between distributions, the data distribution can be easier to approximate by GANs. Thus minimizing the distance between distributions is the key of GANs, which matches the objective of the OT problem.

Arjovsky et al. [2017] minimize an upper bound of the optimal transport cost, *i.e.* the Kantorovich-Rubinstein duality:

$$L_{\mathbf{M}}(\boldsymbol{\mu}, \boldsymbol{\nu}) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{\boldsymbol{x}\sim\boldsymbol{\mu}}[f(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim\boldsymbol{\nu}}[f(\boldsymbol{x})], \tag{2.10}$$

where the supremum is over all the 1-Lipschitz functions $f\colon \mathcal{X} \to \mathbb{R}$, and f works for the discriminator in GANs. The constraints of 1-Lipschitz functions can be done by some clipping tricks on parameters [Arjovsky et al., 2017] or gradients [Gulrajani et al., 2017]. GANs with the optimal transport cost stabilized the quality of generated examples because of the meaningful cost function.

In addtion, instead of reshaping the OT problem to the duality form, Bousquet et al. [2017]; Genevay et al. [2018]; Chen et al. [2018c]; Salimans et al. [2018] directly exploit the smoothed OT cost for training GANs.

Beyond GANs, Courty et al. [2017] transformed the domain adaptation problem to an optimal transport one, and solved it by adding a group-sparsity term on the

transport plan with the Sinkhorn algorithm. The OT theory is also helpful in tag prediction [Frogner et al., 2015], comparing documents [Kusner et al., 2015; Huang et al., 2016], dictionary learning [Rolet et al., 2016] *etc*.

2.3.4 Optimal Transport on Neuron Distributions

In this thesis, we focus on transfer learning with deep networks and propose to preserve the *neuron distribution* during the transfer, relying on the OT theory. We explain the motivations in Chapter 4, but here we would like to clarify the neuron distribution and its link with the optimal transport problem.

We consider that neurons at some layer of the neural network are samples drawn from a conditional distribution given the neurons at the previous layer, and the parameters at the current layer are responsible for generating the samples, *i.e.* the neurons at this layer. This conditional distribution is the neuron distribution that we would like to preserve during transfer learning.

If we split a neural network into two parts from a certain layer, and then consider the neurons at that layer, any permutation among these neurons will not change the representation capacity of the network, even linear transformations on the neurons will not either, provided a corresponding adjustment of the parameters at the next layer. During transfer learning, considering the neurons at the penultimate layer, *i.e.* the output of the feature extractor, the permutations and transformations will probably happen during fine-tuning or other transfer learning approaches. While these transformations are not detrimental for the learning, the approach of preserving the neuron distribution during transfer learning should recognize these transformations and not punish them. The optimal transport metric is able to do that. We detail this idea in Chapter 4.

2.4 Regularization Approaches

In deep learning, training a deep network is difficult [Glorot and Bengio, 2010] and prone to overfitting. A *regularization approach*, or *a regularizer*, is a common solution that can lessen the chance or amount of overfitting for a trainable model and reduce the generalization error of the learned model. A deep network is highly capable of memorizing all the training data if no regularizers are applied. In the following, we will introduce some common regularizers that are classified into three categories

according to the effect. Note that detailed descriptions of these regularizers and others can be found in the book of deep learning [Goodfellow et al., 2017, Chapter 7]. After that, we will discuss how a regularizer implements an inductive bias for a desirable property in the learned model.

2.4.1 Regularizers Bringing Desirable Properties

A very common and classic regularizer is based on the trainable parameters of the model. We usually gather all the parameters in the model as a parameter vector $\boldsymbol{w}$. The L^2 *parameter regularizer* restrains the L^2 norm of $\boldsymbol{w}$ to a small value, in order to force the model to learn useful representations instead of fitting the input-output mappings. The gradients of the L^2 regularizer *w.r.t.* parameters are a smaller-than-one positive scale of current values of parameters, so the L^2 regularizer is also named as *weight decay*. In fact, any norm of parameters can be used as parameter regularizer, with different objectives, but only a few are mostly used, *e.g.* L^2 constrains the parameters in a ball of the parameter space, L^1 is used for sparsity, and Group-Lasso is for group-sparsity. These parameter regularizers can bring different properties to the learned model, and they are very easy to be integrated into the model, especially when using stochastic gradient descent, where the parameter regularizer is added on the loss function and the back-propagation does the rest.

Early stopping prevents the training process from falling into the overfitting. Early stopping has been proved to be equivalent to the L^2 parameter regularizer, at least under the quadratic approximation of the objective function, see [Goodfellow et al., 2014, Section 7.8] for example.

Dropout is a regularizer on the network structure. Srivastava et al. [2014] proposed to randomly "drop out" some connections between layers during the SGD algorithm. At each step, the architecture is different. Each neuron cannot always work together with others, so the neurons are forced to learn robust and independent features, rather than interdependent features. This kind of efficient technique is very useful for training a large neural network.

Model ensemble methods combine several models and predict the output together. This method works because different models will usually not make the same errors. This is also a property of improving the performance that a model desires, despite some additional computation and training time.

2.4.2 Regularizers Creating Synthetic Training Examples

Although the ideal way to avoid overfitting is to collect more labeled data, it is not always feasible in practice. Data augmentation helps on it. Several data augmentation techniques are like random blur (adding noises on the color space), random mirror (randomly inverse horizontally the image), random crop (choose a part of image randomly as input for the network), and random scale (increase or reduce the original dimension of an image). These synthetic transformations of images are equivalent to increase the number of training examples and can improve the performance for many vision tasks in practice.

2.4.3 Regularizers with Good Randomness

Some randomness during training is also helpful to avoid overfitting and can be regarded as regularizer. Some aforementioned regularizers, like data augmentation techniques and dropout, can simultaneously increase the scale of the dataset and provide good randomness.

Other regularizers with good randomness are, for instance, the normalization operations. *Batch normalization* (BN) [Ioffe and Szegedy, 2015] is an efficient technique in deep learning, normalizing the feature maps with the examples in one mini-batch. The normalization step is standard: within each kernel, each activation subtract the sample mean and then is divided by the sample standard deviation. This step does not only normalize the forward activations at each layer, but also prevent the backward gradients from exploding or vanishing. The motivation of BN is to solve the mysterious "internal covariate shift" inside the network but exceptionally, BN did much more beyond that: it accelerates the training process, makes the parameter initialization less important, and improves the performance. Some papers [Santurkar et al., 2018; Kohler et al., 2019] aim at demonstrating why batch normalization works. Before the batch normalization, the limit of convolutional network is a 22-layer GoogLeNet [Szegedy et al., 2015]. Nowadays, the depth can be more than one thousand [He et al., 2016a,b] and even ten thousand [Xiao et al., 2018]. Despite the fact that BN facilitates the optimization process, BN acts like a regularizer because from the view of one given example, it is always combined with different examples in mini-batches, thus always gives different statistics for normalizing the feature maps during training, so it is never deterministic and acts as a turbulence and a regularizer. Several alternatives of batch normalization, like *layer normalization* [Lei Ba et al., 2016], *instance normalization*

[Ulyanov et al., 2016], *group normalization* [Wu and He, 2018], were proposed recently with the same mechanism of normalization but along different dimensions or subsets of feature maps for different tasks.

2.4.4 Regularizers as Inductive Bias for Better Learning

We have briefly introduced the prevailing regularizers in deep learning, principally with deep neural networks and the stochastic gradient descent algorithm. These regularizers can be grouped into three parts as presented in the previous three subsections. Here we would like to discuss about the link between regularizers and the inductive bias, specifically, how a regularizer brings a desirable property for the trainable model.

Simply speaking, the inductive bias is a set of assumptions that are probably advantageous to reduce the generalization error of the model. A classical example of an inductive bias is Occam's razor, assuming that the simplest consistent hypothesis about the target function is actually the best. An overly complex model family is not a reasonable choice for training on few data. Parameter regularizers assume that the model can learn more useful and pertinent representations with constrained values or with less parameters, instead of fitting the data wildly. Thus parameter regularizers can be seen an inductive bias. Since early stopping is equivalent to the L^2 parameter regularizer, then it is also an inductive bias. Dropout presumes from the biological intuition that a set of independent neurons is better than a set of interdependent neurons. Randomly dropping out some neurons can punish this interdependence and thus improve the robustness of the learned model.

The regularizers we propose in this thesis also follow the inductive bias intuition. In transfer learning, a pre-trained model on a large database like ImageNet is not only a starting point for fine-tuning, but also a reference around which exists with high probability a good solution for the target task. Thereby we propose a family of parameter regularizers using the source knowledge as reference in Chapter 3. In Chapter 4, we also focus on preserving the source knowledge. Instead of working on parameters, we introduce another family of regularizers on representations, and make efforts on encouraging the similarity between neuron distributions in source and target domains through optimal transport. We evaluate the proposed regularizers with various experiments, which are presented in the corresponding chapter.

Chapter 3

Parameter Regularizers for Fine-Tuning

Contents

Prologue 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 task. In this chapter, we investigate several regularization schemes that explicitly promote the similarity of the final solution with the initial model. We show the benefit of having an explicit inductive bias towards the initial model, and we eventually recommend a simple L^2 penalty with the pre-trained model being a reference as the baseline of penalty for transfer learning tasks.

3.1 Introduction

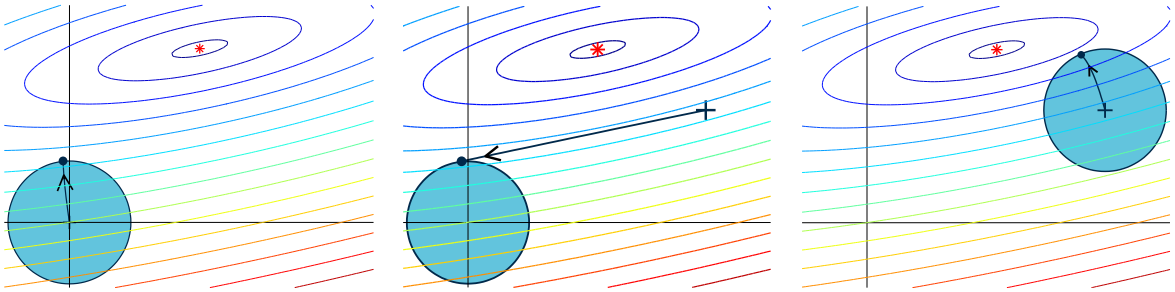
Modern convolutional neural networks are powerful models that can achieve remarkable performance on large-scale image databases, *e.g.* ImageNet [Deng et al., 2009] and Places365 [Zhou et al., 2018], meanwhile, once trained on a large database, they can be refined to solve related but different visual tasks by means of transfer learning, using fine-tuning [Yosinski et al., 2014; Simonyan and Zisserman, 2015], with much less computation time and power consumption than training from scratch.

In transfer learning, some form of knowledge is believed to be extracted by learning from the large-scale database of the source task, and this knowledge is then transferred to the target task by initializing the network with the pre-trained parameters. In inductive transfer learning, particularly fine-tuning, parameter regularizers still constrain the model in the neighborhood of the origin and some parameters may be driven far away from their initial values. However, we will show in the experimental section that some parameters may be driven far away from their initial values during fine-tuning. This leads to important losses of the initial knowledge that is assumed to be relevant for the target problem.

We argue that the standard L^2 parameter regularizer, which drives the parameters towards the origin, is not adequate in the framework of transfer learning, and thereby provides suboptimal results for the target problem. Parameter regularizers, like the standard L^2, are critical and efficient when learning on small databases. When learning from scratch, parameter regularizers are aimed at reducing the generalization error and avoiding overfitting, by implicitly restricting the capacity of the network, that is, the effective size of the search space, implicitly driving the parameters towards the origin. However, we can regularize the parameters towards any value of the parameter space, and better results should be obtained for a value closer to the true one [Goodfellow et al., 2017, Section 7.1.1]. In transfer learning, a reference is given from solving the source task, and hence the network capacity has not to be restricted blindly: We advocate for a coherent parameter regularizer, where the pre-trained model is both used as the starting point of the optimization process and as the reference in the penalty that encodes an explicit inductive bias, so as to help preserve the knowledge embedded in the initial network during fine-tuning. This simple modification keeps the original control of overfitting, by constraining the effective search space around the initial solution, while encouraging committing to the acquired knowledge. We show that it has noticeable effects in transfer learning scenarios. Figure 3.1 provides a didactic illustration for the particular case of the L^2-*SP* parameter regularization in a situation that would for example correspond to linear regression.

The parameter regularizers that encourage similarity with the starting point of the fine-tuning process will be denoted with the *SP* suffix. Despite the existence of several approaches akin to L^2-*SP* in other circumstances as described in the next section, many works (like all state-of-the-art results mentioned in 2.1.5 and 2.1.6) disregard the inconsistency of using L^2 in transfer learning scenarios.

In this chapter, we explore how a coherent explicit inductive bias, encoded by a parameter regularizer, affects the transfer learning process. We consider the inductive transfer learning setting, where the target domain is identical to the source domain, and the target task is different from the source task. We furthermore focus on the case where a vast amount of data was available for training on the source problem, and some limited amount of labeled data is available for solving the target problem. Under this setting, we evaluate -*SP* regularizers based on the L^2, Lasso and Group-Lasso penalties, which can freeze some individual parameters or groups of parameters to the pre-trained values. We also test the L^2-*SP* and *Group-Lasso-SP* variants that use the Fisher information to measure similarity. Our experiments in classification, semantic



(a) (b) (c)

Figure 3.1: Illustration of the inadequacy of the standard L^2 regularization in transfer learning. The plots show the same 2D parameter space in a simple transfer learning situation. The red star represents the minimum of the unregularized risk for the target task; the black cross is the starting point of the optimization process, and the black point represents the result of a gradient-like optimizer, with intermediate solutions represented by the black segment. The ellipses represent the contour levels of the target task, and the large blue circle represents the effective search domain defined by the regularizer (admissible set). The sub-figures correspond three cases: (a) presents the standard learning process with L^2 where there is no transfer learning, (b) is the fine-tuning process with L^2 where the pre-trained model is not beneficial, and (c) shows the case of fine-tuning with L^2-*SP* where the memory of the pre-trained mode is preserved and the optimization is easier.

segmentation and video analyses, using several convolutional network architectures, and additional analyses, indicate that all tested parameter regularization methods using the pre-trained parameters as a reference get an edge over the standard L^2.

We present related work that preserves the knowledge in other transfer learning problems in Section 3.2, and our propositions of parameter regularizers in Section 3.3. Section 3.4 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. Section 3.5 provides some analyses and theoretic insights. More experimental results beyond image segmentation are given in Section 3.6.

3.2 Related Work

The regularization scheme we advocate in this chapter is related to several existing approaches that were proposed to encourage similarity of parameters or representations across different tasks.

In lifelong learning, Li and Hoiem [2017] proposed to use the outputs of the target examples, computed by the original network on the source task, to define a learning scheme retaining the memory of the source tasks 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 interesting results. Kirkpatrick et al. [2017] get sensible improvements by measuring the sensitivity of the parameters of the network learned on the source data thanks to the Fisher information. The Fisher information matrix defines a metric in the parameter space, which is used in their regularizer to preserve the representations learned on the source data, thereby retaining the knowledge acquired on the previous tasks. This scheme, named elastic weight consolidation, was shown to avoid forgetting, but fine-tuning with plain stochastic gradient descent was more effective than elastic weight consolidation for learning new tasks. Hence, elastic weight consolidation may be thought as being inadequate for transfer learning, where performance is only measured on the target task.

In domain adaptation, Rozantsev et al. [2019] proposed a parameter regularization scheme for encouraging the similarity of the representations of the source and the target domains. Their regularizer encourages similar source and target parameters, up to a linear transformation. Still in domain adaptation, besides vision, encouraging similar parameters in deep networks has been proposed in speaker adaptation problems

[Liao, 2013; Ochiai et al., 2014] and neural machine translation [Barone et al., 2017], where it proved to be helpful.

Beyond deep networks, regularization has been a means of building shrinkage estimators for decades. Shrinking towards zero is the most common form of shrinkage, but shrinking towards adaptively chosen targets has been around for some time, starting with Stein shrinkage [Lehmann and Casella, 1998, chapter 5], where it can be related to empirical Bayes arguments. Shrinking towards a reference has also been used in maximum entropy models [Chelba and Acero, 2006] or SVM [Yang et al., 2007; Aytar and Zisserman, 2011; Tommasi et al., 2014]. For example, Yang et al. [2007] proposed an adaptive SVM (A-SVM), which regularizes the squared difference between the parameter vector and an initial parameter vector that is learned from the source database. Then, Aytar and Zisserman [2011] added a linear relaxation to A-SVM and proposed the projective model transfer SVM (PMT-SVM), which regularizes the angle between the parameter vector and the initial one. Experiments in Aytar and Zisserman [2011]; Tommasi et al. [2014] demonstrated that both A-SVM and PMT-SVM were able to outperform standard L^2 regularization with limited labeled data in the target task. These approaches were shown to outperform standard L^2 regularization with limited labeled data in the target task [Aytar and Zisserman, 2011; Tommasi et al., 2014]. They differ from the application to deep networks in several respects, the more important one being that they consider a fixed representation, with which transfer aims at producing similar classification parameters, that is, similar classification rules. For deep networks, transfer aims at learning similar representations upon which classification parameters will be learned from scratch. Hence, even though the techniques we discuss here are very similar regarding the analytical form of the regularizers, they operate on very different objects.

