DISS. ETH NO. XXXX

Towards Cost-Effective and Performance-Aware Vision Algorithms

A dissertation submitted to

ETH ZURICH

for the degree of

Doctor of Science (Dr. sc. ETH Zürich)

presented by

Dengxin Dai
Master of Science
born November 15, 1986
citizen of China

accepted on the recommendation of

Prof. Dr. Luc Van Gool, examiner
Prof. Dr. Bernt Schiele, co-examiner

2016

DEDICATION GOES HERE...

Abstract

Computer vision has leaped forward during the last decade, and now is able to recognize objects of thousands of categories and reconstruct 3D scenes at city- or world-scale. However, the field still has to find means to keep up with the exploration of the massive amounts of data being captured on a daily basis. This is mainly due to the lack of sufficient training annotations and the lack of computational resources. The thesis is dedicated to mitigate the problem.

Firstly, we elaborate two strategies to reduce the annotation costs in order to train vision algorithms: (1) developing smart annotation approaches for efficient, large-scale annotations; and (2) learning better feature representations using unlabeled data; We develop algorithms for the strategies and show - in the context of recognition tasks - that they are able to considerably reduce the annotation costs for the training of recognition algorithms.

Secondly, in addition to reducing annotation cost, we also examine how to reduce the computational cost associated with the training and testing of vision algorithms. This research has lead to two contributions: (1) two efficient solvers for linear and kernel SVM+, significant speeding up the training process of SVM+ to explore privileged information; and (2) a method to allow computationally cheap features to imitate alternative features that perform better but are computationally more expensive. The imitation significantly improves the performance of the cheap features while retaining their efficiency.

Thirdly, as images keep growing in size, vision algorithms need to be more intelligent and self-aware of their performance. To this aim, we have developed two approaches: (1) Performance Forecasting to cheaply predict the success of vision algorithms for particular samples, which can be used for better resource allocation; and (2) Scale-Aware Image Segmentation to re-organize image segmentation hierarchies to better couple hierarchy depth and segmentation scale. The two methods also show potentials in reducing computational time of consecutive vision algorithms.

Zusammenfassung

Zusammenfassung geht hier...

Acknowledgements

Acknowledgement goes here.

Contents

1

Introduction

A picture is worth a thousand of words, and often comes with a story that the photographer wanted to deliver. We probably still remember the moments when our grandparents were telling the stories behind the old pictures at our home, be it about birthday parties, be it about holidays. The purpose of taking pictures has been extended dramatically in the last decades, especially since the invention of digital cameras and the Internet. The inventions make it effortless ever to take pictures, store them, and share them with friends and families. While still used to remember such special moments of our daily life, most images and videos are taken now to support many other practical applications — cameras have been widely used for public security, for product control in factories, for product description in E-commerce, for real-world object reconstruction, for robot control, to name a few.

1.1 Motivation

While the quantity and the quality of images taken everyday are steadily increasing, it is hard to take its full advantage if the data is unorganized. In computer vision, algorithms are developed to tackle the fundamental problems of processing and understanding the visual data, in order to better suits the application context. The most extensively researched problems include image classification, object detection, action recognition, human pose estimation, object tracking in videos, 3D object reconstruction. With algorithms from these sub-fields, images and videos can then be converted into semantic information so that the data can be used to its full potential.

The last decade has witnessed a great progress in computer vision, which comes as an outcome of three factors: 1) introduction of large-scale human annotations, such as ImageNet [Deng et al., 2009] for image classification, PASCAL VOC [Everingham et al., 2010a] and MSCOCO [Lin et al., 2014b] for object detection, KITTI [Geiger et al., 2013] for road scene understanding; 2) development of sophisticated, statistical learning approaches including Support Vector Machines, Adaboost, Random Forests, and Deep

Neural Networks; and 3) accessibility to powerful computing infrastructures such as the GPUs and many software frameworks, such as CuNN, Caffe, Torch, and Theano. These advantages have made the field achieved multiple milestones: computer vision is able to recognize objects of thousands of categories in high accuracy [Krizhevsky et al., 2012a; Simonyan & Zisserman, 2015], transcribe images/videos directly into natural languages [Vinyals et al., 2015], and generate realistic images [Gregor et al., 2015].

Despite the great achievement, the field is still left behind by the exploration of visual data. Visual content is exploding – millions of videos are generated and consumed every second, from the footage of surveillance cameras to the over two billion images and videos uploaded daily to social platforms. However, intelligent and insightful understanding of these data is still far from being reached. We believe there are at least three reasons behind this: 1) visual annotation is expensive to obtain; 2) algorithms in computer vision are often computationally heavy; and 3) developed methods often are not performance-aware.

The first reason is evidenced by the fact that most of the benchmarks still only cover a small portion of the visual data that is available in the wild, and are collected under quite controlled scenarios. The speed of visual annotation cannot match that of visual data acquisition — taking pictures, storing and sharing them are effortless nowadays, while annotating visual data in the form that machine can learn on is tedious and very time-intensive. For example, YouTube has 100 hours of videos (100 million frames) updated every minute, while it required 19 man-years to label 1.2 million internet images [Deng et al., 2009]. The annotations are done in the form of bounding boxes, rather than in more dedicated forms such as full segmentation masks.

The second reason follows the fact that visual data is very high-dimensional, rendering the algorithms computationally heavy in both training and testing. For instance, training a state-of-the-art image classifier [Simonyan & Zisserman, 2015; He et al., 2015] can take days or weeks even with modern GPUs, computing the optical flow of a standard video dataset [Karpathy et al., 2014] can simply take several days, and computing the similarity among images of a large dataset can be very expensive as well [Gong et al., 2013]. This heavy computation hinders the community from quickly exploring new models, easily deploying the trained models to power-limited devices, and readily scaling the developed models.

The last one is due to the trend that the community is mostly focused on developing the next best method for task X, and rarely on improving the self-awareness of the methods, be it a measure of model uncertainty [Kendall et al., 2015; Kondermann et al., 2008], be it the usefulness to down-streamed vision tasks [Yao et al., 2011; Jain et al., 2015]. This performance blindness leads to ineffective solutions in many situations, such as the same amount of resource is allocated to every image regardless of its complexity (difficulty), and the advance made in all the sub-fields cannot be synergized effectively.

1.2 Contributions

The aforementioned problems deliver a strong need to reduce the annotation cost and computational cost of current computer vision methods, and to learn to predict their performance. This thesis is dedicated to provide a collection of methods to attack these problems from multiple perspectives.

First, we elaborate two strategies to reduce the annotation cost:

- Developing efficient visual annotation approaches. A natural and efficient annotation method is developed for object recognition by letting annotators speak. Since drawing scribbles and speaking are very natural to human, our method unleashes the expressive ability of annotators and solves the *what* and *where* problems of object annotation both at the same time, leading to an approach which draws a good trade-off between recognition accuracy and annotation cost.
- Learning feature representation with unlabeled data. A new method is developed to learn a new feature representation on top of standard feature representations. The leaning takes advantage of discriminative learning and ensemble learning, and is able to generate new features specifically tailored to the data at hand.

Secondly, we examine how to reduce the computational cost associated with the training and testing of vision algorithms:

- Developing efficient training algorithms to the standard approach SVM+. SVM+ has shown excellent performance in visual recognition tasks for exploiting privileged information in the training data. We propose two efficient algorithms for solving the linear and kernel SVM+. Experiments show that our proposed algorithms achieve significant speed-ups to the state-of-the-art solvers for SVM+.
- Proposing Metric Imitation (MI), a method to allow computationally cheap features to imitate alternative features which perform better but are computationally more expensive. The leaned transformation significantly boost the performance of cheap features while retaining their efficiency.

Lastly, we investigate performance-aware vision algorithms by making the following contributions:

- Predicting how likely vision algorithms succeed on particular samples. It is true for every vision task that not all images are equally difficulty. We examine how to learn this on the task of example-based texture synthesis (ETS). ETS has been widely used to generate high quality textures of desired sizes from a small example.

However, not all textures are equally well reproducible that way. We predict how synthesizable a particular texture is by ETS and find that texture synthesizability can be learned and predicted efficiently.

- Learning scale-aware image segmentation. Hierarchical image segmentation provides segmentation at different scales in a single tree-like structure. However, they are not aware of the scale information of the regions in them. As such, one might need to work on many different levels of the hierarchy to find the objects in the scene. This work predicts the scales of the regions to modify their depth in the tree to better couple tree depth and region scale. The output of our method is an improved hierarchy, which improves the quality of the hierarchical segmentation representations.

- Evaluating the usefulness of image super-resolution methods to other computer vision tasks. Although it might be believed that image super-resolution is helpful for other vision tasks, this work has formalized the conception and conducted quantitative evaluation. The work can serve as an inspiration for the community to evaluate image super-resolution with respect to the helpfulness to other vision tasks, and to apply it as a pre-processing component if the input images are of low-resolution.

1.3 Organization

Since extensive research has been done in similar spirit, the thesis begins with Chapter 2 examining related work in a broad context. Our developed approaches are presented in Chapters 3 – 9. They are written to be generally self-contained and can be read independently. Finally, Chapter 10 concludes this thesis. A detailed overview of the remaining chapters follows:

In Chapter 2, *Related Work*, we provide a short literature overview of previous art in the direction of reducing the annotation cost and computational cost of vision algorithms and towards self-aware algorithms.

In Chapter 3, *Efficient Visual Annotation by Speech Recognition*, we present our efficient visual annotation approach Draw&Tell and show its efficiency in the context of semantic image segmentation. In order to solve the *what* and *where* problems in visual annotation both at the same time, we let annotators speak the name of the object while they draw strokes on it. The speech is recognized by a speech recognition engine specifically trained for the purpose, and an extension to the fully convolutional neural network is made to learn from the stroke-based 'coarse' annotation. The approach draws a good trade-off between recognition accuracy and annotation cost. The work in this chapter was originally presented in [Dai et al., 2016a].

In Chapter 4, *Representation Learning with Unlabeled Data*, we present our motivation and method of learning a new feature representation with unlabeled data. We then evaluate the method in the context of semi-supervised image classification and image clustering. This work was originally presented in [Dai et al., 2012a] and in [Dai & Van Gool, 2013a].

In Chapter 5, *Fast Training Algorithms for SVM+*, we present two efficient algorithms for training the linear and kernel SVM+. New problems with fewer constraints are formulated in the dual domain, making the problem solvable efficiently by the SMO algorithm of one-class SVM. Experiments show that our proposed algorithms are significantly faster than the the state-of-the-art solvers for SVM+. This work was originally presented in [Li et al., 2016].

In Chapter 6, *Metric Imitation for Efficient Distance Computation*, we present a method called Metric Imitation (MI) to efficiently compute the distance among images. MI learns a transformation to cheap features so that the L2 distance of the transformed features can approximate L2 distance of better-performing features which are more computationally expensive to compute. The method is evaluated on multiple vision tasks. The method was originally presented in [Dai et al., 2015a].

In Chapter 7, *Performance Prediction: Succeed or Fail?*, we present our approach of predicting the success of example-based texture synthesis. To this aim, we collected a dataset with $21,302$ annotated textures and annotated them according to the synthesizability — the quality of synthesized results by texture synthesis methods. A set of relevant features are then defined to regress the value of synthesizability. Extensive experiments show that texture synthesizability is learnable. This work was originally presented in [Dai et al., 2014].

In Chapter **??**, *Performance Prediction: Under-, Properly-, or Over-Processed?*, we present a method to predict the scale of image segments relative to the corresponding objects, and to then apply the prediction to re-align the results from general hierarchical image segmentation so that the depth in the tree structure and the scale of the regions is better coupled. The output of our method is an improved hierarchy, which improves the quality of the hierarchical segmentation representations. The work was originally presented in [Chen et al., 2016].

In Chapter 9, *Performance Evaluation: Helpful for Other Tasks?*, we evaluate the usefulness of image super-resolution methods to other computer vision tasks. Sixes state-of-the-art computer vision algorithms are evaluated on four popular computer vision tasks: boundary detection, semantic image segmentation, digit recognition, and scene recognition. The work confirms that image super-resolution is helpful for other vision tasks when the state-of-the-art approaches are used. The work was originally presented in [Dai et al., 2016b].

2

Related Work

The community has made great efforts in reducing the annotation cost and computational cost of vision algorithms by using a diverse set of techniques. There is also a large body of literature working towards self-aware vision algorithms. This section summarizes the related topics in a broad context. Related work to each of our specific work is discussed in the corresponding chapter.

2.1 Reducing Annotation Cost

Datasets play a critical role in computer vision. They qualitatively 'define' the learning tasks and guide research directions, which has been proved multiple times in the history of computer vision [Baker et al., 2011; Everingham et al., 2010a; Deng et al., 2009]. We limit ourselves to annotations for object recognition. Training annotations for object recognition often come as bounding boxes or full segmentation masks (c.f. Fig.**??**). The most popular ones fall into this category: CamVid [Brostow et al., 2009] and Cityscapes [Cordts et al., 2016] for urban scenes, NYU [Silberman et al., 2012] for indoor scenes, PASCAL [Everingham et al., 2010a] and COCO [Lin et al., 2014b] for general objects, and PASCAL-Context [Mottaghi et al., 2014] for objects in context. Creating datasets for object recognition is very expensive even with excellent annotation tools [Bell et al., 2013; Russell et al., 2008a]. As a result, methods were proposed to reduce the cost. For instance, [Deng et al., 2014; Lin et al., 2014b] exploit the hierarchical structures of object classes to reduce annotation space. Other popular techniques can be roughly grouped into the following categories (with overlaps).

Weakly Supervised Learning

As stated, visual annotation is time-consuming. Many works have developed algorithms to learn from weakly annotated training data. The following are the outstanding examples. [Prest et al., 2012] learns object detectors from weakly labeled videos. [Pathak et al., 2015b; Papandreou et al., 2015] trains convolutional neural networks for semantic image segmentation with image-level annotations. [Khoreva et al., 2016] trains boundary

detector with bounding-box annotations. [Bilen & Vedaldi, 2016] trains object detectors with the training data of image classification. This stream of work proves that with careful design of algorithms, weak supervision is able to yield quite competitive performance as the full supervision. Weak supervision consumes far less effort, which makes the training more affordable. Recent datasets such as Cityscapes [Cordts et al., 2016] start providing annotations of different quality. In the same vein, our work in Chapter 3 tries to learn from weak supervision for semantic image segmentation.

Transfer Learning

Transfer learning is to transfer knowledge learned from one task to another task, or from one domain to another domain, when training data is scarce for the latter scenario. Successful applications in vision include knowledge transfer from videos to images [Kulis et al., 2011; Gopalan et al., 2011; Fernando et al., 2013], knowledge transfer from known classes to unseen classes [Lampert et al., 2009], supervision transfer [Gupta et al., 2015] from annotated RGB images to other data modalities such as depth and flow, and supervision transfer [Hoffman et al., 2014] from classification task to detection task. Transfer learning has gained great success recently in object recognition and has become the standard procedure in training (fine-tuning) deep neural networks [Girshick et al., 2014a; Long et al., 2015]. It successfully lifts the requirement of large training set for the task at hand, and considerably reduces the training time. Our work in Chapter 6 can be considered as a special case of transfer learning by transferring learned manifold from one domain to another.

Semi-supervised Learning

Semi-supervised learning (SSL) aims at enhancing the performance of recognition systems by exploiting an additional set of unlabeled data. SSL is especially helpful when the labeled training data is limited. Due to its great practical value, SSL has a rich literature [Chapelle et al., 2006; Zhu & Goldberg, 2009]. The research in this vein can be classified into four groups based on their underlying techniques: (1) Self-training scheme [Blum & Mitchell, 1998; Guillaumin et al., 2010; Shrivastava et al., 2012], where the system iterates between training recognition models with current 'labeled' training data and augmenting the training set by adding its highly confident predictions in the set of unlabeled data; (2) Label propagation [Zhu et al., 2003; Zhou et al., 2004; Belkin et al., 2006; Fergus et al., 2009; Ebert et al., 2012], where a graph is defined with nodes representing data examples and edges reflecting their similarities, and label information propagates over the graph; (3) Classifier regularization [Joachims, 1999; Bennett & Demiriz, 1998; Leistner et al., 2009; Weston et al., 2012], by enforcing the boundaries to pass through regions with a low density of data samples. Our work in Chapter 4 bases on the assumption of manifold smoothness to learn a new feature representation, and shows pleasant performance on semi-supervised image classification.

Active Learning & Human in the Loop

Active learning or recently Human in the Loop lets human and the computer work collectively to train good vision algorithms. Smart policy needs to be designed or learned in

order to take full advantage of machine systems and to minimize the human effort to machine's uncertainties. The policy varies significantly from task to task: from suggesting the most informative samples to annotate [Vijayanarasimhan & Grauman, 2012; Freytag et al., 2014], to how many object to display [Yao et al., 2012], and to what questions to ask [Russakovsky et al., 2015b; Papadopoulos et al., 2016]. The great success of this technique in computer vision is evidenced by the large amount of academic publications, mainly on image recognition [Joshi et al., 2009a; Branson et al., 2010; Collins et al., 2008], semantic image segmentation [Vezhnevets et al., 2012], and object detection [Russakovsky et al., 2015b; Papadopoulos et al., 2016].

Supervision from Multimodal Data & Web Data
In addition to human annotated training data, other sources of supervision have been exploited to train vision algorithms as well, such as text from web pages or newspapers [Berg et al., 2004; Gupta & Davis, 2008], eye-tracking data [Papadopoulos et al., 2014]. Webly-supervised learning [Chen et al., 2013; Chen & Gupta, 2015a; Divvala et al., 2014] has gained extensive attention in the recent years, where visual recognizer is trained automatically by the images returned by image search engines such as Google, Flickers, and Bing. These works show the great potentials of scaling visual recognition to billion-sized scale, without using human annotations.

Unsupervised Feature Learning
Another group of work in the spirit of learning with limited annotations aims to learn middle- or high-level image representation in an unsupervised manner. The supervision often comes from intelligent exploration of prior knowledge or common sense knowledge. For instance [Coates et al., 2011; Dosovitskiy et al., 2014] generates surrogate classes by clustering or performing transformation to local patches, [Doersch et al., 2015] employees the spatial relationships of image windows in an image, [Wang & Gupta, 2015] exploits the tracking results of objects in videos, and [Agrawal et al., 2015] leverages the ego-motion of cameras. These systems are all able to learn good feature representations without using human annotations. Our work in Chapter 4 learns new feature representations by exploiting the assumption of manifold smoothness from unlabeled data, which is similar to [Coates et al., 2011; Dosovitskiy et al., 2014].

2.2 Reducing Computational Cost

In addition to reducing annotation cost, we also investigate how to reduce the computational cost of vision algorithms. There are numerous great techniques towards efficient vision algorithms, especially those targeted for mobile and robotic applications. A complete overview of the topic is beyond the scope of this thesis, so we only summarize a group that is mostly related to our work Metric Imitation in Chapter 6.

Model Compression, Imitation and Distillation
Often, we are faced with a dilemma where the best performing model is too slow and too large, and the efficient alternatives are not as accurate. This is exactly the place where model compression [Bucilua et al., 2006] comes into play. Model compression is often given other names such as distillation or imitation [Hinton et al., 2014; Romero et al., 2014; Rusu et al., 2016]. The high-level idea is to learn a fast and compact model to approximate or imitate the functional behaviors of a better-performing model which is slow to run. [Bucilua et al., 2006] successfully compresses ensemble classifiers 'into' a compact neural network, and [Hinton et al., 2014; Romero et al., 2014; Rusu et al., 2016] distills the knowledge learned by a deeper or wider neural networks into a lighter neural networks. [Xu et al., 2015b] proves that it is also possible to imitate the effects of edge-aware filters designed with domain knowledge by an efficient convolutional neural network. The results of these works suggest that models of different complexity are needed for learning and evaluation if efficiency is concerned. Complex models are required to extract knowledge form noisy observations; Compact models then distill the learned knowledge from the trained complex models for efficient evaluation. Our work Metric Imitation shares the spirit.

2.3 Self-aware Vision Algorithms

Predicting Model Confidence, Uncertainty or Failure
Performance-blind vision algorithms can be disastrous, as they can fail saliently without even throwing a warning. As such, this research topic is increasingly gaining more attention, though not very popular yet. Notable examples of learning model uncertainty or confidence include [Kendall et al., 2015] for semantic image segmentation, [Kondermann et al., 2008; Aodha et al., 2013] for optical flow, [Kopf et al., 2012] for image completion, [Hartmann et al., 2014] for patch matching, [Park & Yoon, 2015] for stereo, and [Zhang et al., 2014a] for multiple applications such as vanishing line estimation and memorability. Our work texture synthesizability (Chapter 7) enriches this repository by adding another example texture synthesis. Performance-aware algorithms carry other benefits as well. For instance, it can speed up down-streamed algorithms by adaptively allocating computing resource based on the complexity of image samples. Cascading [Viola & Jones, 2001; Felzenszwalb et al., 2010] and active inference [Liu & He, 2015] serve as standing examples.

The Usefulness of Task X for Task Y
Computer vision is a very challgening task, and research has to be conducted in many sub-fields. These sub-fields are intertwined intrinsically, and should benefit from each other in principle. However, this does not always happen in practice, because building a system of one task on top of another task introduces extra modes of errors as well. Thus, it is necessary for the community to regularly evaluate whether sub-field X has

advanced to a level that sub-field Y can generally benefit from it if integrated. Excellent research has been continuously done in this direction. For instance, [Hoiem et al., 2008] proves that surface orientation estimation is already good enough to be helpful in guiding object detection; [Yao et al., 2011] confirms that pose estimation is able to help action recognition; [Jain et al., 2015] analyzes how object detection helps action recognition when the state-of-the-art detectors run over 15000 object classes; [Martinovic et al., 2015] shows that 3D reconstruction of regular buildings is already accurate enough to perform object recognition directly on it. Our work in Chapter 9 consolidates that image super-resolution by state-of-the-art approaches is helpful for general vision tasks.

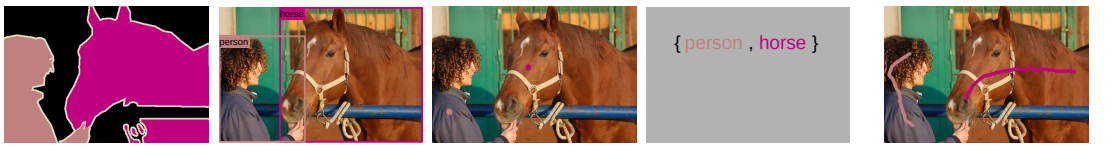
3

Efficient Visual Annotation with Speech Recognition

3.1 Introduction

Tremendous progress has been made during the last few years in object detection and semantic image segmentation due to (1) the success of training deep convolutional neural networks (CNNs) [Krizhevsky et al., 2012b; Donahue et al., 2014a; Simonyan & Zisserman, 2015] on large classification datasets and (2) the flexibility of transferring CNN models pre-trained for image classification to the task of object detection and semantic segmentation where not as much training data is available [Girshick et al., 2014a; Long et al., 2015; Chen et al., 2015a]. Model transfer, often called fine-tuning, partially lifts the strong need for large training sets of CNNs. Yet, the fact remains that CNNs *intrinsically* call for large-scale training data to unleash their learning capacity. As such, we can expect that the limit of CNNs for the tasks is far from reached and that one of the obstacles is the cost of obtaining training samples.

Obtaining training data for object detection or semantic image segmentation consists of two main jobs: answering the *what* and *where* questions. *What* is to annotate what objects are present and *where* is to indicate their locations in the image. As to *what*, annotators usually need to choose class names from some form of menus with the mouse and/or keyboard [Bell et al., 2013]. The procedure can be costly especially when the number of classes considered is large. Imagine the amount of work for annotators to find the class name out of a list of hundreds of names for every object they annotate, not even to mention attributes, views, etc. In other systems [Bearman et al., 2015; Lin et al., 2014b], binary questions ask whether object classes are present, which is also very expensive when the number of classes is large, though exploiting the hierarchy of classes may reduce the cost [Deng et al., 2014; Lin et al., 2014b]. As to *where*, various forms of annotations have been used for training: from the full segmentation masks [Girshick et al., 2014a; Chen et al., 2015a; Zheng et al., 2015], over bounding boxes [Xu et al., 2015a; Dai et al., 2015e; Pathak et al., 2015a] or just single points [Bearman et al., 2015], to nothing spatial at all



(a) mask [ ] (b) b-boxe [ ] (c) point [ ] (d) keyword [ ] **(e) scribble** [+]

Figure 3.1*: A comparison of our annotation (e) to other forms of annotations for semantic segmentation, ranging from (a) full segmentation masks, to (b) bounding boxes, to (c) single points, and to (d) image-level keywords. The devices to obtain these annotations are shown as well: others only use the mouse (+keyboard), while ours also uses voice input through a microphone.*

(image-level annotation) [Verbeek & Triggs, 2007; Papandreou et al., 2015]. As can be seen in Figure 3.1, the more specific the annotation, the more informative it is, but also the more expensive to obtain.

In this work , we present a novel annotation method, which strikes a balance between annotation cost and informativeness provided. In particular, annotators are asked to draw scribbles (strokes) on objects of interest, while saying aloud their class names (attribute if necessary). The scribbles are recorded to solve the *where* problem; the speech is recognized by a specially trained recognition engine to solve the *what* problem. In order to further increase the recognition accuracy of the scribbles (associating to object names), we integrate the speech recognition with a webly-supervised object recognition. The object recognition system is trained with images retrieved from image searching engine directly, and is applied to the image region where the scribble is drawn. An example of the annotation by our method is shown in Figure 3.1, along with that of other methods.

The method is inspired by two observations: (i) *what* and *where* are handled separately in previous methods. For instance, in [Bell et al., 2013] annotators first mask out a region and then assign a label to it; and in [Lin et al., 2014b] annotators label the presence of objects in the first round, followed by positioning them in the second round. This separation is unnecessary and introduces extra cost, because the two problems are interdependent and solved together by the annotators vision. It seems the separation is due to the fact that only a single mode of input devices was used in the previous methods, which is the mouse (+keyboard). We lift this restriction by including the speech channel, which allows annotators to communicate with the computer more naturally and efficiently. (ii) Drawing scribbles is another natural and efficient ability, and has been widely used for interactive image segmentation. But to the best of our knowledge, it has not been applied to create training data for semantic image segmentation. We demonstrate that combining drawing and speaking can tackle both the *what* and *where* problems, leading to an efficient annotation method for semantic image segmentation. We call the method **Draw&Tell**.

Having obtained the annotations, we convert them to soft confidence maps of the corresponding classes by combining interactive image segmentation and ensemble learning.

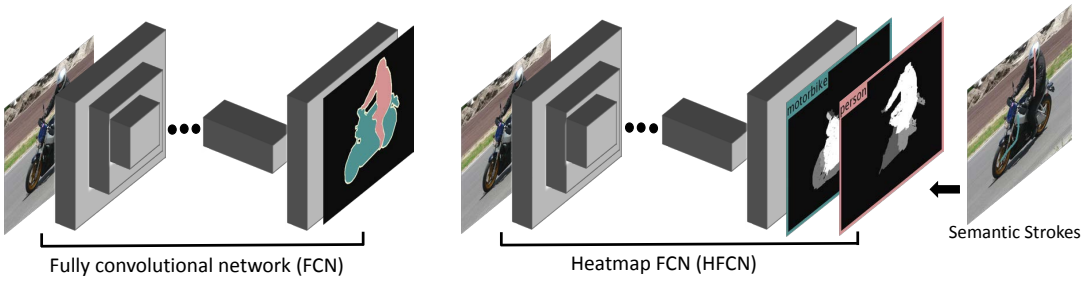


***Figure 3.2**: The framework of the standard fully convolutional network (FCN) [Long et al., 2015] and our heatmap-based FCN.*

We call these soft confidence maps semantic heatmaps. Finally, we extend the standard CNNs model [Long et al., 2015] to accommodate the soft semantic heatmaps as training data rather than the standard crisp labels. The pipeline of the method is shown in Figure 3.2 (right hand side), which also shows the standard fully convolutional network [Long et al., 2015] for ease of comparison. We show in experiments (i) that our annotation method is 11 times faster than pixel-wise annotation, and also faster than conventional image-level annotation; (ii) that our adapted CNNs trained with the annotations yields significantly better results than the CNNs trained with image-level training data, and yields results comparable with those of the CNNs trained with pixel-wise annotations; and (iii) that under the same annotation budget, our annotations, combined with our learning method, yield better results than the conventional precise annotations.

Our contributions are mainly: (i) a new method which combines webly-supervised vision and automatic speech recognition to efficiently create training data for semantic image segmentation, and (ii) a method to train CNNs with scribble-based training data, by converting scribbles to soft heatmaps and extending the standard neural networks. Introducing speech recognition into visual annotation is novel and our method can be used for other vision tasks as well, such as annotations for object detection, part detection, attribute detection and view estimation. The integration of webly-supervised object recognition and speech recognition can also serve as an example of combing vision and speech.

This chapter is organized as follows. Section 3.2 presents related work. Section 3.3 details the annotation method, followed by Section 3.4 which is devoted to the segmentation method. Section 3.5 then reports on the experiments and Section 3.6 concludes the paper.

3.2 Related Work

Previous work related to ours mainly falls into two groups: semantic image segmentation and integration of vision and language (speech).

3.2.1 Semantic Image Segmentation

Methods: There is a rich literature of semantic image segmentation (SIS) [Shotton et al., 2009; Krähenbühl & Koltun, 2011], and the field has made tremendous progress since CNNs were applied. The seminal R-CNN [Girshick et al., 2014a] and the follow-up systems [Long et al., 2015; Chen et al., 2015a; Zheng et al., 2015] yield significantly better performance than previous methods. As noted above, SIS through CNNs has probably not come to full fruition yet due to the small size of the existing training sets. Several successful attempts have been made to use weaker supervision in order to reduce the annotation cost. For instance, [Pathak et al., 2015b] exploits multiple instance learning to train FCN [Long et al., 2015] with image-level annotations. [Papandreou et al., 2015; Dai et al., 2015e; Pathak et al., 2015a] leverage the power of object proposals to train FCN with object bounding-boxes. [Bearman et al., 2015] presents a system to train FCN with point annotation – objects are indicated by single points. In the same vein our work tries to train FCN with annotations that are efficient to obtain.

Supervision: Datasets play a critical role in computer vision. They qualitatively 'define' the learning tasks and guide research directions. Training annotations for SIS often come as full segmentation masks (c.f. Figure 3.1). The most popular ones fall into this category: CamVid [Brostow et al., 2009] for outdoor scenes, NYU [Silberman et al., 2012] for indoor scenes, PASCAL [Everingham et al., 2010a] and COCO [Lin et al., 2014b] for general objects, and PASCAL-Context [Mottaghi et al., 2014] for objects in context. Creating datasets for SIS is very expensive even with excellent annotation tools [Bell et al., 2013; Russell et al., 2008a]. As a result, methods were proposed to reduce the cost. For instance, [Deng et al., 2014; Lin et al., 2014b] exploit the hierarchical structures of object classes to reduce annotation space. [Vijayanarasimhan & Grauman, 2012; Vezhnevets et al., 2012] exploit active learning techniques to suggest the most informative samples to annotate under a budget. [Papadopoulos et al., 2014] employs eye tracking systems to help create training data for object detection. Our proposal is to exploit speech recognition to help dataset creation for SIS.

3.2.2 Integration of Vision and Language (Speech)

Research interest in integrating vision and language has increased recently, e.g. for image/video caption generation [Vinyals et al., 2015; Guadarrama et al., 2013; Thomason et al., 2015; Karpathy & Fei-Fei, 2015; Kulkarni et al., 2013]. The objective is to learn from a corpus of sentences and images / video snippets to generate meaningful descriptions for new images. One of the main research topics is the alignment of representations from the two different domains: vision and language. In terms of language&vision understanding, our goal is more conservative, because the words we handle are restrictive and vision and language are aligned with human help. However, our purpose of the integration is different.

Speech-driven interfaces are going through a renaissance, with popular commercial products like Apple's Siri. The most relevant to our work are those integrating speech and vision. Pixeltone [Laput et al., 2013] and Image spirit [Cheng et al., 2014] are examples that use voice commands to guide image processing and semantic segmentation. The difference between image spirit and our work is that they use voice commands to refine labeling results and we use them to collect training data. Also, image spirit uses voice commands alone, while our method combines speech input with mouse interactions. There also is academic work [Srihari & Zhang, 2000; Kalashnikov et al., 2011; Hazen et al., 2007; Hoogs et al., 2003] and an app [smi, 2015] that use speech to provide image descriptions. Again, our purpose is very different.

3.3 Speech-based Annotation

3.3.1 The Draw & Tell Annotation Tool

This section describes our annotation method, which consists of two parts: drawing scribbles on the objects and telling the system their names. Drawing scribbles is straightforward and we follow popular interactive image segmentation (IIS) methods [Gulshan et al., 2010] to record the mouse tracks. As to the speech recognition, we draw on the state-of-the-art, end-to-end speech recognition systems [Graves & Jaitly, 2014], trained with bidirectional recurrent neural networks under the connectionist temporal classification loss function. The systems successfully avoid the significant human effort in creating the pronunciation dictionary. The characters are then decoded to words with a dictionary of desirable class names by the CTC Beam Search algorithm in [Graves & Jaitly, 2014]. Improving such a system itself is challenging and beyond the scope of this paper, thus we choose to treat the system as a black box and operate on an n-best list of possible object names for each utterance.

Speech recognition is a complex research area in its own right, and there are still problems in recognizing natural speech with high accuracy [Saini & Kaur, 2013]. We, however, want to use it to simplify the task of annotation, for which high accuracy is required. Fortunately, some constraints can be imposed in our case, which our annotation task naturally fulfills. The constraints and solutions are:

- Constraining the vocabulary and syntax of the utterances to ensure robust speech recognition. For our annotation, the vocabulary is restricted to the names of the objects of interest, and we can also instruct annotators to follow a specific syntax.
- Synchronizing the speech input with mouse input. We instruct the annotator to only say the object names while they draw the scribble on the corresponding object, not at any other time. The speech recognition engine uses this synchronization to better identify speeches relevant to the object being annotated.

Figure 3.3: *The interface of Draw&Tell. Annotators can draw scribbles on objects of interest and speak their names in the meanwhile. The names of all the classes of interest are shown at the bottom for reference. The names of the object are obtained by speech recognition and are shown right after the drawing of the scribbles. Annotators can correct the result if it is wrong.*

- Re-ranking the n-best words by integrating information from visual object recognition, as analyzing the visual content in the drawing area provides complementary information. We choose to learn the recognizer in a webly-supervised manner, where images retrieved from image search engines such as Google and Bing are used directly for the training. This integration is also interesting in the wider context of combining vision and speech. Section 3.3.2 elaborates the method.

As a fall-back solution, Draw&Tell provides ways to permit correction of the recognition errors, by either repeating the operation (drawing + speaking) for the same object; or typing the object name directly. The annotator will be asked to review the name list if the system fails multiple times to recognize the speech, mainly because the spoken words are

out of the dictionary. Our implementation is built on top of the open source code of [Maas et al., 2015]. The interface of Draw&Tell is shown in Figure 3.3.

3.3.2 Integration with Webly Supervised Object Recognition

As discussed in the previous section, speech recognition still has problem for high-precision applications like ours. Thus, we propose to re-rank the n-best words (object names) from the speech recognition engine by using visual information. The idea is straightforward as follows: once a scribble is drawn, a corresponding image region is cropped and fed into an object recognition system for visual recognition. The region is generated by choosing an object proposal out of 500 candidates yielded by edge-box [Zitnick & Doll, 2014]: (i) obtaining the enclosing bounding-box of the scribble, and (ii) choosing an object proposal with which the bounding-box has the highest intersection-over-union score. The recognition scores are then fused with the scores from speech recognition to obtain the name of the object.

The object recognition system can be trained with samples collected by conventional annotation methods, if available. We here explore the potential of training it with web images, due to (i) an enormous amount of visual data is online to use for free; and (ii) noisy recognition (up to some level) is acceptable here as it is only used to complement speech recognition, which itself already performs well for a small dictionary of pre-defined words, though not perfect.

We follow the footprint of [Chen & Gupta, 2015b] to train a webly-supervised RCNN [Girshick et al., 2014a] model for object recognition. Our RCNN is trained with a classification loss on images returned by Google directly – images returned by the same keyword are taken as of the same class. The rationale is that the top images returned by Google mostly come with a clean background and a single centered object of the query class. If needed, more sophisticated techniques can be used to prepare the training data, such as seeds-based co-segmentation [Chen & Gupta, 2015b]. In this work, we downloaded 2500 images for each class, and fine-tuned the VGG Net [Simonyan & Zisserman, 2015] pre-trained on ImageNet [Deng et al., 2009]. The integration of speech recognition and webly-supervised object recognition is done by simply fusing their posterior probabilities:

$$P(y|\mathbf{v},\mathbf{x}) = \sigma P(y|\mathbf{v}) + (1-\sigma)P(y|\mathbf{x}) \tag{3.1}$$

where $y \in \{1, ..., K\}$ denotes the object names with K the number of classes, $\mathbf{v}$ the voice represented as spectrograms, $\mathbf{x}$ the cropped image region, and σ a scalar value to balance the two terms. σ is set to 0.7 empirically in this work to put more weight on speech recognition as the object recognizer is trained with noisy data.

PASCAL VOC				MSCOCO			
PocketSphnix	Ours			PocketSphnix	Ours		
	Speech	Web	Combined		Speech	Web	Combined
71.2	77.4	51.0	92.1	57.3	60.0	39.6	75.8

***Table 3.1**: Recognition accuracy (%) of object names, where speech means our method without the help of the object recognition, web denotes only using the object recognition, and combined stands for our final method.*

3.3.3 Annotation Results

Three results are reported: (i) recognition accuracy of object names, (ii) annotation accuracy of the scribbles, and (iii) annotation speed of our method.

Recognition accuracy for object names: Our method is evaluated with a comparison to PocketSphinx [sph, a], a standard speech recognition engine. For fair comparison, we retrained a new dictionary and language model [1] for PocketSphinx as well so that it focuses on the object names of interest. The evaluation is conducted on the box annotation of PASCAL VOC 2012 [Everingham et al., 2010a] and that of MSCOCO dataset [Lin et al., 2014b]. An annotated object in the image is highlighted and its name is displayed. Annotators are asked to draw scribbles on the object while speak aloud the name displayed. Examples are shown in the supplementary material. The recognition result will be compared to the ground truth and the average accuracy is reported. 20 object instances are sampled for each object classes, leading to $400 = 20 \times 20$ objects for PASCAL VOC and $1760 = 20 \times 88$ objects for MSCOCO. Three annotators (graduate students) are asked to perform the annotation, and their results are averaged. Table 3.1 shows the results.

The results shows that our method performs significantly better than the pure speech recognition methods, namely the HMM-GMM based method PocketSphnix, and the bidirectional RNN based method [Graves & Jaitly, 2014; Maas et al., 2015]. Webly-supervised object recognition provides reasonably good results, but itself alone is still not enough to be used for the annotation task. By combining the strength of vision and speech, our method yields excellent results for the 20 classes of PASCAL VOC, and quite good results for the 88 classes of MSCOCO. The results suggest that our method is accurate enough to be used directly for annotation tasks of around 20 classes. For more classes like that in MSCOCO, people often annotate them in a hierarchical manner [Lin et al., 2014b], leading to annotation tasks with smaller number of classes, which makes our method applicable as well. In the rest of this work, we will mainly evaluate our annotation method on the 20 classes of PASCAL VOC.

Annotation Accuracy: We report and analyze our annotation results on PASCAL VOC 2012 [Everingham et al., 2010a] here. We annotated the 20 object classes of PASCAL

[1] They are trained by simply uploading the text corpus to CMUSphinx's web service [sph, b].

Anno-1.5				Anno-3			
object		pixel		object		pixel	
false neg.	false pos.	background	other obj.	false neg.	false pos.	background	other obj.
4.7	3.8	1.7	5.1	4.5	4.0	1.0	3.8

***Table 3.2**: Errors (%) of the annotation: false positives and false negatives at object-level, and the percentages of pixels drawing to the background and objects of other classes for the scribbles which are 'correct' at object-level.*

VOC for $12,031$ images, including all images in the training and validation set of the PASCAL VOC 2012 segmentation benchmark and all images in the augmented dataset from [Hariharan et al., 2011]. The scribbles for the background will be automatically sampled in order to reduce the annotation cost. Two annotators annotated the data, independently, with the same goal of drawing scribbles as much as possible on the objects, but on different budgets: the first one has 1.5 seconds per object, while the second one has 3 seconds. They will be referred as Anno-1.5 and Anno-3. Exemplar annotations are shown in the supplementary material.

The annotation results are listed in Table 3.2. Four types of errors are reported: two at object-level and two at pixel-level. At object-level, we assign the scribbles to their closest object masks (by the number of pixels shared) and check the consistency of their labels. False positives (FP) and false negatives (FN) are reported, which are fairly few. Also, we must be aware that objects of very small size may be confusing to annotators in terms of whether they need to be annotated. These cases contributed the most of these FP and FN cases. Scribbles that are 'correct' at object-level were evaluated for their accuracy at pixel-level. We specify two errors: the percentage of pixels drawn onto background as well as onto other objects. They are actually very small, which means for the correctly detected objects human annotators can spot their position and outline very accurately. Overall, the annotations are precise, because (i) scribbles are flexible, easily adaptable to different object layouts and (ii) drawing scribbles is very natural to human.

Annotation Speed: We compare the speed of our annotation method to four other popular annotation methods (c.f. Figure 3.1): full segmentation masks, bounding boxes, singular points on objects, and image-level keywords. Because all these methods need to loop over all classes to solve the *what* problem, the minimum time - browsing time - is constant for all. According to [Bearman et al., 2015], the time for browsing one image for one class is 1 second, which is consistent with what we found in experiments. The total browsing time per image is 20.0 seconds, which is also the time for image-level annotation. The time for other forms of annotations is the sum of this 20.0 seconds and the time for annotating 2.8 (on average) objects in each image. Point-based annotation [Bearman et al., 2015] costs 3.2 seconds for clicking points on the objects, resulting in 23.2 seconds in total.

For the time of drawing one bounding box, different numbers are reported, varying from 7 to 25.5 seconds [Russakovsky et al., 2015b; Dutt Jain & Grauman, 2013], probably

Mask	Bounding-Box	Point	Keyword	Ours (Speech-based Scribble)	
				Anno-1.5	Anno-3
104.8	39.9	23.2	20.0	6.6	11.1

***Table 3.3**: The annotation speed (seconds per image) of all methods, measured on PASCAL VOC.*

because the types of objects being handled and the quality of the annotation are different. We experimented with images from PASCAL VOC 2012 and drew bounding boxes for 200 randomly chosen objects. The annotation time per bounding box is 7.1 seconds on average, which is quite efficient compared to the numbers reported in the literature. Thus the total time for bounding-box based annotation per image is 39.9 $(20.0 + 2.8 \cdot 7.1)$ seconds. Similarly, we annotated the masks for 200 randomly sampled objects using the annotation tool developed in [Bell et al., 2013]. Annotating each mask takes 30.3 seconds on average. Thus, full-mask based annotation time is 104.8 $(20.0 + 2.8 \cdot 30.3)$ seconds.

For our annotations, we allocate 2.0 seconds for browsing the image, which was found sufficient in the annotation. Drawing one scribble takes 1.5 and 3.0 seconds for Anno-1.5 and Anno-3. The recognition error rate of speech recognition is about 8% for our task, which contributes to the correction time. Putting all together, the annotation time is 6.6 $(2 + 2.8 \cdot 1.5 \cdot (1 + 0.08 + 0.08 \cdot 0.08 + ...))$ seconds per image for Anno-1.5, and 11.1 seconds for Anno-3. All the numbers are listed in Table 3.3. According to the numbers, our annotation is the fastest one (faster than the image-level annotation as well), due to the help by the integration of speech recognition and webly-supervised object recognition, which lets annotators solve the *what* and *where* tasks of object annotation both at the same time. Some of these numbers are quite *rough* estimations, but they reflect the cost to a reasonable level of accuracy.

3.4 Semantic Image Segmentation

We follow recent methods [Girshick et al., 2014a; Long et al., 2015; Zheng et al., 2015] to fine-tune CNNs for semantic image segmentation. To this end, we convert the annotated semantic scribbles to semantic heatmaps – confidence maps of corresponding classes – and then adapt the fully connected CNNs (FCN) [Long et al., 2015] to accommodate the heatmap-based training data.