3.3 -*SP* Regularizers

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

L^2 penalty The current 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:

$$\Omega(\boldsymbol{w}) = \frac{\alpha}{2} \|\boldsymbol{w}\|_2^2 \quad , \tag{3.1}$$

where α is the regularization parameter setting the strength of the penalty and $\|\cdot\|_p$ 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:

$$\Omega(\boldsymbol{w}) = \frac{\alpha}{2} \left\|\boldsymbol{w} - \boldsymbol{w}^0\right\|_2^2 \quad . \tag{3.2}$$

Typically, the transfer to a target task requires some modifications of the network architecture used for the source task, 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:

$$\Omega(\boldsymbol{w}) = \frac{\alpha}{2} \left\|\boldsymbol{w}_{\mathcal{S}} - \boldsymbol{w}_{\mathcal{S}}^0\right\|_2^2 + \frac{\beta}{2} \|\boldsymbol{w}_{\bar{\mathcal{S}}}\|_2^2 \quad . \tag{3.3}$$

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:

$$\Omega(\boldsymbol{w}) = \frac{\alpha}{2} \sum_{j \in \mathcal{S}} \hat{F}_{jj} \left(w_j - w_j^0\right)^2 + \frac{\beta}{2} \|\boldsymbol{w}_{\bar{\mathcal{S}}}\|_2^2 \quad , \tag{3.4}$$

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:

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

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 $(f_1(\boldsymbol{x}^{(i)}; \boldsymbol{w}), \ldots, f_K(\boldsymbol{x}^{(i)}; \boldsymbol{w}))$.

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

$$\Omega(\boldsymbol{w}) = \alpha \left\| \boldsymbol{w}_{\mathcal{S}} - \boldsymbol{w}^0_{\mathcal{S}} \right\|_1 + \frac{\beta}{2} \left\| \boldsymbol{w}_{\bar{\mathcal{S}}} \right\|_2^2 \quad . \tag{3.5}$$

The usual L^1 penalty encourages sparsity; here, by using $\boldsymbol{w}^0_{\mathcal{S}}$ 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* (3.3) 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:

$$\Omega(\boldsymbol{w}) = \alpha \sum_{g=1}^{G} s_g \left\| \boldsymbol{w}_{\mathcal{G}_g} - \boldsymbol{w}^0_{\mathcal{G}_g} \right\|_2 + \frac{\beta}{2} \left\| \boldsymbol{w}_{\bar{\mathcal{S}}} \right\|_2^2 \quad , \tag{3.6}$$

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

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:

$$\Omega(\boldsymbol{w}) = \alpha \sum_{g=1}^{G} s_g \Big(\sum_{j \in \mathcal{G}_g} \hat{F}_{jj} \left(w_j - w_j^0 \right)^2 \Big)^{1/2} + \frac{\beta}{2} \left\| \boldsymbol{w}_{\mathcal{G}_0} \right\|_2^2 .$$

3.4 Experimental Results in Image Classification

In this section, we evaluate the aforementioned parameter regularizers for transfer learning on several pairs of source and target problems, and show the improvements of -*SP* regularizers on the standard L^2 in image classification. We use ResNet [He et al., 2016a] as our base network, presented in Section 2.1.3, since it has proven its wide applicability on transfer learning tasks. The source task is usually a classification task. Conventionally, if the target task is also a classification task, the fine-tuning process starts by replacing the last layer with a new one, randomly generated, whose size is defined by the number of classes in the target task.

3.4.1 Source and Target Databases

For comparing the effect of similarity between the source problem and the target problem on transfer learning, we chose two source databases: ImageNet [Deng et al., 2009] contains 1.2 million labeled images of 1000 objects for generic object recognition, and Places365-Standard [Zhou et al., 2018] has 1.8 million labeled images for 365 categories of scenes for scene classification. Likewise, we have four different databases related to four target problems: Caltech 256 [Griffin et al., 2007] contains different objects for generic object recognition; MIT Indoors 67 (Indoors67) [Quattoni and Torralba, 2009] consists of 67 indoor scene categories; Stanford Dogs 120 (Dogs120) [Khosla et al., 2011] contains images of 120 breeds of dogs; Foods101 [Bossard et al., 2014] collects photos of 101 food categories, and is a much larger database than the previous ones (yet with some noise in terms of image quality and class labels). Each target database is split into training and testing sets following the suggestion of their creators, except Dogs120. Testing set of Dogs120 is a subset of ImageNet training set but has no overlapping with ImageNet validation set. Since ImageNet training set is used as the source database, the evaluation in Dogs120 should avoid using the same images, so we use a part of ImageNet validation set, which contains only those 120 breeds of dogs, for evaluating the performance on Dogs120. Table 3.1 collects details

Table 3.1: Characteristics of the target databases: name and type, numbers of training and test images per class, and number of classes.

Database	task category	# training	# test	# classes
Caltech 256–30	generic object recog.	30	20	257
Caltech 256–60	generic object recog.	60	20	257
MIT Indoors 67	scene classification	80	20	67
Stanford Dogs 120	specific object recog.	100	50	120
Foods101	specific object recog.	750	250	101

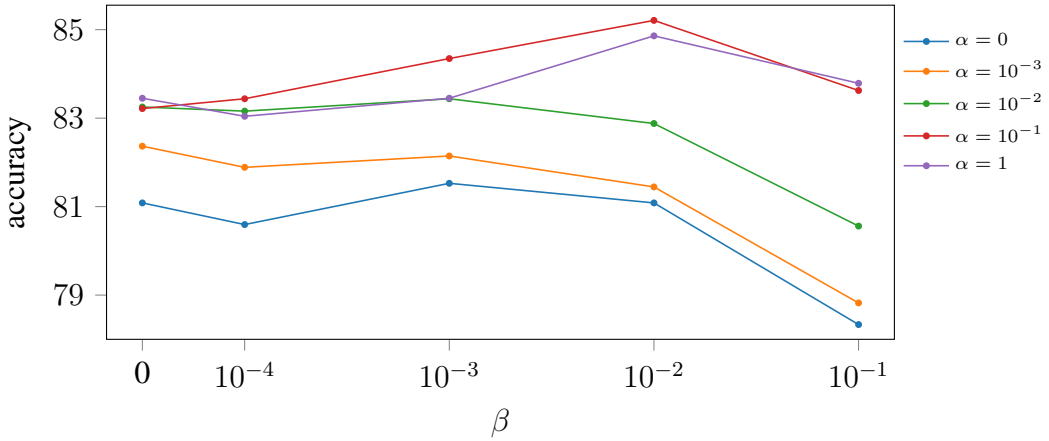


Figure 3.2: Classification accuracy (in %) on Stanford Dogs 120 for L^2-*SP*, according to the two regularization hyperparameters α and β respectively applied to the layers inherited from the source task and the last classification layer (see Equation 3.3).

for all target databases. In addition, we consider two configurations for Caltech 256: 30 or 60 examples randomly drawn from each category for training, and 20 remaining examples for test. Transfer learning may be less necessary when many training target examples are used for the target task, but it still benefits from the source problem.

3.4.2 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 256×256 and subtract the mean activity computed over the training set from each channel, then we adopt random blur, random mirror and random crop to 224×224 for data augmentation. The network parameters are regularized as described in Section

3.3. Note that the parameter regularizers are only applied to weights in convolutional and fully connected layers: the biases and parameters in the normalization layers are not penalized to follow the usual fine-tuning protocol. Cross validation is used for choosing the best regularization hyperparameters α and β: α differs across experiments, and $\beta = 0.01$ is consistently picked by cross-validation for regularizing the last layer. Figure 3.2 illustrates that the test accuracy varies smoothly according to the regularization strength, and that there is a sensible benefit in penalizing the last layer (that is, $\beta \geq 0$) for the best α values. When applicable, the Fisher information matrix is estimated on the source database. The two source databases (ImageNet or Places365) yield different estimates. Regarding testing, we use central crops as inputs to compute the classification accuracy.

Meanwhile, we perform another image pre-processing procedure for matching the state of the art: the aspect ratio of images is kept and images are resized with the shorter edge being 256. Regarding testing, we average the predictions of 10 cropped patches (the center patch, the four corner patches, and all their horizontal reflections) as final decision.

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. Then, under the best configuration, we repeat five times the learning process to obtain an average classification accuracy and standard deviation. All the experiments are performed with Tensorflow [Abadi et al., 2015]. The source code is publicly available for reproducibility purposes. [1]

3.4.3 Comparison across Penalties, Source and Target Databases

A comprehensive view of our experimental results is given in Figure 3.3. Each plot corresponds to one of the four target databases listed in Table 3.1. The red points mark the accuracies of transfer learning when using Places365 as the source database, whereas the 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. Besides

[1] https://github.com/holyseven/TransferLearningClassification

showing that choosing an appropriate source problem is critical in transfer learning (see Afridi et al. [2018]; Ding et al. [2018] for example), for our purpose of evaluating regularizers, these results display similar trends 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 task is not too similar to the source task.

Interestingly, the best source database for Foods101 is Places365 with L^2 regularization and ImageNet for the penalties using the starting point -*SP* as a reference. Considering the relative failure of L^2-*SP-Fisher*, it is likely that Foods101 is quite far from the two sources but slightly closer to ImageNet.

The benefit of the explicit bias towards the starting point is comparable for L^2-*SP* and L^2-*SP-Fisher* penalties; the strategies based on L^1 and Group-Lasso penalties behave rather poorly in comparison. They are even less accurate than the plain L^2 strategy on Caltech 256–30 when the source problem is Places365. Stochastic gradient descent does not handle well these penalties whose gradient is discontinuous at the starting point where the optimization starts. The stochastic forward-backward splitting algorithm [Duchi and Singer, 2009], which is related to proximal methods, leads to substandard results, presumably due to the absence of a momentum term. In the end, we used plain stochastic gradient descent on a smoothed version of the penalties eliminating the discontinuities of their gradients, but some instability remains.

3.4.4 Fine-Tuning from A Similar Source

Table 3.2 displays the results of fine-tuning with L^2-*SP* and L^2-*SP-Fisher*, which are compared to the current baseline of fine-tuning with L^2, and the state-of-the-art references [Ge and Yu, 2017; Martinel et al., 2018]. We report the average accuracies and their standard deviations on 5 different runs. Since we use the same data and the same starting point, runs differ only due to the randomness of stochastic gradient descent and to the parameter initialization of the last layer.

In the first part of Table 3.2 (first three lines), we observe that L^2-*SP* and L^2-*SP-Fisher* always improve over L^2 by a clear margin, and that this improvement is even more important when less data are available for the target problem (Caltech-30 *vs.* Caltech-60 and Foods101 *vs.* others). When less training examples are available for the target problem, the role of the regularizer is more important. Meanwhile, little

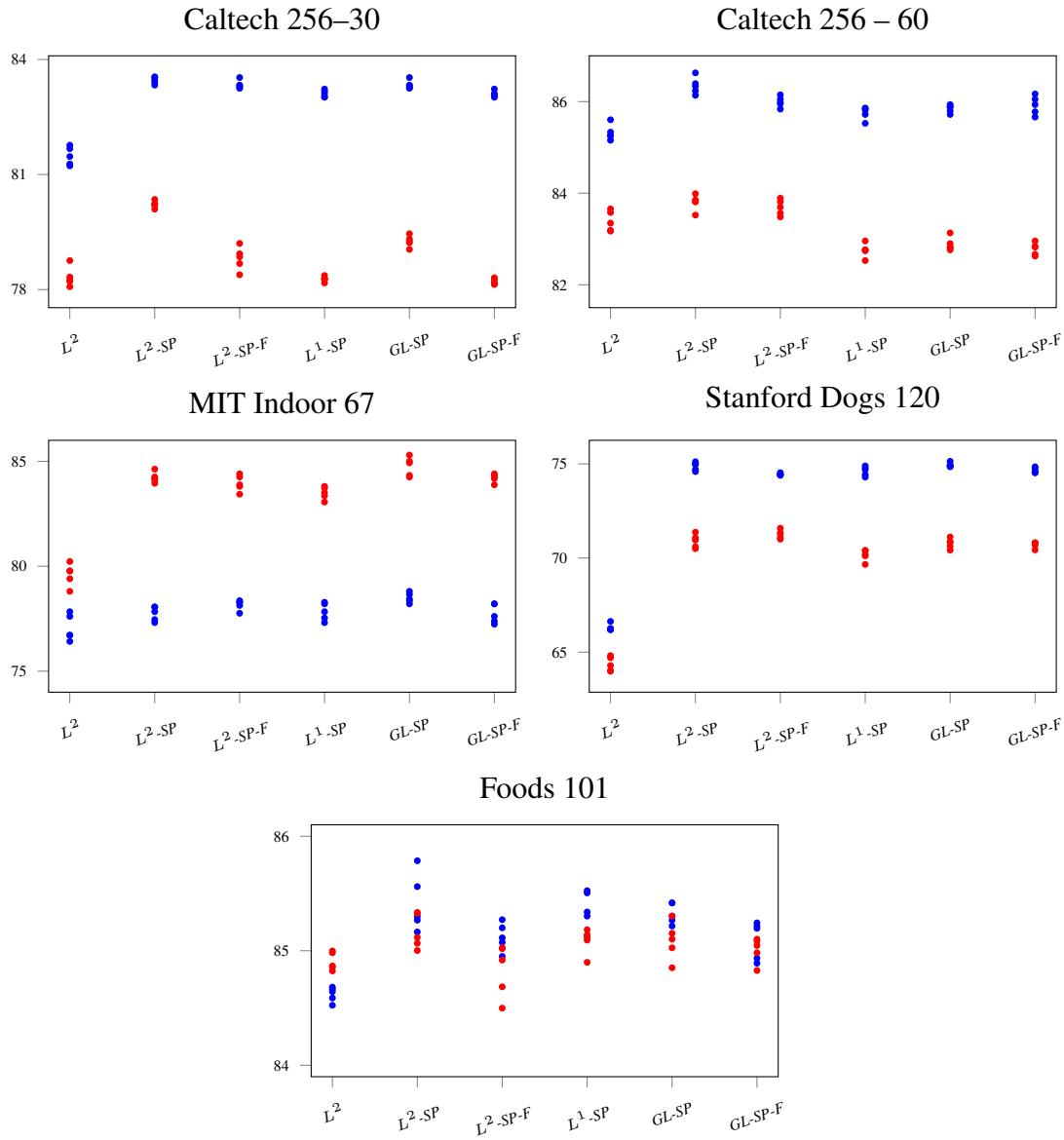


Figure 3.3: Classification accuracies (in %) of the tested fine-tuning approaches on the four target databases, using ImageNet (dark blue dots) or Places365 (light red dots) as source databases. MIT Indoor 67 is more similar to Places365 than to ImageNet; Stanford Dogs 120, Caltech 256 and Foods101 are more similar to ImageNet than to Places365.

Table 3.2: Average classification accuracies (in %) of L^2, L^2-*SP* and L^2-*SP-Fisher* on 5 different runs. The source database is Places365 for MIT Indoors 67 and ImageNet for Caltech 256 and Foods101. References of the state of the art are taken from Ge and Yu [2017], except for Foods101 where it is from Martinel et al. [2018]. For Dogs120, there is no reference that is based on the test set use here to avoid the overlap with Imagenet training set. Enhanced variants respecting the aspect ratio and using 10-crop test are marked with a star (*). Results with the highest accuracy in each part are highlighted in bold.

	Caltech-30	Caltech-60	Indoors67	Dogs120	Foods101
L^2	81.5±0.2	85.3±0.2	79.6±0.5	66.3±0.2	84.6±0.1
L^2-*SP*	**83.5±0.1**	**86.4±0.2**	**84.2±0.3**	**74.9±0.2**	**85.4±0.3**
L^2-*SP-Fisher*	83.3±0.1	86.0±0.1	84.0±0.4	74.4±0.1	85.1±0.1
L^{2*}	82.7±0.2	86.5±0.4	80.7±0.9	67.7±0.3	86.7±0.2
L^2-*SP**	**84.9±0.1**	**87.9±0.2**	**85.2±0.3**	**77.1±0.2**	**87.1±0.1**
L^2-*SP-Fisher**	84.8±0.1	**87.9±0.1**	**85.2±0.1**	76.9±0.1	87.0±0.1
Reference	83.8±0.5	89.1±0.2	85.8	—	90.3

difference is observed between L^2-*SP* and L^2-*SP-Fisher*. Note that we do not report here the performances of training from scratch, but that transfer learning really helps in these setups: we could only reach 76.9% accuracy on Foods101 (with 10 times more computing efforts, that is, number of epochs).

In the second part of Table 3.2, we boost the performance of fine-tuning with L^2, L^2-*SP* and L^2-*SP-Fisher* by exploiting additional training and post-processing techniques, that is, by respecting the aspect ratio of images and by using 10-crop test. The improved results are above state of the art for Caltech–30, and close to state of the art for Indoors67, without making use of the advanced techniques employed by Ge and Yu [2017] and Martinel et al. [2018]. These results show that simply changing the regularizer from L^2 to L^2-*SP* or L^2-*SP-Fisher* is remarkably efficient not only for baseline models, but also for more advanced ones.

3.5 Analyses

Among all -*SP* methods, while the results of different methods are similar, L^2-*SP* and L^2-*SP-Fisher* always reach a better accuracy on the target task. We expected L^2-*SP*-

Fisher to outperform L^2-*SP* since Fisher information provides a relevant metric in parameter space and was shown to help in lifelong learning, but there is no significant difference between the two options in our setups. Since L^2-*SP* is simpler than L^2-*SP-Fisher*, we recommend the former, and we focus on the analysis of L^2-*SP*, although most of the discussion would also apply to L^2-*SP-Fisher*.

3.5.1 Behavior on the Source Task

The variants using the Fisher information matrix behave like the simpler variants using a Euclidean metric on parameters. One reason is that, contrary to lifelong learning, our objective does not favor solutions that retain accuracy on the source task. Hence, the metric defined by the Fisher information matrix is less relevant for our actual objective that only relates to the target task. Table 3.3 reports the drop in performance when the fine-tuned models are applied on the source task, without any retraining, simply using the original classification layer instead of the classification layer learned for the target task. The performance drop is consistently smaller for L^2-*SP-Fisher* than for L^2-*SP*. This confirms that L^2-*SP-Fisher* is indeed a better approach in the situation of lifelong learning, where accuracies on the source tasks matter. In comparison to L^2-*SP-Fisher* and L^2-*SP*, L^2 fine-tuning results in catastrophic forgetting: the performance on the source task is considerably affected by fine-tuning.

The relative drops in performance with Foods101 follow the pattern observed for the other databases except that the decrease is much larger. This may be a sign of the substantial divergence of the data distribution of Foods101 from the one of ImageNet, with a compromise between the source task and the target task met far from the starting point.

3.5.2 Fine-Tuning *vs.* Freezing the Network

Freezing the first layers of a network during transfer learning [Yosinski et al., 2014] is another way to ensure a very strong inductive bias, letting less degrees of freedom to transfer learning. Figure 3.4 shows that this strategy, which is costly to implement if one looks for the optimal number of layers to be frozen, can improve L^2 fine-tuning considerably, but that it is a rather inefficient for L^2-*SP* fine-tuning. Among all possible choices, L^2 fine-tuning with partial freezing is dominated by the plain L^2-*SP* fine-tuning. Note that L^2-*SP-Fisher* (not displayed) behaves similarly to L^2-*SP*.

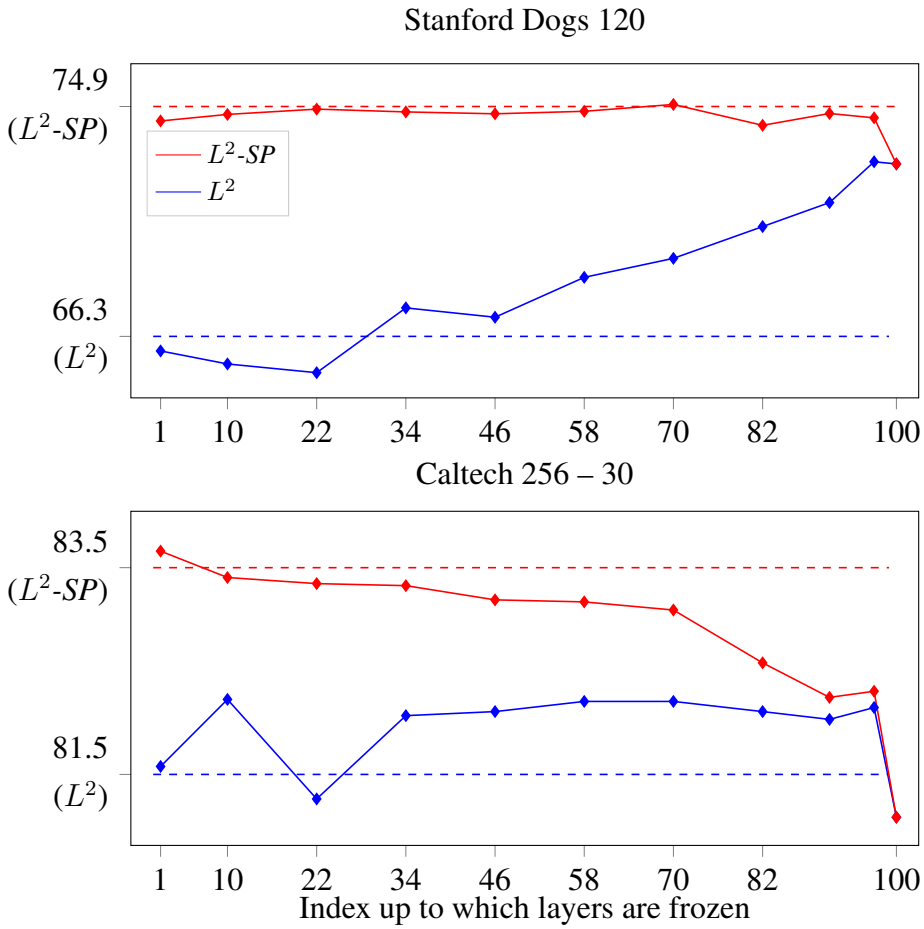


Figure 3.4: Classification accuracies (in %) of fine-tuning with L^2 and L^2-*SP* on Stanford Dogs 120 (top) and Caltech 256–30 (bottom) when freezing the first layers of ResNet-101. The dashed lines represent the accuracies reported in Table 3.2, where no layers are frozen. ResNet-101 begins with one convolutional layer, then stacks 3-layer blocks. The three layers in one block are either frozen or trained altogether.

Table 3.3: Classification accuracy drops (in %, the lower, the better) on the source tasks due to fine-tuning based on L^2, L^2-*SP* and L^2-*SP-Fisher* regularizers. The source database is Places365 for MIT Indoors 67 and ImageNet for Caltech 256, Stanford Dogs 120 and Foods101. The classification accuracies of the pre-trained models are 54.7% and 76.7% on Places365 and ImageNet respectively. Results with the lowest drops are highlighted in bold.

	L^2	L^2-*SP*	L^2-*SP-Fisher*
MIT Indoors 67	24.1	5.3	**4.9**
Caltech 256–30	15.4	4.2	**3.6**
Caltech 256–60	16.9	3.6	**3.2**
Stanford Dogs 120	14.1	4.7	**4.2**
Foods101	68.6	64.5	**53.2**

3.5.3 Layer-Wise Analysis

We complement our experimental results by an analysis relying on the activations of the hidden units of the network, to provide another view on the differences between L^2 and L^2-*SP* fine-tuning. Activation similarities are easier to interpret than parameter similarities, as they provide a view of the network that is closer to the functional prospective we are actually pursuing. Matching individual activations makes sense, provided that the networks slightly differ before and after tuning so that few roles are switched between units or feature maps.

The dependency between the pre-trained and the fine-tuned activations throughout the network is displayed in Figure 3.5, with boxplots of the R^2 coefficients, gathered layer-wise, of the fine-tuned activations with respect to the original activations. This figure shows that, indeed, the roles of units or feature maps have not changed much after L^2-*SP* and L^2-*SP-Fisher* fine-tuning. The R^2 coefficients are very close to 1 on the first layers, and smoothly decrease throughout the network, staying quite high, around 0.6, for L^2-*SP* and L^2-*SP-Fisher* at the greatest depth. In contrast, for L^2 regularization, some important changes are already visible in the first layers, and the R^2 coefficients eventually reach quite low values at the greatest depth. This illustrates in details how the roles of the network units are remarkably retained with L^2-*SP* and L^2-*SP-Fisher* fine-tuning, not only for the first layers of the networks, but also for the

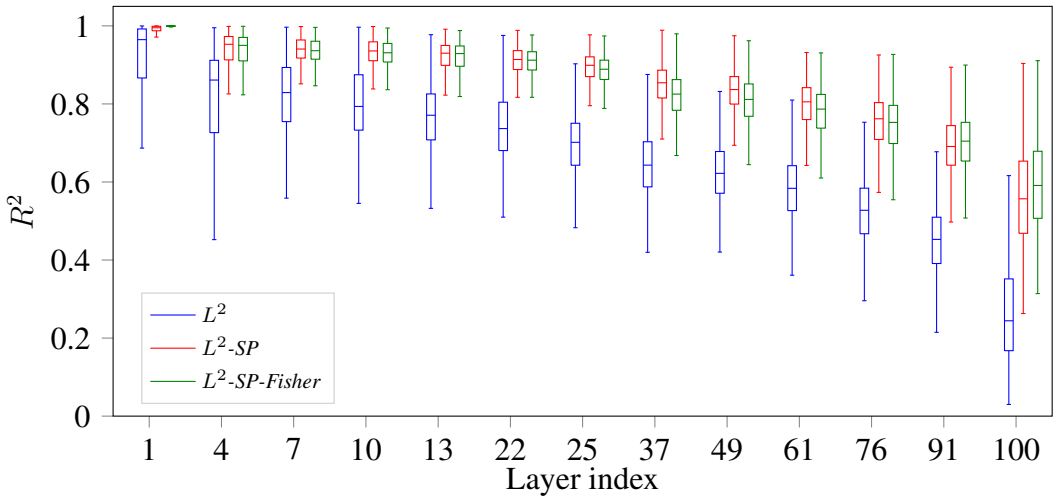


Figure 3.5: R^2 coefficients of determination with L^2 and L^2-*SP* regularizations for Stanford Dogs 120. Each boxplot summarizes the distribution of the R^2 coefficients of the activations after fine-tuning with respect to the activations of the pre-trained network, for all the units in one layer. ResNet-101 begins with one convolutional layer, then stacks 3-layer blocks. We display here only the R^2 at the first layer and at the outputs of some 3-layer blocks.

last high-level representations before classification.

We now look at the diagonal elements of the Fisher information matrix, still computed on ResNet-101 from training inputs of ImageNet. Their distributions across layers, displayed in Figure 3.6, show that the network is more sensitive to the parameters of the first layers, with a high disparity within these layers, and are then steady with most values within one order of magnitude. As a result, L^2-*SP-Fisher* is very similar to L^2-*SP*, except for being more conservative on the first layers. This observation explains the small differences between L^2-*SP* and L^2-*SP-Fisher* that are observed in our transfer learning setups.

3.5.4 Computational Efficiency

The -*SP* regularizers introduce no extra parameters, and they only increase slightly the computational burden. L^2-*SP* increases the number of floating point operations required for a learning step of ResNet-101 by less than 1%. Hence, at a negligible computational cost, we can obtain significant improvements in classification accuracy, and no additional cost is experienced at test time.

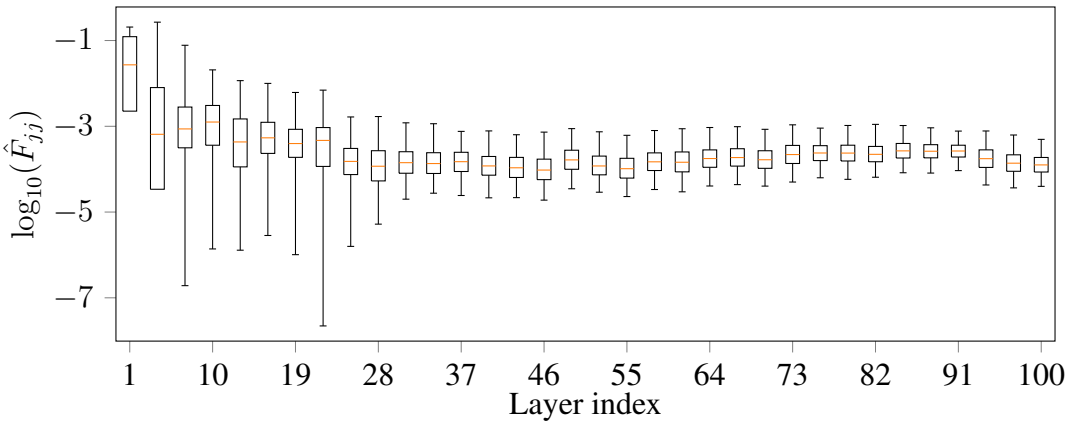


Figure 3.6: Boxplots of the diagonal elements of the Fisher information matrix (log-scale) computed on the training set of ImageNet using the pre-trained model. We display here these elements at the first layer and then at the last layer of all 3-layer blocks of ResNet-101.

3.5.5 Theoretical Insights

Effect of L^2-*SP*

Analytical results are very difficult to obtain in the deep learning framework. Under some (highly) simplifying assumptions, 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.

A similar analysis can be used for L^2-*SP* regularization. Let J be the unregularized objective function and $J^{SP}(\boldsymbol{w}) = J(\boldsymbol{w}) + \frac{\alpha}{2} \|\boldsymbol{w} - \boldsymbol{w}^0\|_2^2$ be the regularized objective function. Let $\widehat{\boldsymbol{w}} = \text{argmin}_{\boldsymbol{w}} J(\boldsymbol{w})$ and $\widehat{\boldsymbol{w}}^{\text{SP}} = \text{argmin}_{\boldsymbol{w}} J^{SP}(\boldsymbol{w})$ be their respective minima. The quadratic approximation of $J(\widehat{\boldsymbol{w}})$ gives

$$\mathbf{H}(\widehat{\boldsymbol{w}}^{\text{SP}} - \widehat{\boldsymbol{w}}) + \alpha(\widehat{\boldsymbol{w}}^{\text{SP}} - \boldsymbol{w}^0) = 0 \ , \tag{3.7}$$

where $\mathbf{H}$ is the Hessian matrix of J w.r.t. $\boldsymbol{w}$, evaluated at $\widehat{\boldsymbol{w}}$. Since $\mathbf{H}$ is symmetric and positive semidefinite, it can be decomposed as $\mathbf{H} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^\mathsf{T}$. Applying the decomposition to Equation (3.7), we obtain the following relationship between $\widehat{\boldsymbol{w}}^{\text{SP}}$ and $\widehat{\boldsymbol{w}}$:

$$\mathbf{P}^\mathsf{T}\widehat{\boldsymbol{w}}^{\text{SP}} = (\mathbf{\Lambda} + \alpha\mathbf{I})^{-1}\mathbf{\Lambda}\mathbf{P}^\mathsf{T}\widehat{\boldsymbol{w}} + \alpha(\mathbf{\Lambda} + \alpha\mathbf{I})^{-1}\mathbf{P}^\mathsf{T}\boldsymbol{w}^0 \ . \tag{3.8}$$

This equation shows that, in the direction defined by the i-th eigenvector of $\mathbf{H}$, $\widehat{\boldsymbol{w}}^{\mathrm{SP}}$ is a convex combination of the projections of $\widehat{\boldsymbol{w}}$ and $\boldsymbol{w}^0$ on that direction. Indeed noting λ_i the eigenvalue corresponding to the i-th eigenvector, the terms of the convex combination are $\frac{\lambda_i}{\lambda_i+\alpha}$ and $\frac{\alpha}{\lambda_i+\alpha}$.

This contrasts with L^2 that leads to a trade-off between the optimum of the unregularized objective function and the origin. Clearly, searching for a solution in the vicinity of the pre-trained parameters is intuitively much more appealing, since it is the actual motivation for using the pre-trained parameters as the starting point of the fine-tuning process.

Bias-Variance Analysis

We propose here a simple bias-variance analysis for the case of linear regression, for which this analysis is tractable. Consider the squared loss function $J(\boldsymbol{w}) = \frac{1}{2}\|\mathbf{X}\boldsymbol{w} - \boldsymbol{y}\|^2$, where $\boldsymbol{y} \in \mathbb{R}^n$ is a vector of continuous responses, and $\mathbf{X} \in \mathbb{R}^{n\times p}$ is the matrix of predictor variables. We use the standard assumptions of the fixed design case, that is: (i) $\boldsymbol{y}$ is the realization of a random variable $\mathbf{Y}$ such that $\mathbb{E}[\mathbf{Y}] = \mathbf{X}\boldsymbol{w}^*$, $\mathbb{V}[\mathbf{Y}] = \sigma^2\mathbf{I}_n$, and $\boldsymbol{w}^*$ is the vector of parameters; (ii) the design is fixed and orthonormal, that is, $\mathbf{X}^T\mathbf{X} = \mathbf{I}_p$. We also assume that the reference we use for L^2-*SP*, *i.e.* $\boldsymbol{w}^0$, is not far away from $\boldsymbol{w}^*$, since it is the minimizer of the unregularized objective function on a large data set: $\boldsymbol{w}^0 = \boldsymbol{w}^* + \boldsymbol{\varepsilon}$, where $\|\boldsymbol{\varepsilon}\| \ll \|\boldsymbol{w}^*\|$.