3.4.1 Scribbles to Heatmaps

We convert the annotated scribbles to semantic heatmaps of all classes considered. Let $I \in \mathbb{R}^{W \times H \times Z}$ be the input image, with size $[W, H]$ and depth Z (3 for RGB images), its semantic heatmaps are denoted by $P^k \in [0, 1]^{W \times H}$, where $k \in \{1, 2, ..., K\}$, and K is

(a) original scribbles (b) augmented scribbles (c) generated semantic heatmaps

***Figure 3.4**: An example of scribble augmentation and heatmap generation: (a) annotated scribbles; (b) augmented scribbles for the background; (c) generated semantic heatmaps for the two objects.*

the number of classes considered. For classes not present in the images, their heatmaps are simply set to zero. For classes which are present, we generate the heatmap for each class individually. The heatmap is generated by a combination of interactive image segmentation (IIS) and ensemble learning. In particular, we take the scribbles of the class considered as the scribbles for the foreground object in the context of IIS, and the scribbles on other objects as that for the background. For instance if the dog class in Figure 3.4(a) is considered, the scribble on the person is taken as the scribble for background. However, as the example shows, only having the one scribble for the entire background is wanting. We want to add scribbles sampled from the rest of the background as well, thus forming an augmented scribble set. See Figure 3.4(b) for the three additional scribbles. With all scribbles in the augmented set, we run the IIS method proposed in [Gulshan et al., 2010] to get one solution to the heapmap P^k (Figure 3.4(d)), with 1 indicating foreground and 0 background. To further increase the robustness, we run the method T times with different augmented scribbles to obtain T such solutions. The final heatmap (shown in Figure 3.4(c)) for class k is computed as the average of all the individual solutions:

$$P^k = 1/T \sum_{t=1}^{T} P_t^k. \tag{3.2}$$

This is inspired by ensemble learning. Below we present the scribble augmentation.

Scribble Augmentation

For each class, three (sufficient in practice) more scribbles are added. For simplicity, they are added sequentially. New scribbles should be far from the foreground for good separation, and far from existing background scribbles to be complementary. Below, we define the distance between the scribbles.

Given an image I, all paths that connect pixel a and pixel b are denoted by $\mathcal{Q}_{ab}$. Suppose we have a path Γ described by the pixels it passes through $\{\Gamma^1, \Gamma^2, ..., \Gamma^r\}$. The distance between the pixels Γ^1 and Γ^r along path Γ is defined as (following [Gulshan et al., 2010]):

$$D(\Gamma) = \sum_{j=1}^{r} \sqrt{d_{eu}(\Gamma^j, \Gamma^{j+1}) + \lambda \|\nabla I(\Gamma^j)\|^2} \ , \tag{3.3}$$

where $d_{eu}(\Gamma^j, \Gamma^{j+1})$ is the Euclidean distance between two consecutive pixels, and $\|\nabla I(\Gamma^j)\|^2$ is the gradient magnitude between (Γ^j, Γ^{j+1}), which is computed by the edge detector of the structured random forest [Dollár & Zitnick, 2013], to avoid texture edges. λ is used to balance the two terms.

Let's denote the existing scribbles as a set of pixels $\mathcal{E}$. Then, the geodesic distance of pixel a to $\mathcal{E}$ is defined as:

$$G(a) = \min_{b \in \mathcal{E}} \min_{\Gamma \in \mathcal{Q}_{ab}} D(\Gamma) \tag{3.4}$$

Without any specific preference, we fix the shape of the new scribble $\bar{\mathcal{E}}$ to a disk of radius $r = 20$ (a balance between localization and informativeness). Then, the geodesic distance from the scribble centered at position a to existing scribbles $\mathcal{E}$ is:

$$G(\bar{\mathcal{E}}_a) = \min_{c \in \bar{\mathcal{E}}_a} G(c). \tag{3.5}$$

The sampling probability for the next scribble is then defined as:

$$Pr(a) = \frac{\exp(G(\bar{\mathcal{E}}_a)^2/\sigma^2)}{\sum_{b=1}^{WH} \exp(G(\bar{\mathcal{E}}_b)^2/\sigma^2)} \tag{3.6}$$

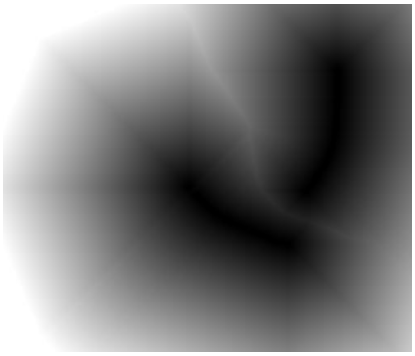
where σ is set adaptively to the average of all $G(\bar{\mathcal{E}}_a)$. $Pr(a)$ needs to be updated each time a new scribble is added. One example of $Pr(a)$ for all values of a is shown in Figure 3.5. The randomness introduced by this sampling allows for the ensemble approach. Note that other information such as objectness [Zitnick & Doll, 2014] can be exploited for sampling the background scribbles. We leave this as future work.

3.4.2 Heatmap-based FCN

FCN [Long et al., 2015] is designed to regress the 2D label map $O \in \{1, 2, ..., K\}^{W \times H}$, directly from the input image I. The output of FCN is K score maps S, one for one class, i.e. $S \in \mathbb{R}^{W \times H \times K}$. The class labels are obtained either by simply taking the best-scoring classes at each pixel [Long et al., 2015] or by using conditional random fields for further refinement [Papandreou et al., 2015]. FCN is trained to minimize the prediction error of cross-entropy over all pixels, and its optimization function is:

$$\mathcal{L} = \sum_{n=1}^{N} \sum_{a=1}^{WH} \mathcal{L}_{na}(S_{na}, O_{na}), \tag{3.7}$$

(a) existing scribbles



(b) probability map

***Figure 3.5**: Probability map for sampling the next new scribble (c.f. Equation 3.6).*

where N is the number of training images and $\mathcal{L}_{na}$ is the per-pixel loss function, which is commonly defined as:

$$\mathcal{L}_{na} = -\log\left(\frac{\exp(S_{na}^{O_{na}})}{\sum_{k=1}^{K}\exp(S_{na}^{k})}\right). \tag{3.8}$$

The loss function is only computed for pixels having ground truth labels. Otherwise, the loss function is set to zero.

Directly applying FCN to our training annotations is problematic, as our annotations are soft semantic heatmaps rather than crisp segmentation masks. To solve this, we extend the loss function in Equation 3.8 to the following:

$$\mathcal{L}'_{na} = \sum_{k=1}^{K} P_{na}^{k}\left[-\log(\frac{\exp(S_{na}^{k})}{\sum_{k'=1}^{K}\exp(S_{na}^{k'})})\right]. \tag{3.9}$$

By the adaptation, the loss function is modulated by the heatmaps of relevant classes – for each pixel, the most confident classes affect the loss function the most. It can be seen that the loss function on crisp segmentation masks in Equation 3.8 is a special case of our new loss function. The new loss function can be optimized the same way as used in standard FCN, with a modification to the loss layer. We call the model heatmap-based FCN (HFCN), and its pipeline is shown in Figure 3.2.

3.5 Experiments

We evaluate HFCN with different settings, and compare it to other competing methods. The goal is to show that training with scribbles-based annotations is a good trade-off between annotation cost and prediction accuracy.

Supervision	Image-level	Point	Bounding-box	Full-mask	Scribbles (ours)		
Method	ConsCNN	WhatPoint	CNN-EM	FCN-8s	HFCN-1.5	HFCN-3	HFCN-3+CRF
mIoU	35.3	42.7	60.6	62.7	56.2	61.9	64.1

***Table 3.4**: The results of different methods with varying levels of supervision on the validation set of PASCAL VOC 2012.*

Method	aerop	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mIoU
Image-level:																					
MIL-FCN	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	24.9
Img2Pix-CNN	25.4	18.2	22.7	21.5	28.6	39.5	44.7	46.6	11.9	40.4	11.8	45.6	40.1	35.5	35.2	20.8	41.7	17.0	34.7	30.4	32.6
Single-Point:																					
What's-point	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	43.6
Bounding-Box:																					
CNN-EM	64.4	27.3	67.6	55.1	64.0	81.6	70.5	76.0	24.1	63.8	58.2	72.1	59.8	73.5	71.4	47.4	76.0	44.2	68.9	50.9	60.8
BoxSup	80.3	31.3	82.1	47.4	62.6	75.4	75.0	74.5	24.5	68.3	56.4	73.7	69.4	72.5	75.1	47.4	70.8	45.7	71.1	58.8	64.6
Full-mask:																					
SDS	63.3	25.7	63.0	39.8	59.2	70.9	61.4	54.9	16.8	45.0	48.2	50.5	51.0	57.7	63.3	31.8	58.7	31.2	55.7	48.5	51.6
FCN-8s	76.8	34.2	68.9	49.4	60.3	75.3	74.7	77.6	21.4	62.5	46.8	71.8	63.9	76.5	73.9	45.2	72.4	37.4	70.9	55.1	62.2
Zoomout	81.9	35.1	78.2	57.4	56.5	80.5	74.0	79.8	22.4	69.6	53.7	74.0	76.0	76.6	68.8	44.3	70.2	40.2	68.9	55.3	64.4
DeepLab-CRF	83.5	36.6	82.5	62.3	66.5	85.4	78.5	83.7	30.4	72.9	60.4	78.5	75.5	82.1	79.7	58.2	82.0	48.8	73.7	63.3	**70.7**
Speech-Scribbles:																					
HFCN-1.5	72.1	32.1	63.2	47.0	59.7	72.1	70.4	72.8	21.6	58.2	41.6	68.2	58.2	71.3	71.5	41.4	56.8	32.1	64.5	52.1	56.4
HFCN-3	76.1	37.4	69.3	53.5	64.9	79.6	74.4	76.4	25.9	62.5	45.8	72.3	62.4	76.8	74.6	45.7	72.1	38.3	68.9	56.2	61.7
HFCN-3+CRF	78.7	37.5	71.7	52.8	64.5	78.6	77.8	79.7	25.9	65.6	49.9	75.1	65.9	79.4	76.6	48.5	74.9	39.9	73.9	59.5	63.9

***Table 3.5**: Comparison to other methods with different levels of supervision on PASCAL VOC 2012 test.*

3.5.1 Experimental Settings

Dataset: We evaluate HFCN on PASCAL VOC 2012 [Everingham et al., 2010a], which comes with three subsets: training (1464 images), validation (1449 images) and test (1456 images), having 20 object classes annotated in full segmentation masks. In order to keep the same settings with previous methods [Long et al., 2015; Pathak et al., 2015a; Papandreou et al., 2015; Bearman et al., 2015], we also extend the training set by adding the extra annotations created by Hariharan *et al.* [Hariharan et al., 2011] (excluding images from the original validation set), ending up with $10,582$ training images. The method is evaluated on the validation set and the test set, under the metric intersection-over-union (IoU) for all the 20 classes.

CNN: We adopt the FCN-8s [Long et al., 2015] model, as it has shown excellent performance and there is code available. The FCN model is adapted from the VGG network [Simonyan & Zisserman, 2015] pre-trained on the ILSVRC dataset [Russakovsky et al., 2015a]. For the optimization, we employ a procedure similar to FCN: the SGD solver is used, with an initial learning rate of 10^{-6}; the network is trained with $80,000$ iterations. HFCN is implemented with the Caffe framework [Jia et al., 2014].

3.5.2 Results

Scribble Augmentation. The scribble for the background is augmented in an ensemble sampling manner, which saves the annotation cost for the background; background is often large and scattered, so annotating it can be costly. The parameter T for the ensemble sampling is evaluated over 10 values: $[1, 2, ..., 10]$ for HFCN-3. The results is shown in Fig 3.6. As the figure shows, the performance increases with T from the beginning and then starts stabilizing. The scribble augmentation from each single round has its own weakness and the combination of multiple rounds 'cancels out' the weaknesses because the weaknesses from different rounds are different due to the randomness in the sampling. This property has been widely explored in the filed of ensemble learning. We use $T = 10$ for all the following experiments.

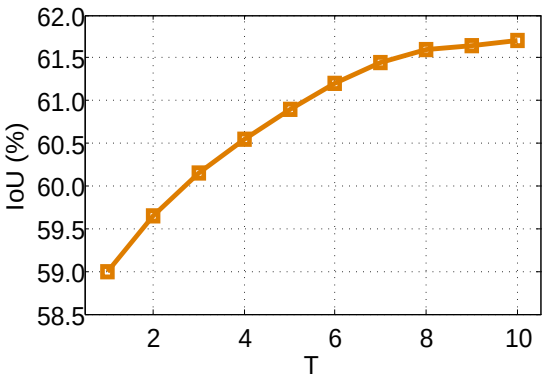


***Figure 3.6**: The performance of HFCN-3 as a function of T.*

Validation Result: We first evaluate the method on the validation set. Table 3.4 show the results, where the results of several other methods are also reported for comparison. For HFCN, we trained it with the two versions of annotations: scribbles from Anno-1.5 and Anno-3. They are referred hereafter by HFCN-1.5 and HFCN-3. Following the literature [Chen et al., 2015a; Pathak et al., 2015a], we also added the refinement by CRF [Krähenbühl & Koltun, 2011] to Anno-3, which is referred by HFCN-3+CRF. It generally true from the table that stronger (more expensive) supervision leads to better performance. However, it also shows that with the scribbles by Anno-1.5, HFCN is already able to yield quite decent results. If the training data is upgraded to the scribbles of Anno-3, the results are comparable to that of FCN-8s [Long et al., 2015] and better than that of CNN-EM [Papandreou et al., 2015], though their training annotations are much more expensive to obtain (*c.f.* Table 3.3). Our method also shows significantly better results than the methods [Bearman et al., 2015; Pathak et al., 2015a] trained with comparable annotation cost. HFCN-3+CRF further improves the performance on top of HFCN-3, showing the benefits of combining graphical model and deep neural networks. Figure 3.7 shows several segmentation examples.

Test Results: We also evaluate HFCN on the test set. Table 3.5 shows all the results. The conclusions drawn on the validation set hold on the test set as well. Our method obtains results which are comparable to other methods trained with more expensive supervision, *i.e.* full masks and bounding boxes, and are better than methods trained with supervision of comparable annotation cost. These annotation costs are only computed on the PASCAL VOC training images, however, some competing methods also use 'extra' supervision.

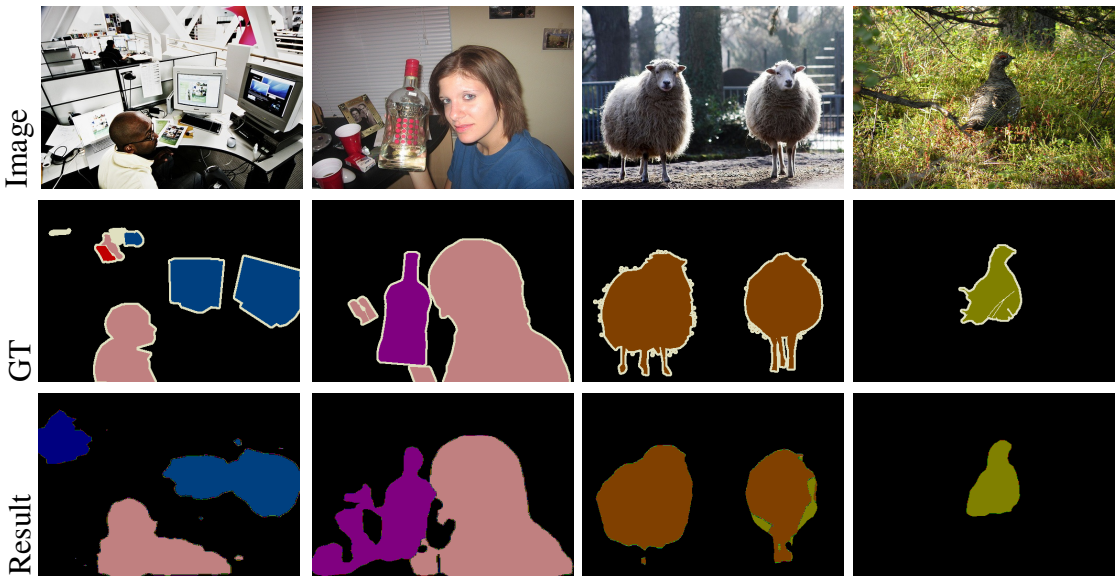


***Figure 3.7**: Examples of the segmentation results on the validation set of PASCAL VOC 2012.*

For instance, Img2Pix-CNN [Pinheiro & Collobert, 2015] learns image prior from another larger set of images; BoxSup [Dai et al., 2015e] uses the MCG object proposal [Arbelaez et al., 2014a] method, which is trained with full-mask supervision from the PASCAL VOC dataset. Also, the bounding boxes used by the two methods [Papandreou et al., 2015; Dai et al., 2015e] are generated from the labeled full segmentation masks, which may transfer some supervision from there because the generated bounding boxes are often tighter than those annotated directly by annotators. The performance of HFCN is improved the same way as other methods [Chen et al., 2015a; Papandreou et al., 2015; Pathak et al., 2015a] by the CRF [Krähenbühl & Koltun, 2011] for further refinement.

Annotation Cost vs. Accuracy: We tabulate the annotation cost and the mIoU of all the methods considered. See Figure 3.8 for the results. From the plot, it is evident that HFCN strikes a good balance between annotation speed and prediction accuracy. The merit makes our method adaptable and applicable to new, customized tasks, which is beneficial to many real vision applications. As more and more forms of supervision are explored, we believe that an evaluation of annotation cost vs. accuracy is very necessary in order to clearly show the trade-offs made by different alternative approaches.

Weak Annotation vs. Strong Annotation. The development of methods with different forms of annotations naturally raises a question: under a fixed annotation budget, should one work for more weak annotations or fewer precise annotations? We answer this question by comparing the performance of HFCN-3 trained on 9900 images annotated by Anno-3 to that of FCN-8 trained on 900 images with full-mask annotations. The training data for FCN-8 is randomly sampled from all the training data. 5 FCN-8 models are trained, each with its own training set, and their results are averaged. The final re-

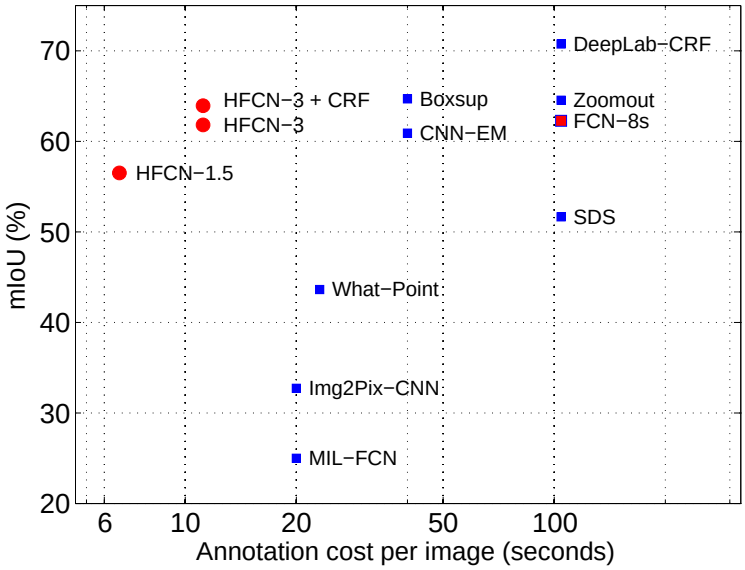


***Figure 3.8**: Annotation vs. segmentation performance (mIoU) of all the methods considered. Evaluated on PASCAL VOC 2012 test.*

sults (mIoU) are: 56.3% for FCN-8 and 60.9% for HFCN-3. The results are exciting, and they suggest that with the same amount of annotation effort, our method gathers more semantic information than the conventional more expensive annotation methods. In a wider context, the results suggest that gathering weak training data can be more helpful than gathering strong training data, if the same amount of annotation effort is given. Learning with a mixture of strong and weak annotations can be interesting as well, as shown in [Dai et al., 2015e]. We leave this as our future work.

Soft Heatmap vs. Hard Segmentation. We investigated whether the soft heatmaps are helpful. To this aim, a baseline is designed for comparison by simply taking the max object probability from the soft maps, and then passing the resulting object masks to FCN-8. We found that by doing so, the performance drops from 61.7 to 59.2. The advantage of soft heatmaps is that objects (parts) with uncertainties are left to be decided and explored by the FCN model that has been trained with objects of higher certainty. The trained FCN model is more knowledgeable than simple thresholding in distinguishing uncertain objects.

3.6 Conclusion

In this work , we developed an annotation method Draw&Tell to create training data for semantic image segmentation. Draw&Tell allows annotators to simply draw scribbles (strokes) on objects and speak their names in the meanwhile, solving the *what* and *where* problems once at the same time. We have proven experimentally that Draw&Tell is faster than other annotation methods, *e.g.* 11 times faster than full-mask annotation, 4 times faster than bounding-box annotations, and 2 times than the image-level annotation. A method of integrating visual information and acoustic information is also proposed for

robust object name recognition. This combination can serve as an example of integrating vision and speech, and inspires research in this direction. Furthermore, we proposed a method that allows CNNs models to learn from scribble-based training data, by converting scribbles to semantic confidence maps and extending standard CNNs to accommodate soft confidence maps. We showed in experiments that our annotation method, coupled with the learning method, yields significantly better results than competing methods trained with annotations of comparable cost, and yields comparable results to the methods trained with significantly more expensive annotations. Introducing speech recognition to visual annotation is helpful especially for tasks on mobile devices. This work is just a very first step in this direction.

4

Representation Learning with Unlabeled Data

4.1 Introduction

Providing efficient solutions to image classification has always been a major focus in computer vision. Recent years have witnessed considerable progress in image classification. However, most popular systems [Lazebnik et al., 2006a; Quattoni & Torralba, 2009; Wang et al., 2010; Xiao et al., 2010; Dai & Yang, 2011; Yang et al., 2014b; Krizhevsky et al., 2012a] heavily rely on manually labeled training data, which is expensive and sometimes impractical to acquire. Despite substantial efforts towards efficient annotation by developing online games [Von Ahn, 2006] or appealing software tools [Russell et al., 2008b], collecting training data for classification is still very time-consuming and tedious. The scarcity of annotations, combined with the explosion of image data, starts shifting focus towards learning with less supervision. As a result, numerous techniques have been developed to learn classification models with cheaper annotations. The most notable ones include unsupervised feature learning [Coates et al., 2011; Dosovitskiy et al., 2014; Doersch et al., 2015; Srivastava et al., 2015], semi-supervised learning [Fergus et al., 2009; Guillaumin et al., 2010; Dai & Van Gool, 2013a], active learning [Jain & Kapoor, 2009; Joshi et al., 2009b], transfer learning [Quattoni et al., 2008; Pan & Yang, 2010], weakly-supervised learning [Prest et al., 2012; Dai et al., 2015a], self-taught learning [Raina et al., 2007], and image clustering [Sivic et al., 2005; Dai et al., 2010a].

In this work, we are interested in the problem of image classification with limited or no annotation. Instead of regularizing the classifiers like most of the previous methods [Bennett & Demiriz, 1998; Joachims, 1999; Kumar Mallapragada et al., 2009; Leistner et al., 2009], we learn a new feature representation using the all available data (labeled + unlabeled). Specifically, we aim to learn a new feature representation by exploiting the distribution patterns of the data to be handled. The setting assumes the availability of unlabeled data in the same or a similar distribution as the test data. This form of weak supervision is naturally available in applications such as semi-supervised image

classification and image clustering, where data in the same or a similar distribution as the test data is available. The learned feature is specifically tuned for the data distribution of interest and performs better for the data than the standard features the method started with. The features to start with for our method can be hand-crafted features [Oliva & Torralba, 2001; Ojala et al., 2002a; Bosch et al., 2007a; Wang et al., 2010], learned features in a supervised manner [Donahue et al., 2014b; Girshick et al., 2014b; Chatfield et al., 2014] or learned features in an unsupervised way [Coates et al., 2011; Dosovitskiy et al., 2014; Doersch et al., 2015; Srivastava et al., 2015; Wang & Gupta, 2015].

Learning with unlabeled data has been quite successful in many fields, for instance in semi-supervised learning (SSL) [Zhou et al., 2004; Weston et al., 2008; Fergus et al., 2009; Leistner et al., 2009; Zhu & Goldberg, 2009; Kingma et al., 2014], in image clustering [Grauman & Darrell, 2006; Dueck & Frey, 2007; Dai et al., 2010a], and in unsupervised feature representation learning [Coates et al., 2011; Dosovitskiy et al., 2014; Srivastava et al., 2015; Wang & Gupta, 2015]. Typically these methods build upon the *local-consistency* assumption that data samples with high similarity should share the same label. This is also called *smoothness of manifold*, and it is often used to regularize the training process for the classifiers or feature representations. In this work, we propose another way to exploit the *local-consistency* assumption to learn a new feature representation. The new feature representation is learned in a discriminative way to capture not only the information of individual images, but also the relationships among images. The learning is conceptually straightforward and computationally simple. The learned features can be fed into any classifiers for the final classification of the unlabeled samples. Thus, the method is agnostic to the classifier choice. This facilitates the deployment of SSL methods, as users often have their favorite classifiers and are reluctant to drop them. For image clustering, we apply the same feature learning methods to all provided images, and then feed the learned features to standard clustering methods such as k-means and Spectral Clustering. Below, we present our motivations and outline the method.

4.1.1 Motivations

People learn and abstract the concepts of object classes well from their intrinsic characteristics, such as colors, textures, and shapes. For instance, *sky* is blue, and a *football* is spherical. We also do so by comparing new object classes to those classes that have already been learned. For example, a *leopard* is similar in appearance to a *jaguar*, but is smaller. This paradigm of learning-by-comparison or characterization-by-comparison is part of Eleanor Rosch's prototype theory [Rosch, 1978], that states that an object's class is determined by its *similarity* to prototypes which represent object classes. The theory has been used successfully in transfer learning [Quattoni et al., 2008], when labeled data of different classes are available. An important question is whether the theory can also be used for feature representation learning when a large amount of unlabeled data is available. This work investigates this problem.

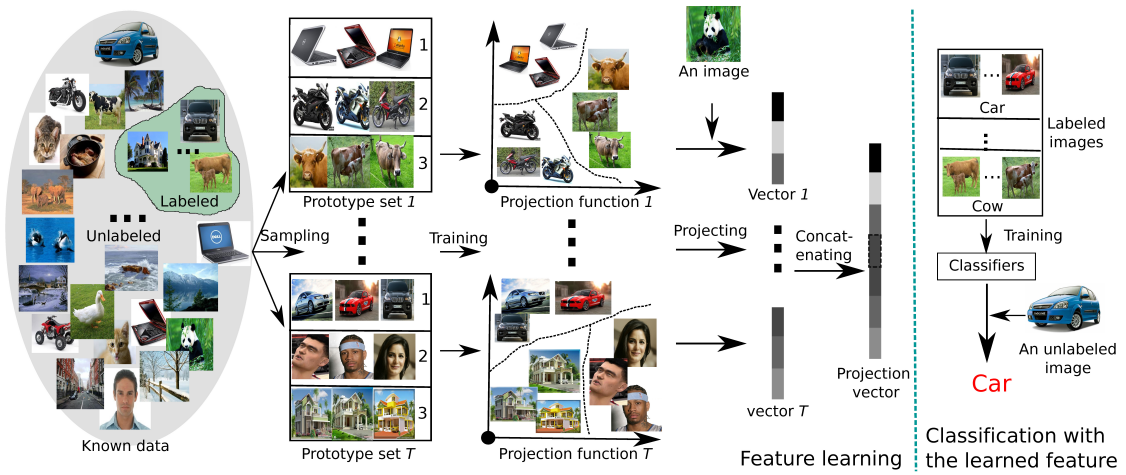


***Figure 4.1**: The pipeline of Ensemble Projection (EP). EP consists of unsupervised feature learning (left panel) and plain classification or clustering (right panel). For feature learning, EP samples an ensemble of T diverse prototype sets from all known images and learns discriminative classifiers on them for the projection functions. Images are then projected using these functions to obtain their new representation. These features are fed into standard classifiers and clustering methods for image classification and clustering respectively.*

To use this paradigm, we first need to create the prototypes automatically from the available data. In keeping with Eleanor Rosch's prototype theory [Rosch, 1978], ideal prototypes should have two properties: 1) images in the same prototype are to be from the same class; and 2) images of different prototypes are to be from different classes. They guarantee meaningful comparisons and reduce ambiguity. Without access to labels of data samples, the prototypes have to be created in an unsupervised way, based on some assumptions. In addition to the widely-used *local-consistency*, we propose another one called *exotic-consistency*, which states that samples that are far apart in the feature space are very likely to come from different classes. The assumptions have been verified experimentally, and will be presented in Section 4.3.1. Based on these two assumptions, it stands to reason that samples along with their closest neighbors can be "good" prototypes, and such prototypes that are far apart can play the role of different classes. According to this observation, we design a method to sample the prototype set from all available images by encoding them on a graph with links reflecting their affinity.

The sampled prototypes are taken as surrogate classes and discriminative learning is yields projection functions tuned to the classes. Images are then linked to the prototypes via their projection values (classification scores) by the functions. Since information carried by a single prototype set is limited and can be noisy, we borrow ideas from ensemble learning [Rokach, 2010] to create an ensemble of diverse prototype sets, which in turn leads to an ensemble of projection functions, to mitigate the influence of the deficiencies of each training set. The idea is that if the deficiency modes of the individual training sets

are different or 'orthogonal', ensemble learning is able to cancel out or at least mitigate their effect. This conjecture is verified with a simulated experiment in Section 4.3.2, and is also supported by the superior performance of our method in real applications. With the ensemble of classifiers, images are then represented by the concatenation of their classification scores – similarities to all the sampled image prototypes – for the final classification, which is in keeping with prototype theory [Rosch, 1978]. We call the method Ensemble Projection (EP). Its schematic diagram is sketched in Figure 4.1.

4.1.2 Contributions

EP was evaluated on eight image classification datasets, ranging from texture classification, over object classification and scene classification, to style classification. For SSL, EP is compared to three baselines and three other methods. For image clustering, EP is compared to the original features it started with. Two standard clustering methods are used: k-means and Spectral Clustering. The experiments show that: (1) EP improves over the original features by exploiting the data distribution of interest, and outperforms competing SSL methods; (2) EP produces promising results for self-taught image classification where the unlabeled data does not follow the same distribution as the labeled ones; (3) EP improves over the original features for image clustering.

The rest of this chapter is organized as follows. Section 4.2 reports on related work. Section 4.3 describes the observations that motivate the method. Section 4.4 is devoted to the approach, followed by experiments in Section 4.5. Section 4.6 concludes the paper.

4.2 Related Work

Our method is generally relevant to image feature learning, semi-supervised learning, ensemble learning, and image clustering.

Supervised Feature Learning: Over the past years, a wide spectrum of features, from pixel-level to semantic-level, have been designed and used for different vision tasks. Due to the semantic gap, recent work extract high-level features, which go beyond single images and are probably impregnated with semantic information. Notable examples are Image Attributes [Farhadi et al., 2009], Classemes [Torresani et al., 2010], and Object Bank [Li et al., 2010a]. While getting pleasing results, these methods all require additional labeled training data, which is exactly what we want to avoid. There have been attempts, *e.g.* [Sharmanska et al., 2012; Yu et al., 2013], to avoid the extra attribute-level supervision, but they still require canonical class-level supervision. Our representation learning however, is fully unsupervised. The pre-trained CNN features [Krizhevsky et al., 2012a; Donahue et al., 2014b; Girshick et al., 2014b; Chatfield et al., 2014] have shown

state-of-the-art performance on various classification tasks. Our feature learning is complementary to their methods. As shown in the experiment, our method can improve on top of the CNN features by exploiting the distribution patterns of the data to be classified. Although the technique of fine-tuning can boost the performance of CNN features for the specific tasks at hand [Oquab et al., 2014; Yosinski et al., 2014], it needs labeled data of a moderate size, which is not always available in our setting. Our method can be understood as unsupervised feature enhancing or fine-tuning.

Unsupervised Feature Learning: Our method is akin to methods which learn middle- or high-level image representation in an unsupervised manner. [Coates et al., 2011] employs k-means mining filters of image patches and then applys the filters for feature computation. [Dosovitskiy et al., 2014] generates surrogate classes by augmenting each patch with its transformed versions under a set of transformations such as translation, scaling, and rotation, and trains a CNN on top of these surrogate classes to generate features. The idea is very similar to ours, but our surrogate classes are generated by augmenting seed images with their close neighbors. The learning methods are also different. [Singh et al., 2012] discovers a set of representative patches by training discriminative classifiers with small, compact patch clusters from one dataset, and testing them on another dataset to find similar patches. The found patches are then used to train new classifiers, which are applied back to the first dataset. The process iterates and terminates after rounds, resulting in a set of representative patches and their corresponding 'filters'. The idea of learning 'filters' from compact clusters shares similarities with what we do, but our clusters are images rather than patches. Other forms of weak supervision have also been exploited to learn good feature representation without human labeled data, and they all obtain very promising results. For instance, [Doersch et al., 2015] uses the spatial relationships of image windows in an image as the supervision to train a neural network; [Wang & Gupta, 2015] exploits the tracking results of objects in videos to guide the training of a neural network to learn feature representations; and [Agrawal et al., 2015] exploits the ego-motion of cameras for the training. These methods aim for general feature representation. Our method, however, is designed to 'tune' or enhance vision features specifically for the datasets on which the vision tasks are performed.

Semi-supervised Learning: SSL aims at enhancing the performance of classification systems by exploiting an additional set of unlabeled data. Due to its great practical value, SSL has a rich literature [Chapelle et al., 2006; Zhu & Goldberg, 2009]. Amongst existing methods, the simplest methodology for SSL is based on the self-training scheme [Blum & Mitchell, 1998] where the system iterates between training classification models with current 'labeled' training data and augmenting the training set by adding its highly confident predictions in the set of unlabeled data; the process starts from human labeled data and stops until some termination condition is reached, *e.g.* the maximum number of iterations. [Guillaumin et al., 2010] and [Shrivastava et al., 2012] presented two methods in this stream for image classification. While obtaining promising results, they both require

additional supervision: [Guillaumin et al., 2010] need image tags and [Shrivastava et al., 2012] image attributes.

The second group of SSL methods is based on label propagation over a graph, where nodes represent data examples and edges reflect their similarities. The optimal labels are those that are maximally consistent with the supervised class labels and the graph structure. Well known examples include Harmonic-Function [Zhu et al., 2003], Local-Global Consistency [Zhou et al., 2004], Manifold Regularization [Belkin et al., 2006], and Eigenfunctions [Fergus et al., 2009]. While having strong theoretical support, these methods are unable to exploit the power of discriminative learning for image classification.

Another group of methods utilize the unlabeled data to regularize the classifying functions – enforcing the boundaries to pass through regions with a low density of data samples. The most notable methods are transductive SVMs [Joachims, 1999], Semi-supervised SVMs [Bennett & Demiriz, 1998], and semi-supervised random forests [Leistner et al., 2009]. These methods have difficulties to extend to large-scale applications, and developing an efficient optimization for them is still an open question. Readers are referred to [Zhu & Goldberg, 2009] for a thorough overview of SSL.

Ensemble Learning: Our method learns the representation from an ensemble of prototype sets, thus sharing ideas with ensemble learning (EL). EL builds a committee of base learners, and finds solutions by maximizing the agreement. Popular ensemble methods that have been extended to semi-supervised scenarios are Boosting [Kumar Mallapragada et al., 2009] and Random Forests [Leistner et al., 2009]. However, these methods still differ significantly from ours. They focus on the problem of improving classifiers by using unlabeled data. Our method learns new representations for images using all data available. Thus, it is independent of the classification method. The reason we use EL is to capture rich visual attributes from a series of prototype sets, and to mitigate the deficiency of the sampled prototype sets. Other work close to ours is Random Ensemble Metrics [Kozakaya et al., 2011], where images are projected to randomly subsampled training classes for supervised distance learning.

Image Clustering: A plethora of methods have been developed for image clustering. [Fergus et al., 2003] modeled objects as constellations of visual parts and estimated parameters using the expectation-maximization algorithm for unsupervised classification. [Sivic et al., 2005] proposed using aspect models to discover object classes from an unordered image collection. Later on, [Sivic et al., 2008] used Hierarchical Latent Dirichlet Allocation to automatically discover object class hierarchies. For scene class discovery, [Dai et al., 2010a] proposed to combine information projection and clustering sampling. These methods assume explicit distributions for the samples. Image classes, nevertheless, are arranged in complex and widely diverging shapes, making the design of explicit models difficult. An alternative strand, which is more versatile in handling structured data, builds on similarity-based methods. [Dueck & Frey, 2007] applied the affinity propagation algorithm of [Frey & Dueck, 2007] for unsupervised image categorization. [Grauman

& Darrell, 2006] developed partially matching image features to compute image similarity and used spectral methods for image clustering. The main difficulty of this strand is how to measure image similarity as the semantic level goes up. Readers are referred to [Tuytelaars et al., 2010] for a survey.

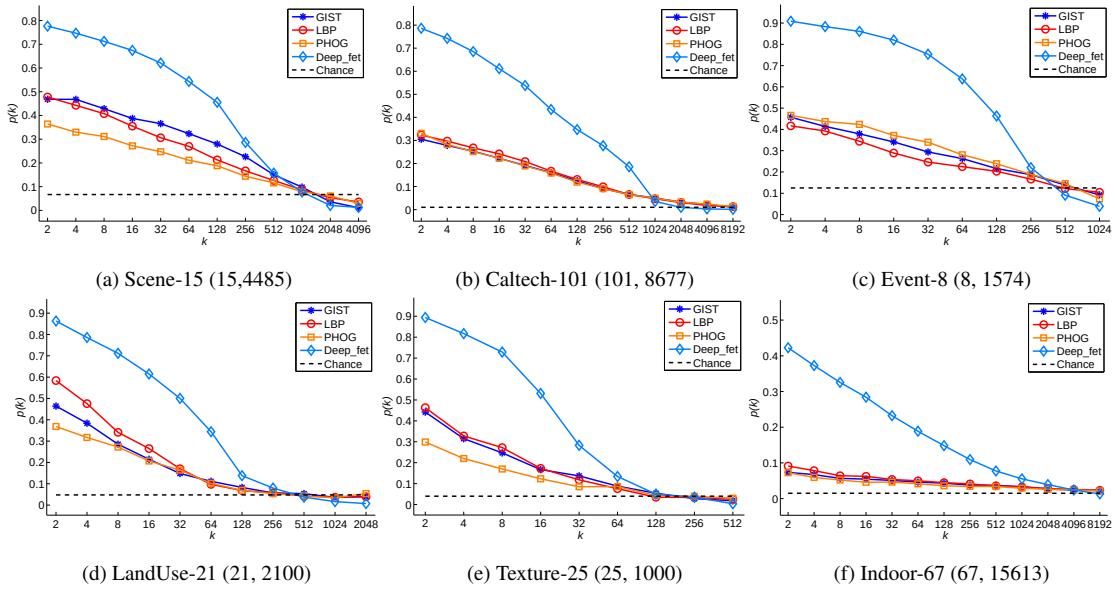
4.3 Observations

In this section, we motivate our approach and explain why it is working. We experimentally verify our assumptions: First, given a standard distance metric over images, do the assumptions *local-consistency* and *exotic-inconsistency* hold, and to what extent? Second, is ensemble learning able to cancel out the deficiency of the individual training sets, given that the number of such training sets are sufficiently large and the deficiency modes of them are different or 'orthogonal'?

4.3.1 Observation 1

The assumptions of *local-consistency* and *exotic-consistency* do hold for real image datasets. An ideal image representation along with a distance metric should ensure that all images of the same class are more similar to each other than to those of other classes. However, this does not strictly hold for most of vision systems in reality. In this section, we want to verify whether the relaxed assumptions *local-consistency* and the *exotic-consistency* hold. These state images are very likely from the same class as their close neighbors, and very likely from different classes than those far from them. In order to examine the assumptions, we tabulate how often an image is from the same class as its k^{th}-nearest neighbor. We refer to the frequency as label co-occurrence probability $p(k)$. $p(k)$ is averaged across images and class labels in the dataset. Four features were tested: GIST [Oliva & Torralba, 2001], PHOG [Bosch et al., 2007a], LBP [Ojala et al., 2002a], and the CNN feature [Chatfield et al., 2014]. The Euclidean distance is used here.

Figure 4.2 shows the results on six datasets (Datasets and features will be introduced in Section 4.5). The results reveal that using the distance metric in conventional ways (*e.g.* clustering by k-means and spectral methods) will result in very noisy training sets, because the label co-occurrence probability $p(k)$ drops very quickly with k. Sampling in the very close neighborhood of a given image is likely to generate more instances of the same class, whereas sampling far-away tends to gather samples of different classes. This suggests that samples along with a few very close neighbors, namely "compact" image clusters, can form a training set for a single class, and a set of such image clusters far away from each other in feature space can serve as good prototype sets for different classes. Furthermore, sampling in this way provides the chance of creating a large number of diverse prototype sets, due to the small size of each sampled prototype set. Also, from



(a) Scene-15 (15,4485) (b) Caltech-101 (101, 8677) (c) Event-8 (8, 1574)

(d) LandUse-21 (21, 2100) (e) Texture-25 (25, 1000) (f) Indoor-67 (67, 15613)

***Figure 4.2**: The label co-occurrence probability $p(k)$: frequency of images having the same label with their k_{th} neighbors. Results on six datasets are shown. The number of classes and the number of images of the datasets are shown as well.*

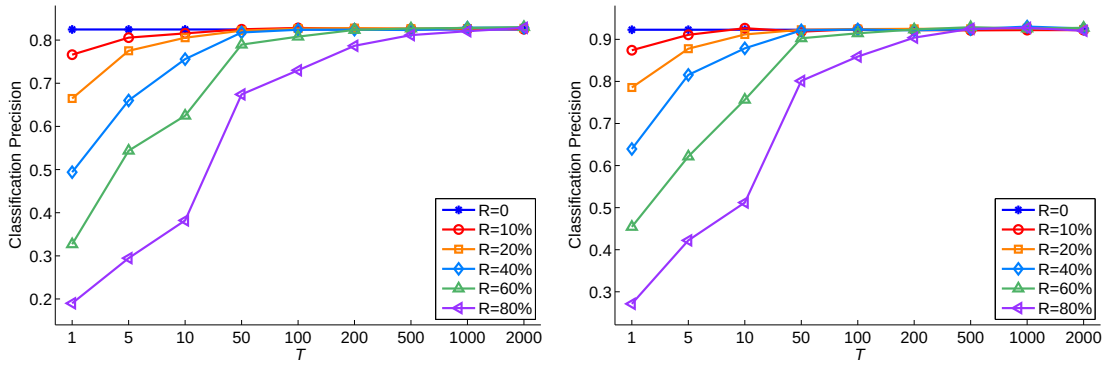
this figure, it is evident that the CNN feature performs significantly better than the rest, which suggests that using the CNN feature in our system is recommendable.

4.3.2 Observation 2

Ensemble learning is able to cancel out or substantially mitigate the deficiency of individual training sets, given that the number of such training sets is sufficiently large and the modes of the deficiency are different or 'orthogonal'.

We examined this idea in supervised image categorization. Given the ground truth data divided into training and test sets: $\mathcal{D} = \{\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{test}}\}$, (i) we artificially synthesized a set of weak training sets (training sets with different modes of deficiency) $\mathcal{D}_t^{\text{train}}, t = 1, \ldots, T$ from training data $\mathcal{D}^{\text{train}}$, and (ii) ensemble learning was then performed on these sets and its performance on test data classification was measured.

In order to guarantee the diversity of the training sets (for ensemble learning), each weak training set $\mathcal{D}_t^{\text{train}}$ is formed by randomly taking 30% of the images in $\mathcal{D}^{\text{train}}$, and randomly re-assigning labels of a fixed percentage R of these images. Hence, $R = 0$ corresponds to the upper performance bound as every sample is assigned its true label. A classifier is trained for each of these weak training sets. At test time, each of these classifiers returns the class label of each image in $\mathcal{D}^{\text{test}}$. The winning label is the mode of the results returned by all the classifiers. Figure 4.3 evaluates this for the Scene-15 dataset [Lazebnik et al., 2006a]. Logistic Regression is used as the classifiers with the CNN feature [Chatfield



(a) Scene-15 (b) Event-8

Figure 4.3: *Classification accuracy of ensemble learning on the the Scene-15 dataset [Lazebnik et al., 2006a] and the Event-8 dataset [Li & Fei-Fei, 2007a], for varying training label noise R and varying number of training trials T. Experiments on other datasets obtain the same trend. Ensemble learning is able to cancell out the deficiency of the training sets even it is very severe (e.g. $R = 80\%$), given that the deficiency modes are different or 'orthogonal' and the number of training sets are sufficiently large. The figure is best viewed in color.*

et al., 2014] as input. When the label noise percentage R is low, the classification precision starts out high and levels quickly with T, as one would expect. But interestingly, for R even as high as 80%, the classification precision, which starts low, converges to a similarly high precision given sufficient weak training sets T (≈ 500). This suggests that ensemble learning is able to cancel out the deficiency of individual training sets. It learns the essence of image classes when the modes of deficiency are different for different training sets, and given a sufficiently large number of such training sets.