We consider the three estimates $\widehat{\boldsymbol{w}} = \mathrm{argmin}_{\boldsymbol{w}} J(\boldsymbol{w})$, $\widehat{\boldsymbol{w}}^{\mathrm{L}^2} = \mathrm{argmin}_{\boldsymbol{w}} J(\boldsymbol{w}) + \frac{\alpha}{2}\|\boldsymbol{w}\|_2^2$ and $\widehat{\boldsymbol{w}}^{\mathrm{SP}} = \mathrm{argmin}_{\boldsymbol{w}} J(\boldsymbol{w}) + \frac{\alpha}{2}\|\boldsymbol{w} - \boldsymbol{w}^0\|_2^2$. Their closed-form formulations are respectively:

$$\begin{cases} \widehat{\boldsymbol{w}} = \mathbf{X}^{\mathsf{T}}\boldsymbol{y} \\ \widehat{\boldsymbol{w}}^{\mathrm{L}^2} = \dfrac{1}{1+\alpha}\mathbf{X}^{\mathsf{T}}\boldsymbol{y} \\ \widehat{\boldsymbol{w}}^{\mathrm{SP}} = \dfrac{1}{1+\alpha}\mathbf{X}^{\mathsf{T}}\boldsymbol{y} + \dfrac{\alpha}{1+\alpha}\boldsymbol{w}^0 \end{cases} \tag{3.9}$$

So that their expectations and variances are:

$$\begin{cases} \mathbb{E}[\widehat{\boldsymbol{w}}] = \boldsymbol{w}^* \\ \mathbb{E}[\widehat{\boldsymbol{w}}^{\mathrm{L}^2}] = \dfrac{1}{1+\alpha}\boldsymbol{w}^* \\ \mathbb{E}[\widehat{\boldsymbol{w}}^{\mathrm{SP}}] = \dfrac{1}{1+\alpha}\boldsymbol{w}^* + \dfrac{\alpha}{1+\alpha}\boldsymbol{w}^0 \\ \qquad\qquad = \boldsymbol{w}^* + \dfrac{\alpha}{1+\alpha}\boldsymbol{\varepsilon} \end{cases} \tag{3.10}$$

$$\begin{cases} \mathbb{V}[\widehat{\boldsymbol{w}}] = \sigma^2 \mathbf{I}_p \\ \mathbb{V}[\widehat{\boldsymbol{w}}^{\mathrm{L}^2}] = \left(\frac{\sigma}{1+\alpha}\right)^2 \mathbf{I}_p \\ \mathbb{V}[\widehat{\boldsymbol{w}}^{\mathrm{SP}}] = \left(\frac{\sigma}{1+\alpha}\right)^2 \mathbf{I}_p \end{cases} \tag{3.11}$$

These expressions show that, without any regularization, the least squared estimate $\widehat{\boldsymbol{w}}$ is unbiased, but with the largest variance. With the L^2 regularizer, the variance is decreased by a factor of $1/(1+\alpha)^2$ but the squared bias is $\|\boldsymbol{w}^*\|^2\alpha^2/(1+\alpha)^2$. The L^2-*SP* regularizer benefits from the same decrease of variance and suffers from the smaller squared bias $\|\boldsymbol{\varepsilon}^*\|^2\alpha^2/(1+\alpha)^2$. It is thus always a better option than L^2 (provided the asumption $\|\boldsymbol{\varepsilon}\| \ll \|\boldsymbol{w}^*\|$ holds), and it is the best option regarding squared error when $\|\boldsymbol{\varepsilon}^*\|\alpha < \sigma$, which is likely when the sample size on the source task is much larger than the sample size on the target task.

Shrinkage Estimation

Using L^2-*SP* instead of L^2 can also be motivated by an analogy with shrinkage estimation [see e.g. Lehmann and Casella, 1998, chapter 5]. Although it is known that shrinking toward any reference is better than raw fitting, it is also known that shrinking towards a value that is close to the "true parameters" is more effective. The notion of "true parameters" is not readily applicable to deep networks, but the connection with Stein shrinking effect may be inspiring by surveying the literature considering shrinkage towards other references, such as linear subspaces. In particular, it is likely that manifolds of parameters defined from the pre-trained network would provide a more relevant reference than the single parameter value provided by the pre-trained network.

3.6 Other Setups

Experiments in the above sections are based on image classification and we have observed significant improvement in accuracy for all -*SP* regularizers. For demonstrating the versatility of -*SP* regularizers, we apply the L^2-*SP* regularizer, the most efficient one among -*SP* regularizers, on more various vision tasks under the inductive transfer learning setting.

In the following of this section, we perform experiments with seven representative approaches that largely benefit from transfer learning in image classification, image semantic segmentation as well as video analysis. All these approaches define a protocol relying at least partly on fine-tuning, originally implemented with weight decay, and will be compared with the L^2-*SP* regularizer. All these experiments show consistent improvement using L^2-*SP*, demonstrating the versatility of the -*SP* regularizers for fine-tuning state-of-the-art convolutional networks across network structures and datasets.

Among these seven approaches, FCN [Long et al., 2015a], ResNet (for image segmentation) [He et al., 2016a], DeepLab [Chen et al., 2018a] and PSPNet [Zhao et al., 2017] have been presented in details in Section 2.1.5. These four approaches are reproduced by ourselves and we compare the L^2 and L^2-*SP* regularizers with the same experimental protocol. As for the other three approaches, *i.e.*, EncNet [Zhang et al., 2018], SegFlow [Cheng et al., 2017] and DSTL [Cui et al., 2018], which will be introduced in the following of this section, we cooperate with the authors and have done the experiments with their original implementations under the very same conditions except a simple change in the parameter regularizer, turning the weight decay to the L^2-*SP* regularizer.

3.6.1 Transfer Learning Approaches

PSPNet Zhao et al. [2017] propose a pyramid pooling module for combining local, intermediate and global-scale information, in order to improve image segmentation performance. The pyramid pooling module is appended to the penultimate layer of the pre-trained model and pools the features maps in different scales. Then the pooled feature maps are upsampled and concatenated with the feature maps of the penultimate layer. The fused features contain local, global and intermediate-scale information, which hints the scene situation and helps the recognition of various categories in this scene.

The network equipped with this module is named as pyramid scene parsing network (*PSPNet*). Zhao et al. [2017] has built it based on an ImageNet-pretrained ResNet [He et al., 2016a], evaluated on the PASCAL VOC dataset [Everingham et al., 2010], ADE20K [Zhou et al., 2017] and Cityscapes [Cordts et al., 2016].

EncNet Zhang et al. [2018] have proposed a context-encoding module to extract the

relation between the global semantic context and the object categories, so as to emphasize the frequent objects in one scenario and de-emphasize the rare ones. The proposed module explicitly captures contextual information of the scene using sparse encoding and learns a set of scaling factors, by which the feature maps are then rescaled for selectively highlighting the class-dependent feature channels. For an image segmentation problem, the highlighted features containing the semantic context information, facilitate the pixel-wise prediction and improve the recognition of small objects. Meanwhile, an additional loss of predicting the presence of each object category is computed from the encoded features to better extract the contextual information.

This proposed approach, which we refer to as *EncNet*, keeping the name from Zhang et al. [2018], is built upon a pre-trained ResNet [He et al., 2016a] and evaluated on the PASCAL VOC dataset [Everingham et al., 2010] and ADE20K [Zhou et al., 2017] for image segmentation.

SegFlow Cheng et al. [2017] have proposed a network architecture, named *SegFlow*, with two branches for simultaneously segmenting video frames pixel-wisely and for computing the optical flow in videos. The segmentation branch is based on ResNet [He et al., 2016a] but modified to a fully-convolutional structure; and the optical flow branch is an encoder-decoder network [Dosovitskiy et al., 2015]. Both segmentation and optical flow branches have feature maps of multiple scales, enabling plausible connections between the two tasks. In *SegFlow*, the two branches are unified in a bi-directional way, i.e. the features from the segmentation branch are concatenated to the optical flow branch and the features from the optical flow branch are concatenated to the segmentation branch. Gradients from either task can pass through both branches and the information in the feature space can be shared maximally.

This unified network is initialized with the pre-trained ResNet [He et al., 2016a] and FlowNetS [Dosovitskiy et al., 2015], and then fine-tuned on the DAVIS 2016 dataset [Perazzi et al., 2016] for video object segmentation and the Scene Flow datasets [Mayer et al., 2016] for optical flow.

Domain Similarity for Transfer Learning Cui et al. [2018] propose to measure the similarity between domains using the Earth Mover's Distance (EMD) and then select from the source domain a subset that is more similar to the target domains for transfer learning. By averaging the image features in each category, the proposed approach is able to compute the similarities between any two domains. With a greedy selection

Table 3.4: Training and test details for segmentation on Cityscapes. Abbreviations used in this table: lr - learning rate; poly lr - polynomial learning rate policy; bs - batch size; bn - batch normalization; rdm scale - random scale; ms test - multi-scale test.

		FCN	ResNet	DeepLab	PSPNet
training	lr policy	fixed lr	poly lr		
	bs×h×w	2×800×800			8×864×864
	bn stats	frozen but trained β and γ			all training
	rdm scale	no			$[0.5, 2.0]$
test	ms test	no			yes
	image size	whole image			864×864 crops

strategy, they choose the top k categories from the source problem with the highest similarities for pre-training the network.

Using the proposed domain similarity estimation, they select two subsets from ImageNet [Deng et al., 2009] and iNaturalist [Van Horn et al., 2018] for transferring on different target databases: Subset A incorporates the 200 most similar categories for each of the seven databases, and Subset B is slightly biased towards bird and dog breeds. Experiments are conducted by pre-training on Subset A or B and then separately fine-tuning on each of the target databases. They compare the performance with pre-training from ImageNet, iNaturalist or the combination of ImageNet and iNaturalist, and show that the performance is improved by pre-training on a closer domain. Note that we refer to their approach as DSTL, the abbreviation of *domain similarity for transfer learning*.

3.6.2 Experimental Setup

In the following, we present the experimental setup for the seven approaches, including evaluation metrics, the source and target datasets and network architectures. For training and implementation details, we refer the reader to the original papers [Long et al., 2015a; Cheng et al., 2017; Zhang et al., 2018; Cui et al., 2018], a public repository[2] for DeepLab and our implementation[3] for PSPNet. The performances of FCN,

[2] https://github.com/DrSleep/tensorflow-deeplab-resnet
[3] https://github.com/holyseven/PSPNet-TF-Reproduce

ResNet, DeepLab and PSPNet may be different from the original works because of some modifications of experimental settings, like crop size and batch size, see Table 3.4, but when comparing with L^2-*SP*, the same setting is used. As for EncNet, SegFlow and DSTL, original experimental protocol are performed except the L^2-*SP* regularizer.

The parameter regularizers are only applied to weights in convolutional and fully connected layers as before.

Evaluation Metrics We recall the evaluation metrics used for measuring the performance on image classification, segmentation and optical flow estimation.

- For classification problems, the common *accuracy* is defined as the ratio of correctly predicted examples to total examples.
- For image segmentation, *pixel accuracy* is defined as the ratio of correctly predicted pixels to total pixels.
- Still for image segmentation, the *mean intersection over union* (mean IoU or mIoU) is more commonly used than pixel accuracy. The intersection over union (IoU) compares two sets: the set of pixels that are predicted to be of a given category and the set of ground truth pixels that truly belong to this category. It measures the discrepancy between the two sets as the ratio of their intersection to their union, *i.e.*. The mIoU is the mean of IoUs over all categories.
- For evaluating the optical flow, the average *endpoint error* (EPE) is defined as the average L^2 distance between the estimated optical flow vector and the ground truth vector at each pixel position.

Datasets As the most famous source database, ImageNet [Deng et al., 2009] contains 1.2 million labeled images of 1000 objects for generic object recognition, as we presented for classification tasks. iNaturalist [Van Horn et al., 2018] is also for classification and consists of 675K images from 5089 species of plants and animals. A selected part of ImageNet and iNaturalist towards bird and dogs breeds is used as the source database for DSTL because the target tasks of DSTL are mainly plants and animals. DSTL computes a similarity between target and source classes to select a subset of the classes of the source domain. This subset is used for pre-training the network, then the pre-trained network is fine-tuned with the target databases, like Birds200 [Welinder et al., 2010], Flowers102 [Nilsback and Zisserman, 2008], Cars196 [Krause

dataset	#images/scenes	#classes	task addressed	note	approach
ImageNet Deng et al. [2009]	∼1.2M	1000	image classification	object-centered	all
iNaturalist Van Horn et al. [2018]	∼675K	5,089	image classification	natural categories	DSTL
Flying Chairs Dosovitskiy et al. [2015]	∼2K scenes	-	optical flow	synthetic	SegFlow

Table 3.5: Source datasets. For each dataset, we provide the order of magnitude of the number of examples, the number of classes, the type of task addressed, and the approach(es) using this dataset for the source problem.

dataset	#images/seq.	#classes	task addressed	approach
Pascal Context Mottaghi et al. [2014]	∼10K	59	image segmentation	EncNet
Scene Flow Mayer et al. [2016]	32 sequences	-	optical flow	SegFlow
DAVIS 2016 Perazzi et al. [2016]	50 sequences	-	video segmentation	SegFlow
CUB200 Welinder et al. [2010]	∼11K	200	image classification	DSTL
Flowers102 Nilsback and Zisserman [2008]	∼10K	102	image classification	DSTL
Stanford Cars Krause et al. [2013]	∼16K	196	image classification	DSTL
Aircraft Maji et al. [2013]	∼10K	100	image classification	DSTL
Food101 Bossard et al. [2014]	∼100K	101	image classification	DSTL
NABirds Van Horn et al. [2015]	∼50K	555	image classification	DSTL

Table 3.6: Target datasets. For each dataset, we provide the order of magnitude of the number of examples, the number of classes, the type of task addressed, and the approach that uses this dataset for the target problem.

et al., 2013], Aircraft100 [Maji et al., 2013], Foods101 [Bossard et al., 2014] and NABirds [Van Horn et al., 2015]. They were collected for the recognition of birds, flowers, cars, aircraft models, foods, and birds respectively. They are relatively smaller than ImageNet and iNaturalist. The transfer learning approach DSTL aims at boosting the classification accuracy of target tasks by exploiting knowledge from the source domain.

As for the source database for image segmentation, besides ImageNet, Microsoft COCO [Lin et al., 2014b] has around 328K images and 2.5M segmented instances for object detection and instance segmentation. We use ImageNet as the source for all segmentation tasks, and also COCO for DeepLab when fine-tuning on Semantic Boundaries Dataset (SBD) [Hariharan et al., 2011]. SBD labeled images from the PASCAL VOC dataset [Everingham et al., 2010] and augmented pixel-wise annotations for image segmentation. The PASCAL Context has the same images as PASCAL VOC [Everingham et al., 2010] but provides detailed annotations for the whole scene, including the background. Different from object-center databases, Cityscapes [Cordts et al., 2016] focuses on urban street scenes and images of Cityscapes are much larger than those mentioned above. Cityscapes consists of 20,000 images with coarse annotations, and 5,000 images with high quality pixel-wise labeling, which are split into a training set (2975 images), a validation set (500 images) and a test set (1525 images). We use SBD as the target databases for PSPNet, PASCAL Context for EncNet, and Cityscapes for FCN, ResNet, DeepLab and PSPNet.

Flying Chairs [Dosovitskiy et al., 2015] is a synthetic dataset for optical flow estimation, generated by modeling chair models, adding them on a background image, and randomly sampling the motion of chairs. Compared with the other dataset concerning optical flow, Flying Chairs can provide enough training data, 22,872 image pairs. For the comparison, Scene Flow [Mayer et al., 2016] is also a synthetic dataset and has three subsets, two of which are used as target databases for SegFlow and have 8,591 and 4,392 annotated training frames respectively. These two subsets are named *Monkaa* and *Driving*. Monkaa is created using parts and pieces from the animated short film Monkaa, and Driving is a naturalistic, dynamic driving scenes with many objects, shadows, reflections and many complex scenarios. Davis 2016 [Perazzi et al., 2016] is another target database of SegFlow and has a total of 50 sequences, 3,455 annotated frames.

We summarize the source and target datasets in Table 3.5 and Table 3.6 respectively, including the size of each dataset, the task related and the approach that uses

this dataset.

Network Structures The source task is usually a classification task since this type of task minimizes the labelling burden. Conventionally, if the target task is also a classification task, like DSTL, the fine-tuning process starts by replacing the last layer with a new one, randomly generated, whose size is defined by the number of classes in the target task. The modification of the network structure in quite light in this situation.

In contrast, for image segmentation and optical flow estimation, where the objective differs radically from image classification, the source network needs to be modified, typically by adding a decoder part, which is much more elaborated than a single fully connected layer. Nevertheless, fine-tuning from the pre-trained network still performs favorably compared to training from scratch in these challenging situations. Here, we follow the exactly the original papers regarding the modifications of the network architectures.

3.6.3 Experimental Results

Table 3.7 compares the results of fine-tuning with L^2 and L^2-*SP* of all approaches on their specific target tasks. We readily observe that fine-tuning with L^2-*SP* in place of L^2 consistently improves the performance, whatever the task, whatever the approach. Some of these improvements are marginal, but we recall that, compared to the baseline methods with L^2 fine-tuning, changing the regularization to L^2-*SP* only adds a subtraction operation per weight during training, and that it does not cost anything during the inference process.

FCN, ResNet, DeepLab, PSPNet We fine-tuned FCN, ResNet, DeepLab and PSPNet with the standard L^2, and L^2-*SP*, all other things being equal. We readily observe that fine-tuning with L^2-*SP* in place of L^2 consistently improves the performance in mean Intersection over Union (mIoU) score, for all networks. The best model (PSPNet-extra with L^2-*SP*) has been evaluated on the test set and is currently on the public benchmark of Cityscapes[4], with 80.3% mIoU, to be compared to 80.2% obtained by Zhao et al. [2017].

[4] https://www.cityscapes-dataset.com/method-details/?submissionID=1148

approach	**target dataset**	**task**	**metric**	L^2	L^2-*SP*
FCN	Cityscapes	image segmentation	mIoU	66.9	67.9
ResNet-101	Cityscapes	image segmentation	mIoU	68.1	68.7
DeepLab	Cityscapes	image segmentation	mIoU	68.6	70.4
DeepLab-COCO	Cityscapes	image segmentation	mIoU	72.0	73.2
PSPNet	Cityscapes	image segmentation	mIoU	78.2	79.4
PSPNet-extra	Cityscapes	image segmentation	mIoU	80.9	81.2
PSPNet	SBD	image segmentation	mIoU	78.3	79.9
EncNet-50	PASCAL Context	image segmentation	mIoU	50.84	51.17
EncNet-101	PASCAL Context	image segmentation	mIoU	54.10	54.12
SegFlow*	DAVIS	video segmentation	IoU	65.5	66.2
SegFlow	DAVIS	video segmentation	IoU	67.4	68.0
SegFlow	Monkaa *Final*	optical flow	EPE	7.90	7.17
SegFlow	Driving *Final*	optical flow	EPE	37.93	30.31
DSTL	Birds200	image classification	accuracy	88.47	89.19
DSTL	Flowers102	image classification	accuracy	97.21	97.68
DSTL	Cars196	image classification	accuracy	90.19	90.67
DSTL	Aircraft100	image classification	accuracy	85.89	86.83
DSTL	Food101	image classification	accuracy	88.16	88.75
DSTL	NABirds	image classification	accuracy	87.64	88.32

Table 3.7: Summary of experimental results. For all metrics except EPE, higher is better. DeepLab-COCO means the network was pre-trained on Microsoft COCO before fine-tuning. PSPNet-extra means the fine-tuning involves the 20K coarsely labeled images while PSPNet only uses densely labeled images. SegFlow marked with '*' does not use the optical flow branch. The optical flow results of SegFlow are evaluated on two subsets of Scene Flow databases [Mayer et al., 2016], i.e. Monkaa and Driving.

approach	α	β	**accuracy**	**mIoU**
EncNet-50 - L^2	1e-4	1e-4	79.09	50.84
EncNet-50 - L^2-*SP*	1e-4	1e-3	79.10	50.31
EncNet-50 - L^2-*SP*	1e-3	1e-4	79.18	51.12
EncNet-50 - L^2-*SP*	1e-4	1e-4	**79.20**	**51.17**
EncNet-101 - L^2	1e-4	1e-4	80.70	54.10
EncNet-101 - L^2-*SP*	1e-4	1e-4	**80.81**	**54.12**

Table 3.8: EncNet pixel accuracy and mIoU on the PASCAL Context validation set according to regularization hyper-parameters.

For PSPNet on SBD, we apply the same protocol, except that images are cropped to 480×480, enabling a larger batch size of 16. The results on the public validation set are again in favor of L^2-*SP*, which reaches 79.9% in mIoU compared to 78.3% for L^2. On the test set, L^2-*SP* reaches 79.8%[5].

EncNet The EncNet-50 and EncNet-101 are based on ResNet-50 and ResNet-101 respectively, pre-trained on ImageNet. The networks are fine-tuned on the PASCAL Context for image segmentation, and their performance are measured by pixel accuracy and mIoU. The improvements in mIoU brought by L^2-*SP* for EncNet-101 are marginal; they are slightly larger for EncNet-50.

Table 3.8 goes into more details. It reports the average test pixel accuracy and mIoU obtained with several values of the hyper-parameters. Pixel accuracy, which is the criterion that is actually used during training, is always improved by L^2-*SP*, even for suboptimal choices of the regularization parameters.

Another interesting observation is that it is relatively safer to increase the α/β ratio than to decrease it. In other words, for controlling the complexity of the overall network, being more conservative on the pre-trained part of the network, that is, retaining its memory, is a better option than being more constrained on its novel part that allows to address the target task.

[5] http://host.robots.ox.ac.uk:8080/anonymous/NAAVTI.html

	Monkaa *Clean*		Monkaa *Final*	Driving *Clean*		Driving *Final*
	val	train+val	val	val	train+val	val
FlowNetS	10.51	6.15	10.47	66.93	23.90	67.15
SegFlow-L^2	7.94	4.49	7.90	37.91	14.35	37.93
SegFlow-L^2-*SP* $\beta = 1.0$	7.55	4.00	7.60	34.20	12.85	35.17
SegFlow-L^2-*SP* $\beta = 0.1$	**7.10**	**3.62**	**7.17**	31.11	**6.04**	30.31
SegFlow-L^2-*SP* $\beta = 0.01$	7.41	3.94	7.52	**30.57**	6.65	**30.14**

Table 3.9: Average endpoint errors (EPEs) on the two subsets of the Scene Flow dataset. The evaluations on the validation set of the Monkaa and Driving datasets use both *forward* and *backward* samples, while evaluations on train+val use only *forward* ones. Results of FlowNetS are from Dosovitskiy et al. [2015], and the hyperparameter α=0.1 for all SegFlow-L^2-*SP* settings.

SegFlow As for the segmentation performance of SegFlow, we have conducted two experiments, fine-tuning without the optical flow branch (SegFlow* in Table 3.7) and fine-tuning the entire model. Both options are evaluated on the DAVIS target task [Perazzi et al., 2016]. The segmentation branch and the optical flow branch of SegFlow are pre-trained on ImageNet [Deng et al., 2009] and FlyingChairs [Dosovitskiy et al., 2015] respectively. When applicable, both branches are regularized towards the pre-trained values by L^2 or L^2-*SP* fine-tuning. The benefits of L^2-*SP* are again systematic and higher than with EncNet.

For the optical flow estimation, we again observe systematic benefits of L^2-*SP*, with still higher impact (note that, for the EPE used to measure performance, the lower, the better). Table 3.9 reports additional results on optical flow estimation with SegFlow. There are two target tasks corresponding to Monkaa with 24 scenes, and Driving with 8 scenes. There are two versions for both datasets: a *Clean* version, which has no motion blur and atmospheric effects and a *Final* version, which includes some blurring effects.

We compare L^2-*SP* with the standard L^2 and test different choices of β when using L^2-*SP* during fine-tuning. We evaluate the optical flow estimation on the validation set and the train+val set. First, as expected, transferring from a synthetic Flying Chairs dataset to a synthetic animated Monkaa dataset is more effective than transferring to a

approach	α	β	**accuracy**
DSTL - Inception-V3 - L^2	4e-5	4e-5	88.47
DSTL - Inception-V3 - L^2-*SP*	1e-4	1e-4	89.07
DSTL - Inception-V3 - L^2-*SP*	1e-3	1e-3	**89.19**
DSTL - Inception-V3 - L^2-*SP*	1e-3	1e-2	88.53
DSTL - Inception-V3 - L^2-*SP*	1e-2	1e-3	89.12
DSTL - Inception-V3 - L^2-*SP*	1e-1	1e-3	89.00

Table 3.10: DSTL classification accuracy using the Inception-V3 network on the Birds200 validation set according to regularization hyper-parameters.

realistic driving scene dataset. Second, Table 3.9 shows that fine-tuning SegFlow with L^2-*SP* does not require an intensive search of hyper-parameters. Compared to L^2, L^2-*SP* performs better on a wide range of β values, covering several orders of magnitude: Suboptimal choices of (α, β) still allow for substancial reductions in errors. In addition, the comparison with FlowNetS, which is the optical flow branch of SegFlow, shows that the benefit of turning from L^2 to L^2-*SP* is sometimes comparable to the one of integrating the segmentation branch. It is likely due to the strong similarity of the domains.

DSTL The source datasets used here for DSTL are subsets of ImageNet [Deng et al., 2009] and iNaturalist [Van Horn et al., 2018], containing 585 categories slightly biased towards bird and dog breeds, i.e. Subset B in Cui et al. [2018]. We pre-train Inception-V3 [Szegedy et al., 2016] on this subset and fine-tune on six target datasets.

As shown in Table 3.7, fine-tuning with L^2-*SP* outperforms the baseline of L^2 on all six datasets. Table 3.10 investigates the classification accuracy with respect to the values of α and β in the validation set of Birds200 [Welinder et al., 2010], which is a dataset of 200 bird species. We have the same observations as with EncNet and SegFlow, i.e. the performance can be easily improved by using the L^2-*SP* penalty, and it is better to have a large α/β ratio.

3.6.4 Analysis and Discussion

Behavior across Network Structures L^2-*SP* behaves very well across all tested network structures. FCN is based on VGG and equipped with dilated convolution operation. DeepLab, PSPNet and EncNet are based on ResNet and strengthened by adding different modules. The segmentation branch of SegFlow is also based on ResNet transformed to fully convolutional; the flow branch is FlowNetS [Dosovitskiy et al., 2015], which is a variant of VGG. For DSTL, Inception-V3 [Szegedy et al., 2016] is used. Despite the various network structures and the diversity of problems addressed, we consistently observe higher performance from fine-tuning with L^2-*SP*.

Choosing α and β A practical problem may reside in the selection of the regularization parameters α and β of Equation (3.3). However, the experimental results show that this selection needs not to be accurate, and that a rule of thumb is to explore a large α/β ratio rather than a small one. When the pre-trained values are relevant, the target task benefits a lot from the pre-trained model, and α can be set to a large value without imposing detrimental constraints. As for β, which applies to the randomly initialized weights in the new layers, a large β would impede a necessary optimization. In practice, we roughly choose α and β by:

(i) starting from the default regularization rate as using the weight decay for both α and β;

(ii) α is free to increase to 10 times larger; β depends on the task but usually the optimal value is larger than the default;

(iii) if the optimization process is not affected a lot, repeat increasing α or β.

3.7 Conclusion

We described and tested some simple regularization approaches for transfer learning with convolutional networks. They all encode an explicit bias towards the solution learned on the source task, resulting in a trade-off between the solution to the target task and the pre-trained parameter that is coherent with the original motivation for fine-tuning. All the regularizers evaluated here have been already used for other purposes or in other contexts, but we demonstrated their relevance for inductive transfer learning with deep convolutional networks.

We show that a simple L^2 penalty using the starting point as a reference, L^2-*SP*, is useful, even if early stopping is used. This penalty is much more effective than the standard L^2 penalty that is commonly used in fine-tuning. It is also more effective and simpler to implement than the strategy consisting in freezing the first layers of a network. We provide theoretical hints and strong experimental evidence showing that L^2-*SP* retains the memory of the features learned on the source database. We thus believe that this simple L^2-*SP* scheme should be considered as the standard baseline in inductive transfer learning, and that future improvements of transfer learning should rely on this baseline.

We also conducted experiments with three state-of-the-art transfer learning approaches, *i.e.* EncNet [Zhang et al., 2018] on image segmentation, SegFlow [Cheng et al., 2017] on video analysis, and DSTL [Cui et al., 2018] on image classification. By doing so, we demonstrate that the L^2-*SP* regularization, used in place of the standard weight-decay, is very effective and versatile: not a single comparison is in favor of L^2 regularization. The L^2-*SP* parameter regularization is extremely simple to implement, requiring only to use the pre-trained model as the reference when computing the L^2 penalty. Furthermore, tuning the hyper-parameters of L^2-*SP* is not unduly time-consuming: the optimal values depend on the task, but a general rule works: large α is usually harmless while large β may be an obstacle for the optimization process. We conclude that for transfer learning tasks, L^2-*SP* is more appealing in intuition and in practice. We eventually recommend this simple L^2-*SP* scheme as the baseline of penalty for transfer learning tasks.

Besides, we tested the effect of more elaborate penalties, based on L^1 norm, Group-L^1 norm, or Fisher information. None of the L^1 or Group-L^1 options seem to be valuable in the context of inductive transfer learning that we considered here, and using the Fisher information with L^2-*SP*, though being better at preserving the memory of the source task, does not improve accuracy on the target task.

Chapter 4

Representation Regularizers for Fine-Tuning

Contents

Prologue Fine-tuning pre-trained deep networks is a practical way of transferring the knowledge learned from a source task to a target task. However, during fine-tuning, this knowledge may be forgotten, and detrimental to the performance on the target task, possibly leading to catastrophic forgetting. In the previous chapter, we focus on the parameter regularizers to counteract forgetting. In this chapter, we address this problem by explicitly preserving the representations resulting from the pre-trained network. Instead of freezing some intermediate representations during fine-tuning, we penalize deviations from the initial representations. Since the representations are not altered by transformations such as permutations of neurons at any layer of the network, our regularization functional relies on optimal transport to punish the deviation between the initial representations and the new ones. In practice, we rely on the optimal transport metrics to assess the deviations from the initial representations. The proposed approach only requires a mild increase in computing for training, thanks to the fast Sinkhorn algorithm, and improves the performance on visual recognition tasks.

4.1 Introduction

Fine-tuning pre-trained deep networks is a practical way of transferring the knowledge learned from a source task to a target task. During this transfer learning process, some form of knowledge is believed to be extracted by learning from the large-scale dataset of the source task. This knowledge is then transferred to the target task by initializing the model with the pre-trained parameters, but the knowledge may easily be forgotten during fine-tuning, and can harm the performance on the target task. An inductive bias towards the learned knowledge has been proved to be beneficial to address this problem, and can be implemented through regularizers. For example, the parameter regularizers, as presented in Chapter 3, constrain the parameters to remain in the vicinity of the initial values, in order to preserve the knowledge encoded in these parameters.

However, regularizing in the parameter space is uncomfortable, since the relationship between parameters and the function implemented by a deep convolutional network is at best obscure. As a result, knowledge is stored in parameters in an inexplica-

ble way, but it concretely incarnates in representations of the input data. Specifically, we transform the input data through the parametric model and obtain the representations, based on which we then do the tasks, like classification or logistic regression. Parameters are important for having good representations, but the representations are in fact the decisive aspect for the task. So in this chapter, instead of working on parameter regularizers for avoiding catastrophic forgetting, we propose to exploit regularizers of penalizing the deviations from the initial representations during transfer learning through aligning the *neuron distribution* to the initial one.

We recall the notion of the neuron distribution as mentioned in Section 2.3.4. At some layer of the neural network, we consider that neurons are samples drawn from a conditional distribution given the neurons at the previous layer, and the parameters at the current layer are responsible for generating the samples, *i.e.* the neurons at this layer. This conditional distribution is the neuron distribution that we would like to preserve during transfer learning. In our modeling, we consider the set of vectors, where each vector is the activations of each neuron from a number of data examples. Each neuron is a sample in our modeling and measured by a number of data examples. This modeling has also been applied for analyzing the representations of neural networks [Raghu et al., 2017].

During transfer training, matching directly each neuron to the original value is often a suboptimal choice. Without any restriction or even with *-SP* regularizers, some neurons will swap, that is, permutations or linear transformations of neurons will happen during transfer learning, as shown in Section 4.6.2. We also note that any permutation of neurons at any layer does not change the representations, neither affect the expressive capacity of the network, due to the structure of neural networks. So the regularization should be able to recognize or rectify these permutations. Thereby, without punishing these permutations, our proposed representation regularizers focus on aligning the neuron distribution to the initial one, where the difference between distributions can be described by the *optimal transport theory*. In practice, we rely on the smoothed solvers of the optimal transport problem to assess the deviations of representations from the initial ones, and those solvers only require a mild increase in computing for training, thanks to the fast Sinkhorn algorithm.

In this chapter, we still address the inductive transfer learning setting, where the target data distribution is identical or similar to the source data distribution, and the target task is different from the source task. We furthermore focus on the case where a vast amount of data was available for training on the source problem, and some

limited amount of labeled data is available for solving the target problem. Under this setting, we propose a novel regularizer of aligning the representations with the initial ones during fine-tuning, with respect to neurons in the network. We compare with two extreme baselines, *i.e.*, (1) identity mapping (without transport), (2) random transport (with a random matrix as the transport plan). From the comparison, we show that the proposed regularizer based on the optimal transport is better than the two baseline approaches. We then laterally compare the representation regularizers with parameter regularizers, and conclude that representation regularizers outperform parameter regularizers in most cases, at the cost of a mild increase in training time. We also propose some analyses and discussions about the transport effect and the neuron distributions, and share our attempts of other representation regularizers.

We present related work in Section 4.2, then a brief recall of the optimal transport problem and the solutions. We introduce the proposed regularizer and two baselines for comparison in Section 4.4. In Section 4.5, we show the experimental results in favor of our proposed regularizer based on the optimal transport. More analyses and discussions are provided in Section 4.6. Some other forms of regularizers that are also based on the optimal transport and yet futile are also noted in Section 4.7. Finally we conclude this chapter in Section 4.8.

4.2 Related Work

The regularizer we propose in this chapter is related to two parts of work. We first review some approaches that work on representations for transfer learning. Then we recall some applications based on the optimal transport theory.