We were inspired by the two observations, and would like to investigate whether the assumptions of *local-consistency* and *exotic-consistency* are enough to generate a set of such weak training sets in an unsupervised manner, with which ensemble learning is able to learn useful visual attributes for semi-supervised image classification and image clustering.

4.4 Our Approach

The training data consists of both labeled data $\mathcal{D}_l = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and unlabeled data $\mathcal{D}_u = \{\mathbf{x}_i\}_{i=l+1}^{l+u}$, where $\mathbf{x}_i$ denotes the feature vector of image i, $y_i \in \{1, ..., C\}$ represents its label, and C is the number of classes. For image clustering, $l = 0$, and u is the total number of images. Most previous semi-supervised learning (SSL) methods learn a classifier $\phi : \mathcal{X} \mapsto \mathcal{Y}$ from $\mathcal{D}_l$ with a regulation term learned from $\mathcal{D}_u$. Our method learns a new image representation $\mathbf{f}$ from all known data $\mathcal{D} = \mathcal{D}_l \cup \mathcal{D}_u$, and trains standard

classifier models ϕ with $\mathbf{f}$. $\mathbf{f}_i$ is a vector of similarities of image i to a series of sampled image prototypes.

Assume that Ensemble Projection learns knowledge from T prototype sets $\mathcal{P}^{t,t\in\{1,...,T\}} = \{(s_i^t, c_i^t)\}_{i=1}^{rn}$, where $s_i^t \in \{1, ..., l+u\}$ is the index of the i^{th} chosen image, $c_i^t \in \{1, ..., r\}$ is the pseudo-label indicating which prototype s_i^t belongs to. r is the number of prototypes (surrogate classes) in $\mathcal{P}^t$, and n the number of images sampled for each prototype (class) (*e.g.* $r = 3$ and $n = 3$ in Figure 4.1). Below, we first present our sampling method for creating a single prototype set $\mathcal{P}^t$ in the tth trial, followed by EP.

4.4.1 Max-Min Sampling

As stated, we want the prototypes to be inter-distinct and intra-compact, so that each one represents a different visual concept. To this end, we design a 2-step sampling method, termed Max-Min Sampling. The Max step is based on the *exotic-consistency* and caters for the inter-distinct property; the Min-step is based on the *local-consistency* assumption and caters for the intra-compact requirement. In particular, we first sample a skeleton of the prototype set, by looking for image candidates that are strongly spread out, i.e. at large distances from each other. We then enrich the skeleton to a prototype set by including the closest neighbors of the skeleton images. The algorithm for creating $\mathcal{P}^t$ is given in Algorithm 1. For the skeleton, we sampled m hypotheses – each one consists of r randomly sampled images. For each hypothesis, the average pairwise distance between the r images is then computed. Finally, we take the hypothesis yielding the largest average mutual distance as the skeleton. This simple procedure guarantees that the sampled seed images are far from each other. Once the skeleton is created, the Min-step extends each seed image to an image prototype by introducing its n nearest neighbors (including itself), in order to enrich the characteristics of each image prototype and reduce the risk of introducing noisy images. The pseudo-labels are shared by all images specifying the same prototype. It is worth pointing out that the randomized Max-step may not generate the optimal skeleton. However, it serves its purpose well. For one thing, we do not need the optimal one – we only need the prototypes to be *far apart*, not *farthest apart*. Moreover, randomization allows diverse visual concepts to be captured in different $\mathcal{P}^t$'s. The influence of the optimality of each single skeleton is tested in Section 4.5.1. The Euclidean distance is also used here.

4.4.2 Ensemble Projection

We now explore the use of the image prototype sets created in Section 4.4.1 for a new image representation. Because the prototypes are compact in feature space, each of them implicitly defines a visual concept (image attribute). This is especially true when the dataset $\mathcal{D}$ is sufficiently large, which is to be expected given the vast number of unlabeled

Algorithm 1: Max-Min Sampling in t^{th} trial

Data: Dataset $\mathcal{D}$
Result: Prototype set $\mathcal{P}^t$
begin
  $\hat{e} = 0$; /* Max-step */
  while *iterations* $\leq m$ **do**
    $\mathcal{V} = \{r$ random image indexes$\}$;
    $e = \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} \text{dis}(\mathbf{x}_i, \mathbf{x}_j)$;
    if $e > \hat{e}$ **then**
      $\hat{e} = e$;
      $\hat{\mathcal{V}} = \mathcal{V}$;
    end
  end
  for $i \leftarrow 1$ **to** r **do** /* Min-step */
    $\mathbf{s}_i^t =$ stacked indexes of the n nearest neighbors of $\mathcal{V}(i)$ in $\mathcal{D}$;
    $\mathbf{c}_i^t = (i, i, ..., i) \in \mathbb{R}^n$;
  end
  $\mathbf{s}^t = (\mathbf{s}_1^t, ..., \mathbf{s}_r^t) \in \mathbb{R}^{rn}$ $\mathbf{c}^t = (\mathbf{c}_1^t, ..., \mathbf{c}_r^t) \in \mathbb{R}^{rn}$;
  $\mathcal{P}^t = \{(s_i^t, c_i^t)\}_{i=1}^{rn}$;
end

images that are available. Since the information carried by a single prototype set $\mathcal{P}^t$ is quite limited and noisy, we borrow an idea from ensemble learning (EL), namely to create an ensemble of T such sets to accumulate wisdom from a brood set of training images. A sanity check of this was already presented for a simulated situation in Section 4.3.2.

As is well-known, EL benefits from the precision of its base learners and their diversity. To obtain a good precision, discriminative learning method is employed for the base learner $\phi_t(.)$; logistic regression is used in our implementation to project each input image $\mathbf{x}$ to the image prototypes to measure the similarities. This choice is both due to its training efficiency and because lower capacity models are better suited for the sparse, small-size datasets under consideration. To achieve a high diversity, randomness is introduced in different trials of Max-Min Sampling to create an ensemble of diverse prototype sets, so that a rich set of image attributes are captured. The vector of all similarities is then concatenated and used as a new image representation $\mathbf{f}$ for the final classification. A standard classifier (*e.g.* SVMs, Boosting, or Random Forest) can then be trained on $\mathcal{D}_l$ with the learned feature $\mathbf{f}$ for the semi-supervised classification, as unlabeled data has already been explored when obtaining $\mathbf{f}$. Likely, image clustering is performed by injecting the learned feature to a standard clustering method. The whole procedure of EP is presented in Algorithm 2. By now, the whole pipeline in Figure 4.1 has been explained.

Algorithm 2: Ensemble Projection

Data: Dataset $\mathcal{D}$ with image presentation $\mathbf{x}_i$
Result: Dataset $\mathcal{D}$ with image presentation $\mathbf{f}_i$
begin
  for $t \leftarrow 1$ **to** T **do**
    Sample $\mathcal{P}^t = \{(s_i^t, c_i^t)\}_{i=1}^{rn}$ using Algorithm 1 ;
    Train classifiers $\phi^t(.) \in \{1, ..., r\}$ on $\mathcal{P}^t$;
  end
  for $i \leftarrow 1$ **to** $l + u$ **do**
    for $t \leftarrow 1$ **to** T **do**
      Obtain projection vector: $\mathbf{f}_i^t = \phi^t(\mathbf{x}_i)$;
    end
    $\mathbf{f}_i = ((\mathbf{f}_i^1)^\top, ..., (\mathbf{f}_i^T)^\top)^\top$;
  end
end

4.5 Experiments

The effectiveness of the approach is evaluated in the situations of: (1) semi-supervised image classification, where the amount of labeled data is sparse relative to the total amount of data; and (2) image clustering, where no labeled data is provided. In this section, we will first introduce the datasets and the features used, followed by experimental results for the two tasks and their corresponding analysis.

Datasets: The method is evaluated on diverse classification tasks: texture classification, object classification, scene classification, event classification, style classification, and satellite image classification. Eight standard datasets are used for the evaluation:

- Texture-25 [Lazebnik et al., 2005]: 25 texture classes, with 40 samples per class.
- Caltech-101 [Fei-Fei et al., 2004]: 101 object classes, with 31 to 800 images per class, and 8677 images in total,
- STL-10 [Coates et al., 2011]: 10 object classes including airplane, bird, car, cat, deer, dog, horse, monkey, ship, truck, with 500 training images per class, 800 test images per class, and 100000 unlabeled images for unsupervised learning.
- Scene-15 [Lazebnik et al., 2006a]: 15 scene classes with both indoor and outdoor environments, 4485 images in total. Each class has 200 to 400 images.
- Indoor-67 [Quattoni & Torralba, 2009]: 67 indoor classes such as shoe shop, mall and garage, with a total of 15620 images and at least 100 images per class.

- Event-8 [Li & Fei-Fei, 2007a]: 8 sports event classes including rowing, badminton, polo, bocce, snowboarding, croquet, sailing, and rock climbing, with a total of 1574 images.
- Building-25 [Xu et al., 2014]: 25 architectural styles such as American craftsman, Baroque, and Gothic, with 4794 images in total.
- LandUse-21 [Yang & Newsam, 2010]: 21 classes of satellite images in terms of land usage, such as agricultural, airplane, forest. There are 2100 images in total, with 100 images per class.

Features: The following three features were used in our earlier papers [Dai et al., 2012c; Dai & Van Gool, 2013a] due to their simplicity and low dimensionality: GIST [Oliva & Torralba, 2001], Pyramid of Histogram of Oriented Gradients (PHOG) [Bosch et al., 2007a], and Local Binary Patterns (LBP) [Ojala et al., 2002a]. However, these features are obsolete and yield results inferior than alternative features recently developed for image classification. In this work, we replaced them with the CNN features [Donahue et al., 2014b; Chatfield et al., 2014]. These were obtained from an off-the-shelf CNN pre-trained on the ImageNet data. They were chosen as CNN features have achieved state-of-the-art performance for image classification [Krizhevsky et al., 2012a; Dosovitskiy et al., 2014]. For implementation, we used the MatConvNet [Vedaldi & Lenc, 2014] toolbox, with a 21-layer CNN pre-trained model being used. The convolutional results at layer 16 were stacked as the CNN feature vector, with dimensionality of 4096. We also tested the LLE-coded SIFT feature [Wang et al., 2010]. However, it is not on par with the CNN features.

Competing methods: For semi-supervised classification, six classifiers were adopted to evaluate the method, with three baselines: k-NN, Logistic Regression (LR), and SVMs with RBF kernels, and three semi-supervised classifiers: Harmonic Function (HF) [Zhu et al., 2003], LapSVM [Belkin et al., 2006], and Anchor Graph (AG) [Liu et al., 2010]. HF formulates the SSL learning problem as a Gaussian Random Field on a graph for label propagation. LapSVM extends SVMs by including a smoothness penalty term defined on the Laplacian graph. AG aims to address the scalability issue of graph-based SSL, and constructs a tractable large graph by coupling anchor-based label prediction and adjacency matrix design. For image clustering, we compare our learned feature to the original CNN feature with two standard clustering algorithms: k-means and Spectral Clustering. Existing systems for image clustering often report performance on relatively easy datasets and it is hard to compare with them on these standard classification datasets.

Experimental settings: We conducted four sets of experiments: (1) compare our method with competing methods for semi-supervised image classification on the eight datasets, where the unlabeled images are from the same class as the labeled ones; (2) evaluate the robustness of our method against the choice of its parameters and classifier models in the context of semi-supervised image classification; (3) evaluate the performance of our

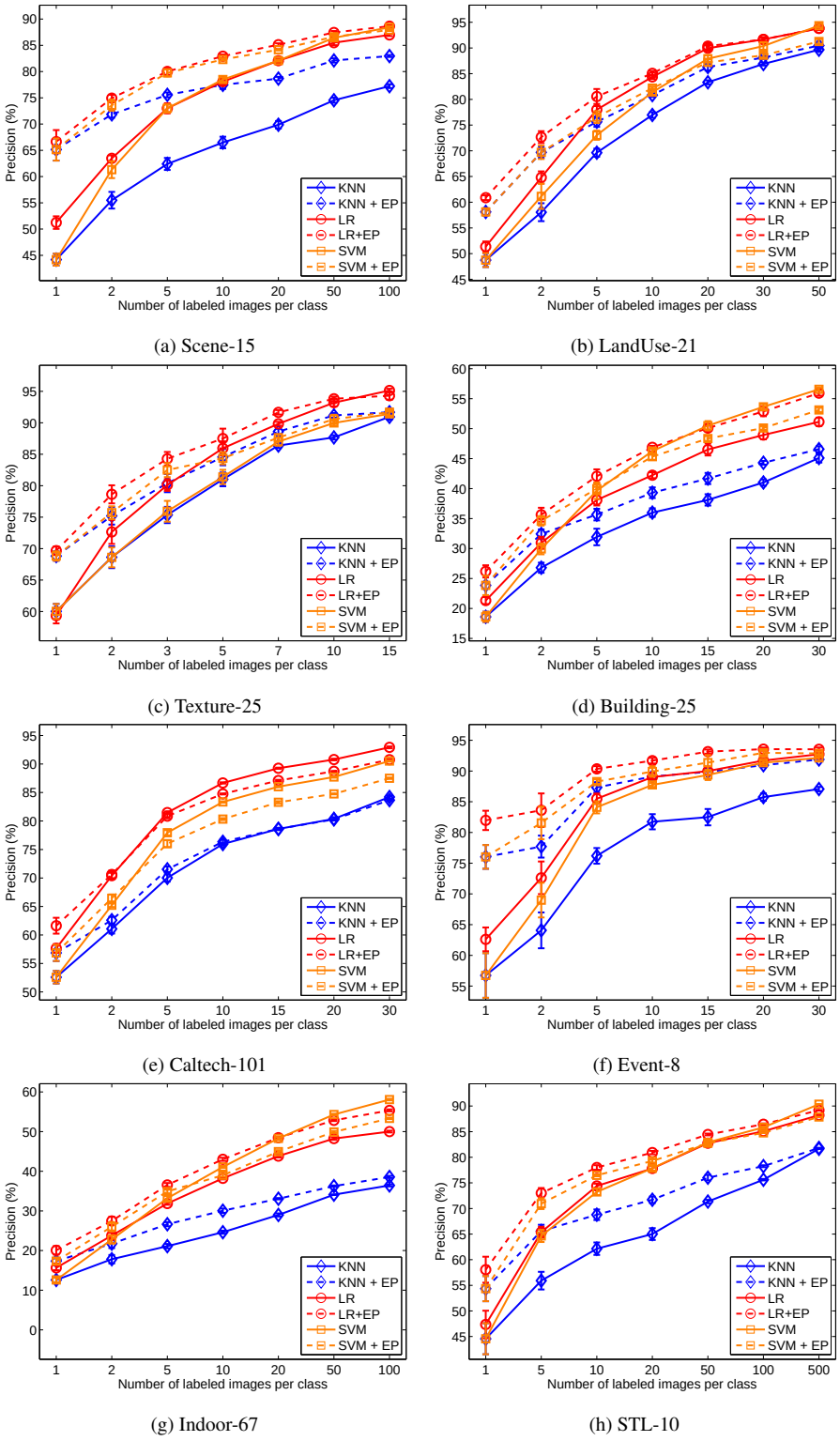
Methods	Scene-15	LandUse-21	Texture-25	Building-25	Event-8	Caltech-101	Indoor-67	STL-10
k-NN	62.4	69.6	81.0	31.9	76.2	70.0	21.1	55.9
k-NN + **EP**	75.6	75.6	<u>84.5</u>	35.7	87.3	71.5	26.6	65.6
LR	73.0	<u>78.0</u>	85.9	38.1	85.5	**81.5**	31.9	65.4
LR + **EP**	**80.0**	**80.6**	**87.5**	**42.1**	**90.4**	<u>80.9</u>	**36.6**	73.0
SVMs	73.0	73.0	81.4	39.6	84.1	77.9	33.2	64.6
SVMs + **EP**	<u>79.8</u>	76.6	84.4	<u>40.0</u>	<u>88.3</u>	76.0	<u>34.9</u>	70.9
HF	45.2	39.1	67.9	18.5	15.6	70.6	7.4	10.3
HF + **EP**	61.5	52.3	74.6	21.2	27.1	75.8	12.1	38.1
AG	72.8	51.3	50.5	34.8	51.0	67.7	24.7	**76.0**
AG + **EP**	78.3	58.5	32.1	37.5	50.5	66.3	24.9	<u>74.9</u>

Table 4.1: *Precision (%) of image classification on the eight datasets, with* 5 *labeled training examples per class. "+ EP" indicate that classifiers working with our learned feature as input rather than the original CNN. The best performance is indicated in* ***bold****, and the second best is* <u>underlined</u>*.*

method for the task of self-taught image classification on the STL-10 dataset, where the feature is learned from the unlabeled images and the performance is tested on the labeled set; and (4) evaluate our method for the task of image clustering on the eight datasets.

For all experimental setups except (2), the same set of parameters were used for all the classifiers. We used $k = 1$ for the k-NN classifier, L2-regularized LR of LIBLINEAR [Fan et al., 2008] with $C = 15$, and the SVMs with RBF kernel of LIBSVM [Chang & Lin, 2011] with $C = 15$ and the default g, *i.e.* $g = 1/4096$. For LapSVM, we used the scheme suggested by [Belkin et al., 2006]: γ_A was set as the inductive model, and γ_I was set as $\frac{\gamma_I l}{(l+u)^2} = 100\gamma_A l$. For HF, the weight matrix was computed with the Gaussian function $e^{-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2}$, where σ is automatically set by using the self-tuning method [Zelnik-Manor & Perona, 2004]. For AG, we followed the suggestion from the original work [Liu et al., 2010] and used the following for both our learned feature and the original CNN feature: 1000 anchors and features reduced to 500 dimensions via PCA.

As to the parameters of our method, a wide variety of values for them were tested in experimental setup (2). In experimental setups (1), (3) and (4), we fixed them to the following values: $T = 100$, $r = 30$, $n = 6$, and $m = 50$, which leads to a feature vector of 3000 dimensions. Note that the learned feature may contain redundancy across different dimensions, as some prototype sets are similar to others. We leave the task of selecting useful features to the discriminative classifiers.



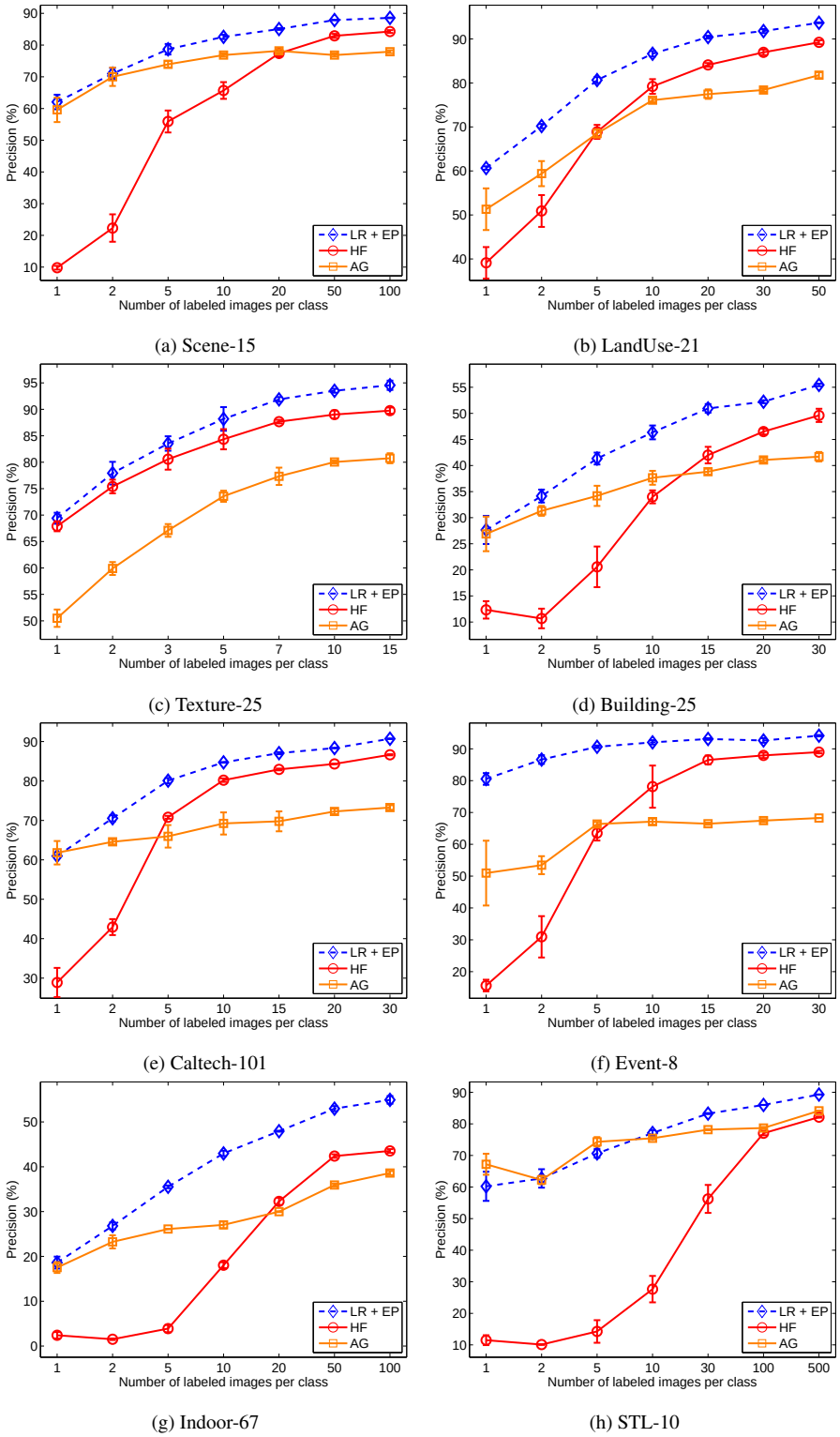
(a) Scene-15 (b) LandUse-21

(c) Texture-25 (d) Building-25

(e) Caltech-101 (f) Event-8

(g) Indoor-67 (h) STL-10

Figure 4.4: *Classification results of Ensemble Projection (EP) on the eight datasets, where three classifiers are used: k-NN, Logistic Regression, and SVMs with RBF kernels. All methods were tested with two feature inputs: the original deep feature and the learned feature by EP on top of it (indicated by "+ EP").*



(a) Scene-15 (b) LandUse-21

(c) Texture-25 (d) Building-25

(e) Caltech-101 (f) Event-8

(g) Indoor-67 (h) STL-10

Figure 4.5: *Classification results of Ensemble Projection (EP) on the eight datasets, where three classifiers are used: k-NN, Logistic Regression, and SVMs with RBF kernels. All methods were tested with two feature inputs: the original deep feature and the learned feature by EP on top of it (indicated by "+ EP").*

4.5.1 Semi-supervised Image Classification

In this section, we evaluate all methods across all datasets for semi-supervised image classification. Different numbers of training images per class were tested: Scene-15 and Indoor-67 with {1, 2, 5, 10, 20, 50, 100}, LandUse-21 with {1, 2, 5,10, 20, 30, 50}, Texture-25 with {1, 2, 3, 5, 7, 10 , 15}, Building-25, Event-8, and Caltech-101 with {1, 2, 5, 10, 15, 20, 30}, and STL-10 with {1, 5, 10, 20, 50, 100, 500}. The different choices are due to the different structures of the datasets: different number of classes and different number of images per class. In keeping with most existing systems for semi-supervised classification [Zhu et al., 2003; Zhou et al., 2004; Liu et al., 2010; Fergus et al., 2009; Ebert et al., 2010; Pitelis et al., 2014], we evaluate the method in the transductive manner, where we take the training and test samples as a whole, and randomly choose labeled samples from the whole dataset to learn and infer labels of other samples whose labels are held back as the unlabeled samples. The reported results are the average performance over 5 runs with random labeled-unlabeled splits.

Comparison to baselines: Figure 4.4 shows the results of the three baseline classifiers with our learned feature and the original CNN feature as input, and Table 4.1 lists the results of all methods when 5 labeled training samples are available for each class. From the figure, it is easy to observe that the three plain classifiers k-NN, LR and SVMs perform consistently better when working with our feature than working with the original CNN features. This is, of course, not a fair comparison, as our feature has been learned with the help of unlabeled samples, while the CNN features not. However, this experiment serves as a good sanity check: given the access to the unlabeled samples, does the proposed feature learning improve the performance of the system over the original feature? The figure shows clear advantages of our method over the original CNN feature across different datasets and classifiers. The most pronounced improvement occurs in the scenarios where a small number of labeled training samples is available, *e.g.* from 1 to 5. This is exactly what the method is designed for – classification tasks where the labeled training samples are sparse relative to the available unlabeled samples. Since LR performs generally the best when working with our learned feature, we will take LR + EP as our method to compare to other SSL methods. The comparison is made in the next section.

Comparison to other SSL Methods: In this section, we compare our method (LR + EP) with the three SSL methods HF, AG, and LapSVM. The classification precision is reported for HF and AG, while the mean average precision (mAP) of C rounds of binary classification is used for LapSVM. This is because the implementation of LapSVM from the authors performs binary classification [Belkin et al., 2006]. Because LapSVM is computationally expensive, we only compare our method to it for the scenario where 5 labeled training samples per class are used.

Figure 4.5 shows the results of our method (LR + EP) and that of HF and AG, and Table 4.1 lists the precision of the methods when 5 labeled training examples per class are

Methods	Scene-15	LandUse-21	Texture-25	Building-25	Event-8	Caltech-101	Indoor-67	STL-10
LR + **EP**	**84.8**	**85.6**	**95.1**	**39.2**	**91.7**	**73.1**	**33.2**	**81.5**
HF	81.4	84.2	94.1	37.9	89.5	71.6	25.1	78.1
LapSVM	79.2	82.3	91.4	35.8	86.2	56.4	29.3	69.3

***Table 4.2**: MAP (%) of semi-supervised classification on the eight datasets, with 5 labeled training examples per class. "LR + EP" indicate Logistic Regression with our learned feature as input. The other two classifiers use the original CNN feature as input. The best number is indicated in **bold**.*

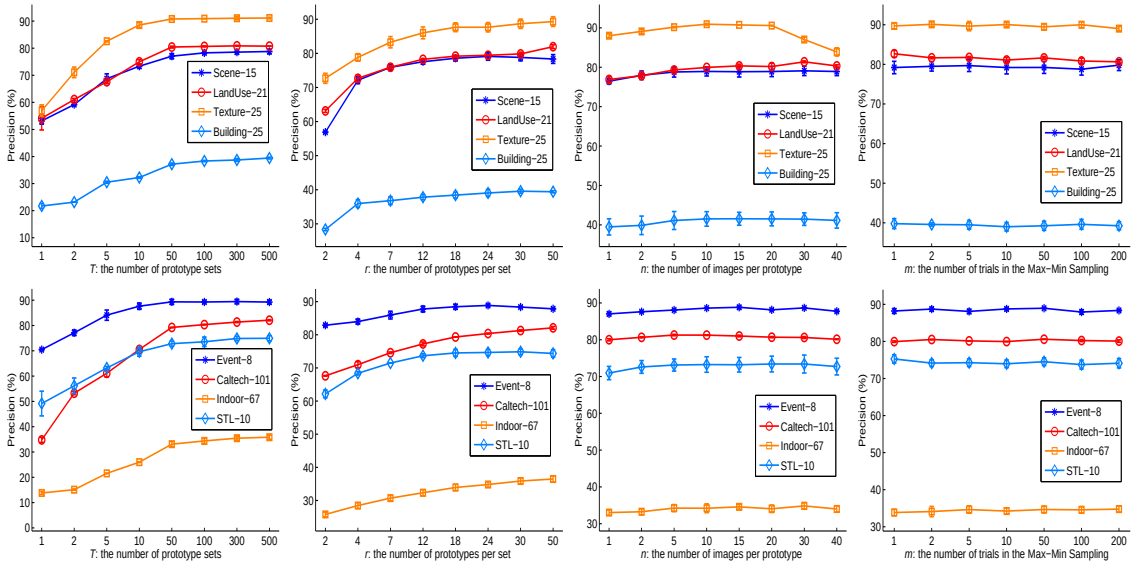
used. Table 4.2 lists the mAP of our method, HF and LapSVM, when 5 labeled training samples are available for each class. The figure and the tables show that our method outperforms the competing SSL methods consistently for semi-supervised image classification. For instance, if 5 labeled training examples per class are used, our method (LR + EP) improves over the best competing method AG by 7.2% in terms of precision on Scene-15, and by 11.9% on Indoor-67. This suggests that our method can achieve superior results for semi-supervised image classification, even when combined with very standard classifiers. It can be found from the figure and tables that graph-based SSL methods such as HF and AG are not very stable. This is mainly due to their sensitivity to the graph structure, which was observed in [Kingma et al., 2014] as well.

The superior performance of our method can be ascribed to two factors: (1) in addition to the *local-consistency* assumption, our method also exploits the *exotic-consistency* assumption; (2) the discriminative projections abstract high-level attributes from the sampled prototypes, *e.g.* being more "yellow-smooth" than "dark-structured". As already proven in fully supervised scenarios [Farhadi et al., 2009; Quattoni et al., 2008], prototype-linked, attribute-based features are very helpful for image classification.

We further investigate the complementarity of our learned feature with other SSL methods for semi-supervised classification. It is interesting to see from the bottom panel of Table 4.1 that using such combinations boosts the performance also. This suggests that our scheme of exploiting unlabeled data and the previous ones doing so capture complementary information. However, using the standard Logistic Regression generally yields the best results for our learned feature.

Robustness to Parameters

In this section, we examine the influence of the parameters of our method on its classification performance. They are the total number of prototype sets T, the number of prototypes in each set r, the number of images in each prototype n, and the number of skeleton hypotheses m used in Max-Min Sampling. LR was used as the classifier here.



(a) T (b) r (c) n (d) m

***Figure 4.6**: Performance of EP as a function its parameters T, r, n, and m, where LR is employed with 5 labeled training images per class.*

The parameters were evaluated as follows. Each time the value of one changes while the other ones being kept fixed to the values described in the experimental settings.

Figure 4.6 shows the results over a range of their values. The figure shows that the performance of our method increases pretty fast with T, but then stabilizes quickly. It implies that the method benefits from exploiting more "novel" visual attributes (image prototypes). After T increases to some threshold (*e.g.* 50 for the eight datasets), basically no new attributes are added, and performance stops going up much. For r, the figure shows that the performance generally increases with it. This is expected because a large r leads to precise attribute assignment. In other words, a large r generates more prototypes per set, thus increasing the possibility of linking every image to its desirable attribute. However, we seen that when r outpaces 24, the increase is not worth the computing time. A larger r would lead to confusing attributes, as it starts to draw very similar or even identical samples into different prototypes. Also, a large r results in high-dimensional features, which in turn cause over-fitting.

For n, a similar trend was obtained – as n increases, the characteristics of the prototypes are enriched, thus boosting the performance. But beyond some threshold (*e.g.* 10 in our experiments), more noisy images are introduced, thus degrading the performance. One possible solution to further enrich the training samples of each prototype is to perform image transformations such as *translation*, *rotation*, and *scaling* to the seed images, and to add the transformed images into the prototype. This technique of enriching training data has been successfully used recently for image classification [Paulin et al., 2014] and for feature learning [Dosovitskiy et al., 2014]. For m, Figure 4.6 shows that it does not

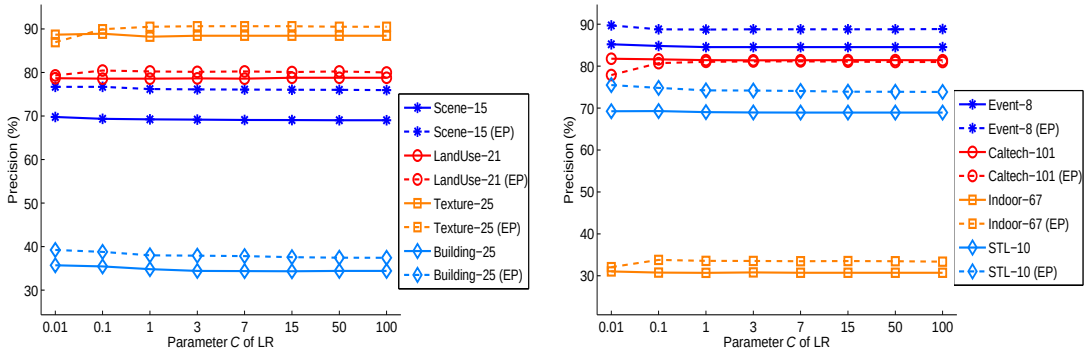


***Figure 4.7**: Comparison of our learned feature to the CNN feature [Chatfield et al., 2014], with different LR models.*

affect the performance as much as the three parameters analyzed so far. This does not mean that there is no need to use the *exotic-consistency* assumption. Instead, it suggests that a random selection of r images from a dataset of $l + u$ images already fulfills the requirement of the assumption: images should be apart from each other. This is generally true because $r << l + u$ holds for the datasets considered.

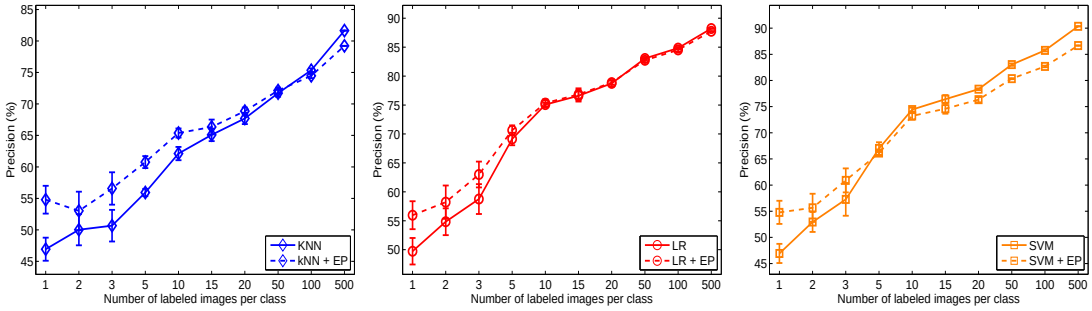
Although the performance of EP will be affected by the choice of its parameters, we can see from Figure 4.6 that each of the parameters has a wide range of reasonable values to choose from. It is not difficult to choose a set of parameter values that produces better results than competing methods (*c.f.* Figure 4.6 and Table 4.1). Also, the parameters are quite intuitive and their roles are similar to the parameters of some other methods: analogues of m, n and T can be found in RANSAC, k-NN, and Bagging, for instance.

Robustness to Classifier Models

In this section, we evaluate the robustness of our learned features against classifier models. Different values of the balancing parameter C between model accuracy and model complexity were tested for the LR classifier across the eight datasets. 5 labeled training examples per class were used. A set of values $\{0.01, 0.1, 1, 5, 15, 50, 100\}$ were tested for the parameter C of LR. Figure 4.7 shows the results. It is evident from the figure that our learned feature consistently outperforms the original CNN feature over a large range of parameter values for the classifier models. This property is important for semi-supervised classification, as labeled data is limited in this scenario and probably cannot afford model selection techniques such as Cross-Validation.

Efficiency

Although additional time is needed for feature learning (the direct use of the original feature needs no training at this stage), our method is efficient. The efficiency is due to



(a) KNN (b) LR (c) SVMs

Figure 4.8: *Self-taught classification results on dataset STL-10, where EP is learned from the unlabeled images. The classifiers were tested with deep features, and our learned feature from it (indicated by "+ EP").*

two reasons: 1) Training logistic regression is very efficient; and 2) the performance of our method stabilizes quickly with respect to T as Figure 4.6 shows. The training on the datasets takes $2-6$ minutes on a Core i5 2.80 GHz desktop PC. Furthermore, our method is inherently parallelizable and can take advantage of multi-core processors. It is worth noting that this extra-training time is compensated by using a simpler classifier such as logistic regression for the classification.

4.5.2 Self-taught Image Classification

In order to evaluate the generality of our method, we tested it in a more general scenario, where the unlabeled data is the set of $100,000$ unlabeled images from the STL-10 dataset. Projection functions were learned from this unlabeled dataset and the performance was tested on the STL-10 dataset. Again, we held the training image and test images as a whole, and chose only a small fraction as training images (for the classifiers) with others as test images for evaluation. The average accuracy of 5 runs with random training-test splits was reported. Figure 4.8 shows the classification performance with different numbers of labeled training images per class. From the figure and table, it can be observed that our learned feature from the random image collection still outperforms the original CNN feature when the number of labeled training images is small. This is a very helpful property for semi-supervised learning, as it happens quite often that one has no prior access to the data to be classified. The success could be ascribed to the fact that the "universal visual world" (the random image collection) contains abundant high-level, valuable visual attributes such as "blue and open" in some image clusters and "textured and man-made" in others. Exploiting these "hidden" visual attributes is very beneficial for narrowing down the semantic gap between low-level features and high-level classification tasks.

However, the figure also shows that as the number of labeled training images increases, the advantage of our learned feature vanishes. The method even produces worse results

Methods	Scene-15	LandUse-21	Texture-25	Building-25	Event-8	Caltech-101	Indoor-67	STL-10
k-means	65.5	56.3	59.2	39.3	76.7	67.7	33.1	57.0
k-means + **EP**	<u>71.5</u>	<u>63.6</u>	**73.1**	**43.8**	**87.3**	<u>69.4</u>	<u>37.0</u>	<u>63.5</u>
Spectral Clustering	69.6	59.8	66.6	33.3	82.7	68.2	31.5	52.8
Spectral Clustering + **EP**	**73.6**	**65.2**	<u>70.1</u>	<u>41.0</u>	<u>86.5</u>	**70.7**	**37.2**	**66.4**

***Table 4.3**: Purity (%) of image clustering on the eight datasets, where the CNN feature [Chatfield et al., 2014] and our learned feature from it (indicated by + EP) are used. The best results are indicated in **bold**, and the second best is <u>underlined</u>.*

than the original CNN feature when the number of training samples is large. This is to be expected as the method is designed to improve classification systems by exploiting unlabeled data. Therefore, when a sufficient number of labeled images are available, introducing additional unlabeled ones may hurt the system. This is a general, open problem for semi-supervised learning (self-taught learning) [Li & Zhou, 2011]. One possible solution is to study when the classification systems should switch from semi-supervised learning to fully supervised learning. Another solution could be to use the labeled training images directly as the skeleton to generate the prototype sets. This strategy, however, is more limited than ours, and is difficult to use for tasks, such as image clustering, where no labeled samples are available. We leave these issues as future work.

4.5.3 Image Clustering

In this section, we evaluated our learned feature for the task of image clustering. Given a collection of images without any labels, the task is to group them so that images in the same group are more (semantically) similar to each other than to those in other groups. We follow existing work [Sivic et al., 2005; Tuytelaars et al., 2010; Dai et al., 2010a, 2012c; Faktor & Irani, 2012] and evaluate the task on the image classification datasets, in particular on the eight datasets used for semi-supervised image classification. To the best of our knowledge, we are the first to evaluate the performance of image clustering on as many as eight standard classification datasets, some of which are still very challenging for supervised image classification. Most clustering methods have been tested only on relatively simple datasets, such as 4, 7 and 20 classes of the Caltech dataset, and 5 classes of the ETH shape dataset.

Since our main aim is to validate whether the proposed learning is able to boost the performance of the original feature for image clustering, we chose two standard clustering algorithms – Spectral Clustering and k-means – to compare the two features. As to the implementation, we use the parallel implementation of [Chen et al., 2011] for Spectral Clustering and the vl-feat library of [Vedaldi & Fulkerson, 2008] for k-means algorithm. Since Spectral Clustering and k-means both require the number of clusters as a parameter,

we set it to the number of semantic classes of the datasets, leading to weakly-supervised image clustering.

Table 4.3 lists the results of the two features when combined with k-means and Spectral Clustering. Purity is used as the evaluation criterion, which measures the percentage of images from the dominant class within their clusters, averaged over all clusters. The dominant class of a cluster is the (semantic) class that has more image members than other classes in the cluster. It is easy to see from the table that features learned by EP outperform the original CNN features for image clustering by a considerable margin. For instance, when k-means is used, EP outperforms the CNN feature by 9.6% on Event-8, and by 6.5% on STL-10; when Spectral Clustering is used, the improvement is 4.0% on Scene-15, and 5.7% on Indoor-67. Again, our feature is learned from the original CNN feature, but goes beyond one single image and captures the *similarity* relationship among images. The superior performance of the learned feature suggests that it is worth some effort to analyze properties of the datasets to learn a better feature representation before performing image clustering. This is useful for the task of clustering, as all the data is available to use from the very beginning. This pre-processing step of analyzing datasets has not yet raised much attention in the community. We hope that this work will stimulate more efforts in this direction.

4.6 Conclusion

This chapter has tackled the problem of feature learning for the task of semi-supervised image classification and image clustering. We proposed as novel concept the *exotic-consistency* assumption and designed a simple, yet effective feature learning method to use it along with *local-consistency* to exploit the available, unlabeled data. By using the assumptions, we generate a diverse set of training data for surrogate classes to learn visual attributes in a discriminative way. By doing so, images are classified and linked to the surrogate classes – images are represented with their affinities to a rich set of discovered image attributes for classification and clustering. Experiments on eight standard datasets showed the superior performance of the learned feature for both semi-supervised image classification and image clustering. In addition, the method is conceptually simple, computationally efficient, and flexible to use. The future work is to extend the method to image segmentation and image semantic segmentation.

5

Fast Training Algorithms for SVM+

The advance speed of a field such as computer vision is partly determined by how fast researchers can evaluate new ideas for new tasks, and/or on new data. One of the main obstacles on the way is the time-consuming training process, given the fact that a large portion of vision problems are highly complex. Thus, simplified training algorithms to standard approaches are of a strong need. In this chapter, we examine the training of algorithms in Learning with Privileged Information, and propose fast training algorithms for the standard algorithm SVM+, making the training of this task more affordable.

5.1 Introduction

Many computer vision tasks contain privileged information that only exists in the training data, and not available during the test stage. For example, the training images of many datasets for image recognition are annotated with priviledged information such as *attributes*, *object bounding boxes*, *textual descriptions*, *depth information*. Although the raw test images in the real-world applications are not associated with such information, it has been demonstrated that such information is useful for learning classifiers with better recognition performance [Feyereisl et al., 2015; Lampert et al., 2013; Sharmanska et al., 2013; Vapnik & Vashist, 2009; Wang & Ji, 2015; Zhang et al., 2014b].

This problem is known as the *Learning Using Privileged Information (LUPI)* problem [Vapnik & Vashist, 2009]. Different from the traditional learning paradigm, where the training data and the test data have the same representation, LUPI leverages training data containing additional information that is only available during the training process, not in the testing process. Such additional information in the training data is referred to as privileged information or hidden information.

Following the LUPI paradigm, Vapnik and Vashist [Vapnik & Vashist, 2009] proposed an SVM-like algorithm called *SVM+*, in which they replace the slack variables in the standard SVM with a slack function defined in the privileged feature space. Through the

slack function, the additional privileged information is used to model the loss function, which guides the hyperplane learning in the main feature space. In contrast, the slack variables in the standard SVM are only constrained to non-negative values, which is often less effective than the slack function in SVM+.

Although SVM+ can be formulated as a quadratic programming problem in the dual form similarly as the standard SVM, it is still non-trivial to efficiently solve it, because the introduction of the slack function leads to more constraints, and makes the number of dual variables doubled. While an SMO-style algorithm was developed in [Pechyony et al., 2010], the working set selection method is complicated, and the algorithm is also slow in practice. Moreover, it is unclear how to apply it to linear SVM+ without calculating the kernel matrix, which is becoming more crucial, due to rapidly increasing data in real-world applications.

In this work, we propose two efficient algorithms for solving linear SVM+ and kernel SVM+, respectively. In particular, inspired by linear SVM [Hsieh et al., 2008], we augment the feature vector with an additional constant element, and absorb the bias term in the decision function into the weight vector, which leads to a dual form with simpler constraints. The new linear SVM+ formulation can be efficiently solved by using the dual coordinate descent method, in a similar way to linear SVM. For kernel SVM+, we further propose to apply the ℓ_2-loss, which leads to a simpler optimization problem in the dual form with only half of dual variables when compared with the original SVM+. More interestingly, we show that the resultant dual form is in analogy to one-class SVM, which can thus be efficiently solved using the efficient sequential minimal optimization (SMO) algorithm [Platt, 1998] with the existing state-of-the-art SVM solvers such as LIBSVM [Fan et al., 2005; Chang & Lin, 2011].

We implement our algorithms for linear and kernel SVM+ based on LIBLINEAR [Fan et al., 2008] and LIBSVM [Chang & Lin, 2011], respectively. We conduct extensive experiments on three tasks: digit recognition on the MNIST+ dataset [Vapnik & Vashist, 2009], scene recognition on the Scene-15 dataset [Lazebnik et al., 2006b], and web image retrieval on the NUS-WIDE dataset [Chua et al., 2009]. The results demonstrate that our proposed algorithms achieve significant speed-up than the state-of-the-art solvers for solving the linear and kernel SVM+ problems.

5.2 Related Work

Learning Using Privileged Information (LUPI), as a new learning paradigm, was first proposed by Vapnik and Vashist [Vapnik & Vashist, 2009], therein with the SVM+ algorithm. After that, many variants of SVM+ have been proposed for solving different tasks [Liang & Cherkassky, 2008; Fouad et al., 2013; Xu et al., 2015c; Sharmanska et al., 2013; Wang & Ji, 2015; Li et al., 2014]. In [Liang & Cherkassky, 2008], Liang and Cherkassky

developed a multi-task learning approach based on SVM+. In [Ji et al., 2012], a multi-task multi-class extension of SVM+ was proposed. Fouad *et al.* [Fouad et al., 2013] designed a two-step approach for metric learning, and Xu *et al.* [Xu et al., 2015c] formulated a convex formulation for metric learning using privileged information based on the information theory metric learning (ITML) method. Sharmanska *et al.* [Sharmanska et al., 2013] proposed the Rank Transfer method for utilizing privileged information, and demonstrated the effectiveness of privileged information in various computer vision tasks. In [Li et al., 2014], Li *et al.* extended SVM+ to the multi-instance learning scenario for image retrieval and object recognition by learning using web data. In [Wang & Ji, 2015], Wang *et al.* proposed a classifier learning algorithm for utilizing privileged information. In [Feyereisl et al., 2015], Feyereisl *et al.* extended structure SVM to exploiting privileged information for object localization.

Vapnik and Vashist also showed that SVM+ has a faster convergence rate than the standard SVM method under certain conditions [Vapnik & Vashist, 2009]. A more thorough theoretic study of SVM+ can be found in [Pechyony & Vapnik, 2010]. In [Vapnik & Izmailov, 2015], two mechanisms for LUPI are further explained. Lapin *et al.* [Lapin et al., 2014] also discovered the relationship between SVM+ and weighted SVM, while Li *et al.* [Li et al., 2014] discussed the connection of the unsupervised domain adaptation method and SVM+.

One closely related work is [Pechyony et al., 2010], in which an SMO-style algorithm was developed for the SVM+ problem. However, as the two sets of dual variables introduced for the constraints related to the main and privileged features in the primal form are tangled together in the constraints of dual problem, it is non-trivial to design the working set selection method for the SMO algorithm, and leads to a complicated algorithm that is less efficient in practice. Moreover, it is also unclear that how to apply the the SMO-style algorithm for solving the linear SVM+ problem without calculating the kernel matrix. In contrast, in this work, by absorbing the bias term into the weight vector of the SVM+ classifier, we arrive at an optimization problem with simpler constraints in the dual form. Thus, the conventional dual coordinate descent algorithm can be applied to efficiently solve linear SVM+. Moreover, for the SVM+ form, we further apply the ℓ_2-loss and obtain a smaller dual problem with only a half of dual variables of the original SVM+ formulation. The new dual form shares a similar formulation with one-class SVM, and thus can be solved by using the SMO algorithm implemented in the existing SVM solvers such as LIBSVM [Chang & Lin, 2011] without developing any new specific SMO algorithm. We demonstrate that our proposed two algorithms are more efficient than the SMO-style algorithm in [Pechyony et al., 2010].

In a more broader context, there are works proposed for solving tasks in which training and test data use asymmetric information source, which are less related to SVM+ [Chen et al., 2012, 2014; Dai et al., 2015b; Ding et al., 2014; Gupta et al., 2015; Hinton et al., 2014; Hoffman et al., 2014; Lampert et al., 2013; Srivastava & Salakhutdinov, 2012; Zhang et al., 2014b].

5.3 Learning Using Privileged Information

In the following, we denote a vector (*resp.*, matrix) with a lower (*resp.*, upper) case letter in boldface. For example, $\mathbf{a}$ represents a vector, and $\mathbf{A}$ a matrix. The transpose of a vector or matrix is denoted by using the superscript $'$. We use $\mathbf{0}_n, \mathbf{1}_n \in \mathbb{R}^n$ to represent the column vector with n zeros and ones, respectively. We also simply use $\mathbf{0}$ and $\mathbf{1}$ instead of $\mathbf{0}_n$ and $\mathbf{1}_n$, when the dimensionality is obvious. Moreover, $\mathbf{a} \circ \mathbf{b}$ (*resp.*, $\mathbf{A} \circ \mathbf{B}$) denotes the element-wise product between two vectors (*resp.*, matrices).

In the Learning Using Privileged Information (LUPI) paradigm [Vapnik & Vashist, 2009], the training samples contain additional information that is not available in the test stage. Formally, we represent the training data as $\{(\mathbf{x}_i, \tilde{\mathbf{x}}_i, y_i)|i = 1, \ldots, n\}$, where n is the total number of training samples, $\mathbf{x}_i \in \mathbb{R}^D$ is the main feature vector of the i-th training sample with D being the feature dimensionality, $\tilde{\mathbf{x}} \in \mathbb{R}^{\tilde{D}}$ is the corresponding privileged feature vector with $\tilde{D}$ being its dimension. The goal of LUPI is to learn a decision function $f(\mathbf{x})$ for classifying any test sample $\mathbf{x} \in \mathbb{R}^D$ in the main feature space.

In [Vapnik & Vashist, 2009], an SVM based approach called *SVM+* was proposed. Similar to SVM, the decision function in SVM+ is represented as $f(\mathbf{x}) = \mathbf{w}'\phi(\mathbf{x}) + b$, where $\phi(\,\cdot\,)$ is a feature mapping induced by the kernel on training data, $\mathbf{w}$ is the weight vector, and b is the bias term. The objective of SVM+ can be written as,

$$\min_{\tilde{\mathbf{w}},\tilde{b},\mathbf{w},b} \frac{1}{2}\left(\|\mathbf{w}\|^2 + \gamma\|\tilde{\mathbf{w}}\|^2\right) + C\sum_{i=1}^{n}\xi(\tilde{\mathbf{w}}, \tilde{b}, \psi(\tilde{\mathbf{x}}_i)) \tag{5.1}$$

$$\text{s.t. } y_i(\mathbf{w}'\phi(\mathbf{x}_i) + b) \geq 1 - \xi(\tilde{\mathbf{w}}, \tilde{b}, \psi(\tilde{\mathbf{x}}_i)), \tag{5.2}$$

$$\xi(\tilde{\mathbf{w}}, \tilde{b}, \psi(\tilde{\mathbf{x}}_i)) \geq 0, \tag{5.3}$$

where $\xi(\tilde{\mathbf{w}}, \tilde{b}, \psi(\tilde{\mathbf{x}}_i)) = \tilde{\mathbf{w}}'\psi(\tilde{\mathbf{x}}) + \tilde{b}$ is the slack function defined in the privileged feature space, $\psi(\,\cdot\,)$ is the feature mapping function for the privileged features, and $\tilde{\mathbf{w}}$ and $\tilde{b}$ are respectively the weight vector and bias term for the slack function. It can be observed that SVM+ replaces the slack variables in the standard SVM formulation with the slack function $\xi(\tilde{\mathbf{w}}, \tilde{b}, \psi(\tilde{\mathbf{x}}_i))$. As a result, the loss occurred by the training samples $\mathbf{x}_i$ can be regularized by the privileged information $\tilde{\mathbf{x}}_i$, therefore the hyperplane of SVM+ classifier can be tuned by using the privileged information during the training process.

Let us introduce two sets of dual valuables $\{\alpha_i|_{i=1}^n\}$ and $\{\zeta_i|_{i=1}^n\}$ for the constraints in Equation 5.2 and Equation 5.3, respectively, and also denote two vectors $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]' \in \mathbb{R}^n$ and $\boldsymbol{\zeta} = [\zeta_1, \ldots, \zeta_n]' \in \mathbb{R}^n$. We arrive at the dual form of SVM+,

$$\begin{aligned}\min_{(\boldsymbol{\zeta},\boldsymbol{\alpha})\in\mathcal{A}} \quad & \frac{1}{2}(\boldsymbol{\alpha}\circ\mathbf{y})'\mathbf{K}(\boldsymbol{\alpha}\circ\mathbf{y}) - \mathbf{1}'\boldsymbol{\alpha} \\ & + \frac{1}{2\gamma}(\boldsymbol{\alpha} + \boldsymbol{\zeta} - C\mathbf{1})'\tilde{\mathbf{K}}(\boldsymbol{\alpha} + \boldsymbol{\zeta} - C\mathbf{1}),\end{aligned} \tag{5.4}$$

where $\mathcal{A} = \{(\boldsymbol{\zeta}, \boldsymbol{\alpha}) | \mathbf{y}'\boldsymbol{\alpha} = 0, \mathbf{1}'(\boldsymbol{\alpha} + \boldsymbol{\zeta} - C\mathbf{1}) = 0, \boldsymbol{\alpha} \geq 0, \boldsymbol{\zeta} \geq 0\}$ is the feasible set of $(\boldsymbol{\alpha}, \boldsymbol{\zeta})$, $\mathbf{K} \in \mathbb{R}^{n \times n}$ is the kernel matrix based on the main training features with each element being $K_{ij} = \phi(\mathbf{x}_i)'\phi(\mathbf{x}_j)$, and $\tilde{\mathbf{K}} \in \mathbb{R}^{n \times n}$ is the kernel matrix based on the privileged features with each element being $\tilde{K}_{ij} = \psi(\tilde{\mathbf{x}}_i)'\psi(\tilde{\mathbf{x}}_j)$. After solving the above problem, the weight vectors $\mathbf{w}$ and $\tilde{\mathbf{w}}$ can be reconstructed based on the Karush-Kuhn-Tucker (KKT) conditions as,

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \phi(\mathbf{x}_i), \tag{5.5}$$

$$\tilde{\mathbf{w}} = \frac{1}{\gamma} \sum_{i=1}^{n} (\alpha_i + \zeta_i - C)\psi(\tilde{\mathbf{x}}_i). \tag{5.6}$$

Although the dual problem in Equation 5.4 is a quadratic programming problem, solving it with the existing optimization toolboxes is certainly undesirable for real-world visual recognition tasks. However, it is non-trivial to develop efficient algorithm like the SMO algorithm for solving the standard SVM. The main difficulty comes from the constraint $\mathbf{1}'(\boldsymbol{\alpha} + \boldsymbol{\zeta} - C\mathbf{1}) = 0$, in which two sets of dual variables are tangled together, which makes the working set selection method become complicated [Pechyony et al., 2010].

5.4 Dual Coordinate Descent Algorithm for Solving Linear SVM+

In this section, we present the dual coordinate descent method for solving the linear SVM+. In particular, we absorb the bias term into the weight vector $\mathbf{w}$ by appending a constant entry of 1 to each feature vector, *i.e.* , $\mathbf{x} \leftarrow [\mathbf{x}', 1]'$, and $\mathbf{w} \leftarrow [\mathbf{w}', b]'$ for the main features, and $\tilde{\mathbf{x}} \leftarrow [\tilde{\mathbf{x}}', 1]'$, and $\tilde{\mathbf{w}} \leftarrow [\tilde{\mathbf{w}}', \tilde{b}]'$ for privileged information. For ease of presentation, we still use $\mathbf{w}$ and $\mathbf{x}$ (*resp.*, $\tilde{\mathbf{w}}$, $\tilde{\mathbf{x}}$) to represent the weight vector and feature vector for the main (*resp.*, privileged) features.

Now, the objective of linear SVM+ can be reformulated as follows,

$$\begin{aligned} \min_{\tilde{\mathbf{w}}, \mathbf{w}} \quad & \frac{1}{2}\left(\|\mathbf{w}\|^2 + \gamma\|\tilde{\mathbf{w}}\|^2\right) + C\sum_{i=1}^{n} \tilde{\mathbf{w}}'\tilde{\mathbf{x}} \\ \text{s.t.} \quad & y_i \mathbf{w}'\mathbf{x}_i \geq 1 - \tilde{\mathbf{w}}'\tilde{\mathbf{x}}, \\ & \tilde{\mathbf{w}}'\tilde{\mathbf{x}}_i \geq 0, \end{aligned} \tag{5.7}$$

and its dual form can be written as,

$$\begin{aligned} \min_{(\boldsymbol{\zeta}, \boldsymbol{\alpha}) \in \mathcal{A}} \quad & \frac{1}{2}(\boldsymbol{\alpha} \circ \mathbf{y})'\mathbf{K}(\boldsymbol{\alpha} \circ \mathbf{y}) - \mathbf{1}'\boldsymbol{\alpha} \\ & + \frac{1}{2\gamma}(\boldsymbol{\alpha} + \boldsymbol{\zeta} - C\mathbf{1})'\tilde{\mathbf{K}}(\boldsymbol{\alpha} + \boldsymbol{\zeta} - C\mathbf{1}), \end{aligned} \tag{5.8}$$

where the feasible set becomes $\mathcal{A} = \{(\boldsymbol{\zeta}, \boldsymbol{\alpha}) | \boldsymbol{\alpha} \geq 0, \boldsymbol{\zeta} \geq 0\}$, $\mathbf{K} \in \mathbb{R}^{n \times n}$ is the linear kernel matrix of main features defined as $K_{ij} = \mathbf{x}_i' \mathbf{x}_j$, and $\tilde{\mathbf{K}} \in \mathbb{R}^{n \times n}$ is the linear kernel matrix of privileged features defined as $\tilde{K}_{ij} = \tilde{\mathbf{x}}_i' \tilde{\mathbf{x}}_j'$. Compared to the dual form of the original SVM+ formulation Equation 5.4, the objective function remains the same, but we do not have the constraints $\mathbf{y}'\boldsymbol{\alpha} = 0, \mathbf{1}'(\boldsymbol{\alpha} + \boldsymbol{\zeta} - C\mathbf{1}) = 0$ in Equation 5.8, after absorbing of the bias terms in Equation 5.7. The weight vectors $\mathbf{w}$ and $\tilde{\mathbf{w}}$ can be represented using the dual variables in the same way as in Equation 5.5 and Equation 5.6 without using the feature mapping functions (*i.e.* , we can set $\phi(\mathbf{x}_i) = \mathbf{x}_i$ and $\psi(\tilde{\mathbf{x}}_i) = \tilde{\mathbf{x}}_i$.

Let us define $\mathbf{Q} = \begin{bmatrix} \mathbf{K} \circ (\mathbf{y}\mathbf{y}') + \frac{1}{\gamma}\tilde{\mathbf{K}} & \frac{1}{\gamma}\tilde{\mathbf{K}} \\ \frac{1}{\gamma}\tilde{\mathbf{K}} & \frac{1}{\gamma}\tilde{\mathbf{K}} \end{bmatrix} \in \mathbb{R}^{2n \times 2n}$, $\boldsymbol{\beta} = [\boldsymbol{\alpha}', \boldsymbol{\zeta}']' \in \mathbb{R}^{2n}$, $\mathbf{e} = [(\mathbf{1} + \frac{C}{\gamma}\tilde{\mathbf{K}}\mathbf{1})', (\frac{C}{\gamma}\tilde{\mathbf{K}}\mathbf{1})']' \in \mathbb{R}^{2n}$, the problem in Equation 5.8 can be rewritten in a more concise form as,

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & \frac{1}{2}\boldsymbol{\beta}'\mathbf{Q}\boldsymbol{\beta} - \mathbf{e}'\boldsymbol{\beta} \\ \text{s.t.} \quad & \boldsymbol{\beta} \geq 0. \end{aligned} \tag{5.9}$$

Now we develop a coordinate descent algorithm for the problem in Equation 5.9 similarly as that for linear SVM [Hsieh et al., 2008]. In the dual coordinate descent algorithm, we iteratively update one entry of $\boldsymbol{\beta}$ each time. This process is repeated until the stopping criterion is reached.

In particular, let us denote $f(\boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{\beta}'\mathbf{Q}\boldsymbol{\beta} - \mathbf{e}'\boldsymbol{\beta}$, and we propose to update the i-th entry as $\beta_i \leftarrow \beta_i + d$, where d is a scalar variable that needs to be solved. By defining $\boldsymbol{\delta}^{(i)} = [0, \ldots, 0, 1, 0, \ldots, 0]$ as the vector with all zeros except the i-th entry being one, the subproblem for solving d can be formulated as

$$\min_{d} f(\boldsymbol{\beta} + d\boldsymbol{\delta}^{(i)}) \quad \text{s.t. } \boldsymbol{\beta} + d\boldsymbol{\delta}^{(i)} \geq 0, \tag{5.10}$$

which has an analytic solution as follows,

$$d = \max(-\beta_i, -\frac{\nabla_i f(\boldsymbol{\beta})}{Q_{ii}}) \tag{5.11}$$

where Q_{ii} is the (i, i)-th entry of the matrix $\mathbf{Q}$ in Equation 5.9, and $\nabla_i f(\boldsymbol{\beta})$ is the gradient of $f(\boldsymbol{\beta})$ *w.r.t.* β_i.

Now we discuss how to efficiently calculate the right-hand side of Equation 5.11. The scalar Q_{ii} can be pre-computed efficiently, so the major problem is the calculation of $\nabla_i f(\boldsymbol{\beta})$. In particular, the gradient of $f(\boldsymbol{\beta})$ can be written as $\nabla f(\boldsymbol{\beta}) = \mathbf{Q}\boldsymbol{\beta} - \mathbf{e}$. Then, for any given index i, the gradient of $f(\boldsymbol{\beta})$ *w.r.t.* β_i can be calculate as,

$$\nabla_i f(\boldsymbol{\beta}) = (\mathbf{Q}\boldsymbol{\beta})_i - e_i = \sum_{j=1}^{2n} Q_{ij}\beta_j - e_i, \tag{5.12}$$

where Q_{ij} is the (i,j)-th element of the matrix $\mathbf{Q}$, and e_i is the i-th element of the vector $\mathbf{e}$.

We discuss the detailed calculations in two cases. Let us denote a matrix $\mathbf{H} = \mathbf{K} \circ \mathbf{y}\mathbf{y}' \in \mathbb{R}^{n \times n}$. For $\forall i = 1, \ldots, n$, we have

$$\sum_{j=1}^{2n} Q_{ij}\beta_j = \sum_{j=1}^{n} H_{ij}\alpha_j + \frac{1}{\gamma}\sum_{j=1}^{n} \tilde{K}_{ij}\alpha_j + \frac{1}{\gamma}\sum_{j=1}^{n} \tilde{K}_{ij}\zeta_j$$

where H_{ij} is the (i,j)-th element of the matrix $\mathbf{H}$. We also have $e_i = 1 + \frac{C}{\gamma}\sum_{j=1}^{n} \tilde{K}_{ij}$. By applying the KKT conditions *w.r.t.* $\mathbf{w}$ and $\tilde{\mathbf{w}}$ in Equation 5.5 and Equation 5.6, the gradient of f *w.r.t.* β_i in Equation 5.12 can be written as,

$$\nabla_i f(\boldsymbol{\beta})_i = y_i \mathbf{w}'\mathbf{x}_i - 1 + \tilde{\mathbf{w}}'\tilde{\mathbf{x}}_i, \quad \forall 1 \leq i \leq n \tag{5.13}$$

which takes $O(D + \tilde{D})$ time complexity, and it can be further speed-up if the feature vectors $\mathbf{x}$ and $\tilde{\mathbf{x}}$ are sparse.

Similarly, for $\forall i = n+1, \ldots, 2n$, we have

$$\sum_{j=1}^{2n} Q_{ij}\beta_j \quad = \quad \frac{1}{\gamma}\sum_{j=1}^{n} \tilde{K}_{ij}\alpha_j + \frac{1}{\gamma}\sum_{j=1}^{n} \tilde{K}_{ij}\zeta_j,$$

and $e_i = \frac{C}{\gamma}\sum_{j=1}^{n} \tilde{K}_{ij}$. By using the KKT condition for $\tilde{\mathbf{w}}$ in Equation 5.6, the gradient in Equation 5.12 can be written as,

$$\nabla_i f(\boldsymbol{\beta})_i = \tilde{\mathbf{w}}'\tilde{\mathbf{x}}_i, \quad \forall n+1 \leq i \leq 2n, \tag{5.14}$$

which takes $O(\tilde{D})$ time complexity only, and can also be further speed-up if the feature vector $\tilde{\mathbf{x}}$ is sparse. This completes the calculation of Equation 5.11.

After solving d, we can update the i-th dual variable β_i by using $\beta_i \leftarrow \beta_i + d$. Considering only one dual variable is modified each time, the weight vectors can be updated efficiently at each iteration. Note that we have $\alpha_i = \beta_i$ when $1 \leq i \leq n$, and $\zeta_{i-n} = \beta_i$, when $n+1 \leq i \leq n$. Based on Equation 5.5 and Equation 5.6, the updating rules for $\mathbf{w}$ and $\tilde{\mathbf{w}}$ can be written as

$$\mathbf{w} \quad \leftarrow \quad \mathbf{w} + dy_i\mathbf{x}_i, \quad \text{if } 1 \leq i \leq n \tag{5.15}$$

$$\tilde{\mathbf{w}} \quad \leftarrow \quad \tilde{\mathbf{w}} + \frac{1}{\gamma}d\tilde{\mathbf{x}}_i, \quad \text{if } 1 \leq i \leq 2n \tag{5.16}$$

We summarize our dual coordinate descent algorithm for solving linear SVM+ in Algorithm 3. We first initialize the weight vectors as $\mathbf{w} = \mathbf{0}_D$, and $\tilde{\mathbf{w}} = -\frac{C}{\gamma}\sum_{i=1}^{n} \tilde{\mathbf{x}}_i$ (*i.e.* , the solution of $\mathbf{w}$ and $\tilde{\mathbf{w}}$ when $\boldsymbol{\alpha} = \mathbf{0}$ and $\boldsymbol{\zeta} = \mathbf{0}$). Then, each time, we randomly pick up an index i, and solve d (*i.e.* , the change of β_i) using Equation 5.11. In particular,

Algorithm 3: Dual coordinate descent algorithm for solving the linear SVM+ problem in Equation 5.7

Input: $\{(\mathbf{x}_i, \tilde{\mathbf{x}}_i, y_i)|_{i=1}^n\}$, C, and γ.

Initialize $\mathbf{w} = \mathbf{0}$, and $\tilde{\mathbf{w}} = -\frac{C}{\gamma}\sum_{i=1}^n \tilde{\mathbf{x}}_i$.
Set $Q_{ii} = \mathbf{x}_i'\mathbf{x}_i$ for $1 \leq i \leq n$, and $Q_{ii} = \tilde{\mathbf{x}}_i'\tilde{\mathbf{x}}_i$ for $n+1 \leq i \leq 2n$.
repeat
  Randomly pick an index i.
  if $1 \leq i \leq n$ **then**
    Calculate $\nabla_i f(\boldsymbol{\beta})$ using Equation 5.13.
  else
    Calculate $\nabla_i f(\boldsymbol{\beta})$ using Equation 5.14.
  end if
  Calculate d using Equation 5.11 based on Q_{ii} and $\nabla_i f(\boldsymbol{\beta})$.
  if $1 \leq i \leq n$ **then**
    Update $\mathbf{w}$ using Equation 5.15.
  end if
  Update $\tilde{\mathbf{w}}$ using Equation 5.16.
until The convergence criterion is reached.

Output: Weight vectors $\mathbf{w}$ and $\tilde{\mathbf{w}}$.

when $1 \leq i \leq n$, we calculate the gradient $\nabla_i f(\boldsymbol{\beta})$ using Equation 5.13, and update the weight vectors $\mathbf{w}$ and $\tilde{\mathbf{w}}$ using Equation 5.15 and Equation 5.16, respectively; when $n+1 \leq i \leq 2n$, we calculate the gradient $\nabla_i f(\boldsymbol{\beta})$ using Equation 5.14 and update the weight vector $\tilde{\mathbf{w}}$ only using Equation 5.16. The above process is repeated until the objective converges. Actually, the problem in Equation 5.9 can be treated as a special form of the linear SVM problem discussed in [Hsieh et al., 2008], so the convergence of our algorithm follows that of the dual coordinate descent algorithm for linear SVM. We implement our algorithm based on the LIBLINEAR software [Fan et al., 2008].

5.5 SMO Algorithm for Solving Kernel SVM+

In this section, we develop an efficient algorithm for solving kernel SVM+ based on the ρ-SVM formulation. Similar to linear SVM+, we also augment the feature vector in the nonlinear feature space, so we can absorb the bias term into the weight vector, *i.e.* , $\phi(\mathbf{x}_i) \leftarrow [\phi(\mathbf{x}_i)', 1]'$ and $\mathbf{w} \leftarrow [\mathbf{w}', b]'$ for the main features (*resp.*, $\psi(\tilde{\mathbf{x}}_i) \leftarrow [\psi(\tilde{\mathbf{x}}_i)', 1]'$ and $\tilde{\mathbf{w}} \leftarrow [\tilde{\mathbf{w}}', \tilde{b}]'$ for the privileged features)[1]. For ease of presentation, we still use $\phi(\mathbf{x}_i)$

[1]Note we usually do not know the explicit form of $\phi(\cdot)$, but rather the inner product of the augmented features can be simply calculated by adding one, *i.e.* , $\phi(\mathbf{x}_i)'\phi(\mathbf{x}_j) \leftarrow \phi(\mathbf{x}_i)'\phi(\mathbf{x}_j) + 1$, which also means

and $\mathbf{w}$ to denote the nonlinear feature and weight vector for the main features (*resp.*, $\psi(\tilde{\mathbf{x}}_i)$ and $\tilde{\mathbf{w}}$ for privileged features) below.

Specifically, after using the augmented features, the decision function can be represented as $f(\mathbf{x}) = \mathbf{w}'\phi(\mathbf{x})$. We use the squared hinge loss for ρ-SVM, which leads to the ℓ_2 loss ρ-SVM formulation as follows,

$$\begin{aligned} \min_{\mathbf{w},b,\rho} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{2}C\sum_{i=1}^{n}\xi_i^2 - \rho \\ \text{s.t.} \quad & y_i(\mathbf{w}'\phi(\mathbf{x}_i)) \geq \rho - \xi_i. \end{aligned} \tag{5.17}$$

By replacing each slack variable ξ_i with the slack function $\xi(\tilde{\mathbf{x}}_i) = \tilde{\mathbf{w}}'\psi(\tilde{\mathbf{x}}_i)$, we arrive at the primal form of ℓ_2-SVM+ as follows,

$$\begin{aligned} \min_{\tilde{\mathbf{w}},\mathbf{w},\rho} \; & \frac{1}{2}\left(\|\mathbf{w}\|^2 + \gamma\|\tilde{\mathbf{w}}\|^2\right) + \frac{1}{2}C\sum_{i=1}^{n}\left(\tilde{\mathbf{w}}'\psi(\tilde{\mathbf{x}}_i)\right)^2 - \rho \\ \text{s.t.} \; & y_i(\mathbf{w}'\phi(\mathbf{x}_i)) \geq \rho - \tilde{\mathbf{w}}'\psi(\tilde{\mathbf{x}}_i). \end{aligned} \tag{5.18}$$

Now we derive the dual form of our ℓ_2-SVM+ formulation in Equation 5.18. By introducing dual variables $\alpha_1, \ldots, \alpha_n$ for the constraints in Equation 5.18, we write its Lagrangian as follows,

$$\begin{aligned} \mathcal{L} = & \frac{1}{2}\left(\|\mathbf{w}\|^2 + \gamma\|\tilde{\mathbf{w}}\|^2\right) + \frac{1}{2}C\sum_{i=1}^{n}\left(\tilde{\mathbf{w}}'\psi(\tilde{\mathbf{x}}_i)\right)^2 - \rho \\ & - \sum_{i=1}^{n}\alpha_i\left(y_i(\mathbf{w}'\phi(\mathbf{x}_i)) - \rho + \tilde{\mathbf{w}}'\psi(\tilde{\mathbf{x}}_i)\right), \end{aligned} \tag{5.19}$$

Let us denote a vector $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]'$. By setting the derivatives of Equation 5.19 *w.r.t.* to the primal variables $\mathbf{w}, \tilde{\mathbf{w}}, \rho$ to zeros, we obtain the constraint $\boldsymbol{\alpha}'\mathbf{1} = 1$ as well as two KKT conditions for $\mathbf{w}$ and $\tilde{\mathbf{w}}$ as follows,

$$\mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \phi(\mathbf{x}_i), \tag{5.20}$$

$$\tilde{\mathbf{w}} = \sum_{i=1}^{n}\alpha_i\left(\gamma\mathbf{I} + C\mathbf{P}\mathbf{P}'\right)^{-1}\psi(\tilde{\mathbf{x}}_i), \tag{5.21}$$

where $\mathbf{P} = [\psi(\tilde{\mathbf{x}}_1), \ldots, \psi(\tilde{\mathbf{x}}_n)]$ is the data matrix of privileged features in the nonlinear feature space.

---

the kernel matrix based on the augmented features can be calculated by $\mathbf{K} \leftarrow \mathbf{K} + \mathbf{1}\mathbf{1}'$. The same approach can be applied to the privileged features.

Algorithm 4: Algorithm for solving the ℓ_2-SVM+ problem in Equation 5.18

Input: $\mathbf{K}, \tilde{\mathbf{K}} \in \mathbb{R}^{n \times n}$, C, and γ.

1: Calculate $\mathbf{G} = \frac{1}{C}\mathbf{I} - \frac{1}{C}\left(\mathbf{I} + \frac{C}{\gamma}\tilde{\mathbf{K}}\right)^{-1}$.

2: Set $\mathbf{Q} = \mathbf{H} + \mathbf{G}$, and $\nu = 1/n$.

3: Obtain $\boldsymbol{\alpha}$ by solving the problem in Equation 5.23 with the one-class SVM solver in LIBSVM.

Output: Dual variable vector $\boldsymbol{\alpha}$.

Substituting the two equations in Equation 5.20 and Equation 5.21 into the Lagrangian in Equation 5.19, we arrive at the dual form of Equation 5.18 as follows,

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}'(\mathbf{H} + \mathbf{G})\boldsymbol{\alpha} \\ \text{s.t.} \quad & \mathbf{1}'\boldsymbol{\alpha} = 1, \quad \boldsymbol{\alpha} \geq 0, \end{aligned} \tag{5.22}$$

where $\mathbf{G} = \mathbf{P}'\left(\gamma\mathbf{I} + C\mathbf{P}\mathbf{P}'\right)^{-1}\mathbf{P}$, $\mathbf{H} = \mathbf{K} \circ (\mathbf{y}\mathbf{y}')$, and $\mathbf{K}$ is the kernel matrix of augmented main features with $K_{ij} = \phi(\mathbf{x}_i)'\phi(\mathbf{x}_j)$ being its (i,j)-th element. Based on the equation $\left(\gamma\mathbf{I} + C\mathbf{P}\mathbf{P}'\right)^{-1}\mathbf{P} = \mathbf{P}\left(\gamma\mathbf{I} + C\mathbf{P}'\mathbf{P}\right)^{-1}$, we have $\mathbf{G} = \mathbf{P}'\mathbf{P}\left(\gamma\mathbf{I} + C\mathbf{P}'\mathbf{P}\right)^{-1} = \tilde{\mathbf{K}}\left(\gamma\mathbf{I} + C\tilde{\mathbf{K}}\right)^{-1}$, where $\tilde{\mathbf{K}}$ is the kernel matrix of augmented privileged features with $\tilde{K}_{ij} = \psi(\mathbf{x}_i)'\psi(\mathbf{x}_j)$ being its (i,j)-th element. By further applying the Woodbury Identity, we obtain $\mathbf{G} = \frac{1}{C}\mathbf{I} - \frac{1}{C}\left(\mathbf{I} + \frac{C}{\gamma}\tilde{\mathbf{K}}\right)^{-1}$.

Note the problem in Equation 5.22 is a quadratic programming problem with the number of dual variables being n, which is only half number of variables in the dual form of the original SVM+ method in Equation 5.4. Moreover, the constraints in Equation 5.22 are also simpler, so it can be solved by using the efficient SMO algorithm. Actually, the problem in Equation 5.22 shares a similar form with one-class SVM in the LIBSVM software [Chang & Lin, 2011]. In particular, we write the dual form of one-class SVM as follows (Eqn. (8) in [Chang & Lin, 2011]),

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}'\mathbf{Q}\boldsymbol{\alpha} \\ \text{s.t.} \quad & \mathbf{1}'\boldsymbol{\alpha} = \nu n, \quad 0 \leq \boldsymbol{\alpha} \leq 1, \end{aligned} \tag{5.23}$$

where $\mathbf{Q}$ is the kernel matrix in one-class SVM, ν is a predefined parameter, n is total the number of training samples. Note the constraints $\boldsymbol{\alpha} \geq \mathbf{0}$ and $\mathbf{1}'\boldsymbol{\alpha} = 1$ in Equation 5.22 imply $0 \leq \boldsymbol{\alpha} \leq 1$. By setting $\mathbf{Q} = (\mathbf{H} + \mathbf{G})$ and $\nu = \frac{1}{n}$, the dual problem of ℓ_2-SVM+ in Equation 5.22 can be converted to the optimization problem in Equation 5.23. Therefore, we ignore the details of the SMO algorithm here, and employ the SMO implementation in LIBSVM to solve our ℓ_2-SVM+ problem.

The detailed algorithm for solving ℓ_2-SVM+ is described in Algorithm 4. We first calculate the matrix $\mathbf{G}$ using the kernel matrix of privileged features. Then we call the

one-SVM solver in LIBSVM to obtain the dual variable vector $\boldsymbol{\alpha}$. Finally, the decision function can be written as $\mathbf{f}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i \phi(\mathbf{x}_i)' \phi(\mathbf{x})$, where $\mathbf{x}$ is the test sample, and $\phi(\cdot)$ is the augmented nonlinear feature vector.

Although in our algorithm a matrix inverse operator is involved when calculating the matrix $\mathbf{G}$, the size of $\mathbf{G}$ is only $n \times n$, and the subsequent QP problem is also only with n dual variables. With the excellent implementation of matrix inverse function such as that in MATLAB, and the SMO implementation in LIBSVM, our algorithm is much faster than the existing SVM+ solver [Pechyony et al., 2010] in practice.

Moreover, in some scenarios such as multi-instance learning using privileged information [Niu et al., 2015], it often needs to iteratively solve the SVM+ problem. In this case, after calculating the matrix inverse, we only need to iteratively solve the one-class SVM, which is much more efficient than the traditional method which needs to iteratively optimize a QP problem for solving the SVM+ problem (see Section 5.6.2 for details).

5.6 Experiments

In this section, we evaluate the efficiency of our proposed two algorithms for linear and kernel SVM+, and compare them with the existing SVM+ solvers for the image classification and web image retrieval tasks. Considering we solve two variants of SVM+ instead of the original SVM+ formulation, we also report the classification/retrieval performance for comparisons.

5.6.1 Image Classification

We conduct the experiments for two image recognition tasks: digit recognition and scene recognition.

Datasets: In the digit classification task, Vapnik and Vashist [Vapnik & Vashist, 2009] have shown that the additional textual descriptions in training data are helpful for learning a better classifier. We employ the benchmark MNIST+ dataset used in [Vapnik & Vashist, 2009; Pechyony et al., 2010] for classifying the images as two digits "5" and "8". The dataset is constructed by using a subset of the MNIST dataset containing the images of the digits "5" and "8". It is split into a training set of 100 images (50 images with digit "5", and 50 images with digit "8"), a validation set of $4,002$ images, and a test set of $1,866$ images. All the images are resized into 10×10 pixels, and the 100-d vector of raw pixels is used as the main feature vector for each image. Each training image is additionally supplied with a poetic description (see [Vapnik & Vashist, 2009] for the examples), which is converted into a 21-d textual feature vector, and used as the privileged information in SVM+.

For the scene recognition task, we employ the Scene-15 dataset[Lazebnik et al., 2006b], which contains $4,485$ images of 15 different scenes. We split the data into three sets: 300 images as the training set, 40% images as the test set, and the rest as the validation set. We downsample all the images by factor 3 ($1/3$ width and height of the original images), because small images are preferred in real-world applications for saving the storage of visual data at test time. The CNN features [Krizhevsky et al., 2012b] are extracted with the CAFFE framework [**?**], which has shown good performance in various visual recognition tasks. During training, the CNN fetures extracted from the original images are treated as privileged information, and the CNN features extracted from the downsampled images as the main features. The experimental setting is inspired by [Dai et al., 2016b], where high-resolution images yield superior performance than low-resolution images for standard vision tasks. PCA is applied on two types of features to obtain 100-d compact representations.

Baselines: In our experiments, we consider two settings, the linear case and the nonlinear case. In the linear case, there is no solver specifically designed for linear SVM+. So we mainly compare our method with the following two baselines,

- LIBLINEAR: The standard linear SVM without using privileged information implemented in LIBLINEAR [Fan et al., 2008]. We include it as a baseline for investigating the effectiveness of our linear SVM+ formulation.
- gSMO: The SMO-style algorithm for solving SVM+ in the dual form, which is proposed in [Pechyony et al., 2010]. We use the released C++ implementation from the authors.

Note that the gSMO method was designed for solving kernel SVM+, but it can still take the feature vectors as the input. So we also include it as a baseline for the linear case.

For the nonlinear case, the Gaussian kernel is used for all the methods, in which the bandwidth parameter is set as the mean of distances between all training samples by default. Besides the aforementioned gSMO method, we also compare our ℓ_2-SVM+ method with the existing state-of-the-art solvers of SVM+, and also include the standard SVM solved by LIBSVM as a baseline for investigating the effectiveness of SVM+.

- SVM: The standard SVM without using privileged information, whihc is solved by using LIBSVM [Fan et al., 2008]. We include it as a baseline for investigating the effectiveness of our ℓ_2-loss SVM+ formulation.
- CVX-SVM+: An implementation based on the CVX optimization toolbox provided in [Liang & Cherkassky, 2008]. It directly solves the quadratic programming problem in (Equation 5.4) using the QP solver in CVX.
- MAT-SVM+: We additionally include the QP solver implemented in MATLAB R2014b as a baseline, and employ it to solve the QP problem in (Equation 5.4).

For all methods, the tradeoff parameters are determined based on the validation set. One-vs-all strategy is used on the Scene-15 dataset, which contains 15 classes. The classification accuracy on the test set is reported for performance evaluation. The training time of all methods is measured on a workstation with Intel(R) Core i7-3770K CPU@3.50GHz and 16 GB RAM.

Experiments on Linear SVM+

Experimental results: The classification accuracies and training time of all methods on two datasets are summarized in Table 5.1. In terms of the classification accuracy, we observe that both our linear SVM+ algorithm and gSMO achieve better results than the baseline linear SVM method, which demonstrates the effectiveness of exploiting the poetic description as privileged information for improving the image based digit recognition task. Our method achieves slight better results than the gSMO method, possibly because we use a new variant of SVM+, and our dual coordinate descent algorithm guarantees the optimal solution.

In terms of the training time, both SVM+ algorithms are slower than the baseline linear SVM, because that more complicated objective functions are used to incorporate privileged information. Our new dual coordinate descent algorithm for linear SVM+ is much faster than gSMO, which generally takes only a bit more training time than the linear SVM as shown in Table 5.1.

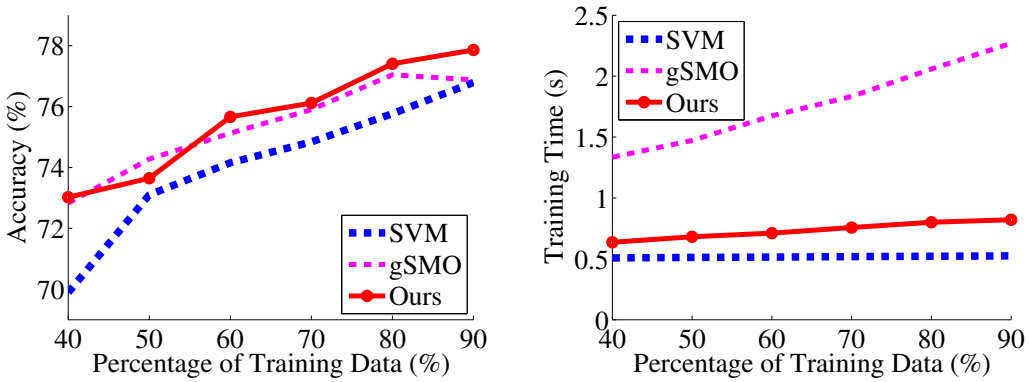
Results using different number of training samples: We further take the Scene-15 dataset to investigate the accuracies and efficiency of different methods *w.r.t.* different number of training samples. Similarly as in [Vapnik & Vashist, 2009; Pechyony et al., 2010], we randomly sample $40\%, 50\%, 60\%, 70\%, 80\%$ and 90% training samples for learning SVM and SVM+ classifiers. The experiments are repeated for 10 times, and we plot the average accuracies and training time of different methods in Figure 5.1. While the accuracies of all methods become higher when the number of training data increases, our linear SVM+ algorithm consistently outperforms the linear SVM method, and generally achieves slight better performance than the gSMO method. In term of the training time, it can be observed that the training time of our method increases linearly *w.r.t.* the number of training samples, and is several times faster than the baseline gSMO method.

Convergence: As mentioned in Section 5.4, our algorithm can be treated as a special form of the linear SVM discussed in [Hsieh et al., 2008]. So it shares the similar convergence property as the dual coordinate descent algorithm for linear SVM. To verify the convergence of our algorithm, in Figure 5.2, we take the MNIST+ dataset as an example to plot the objective values of our algorithm when the number of iterations[2] increases. It can be observed that our algorithm converges well. The objective value decreases very

[2]Similarly as in LIBLINEAR, one iteration refers to that we pass all training samples once.

Table 5.1: *Accuracies (%) and training time of all methods on the MNIST+ and Scene-15 datasets in linear case. Our results are highlighted in boldface.*

		MNIST+		Scene-15	
		Accuracy	Time (ms)	Accuracy	Time (s)
SVM		81.73	0.6	77.56	0.76
SVM+	gSMO	84.35	72.1	78.05	2.52
	Ours	**84.91**	**9.5**	**78.10**	**0.87**



(a) *(b)*

Figure 5.1: *The accuracies (Figure (a)) and the training time (Figure (b)) of different methods for solving linear SVM+ when using different number of training samples on the Scene-15 dataset.*

fast within the first ten iterations, and continues to decrease as the number of iterations increases.

Experiments on Kernel SVM+

Experimental results: We report the classification accuracies and training time of all methods on two datasets when using the Gaussian kernel in Table 5.2. On the MNIST+ dataset, we observe that the SVM+ algorithms again achieve better results than the baseline SVM algorithm, due to the utilizing of poetic descriptions as privileged information. However, on the Scene-15 dataset, gSMO and CVX-SVM+ are worse than the standard SVM method. Our ℓ_2-SVM+ algorithm achieves better result than the baseline algorithms, showing our new formulation for kernel SVM+ is effective for the scene classification problem on this dataset.