Working on the network representations is not uncommon in transfer learning. Hinton et al. [2015] transferred the knowledge to a smaller target model by using a soft label for each training example that is computed by the source model in the softmax output for the source task. Tzeng et al. [2015] proposed to encourage the unlabeled target examples to approach the soft labels computed by the source model, and applied to domain adaptation, where the source and target tasks are the same but the domains are different, and usually only a few labeled data are available for the target problem. Similarly, Li and Hoiem [2017] registered the soft responses of target examples from the source model, and improved the performance in lifelong learning. Tzeng et al. [2014]; Long et al. [2015b]; Rozantsev et al. [2019] proposed to encourage the network to learn the domain-invariant representations via reducing the maximum

mean discrepancy between source and target data distributions.

The optimal transport theory has a wide range of applications. In terms of deep networks, the optimal transport is quite popular in generative models [Arjovsky et al., 2017; Gulrajani et al., 2017] with the Kantorovich-Rubinstein duality, which searches an optimal function under 1-Lipschitz constraint for solving the OT problem. The generative adversarial networks (GANs) model with deep networks the processing of generating examples. Thus minimizing the distance between distributions is the objective of GANs, which matches the objective of the OT problem. Instead of reshaping the OT problem to the duality form, Genevay et al. [2018]; Chen et al. [2018c]; Bousquet et al. [2017] and more directly exploit the regularized OT cost for training GANs. Beyond GANs, Courty et al. [2017] transformed the domain adaptation problem to an optimal transport one by adding a group-sparsity term on the transport plan, and solved it with Sinkhorn algorithm. The optimal transport theory is also helpful in tag prediction [Frogner et al., 2015], comparing documents [Kusner et al., 2015; Huang et al., 2016], dictionary learning [Rolet et al., 2016] *etc*.

4.3 A Reminder on the Optimal Transport Problem and the Sinkhorn Solvers

Let $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ be two discrete probability distributions: $\boldsymbol{\mu} = \sum_{i=1}^{n} \mu_i \delta_{\boldsymbol{x}_i}$ and $\boldsymbol{\nu} = \sum_{i=1}^{m} \nu_i \delta_{\boldsymbol{y}_i}$, where $\sum_{i=1}^{n} \mu_i = \sum_{i=1}^{m} \nu_i = 1$, and $\delta_{\boldsymbol{x}}$ is the Dirac delta function at position $\boldsymbol{x}$. Then with a defined cost function d and the cost matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$, where $\mathbf{M}_{ij} = d(\boldsymbol{x}_i, \boldsymbol{y}_j)$, we can give the optimal transport cost in the Kantorovich-relaxed OT problem:

$$L_{\mathbf{M}}(\boldsymbol{\mu}, \boldsymbol{\nu}) = \min_{\mathbf{P} \in U(\boldsymbol{\mu}, \boldsymbol{\nu})} \langle \mathbf{P}, \mathbf{M} \rangle_F, \tag{4.1}$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product, $U(\boldsymbol{\mu}, \boldsymbol{\nu})$ is the set of all possible joint distributions of $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$. The optimal joint distribution that minimizes the transport cost is the optimal transport plan $\mathbf{P}_0$:

$$\mathbf{P}_0 = \operatorname*{argmin}_{\mathbf{P} \in U(\boldsymbol{\mu}, \boldsymbol{\nu})} \langle \mathbf{P}, \mathbf{M} \rangle_F. \tag{4.2}$$

Linear programming is a solver for the OT problem because of the linear objective function and linear constraints, however, its computational budget increases in a cubic rate or more [Pele and Werman, 2009] with the n or m increasing. It is difficult to make it applicable in practice with a large model or a large dataset.

Recently, Cuturi [2013] proposed to search for an approximate solution to the OT problem by computing

$$\mathbf{P}_0^{\lambda} = \underset{\mathbf{P} \in U(\boldsymbol{\mu}, \boldsymbol{\nu})}{\operatorname{argmin}} \langle \mathbf{P}, \mathbf{M} \rangle_F - \frac{1}{\lambda} h(\mathbf{P}), \tag{4.3}$$

where $\lambda \in (0, +\infty)$ and $h(\mathbf{P}) = -\sum_{i,j} p_{ij} \log p_{ij}$ is the entropy. The solver to the problem 4.3 is the simple iterative method Sinkhorn-Knopp algorithm. The Sinkhorn iterations are sometimes numerically unstable and sensible to the choice of λ. Our experiments often encounter that instability, whereas it can be relieved by the proximal point algorithm with any Bregman divergence, suffering from slightly more computation cost, as proposed by [Peyré and Cuturi, 2018, Remark 4.9] and Xie et al. [2018].

4.4 Representation Regularizers

In this section, we detail the proposed regularizer on representations. We also compare the regularizer via the optimal transport theory with two baselines: one without any transport, and another with a random matrix.

We consider a pre-trained neural network of $L+1$ layers, which has L layers before the final classification/regression layer. Let $\mathbf{A}_l^{(t)} \in \mathbb{R}^{d_l \times n}$ be the representations after t fine-tuning iterations of the n examples of a given batch provided by the activations of the d_l neurons at layer l, where $1 \leq l \leq L$. For simplicity, we note $\mathbf{A}^{(t)}$ the representations at L^{th} layer $\mathbf{A}_L^{(t)}$. We indicate that $\mathbf{A}^{(0)}$ are the representations computed from the pre-trained model on the (same) target examples.

4.4.1 Representation Regularization via Optimal Transport

We introduce the regularization approach based on the OT theory, by starting to define the cost matrix for the transport:

$$\mathbf{M}_{ij}^{(t)} = \left\| \mathbf{A}_{i\cdot}^{(t)} - \mathbf{A}_{j\cdot}^{(0)} \right\|_2 \quad , \tag{4.4}$$

where $\mathbf{A}_{i\cdot}^{(t)} \in \mathbb{R}^n$ is the i^{th} row of $\mathbf{A}^{(t)}$, gathering the activations of the i^{th} neuron from n data examples, and $\|\cdot\|_p$ is the p-norm of a vector. Then we search the entropic solution for the optimal transport problem as defined in Equation 4.3, via Sinkhorn-Knopp algorithm [Cuturi, 2013] or IPOT algorithm [Xie et al., 2018] to obtain the optimal transport matrix $\mathbf{P}^{(t)}$

$$\mathbf{P}^{(t)} = \underset{\mathbf{P} \in \Pi(\mu, \nu)}{\operatorname{argmin}} \langle \mathbf{P}, \mathbf{M}^{(t)} \rangle_F - \frac{1}{\lambda} h(\mathbf{P}) \quad . \tag{4.5}$$

The proposed representation regularizer is then given by the approximate optimal transport cost:

$$\Omega_P = \sum_{i,j} \mathbf{P}_{ij}^{(t)} \mathbf{M}_{ij}^{(t)} = \langle \mathbf{P}^{(t)}, \mathbf{M}^{(t)} \rangle_F \quad . \tag{4.6}$$

If there is no swap between neurons during learning, Ω_P will be equivalent to penalizing the trace of $\mathbf{M}_{ij}^{(t)}$ since $\mathbf{P}^{(t)}$ will be a scaled identity matrix. In the case of a permutation of neurons, $\mathbf{P}^{(t)}$ is able to indicate the permutation since it searches for the minimal cost of transporting, and thus Ω_P is invariant to the permutation of neurons.

We have several comments on the proposed representation regularization.

- We would like to emphasize that we consider the neurons as samples from an underlying *neuron distribution*. This is different from the data distribution as in Arjovsky et al. [2017]; Genevay et al. [2018]. The number of neurons d_L is thus the effective number of samples in our case and the batch size n is the length of each neuron.
- The cost matrix $\mathbf{M}(t)$ depends on the mini-batches, so does the optimal transport plan $\mathbf{P}^{(t)}$. Both of them will change along the updates. We can see them as a noisy estimation of the similarity among the neurons at iteration t, in order to learn a model that is more robust to the batch variability.
- For numerical stability of gradients through $\mathbf{M}_{ij}^{(t)}$, we employ a small constant ϵ for $\mathbf{M}_{ij}^{(t)}$, *i.e.* $\mathbf{M}_{ij}^{(t)} = \max\left(\left\|\mathbf{A}_{i\cdot}^{(t)} - \mathbf{A}_{j\cdot}^{(0)}\right\|_2, \epsilon\right)$, in order to avoid dividing by zero during the computation of gradients when $\mathbf{M}_{ij}^{(t)} = 0$.
- For numerical stability and fast computation, we do not compute the gradients of Ω_P through the Sinkhorn-Knopp iterations. This can be guaranteed by the envelop theorem and the Theorem 4.1 from Bonnans and Shapiro [1998] in theory, despite the numerical difference in practice. See the discussions in Section 4.6.4.

4.4.2 Two Baselines

Identity Mapping (No Transport) The situation where there is no transport, is equivalent to the case where $\mathbf{P}^{(t)}$ is always a scaled identity matrix, and the regularizer boils down to

$$\Omega_I = \sum_{i=1}^{d_L} \left\|\mathbf{A}_{i\cdot}^{(t)} - \mathbf{A}_{i\cdot}^{(0)}\right\|_2 \quad . \tag{4.7}$$

It is a special case of the OT regularizer Ω_P when $\mathbf{P}^{(t)} = \frac{1}{d_L}\mathbf{I}_{d_L}$ for all t. This regularizer thus encourages with hard constraints each neuron to output similar activations, despite the transformations among neurons during fine-tuning. Mathematically, this regularizer Ω_I has two similar forms, $\Omega_{square} = \sum_{i,j}(\mathbf{A}_{ij}^{(t)} - \mathbf{A}_{ij}^{(0)})^2$, which replaces the standard L^2 norm in Equation 4.7 by the squared Euclidean distance, and $\Omega_{data} = \sum_{j=1}^{n} \left\| \mathbf{A}_{\cdot j}^{(t)} - \mathbf{A}_{\cdot j}^{(0)} \right\|_2$, which encourages the network to yield similar representations to the original ones for each data sample. These three regularizers basically deliver the same performance according to our preliminary experiments, so we only show the results of Ω_I.

Random Transport During our experiments, we have observed that the optimal transport plan $\mathbf{P}^{(t)}$ does not show the convergence until the end of fine-tuning, see Section 4.6.3. We thus would like to exclude the possibility that the improvement may benefit from the randomness. Thereby, instead of searching the optimal transport plan $\mathbf{P}^{(t)}$ and resolving the optimal transport problem, we simply penalize the representations using a random matrix $\mathbf{R}^{(t)} \in \mathbb{R}^{d_L \times d_L}$ such that $\mathbf{R}_{ij}^{(t)} \geq 0$ and $\sum_{i,j} \mathbf{R}_{ij}^{(t)} = 1$, and the regularizer is

$$\Omega_R = \sum_{i,j} \mathbf{R}_{ij}^{(t)} \mathbf{M}_{ij}^{(t)} = \langle \mathbf{R}^{(t)}, \mathbf{M}^{(t)} \rangle_F \ . \tag{4.8}$$

Whereas Ω_I considers that the role of each neuron should be kept during fine tuning between the source and target task, Ω_R performs random correspondence between neurons at each step: Ω_R can be interpreted as a randomized regularizer for driving the mean (w.r.t. examples) averaged (w.r.t. neurons) activations close to their original values. In other words, whereas Ω_I is very stringent regarding neuron-to-neuron correspondance, Ω_R is extremely loose, which makes these two regularizers natural baselines for comparing with Ω_P regularization.

Note that there are algorithms for generating a random doubly stochastic matrix by an iterative process [Sinkhorn, 1964; Cappellini et al., 2009]. However, for computational reasons, we adopt a simpler scheme here, where we generate the entries of $\mathbf{R}^{(t)}$ from a uniform distribution followed by a single normalization ensuring that the entries sum up to one. Our matrix $\mathbf{R}^{(t)}$ is thus less constrained than a genuine transport matrix but is a transport matrix in expectation.

4.5 Experimental Results

In this section, we evaluate the aforementioned regularizers for transfer learning on five fine-grained datasets of visual recognition, and show the effectiveness of the OT regularizer Ω_P. We use the standard ResNet [He et al., 2016a] as the backbone network because of its wide applicability on transfer learning tasks. Since all the target tasks we consider are classification task, we start the conventional fine-tuning process by replacing the last layer with a new one with the size being defined by the number of classes in the target task.

4.5.1 Datasets

We choose ImageNet [Deng et al., 2009] as the source dataset for its large scale and balanced distribution of all categories, ensuring that the source task is rich enough. As for the target datasets, we choose several widely-applied datasets for transfer learning on aircraft models [Maji et al., 2013], birds [Welinder et al., 2010], cars [Krause et al., 2013], dogs [Khosla et al., 2011] and foods [Martinel et al., 2018]. Each target dataset is split into training and test sets following the suggestion of their creators, except for Stanford Dogs 120, whose original test set is a subset of the training set of ImageNet. Since ImageNet training set is used as the source dataset, the evaluation in Dogs120 should avoid using the same images, so we use a part of ImageNet validation set, which contains only those 120 breeds of dogs, for evaluating the performance on Dogs120.

4.5.2 Experimental Details

We describe our experimental settings. For all details, readers can refer to the source code, which will be publicly available soon for reproducibility purposes.

Pre-processing and Post-processing The pre-processing of images involves image resizing and data augmentation. We keep the aspect ratio of images and resize the images with the shorter edge being 256. We adopt random blur, random mirror and random crop to 224×224 for data augmentation during training. Regarding testing, we resize the image in the same ways as training, and then we average the scores of 10 cropped patches (the center patch, the four corner patches, and all their horizontal reflections) as final decision.

Stochastic Gradient Descent with Momentum SGD with momentum 0.9 is used for the optimization solver. We run 9000 iterations and divide the learning rate by 10 after 6000 iterations for all target tasks, except Foods101 for which we run 16000 iterations and divide the learning rate after 8000 and 12000 iterations. The batch size is 64. As for the learning rates, we use the cross validation for choosing the best learning among $\{0.005, 0.01, 0.02, 0.04\}$.

Regularizers Parameter regularizers follow the same experimental setup as in Chapter 3, and we also evaluate the case where no regularizer is applied. As for the regularizers on representations, we evaluate those described in Section 4.4, without concurrently applying parameter regularizers. There are 33 three-layer residual units in ResNet-101, and the penultimate layer we consider is effectively the output of the last unit. In practice, besides regularizing the representations of the penultimate layer, we penalize one additional layer in the middle of the network for a better effect of preserving the presentations from the *deep* network. We apply the regularizer Ω_P on the output of the $\{9^{\text{th}}, 19^{\text{th}}, 29^{\text{th}}\}$ residual unit separately, compare their performances, and find that penalizing the activations of the 19^{th} residual unit is the best. The regularization hyper-parameter is selected from a range of five logarithmically spaced values from 10^{-4} to 1 by cross validation.

4.5.3 Comparison across Regularizers

Table 4.1 shows the results of fine-tuning with different regularizers on five different target datasets. We report the average accuracies and their standard deviations on 5 different runs. Since we use the same data and the same starting point, runs differ only due to the randomness of stochastic gradient descent and to the parameter initialization of the last layer. Since Foods101 is a relatively large dataset and contains some noises, regularizers barely vary on it.

The results of Table 4.1 confirm that the L^2-*SP* regularizer is a better choice than the standard weight decay or the absence of regularization. We also observe that the identity transport Ω_I behaves in general similarly to L^2-*SP*, suggesting that penalizing directly the departures from the initial representations produces similar effects to constraining on the parameters here.

For our propose of evaluating regularizers, we can notice that the benefits from optimal transport are the largest among all regularizers. Compared to the identity

Table 4.1: Average classification precision (in %) of no-regularized, L^2, L^2-*SP*, Ω_I, Ω_R and Ω_P using ten-crop test. Each experiment is repeated 5 times to obtain the average and the standard deviation. We also provide the average accuracy of all five tasks at the last column.

datasets	Aircraft100	Birds200	Cars196	Dogs120	Foods101	mean
none	83.95±0.37	80.64±0.30	90.21±0.12	69.53±0.29	86.59±0.06	82.18
Ω_{L^2}	83.64±0.40	80.57±0.36	90.51±0.19	69.79±0.29	86.85±0.09	82.27
$\Omega_{L^2\text{-}SP}$	83.94±0.39	81.10±0.24	90.73±0.12	77.05±0.19	**87.15±0.14**	83.99
Ω_I	83.44±0.45	82.25±0.19	90.40±0.20	77.15±0.17	86.88±0.07	84.02
Ω_R	84.68±0.26	81.50±0.30	**91.55±0.12**	71.28±0.61	86.88±0.04	83.18
Ω_P	**85.19±0.36**	**82.37±0.24**	91.29±0.13	**77.43±0.13**	87.06±0.06	**84.67**

transport, the optimal transport Ω_P is better on all five datasets: searching an optimal transport plan helps to boost the performance ; in other words, penalizing the deviations from initial representations in terms of distributions is beneficial to the performance on the target task. Surprisingly, the random transport Ω_R is sometimes quite effective, even achieving the highest accuracy on Cars196, but not always, *e.g.* on Birds200 and Dogs120, note that The random transport Ω_R is not always an optimal choice for transfer learning as it ignores the knowledge from the source task and violates the objective of transfer learning. On the contrary, the regularizer of optimal transport Ω_P yields stable and high performance on datasets that are either different from or similar to the source dataset.

4.6 Analyses and Discussions

We provide some analyses and discussions about the proposed regularizer on representations based on the optimal transport theory.