In terms of the training time, we observer that our newly proposed algorithm is the most efficient one among all SVM+ algorithms, and generally achieves order-of-magnitude speedup over the second fastest algorithm (MAT-SVM+ on the MNIST+ dataset, and

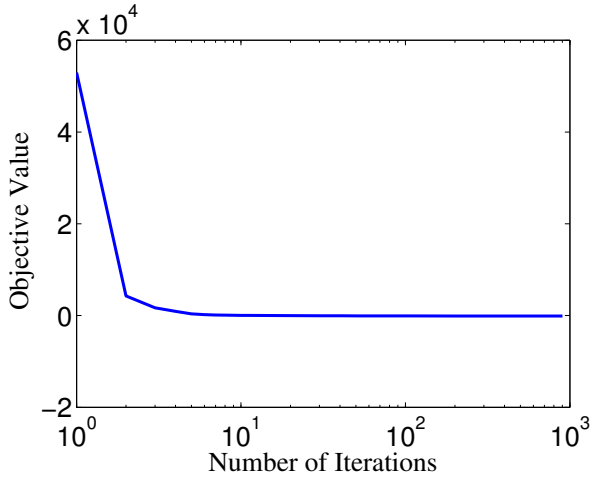


***Figure 5.2**: The objective of our coordinate descent algorithm for linear SVM+ on MNIST+ dataset.*

***Table 5.2**: Accuracies (%) and training time of all methods on the MNIST+ and Scene-15 dataset using the Gaussian kernel. Our results are highlighted in boldface.*

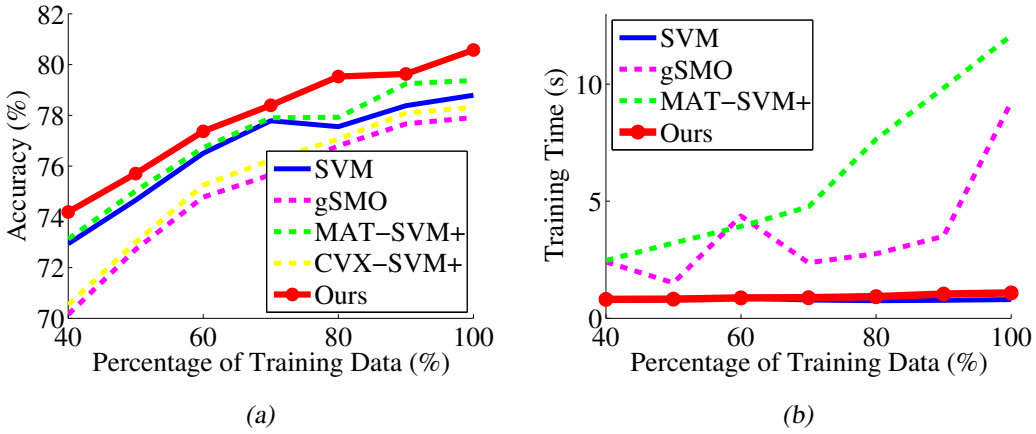
		MNIST+		Scene-15	
		Acc.	Time (ms)	Acc.	Time (s)
SVM		92.34	0.7	78.79	0.80
SVM+	gSMO	92.77	78.8	77.78	9.23
	CVX-SVM+	93.14	767.7	78.32	47.16
	MAT-SVM+	93.14	54.9	79.38	12.07
	Ours	**93.15**	**1.3**	**80.57**	**1.08**

gSMO on the Scene-15 dataset). The results demonstrate the efficiency of our new reformulation of kernel SVM+. With the reformulation, we convert it as a one-class SVM problem, and take advantage of the existing state-of-the-art SMO implementation in LIBSVM to solve it.

Results using different number of training samples: Similarly as for the linear case, in Figure 5.3, we take the Scene-15 dataset to plot the classification accuracies and training time of different methods by using different percentages of training data. The CVX-SVM+ method is not included when reporting the training time for better visualization. We observe that our ℓ_2-SVM+ algorithm outperforms other SVM+ algorithms in terms of both classification accuracy and efficiency when varying the number of training samples.

5.6.2 Web Image Retrieval

In this subsection, we demonstrate the advantage of our ℓ_2-SVM+ algorithm for solving the multiple instance learning using privileged information problem. We employ the mi-SVM-PI algorithm proposed in [Niu et al., 2015] to evaluate different SVM+ solvers. The mi-SVM-PI algorithm needs to iteratively solves the SVM+ problem, and simultaneously



(a) *(b)*

***Figure 5.3**: The accuracies (Figure (a)) and training time (Figure (b)) of different methods for solving kernel SVM+ when using different number of training samples on the Scene-15 dataset.*

infer the labels for training samples under MIL constraints. We compare the mi-SVM-PI method based on our ℓ_2-SVM+ algorithm, with their original implementation based on MATALB.

Experimental setting: It has been shown that the textual descriptions associated with the web images are effective for learning better classifiers using multiple instance learning approaches [Li et al., 2014; Niu et al., 2015]. Following [Li et al., 2014; Niu et al., 2015], we employ the NUS-WIDE web image dataset, which contains $269,648$ web images crawled from the image sharing website *Flickr*. It is officially split into a training set of 60% images, and test set of 40% images. All the images are accompanied with textual tags provided by *Flickr* users. The test images are annotated for 81 concepts. Similarly as in [Li et al., 2014], we extract the DeCAF_6 features [Donahue et al., 2014c], which leads to a 4096-dim feature vector for each web image. For the training data, we also extract a 200-dim term frequency feature from the associated textual tags of each image, and use it as the privileged information. 25 positive bags (*resp.*, negative bags) are constructed with each bag containing 15 relevant images (*resp.*, irrelevant images) as the training data for each concept. The Gaussian kernel is used for the visual features, and linear kernel is used for the textual features. The image retrieval performance is evaluated on the test set, in which only the visual features are extracted. The average precision based on the top-ranked 100 test images is used for performance evaluation, and the mean average precision (MAP) over 81 concepts is reported. Moreover, the training time of all methods over 81 concepts is reported for efficiency evaluation.

Experimental results: We report the MAPs and training time of different methods in Table 5.3. From the table, we observe that the SVM+ methods using both solvers achieve better results than the standard SVM method, because of using additional textual infor-

Table 5.3: *MAP (%) and training time (s) of different methods on the NUS-WIDE dataset. Our results are highlighted in boldface.*

		MAP	Time (s)
SVM		54.41	9.80
SVM+	MAT-SVM+	55.63	204.10
	Ours	**55.68**	**19.44**
mi-SVM-PI	MAT-SVM+	59.11	765.47
	Ours	**59.60**	**24.26**

mation in the training process. Moreover, by incorporating the multi-instance learning approach, the MAPs of mi-SVM-PI method based on both solvers are further improved. In both cases, the methods based on our ℓ_2-SVM+ achieve slight better results than the ones based on MAT-SVM+.

In terms of the efficiency, we observe that our ℓ_2-SVM+ again achieves order-of-magnitude speed-up when compared with MAT-SVM+. When incorporating the multi-instance learning approach, the mi-SVM-PI based on our ℓ_2-SVM+ uses only a bit more time than ℓ_2-SVM+, because we only need to calculate the matrix inverse once for each concept, and iteratively solve the one-class SVM problem. In contrast, the mi-SVM-PI based on MAT-SVM+ needs to iteratively solve the QP problem introduced by the SVM+ problem. Therefore, our method is more than 30 times faster than MAT-SVM+ on the NUS-WIDE dataset.

5.7 Conclusion

In this work, we have proposed two new algorithms for solving the linear and kernel SVM+. By reformulating the original SVM+ method, we obtain the dual problem with simpler constraints. Then we develop an efficient dual coordinate descent algorithm to solve the linear SVM+ problem. We also show that the kernel SVM+ using the ℓ_2-loss can be converted to the one-class SVM problem, and thus can be efficiently solved by using the SMO algorithm implemented in the existing SVM solvers such as LIBSVM. Comprehensive experiments on three tasks have demonstrated the efficiency of our proposed algorithms for linear and kernel SVM+.

6

Efficient Metric Computation via Imitation

Computers may be fast, but they are not infinitely fast. And memory may be inexpensive, but it is not free. Thus, it is always a merit to have algorithms which run fast and consumes less memory. In this chapter, we investigate the problem of boosting cheap (computationally and in terms of memory consumption) methods by imitating the desirable behaviors of better-performing but expensive (also computationally and in terms of memory consumption) alternatives. We restrict ourselves to the task of metric computation, as it is ubiquitous in vision applications, as many as classification, retrieval, clustering, and segmentation.

6.1 Introduction

The problem of computing good metrics is of great theoretical and practical interest in information processing. Applications include clustering [Xing et al., 2003; Wu et al., 2012], image annotation [Guillaumin et al., 2009a], retrieval [Frome et al., 2007; Kulis et al., 2009], classification [Belhumeur et al., 1997; Frome et al., 2007; Weinberger & Saul, 2009], among others. It is often addressed by metric learning [Baghshah & Shouraki, 2009; Kostinger et al., 2012; Guillaumin et al., 2009a; Weinberger & Saul, 2009; Qin et al., 2014], with great success in the past years. The drawback of metric learning is that it requires some form of human annotation (*e.g.* full category labels [Belhumeur et al., 1997; Frome et al., 2007; Weinberger & Saul, 2009; McFee & Lanckriet, 2010], pairwise constraints [Baghshah & Shouraki, 2009; Kulis et al., 2009; Kostinger et al., 2012; Simonyan et al., 2014]), which is expensive to acquire.

To address this problem, this work aims to learn good metrics for cheap features without using human annotation. We draw inspiration from transfer learning and propose a novel approach, coined Metric Imitation (MI). MI takes state-of-the-art, off-the-shelf features as source features (SFs) and cheap features as target features (TFs) to learn good metrics

for the TFs by imitating the metrics computed over the SFs. MI is a general framework and can at least be applied to: 1) efficiently learning good TF metrics when SFs are more powerful than TFs, but are computationally more expensive and/or more memory-hungry; and 2) creating good metrics for TFs when SFs contain privileged information but are not available at testing time.

MI comes out of a marriage of advances in metric learning and transfer learning. On the one hand, metric learning now is able to learn good metrics over general features for specific tasks [Kulis, 2012], with human supervision. On the other hand, transfer learning can now transfer knowledge (often classifiers) learned in one domain to another [Pan & Yang, 2010], for which no further human supervision is necessary. Observing both developments then begs the question whether metrics can be computed over one feature (*i.e.* SFs) to then be transferred to the domain of another feature (*i.e.* TFs) and automatically supervise the metric learning process there. In this work, we demonstrate this for several vision tasks. The main advantages of the method are: 1) it performs in an unsupervised manner, i.e. without human annotation; 2) it can be efficient as only TFs are needed during test time; 3) it can inject domain knowledge carried through SFs to TFs for metric computation.

The metric is learned in the framework of Mahalanobis distance learning, *i.e.* a linear mapping function of TFs is learned and applied prior to performing the Euclidean distance metric. Specifically, the method works as follows: 1) it translates the properties of metrics computed over SFs into manifold geometries; 2) it transfers the manifold to the domain of the TFs as view-independent properties; and 3) it learns a mapping function of the TFs so that the transferred manifold is approximated as well as possible in the transformed space. By doing this, the *neighborhood properties* of data computed over the SFs are preserved in the transformed space of TFs, *i.e.* close neighbors are still close. The reason for ensuring this is that neighbors search is crucial in many vision applications, such as clustering, retrieval, and classification.

In accordance with our previous remarks, the usability of MI is validated in two scenarios of metric learning: 1) compute good metrics for cheap features; and 2) transfer privileged information. For the first scenario, MI was tested on instance-based object retrieval using the INRIA Holiday dataset [Jegou et al., 2008] and the UKbench dataset [Nistér & Stewénius, 2006], and on category-based image retrieval and image clustering using four other datasets: Scene-15 [Lazebnik et al., 2006a], CUReT-61 [Dana et al., 1999], Caltech-101 [Fei-Fei et al., 2004], and Event-8 [Li & Fei-Fei, 2007b]. Three sophisticated, high-dimensional features were used as the SFs: object-bank [Li et al., 2010b], SIFT-llc [Wang et al., 2010], and the CNN feature [Chatfield et al., 2014], and three general, cheap features used as the TFs: GIST [Oliva & Torralba, 2001], LBP [Ojala et al., 2002b], and PHOG [Bosch et al., 2007b]. Extensive experiments show that MI consistently and significantly outperforms the metrics computed directly over the same TFs, while getting very close to the metrics from the computationally more expensive SFs in some cases. For the second scenario, MI was evaluated on example-based image super-

resolution. Patches of high-resolution images, which are not available at testing time, are used as SFs and patches of low-resolution images as TFs. Experiments show that MI is able to improve the performance of k-NN-based methods for image super-resolution.

6.2 Related Work

Our method is generally relevant to Metric Learning, Feature Embedding, and Transfer Learning.

6.2.1 Metric Learning

There has been a great deal of research devoted to metric learning to tune distance functions with human supervision. Typically, metric learning methods follow as guiding principle to learn a mapping that shrinks the distances between similar objects, while expanding the distance between dissimilar objects. Since a comprehensive overview is beyond the scope of this work, we summarize the most relevant ones according to the types of annotation used. Readers are referred to [Kulis, 2012] for a survey.

Class label is one of the most popular annotation types for metric learning. Notable examples include Neighborhood Component Analysis (NCA) [Goldberger et al., 2004], Collapsing Classes (CC) [Globerson & Roweis, 2005] and Large-Margin Nearest Neighbors (LMNN) [Weinberger & Saul, 2009]. NCA maximizes the expected number of correctly retrieved points under a stochastic neighbor selection rule. CC optimizes a similar stochastic neighbor selection rule while attempting to collapse each class to a single point. The idea enforces more regularity on the output space than NCA and leads to a convex optimization problem, but the assumption that entire classes can be collapsed to distinct points rarely holds in practice. LMNN [Weinberger & Saul, 2009] relaxes this strong assumption by defining target neighbors as the k closest samples in the same class, and maximizes margins between distances to target neighbors and to other points.

Pairwise Constraints correspond to another popular type of annotation used in metric learning, in which similar (dissimilar) pairs of objects are indicated. Earlier work by Xing *et al.* [Xing et al., 2003] formulates the clustering problem as a semi-definite programming task under similarity and dissimilarity constraints. With similar annotations, Davis *et al.* [Davis et al., 2007] tackle the metric learning problem with an information theoretic approach, namely by minimizing the Kullback-Leibler divergence between a learned and a prior matrix, expressing the pairwise constraints. Guillaumin *et al.* [Guillaumin et al., 2009b] offer a probabilistic view on learning distance metrics with pairwise constraints and solve the problem by MAP. Following the spirit of [Davis et al., 2007; Guillaumin et al., 2009b], Kostinger *et al.* [Kostinger et al., 2012] present the KISS framework by

formulating the problem as a likelihood ratio test, which avoids complex optimization problems.

While all these methods perform well, human annotations have to be provided. As a result, methods requiring weaker or no supervision are desired. Our method offers this by learning metrics over cheap features by imitating the standard metrics computed over other, sophisticated off-the-shelf features, which either contain privileged information or are more useful for the tasks at hand but too computationally expensive.

6.2.2 Embedding

There is a great deal of work about embedding for nonlinear dimensionality reduction, with the philosophy that many high-dimensional data can be characterized by a low-dimensional nonlinear manifold. The methods broadly fall into two categories: those providing a mapping function for out-of-sample extension and those just offering a visualization. We will briefly overview the methods which provide mapping functions. Embedding methods proceed mainly in two steps: 1) quantify manifold properties (local geometry); and 2) seek a low-dimensional space so that the learned properties are mostly preserved. A large range of manifold properties have been considered. For instance, Multi-dimensional Scaling [Cox & Cox, 2008] and its extension Isomaps [Tenenbaum et al., 2000] are designed to capture and preserve pairwise distances between points. Locally-linear embedding (LLE) [Roweis & Saul, 2000; He et al., 2005] computes the local linearity of the manifold and attempts to preserve that in the low-dimensional space. Laplacian Eigenmaps (LapEigen) [Belkin & Niyogi, 2001], and Hessian LLE (HLLE) [Donoho & Grimes, 2003]. proceed in a similar manner as LLE does, but the properties captured are the locality of the manifold, and the local curvature of the manifold. Due to space limitations interested readers are referred to the comparative overview in [van der Maaten et al., 2009].

Our approach draws inspiration from and extends the traditional embedding methods. The method imitates the manifold properties obtained from SFs by learning a transformation for the TFs, trading off a slight drop in precision against a significant gain in efficiency. The method also avoids feature embedding in the high-dimensional space of SFs, which is time-consuming.

6.2.3 Transfer Learning and Domain Adaptation

Our method also is akin to transfer learning. Transfer learning addresses the problem of transferring the knowledge learned in a source domain to a different target domain. Successful applications in vision include knowledge transfer from one data source to another (*e.g.* videos to images) [Kulis et al., 2011; Gopalan et al., 2011; Fernando et al., 2013]

and knowledge transfer from known classes to unseen classes [Lampert et al., 2009]. The difference between such work and ours is in the form of the knowledge. These methods transfer definitions of classes (learned classifiers) to a new domain where few or no training data is available, by reducing the discrepancy from source domain to target domain. Our goal, however, is to transfer the manifold structure from our source domain to a target domain to imitate the metrics computed in the source domain, where features in the source domain are kept intact. This distinction leads to different algorithms and applications.

6.3 Metric Imitation

In this section, we will introduce Metric Imitation (MI) in detail. Like most metric learning or embedding methods, MI operates in two modes: training and mapping. Training learns the projection transformation matrix A guided by the transferred manifold, while 'mapping' performs the learned transformation for better distance metrics during testing.

The generic problem of MI can be summarized as follows. Given a set of n training images $\mathcal{D} = \{I_1, I_2, ..., I_n\}$, and their corresponding SFs $\mathcal{D}_s = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \in \mathbb{R}^{m \times n}$ and TFs $\mathcal{D}_c = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n\} \in \mathbb{R}^{q \times n}$ (often $q \ll m$), the goal is to find a transformation matrix $A \in \mathbb{R}^{q \times q}$ that maps the n TFs $\mathbf{y}$'s to n transformed points $\bar{\mathcal{D}}_c = \{\bar{\mathbf{y}}_1, \bar{\mathbf{y}}_2, ..., \bar{\mathbf{y}}_n\}$, where $\bar{\mathbf{y}}_i = A^T \mathbf{y}_i$. The guiding principle of the learning is to approximate the manifold defined by all $\mathbf{x}$'s by the manifold defined over all $\bar{\mathbf{y}}$'s. Note that the transformation matrix A can be a low-rank matrix for the purpose of further dimension reduction. We force it to be a full-rank matrix, as the dimensionality of $\mathbf{y}$ is itself quite low.

6.3.1 Training

The training part comes with two components: learning the intrinsic manifold over the input $\mathbf{x}$'s, and learning a transformation that has to be applied to the $\mathbf{y}$'s such that the 'intrinsic' manifold is approximated as closely as possible in the learned space of $\bar{\mathbf{y}}$'s.

1. **Learning the intrinsic manifold over SFs**: the aim of this step is to capture the 'intrinsic manifold', i.e. the manifold underlying the input data. Since visual data is highly nonlinear, its global manifold layout tends to be captured better by local properties, such as Local Linear Embedding (LLE) [Roweis & Saul, 2000], Laplacian Eigenmaps (LapEigen) [Belkin & Niyogi, 2001], or Hessian LLE (HLLE) [Donoho & Grimes, 2003]. Compared to methods for global properties, such as Isomap [Tenenbaum et al., 2000], local ones have several advantages. They typically yield better results, and are faster when advantage is taken of matrix sparsity. Local methods share that they first induce a local neighborhood structure over the input data, and then quantify the corresponding local properties over the neighborhood structure. In these methods, the

learned manifold is used for dimensionality reduction [Roweis & Saul, 2000; Belkin & Niyogi, 2001; Donoho & Grimes, 2003]. However, we use the manifold as a guide to learn a transformation of other features, which are much cheaper to compute and store, so that the learned manifold is approximated with good precision in the transformed space. Below come the two steps of the learning.

(a) Constructing an adjacency graph: by taking each image I in $\mathcal{D}$ as a node, the adjacency graph $G = (\mathcal{D}, \mathcal{E})$ can be constructed in the following way: a directed edge from node i to node j is added, *i.e.* $e_{ij} = 1$, if I_j is among the K nearest neighbors of I_i. The features in source domain $\mathbf{x}$ are used to compute the neighbors under the L_2 distance, but any distance measure can be applied. The global structure of the nonlinear manifold is captured by the interaction of these overlapping local structures. The largest connected component is used for further analysis if the graph is not fully connected.

(b) Quantifying the local properties of the manifold: The local properties (relations) of the manifold can be quantified in a variety of ways, including local linearity as in LLE [Roweis & Saul, 2000; He et al., 2005], local pairwise distance as in LapEigen [Belkin & Niyogi, 2001], local 'curviness' as in HLLE [Donoho & Grimes, 2003], etc. Here, we use two of the most representative ones: local linearity and local pairwise distance. For the reason of efficiency, their linear variants are used.

- LLE [Roweis & Saul, 2000] assumes that the manifold is locally linear, *i.e.* a data point $\mathbf{x}_i$ is characterized by encoding $\mathbf{x}_i$ as a linear combination W_i (reconstruction weights) of its k nearest neighbors $\mathbf{x}_{ij}$. The weights W are learned by minimizing the following reconstruction error:

$$\begin{aligned} \underset{W}{\text{minimize}} \sum_{i=1}^{n} \|\mathbf{x}_i - \sum_{j=1}^{n} W_{ij}\mathbf{x}_j\|^2 \\ \text{s.t.} \sum_{j=1}^{n} W_{ij} = 1 \end{aligned} \tag{6.1}$$

  where W_{ij} is only allowed to have non-zero value if $e_{ij} = 1$.

- LapEigen [Belkin & Niyogi, 2001] characterizes the local manifold structure by encoding the pairwise distances between close neighbors. The neighbors are defined also over the graph G. The weight matrix W is defined as a Gaussian kernel function over the distance of neighboring nodes to reflect the proximity of data points:

$$w_{ij} = \begin{cases} \exp -\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}, & \text{if } e_{ij} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{6.2}$$

  where σ modulates the decreasing rate of W with the distance.

Metric Imitation (MI) learns the structure of the manifold underlying the $\mathbf{x}$'s and describes its local properties through the weight matrix W. The learned W is then transferred to the domain of the $\mathbf{y}$'s. There a linear mapping is learned, i.e. a matrix $A \in \mathbb{R}^{q \times q}$ to be applied to the $\mathbf{y}$'s in order to mimic corresponding manifold characteristics: 1) for LLE, a transformation is learned for $\mathbf{y}$'s so that all mapped data points $\bar{\mathbf{y}}$'s are reconstructed well from their neighbors (defined in graph G) with the learned reconstruction weight W; 2) for LapEigen, MI learns a projection so that all nearby points in the space of the $\mathbf{x}$'s are still nearby points in the mapped space of the $\bar{\mathbf{y}}$'s.

2. **Approximating the manifold by using the TFs**: with the learned manifold property W from SFs (in the manner of LLE or LapEigen), the aim of this step is to compute a linear projection matrix A for TFs so that the manifold properties learned over the SFs are preserved as much as possible in the transformed space. Specifically, MI minimizes the following cost function:

$$\underset{\bar{\mathbf{y}}}{\text{minimize}} \sum_{i=1}^{n} \|\bar{\mathbf{y}}_i - \sum_{j=1}^{n} W_{ij}\bar{\mathbf{y}}_j\|^2. \tag{6.3}$$

The cost function is very similar to that in Equation 6.1, but here the weight matrix W is fixed and the optimization is performed over the $\bar{\mathbf{y}}$. For the sake of efficiency, $\bar{\mathbf{y}}$ is constrained to be a transformed version of $\mathbf{y}$ by a linear transformation: $\bar{\mathbf{y}} = A^T\mathbf{y}$. According to the induction of [He et al., 2005], solving Equation 6.3 can be reduced to solving the following generalized eigenvector problem:

$$YMY^T\mathbf{a} = \lambda YY^T\mathbf{a}, \tag{6.4}$$

where

$$\begin{aligned} Y &= (\mathbf{y}_1, ..., \mathbf{y}_n), \\ M &= (I - W)^T(I - W), \\ I &= diag(1, ..., 1). \end{aligned}$$

Solving Equation 6.4 leads to q eigenvectors $\mathbf{a}_1, ..., \mathbf{a}_q$, with eigenvalues ordered from the smallest to the largest. Thus, the embedding is performed as:

$$\mathbf{y}\tilde{\mathbf{o}}\bar{\mathbf{y}} \doteq A^T\mathbf{y}, \tag{6.5}$$

where the projection matrix $A = (\mathbf{a}_1, \mathbf{a}_1, ..., \mathbf{a}_q)$.

6.3.2 Mapping

During the test phase, MI only computes the TFs $\mathbf{y}$'s, for the test images, and then computes the distance metric over them with the learned projection matrix A:

$$d(\mathbf{y}_i, \mathbf{y}_j) = \|\bar{\mathbf{y}}_i - \bar{\mathbf{y}}_j\|^2 = \|A^T\mathbf{y}_i - A^T\mathbf{y}_j\|^2, \tag{6.6}$$

which is then used for the task at hand. Computing the Euclidean distance in this transformed space equals to computing the following Mahalanobis distance function in the original feature space:

$$\begin{aligned} d(\mathbf{y}_i, \mathbf{y}_j) =& (\mathbf{y}_i - \mathbf{y}_j)^T H (\mathbf{y}_i - \mathbf{y}_j) \\ =& (\mathbf{y}_i - \mathbf{y}_j)^T A^T A (\mathbf{y}_i - \mathbf{y}_j). \end{aligned} \tag{6.7}$$

The form of Equation 6.6 is preferred to that of Equation 6.7, because many approximate, fast nearest neighbor techniques have been developed for the euclidean metric. Since the space is learned to preserve the neighborhood properties of the 'intrinsic' manifold, superior performance over common metrics on original TFs is expected. Although our mapping and the embedding for dimensionality reduction are similar technically, they serve totally different purposes. The advantages of MI over the embedding for dimensionality reduction of SFs are twofold: 1) computationally more efficient, as SFs are not needed by MI during testing, and 2) more robust, as fewer parameters are needed to learn the projection, as $q \ll m$. The advantage of MI over the embedding for TFs is that the 'privileged' information in the source domain is injected into the target domain at little extra cost.

6.4 Experiments

We evaluated Metric Imitation (MI) in the two aforementioned scenarios: 1) learning of good metrics when the source features (SFs) are more powerful but computationally more expensive than the target features (TFs); 2) when SFs contain privileged information but are not available during test time. For the first, MI was evaluated on image clustering, category-based image retrieval, and instance-based object retrieval. For the second, MI is evaluated on example-based super-resolution. For all experiments of these four tasks, MI uses identical parameter settings: the size of the neighborhood K was set to 40 and σ in Equation 6.2 was set to the mean of the L_2 distance between all feature vectors $\|\mathbf{x}_i - \mathbf{x}_j\|^2$, where $i, j \in \{1, ..., n\}$. These values have been set empirically, but our experiments show that MI is not very sensitive to these choices. If needed, a cross-validation on hold-out sets can be considered.

6.4.1 Image Clustering

As the amount of image data is on a rampant increase, unsupervised methods to group objects into semantically relevant clusters are in great demand. Such clustering is one of the core problems in modern computer vision. In this section, we apply MI to image clustering.

Datasets: The following four popular datasets are used: Scene-15 [Lazebnik et al., 2006a], CUReT-61 [Dana et al., 1999], Caltech-101 [Fei-Fei et al., 2004], and Event-8 [Li & Fei-Fei, 2007b]. They contain 4585 images of 15 classes of general scenes, 5612 images of 61 texture classes, 8677 images of 101 object classes, and 1574 images of 8 classes of sports events, respectively.

Features: In order to sufficiently sample the space of possible features, three different SFs and three different TFs are chosen. The SFs are: SIFT-llc [Wang et al., 2010], object-bank(OB) [Li et al., 2010b], and the CNN feature [Chatfield et al., 2014]. The chosen TFs are: GIST [Oliva & Torralba, 2001], LBP [Ojala et al., 2002b], and PHOG [Bosch et al., 2007b]. They were chosen because they are popular in the community, come with available code, and are based on different underlying techniques. Another more important reason is that the three SFs are representatives of the state-of-the-art features but computationally expensive (relatively), and the three TFs are low-dimensional and computationally very cheap.

SIFT-llc is a representative image representation based on local descriptors, which utilizes the locality constraints to encode local descriptors and integrates the results by max-pooling. The CNN feature [Chatfield et al., 2014] is obtained by vectorizing the convolutional results of the deep convolutional neural network [Krizhevsky et al., 2012b], which is trained on the ImageNet dataset. OB performs object detectors as filters and collects the response of these semantic filters over an image. The features were computed as follows. SIFT-llc[1] and OB[2] were computed with their default parameters in the authors' implementations. For the CNN feature, the MatConvNet package [Vedaldi & Lenc, 2014] was used with a pre-trained CNN model. The convolutional results at layer 16 were stacked as the CNN feature vector, with dimensionality 4096. GIST was computed with only one cell. For LBP, the uniform one was used with PCA for further dimensionality reduction. For PHOG, the 2-layer pyramid was used. The dimensionalities of the six features are listed in Table 6.1, from which it is evident that the TFs are much lighter and efficient to compute and handle, in line with the aims of MI.

Evaluation: We follow [Tuytelaars et al., 2009; Dai et al., 2010b, 2012b] and used Purity (bigger=better) for evaluation. It measures how pure the discovered clusters are according to the ground truth class labels. Spectral Clustering was used on top of MI for the clustering. We compared to Spectral Clustering with the SFs and TFs as inputs. The K-means algorithm was also tried, but it generally produces similar or poorer results than Spectral Clustering. As suggested in [Dai et al., 2012b], we use the χ^2 distance for LBP and PHOG, and the Euclidean distance for GIST. Euclidean distance is used for all the SFs.

Results: 50% of the images (without labels) were used for training and the rest used for testing. Table 6.2 lists the results of all methods on the four datasets over 5 random

[1] http://www.ifp.illinois.edu/~jyang29/LLC.htm
[2] http://vision.stanford.edu/projects/objectbank

***Table 6.1**: The dimensionality of the features.*

TFs			SFs		
GIST	PHOG	LBP	CNN	SIFT-llc	OB
20	40	59	4096	21504	44604

***Table 6.2**: Purity of clustering by Metric Imitation (MI), where 50% of the images are used for training and the rest for testing.*

	TFs	**MI**		SFs	**MI**		SFs	**MI**		SFs
Lap	LBP	LLE	Lap	SIFT-llc	LLE	Lap	CNN	LLE	Lap	OB
Scene-15	0.36	0.40	0.46	0.49	0.47	0.48	0.69	0.42	0.48	0.54
CUReT-61	0.33	0.44	0.46	0.39	0.33	0.41	0.60	0.31	0.37	0.44
Caltech-101	0.32	0.34	0.34	0.51	0.37	0.36	0.68	0.37	0.35	0.52
Event-8	0.39	0.46	0.46	0.57	0.47	0.47	0.82	0.48	0.48	0.46

training-test splits. Due to space limitations only one type of TFs, LBP, is listed here (same for the following experiments). GIST and PHOG generally yield similar trends than LBP, and their outcomes are reported in the supplementary material. It is expected to find that the SFs perform significantly better than the TFs on image clustering. This allows MI to learn 'useful' knowledge from the SFs. From the table, it is evident that MI is able to learn a better metric for the TFs than the common, generic ones (*e.g.* χ^2 and L_2). For instance, on Scene-15 MI improves the purity of LBP from 0.36 to 0.48 if the CNN feature is used as the SFs and LapEigen is chosen. On Event-8, MI improves the purity of LBP from 0.38 to 0.48 when OB is used as the SFs and LLE is used. Similar trends can be observed across all datasets, all SFs, and the two ways of encoding manifold properties. This shows that MI efficiently provides good metrics for image clustering in general, and that the framework is robust to the specific choices of its components.

Across classes: In order to check the generality of MI, we also tested MI in a wilder scenario, where training and test images are from totally different classes. Half of the classes (floored to the closest integer) were used for training and the rest of classes used for testing, *i.e.* 7 classes used for training for Scene-15, 30 for CUReT-61, 50 for Caltech-101, and 4 for Event-8. All methods run over 5 random training-testing splits and the average results are reported in Table 6.3. The table shows that even when the training images and test images are from different distributions, MI is still able to improve the performance over the standard metrics of TFs. This suggests that MI is also able to work in a fully unsupervised manner. This property is important, especially for online applications where test data is not available beforehand.

Baselines and competing methods: We compared MI also to two baselines and one competing method. The first baseline is to reduce the dimensionality of SFs via PCA, and then regress the reduced SFs by TFs. The method does not work as well as MI, only producing an improvement of 20% to 40% of that of MI, while being much slower as the variance matrix of the high-dimensional SFs is needed and the regression itself is also

***Table 6.3**: Purity of clustering by Metric Imitation (MI) across classes, where half of the classes are used for training and others for testing.*

	TFs	**MI**		SFs	**MI**		SFs	**MI**		SFs
	LBP	LLE	Lap	SIFT-llc	LLE	Lap	CNN	LLE	Lap	OB
Scene-15	0.63	0.67	0.70	0.85	0.65	0.66	0.90	0.61	0.59	0.74
CUReT-61	0.62	0.62	0.64	0.65	0.66	0.69	0.77	0.51	0.58	0.68
Caltech-101	0.57	0.62	0.60	0.73	0.59	0.57	0.77	0.64	0.63	0.70
Event8	0.70	0.72	0.74	0.80	0.70	0.72	0.89	0.75	0.73	0.80

***Table 6.4**: The computational time (in seconds) of a full matrix of pairwise distance with different features, where MI(X) denotes that Metric Imitation for feature X.*

	SFs			**MI**			**TFs**		
	GIST	PHOG	LBP	GIST	MI	MI	CNN	SIFT-llc	OB
Scene-15	0.41	0.47	0.53	0.41	0.48	0.53	21.92	104.83	215.56
CURet-61	0.61	0.71	0.78	0.62	0.71	0.84	32.68	161.29	325.97
Caltech-101	1.51	1.73	1.96	1.51	1.75	2.01	78.67	388.01	796.86
Event-8	0.05	0.06	0.08	0.05	0.06	0.08	2.76	13.56	26.69

***Table 6.5**: MAP of category-based image retrieval by MI, with 50% images used for training and the rest for testing, and recall set to 0.1.*

	TFs	**MI**		SFs	**MI**		SFs	**MI**		SFs
	LBP	LLE	Lap	SIFT-llc	LLE	Lap	CNN	LLE	Lap	OB
Scene-15	0.50	0.54	0.55	0.60	0.58	0.58	0.72	0.56	0.57	0.65
CUReT-61	0.83	0.90	0.87	0.90	0.90	0.90	0.95	0.86	0.84	0.90
Caltech-101	0.35	0.39	0.38	0.57	0.41	0.41	0.79	0.40	0.40	0.60
Event-8	0.44	0.53	0.52	0.70	0.53	0.55	0.89	0.51	0.50	0.60

***Table 6.6**: MAP of category-based image retrieval by MI with the concatenation of LBP, GIST and PHOG (LGP) used as the TFs. 50% images are used for training and the rest for testing. Recall is set to 0.1.*

	TFs	**MI**		SFs	**MI**		SFs	**MI**		SFs
	LGP	LLE	Lap	SIFT-llc	LLE	Lap	CNN	LLE	Lap	OB
Scene-15	0.52	0.60	0.61	0.60	0.64	0.64	0.72	0.62	0.63	0.65
CUReT-61	0.84	0.95	0.93	0.90	0.94	0.96	0.95	0.92	0.90	0.91
Caltech-101	0.42	0.48	0.46	0.57	0.51	0.51	0.79	0.48	0.48	0.59
Event-8	0.52	0.63	0.63	0.70	0.65	0.64	0.88	0.60	0.56	0.58

computationally heavy. Also, this baseline is to 'reconstruct' the SFs, which is a harder problem than what MI aims to solve; MI learns (imitates) the 'intrinsic' neighborhood behaviors, which matter most for many vision applications. The second baseline is to sample a collection of constraints from the distance matrix of SFs, and to use them, along with TFs, as the input for standard metric learning (ML) methods. We tried three well-

***Table 6.7**: MAP of image retrieval by MI on the Holidays and UKbench datasets, when the recall is set to 1.0.*

	TFs	**MI**		SFs	**MI**		SFs	**MI**		SFs
	LBP	LLE	Lap	SIFT-llc	LLE	Lap	CNN	LLE	Lap	OB
Holiday	0.38	0.50	0.48	0.66	0.50	0.49	0.72	0.48	0.46	0.48
Ukbench	0.33	0.39	0.38	0.63	0.44	0.39	0.86	0.36	0.38	0.58

known ML methods: LMNN [Weinberger & Saul, 2009], KISS [Kostinger et al., 2012], and MLRank [McFee & Lanckriet, 2010]. None of them performed well, mainly because these ML methods assume the constraints to be noise-free and perform 'strong' supervised learning with them. The sampled constraints, however, are noisy. MI translates the 'oracle' metric to manifold structures in a more robust way by LLE or LapEigen, which makes it more suitable to exploit the weak supervisory information. We also compared our method to [Gong & Lazebnik, 2011], where CCA is used to improve one view of the data by using another view of data which contains privileged information. We found that the results are similar to that of our first baseline as both of them mimic the global behavior of SFs. Also, CCA is slower than MI for our tasks, because the correlation matrix of the two features needs to be learned, which has a large number of values.

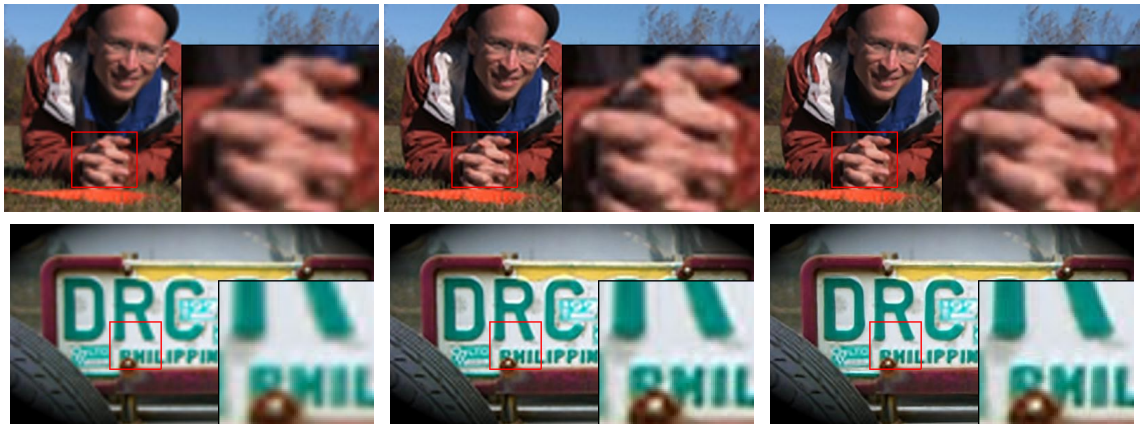
Time complexity: The computation time for a full pairwise distance metric for the four datasets is shown in Table 6.4, where MI is compared to the Euclidean distance metric over SFs and TFs directly. The time was measured on a desktop PC, using a single Core i5 with 2.8 GHz. From the table, it is clear that computing metrics is much faster over TFs than over SFs, as they are of much lower dimensions, as shown in Table 6.1. It can also be observed from the table that MI needs only marginally more computation time than the original TFs, while improving the quality of metrics for the TFs significantly. This proves the usefulness of MI, especially for scenarios with limited computing power and limited memory budget.

6.4.2 Category-based image retrieval

Another typical test-bed for metric learning is retrieval. In this section, we apply MI to the task of category-based image retrieval, where the goal is to retrieve images of the same category as that of the query image. The same datasets and features are used as for image clustering in Section 6.4.1.

Evaluation: Again, 50% of images were used for training (without using labels) and another 50% for testing. The labels of test images were used for evaluation, where each single image was taken as the query once and the mean average precision (MAP) was computed. The recall was set to 0.1 as precision of the top images matters most.

Results: Table 6.5 shows all the results of the four datasets. It can be observed that MI serves its purpose well; it improves the precision of image retrieval considerably over the



Bicubic JOR **JOR + MI**

***Figure 6.1**: Examples with an upscaling factor $\times 4$. Best seen on the screen. Images are obtained from the Internet.*

common metrics of the low-dimensional TFs. For instance, on CUReT-61 the precision is improved from 0.83 to 0.90, and on Caltech-101 the precision is improved from 0.35 to 0.41. These results show that MI is very helpful for retrieval, because 1) labeled training data is often not available in retrieval as the datasets are often unorganized; 2) an efficient solution is desirable as retrieval is often performed as an interactive task.

Concatenation of all TFs: We also tested the performance of MI when the concatenation of LBP, GIST, and PHOG features is used as TFs. Table 6.6 shows the results. As can be seen from Table 6.6 and Table 6.5 the concatenation does improve the performance of MI over that of using single TFs. The most interesting observation is that MI gets very close to or is on par with the expensive, high-dimensional SFs. This implies that the cheap, low-dimensional features contain a lot of useful information as long as the right transformation is discovered and ensures that they are 'read' in a more useful way. MI is designed exactly for this purpose.

6.4.3 Instance-based Object Retrieval

MI was also tested for instance-based object retrieval, where the goal is, given an object instance, to retrieve all instances of the same object class. The latter can be captured from different views, from different distances, etc. We used the same features as for image clustering in Section 6.4.1.

Datasets: Two of the most popular datasets for this task were used: the INRIA Holidays dataset [Jegou et al., 2008] and the UKbench dataset [Nistér & Stewénius, 2006]. Holidays provides 500 image groups with $1,491$ images in total and for a very large variety of scene types. Each group consists of photos of the same scene/object taken under differ-

ent conditions: rotations, viewpoint and illumination changes. UKbench contains $10,200$ images of $2,550$ objects, 4 images for each object from different views and distances.

Results: Half of the object classes (floored to closest integer) were used for training (to learn the mapping function) and the rest for testing. MAP was again used as the criterion. The results for LBP are listed in Table 6.7. It is interesting to see that even when trained on different datasets without using any labels, MI is able to learn good metrics for object retrieval. The learned metrics are significantly better than the standard ones over the TFs. For instance, the MAP of LBP is improved from 0.38 to 0.50 on Holidays and from 0.33 to 0.44 on the UKbench dataset, when the CNN feature is used as the SFs. Yet, there is still room for improvement when compared against the performance of the state-of-the-art object retrieval systems (*e.g.* [Qin et al., 2014]). There are two main reasons: 1) MI is trained in a fully unsupervised manner: trained on different objects and without using any human annotations; 2) MI uses cheap, low-dimensional features for the sake of efficiency (the difference to [Qin et al., 2014] in terms of feature dimensionality is as large as 3 orders of magnitude). The goal of MI is different, i.e. providing an efficient solution and avoiding any human annotations.

6.4.4 Image Super-resolution

The goal of image super-resolution is to generate high-resolution (HR) images from low-resolution (LR) ones. Example-based learning methods have proven successful in learning from a collection of patch pairs: LR patches and corresponding HR patches [Chang et al., 2004; Timofte et al., 2013; Dong et al., 2014; Dai et al., 2015d]. In this section, we apply MI to example-based image super-resolution. In order to better show the advantage of MI, we follow the approaches which are directly based on k-NN search [Freeman et al., 2002; Chang et al., 2004; Dai et al., 2015d]. The very recent method JOR [Dai et al., 2015d] was employed for comparison. JOR jointly learns a collection of regressors from LR patches to HR ones, which collectively yield the smallest super-resolving error for all training data, and selects the most appropriate regressor for each test patch via a voting scheme by its k nearest neighbors from the training samples. MI replaces the standard L_2 metric in the k-NN search by the learned metric, and keeps the rest of the system intact. By the replacement, the method (JOR + MI) is now trying (via imitation) to retrieve neighbors defined over HR patches: searching for LR patches whose corresponding HR patches are similar to the desired HR patch of the test LR patch. This exactly tallies with the goal of example-based image super-resolution.

Since the number of training samples for image super-resolution is much larger (often millions) than that for other vision tasks considered in this work, solving the eigenvector problem of MI directly is intractable. We employed the eigenfunction technique [Fergus et al., 2009] for an approximate solution. We tested MI with 0.5 million training patches. The method was evaluated on the two standard datasets Set5 and Set14, and MI improves

the PSNR from 32.30 to 32.53 on Set5, and from 28.90 to 29.10 on Set14 for an upscaling factor of $\times 3$. For a factor $\times 4$, MI improves JOR from 29.94 to 30.15 on Set5 and from 27.13 to 27.25. The method JOR + MI trained with 0.5 million patches is on par with the JOR trained with 5 million patches [Dai et al., 2015d], but is considerably faster both at training and testing. The method also outperforms the very recent method SRCNN [Dong et al., 2014] using deep convolutional network, in terms of both performance and speed (in terms of speed at testing, SRCNN is on par with JOR trained from 5 million patches [Dai et al., 2015d]). We also trained the method with 5 million patches, but observed no further improvement. A possible reason is that once samples densely cover the space, standard metrics are sufficient to find good neighbors. We leave the investment to learn more effectively from very large datasets as our future work. However, we believe that the case with 0.5 million patches suffices to prove the concept: MI yields an efficient solution to image super-resolution without sacrificing performance. Two image examples are shown in Figure 6.1. The examples and the numbers show that MI improves the results of JOR by transferring domain knowledge from the space of HR patches into that of LR patches.

6.5 Conclusion

The work proposed the novel Metric Imitation method to learn good metrics for features in one domain (target features, TFs) by imitating the metrics computed over features of another domain (source features, SFs), and this even without supervision. MI translates the neighborhood behavior of SFs into manifold geometry, and transfers it to the domain of TFs as a guide for metric learning. MI then seeks a linear mapping function of the TFs so that the transferred manifold can be approximated well, which leads to an imitation of the metrics computed over the SFs. The method is easy to understand and easy to implement. The usefulness of MI has been corroborated by two scenarios with four popular vision tasks. Extensive experiments have shown that MI is able to learn significantly better metrics over cheap TFs, while only slightly increasing the time complexity during testing. We hope the idea of metric imitation and manifold transfer will also prove useful for metric learning and transfer learning. The code of this work is available at `www.vision.ee.ethz.ch/~daid/MetricImitation`.

7

Performance Prediction: Succeed or Fail?

As stated in the introduction, success-aware (failure-aware) vision algorithms is of paramount importance in real applications, as the asset avoids disastrous consequences due to silent system failures. It is also useful in view of efficient algorithms, because it enables dynamic resource allocation based on the complexity of different visual samples. In this chapter, we present a method to predict the performance of texture synthesis methods for given texture examples.

7.1 Introduction

A substantial amount of work has been devoted to synthesising textures from examples [Heeger & Bergen, 1995; Portilla & Simoncelli, 2000; Efros & Freeman, 2001; Kwatra et al., 2003; Lefebvre & Hoppe, 2005; Ma et al., 2011; Dai et al., 2013]. We will refer to such example-based texture synthesis as 'ETS' in the remainder of this work. Even if the set of textures that can be successfully synthesised that way has steadily been growing, it is often not clear beforehand whether ETS would be successful for a specific texture sample. It would be interesting if we were able to predict its synthesizability – how well its underlying visual patterns can be re-synthesized by learning only from the sample. Even if challenging, the task may be doable, given that other qualitative image characteristics could be quantified, like quality [Luo & Tang, 2008], memorability [Isola et al., 2011], or interestingness [Gygli et al., 2013].

While ETS has proven to be a powerful tool to generate large-scale textures [Wei et al., 2009], providing texture examples is not straightforward [Lockerman et al., 2013]. ETS systems typically expect a rectangular sample image, representing a head-on view of a flat, outlier-free textured surface. Not just any example image returned by an image searching engine (by typing keywords) will do. Such retrieved images usually contain outliers, cluttered backgrounds, distorted texture surfaces, or even objects and complex scenes. Being

(a) 0.84 (b) 0.80 (c) 0.72

(d) 0.57 (e) 0.54 (f) 0.51

(g) 0.41 (h) 0.35 (i) 0.32

(j) 0.28 (k) 0.18 (l) 0.14

***Figure 7.1**: Synthesizability of texture examples detected by our system. The values are in $[0, 1]$ and a higher value means the example is easier to synthesize. All images are of 300×300 pixels.*

able to rank retrieved images in terms of their synthesizability can then at least perform an initial selection. It can also be used to trim images to regions with good synthesizability, by e.g. removing undesirable background. Furthermore, synthesizability can help select an appropriate ETS method. The optimal approach – also taking into account speed and stability – will depend on the texture and the application. Quilting [Efros & Freeman, 2001] is very potent, for instance, but will tend to produce verbatim repetitions that become salient when larger areas need to be synthesized. It would be good if in such case one could take recourse to an alternative method that does not produce such issues. Last but not least, studying synthesizability as a general image property is interesting *per se*.

Figure 7.1 shows the synthesizability scores assigned to some texture samples by our system. Figure 7.11 illustrates the trimming of an image to its most synthesizable, rectangular regions (the red cut-outs).

In order to learn image synthesizability and evaluate its performance, we have collected a fairly large texture dataset of $21,302$ texture images. This dataset has been manually annotated in terms of the synthesizability of each image. The synthesizability is characterised as the 'goodness' of the 'best' synthesis result as obtained by a set of ETS methods. The 'goodness' is quantified as one of three levels: good, acceptable, and bad. See Figure 7.2 for examples.

As to the automated synthesizability scoring, a series of features that would seem to be connected with the task are defined. A scoring function is then learned from the collection of annotated data. The experimental results show that automated synthesizability scoring is possible.

Our main contribution are: (1) to learn the image property synthesizability methodologically; (2) to design several novel features for qualitative texture analysis (esp. 'textureness', homogeneity, repetitiveness, and irregularity); and (3) to offer a fairly large texture dataset together with synthesizability annotations;

The remainder of the work is organized as follows. Section 7.2 reports related work. Section 7.3 is devoted to our dataset, followed by our features and learning method in Section 7.4. Section 7.5 presents our experiments and Section 7.6 concludes.

7.2 Related work

Example-based texture synthesis. Techniques of example-based texture synthesis can be broadly categorized into four categories: feature-oriented synthesis [Heeger & Bergen, 1995; De Bonet, 1997; Portilla & Simoncelli, 2000; Galerne et al., 2011], Markov Random Fields (MRFs) methods [Paget & Longstaff, 1998; Zhu et al., 1998; Zalesny et al., 2005], neighborhood-based methods [Efros & Freeman, 2001; Kwatra et al., 2003, 2005; Dai et al., 2013], and tile-based methods [Cohen et al., 2003; Liu et al., 2004]. The first

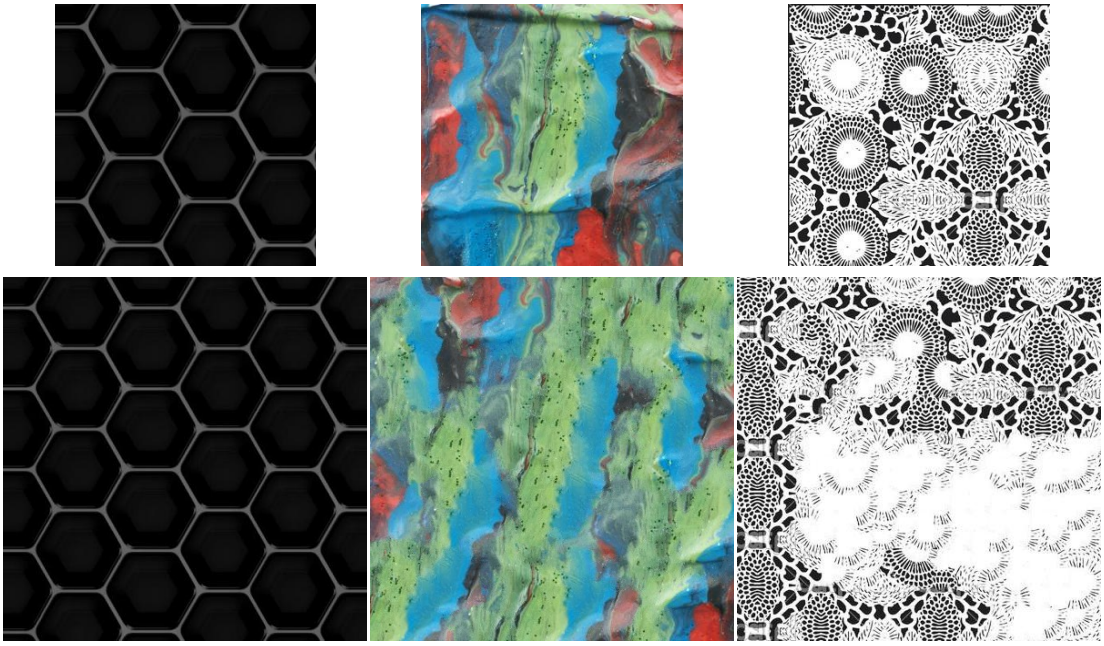
group learns the statistics of carefully designed features and leads the synthetic images to have/achieve similar values, e.g. color histograms [Heeger & Bergen, 1995], multi-band spatial frequencies [De Bonet, 1997], and wavelet features [Portilla & Simoncelli, 2000]. This group of methods are stable and do not generate verbatim repetition, but the main challenge lies in designing a common set of features that is able to capture the essence of all kinds of textures. The second group considers textures as instances of MRFs. Parameters of the MRFs are estimated from the texture examples and new textures are then sampled from the model. Multi-scale neighborhood-based MRFs are learned in [Paget & Longstaff, 1998] and pairwise clique-based MRFs in [Zalesny et al., 2005]. This strand is theoretically well-founded, but is computationally expensive. The third group generate textures by copying pixels or patches from the exemplar inputs [Efros & Leung, 1999; Efros & Freeman, 2001; Kwatra et al., 2003, 2005; Dai et al., 2013]. Unlike the first two groups they do not provide cues for texture analysis, but are often more efficient and tend to work for a larger variety of textures. The last group assemble new textures out of a set of (rectangular) tiles cropped from example images. This stream of methods are very efficient once the tiles are estimated. However, identifying these tiles is non-trivial: [Liu et al., 2005] handled this problem by estimating the translation symmetries and [Dai et al., 2013] through semantic labeling.

Texture recognition. Our work is also related – albeit rather weakly – to material recognition. Features for material classification include statistics of filter responses [Leung & Malik, 2001; Manjunath & Ma, 1996; Schmid, 2001], joint intensity distributions within a compact neighborhood [Varma & Zisserman, 2009; Liu et al., 2011], geometric features over topographic maps [Xia et al., 2010], and high-level semantic attributes [Matthews et al., 2013]. Similarly, we need to design appropriate features for this new task.

7.3 Data collection

Although it stands to reason that some textures are easier to synthesise than others, quantifying this expectation has not been addressed. In order to learn to predict synthesizability, we collected a texture dataset and annotated it in terms of synthesizability. $40,000$ images were downloaded from Google, Bing, and Flickr by providing 60 keywords. The keywords used are to cover common material classes such as glass, water, stone, plastic, fabric, leather, metal and paper, and to cover common geometric texture attributes such as stochastic, repetitive, lined, speckled, wrinkled and cracked. All images were truncated to 300×300 pixels, with images smaller than this not being used. Since the retrieved images are very 'noisy', with some not showing textures, being of low quality, or being severely watermarked, we made a manual selection for the truncated images. Finally, we ended up with a dataset of $21,302$ texture samples.

For the annotation, we characterize the synthesizability of a texture as the 'goodness' of the best synthesized image among those generated by a selected set of ETS meth-



(a) good, Quilting (b) acceptable, MMRF (c) bad, NULL

***Figure 7.2**: Three texture examples from our dataset with their annotations of synthesizability. Top: texture exemplars; bottom: synthesized textures.*

ods. A good synthesized image should be as similar as possible to the input example and should not have visible artifacts such as seams, blocks and ill-shaped edges, and should not contain salient repetitions of sub-patterns in a verbatim fashion, if that is not the case in the original. Since no single ETS method performs better than all others on all kinds of textures, the annotator got the choice between the results of four specific methods, that are based on different methodologies: an image quilting method [Efros & Freeman, 2001], a multi-scale Markov Random Field (MMRF) method [Zalesny et al., 2005], a wavelet-based parametric method [Portilla & Simoncelli, 2000], and a random phase synthesis [Galerne et al., 2011]. While future work will probably yield more powerful ETS methods still, this dataset constitutes an initial benchmark, based on the current state-of-the-art in ETS. The final outcome of the annotation for a texture example is the 'goodness' of the synthesized result (among the 4) that an expert annotator considered best. This goodness was expressed as one of 3 levels: good, acceptable, and bad, assigned synthesizability scores of 1, 0.5 and 0, resp. The 'best' method of each texture example was also recorded to learn which method is the best to synthesize a given texture example. This was only performed for 'good' and 'acceptable' images; 'bad' ones were assigned to 'NULL'. Figure 7.2 shows examples of such annotation. In total, 25.5% samples were labeled bad, 39.7% acceptable and 34.8% good.

7.4 Learning image synthesizability

In this section, we investigate the visual features relevant to image synthesizability. We start from general image features, to move on to our designed texture features, and to the learning method.

7.4.1 General features

Local patterns. Local binary patterns (LBP) [**?**] have been widely used in texture recognition and such features achieved s-o-a classification performance [Liu et al., 2011]. Thus, we included uniform LBP.

Filter responses. Using image filters has become one of the most popular tools for texture analysis [Leung & Malik, 2001; Manjunath & Ma, 1996] and synthesis [Zhu et al., 1998]. Thus, filter bank responses may be helpful for learning synthesizability too. The Schmid Filter Bank [Schmid, 2001] is employed with 13 rotationally invariant filters at 5 scales.

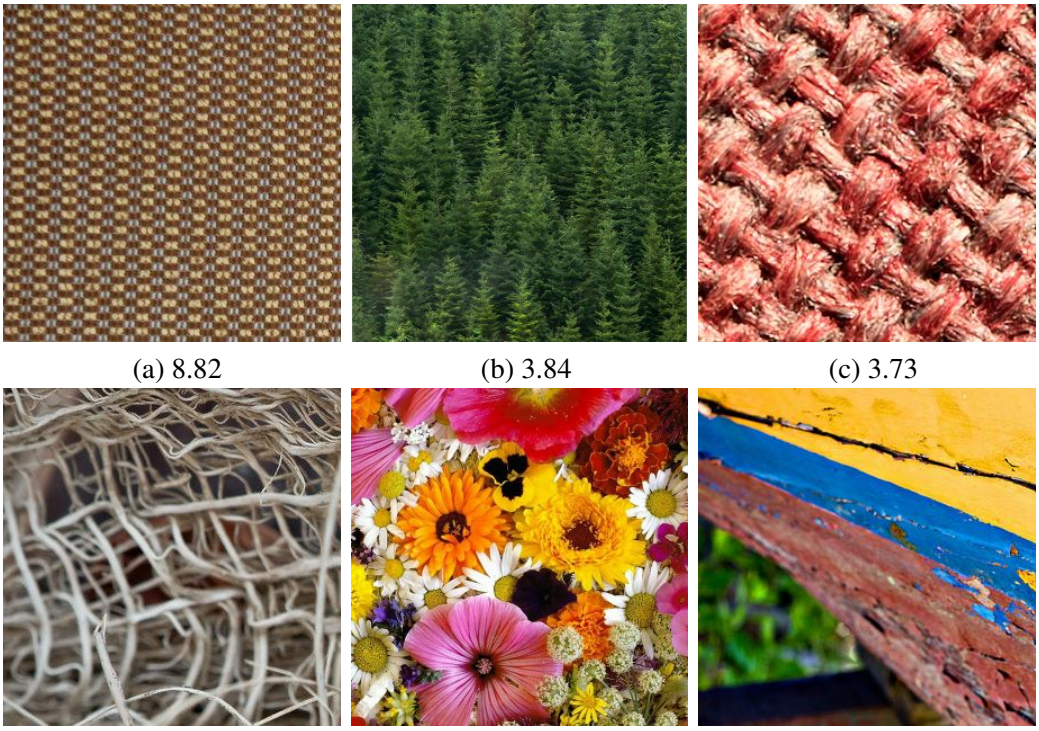
GIST features. Frequency analysis has proven very useful for texture analysis/synthesis [Gool et al., 1985; Manjunath & Ma, 1996; De Bonet, 1997], so features of this kind can best be included here as well. GIST [Oliva & Torralba, 2001] is used, where the implementation resizes images to 256×256 pixels, only considers one grid, and produces a feature vector of dimension 20.

7.4.2 Designed features

'Textureness'. Objects and scenes are more difficult to synthesize than actual textures. We train a classifier to distinguish textures from objects and scenes. The UIUC texture dataset [Lazebnik et al., 2005] delivered the positive samples (textures), and the 15-Scene dataset [Lazebnik et al., 2006a] the negative ones (objects/scenes). Linear SVMs were used as the classifier with GIST [Oliva & Torralba, 2001] as the feature. The classification score is taken as 'textureness'.

Homogeneity. Homogeneous textures are easier to synthesize than heterogeneous ones. Thus, it is desirable to have a feature measuring homogeneity. One possibility is based on co-occurrence matrices [Tuceryan & Jain, 1993], but is low-level and quite noise sensitive. We here propose a simple, yet more robust method based on our definition of homogeneity.

Definition: The homogeneity of an image is the expectation of visual similarity between two randomly-chosen local regions of the image.



(a) 8.82 (b) 3.84 (c) 3.73

(d) 3.12 (e) 1.80 (f) 1.63

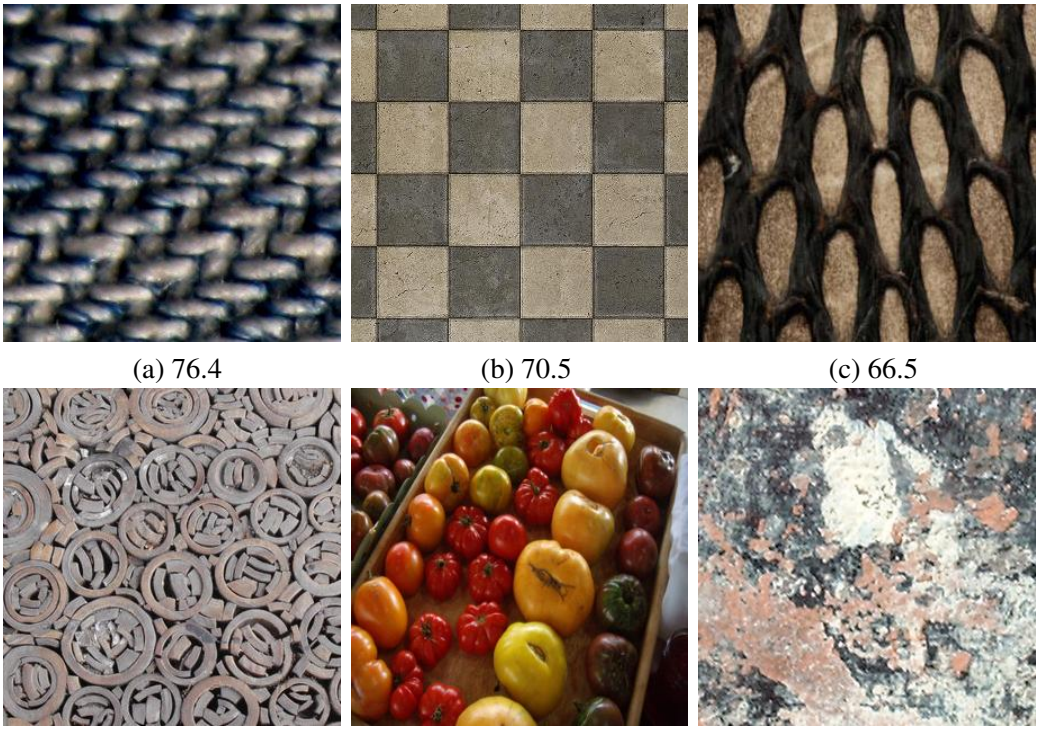
***Figure 7.3**: The homogeneity of texture examples detected by our method. Images are all of* 300×300 *pixels.*

In particular, given an image $X \in \mathbb{R}^{H \times W}$, we measure the average similarity over T (80 in the implementation) trials. In trial t, two regions R_1^t and R_2^t are sampled from X, and their distance $d(R_1^t, R_2^t)$ is measured. The homogeneity of X is then:

$$\text{Hom}(X) = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{d(R_1^t, R_2^t)}. \tag{7.1}$$

where $d(\,\cdot\,,\,\cdot\,)$ is the Euclidean distance, R_1^t and R_2^t are of the same size $\lfloor H/3 \rfloor \times \lfloor W/3 \rfloor$, and the positions of their top-left corners are sampled uniformly, at random from $\{(i,j) : i \in \{1, ..., \lfloor 2H/3 \rfloor\}, j \in \{1, ..., \lfloor 2W/3 \rfloor\}\}$. The regions are represented with bag-of-words. The dictionary is learned from X by k-means with 30 'word' centres and with 10×10 patches around every pixel (RGB values are used). See Figure 7.3 for the homogeneity of six texture examples detected by the method. Our homogeneity is more effective than the co-occurrence one [Tuceryan & Jain, 1993] because it uses regions rather than single pixels. It is also more robust because the word histograms yield some spatial invariance.

Repetitiveness. Textures are usually referred to as visual surfaces composed of repeating patterns, that are similar in appearance [Wei et al., 2009]. FFT features, of which the power spectrum is directly related to auto-correlation, have been used very early on [Gool et al., 1985; Tuceryan & Jain, 1993; Liu & Picard, 1996]. For periodic patterns, the auto-



(a) 76.4 (b) 70.5 (c) 66.5

(d) 49.7 (e) 39.4 (f) 32.4

***Figure 7.4**: The repetitiveness of texture examples detected by our method. Images are all of* 300×300 *pixels.*

correlation function is strongly peaked. Here we propose a related measure, also aimed at capturing imperfect repetitions (Figure 7.4), that is defined in the spatial domain.

The method draws on normalized cross correlation (NCC): an image $X \in \mathbb{R}^{H \times W}$ is cross-correlated with itself, generating an NCC matrix $D \in \mathbb{R}^{(2H-1)\times(2W-1)}$. The elements in the matrix are divided by the number of pixels involved in their calculation (different overlap as the image is shifted across itself). The borders of the matrix are not used due to the insufficient overlap there. The idea is that if X is repetitive, the following two properties should hold: (1) for a random moderate-sized region R of D, the difference between its maximum value and its minimum value should be large; (2) the minimum values of a set of randomly sampled R's (of the same size) should be very close. The philosophy behind (1) is that for repetitive textures, the auto-correlation function should exhibit peaks and valleys. Property (2) is derived from the fact that the distances between all 'repeated' versions should be similar.

Denoting by $Max(R_t)$ and $Min(R_t)$ the maximum and minimum values of the tth region R_t, we quantify the repetitiveness of X as

$$\text{Rep}(X) = \left(\frac{1}{T} \sum_{t=1}^{T} \frac{Max(R_t)}{Min(R_t)} \right) \times \frac{1}{\sigma(Min(R_t))} \tag{7.2}$$

where T (80 in the implementation) is the number of randomly sampled R's in D, and $\sigma(z)$ is the standard deviation of z. The size of R is set to $\lfloor H/5 \rfloor \times \lfloor W/5 \rfloor$. Too small a size cannot capture large-scale repetition, and too large a size looses discrimination power. See Figure 7.4 for examples of detected repetitiveness. Repetitiveness is akin to the Harmonicity feature of [Liu & Picard, 1996], but repetitiveness is more robust due to its pooling over local regions.

Irregularity. Irregular textures are harder to synthesize than regular ones [Liu et al., 2004], so we conjecture that the irregularity of textures is also relevant to their synthesizability. Although the irregularity of textures has been suggested before, we still lack a method to measure it computationally. We propose Ensemble Composition (EC) for such quantification. The idea is that if a texture is regular, composing any of its regions using image chunks from outside will be *cheap* (See images in Figure 7.5 to get the idea). We again do this over an ensemble – over T trials (80 in the implementation), we use the average composition energy to indicate texture irregularity. In the t-th trial, given an image X, we denote by R^t the region to compose, and by Y^t the rest of the image. The composition should have two properties: (1) the composited region should be similar to R^t; (2) the chunks from Y^t should be as continuous (large) as possible.

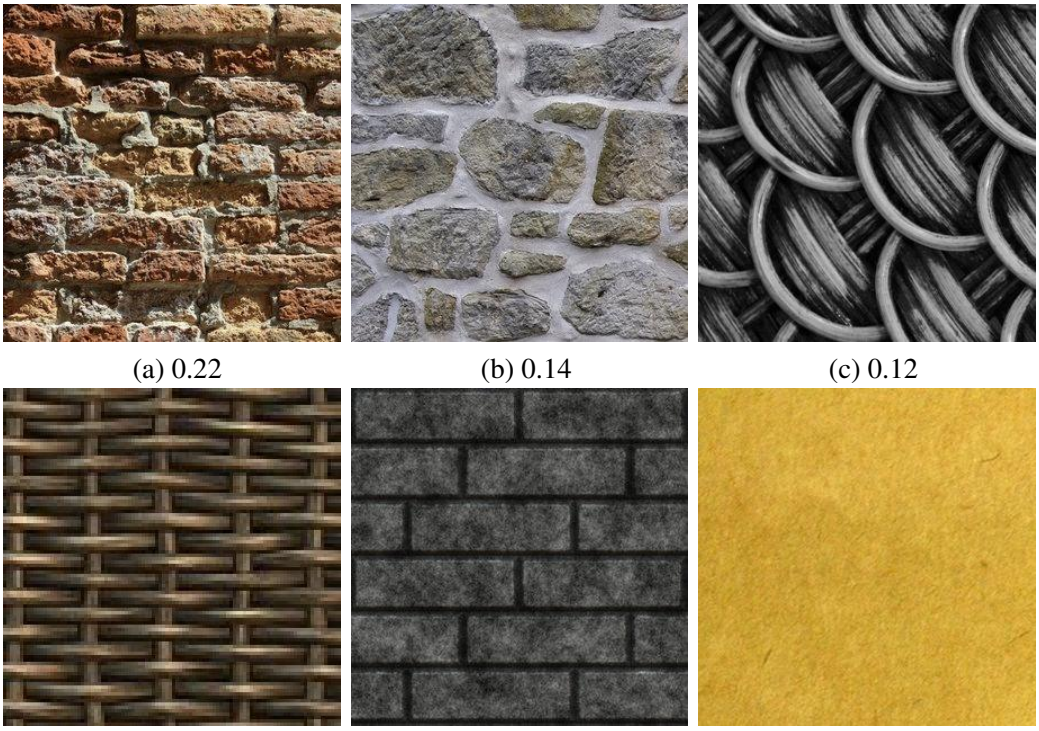
We formulate the composition task as a graph labeling problem with the following energy:

$$E(R^t) = \sum_{i \in R^t} D_i(l_i) + \lambda \sum_{\{i,j\} \in \mathcal{N}} V(l_i, l_j) \tag{7.3}$$

where l_i is the label assigned to pixel i in region R^t, and $\mathcal{N}$ is the neighborhood set of pixels in R^t. The label l_i represents the pre-defined offsets $\mathbf{s}_{l_i}$ between the composed pixels and composing pixels in the 2D image domain, that is, $l_i \in \{1, ..., \#(X)\}$. $D_i(l_i)$ denotes the cost of assigning the l_ith label to the ith pixel of R^t, and it is defined to reflect the similarity of pixel i and corresponding shifted pixel $i + \mathbf{s}_{l_i}$. To counteract noise, we use the Euclidean distance between the Schmid Filter responses [Schmid, 2001]; positive infinity is used when the shifted position falls outside of Y^t. For the smoothing term $V(l_i, l_j)$, we use the Potts model, i.e. $V(l_i, l_j) = 0$ if $l_i = l_j$ and 1 otherwise. λ is set to 50 to balance the two energy terms. By performing T trials, the irregularity of texture X is then defined as:

$$\text{IReg}(X) = \frac{1}{T} \sum_{t=1}^{T} E(R^t). \tag{7.4}$$

The energy is optimized by multi-label graph-cuts [Boykov et al., 2001]. In order to speed the optimization up, we employed the technique of dominant offsets proposed in [He & Sun, 2012] – only the dominant offsets (60 in our implementation) were considered as the labels. We also approximated the nearest neighbor search (for dominant offsets) by clustering patches into clusters (200 in our case) – patches in the same cluster are considered as neighbors. Our texture irregularity is similar to Boiman's image irregularity [Boiman & Irani, 2007], but we focus on textures and compose the image from itself instead of



(a) 0.22 (b) 0.14 (c) 0.12

(d) 0.10 (e) 0.08 (f) 0.05

***Figure 7.5**: The irregularity of texture examples captured by Ensemble Composition. Images are all of* 300×300 *pixels.*

composing general scenes from a dataset. Also, we provide an irregularity score for a given image by a new ensemble method.

It is noteworthy that homogeneity, repetitiveness, and regularity capture different properties. For instance, the texture in Figure 7.3(b) is homogeneous, but not repetitive and not regular. The texture in Figure 7.5(f) is homogeneous and regular, but not repetitive. In a nutshell, the 4 designed features are not orthogonal (*e.g.* repetitive textures are normally regular as well), but are complementary nonetheless. Moreover, we do not claim that the 7 features (general + designed) are optimal or exhaustive. Other features such as entropy, coarseness, directionality, could also be relevant to synthesizability.

7.4.3 Learning method

We attempt to computationally quantify texture synthesizability and to use this to aid synthesis. To those ends, we train (1) a regression model on the synthesizability scores (1, 0.5 and 0) to predict the synthesizability of a given image, and (2) an additional classifier to suggest the 'best' ETS method to synthesize it. Random Forest [Breiman, 2001] was used for both training tasks due to its fast speed. 30 trees were used for the forest.

Features	LBP	SFilter	GIST	Textureness	Homogeneity	Repetitiveness	Irregularity	General	Designed	All
>=Acceptable	88.1	76.8	84.1	77.2	88.7	82.2	76.7	88.5	93.1	94.5
>=Good	57.0	37.8	52.9	38.5	62.8	45.6	40.0	60.2	73.4	75.5

***Table 7.1**: The average precision (%) of synthesizability prediction with all individual features and as combinations, when recall is 1.*

7.5 Experiments

7.5.1 Learning synthesizability

In this section, we evaluate the contribution of all features to the prediction of synthesizability and to what degree it is learnable. All 7 single features and their 3 combinations were evaluated. The 3 combinations are: combination of the 3 general features (General), combination of the 4 designed features (Designed), and combination of all features (All). 30% of the dataset images were used for training, the rest for testing. We report results over 5 random training-testing splits. For evaluation, we performed two retrieval tasks and evaluated the average precision for different levels of recall: (1) retrieve images with 'good' scores (>=good); (2) retrieve images with 'good' or 'acceptable' scores (>=acceptable).

Quantitative evaluation. Table 7.1 shows the results for different, single as well as combined features when recall is set to 1, and Figure 7.8 shows the results for different levels of recall when all features are used. The table shows that every single feature is helpful. Homogeneity performs the best. It is also interesting that the combination of the 4 designed features performs substantially better than the combination of the 3 general texture features. This suggests that the designed features are indeed particularly relevant to synthesizability. This said, general texture features add to the power of the mix, given that the combination of all features yields the best performance. Also, from the highest precision scores (94.5% for >=acceptable, and 75.5% for >=good) we can conclude that image synthesizability is learnable and predictable. If only a fraction of well-synthesizable textures need to be retrieved, a very high precision can be expected (See Figure 7.8). This is very useful for choosing synthesizable textures from internet images.

Qualitative evaluation. Figure 7.1 and Figure 7.6 show examples together with their predicted synthesizability. The synthesizability predictor here was trained with all annotated images except for the image itself given for prediction. As can be seen, homogeneous, repetitive, and regular texture examples obtain higher scores. The low scores are caused by many factors, such as outliers, surface distortions, and complex structures. In Figure 7.6, the 'best' synthesised images by the ETS methods are also shown. The quality of the synthesized images is largely consistent with the predicted synthesizability score. This is crucial because it allows us to detect textures – also as image parts, see shortly – that

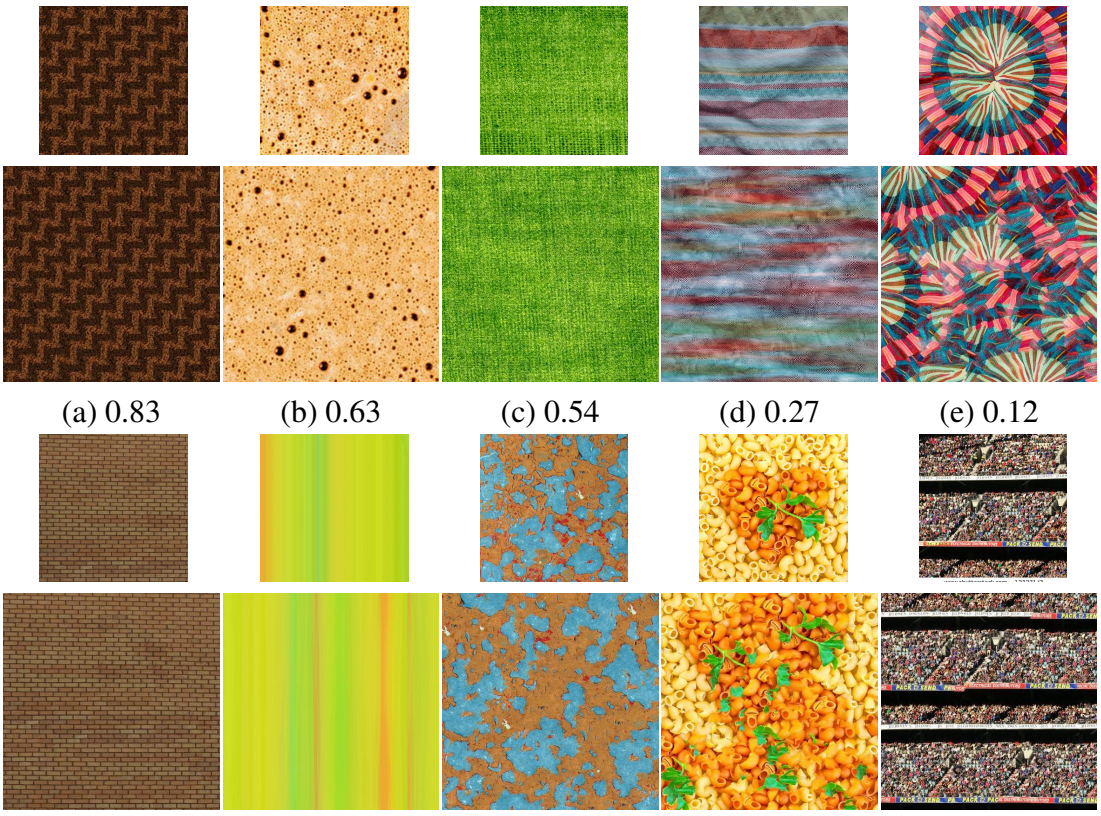
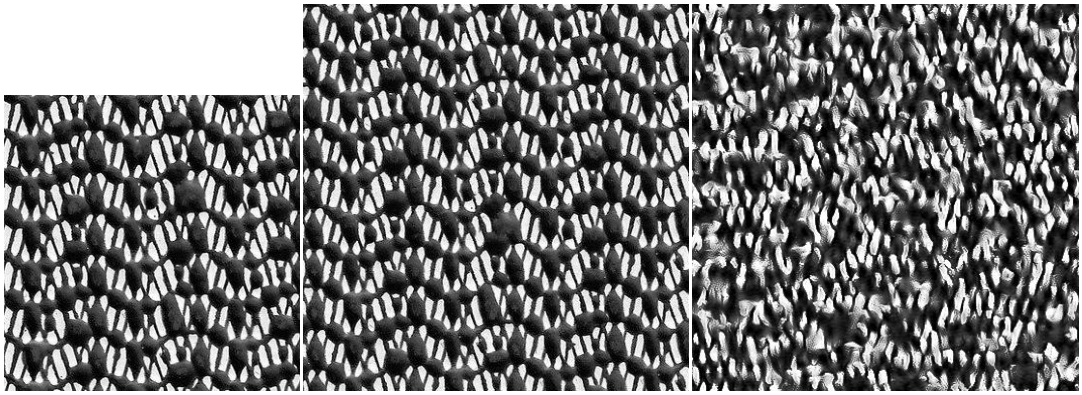


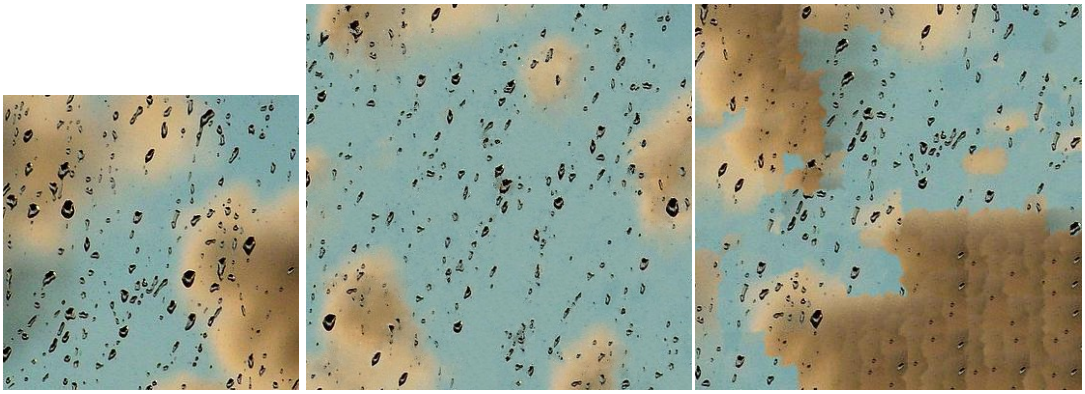
***Figure 7.6**: Synthesizability scores of texture examples and the 'best' synthesized textures by ETS methods. Top: exemplar; bottom: synthesized.*

can be synthesized well. As already claimed, the system can also suggest the 'best' synthesis method for a given texture example. Figure 7.7 shows two such examples, where results of the suggested method and results of a randomly chosen method are compared. It can be seen that our 'adaptive selection' is superior to random guessing. This is due to the fact the ETS methods all have their own philosophies and each one works better than the others for some textures, which necessariate an adaptive selection for the 'best' synthesis methods for a given texture example.

Failure cases. Of course, the method fails sometimes. The typical false positives (a high synthesizability score assigned to an image hard to synthesize) are images with fine, global, but irregular structures, *e.g.* crumpled paper and fabric, wood with year rings, foliage nerves, or hairs. It is hard to for the features to capture these subtle, but semantically crucial information. The typical false negatives (low score for synthesizable images) are heterogeneous textures such as some rust and cloud examples. This is probably because the space of valid textures for those is very large so that synthesized textures more easily fall inside the space. See Figure 7.9 for such examples.



(a) exemplar (b) suggested (Quilting) (c) random (Wavelet)



(d) exemplar (e) suggested (MMRF) (f) random (Quilting)

Figure 7.7*: The synthesized results of two texture examples by our suggested method and a randomly chosen method.*

Synthesizability with scales Textures in an image can be perceived and may differ at different scales. Thus, it is interesting to see how scales of textures affects their synthesizability. Figure 7.10 shows an example, where three scales of the same texture are used as the examples for synthesizability prediction. Zooming in, the synthesizability score drops – as long as the same 'textons' matter – as those textons increasingly take on the role of individual objects. This is in keeping with human intuition.

7.5.2 Trimming texture examples

In this section, image synthesizability is used for trimming images to more synthesizable parts. Given an image, the synthesizability of subimages is computed and compared. The most synthesizable subimage is then suggested. See Figure 7.11 for examples. 500 subwindows were randomly sampled, with a minimum size of 100×100 pixels and maximum size that of the entire image. The figure suggests that our method performs well for this task. It is thus possible to pick synthesizable texture examples from unconstrained images.

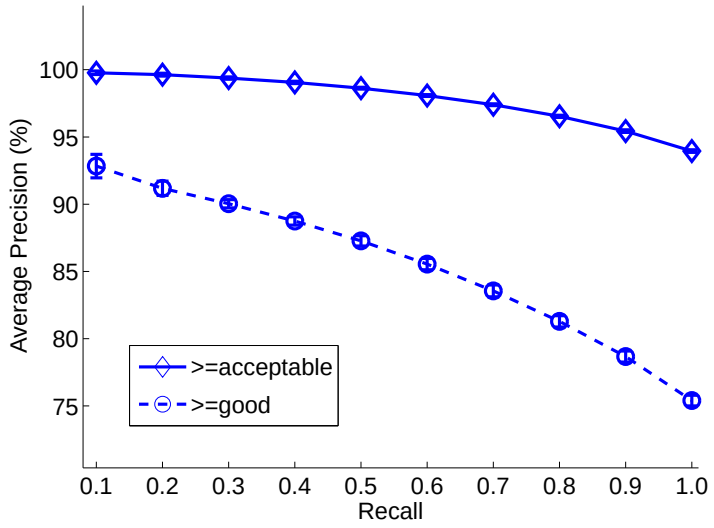
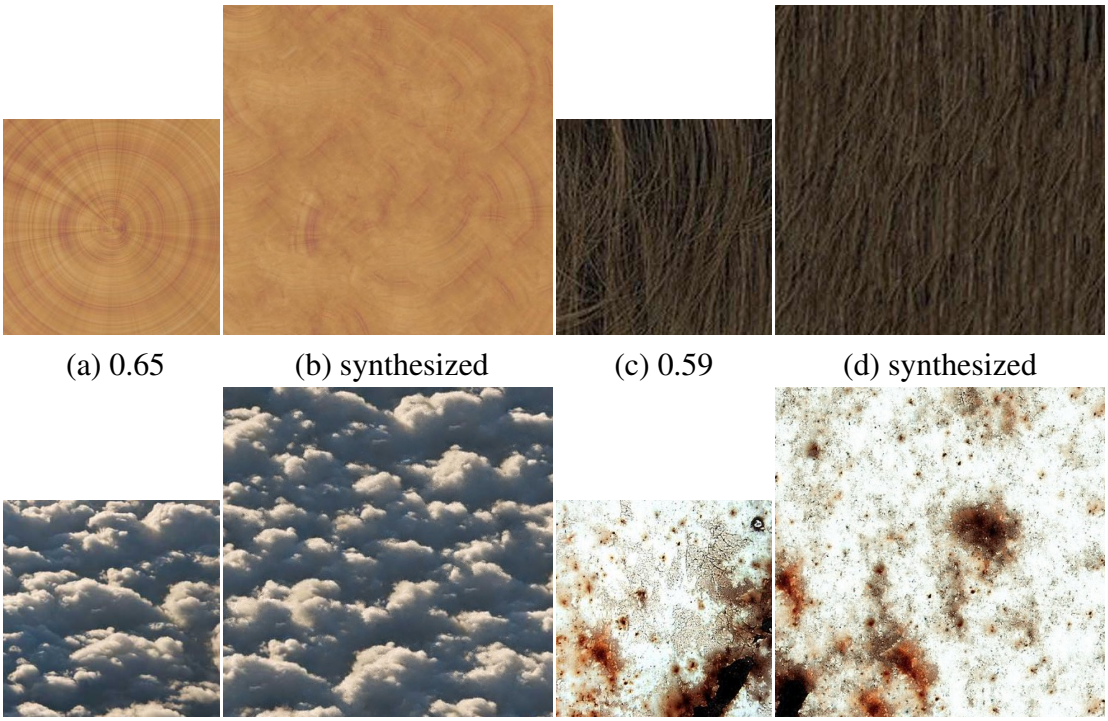


Figure 7.8: *The average precision of synthesizability prediction for different levels of recall, when all features are used.*



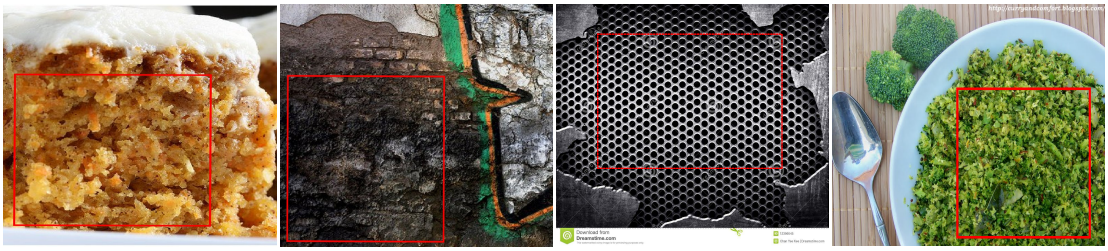
(a) 0.65 (b) synthesized (c) 0.59 (d) synthesized

(e) 0.35 (f) synthesized (g) 0.25 (h) synthesized

Figure 7.9: *Failure cases: the top shows false positives and the bottom false negatives. Exemplars are of* 300×300 *pixels.*

(a) 0.22 (b) 0.34 (c) 0.43

***Figure 7.10**: Synthesizability for different scales of the same texture.*



(a) RS:0.50, IS:0.19 (b) RS:0.40, IS:0.22 (c) RS:0.53, IS:0.40 (d) RS:0.51, IS:0.23

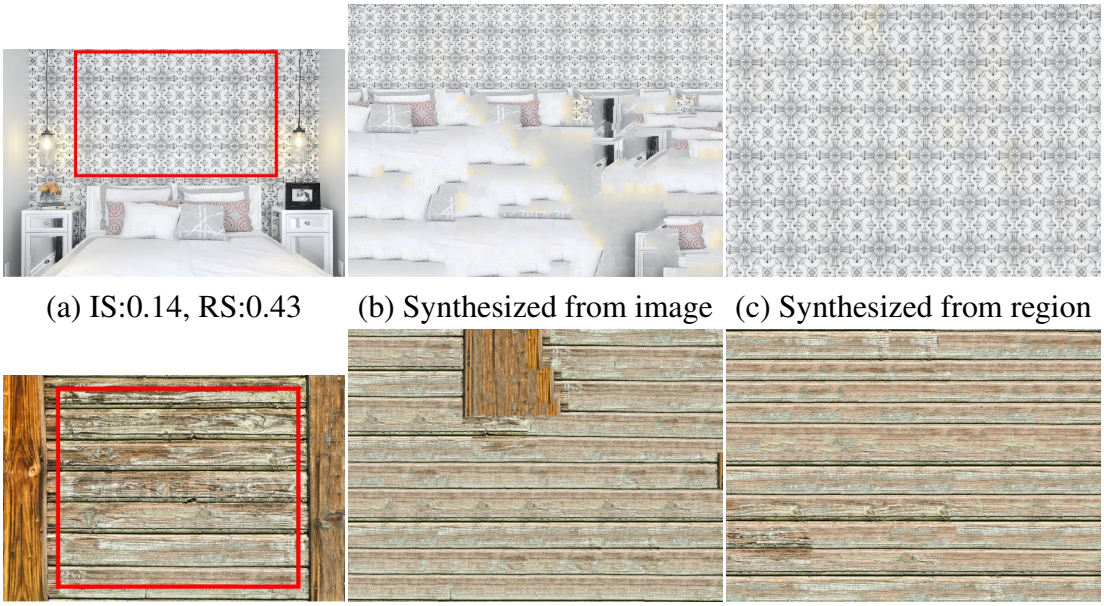
***Figure 7.11**: The most synthesizable region as detected. The synthesizability of the whole images (IS) vs. the selected region (RS) are given.*

Figure 7.12 illustrates that synthesis is superior for the selected windows compared to using the entire image. Note that if two windows receive the same/close synthesizability score, we prefer the larger window.