4.6.1 Comparison with Transporting Data

The regularizer Ω_P does not apply on the data distribution as in other works [Arjovsky et al., 2017; Genevay et al., 2018; Courty et al., 2017]. We propose to penalize the distance between *the neuron distributions* based on their activations for two reasons.

First, at some layer of the deep network, especially at the penultimate layer, the representations will not change if we permutate the neurons of that layer, and neither will the capacity of the network. As we can see the possible transport during transfer learning (shown in Figure 4.1), the regularizers should not punish these permutations. Transporting on the activations of neurons seems appealing to work towards the goal, while transporting data seems impossible.

Second, transporting data focuses on the data distribution. However, during transfer learning, accessing to the source data is sometimes difficult and computational. Neuron distributions on the target data may be changed because of the updates of parameters, and we need to align the neuron distribution to the old one, in order to preserve the knowledge from the source task.

In addition, in the situation where the optimal transport is based on data distributions, a small batch size may lead to an unreliable approximation of optimal transport metrics, while a large batch size is beneficial to the estimation but increases quasi *quadratically* the computation time with the iterative Sinkhorn-Knopp algorithm. The batch size cannot be easily chosen. As for the regularizer Ω_P, batch size is relatively less important since we estimate the neurons with mini-batches for the learned model being more robust to the batch variability, and we simply choose the batch size to be the optimal value as in the standard fine-tuning process. As for the computation burden, increasing the batch size *linearly* increases the computation of Euclidean distance, but not the Sinkhorn-Knopp algorithm.

4.6.2 Possible Transport

The permutations among neurons during transfer learning can be computed thanks to the optimal transport on the representations of pre- and post-transfer. We start with parameters being constrained by L^2-*SP* during fine-tuning. Specifically, for each of the five target datasets, we randomly choose 3 000 examples, then pass them through the pre-trained model and the fine-tuned model with L^2-*SP*, to obtain their representations at each residual unit of the network ResNet-101, based on which we compute the optimal transport plans. Figure 4.1 shows the trace of the transport plans along the network, noting that the maximum value for the trace is 1.0, where the transport plan is evenly distributed in each diagonal element, that is, each neuron at that layer during transfer learning entirely maintains the original position for the representations. However, only one of the five target datasets, Dogs120 remains unchanged from the

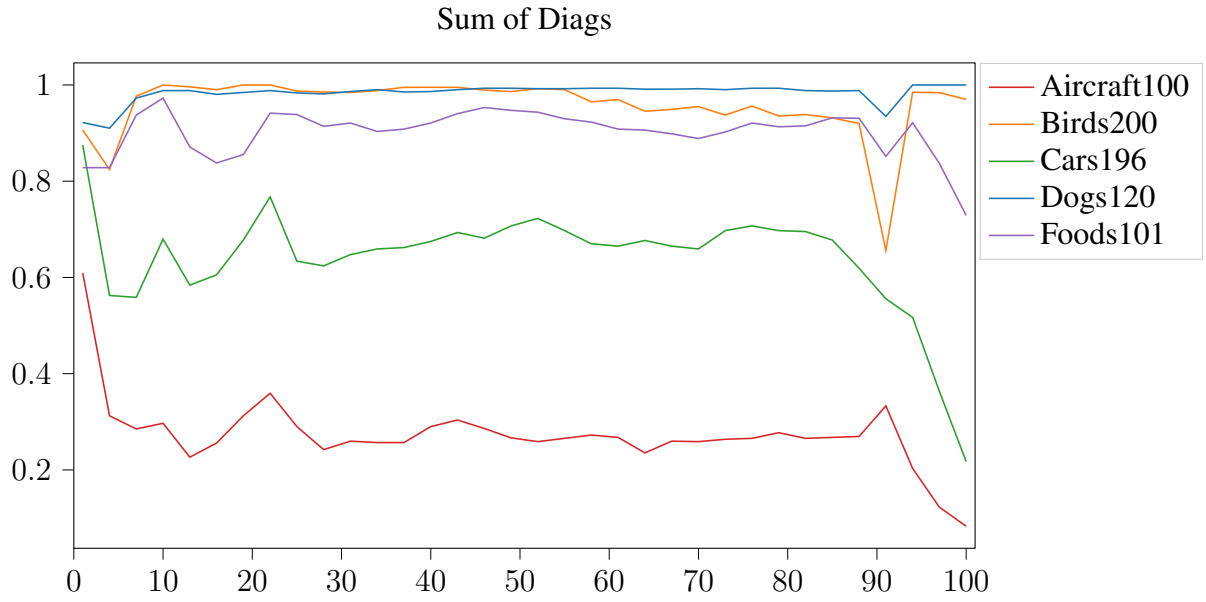


Figure 4.1: Traces of the optimal transport plans at the output of each ResNet-101 unit. The transport is computed between representations obtained through the pre-trained model and those obtained through the fine-tuned model by L^2-*SP*.

beginning to the end of the network. That is because the training data distribution of Dogs120 is exactly the same as that of ImageNet, and the categories in Dogs120 to be classified are all in ImageNet. The dataset Birds200 has some images overlapped with ImageNet, but the task of Birds200 focuses on a more detailed classification of bird races. Thus for achieving a related but different task, the representations of Birds200 after transfer learning have slightly varied at end and permutated some. As for Foods101, Cars196 and Aircraft100, we can notice that the neurons all along the network have been permutated, more or less, after transfer learning.

We believe that the knowledge from the source task is beneficial for the target task, and an inductive bias towards the source knowledge is good for the transfer learning problem. The neurons do not always remain at their initial positions, and hence simply penalizing the Euclidean distances between them and their initial values is often a suboptimal choice. We thus propose to exploit the optimal transport for bypassing the permutations or linear transformations and encouraging the preservation of the source knowledge.

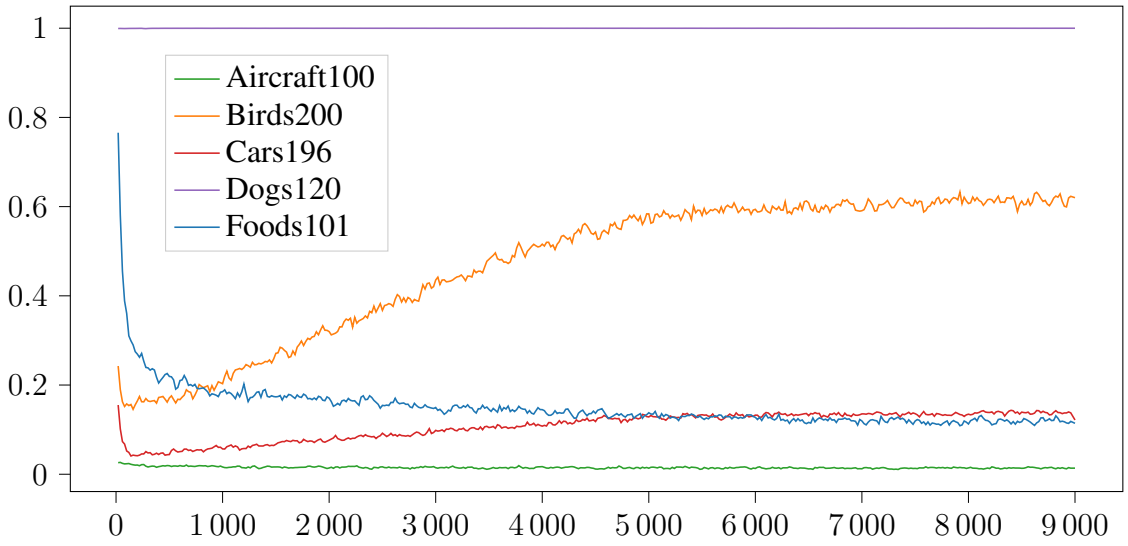


Figure 4.2: Traces of the optimal transport plans at the penultimate layer during training. Note that for Foods101, the fine-tuning process stops after 16 000 iterations, while during the last 7 000 iterations, the trace of the optimal transport plan follows the tendency as shown in this figure and converges around 0.11.

4.6.3 Effective Transport

Our regularizer scheme is based on the optimal transport, and able to take into account the permutations among neurons. Figure 4.2 displays the evolution of the optimal transport plans $\mathbf{P}^{(t)}$ during fine-tuning, measured in traces. We can notice that the computed optimal transport is very different from the identity transport on all target datasets, except Dogs120. Even on Dogs120, the transport is not exactly the identity and there are always a few neurons that are transformed during fine-tuning. These traces demonstrate that the identity is not always the optimal choice for regularizing the neurons during transfer learning, in order to preserve the representations, and the choice from the optimal transport may be a better option for addressing this problem. Comparing each curve in Figure 4.2, we can observe that the optimal transport does not behave the same on these five target datasets, as these datasets differ from the source dataset in different degrees, but the optimal transport is able to find the peculiar transform to each of the five target datasets and obtain better results than the identity or random transport.

4.6.4 The Envelope Theorem

The gradients of Ω_P can be compued in two ways: either we compute the gradients through the iterations of the Sinkhorn-Knopp algorithm, as proposed by Genevay et al. [2018]; or we apply the envelop theorem to avoid tracing the gradients through these iterations, as proposed by Xie et al. [2018]. For numerical stability and fast computation, we adopt the second option. We restate the two theorems and note that the conditions of using the two theorems are not satisfied in practice because of the numerical difference between the optimal solution and the computed one.

Theorem 4.1. ***Envelop theorem.*** *Let $f(\boldsymbol{p}, \boldsymbol{w})$ and $g_j(\boldsymbol{p})$, $j = 1, 2, \ldots, m$ be real-valued continuously differentiable functions on $\mathbb{R}^{n+l}$, where $\boldsymbol{p} \in \mathbb{R}^n$ are choice variables, and $\boldsymbol{w} \in \mathbb{R}^l$ are trainable parameters, and consider the problem of choosing $\boldsymbol{p}$, for a given $\boldsymbol{w}$, so as to:*

$$\max_{\boldsymbol{p}} f(\boldsymbol{p}, \boldsymbol{w}) \; s.t. \; g_j(\boldsymbol{p}) \geq 0, j = 1, 2, \ldots, m \text{ and } m \geq 0.$$

Now let $\boldsymbol{p}_0$ be the solution that maximizes the objective function f subject to the constraints and define the value function $V(\boldsymbol{w}) \equiv f(\boldsymbol{p}_0, \boldsymbol{w})$. If V is continuously differentiable, then

$$\frac{dV(\boldsymbol{w})}{d\boldsymbol{w}} = \frac{\partial f(\boldsymbol{p}_0, \boldsymbol{w})}{\partial \boldsymbol{w}}$$

Thus the derivative of the optimal transport regularizer Ω_P over $\boldsymbol{w}$ can be simply evaluated at $\mathbf{P}^{(t)}$. However, the imperfection here is that $\mathbf{P}^{(t)}$ is computed from an iterative algorithm, and the difference between the true optimal solution and the computed one always exists, due to the iteration stopping criteria and the numerical precision. The approximate gradient problem was addressed by Bach et al. [2004] with approximate optimality conditions.

The same conclusion can be drawn from the theorem below.

Theorem 4.2. *(Bonnans and Shapiro [1998]) Let X be a metric space and U be a normed vector space. Suppose that for all $x \in X$ the function $f(x, \cdot)$ is differentiable, that $f(x, u)$ and $\frac{\partial f(x,u)}{\partial u}$ are continuous on $X \times U$ and let Φ be a compact subset of X. Let define the optimal value function as $v(u) = \inf_{x \in \Phi} f(x, u)$. The optimal value function is directionally differentiable. Furthermore, if for $u_0 \in U$, $f(\cdot, u_0)$ has a unique minimizer x_0 over Φ then $v(u)$ is differentiable at u_0 and $\frac{dv(u_0)}{du} = \frac{\partial f(x_0,u)}{\partial u}$.*

4.7 Other Regularizers Tested

Before we conclude this chapter, we introduce some regularizers that we have tested without success.

Normalizing $\mathbf{A}^{(t)}$ After checking the cost matrices $\mathbf{M}$, we found that some neurons have over ten times larger magnitude values than others. So an operation of normalization was tested:

$$\mathbf{A}_{i\cdot} = \frac{\mathbf{A}_{i\cdot} - m_i}{\sigma_i} \quad , \tag{4.9}$$

where m_i and σ_i are the mean and standard deviation of $\mathbf{A}_{i\cdot}$. This normalization harms a little classification accuracy compared to Ω_P.

Mapping from Source to Target Instead of reducing the distance between distributions, we considered transforming the examples from the source task to the target task and aligning the target examples with transported source examples:

$$\Omega = \left\|\mathbf{A}^{(t)} - \mathbf{A}^{(0)}\mathbf{P}^{(t)}\right\|_2^2 \quad . \tag{4.10}$$

However, it harms quite a lot the performance.

Mapping from Target to Source Similarly, we tried to align the transported target examples to the source examples:

$$\Omega_{OT-3} = \left\|(\mathbf{P}^{(t)}\mathbf{A}^{(t)} - \mathbf{A}^{(0)}\right\|_2^2 \quad , \tag{4.11}$$

and use the transported neurons $\mathbf{P}^{(t)}\mathbf{A}^{(t)}$ to do the classification. The problem is the choice of the transport plan for inference. We tested transport plans among the last iterations, none of them helped; the identity matrix wass the best among all choices, but was not as good as Ω_P.

Gromov-Wasserstein Instead of aligning the neuron distributions, we try to preserve the structure of neurons from source to target via the Gromov-Wasserstein distance [Peyré et al., 2016]. The Gromov-Wasserstein problem is also an optimal transport problem, but instead of transporting directly the examples from one domain to another, it considers the inter-distances within one domain and transports to the inter-distances in another domain. Intuitively, it compares the structure of one domain with the structure of another domain. We didn't get any results from this regularizer because the computation is too heavy and numerically unstable.

Gaussian Distribution within Each Neuron Instead of computing Euclidean distance between $\mathbf{A}_{i\bullet}^{(0)}$ and $\mathbf{A}_{j\bullet}^{(t)}$, we computed their Wasserstein distance. We supposed that samples from the set of scalars $\mathbf{A}_{i\bullet}^{(t)}$ are drawn from a Gaussian distribution. The order-2 Wasserstein distance, endowed with Euclidean distance, between two Gaussian distributions has closed-form solution:

$$W^2(X,Y) = \|m_1 - m_2\|_2^2 + \mathrm{Tr}(\Sigma_1 + \Sigma_2 - 2(\Sigma_1^{1/2}\Sigma_2\Sigma_1^{1/2})^{1/2}) \ , \tag{4.12}$$

where $X \sim \mathcal{N}(m_1, \Sigma_1)$ and $Y \sim \mathcal{N}(m_2, \Sigma_2)$. In this way, Ω_P computes the distance between distributions of neurons in source and target domains of Gaussian distributions, estimated on examples in the mini-batch. However, one problem is ignored, that the examples in the set of $\mathbf{A}_{i\bullet}^{(t)}$ are in order and this order is followed for all neurons, but W^2 distance does not care about this order.

4.8 Conclusion

In this chapter, we coped with the inductive transfer learning scenarios where a large source database is available and fine-tuning is applied. We proposed to exploit the source knowledge to help a better transfer learning through regularizers. Instead of working on parameters as in Chapter 3, we focused on representations. We proposed a novel regularizer Ω_P of decreasing the distance of distributions on two sets of representations, one is computed from the pre-trained model that stores the source knowledge and another is from the training model, in order to more substantially benefit from the source task. We focused on aligning the neuron distributions rather than data distributions, since each neuron at some layer in the deep network can be considered as an independent sample drawn from a conditional distribution given neurons from the previous layer. Note that neurons may suffer from the permutations and transformations during transfer learning while these permutations do not harm the capacity of the neural network, we thus proposed to employ the optimal transport theory to compute the distance between distributions, in order to not punish the possible transformations among neurons and preserve the source knowledge.

We conducted experiments on five different target problems, using the convolutional network pre-trained on ImageNet, and compared the proposed regularizer with parameter regularizers and two other baselines of representation regularizers, the identity transport and the random transport. From the experimental results, we observed

that the proposed regularizer Ω_P globally outperforms the other regularizers, with being the best on three of five datasets, and the second best on the other two. The identity transport is able to achieve the equal performance to L^2-*SP*, showing that constraining the representations on the initial values produces similar effects to constraining on the parameters on these five datasets. Despite that the random transport unexpectedly achieved the best on one dataset, it is very different from Ω_P as it does not carry any knowledge from source and the improvement is totally from the randomness. On the contrary, Ω_P searches an optimal transport for preserving the knowledge from the source. Moreover, the increased computation from searching the optimal transport plans is mild for training, thanks to the fast Sinkhorn algorithm, and nothing for inference.

Chapter 5

Contributions and Perspectives

In this chapter we review the contributions of this thesis, and discuss possible tracks for future research.

5.1 Contributions

Deep networks demonstrate excellent performance in computer vision, natural language processing, speech recognition, machine translation as well as board games. Training deep networks from scratch requires large computation resources and countless data examples, and difficult due to a delicate gradient-based optimization process. Compared with training from scratch, transfer learning with deep networks relieves the pain, facilitating the optimization process with a good initialization of parameters, and alleviating the overfitting problem when few labeled data are available on the target problem, thanks to the accumulated/learned knowledge from solving previous source problems. However, transfer learning with deep networks may easily forget the knowledge gained from the source, thus harm the performance on the target problem.