7.6 Conclusions

This chapter proposed synthesizability as a novel texture property and developed a computational predictor for it. We constructed a fairly large texture dataset and calibrated it according to synthesizability. A set of texture features have been proposed and, in some cases, designed for the learning, such as 'textureness', homogeneity, repetitiveness, irregularity. Extensive experiments show that image synthesizability can be learned and predicted computationally. It can be used to find good texture examples for synthesis, to detect good textures from unconstrained images for synthesis, and to choose an appropriate method to do so.

Our approach can be seen as kind of a 'winner-uses-all' strategy. Rather than aiming for the next best method, the idea is to rather pick one case-optimal method among several existing alternatives, with the goal of reaching success rates better than those of any in-



(a) IS:0.14, RS:0.43 (b) Synthesized from image (c) Synthesized from region

(d) IS:0.13, RS:0.50 (e) Synthesized from image (f) Synthesized from region

***Figure 7.12**: Synthesis results for the images on the left, starting from the entire image (IS; mid column) vs. from the selected region (RS; right column).*

dividual method. Such eclectic strategy could also be tried for other tasks: rather than creating ever stronger methods, choose from existing methods based on some preprocessing.

8

Performance Prediction: Under-, Properly-, or Over-Performed?

In addition to predicting the success or failure of an algorithm (Chapter 7), we also investigate the problem of 'fine-grained' performance prediction: is an algorithm under-, properly, or over-performed for the given visual sample? This 'fine-grained' prediction provides the algorithm a direction to alter its choices of parameters in order to generate more desirable results. The prediction also facilitates the use of the results by downstreamed applications. We propose a prove-of-concept method method for the task of image segmentation.

8.1 Introduction

Generic image segmentation has been part of computer vision and image processing communities since the advent of these fields many decades ago. The definition of the problem, although vague, is easy to give and understand: "to divide the pixels of an image into different pieces, where each piece represents a distinguished *thing* in the image." Martin *et al.* [Martin et al., 2001a] provided these instructions to annotators to create the Berkeley Segmentation Database (BSDS), which proved that the problem of image segmentation was, indeed, well defined, as humans provided consistent partitions of the images *up to refinement*. In other words, image segmentation is inherently a multi-scale problem.

We refer to *flat* image segmentation techniques as those whose output is a single partition of the image pixels into sets [Shi & Malik, 2000; Comaniciu & Meer, 2002a; Felzenszwalb & Huttenlocher, 2004]. In these cases, in order to capture the aforementioned multi-scale nature of objects, one needs to sweep different parameterizations to obtain multiple partitions that contain the different scales when working with flat segmentation techniques.

On the other hand, hierarchical segmentation produces a single multi-scale structure that aims at capturing the objects at all scales [Arbelaez et al., 2011; Kim et al., 2013; Salem-

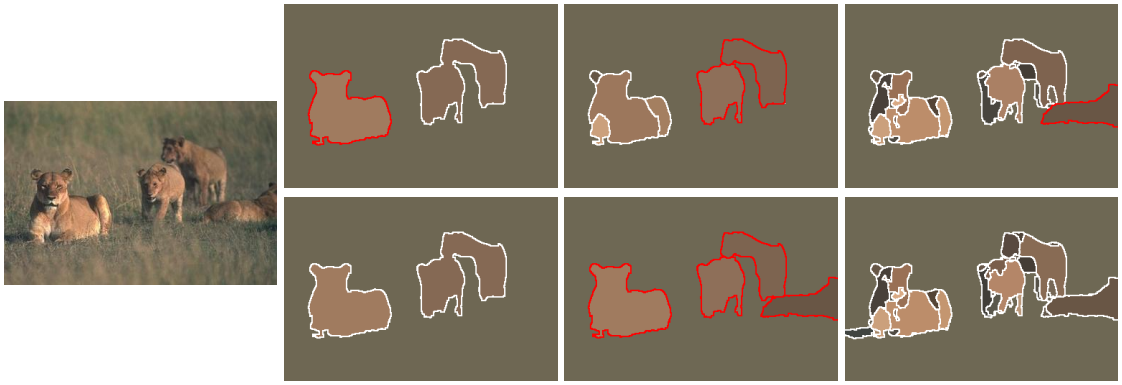


***Figure 8.1**: **Example of improved hierarchy alignment**: The original hierarchy (top row) needs three different flat partitions to represent the four objects (highlighted in red). Our aligned hierarchy (bottom row) correctly puts all objects in the same level.*

bier & Garrido, 2000; Ren & Shakhnarovich, 2013a; Arbelaez et al., 2014b]. These types of structures have been successfully used in image filtering [Salembier & Garrido, 2000], semantic segmentation [Lempitsky et al., 2011], object proposals generation [Arbelaez et al., 2014b], or video segmentation [Xu et al., 2013; Varas et al., 2015].

The representation power of these hierarchies comes at a cost, however, which is the difficulty to handle them from a practical (coding) point of view. While a flat partition can be represented by a matrix of labels of each pixel, hierarchical structures need a much more complex representation. In this context, the Ultrametric Contour Map (UCM) [Arbelaez et al., 2011] representation is the one that gained more traction and it is widely used in the literature. In it, *flattening* the hierarchy can be achieved by simply *thresholding* the UCM.

The process of *flattening* or *pruning* a hierarchy is therefore of paramount importance for segmentation, because it is the main proxy used towards the final application. This work presents a novel technique to improve the flattening of any given hierarchy, that is, to get better flat partitions from the same hierarchical segmentation.

Figure 8.1 motivates this work. In the first row we can see different flat partitions extracted from the same hierarchy. To get the regions representing the four lions we need to search in three different flat partitions, extracted at three different levels of the hierarchy. The second row shows our results, where the same hierarchy is *aligned* to have all objects represented in the same flat partition.

In other words, the threshold level of the hierarchy better relates with the scale of the objects, not only in the same image, but also across images. To further grasp the intuition of our work, Figure 8.2 shows a UCM and its interpretation as a region tree (a). In it, the needed regions to form the car are spread into different scale levels (thresholds of the UCM), as marked by the red band. Our proposed realigned hierarchy (b) aims at containing them all in the same scale.

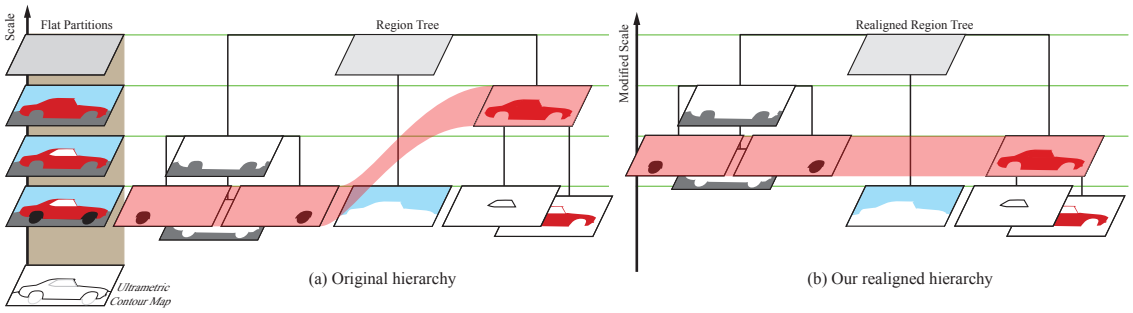


***Figure 8.2**: **Our proposed hierarchy realignment**: Given a hierarchy (a) in which the objects at the same scale are not well aligned (represented in the same scale level), we produce a realigned hierarchy (b) that has the similar-scale regions in the same level.*

Since the hierarchies are constructed based on low-level features (edges or color), the scale of the objects is not imposed to be coherent. We propose to learn the concept of object scale from mid-level features within the hierarchy. Our objective is to take advantage of these mid-level features as much as possible without getting to high-level features that would allow us to go beyond scale. This way, the global approach would be to construct the hierarchies using low-level features, and then exploit mid-level features to realign them, thus taking the maximum advantage of the most simple features possible.

Our alignment also aims at providing a global alignment among different images, that is, providing levels of scale that keep meaning even when changing images, allowing higher-level methods to generalize in a more straightforward manner. Specifically, we train a regressor to predict whether each region of the hierarchy is oversegmented, undersegmented, or correctly segmented; and we rescale the hierarchy according to the prediction of this classifier. Back to the example in Figure 8.1, the majority of regions in the first column (bottom) are undersegmented, in the middle column they are correctly segmented, and oversegmented in the last column.

We perform comprehensive experiments on BSDS500 using four different hierarchical segmenters. We obtain consistent improvements on all hierarchies which proves the usefulness of our approach and its generalization power. The remainder of the work is organized as follows. First, Section 8.2 gives a brief overview of the related work. Then Section 8.3 presents our algorithm for re-scaling and aligning hierarchies. We demonstrate the effectiveness of our method in the experiments in Section 8.4 and draw the conclusions in Section 8.5.

8.2 Related Work

Hierarchical Segmentation There is a rich literature of hierarchical segmentation. As stated in the introduction, our focus in this work is not to develop a better hierarchical segmentation algorithm, but to provide a better alignment of a given hierarchy. Hierarchical segmentation typically starts from various local information embedded in an affinity matrix, such as Pointwise Mutual Information [Isola et al., 2014], or multiscale local brightness, color, and texture cues [Arbelaez et al., 2011]. It then greedily constructs a hierarchy of regions by iteratively merging the most similar sets of regions according to a certain metric. The result of hierarchical segmentation is commonly represented as an Ultrametric Contour Map (UCM), where different levels of segmentation can be produced by applying different thresholds to UCM. This work proposes to realign the hierarchies in order to make the thresholds of the UCM more closely related to the scale of objects. Hierarchical segmentation has become the major trend in image segmentation and most of top-performance segmenters [Arbelaez et al., 2011, 2014b; Ren & Shakhnarovich, 2013a; Kim et al., 2013] fall into this category.

Multiple Segmentations Working with multiple segmentations at the same time has been used in the computer vision community for a long time, with the idea that, while none of the segmentations is likely to partition the image perfectly, some parts in some segmentations might be useful. Hoiem *et al.* [Hoiem et al., 2005] use this idea to estimate the scene structure. A similar idea was exploited by Russell *et al.* [Russell et al., 2006] to discover objects, and by Malisiewicz *et al.* [Malisiewicz & Efros, 2007] to improve the spatial support of regions for recognition. By realigning the hierarchies we aim to minimize the number of partitions from a hierarchy needed to obtain reasonable results, since we concentrate same-scale regions in the same partition. Our work also shares some similarities with [Xu et al., 2013], where they flatten supervoxel hierarchies in videos by finding a slice with uniform entropy.

Predicting Segmentation Quality by Classification Classification has been exploited to predict segmentation quality in many works. Ren *et al.* [Ren & Malik, 2003] use a linear classifier base on Gestalt features [Palmer, 1999] to distinguish good and bad segmentations. Their negative training data are generated by randomly placing a ground-truth mask over an image. A similar idea is used to select parameters by Peng *et al.* [Peng & Veksler, 2008] to select λ in graph-cut based interactive segmentation. They compute the segmentation with different λ, then select the one with highest predicted quality. More recently, Carreira *et al.* [Carreira & Sminchisescu, 2010], Arbelaez *et al.* [Arbelaez et al., 2014b], and Endres *et al.* [Endres & Hoiem, 2014a] use a regression forest to predict the good overlap between segments (object proposals) and ground truth objects. We use similar features to [Carreira & Sminchisescu, 2010], which are based on graph partition properties, region properties, and Gestalt properties.

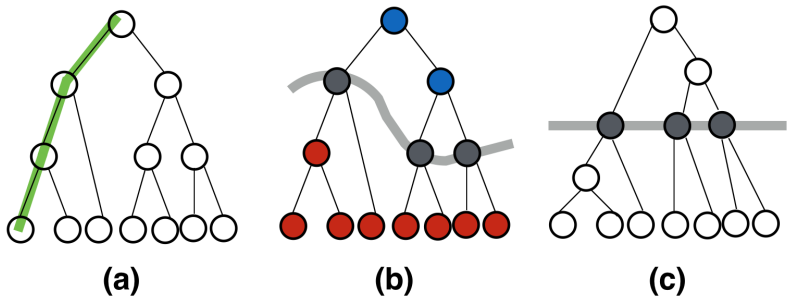


***Figure 8.3**: Examples of the slices and paths of the segmentation tree, where one path of the tree is shown in green (a) and one slice is shown in grey (b). In (b), all nodes in blue are in $\mathcal{L}^-$, and all nodes in red are in $\mathcal{L}^+$. Our approach re-aligns the hierarchy using the anchor slice. The aligned tree is shown in (c).*

Scale-aware Vision Algorithms Our work also bear a resemblance to the scale-aware algorithms for other vision tasks. For instance, exploiting the scale information has proven helpful for semantic image segmentation [Chen et al., 2015b; Bell et al., 2015] and pedestrain detection [Li et al., 2015]. [Dai et al., 2016b] show that vision algorithms employing resolution-improved images perform better than the algorithms using low-resolution images directly. Other scale-aware applications include object tracking [Klodt et al., 2013] and image thumbnailling [Sun & Ling, 2013].

8.3 Flattening and Re-scaling Hierarchies

As discussed in the introduction, while segmentation hierarchies contain a rich multiscale decomposition of the image, it is not trivial to distill such knowledge because the hierarchies generated by current methods are not fully scale-aware. Simply taking a layer yields a segmentation of which some parts are under-segmented while others are over-segmented. In this section, we present our method which aligns the scales of segmentation hierarchies, making image hierarchies easier to use in practice. We start with scale labeling, and then present the alignment strategy.

8.3.1 Flattening Hierarchies via Scale Labeling

Let's denote the segmentation tree of image I by $\mathcal{T}$, with node v_i indicating its i-th node. The nodes correspond to regions (segments) of I. Given $\mathcal{T}$, our task is to find a tree slice $\mathcal{L}$ to divide all nodes v_i's (segments) into three groups: $\mathcal{L}^-$, $\mathcal{L}$, and $\mathcal{L}^+$ indicating under-, properly- and over-segmented, respectively. See Figure 8.3(b) for an example of nodes in the three groups.

The visual representation of a slice can be seen in Figure 8.2 as red bands covering different regions. An example of the three types of slices can be found in Figure 8.1 (bottom row), where the left partition is mainly oversegmented, the middle one correctly segmented, and the right one undersegmented.

The problem is formulated as a three-class labeling problem. For each node v_i, we use $x(v_i) \in \{-1, 0, 1\}$ as its class label, with -1, 0, and 1 indicating the membership of v_i to $\mathcal{L}^-$, $\mathcal{L}$, and $\mathcal{L}^+$ respectively. Assume now that a function $f(v_i) : v_i \rightarrow [-1, 1]$ is provided to measure the granularity of image segments, where negative values stand for under-segmented, 0 for properly-segmented, and positive for over-segmented regions. The magnitude of $f(v_i)$ signals the deviation from being properly-segmented. Section 8.3.1 present the proposed learning algorithm for $f(v_i)$.

The labeling of all v_i's could be done by greedily taking the best-scoring class for each node. However, not any labeling represents a valid slice of the tree. Following the definition in [Pont-Tuset & Marques, 2012; Xu et al., 2013], a tree slice is a set of nodes such that every path $\mathcal{P}_n, n \in \{1, 2, ..., N\}$ from the leaf node $\bar{v}_n$ to the root node v_0 contains one and only one node v in the slice. See Figure 8.3 for the examples of the slices and paths.

From the nature of segmentation hierarchies, the labels of parent nodes v_i^p should be equal or smaller than their child nodes v_i. Intuitively, if a region is correctly segmented, the parent cannot be oversegmented. On the other hand, the parent of an undersegmented region will also be undersegmented. Putting the two constraints together, the labeling problem can be formulated as:

$$\begin{aligned} \hat{\mathbf{X}} &= arg\min_{\mathbf{X}} E(\mathbf{X}) \\ E(\mathbf{X}) &= \sum_{v_i \in \mathcal{L}} \#(v_i) \cdot \|f(v_i)\|^2 + \lambda \sum_{v_i \notin \mathcal{L}} \#(v_i) \cdot l(v_i) \\ \text{s.t} \quad \forall n : & \quad \sum_{v \in \mathcal{P}_n} \mathbb{1}_{\mathcal{L}}(v) = 1 \\ \forall v : & \quad x(v) >= x(v^p) \end{aligned} \tag{8.1}$$

where $\#(v)$ is the size (number of pixels) of segment (node) v, λ is a weighting value for the two energy terms, and $l(v_i)$ is the loss function defined for $v_i \in \{\mathcal{L}^-, \mathcal{L}^+\}$:

$$l(v_i) = \max(0, -f(v_i) \cdot x(v_i)). \tag{8.2}$$

The loss function penalizes two contradictory cases: (i) segments in the group of under-segmented that receive positive scores; and (ii) segments in the group of over-segmented that receive negative scores. The problem will be solved via dynamic programming, as explained in the following section.

Algorithm 5: Dynamic Programming in a Tree

Input: tree node v_i
if v_i is a leaf node **then**
  $\mathcal{C}_{v_i} \leftarrow \#(v_i) \cdot \max(0, f(v_i))$
  $E^*_{v_i} \leftarrow \#(v_i) \cdot \|f(v_i)\|^2$
else
  $\mathcal{C}_{v_i} \leftarrow \sum_{v_j \in \{v^c\}} \mathcal{C}_{v_j} + \#(v_i) \cdot \max(0, f(v_i))$
  $E^*_{v_i} \leftarrow$
  $min(\sum_{v_j \in \{v^c\}} E^*_{v_j} + \lambda \cdot \#(v_i) \cdot \max(0, -f(v_i)), \#(v_i) \cdot \|f(v_i)\|^2 + \lambda \cdot \sum_{v_j \in \{v^c\}} \mathcal{C}_{v_j})$
end if
return $\mathcal{C}_{v_i}, E^*_{v_i}$

Inference by Dynamic Programming

The optimization problem in Equation 8.1 is highly structured and can be solved recursively by Dynamic Programming. For the subtree rooted at node v, its optimal slice $\mathcal{L}(v)$ is either the node v itself or the union of the optimal slices of all its child nodes v^c's, depending on whose energy is lower. Thus, the problem has optimal substructure [Cormen, 2009] and so it naturally fits to the framework of dynamic programming to find the global optimal solution.

The problem proceeds from bottom to the top of the tree. For each subtree rooted at the current node v, the energy of $v \in \mathcal{L}(v)$ is computed and the energy of the optimal slices of all its child nodes is requested for comparison. The algorithm traverses back, and all comparison will be completed when the algorithm reaches the root node, and the global optimal of Equation 8.1 is obtained. The method is highly efficient with complexity $\mathcal{O}(N)$, where N is the total number of nodes. The global optimal of the energy can be found by applying Algorithm 5 to the root node, and the optimal slice is the corresponding set of nodes labeled to 0.

Predicting the Scales of Segments

In order to predict the scales (under-, properly-, or over-segmented) of the segments, we follow the route of modern computer vision systems to learn a predictor from human-annotated training data. To this end, we define a measure to compare the scale of an image segment $\mathbf{r}$ to that of the corresponding human-annotated segment $\mathbf{g}$. The correspondence is built up by computing the overlap between computer-generated segments and human-annotated ones – the most-overlapping human-annotated segment is taken as the ground-truth of the computer-generated ones. The overlap is computed with the Intersection over Union (IoU).

After having the ground-truth segment $\mathbf{g}$, the scale of the segment $\mathbf{r}$ is then defined as:

$$S(\mathbf{r}) = \frac{\#(\mathbf{g}) - \#(\mathbf{r})}{\max(\#(\mathbf{r}), \#(\mathbf{g}))}. \tag{8.3}$$

The value of $S(\mathbf{r})$ is in $[-1, 1]$, with negative values for *under-*, 0 for *properly-* and positive values for *over-segmented* regions, the magnitude of the values representing the extent of being under- or over-segmented, which casts to what we expected from $f(v)$ (*c.f.* Section 8.3.1).

With Equation 8.3, the *scales* of the segments by segmentation methods can be computed and used as the training data to train our scale predictor.

As to the learning method, we employ a regression forest as the predictor $f(v)$. As to the features, we use a set of low-, and middle-level features, mainly following the work done for object proposals [Carreira & Sminchisescu, 2010; Arbelaez et al., 2014b]. The features are designed to capture a variety of region properties, and the detailed list of the features is provided in Section 8.4.1.

The main difference between our prediction and the previous work [Carreira & Sminchisescu, 2010; Ren & Malik, 2003; Arbelaez et al., 2014b] is that they predict the quality of segments, while we predict the scale of the segments. Although numerous measures have been proposed, it is still very hard to quantify the quality of segments. The granularity of segments, however, is easier to quantify, and it also provides more specific information such as under-segmented or over-segmented.

8.3.2 Hierarchy Re-scaling with Labeled Scales

After setting the optimal slice, we use it as an anchor slice to stretch the segmentation tree correspondingly. In our experiments, we use the threshold value of each optimal node as a control point, and linearly interpolate the original hierarchy.

Segmentation tree can be represented in the form of Ultra-Contour-Map(UCM) [Arbelaez et al., 2011]. UCM is a matrix with the size $(2h-1)*(2w-1)$, where the h is the height of the original image, and w is the width. Therefore for each pair of neighboring pixels in the image, the value in the UCM matrix represents their boundary strength. And usually it is a value between 0 and 1. Segmentation of a certain scale can be extracted by thresholding the UCM with corresponding value. Without losing any generality, our algorithm directly manipulate on UCM due to its popularity and simplicity. The algorithm is summarized in Algorithm 6. We assume the functions **Boundary** to find the corresponding elements of boundary of a region $\mathbf{r}$ in the UCM matrix, and **InnerArea** to find its inner area.

By the presented algorithm, we perform locally linear transform on UCM map, and align the optimal slice to threshold 0.5, which makes it easier for later use. No information in the original hierarchy is lost during the rescaling process.

Algorithm 6: Rescaling Hierarchy

Input: Optimal Slice $\mathcal{S}$, UCM map M_{ucm}
for $\mathbf{r} \in \mathcal{S}$ **do**
  $\mathbf{b} \leftarrow$ **Boundary**$(\mathbf{r})$
  $\mathbf{a} \leftarrow$ **InnerArea**$(\mathbf{r})$
  $m \leftarrow min(M_{ucm}(\mathbf{b}))$
  $M_{ucm}(\mathbf{a}) \leftarrow M_{ucm}(\mathbf{a}) * 0.5/m$
end for
$\mathbf{b_{all}} \leftarrow$ **Boundary**$(\mathcal{S})$
$m_{min} \leftarrow min(M_{ucm}(\mathbf{b_{all}}))$
$M_{ucm}(\mathbf{b_{all}}) \leftarrow m_{min} + \frac{(M_{ucm}(\mathbf{b_{all}}) - m_{min})}{2(1-m_{min})}$

8.4 Experiments

We evaluate our approach on the segmentation hierarchies generated by multiple different segmentation methods, and further examine its usefulness on the task of object segmentation. The goal is to demonstrate that the proposed method is able to improve general segmentation hierarchies and the improvement is reflected to high-level vision tasks as well.

8.4.1 Experiment Settings

Dataset: We benchmark the performance of our approach on BSDS500 dataset [Arbelaez et al., 2011], which includes 500 images, with 200 for training, 100 for validation, and 200 for testing. Each image is annotated by 5 different people on average. As to evaluation, we deploy three standard metrics: Segmentation Covering (SC), Probabilistic Rand Index (PRI), and Variation of Information (VI). Readers are referred to [Arbelaez et al., 2011] for details about the dataset and the evaluation metrics.

Candidate methods: As to the candidate hierarchical segmentation methods, we chose the following four methods due to their popularity and good performance:

- gPb-owt-ucm [Arbelaez et al., 2011]: a widely-used hierarchical segmentation method. Discriminative features are learned for local boundary detection and spectral clustering is applied on top of it for boundary globalization.
- MCG [Arbelaez et al., 2014b]: a unified framework for segmentation and object proposals. It combines information from multiple resolutions of the image and achieves the state-of-the-art results for both image segmentation and object proposals.

- SCG [Arbelaez et al., 2014b]: the single-resulution, faster version version of MCG. It gets competitive results at a fraction of the cost of MCG.
- PMI [Isola et al., 2014]: a recent work for unsupervised boundary detection. It can be applied for image segmentation as well in order to generate a hierarchical segmentation.

Training: The training set and the validation set of BSDS500 are pooled together as the training set for our regression forest. The four segmentation methods are used to generate hierarchies, over which the training samples (segments) are extracted. We train method-specific regression forests as the scale predictor. Since a large portion of regions in the hierarchies are very small and features extracted from them are not reliable, we exclude regions smaller than 50 pixels for the training of the predictor.

Specifically, for each region $\mathbf{r}$, we find its corresponding ground-truth region $\mathbf{g}$ by taking the human-annotated one with the highest covering score. The relative scale of $\mathbf{r}$ is then computed with Equation 8.3 for the regression target of $\mathbf{r}$. As to the features for $\mathbf{r}$, we draw on the success of object proposals [Carreira & Sminchisescu, 2010; Arbelaez et al., 2014b]. There, a large pool of middle-level features have been defined for segment description. The features used are summarized as follows:

- Graph partition properties: cut, ratio cut, normalized cut, unbalanced normalized cut.
- Region properties: area, perimeter, bounding box size, major and minor axis lengths of the equivalent ellipse, eccentricity, orientation, convex area, Euler number.
- Gestalt properties: inter- and intra-region texton similarity, inter- and intra-region brightness similarity, inter- and intra-region contour energy, curvilinear continuity, convexity.

Readers are referred to [Carreira & Sminchisescu, 2010] for the details of these features. This list is definitely not exhaustive. More high-level features, *e.g.* by object detection, could be added.

Although these features are simple, extracting them for all layers of the segmentation hierarchies can be costly. Doing so is also unnecessary, as most segments at one layer are very similar to those from the parent layer and the child layer. Thus, we only extract the features from a subset of layers sub-sampled from the hierarchies. The layers are uniformly sampled over the range of UCM values.

As to the parameters of our method, we set 100 trees for the random forest. λ in Equation 8.1 is set to 0.1 to balance information from the three groups, because there are more segments over and under the optimal slice $\mathcal{L}$.

	Covering (↑)		PRI (↑)		VI (↓)	
	ODS	OIS	ODS	OIS	ODS	OIS
MCG	0.61	0.67	0.83	0.86	1.57	1.39
MCG-aligned	0.63	0.68	0.83	0.86	1.53	1.38
SCG	0.60	0.66	0.83	0.86	1.63	1.42
SCG-aligned	0.61	0.67	0.83	0.86	1.61	1.41
gpb	0.59	0.65	0.83	0.86	1.69	1.48
gpb-aligned	0.60	0.66	0.83	0.86	1.66	1.46
PMI	0.53	0.59	0.76	0.81	2.03	1.80
PMI-aligned	0.54	0.59	0.76	0.81	2.01	1.80

***Table 8.1**: The results of our aligned hierarchies with a comparison to the original hierarchies.*

	Covering (↑)		PRI (↑)		VI (↓)	
	ODS	OIS	ODS	OIS	ODS	OIS
Ncut [Shi & Malik, 2000]	0.45	0.53	0.78	0.80	2.23	1.89
Felz-Hutt [Felzenszwalb & Huttenlocher, 2004]	0.52	0.57	0.80	0.82	2.21	1.87
Mean Shift [Comaniciu & Meer, 2002b]	0.54	0.58	0.79	0.81	1.85	1.64
Hoiem [Hoiem et al., 2011]	0.56	0.60	0.80	0.77	1.78	1.66
gPb-owt-ucm [Arbelaez et al., 2011]	0.59	0.65	0.83	0.86	1.69	1.48
ISCRA [Ren & Shakhnarovich, 2013b]	0.59	0.66	0.82	0.85	1.60	1.42
PFE+mPb [Yu et al., 2015]	0.62	0.67	0.84	0.86	1.61	1.43
PFE+MCG [Yu et al., 2015]	0.62	0.68	0.84	0.87	1.56	1.36
MCG [Arbelaez et al., 2014b]	0.61	0.67	0.83	0.86	1.57	1.39
MCG+Ours	0.63	0.68	0.83	0.86	1.53	1.38

***Table 8.2**: Segmentation results on BSDS500 test set, with a comparison to the state-of-the-art competitors.*

8.4.2 Results

The results of our method evaluated on top of the four candidates segmentation approaches are summarized in Table 8.1. As shown in the table, the improvements achieved by our alignment are considerable and, more importantly, they are consistent across different methods. The method improves more on ODS than OIS, this is because OIS accesses the ground-truth segmentations to search for the best-performing threshold, which somehow diminish the learned knowledge. We argue that ODS is more practical than OIS in a real vision systems, because for real applications there is no human-annotated segmentations.

Figure 8.5 shows qualitative results of different hierarchies. Our approach shows a consistent improvement over the original results. Since our approach if scale-aware, regions are of similar scale across images after alignment. Thus our method demonstrates better ability of preserving regions across images. In Figure 8.6 we show segmentation examples of MCG and aligned MCG by our method. As the figure shows, the aligned hierarchies generate characteristics closer to what human expect when flat segmentations are sampled out of the hierarchies. More particularly, after alignment, sampled segmentations of the hierarchies generate consistent responses across all parts of the image: all parts under-segmented, to all parts properly-segmented, and finally to all over-segmented while sampling from the top to the bottom of the hierarchies. This alignment greatly simplifies the use of hierarchical image segmentation for other vision tasks. Figure 8.5 shows qualitative results with different hierarchies. Our approach shows a consistent improvement over the original results. Since our approach is scale-aware, regions at the same level of the hierarchy are of similar scales across all areas of the images after the alignment. Thus our method demonstrates better ability of preserving regions across images.

We also tested the method in the scenario where the random forests are trained with segments from all of the four methods, and applied to all of them at test time. This gives slightly poorer results but in turn shows that our method can be applied in a method-agnostic approach.

8.4.3 Comparison to Other Methods

As the previous section shows, the MCG aligned by our method generally performs the best. Here, we compare MCG-aligned to other competing methods. The results are summarized in Table 8.2 and demonstrate that segmentation quality can be improved by our alignment. In particular, the aligned MCG achieves the best result in Covering and VI. After alignment, the results are on par with the newest method of PFE+MCG [Yu et al., 2015]. It is noteworthy that our method and theirs are complementary, and the combination of the two may yield even better results. Their method is to improve feature embedding for a better local distance measure, while we aim to improve the hierarchy of existing segmentation methods.

8.4.4 Evaluation towards Object Segmentation

Segmentation *per se* is rarely the final objective of real applications, it is rather a middle tool towards, for instance, object segmentation [Arbelaez et al., 2014b] or semantic segmentation [Lempitsky et al., 2011]. This section is devoted to show that better aligned hierarchies also help in this scenario.

In this case we firstly perform the evaluation using the object annotations provided on the BSDS300 set by [Endres & Hoiem, 2014b] (we retrain on only BSDS300 train instead of

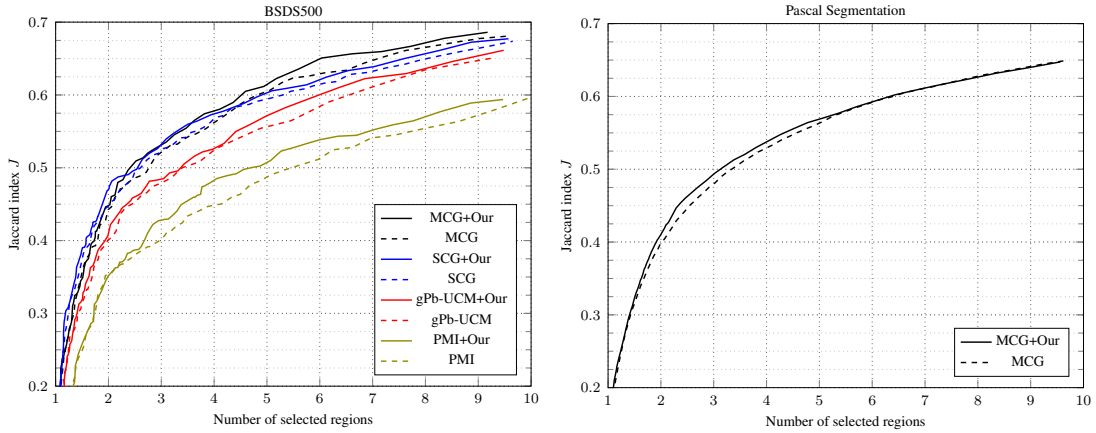


***Figure 8.4**: **Flattened hierarchies for object detection**: Achievable quality by an oracle with respect to the number of regions needed*

BSDS500). The intuitive idea is to measure how well we can segment these objects by *selecting* regions from the different flattened hierarchies.

Figure 8.4 shows the achievable quality that an oracle could reached if selecting the regions from the original hierarchies or the ones with our newly-proposed alignment. The X axis corresponds to the number of needed regions, *i.e.* , the lower the better.

We can observe that the aligned hierarchies consistently need less regions to get the same quality in all the tested hierarchies. In PMI, for instance, we need to select 5 regions to achieve the same quality that we can get with 4 on the aligned hierarchy. The combinatorial space of all possible 4-region combinations is significantly smaller and thus the search is more probable to succeed. On the other direction, if we limit the number of regions we get improvements up to 3 points (9%) in the achievable quality.

To further illustrates the scalability of the hierarchy alignment on larger dataset, we evaluated our alignment algorithm on Pascal VOC 2012 Segmentation set [Everingham et al., 2010b] and Microsoft COCO [Lin et al., 2014a]. We retrain our scale predictor using the training set of Pascal 2012. In Pascal 2012 dataset, only the segmentation of foreground objects are given, in contrast to BSDS which is fully annotated. Thus during training we only consider all the segments that have overlap with foreground object annotations. The scale predictor is trained as described in Section 8.3.1, the only difference is that g can only be foreground object. This strategy introduces extra bias towards foreground objects, because no information about the scale of background is given in the training process. However, we are still able to improve alignment of segmentation hierarchies . As shown in Figure 8.4, we see that for the range of 2-3 regions (the one in which the MCG object proposal work), the aligned hierarchy provides a 2.5-point improvement ($\sim$6%), which shows that our method generalizes to larger datasets.

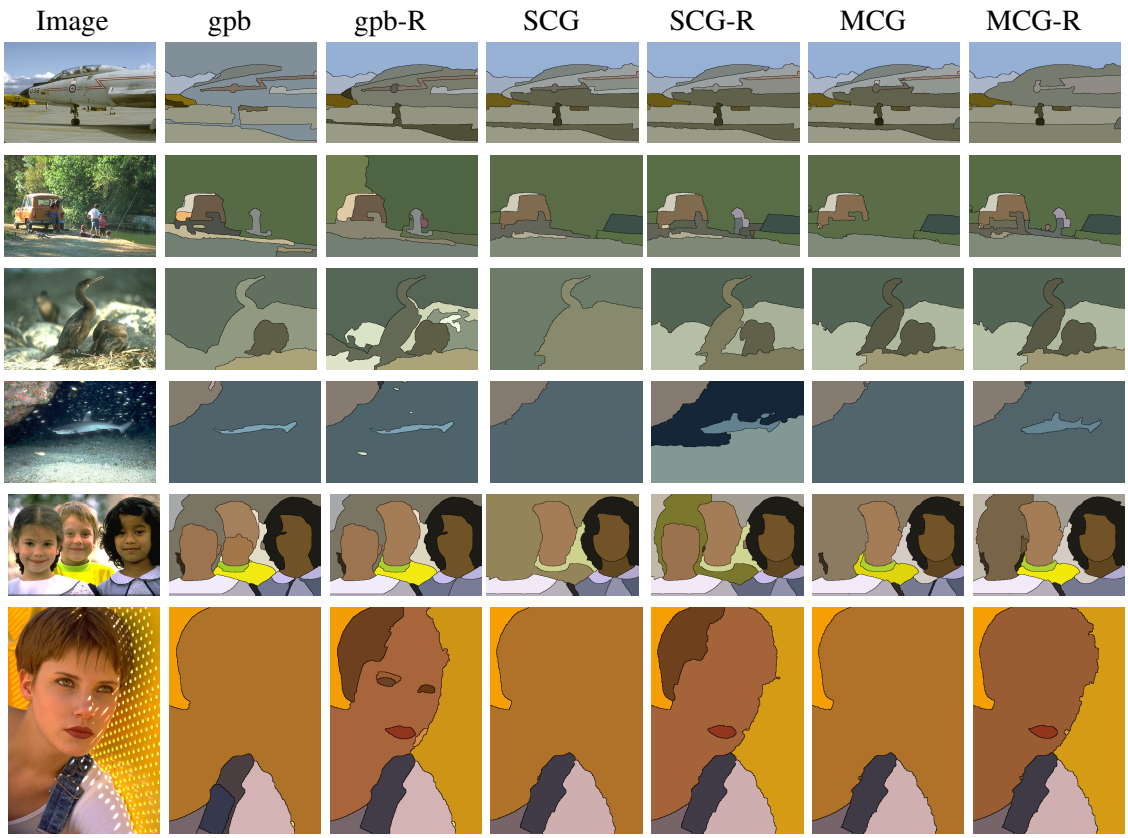


***Figure 8.5**: Comparison of segmentation results, hierarchies are flattened by optimal dataset scale(ODS).*

8.4.5 Running Time

Our approach takes approximately 3 seconds in total for each image, of which 2.39 seconds are spent on feature extraction from the segments. The prediction of regression forest takes about 0.45 seconds, and the dynamic programming takes 0.05 seconds for the inference. Finally, 0.11 seconds are spent for re-scaling the UCM. All times are measured on a standard desktop machine.

8.5 Conclusion

In this work, we presented a novel technique to align segmentation hierarchies. We proposed a method to learn and predict the scale of segments. We formulated the scale prediction for the segments in a hierarchy as a graph label problem, which is solved by dynamic programming. With the labeled scales as constraints, we then re-align the segmentation hierarchies by stretching the UCM maps. The method is evaluated on four

different segmentation hierarchies on BSDS500, and it consistently improves their quality. We also showed that the improvement of segmentation hierarchies by our alignment is reflected well to a higher-level task of getting object segmentations.

Figure 8.6: *Results of MCG(first row) and MCG results improved by our approach(second row). Original images are shown in the left most. Segmentations of optimal-dataset-sclae(ODS) are given in the middle. And from left to right are different scales, from fine to coarse. Red bounding box indicates the scale with best results achieved by MCG, and blue box for ours. It can be seen that our approach provides better alignment, both across images and within one image.*

9

Performance Evaluation: Helpful for Other Tasks?

As stated in the introduction, verifying whether the solution to one vision task is good enough to be generally useful for other vision tasks allows for better synergy to solve the problem of computer vision. In this chapter, we evaluate the usefulness of image super-resolution for other vision tasks, when the state-of-the-art super-resolution methods are used.

9.1 Introduction

Image super-resolution (ISR) aims to sharpen smooth rough edges and enrich missing textures in images that have been enlarged using a general up-scaling process (such as a bilinear or bicubic process), thereby delivering an image with high-quality resolution [Freeman et al., 2002; Yang et al., 2010; Zeyde et al., 2012; Timofte et al., 2013; Dong et al., 2014; Dai et al., 2015d]. ISR systems can be used to adapt images to displaying devices of different dimensions, to map image textures to 2D/3D shapes, and to deliver pleasing visualization for data that are inherently low-resolution such as image or videos from surveillance cameras. Despite the popularity of ISR in the past years, their performance has merely been evaluated perceptually and/or by evaluation criteria reflecting perceptual quality such as PSNR and SSIM. Therefore, it is still unclear whether ISR is helpful in general to other vision tasks and whether the perceptual criteria are able to reflect the usefulness. This work answers the questions.

We here present reasons why ISR can be helpful for other vision tasks, in addition to improving perceptual quality. As we know, most of current vision systems consist in two phases: training and testing. Although features have been designed to overcome the influence of scale changes, it is still a blessing if 1) the training and testing images are of the same/similar resolution; and/or 2) input images can be converted to the resolution at

which the features and the models were designed. It happens quite common that training and testing data are of different resolutions, *e.g.* training images are from expensive sensors while testing images from cheap ones. If testing images are of higher resolution, down-sampling them with linear filters does the job. If the opposite holds, however, sophisticated ISR methods are required to super-resolve the testing images. Also, vision systems are often designed and optimized (*e.g.* the features) for images of the most 'popular' resolution at the time. ISR is useful to super-resolve images which are of lower-resolution than the images for which the features and models are designed and learned. One example is object recognition with surveillance cameras: popular features [Lowe, 2004; Dalal & Triggs, 2005; Razavian et al., 2014] for object recognition are designed for normal images which are of higher-resolution than surveillance scenes in general. For this case, even the training data and testing data are of the same resolution, ISR is still helpful by enabling feature extraction at an appropriate resolution.

In order to sufficiently sample the space of ISR methods and potential vision tasks, six ISR methods are chosen and evaluated on four popular vision applications. The ISR methods are Zeyde *et al.* [Zeyde et al., 2012], ANR [Timofte et al., 2013], A+ [Timofte et al., 2014], SRCNN [Dong et al., 2014], JOR [Dai et al., 2015d], and SRF [Schulter et al., 2015]. They are chosen because 1) they are popular and representative; 2) they have code available; and 3) they are computationally efficient. The four vision applications include image boundary detection, semantic image segmentation, digit recognition, and scene recognition. The tasks are chosen because they are representatives of current low- and high-level vision tasks. The data of digits is chosen because low-resolution inputs are very likely to occur in this field. For all the tasks, we apply standard approaches with varying modes of the input images: from low-resolution images, to super-resolved images by the six ISR methods, and to the high-resolution images. The experimental results suggest that ISR is helpful for these vision tasks if the resolution of the input images are low, and that the standard evaluation criteria, including PSNR, SSIM, IFC, and NQM, correlate generally well with the usefulness of ISR methods, but should not be used as the full proxies of the usefulness if high precision is required.

The work is organized as follows. Section 9.2 reports related work. Evaluation on the four vision tasks are conducted in Section 9.3.1 to Section 9.3.4. Finally, the paper concludes in Section 9.4.