The main idea through this thesis is to preserve the knowledge gained from the source problem, and implement an inductive bias encoded by regularization approaches. We address the transfer learning problems and focus on a practical setting where a large-scale source database is available and fine-tuning is applied, and present two different regularizers, those on parameters in Chapter 3 and those on extracted features in Chapter 4.

The parameter regularizers are critical and efficient when (transfer) learning on small databases. The traditional parameter regularizer is the standard L^2, also called

weight decay, for decaying the norm of the parameter vector and driving the parameters towards the origin. This, however, conflicts with the initialization of parameters for the fine-tuning process, which adopts a pre-trained model from the source problem as the starting point that is not necessarily close to the origin. We thus proposed a coherent parameter regularization approach, where the pre-trained model is both used as the starting point of the optimization process and as the reference in the penalty that encodes an explicit inductive bias, so as to help preserve the knowledge embedded in the initial network during fine-tuning. The parameter regularizers that encourage similarity with the starting point of the fine-tuning process were denoted with the *SP* suffix.

We evaluated -*SP* regularizers based on the L^2, Lasso and Group-Lasso penalties, which can freeze some individual parameters or groups of parameters to the pre-trained values. We also tested the L^2-*SP* and *Group-Lasso-SP* variants that use the Fisher information to measure similarity. They all encode an explicit bias towards the solution learned on the source task, resulting in a trade-off between the solution to the target task and the pre-trained parameter that is coherent with the original motivation for fine-tuning, and they all obtain comparable results on the classification tasks. We also provided theoretical hints and strong experimental evidence showing that L^2-*SP* retains the memory of the features learned on the source database. Based on L^2-*SP*, we conducted experiments with several state-of-the-art transfer learning approaches, demonstrating that L^2-*SP* is very effective and versatile: not a single comparison is in favor of the standard L^2 regularization. We thus believe that this simple L^2-*SP* scheme should be considered as the standard baseline in inductive transfer learning, and that future improvements of transfer learning should rely on this baseline.

Good parameters are important for yielding good representations, but the representations are in fact the decisive aspect for the task. Thereby, instead of penalizing parameters, Chapter 4 focuses on penalizing the features that are computed through the deep network, *i.e. representations*. We proposed a novel regularizer for penalizing the distance of neuron distributions that encode the representations of the target examples before and during transfer. Representations are invariant to the permutations of neurons, hence we propose to employ optimal transport to compute the distance between distributions, in order to not penalize the possible transformations among neurons and preserve the source knowledge. We compared the regularizer based on the optimal transport with the identity mapping and the random transport, and parameter regularizers. From the experiments, the proposed regularizer globally outperforms

the other regularizers.

5.2 Perspectives

In terms of parameter regularizers, there are several avenues of further research:

- L^1-*SP* and *GL-SP* suffer from the undefined gradients at beginning of, or even during, fine-tuning. The approximation of these regularizers is not an ideal solution. More efforts can be done with forward-backward splitting algorithm and proximal gradient descent [Duchi and Singer, 2009]. The sub-gradients *w.r.t.* L^1-*SP* and *GL-SP* can be derived in closed-form (hopefully), and update the parameters. However, the absence of a momentum term in the stochastic/proximal gradient descent iterations, downgrades the performance. Thus some research efforts on the momentum in proximal gradient descent may be needed.
- The Group-Lasso regularizer gives the same scale of gradients for the parameters in the same group, which suits well the convolutional kernel parameters as they work together to extract a feature. Sadly, due to the instability of sub-gradients, *GL-SP* is not the best among all -*SP* regularizers. Considering that group-lasso matches well the convolutional kernel, it will be interesting to explore more on the *GL-SP* regularizer.
- Fisher information matrix (FIM) is a natural metric for computing the informational difference between probability measures, its diagonal elements indicate the importance of parameters on the source problem. FIM is obviously the Hessian of the Kullback-Leibler divergence, or equivalent to the cross-entropy objective function when computing gradients of parameters, and relates to the natural gradients, which should be interesting if combined with parameter regularizers or in other possible ways. In brief, FIM seems a perfect metric on the parameter regularizer, especially when coupled with L^2-*SP*. We are surprised by the experimental results of L^2-*SP* and L^2-*SP-Fisher*, which states that FIM is not much helpful. Maybe the usage of batch normalization affects the estimate of Fisher information, maybe a more advanced coupling is needed. More research could be based on FIM.
- Batch normalization (BN) complicates the usage of regularizers. For instance, with BN, L^2-*SP* can be relaxed to the cosine similarity, because the normalization will eliminate the scale effect. We have compared the cosine similarity with

L^2-*SP*, which is not reported in this thesis, and found no difference. Moreover, the theory about BN is still lacking [Santurkar et al., 2018; Kohler et al., 2019], and its impact on the regularization could be also interesting.

Possible future directions for representations/neurons are:

- The random transport regularizer is surprisingly good on a few datasets. According to additional experiments (which are not reported in this thesis), the randomness is not the key to the improvement, because a random transport plan without randomness at each step $\mathbf{R}^{(t)} = \mathbf{R}^{(0)}$ can have the same performance. Furthermore, a scaled all-ones matrix, *i.e.* $\mathbf{P}^{(t)}_{ij} = (\frac{1}{d_L})^2$, which is the expectation of the random transport, yields also the same performance. It should be interesting and also surprising to deepen on the fact that the average transport is beneficial for the final discrimination task.
- Optimal transport can be used for measuring the distance between data distributions, and thus for choosing the best source dataset for a specific target dataset.
- Not limited to transfer learning, we would like to evaluate some simple approaches of exploiting the entanglement of class manifolds in feature space, *i.e.* the closeness of pairs of points from the same class and the distance of pairs of points from different classes. The simple cosine similarity or the soft nearest neighbor loss [Frosst et al., 2019] can be used to measure the entanglement. This can be used not only for the objective of training models, but also as an analytic tool for checking whether the learned representations are good or not with respect to a learning problem.
- The capacity of deep networks is sometimes large enough to memorize the whole dataset, even a large one [Zhang et al., 2017]. After training on the source domain, there are already some neurons that do not contribute to the discrimination as much as others, and a few of them contribute nothing. Although the regularizer Ω_P searches an optimal transport plan for better transfer learning and performs quite well, the performance could still be boosted by excluding the unactivated neurons during the optimal transport, or assigning them new roles [Li et al., 2019].
- Instead of aligning the distributions between source and target problems for preserving the knowledge, we can also maximize the mutual information between them during transfer learning, which is intuitively plausible. The estimation of

the mutual information between data distributions or neuron distributions can be done by neural networks [Belghazi et al., 2018] through Donsker-Varadhan representation [Donsker and Varadhan, 1983].

Publications

- Xuhong Li, Yves Grandvalet, and Franck Davoine. Explicit inductive bias for transfer learning with convolutional networks. In *International Conference on Machine Learning* (ICML), pages 2830–2839, 2018.
- Xuhong Li, Franck Davoine, and Yves Grandvalet. A simple weight recall for semantic segmentation: Application to urban scenes. In 2018 IEEE *Intelligent Vehicles Symposium* (IV), pages 1007–1012. IEEE, 2018.
- Xuhong Li, Yves Grandvalet, Franck Davoine. A Baseline Regularization Scheme for Transfer Learning with Convolutional Neural Networks. (submitted to the Pattern Recognition journal with feedback for minor revisions)
- Xuhong Li, . Transfer Learning with CNN: Memory Improves Transferring Skills. (conference paper under review)

Bibliography

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, Vincent 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.

Muhammad Jamal Afridi, Arun Ross, and Erik M Shapiro. On automated source selection for transfer learning in convolutional neural networks. *Pattern Recognition*, 73:65–75, 2018.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning (ICML)*, pages 214–223, 2017.

Yusuf Aytar and Andrew Zisserman. Tabula rasa: Model transfer for object category detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2252–2259, 2011.

Francis R Bach, Gert RG Lanckriet, and Michael I Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *International Conference on Machine Learning (ICML)*, page 6. ACM, 2004.

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(12):2481–2495, 2017.

Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. Regularization techniques for fine-tuning in neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1489–1494, 2017.

Jonathan Baxter. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine learning*, 28(1):7–39, 1997.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Devon Hjelm, and Aaron Courville. Mutual information neural estimation. In *International Conference on Machine Learning (ICML)*, pages 530–539, 2018.

J Frédéric Bonnans and Alexander Shapiro. Optimization problems with perturbations: A guided tour. *SIAM review*, 40(2):228–264, 1998.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European Conference on Computer Vision (ECCV)*, pages 446–461, 2014.

Olivier Bousquet, Sylvain Gelly, Ilya Tolstikhin, Carl-Johann Simon-Gabriel, and Bernhard Schoelkopf. From optimal transport to generative modeling: the vegan cookbook. *arXiv preprint arXiv:1705.07642*, 2017.

Valerio Cappellini, Hans-Jürgen Sommers, Wojciech Bruzda, and Karol Życzkowski. Random bistochastic matrices. *Journal of Physics A: Mathematical and Theoretical*, 42(36):365209, 2009.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Ciprian Chelba and Alex Acero. Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language*, 20(4):382–399, 2006.

Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

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. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(4):834–848, 2018a.

Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 801–818, 2018b.

Liqun Chen, Shuyang Dai, Chenyang Tao, Haichao Zhang, Zhe Gan, Dinghan Shen, Yizhe Zhang, Guoyin Wang, Ruiyi Zhang, and Lawrence Carin. Adversarial text generation via feature-mover's distance. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4666–4677, 2018c.

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 (ICML)*, pages 793–802, 2018d.

Jingchun Cheng, Yi-Hsuan Tsai, Shengjin Wang, and Ming-Hsuan Yang. SegFlow: Joint learning for video object segmentation and optical flow. In *IEEE International Conference on Computer Vision (ICCV)*, pages 686–695. IEEE, 2017.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, 2016.

Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(9):1853–1865, 2017.

Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4109–4118, 2018.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2292–2300, 2013.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

Zhengming Ding, Ming Shao, and Yun Fu. Incomplete multisource transfer learning. *IEEE Transactions on Neural Networks and Learning Systems*, 29(2):310–323, 2018.

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 *International Conference on Machine Learning (ICML)*, pages 647–655, 2014.

Monroe D Donsker and SR Srinivasa Varadhan. Asymptotic evaluation of certain markov process expectations for large time. iv. *Communications on Pure and Applied Mathematics*, 36(2):183–212, 1983.

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2758–2766, 2015.

John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934, 2009.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12 (Jul):2121–2159, 2011.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(4): 594–611, 2006.

Charlie Frogner, Chiyuan Zhang, Hossein Mobahi, Mauricio Araya, and Tomaso A Poggio. Learning with a wasserstein loss. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2053–2061, 2015.

Nicholas Frosst, Nicolas Papernot, and Geoffrey Hinton. Analyzing and improving representations with the soft nearest neighbor loss. In *International Conference on Machine Learning (ICML)*, 2019.

Weifeng Ge and Yizhou Yu. Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10–19, 2017.

Aude Genevay, Gabriel Peyre, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1608–1617, 2018.

Ross Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 249–256, 2010.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Adaptive Computation and Machine Learning. MIT Press, 2017.

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

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5767–5777, 2017.

Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *IEEE International Conference on Computer Vision (ICCV)*, pages 991–998, 2011.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 37(9):1904–1916, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)*, pages 630–645. Springer, 2016b.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988. IEEE, 2017.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL http://arxiv.org/abs/1503.02531.

Matthias Holschneider, Richard Kronland-Martinet, Jean Morlet, and Ph Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets*, pages 286–297. Springer, 1990.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7132–7141, 2018.

Gao Huang, Chuan Guo, Matt J Kusner, Yu Sun, Fei Sha, and Kilian Q Weinberger. Supervised word mover's distance. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4862–4870, 2016.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708, 2017.

Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2462–2470, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(1):221–231, 2013.

Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetting learning in deep neural networks. In *AAAI Conference on Artificial Intelligence*, 2018.

Leonid Kantorovich. On the transfer of masses (in russian). *Doklady Akademii Nauk*, pages 227–229, 1942.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7482–7491, 2018.

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.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

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*, 114(13):3521–3526, 2017.

Jonas Kohler, Hadi Daneshmand, Aurelien Lucchi, Thomas Hofmann, Ming Zhou, and Klaus Neymeyr. Exponential convergence rates for batch normalization: The power of length-direction decoupling in non-convex optimization. In *International Conference on Artificial Intelligence and Statistics (ICAIS)*, pages 806–815, 2019.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D object representations for fine-grained categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 554–561, 2013.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International Conference on Machine Learning (ICML)*, pages 957–966, 2015.

Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE Transactions on Neural Networks*, 8(1):98–113, 1997.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Erich L Lehmann and George Casella. *Theory of point estimation*. Springer, 2 edition, 1998.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Xingjian Li, Haoyi Xiong, Hanchao Wang, Yuxuan Rao, Liping Liu, and Jun Huan. Delta: Deep learning transfer using feature map with attention for convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(12):2935–2947, 2017.

Hank Liao. Speaker adaptation of context dependent deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7947–7951. IEEE, 2013.

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *International Conference on Learning Representations (ICLR)*, 2014a.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755, Zurich, September 2014b.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 21–37. Springer, 2016.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015a.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International Conference on Machine Learning (ICML)*, pages 97–105, 2015b.

Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5695–5703, 2016.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

Niki Martinel, Gian Luca Foresti, and Christian Micheloni. Wide-slice residual networks for food recognition. In *Winter Conference on Applications of Computer Vision (WACV)*, pages 567–576, 2018.

Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4040–4048, 2016.

John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *IEEE International Conference on Robotics and automation (ICRA)*, pages 4628–4635. IEEE, 2017.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3994–4003, 2016.

Gaspard Monge. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences*, pages 666–704, 1781.

Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 891–898, 2014.

Yurii E Nesterov. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547, 1983.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large mumber of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1520–1528, 2015.

Tsubasa Ochiai, Shigeki Matsuda, Xugang Lu, Chiori Hori, and Shigeru Katagiri. Speaker adaptive training using deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6349–6353. IEEE, 2014.

Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1717–1724, 2014.

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

Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2): 199–210, 2011.

George Papandreou, Iasonas Kokkinos, and Pierre-André Savalle. Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 390–399, 2015.

Ofir Pele and Michael Werman. Fast and robust earth mover's distances. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 460–467. IEEE, 2009.

Anastasia Pentina and Christoph H Lampert. Lifelong learning with non-iid tasks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1540–1548, 2015.

Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 724–732, 2016.

Gabriel Peyré, Marco Cuturi, and Justin Solomon. Gromov-wasserstein averaging of kernel and distance matrices. In *International Conference on Machine Learning (ICML)*, pages 2664–2672, 2016.

Gabriel Peyré and Marco Cuturi. Computational optimal transport. *arXiv preprint arXiv:1803.00567*, 2018.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017.

Xiaojuan Qi, Renjie Liao, Zhengzhe Liu, Raquel Urtasun, and Jiaya Jia. GeoNet: Geometric neural network for joint depth and surface normal estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 283–291, 2018.

Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 413–420, 2009.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and inter-

pretability. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6076–6085, 2017.

Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

Antoine Rolet, Marco Cuturi, and Gabriel Peyré. Fast dictionary learning with a smoothed wasserstein loss. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 630–638, 2016.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

Artem Rozantsev, Mathieu Salzmann, and Pascal Fua. Beyond sharing weights for deep domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 41(4):801–814, 2019.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Tim Salimans, Han Zhang, Alec Radford, and Dimitris Metaxas. Improving gans using optimal transport. In *International Conference on Learning Representations (ICLR)*, 2018.

Filippo Santambrogio. Optimal transport for applied mathematicians. *Birkäuser, NY*, 55:58–63, 2015.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems (NIPS)*, pages 2483–2493, 2018.

Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2014.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) workshop*, pages 806–813, 2014.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35(2):876–879, 1964.

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.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning (ICML)*, pages 1139–1147, 2013.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI Conference on Artificial Intelligence*, 2017.

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

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Tatiana Tommasi, Francesco Orabona, and Barbara Caputo. Learning categories from few examples with multi model knowledge transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 36(5):928–941, 2014.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4068–4076, 2015.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 595–604, 2015.

Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8769–8778, 2018.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1385–1392, 2013.

P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel S Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks. In *International Conference on Machine Learning (ICML)*, pages 793–802, 2018.

Yujia Xie, Xiangfeng Wang, Ruijia Wang, and Hongyuan Zha. A fast proximal point method for computing wasserstein distance. *arXiv preprint arXiv:1802.04307*, 2018.

Jun Yang, Rong Yan, and Alexander G Hauptmann. Adapting SVM classifiers to data with shifted distributions. In *IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 69–76, 2007.

Yongxin Yang and Timothy Hospedales. Deep multi-task representation learning: A tensor factorisation approach. In *International Conference on Learning Representations (ICLR)*, 2017.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems (NIPS)*, pages 3320–3328, 2014.

Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4353–4361, 2015.

Jure Zbontar, Yann LeCun, et al. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17 (1-32):2, 2016.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833, 2014.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017.

Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7151–7160, 2018.

Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2881–2890, 2017.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 633–641, 2017.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(6):1452–1464, 2018.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8697–8710, 2018.