9.2 Related Work

There is a large body of work addressing image super-resolution task. We breifly summarize them. The oldest direction is represented by variants of interpolation, such as Bilinear and Bicubic [Duchon, 1979; Thévenaz et al., 2000]. They represent the simplest and the most popular methods. However, they often produce visual artifacts such as blurring, ringing, and blocking, which follows the fact that their assumptions of smoothness

and band-limited image data hardly hold in real cases. Due to these reasons, more realistic priors and regularization have been developed, such as the sparse derivative priors in [Tappen et al., 2003], the PDE-based regularization in [Tschumperle & Deriche, 2005], the edge smoothness prior in [Dai et al., 2007], and gradient profile [Sun et al., 2008]. Despite the improvement by these methods, the explicit forms of prior are still insufficient to express the richness of real-world image data.

In recent years, example-based image super-resolution has raised the most attention due to its good performance and simplicity. In this stream, the task is to learn a mapping function from a collection of LR images and their corresponding high-resolution (HR) ones. The LR and HR data can be collected from the test image itself or from an external dataset. Methods [Freedman & Fattal, 2011; Glasner et al., 2009; Yang et al., 2011; Huang et al., 2015] in the former stream draw on the 'self-similarity' of images across scales, and have obtained great success. However, they are normally relatively slow because on-line learning is needed for the dictionary. Methods in the latter group rely on extra training data, unleashing the learning capacity of many learning methods. The KNN method [Freeman et al., 2002] and its variants [Chang et al., 2004; Yang & Yang, 2013; Dai et al., 2015d,c] have gained great attention. More sophisticated learning methods such as Sparse Coding [Yang et al., 2010; Kim & Kwon, 2010; Timofte et al., 2013, 2014; Wang et al., 2015], SVM [Ni & Nguyen, 2007], Random Forests [Schulter et al., 2015; Salvador & Perez-Pellitero, 2015; Riegler et al., 2015], and Deep Neural Network [Dong et al., 2014; Cui et al., 2014; Wang et al., 2015] have been applied widely to the task as well. One exceptional work is [Sun & Hays, 2012], using scene matching with internet images for image super-resolution. Since example-based methods with extra training data obtain state-of-the-art performance for ISR, our evaluation is focused mostly on this stream.

There is also a survey paper on ISR [Nasrollahi & Moeslund, 2014], providing an excellent summary of the theory and applications of ISR. [Timofte et al., 2015] exploits seven ways to improve the performance of general example-based ISR methods. The work most relevant to ours is [Yang et al., 2014a], where different ISR methods are evaluated. While sharing similarities, the two methods still differ significantly. [Yang et al., 2014a] conducted user studies for perceptual evaluation, solely with visual comparison and under evaluation criteria such as PSNR and SSIM. Our work, however, integrates ISR methods into systems of other vision applications and evaluates the usefulness of ISR to these vision tasks. There are also works employing ISR to improve the quality (resolution) of the input images of other vision algorithms, such as [Hennings-Yeomans et al., 2008] for face recognition and [Jing et al., 2015] for pedestrian identification. However, these tasks are specific and the ISR methods used are highly specialized. Our work, however, evaluates general ISR methods with a variety of popular vision tasks.

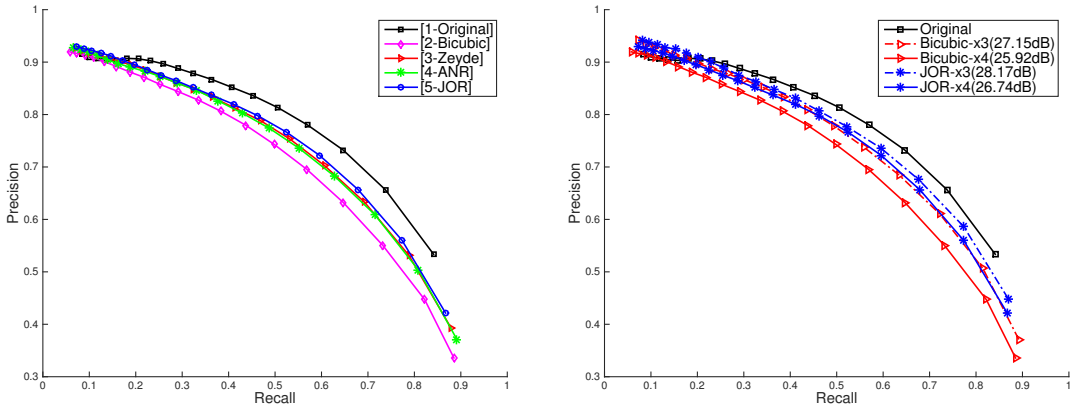
BSDS300		Bicubic	Zeyde *et al.*	ANR	SRCNN	A+	JOR	SRF	Original
×**3**	PSNR	27.15	27.87	27.88	28.10	**28.18**	28.17	28.17	—
	SSIM	0.736	0.770	0.773	0.777	**0.781**	**0.781**	0.780	—
	IFC	2.742	3.203	3.248	3.131	**3.374**	3.360	3.366	—
	NQM	27.42	31.80	31.95	31.28	32.35	**32.41**	32.40	—
	AUC	0.647	**0.675**	0.665	0.668	**0.675**	0.674	0.674	0.696
×**4**	PSNR	25.92	26.51	26.51	26.66	**26.77**	26.74	26.74	—
	SSIM	0.667	0.697	0.699	0.702	**0.709**	0.707	0.707	—
	IFC	1.839	2.195	2.231	2.117	**2.325**	2.316	2.293	—
	NQM	21.15	24.30	24.37	24.19	**24.98**	24.96	**24.98**	—
	AUC	0.595	0.647	0.635	0.650	**0.656**	0.655	0.652	0.696

***Table 9.1**: Average PSNR, SSIM, IFC, NQM values of ISR methods on BSDS300 and average AUC values of boundary detection via CBD [Isola et al., 2014] on the super-resolved images by the ISR methods and the original images. The best one is shown in **bold** and the second best underlined.*

9.3 Evaluation

In this section, we briefly describe the six ISR methods: Zeyde *et al.* [Zeyde et al., 2012], ANR [Timofte et al., 2013], A+ [Timofte et al., 2014], SRCNN [Dong et al., 2014], JOR [Dai et al., 2015d], and SRF [Schulter et al., 2015], followed by the evaluation on the four vision tasks. The six methods, starting out with the results of Bicubic interpolation, learn from examples to recover the missing high-frequency parts. As to the examples, the six methods are all trained with the same training dataset from [Yang et al., 2010], which consists of 91 images of flowers, faces, *et al.* . For implementation, we use the codes provided by the authors. Readers are referred to their papers for details. As to scaling factors, we evaluate with $\times 3$ and $\times 4$, which are commonly used in previous papers.

For datasets, we use the standard ones for the four vision tasks, though not the most challenging ones. To generate inputs for our evaluation, we downscale the original images of the datasets by factors $\times 3$ and $\times 4$ to create the low-resolution (LR) images and then upscale them by each of the six ISR methods to the resolution of the original images, which are then used as the inputs for the vision tasks. The standard approaches to the four tasks are then applied to all the six super-resolved versions of the images. The corresponding performances are recorded to evaluate the usefulness of the ISR methods for the vision tasks, with a comparison to Bicubic Interpolation, and the original images. We also evaluate the ISR methods on these datasets with four standard perceptual criteria [Yang et al., 2014a], namely PSNR, SSIM, IFC, and NQM, in order to see their correlation to the usefulness of ISR to these vision tasks.



(a) PR curves with scaling factor x4

(b) PR curves with scaling factor x3 and x4

***Figure 9.1**: Average PR curves of boundary detection via CBD [Isola et al., 2014] on the super-resolved images by some of the five ISR methods and on the original images of BSDS300. (a) curves for scaling factor x4, where SRCNN and A+ are not shown for visual clarity as they are very similar to JOR. (b) a comparison of scaling factor x3 and x4, where only Bicubic Interpolation and JOR are shown for visual clarity.*

9.3.1 Boundary Detection

Boundary Detection (BD) is a very popular low-level vision task and serves as a crucial component for many high-level vision systems [Martin et al., 2001b; Isola et al., 2014]. This section evaluates the usefulness of ISR methods for BD. We use Crisp Boundary Detection [Isola et al., 2014] (CBD), which is an unsupervised algorithm, deriving an affinity measure with point-wise mutual information between pixels and utilizing this affinity with spectral clustering method to detect boundaries. It produces pixel-level boundaries and achieves state-of-art results. The performances are evaluated on the BSDS300 dataset [Martin et al., 2001b]. The whole dataset consists of 300 images (200 for training and 100 for testing) along with human annotations. The quality of detected boundaries is evaluated by precision-recall (PR) curves, following Berkeley Benchmark [Martin et al., 2001b].

Table 9.1 lists the AUC values of BD on the eight sets of images, along with the values of PSNR, SSIM, IFC, and NQM of corresponding ISR methods. Fig. 9.1 shows the average PR curves. From the table and the figure, it can be observed that ISR methods do improve, over simple interpolation, the performance of BD when input images are of low-resolution. For instance, JOR improves the AUC by 0.06 when factor x4 is considered. This is because ISR methods perform better in increasing the resolution of the LR images to the resolution for which the BD method (CBD [Isola et al., 2014] in this case) was designed. CBD uses highly localized features to predict pixel-level boundaries, whose accuracy is affected largely by the recovered details locally. As a result, the six learning-based ISR methods all perform better than Bicubic Interpolation. This suggests that ISR

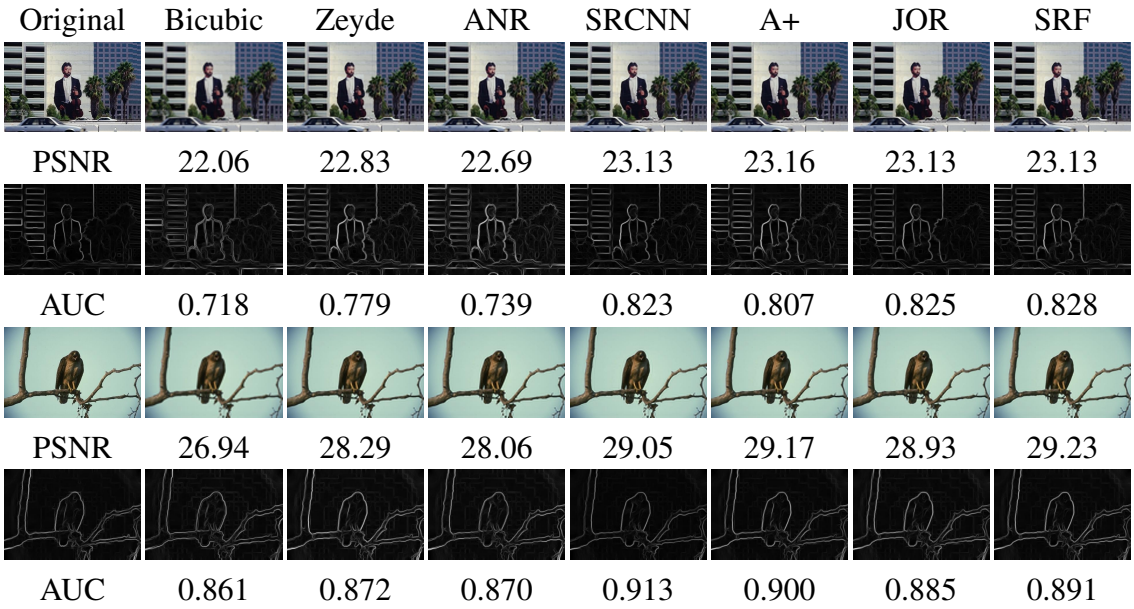


***Figure 9.2**: Super-resolved examples with their PSNR values and corresponding detected boundary maps by CBD [Isola et al., 2014] with their AUC values. Better seen on the screen.*

should be considered as a pre-processing step for BD if the input images are of LR. One may argue that adapting or re-training the BD method may increase its performance for LR images. It is true, but we have to admit that adapting or re-training the approach requites expertise of BD and deep understanding of the approach used. Enhancing the resolution of LR inputs, however, is much more straightforward for general practitioners, especially given the fact that BD is just one of such examples as shown in following sections.

It can also be found that the four standard perceptual criteria correlate quite well with the usefulness of ISR methods for the task of BD. ISR methods which yield better perceptual quality (based on the four perceptual criteria) often obtain better boundary detection results. However, perceptual criteria should not be considered as full proxies for the usefulness of ISR methods to BD. For instance, SRCNN outscores A+ in terms of PSNR while having a lower AUC value, when factor $\times 3$ is used. This suggests that measuring the usefulness of ISR methods for BD directly in a real system is necessary if a high precision is requited. In general, SRCNN, A+, JOR and SRF are among the most useful ISR methods for the task of BD for the dataset and approach considered. The third finding from the table and figure is that ISR methods are more useful when the scaling factor is larger, which means they are more needed when the input images are of very low-resolution.

In Fig. 9.2, we show visual examples, with the super-resolution results and their corresponding BD results. From the figure, it is evident that example-based ISR methods

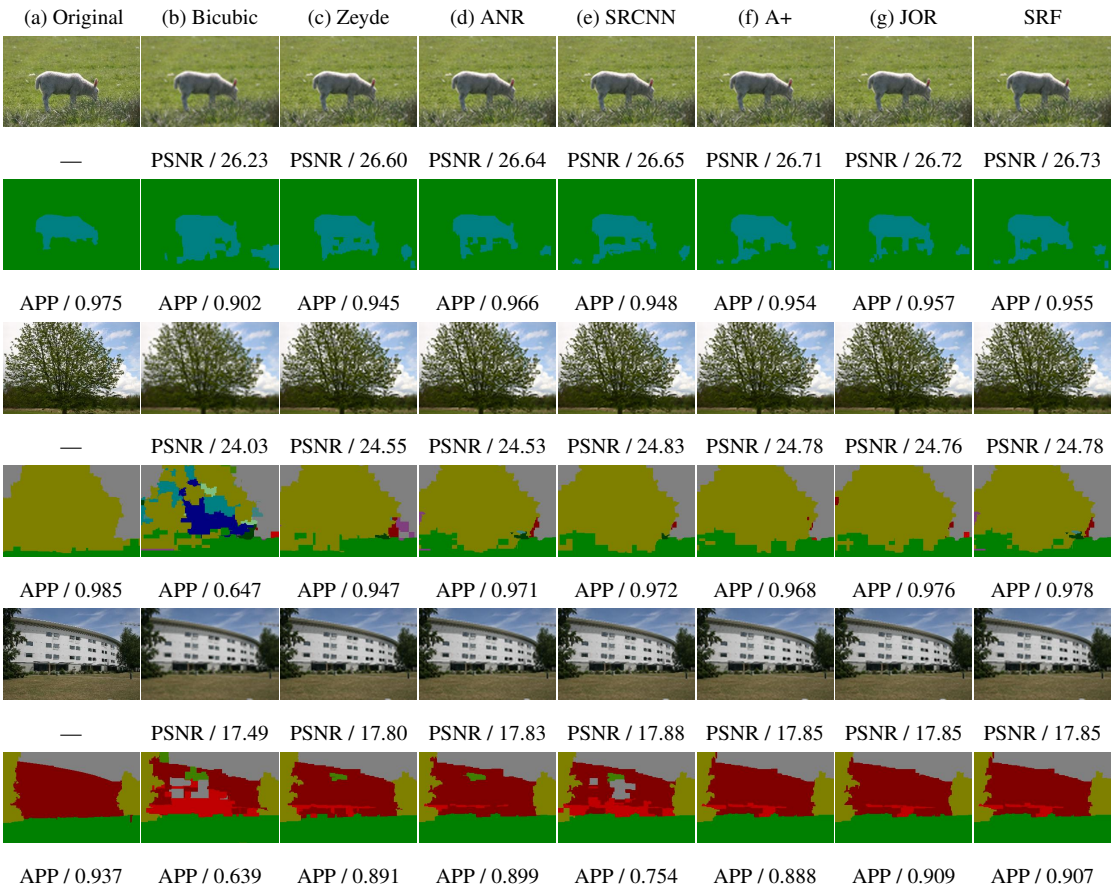


***Figure 9.3**: Examples for semantic image segmentation: super-resolved images with their PSNR values and the corresponding labeling results with their average precision over pixels (APP) are shown. Better seen on the screen.*

improve the quality of BD results with sharper true boundaries and fewer spurious ones. However, there is still a large room for improvement as the OB results on the super-resolved images by the ISR methods are still substantially worse than the result on the original ('HR') image.

9.3.2 Semantic Image Segmentation

In this section, we consider the task of semantic image segmentation, which aims to assign a semantic label to each pixel of the image, such as *tree*, *road*, and *car*. It is a very popular high-level vision task with a large number of methods proposed [Shotton, 2008; Gould et al., 2014; Long et al., 2014]. We follow the footsteps of most previous works on semantic image segmentation and choose the standard MSRC-21 [Shotton et al., 2006] dataset for the evaluation. MSRC-21 consists of 591 images of 21 semantic categories. For the segmentation method, we employ the recent approach [Gould et al., 2014] for its

***Table 9.2**: Average PSNR, SSIM, IFC, NQM, and labeling accuracy on MSRC-21 dataset, where APP indicates Average Precision over Pixels, and APC means Average Precision over Classes. The best performance is shown in **bold** and the second best is underlined.*

MSRC-21		Bicubic	Zeyde *et al.*	ANR	SRCNN	A+	JOR	SRF	Original
×**3**	PSNR	25.29	26.02	26.00	26.21	26.28	26.28	**26.35**	—
	SSIM	0.689	0.726	0.728	0.733	0.737	0.737	**0.738**	—
	IFC	2.677	3.214	3.250	3.131	3.390	3.396	**3.640**	—
	NQM	19.56	22.48	22.47	22.64	23.10	23.16	**23.20**	—
	APP	0.692	0.762	0.770	0.777	0.780	**0.783**	0.782	0.844
	APC	0.592	0.662	0.674	0.681	0.684	**0.687**	0.685	0.743
×**4**	PSNR	24.04	24.65	24.63	24.77	24.88	24.86	**24.90**	—
	SSIM	0.608	0.641	0.643	0.646	0.654	0.652	**0.660**	—
	IFC	1.694	2.043	2.066	1.992	2.171	2.151	**2.301**	—
	NQM	14.75	16.56	16.55	16.73	17.10	**17.12**	16.99	—
	APP	0.582	0.665	0.677	0.673	**0.682**	0.674	0.674	0.844
	APC	0.505	0.569	0.584	0.588	0.591	0.586	**0.605**	0.743

simplicity in order to better show the influence of ISR. [Gould et al., 2014] presents a fast approximate nearest neighbor algorithm for image labeling. They build a super-pixel graph from annotated set of training images. At test time, they transfer labels from the training images to the test image via matching super-pixels in the graph. The distance between super-pixels in the feature space is approximated by edge distance in the super-pixel graph where the edge weights are learned from the training set. This method shows comparable results to the state-of-the-art methods. For the implementation, we use the authors' code with the default settings.

In order to evaluate the ISR methods for semantic image segmentation, we train the method [Gould et al., 2014] with the original training images (*e.g.* the HR images) and test the trained model on eight versions of the testing images, created by down-sampling the original images and then up-solving them by the ISR methods to the resolution of the original images. Again, the performance is tested for scaling factor x3 and x4. Table 9.2 lists the results of all ISR methods, where the average precision over pixels (APP) and the average precision over classes (APC) are reported, along with the values of the four perceptual criteria. As we can see from the table, all the six ISR methods yield significantly better results than Bicubic Interpolation. Putting it into another way, these learning-based super-resolution systems, in addition to improving visual quality of LR images, do facilitate semantic labeling tasks and improve the performance substantially when the resolution of the testing images are lower than that of the training images. The results suggest that it is worth effort to integrate ISR methods into real image labeling systems if the resolutions of training and testing images are distinctive. This is highly probably the case for real semantic labeling systems where training images on the server

SVHN		Bicubic	Zeyde *et al.*	ANR	SRCNN	A+	JOR	SRF	Original
×3	PSNR	33.39	35.40	**35.73**	35.03	34.85	34.90	34.82	—
	SSIM	0.912	0.946	**0.949**	0.946	0.946	0.948	0.948	—
	IFC	2.050	2.331	**2.417**	2.291	2.389	2.346	2.355	—
	NQM	10.23	12.59	**12.91**	12.16	12.17	12.21	12.19	—
	Accuracy	0.766	0.774	0.777	**0.779**	0.778	0.775	0.778	0.793
×3[R]	PNSR	33.39	36.30	36.53	35.99	37.20	**37.26**	37.12	—
	SSIM	0.912	0.951	0.953	0.947	0.963	**0.964**	0.963	—
	IFC	2.050	2.484	2.550	2.427	2.726	**2.730**	2.701	—
	NQM	10.23	13.30	13.53	13.04	14.24	**14.25**	14.17	—
	Accuracy	0.766	0.775	0.774	0.773	0.783	**0.786**	0.778	0.793
×4	PSNR	29.08	30.63	30.72	**30.83**	30.45	30.47	30.11	—
	SSIM	0.787	0.842	0.847	0.849	0.847	**0.850**	0.845	—
	IFC	1.262	1.352	1.367	**1.368**	1.365	1.339	1.287	—
	NQM	6.211	7.864	7.978	**8.021**	7.732	7.720	7.453	—
	Accuracy	0.712	0.731	0.731	0.730	**0.737**	0.722	0.729	0.795
×4[R]	PNSR	29.08	31.07	31.06	31.00	**32.00**	31.71	31.77	—
	SSIM	0.787	0.858	0.862	0.856	**0.887**	**0.887**	0.886	—
	IFC	1.262	1.456	1.466	1.427	**1.639**	1.599	1.589	—
	NQM	6.211	8.268	8.286	8.209	**9.162**	8.872	8.962	—
	Accuracy	0.712	0.735	0.730	0.732	0.744	0.744	**0.749**	0.795

***Table 9.3**: Results of digit recognition on the SVHN dataset. The k-NN classifier is trained and applied on HOG features of each pair of super-resolved training and test sets. Methods marked with [R] are retrained using the unused digits of the SVHN dataset. The best performance is shown in **bold** and the second best is underlined.*

side are from expensive sensors and testing images on the user side are from cheap sensors such as cameras of an mobile phone. Another observation from the table is that the standard perceptual evaluation criteria correlate quite well with the usefulness of ISR methods for semantic image segmentation. This implies that good visual quality also facilitates computer systems for recognition. This can ascribed to the fact that the semantics are defined by human and computer are trained to conduct a human vision task which is of course very relevant to the perceptual quality of images. Also, ISR methods are more useful when the scaling factor is larger, which means they are more needed when the input images are of very low-resolution. The observation is consistent with the one we had for BD in Sec. 9.3.1.

In Fig. 9.3, we show three image examples, with the super-resolution results and their corresponding labeling results. From the figure, it is evident that ISR methods improve the quality of the labeling results. For instance, in the third example, results of Bicubic Interpolation labeled a large area of the building to sky, which is probably due to the detailed textures on the building are missing in the interpolated image. The missing texture are recovered (to some extend) by the example-based ISR methods, leading to better labeling results. Also, it can be found that RGB images that have small difference in perception

may lead to totally different labeling results, e.g. the tree in the second example. This implies that there is still room for computer recognition systems to improve in order to be as robust as human vision.

9.3.3 Digit Recognition

In this section, we test the usefulness of ISR methods for the task of digit recognition where the training images and the test image are both of low-resolution. We use the Street View House Numbers (SVHN) [Netzer et al., 2011] dataset which contains more than 100,000 images of house numbers obtained from Google Street View. Each image presents a single digit at its center and has the same size of 32×32 pixels. We select 26,032 and 10,000 images from the dataset as our training and test set. In order to evaluate the usefulness of ISR methods for digit recognition, we here down-sample all the images by factor x3 and factor x4, and up-sample the down-sampled images to the resolution of the original images by the ISR methods. As the SVHN dataset merely presents numbers from 0 to 9, it is highly specific and quite different from the training dataset from [Yang et al., 2010] that is used to train the ISR methods. Therefore, we re-trained all ISR methods with the unused images from the SVHN dataset, to study the generality of ISR methods. After adding the re-trained methods, we now have twelve datasets of super-resolved results, one dataset from Bicubic Interpolation and one dataset of the original images. As to the classifier, we use the k-NN with $k = 5$ for each of the eight image sets with HOG feature [Dalal & Triggs, 2005] as input. Other values of k yield a similar trend.

The classification performance is listed in Table 9.3. The table demonstrates that ISR methods do improve the performance of digit recognition over simple interpolation, and that the four perceptual criteria correlate quite well with the usefulness of ISR methods for digit recognition. The reason of the improvement is that HOG feature was designed for images of normal resolution, so by applying ISR methods to the LR input images, HOG can be extracted from images of suitable resolution. However, we find that with the standard, general training dataset [Yang et al., 2010], Zeyde et al. and ANR perform better than SRCNN, A+ , JOR and SRF, which is different from the results of the previous two tasks with general images. This observation suggests that Zeyde et al. and ANR are more generally applied than the other four state-of-the-art ISR methods. One possible reason is that models of higher complexity are more likely to overfit to the training data. The problem can be solved by re-training the model with data of similar distribution as the test data. We re-trained all the six method with unlabeled digits in SVHN, and as expected the performance is improved significantly, according to the four perceptual criteria or recognition accuracy. See Table 9.3 for the improvement. After re-training, the four methods SRCNN, A+, JOR, and SRF yield the best results. In Fig. 9.4, we show two digits, along with their super-resolved results by factor x3 and the PSNR values. From the figure, it is clear to see the artifacts generated by the ISR methods trained with general training data. The introduced artifacts lead to noisy HOG features, which in turn

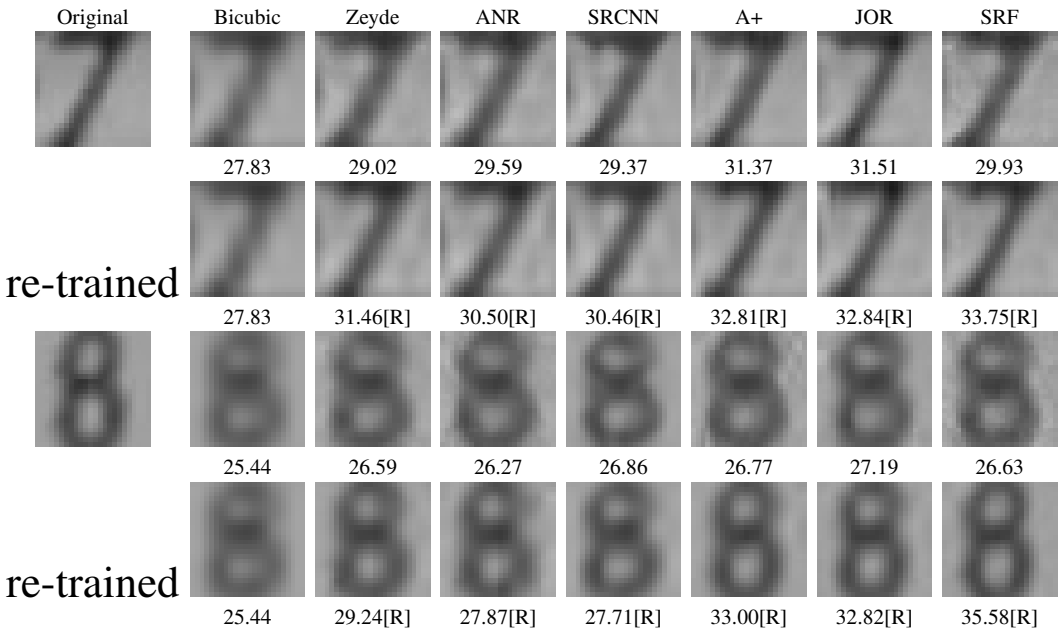


***Figure 9.4**: Super-resolved results and corresponding PSNR values of two digits. Methods marked with [R] are retrained using the unused digits of the SVHN dataset. Better seen on the screen.*

confuse the classifier. All the evidence leads to conclusions similar to that drawn for boundary detection and semantic image segmentation: (1) ISR methods are generally helpful for recognizing digits of low-resolution; and (2) perceptual evaluation criteria reflect the usefulness of ISR to digit recognition quite well. In addition, we find that the performance of ISR methods will improve significantly if they are re-trained with domain specific data.

9.3.4 Scene Recognition

In this section, we evaluated six ISR methods on the task of scene recognition. We tested the methods on the Scene-15 dataset [Lazebnik et al., 2006a], which has been widely used for image classification and clustering [Lazebnik et al., 2006a; Dai et al., 2012b; Dai & Van Gool, 2013b]. Scene-15 contains 15 scene categories in both indoor and outdoor environments. Each category has 200 to 400 images, and they are of size 300×250 pixels on average. We use the same experimental designs as for the previous tasks: down-sampling all the images by factor x3 and factor x4, and up-sampling the down-sampled images to the resolution of the original images by the six ISR methods, thus resulting in six super-resolved datasets for each scaling factor, one for bicubic interpolation, and one for the original (HR) images. As to the features, we use the Convolutional Neural Network (CNN) features [Chatfield et al., 2014], obtained from an off-the-shelf CNN model pretrained on the ImageNet. The feature is chosen as CNN feature has achieved

Scene-15		Bicubic	Zeyde	ANR	SRCNN	A+	JOR	SRF	Original
×3	PSNR	25.12	25.85	25.87	26.10	**26.19**	26.18	26.13	—
	SSIM	0.73	0.78	0.77	0.78	0.79	0.79	**0.80**	—
	IFC	2.82	3.34	3.43	3.20	3.58	**3.60**	3.58	—
	NQM	19.75	22.69	22.73	22.81	23.39	**23.43**	23.30	—
	Accuracy	0.770	0.777	0.777	0.780	**0.782**	**0.782**	0.778	0.809
×4	PSNR	24.32	24.99	24.95	25.06	**25.24**	25.22	25.19	—
	SSIM	0.674	0.701	0.702	0.704	0.720	0.719	**0.722**	—
	IFC	1.597	1.923	1.911	1.806	**2.021**	2.010	2.014	—
	NQM	14.43	16.12	16.05	16.07	**16.62**	16.61	16.57	—
	Accuracy	0.735	0.752	0.753	0.748	**0.754**	0.753	0.753	0.809

***Table 9.4**: Average PSNR, SSIM, IFC, NQM values and the accuracy of scene recognition on Scene-15 dataset.*

state-of-the-art performance for image classification [Chatfield et al., 2014]. It is worth noticing that the training and testing data are processed the same way, *i.e.* down-sampled by bicubic interpolation and up-sampled by the same ISR method (one of the six). The convolutional results at layer 16 were stacked as the CNN feature vector, with dimensionality of 4096. As to the classification, we use 15 images per class as the training samples, and the rest left for testing.

The classification accuracies over 10 random training-testing splits are averaged and reported in Table 9.4, along with the results according to the four perceptual criteria. The table shows that learning-based ISR methods are helpful for scene recognition with the deep neural network when the input images are of low-resolution. The four perceptual criteria also correlate generally well with usefulness of ISR methods for this task, which is in line with the conclusions drawn for previous vision tasks. Images at multiple scales have recently been employed for training deep neural networks [Yoo et al., 2015; Long et al., 2014], and they show improvement over a single scale. It is interesting to see how ISR methods help to generate multiple scales of the input images to train better neural networks. We leave this as our future work.

9.4 Discussion and Conclusion

We have evaluated the usefulness of image super-resolution (ISR) for a variety of different vision tasks. Six ISR methods have been employed and evaluated on four popular vision tasks. Three general conclusions can be drawn from experiments on the four tasks: 1) ISR methods are helpful in general for other vision tasks when the resolution of input images are low; 2) standard perceptual criteria, namely PSNR, SSIM, IFC, NQM, correlate quite well with the usefulness of ISR methods for the vision tasks, but they are not accurate enough to be used as full proxies; and 3) even with the state-of-the-art ISR methods, the

performance with the super-resolved images are still significantly inferior to that with the original, high-resolution images.

Although it is generally believed that ISR methods is helpful for other vision tasks, this work has formalized the common conception and conducted quantitative evaluation. We hope this work will be an inspiration for the community to integrate ISR methods into other vision systems when the input images are of low-resolution or when multiple resolutions are needed, and to evaluate ISR methods in real vision tasks, in addition to merely inspecting the visual quality. The work may inspire the community to design super-resolution algorithm for specific vision task rather than merely levering the perceptual criteria.

We acknowledge that for some tasks, the approaches and the datasets do not represent the state of the arts. However, they are standard ones and we believe they are sufficient to support the conclusions. Method evaluation on more vision tasks with more challenging datasets, testing multiple approaches for the same task, and testing different parameter settings for the same approach constitute our future work. The code and data of this work are available at www.vision.ee.ethz.ch/~daid/SR4VisionTask.

10

Conclusion

wawawa

Bibliography

(????a). Cmu sphinx. http://cmusphinx.sourceforge.net/.

(????b). Cmu sphinx. http://www.speech.cs.cmu.edu/tools/lmtool-new.html.

(2015). Smile - smart photo annotation. https://play.google.com/store/apps/details?id=com.neuromorphic.retinet.smile.

Agrawal, P., Carreira, J., & Malik, J. (2015). Learning to see by moving. In *International Conference on Computer Vision*, (pp. 37–45).

Aodha, O. M., Humayun, A., Pollefeys, M., & Brostow, G. J. (2013). Learning a confidence measure for optical flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(5), 1107–1120.

Arbelaez, P., Maire, M., Fowlkes, C., & Malik, J. (2011). Contour detection and hierarchical image segmentation. *IEEE TPAMI*, *33*(5), 898–916.

Arbelaez, P., Pont-Tuset, J., Barron, J., Marques, F., & Malik, J. (2014a). Multiscale combinatorial grouping. In *CVPR*.

Arbelaez, P., Pont-Tuset, J., Barron, J., Marques, F., & Malik, J. (2014b). Multiscale combinatorial grouping. In *CVPR*, (pp. 328–335). IEEE.

Baghshah, M. S., & Shouraki, S. B. (2009). Semi-supervised metric learning using pairwise constraints. In *IJCAI*.

Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M., & Szeliski, R. (2011). A database and evaluation methodology for optical flow. *IJCV*, *92*(1), 1–31.

Bearman, A., Russakovsky, O., Ferrari, V., & Fei-Fei, L. (2015). What's the point: Semantic segmentation with point supervision. *arXiv:1506.02106*.

Belhumeur, P. N., Hespanha, J. a. P., & Kriegman, D. J. (1997). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *TPAMI*, *19*(7), 711–720.

Belkin, M., & Niyogi, P. (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*.

Belkin, M., Niyogi, P., & Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR*, *7*(36), 2399–2434.

Bell, S., Upchurch, P., Snavely, N., & Bala, K. (2013). Opensurfaces: A richly annotated catalog of surface appearance. In *SIGGRAPH*.

Bell, S., Zitnick, C. L., Bala, K., & Girshick, R. (2015). Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. *arXiv preprint arXiv:1512.04143*.

Bennett, K. P., & Demiriz, A. (1998). Semi-supervised support vector machines. In *Advances in Neural Information Processing Systems*, (pp. 368–374).

Berg, T. L., Berg, A. C., Edwards, J., Maire, M., White, R., Teh, Y.-W., Learned-Miller, E., & Forsyth, D. A. (2004). Names and faces in the news. In *CVPR*.

Bilen, H., & Vedaldi, A. (2016). Weakly supervised deep detection networks. In *CVPR*.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, (pp. 92–100).

Boiman, O., & Irani, M. (2007). Detecting irregularities in images and in video. *IJCV*, *74*(1), 17–31.

Bosch, A., Zisserman, A., & Muoz, X. (2007a). Image classification using random forests and ferns. In *ICCV*.

Bosch, A., Zisserman, A., & Muoz, X. (2007b). Image classification using random forests and ferns. In *ICCV*.

Boykov, Y., Veksler, O., & Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *PAMI*, *23*(11), 1222–1239.

Branson, S., Wah, C., Schroff, F., Babenko, B., Welinder, P., Perona, P., & Belongie, S. (2010). Visual recognition with humans in the loop. In *ECCV*, (pp. 438–451).

Breiman, L. (2001). Random forest. *MACH LEARN*, *45*(1), 5–32.

Brostow, G. J., Fauqueur, J., & Cipolla, R. (2009). Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, *30*(2), 88–97.

Bucilua, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, (pp. 535–541). ACM.

Carreira, J., & Sminchisescu, C. (2010). Constrained parametric min-cuts for automatic object segmentation. In *CVPR*, (pp. 3241–3248). IEEE.

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, *2*, 1–27.

Chang, H., Yeung, D.-Y., & Xiong, Y. (2004). Super-resolution through neighbor embedding. *CVPR*, *01*, 275–282.

Chapelle, O., Schölkopf, B., & Zien, A. (Eds.) (2006). *Semi-Supervised Learning*. Cambridge, MA: MIT Press.

Chatfield, K., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*.

Chen, J., Liu, X., & Lyu, S. (2012). Boosting with side information. In *ACCV*.

Chen, L., Li, W., & Xu, D. (2014). Recognizing RGB images by learning from RGB-D data. In *CVPR*, (pp. 1418–1425).

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2015a). Semantic image segmentation with deep convolutional nets and fully connected crfs.

Chen, L.-C., Yang, Y., Wang, J., Xu, W., & Yuille, A. L. (2015b). Attention to scale: Scale-aware semantic image segmentation. *arXiv preprint arXiv:1511.03339*.

Chen, W.-Y., Song, Y., Bai, H., Lin, C.-J., & Chang, E. (2011). Parallel spectral clustering in distributed systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *33*(3), 568–586.

Chen, X., & Gupta, A. (2015a). Webly supervised learning of convolutional networks. In *ICCV*.

Chen, X., & Gupta, A. (2015b). Webly supervised learning of convolutional networks. In *ICCV*.

Chen, X., Shrivastava, A., & Gupta, A. (2013). Neil: Extracting visual knowledge from web data. In *IEEE International Conference on Computer Vision*, (pp. 1409–1416).

Chen, Y., Dai, D., Pont-Tuset, J., & Van Gool, L. (2016). Scale-aware alignment of hierarchical image segmentation. In *Computer Vision and Pattern Recognition (CVPR)*.

Cheng, M.-M., Zheng, S., Lin, W.-Y., Vineet, V., Sturgess, P., Crook, N., Mitra, N. J., & Torr, P. (2014). Imagespirit: Verbal guided image parsing. *ACM Trans. Graph.*, *34*(1), 3:1–3:11.

Chua, T.-S., Tang, J., Hong, R., Li, H., Luo, Z., & Zheng, Y. (2009). NUS-WIDE: a real-world web image database from National University of Singapore. In *CIVR*.

Coates, A., Ng, A. Y., & Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, (pp. 215–223).

Cohen, M. F., Shade, J., Hiller, S., & Deussen, O. (2003). Wang tiles for image and texture generation. In *SIGGRAPH*.

Collins, B., Deng, J., Li, K., & Fei-Fei, L. (2008). Towards scalable dataset construction: An active learning approach. In *ECCV*.

Comaniciu, D., & Meer, P. (2002a). Mean shift: a robust approach toward feature space analysis. *IEEE TPAMI*, *24*(5), 603 –619.

Comaniciu, D., & Meer, P. (2002b). Mean shift: A robust approach toward feature space analysis. *IEEE TPAMI*, *24*(5), 603–619.

Cordts, M., Omra, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., & Schiele, B. (2016). Cityscapes dataset for semantic urban scene understanding. In *CVPR*.

Cormen, T. H. (2009). *Introduction to algorithms*. MIT press.

Cox, M. A. A., & Cox, T. F. (2008). Multidimensional scaling. In *Handbook of Data Visualization*, (pp. 315–347).

Cui, Z., Chang, H., Shan, S., Zhong, B., & Chen, X. (2014). Deep network cascade for image super-resolution. In *ECCV*.

Dai, D., Kroeger, T., Li, W., & Van Gool, L. (2016a). Draw&tell: Efficient annotation for semantic image segmentation by drawing and speaking. In *in submission to ECCV*.

Dai, D., Kroeger, T., Timofte, R., & Van Gool, L. (2015a). Metric imitation by manifold transfer for efficient vision applications. In *IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 3527–3536).

Dai, D., Kroeger, T., Timofte, R., & Van Gool, L. (2015b). Metric imitation by manifold transfer for efficient vision applications. In *CVPR*.

Dai, D., Kroeger, T., Timofte, R., & Van Gool, L. (2015c). Metric imitation by manifold transfer for efficient vision applications. In *CVPR*.

Dai, D., Prasad, M., Leistner, C., & Gool, L. V. (2012a). Ensemble partitioning for unsupervised image categorization. In *ECCV*.

Dai, D., Prasad, M., Leistner, C., & Gool, L. V. (2012b). Ensemble partitioning for unsupervised image categorization. In *ECCV*.

Dai, D., Prasad, M., Leistner, C., & Van Gool, L. (2012c). Ensemble partitioning for unsupervised image categorization. In *European Conference on Computer Vision*, (pp. 483–496).

Dai, D., Riemenschneider, H., Schmitt, G., & Van Gool, L. (2013). Example-based facade texture synthesis. In *ICCV*.

Dai, D., Riemenschneider, H., & Van Gool, L. (2014). The synthesizability of texture examples. In *CVPR*.

Dai, D., Timofte, R., & Van Gool, L. (2015d). Jointly optimized regressors for image super-resolution. In *Eurographics*.

Dai, D., & Van Gool, L. (2013a). Ensemble projection for semi-supervised image classification. In *International Conference on Computer Vision*, (pp. 2072–2079).

Dai, D., & Van Gool, L. (2013b). Ensemble projection for semi-supervised image classification. In *ICCV*.

Dai, D., Wang, Y., Chen, Y., & Van Gool, L. (2016b). Is image super-resolution helpful for other vision tasks? In *WACV*.

Dai, D., Wu, T., & Zhu, S. C. (2010a). Discovering scene categories by information projection and cluster sampling. In *IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 483–496).

Dai, D., Wu, T., & Zhu, S. C. (2010b). Discovering scene categories by information projection and cluster sampling. In *CVPR*.

Dai, D., & Yang, W. (2011). Satellite image classification via two-layer sparse coding with biased image representation. *IEEE Geosci. Remote Sensing Lett.*, *8*(1), 173–176.

Dai, J., He, K., & Sun, J. (2015e). Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *ICCV*.

Dai, S., Han, M., Xu, W., Wu, Y., & Gong, Y. (2007). Soft edge smoothness prior for alpha channel super resolution. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, (pp. 1–8).

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *1*, 886–893.

Dana, K. J., van Ginneken, B., Nayar, S. K., & Koenderink, J. J. (1999). Reflectance and texture of real-world surfaces. *ACM Trans. Graph.*, *18*(1), 1–34.

Davis, J. V., Kulis, B., Jain, P., Sra, S., & Dhillon, I. S. (2007). Information-theoretic metric learning. In *ICML*.

De Bonet, J. S. (1997). Multiresolution sampling procedure for analysis and synthesis of texture images. In *SIGGRAPH*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *CVPR*.

Deng, J., Russakovsky, O., Krause, J., Bernstein, M. S., Berg, A., & Fei-Fei, L. (2014). Scalable multi-label annotation.

Ding, J., Shao, M., & Fu, Y. (2014). Latent low-rank transfer subspace learning for missing modality recognition. In *AAAI*.

Divvala, S., Farhadi, A., & Guestrin, C. (2014). Learning everything about anything: Webly-supervised visual concept learning. In *CVPR*.

Doersch, C., Gupta, A., & Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *International Conference on Computer Vision*, (pp. 1422–1430).

Dollár, P., & Zitnick, C. L. (2013). Structured forests for fast edge detection. In *ICCV*.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2014a). Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2014b). Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning*.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2014c). DeCAF: A deep convolutional activation feature for generic visual recognition. In *ICML*.

Dong, C., Loy, C. C., He, K., & Tang, X. (2014). Learning a deep convolutional network for image super-resolution. In *ECCV*.

Donoho, D. L., & Grimes, C. (2003). Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proc Nat Acad Sci USA*, *100*(10), 5591–5596.

Dosovitskiy, A., T. Springenberg, J., Riedmiller, M., & Brox, T. (2014). Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems*, (pp. 766–774).

Duchon, C. E. (1979). Lanczos Filtering in One and Two Dimensions. *J. Appl. Meteorology*, *18*, 1016–1022.

Dueck, D., & Frey, B. J. (2007). Non-metric affinity propagation for unsupervised image categorization.

Dutt Jain, S., & Grauman, K. (2013). Predicting sufficient annotation strength for interactive foreground segmentation. In *ICCV*.

Ebert, S., Fritz, M., & Schiele, B. (2012). Semi-supervised learning on a budget: Scaling up to large datasets. In *ACCV*.

Ebert, S., Larlus, D., & Schiele, B. (2010). Extracting structures in image collections for object recognition. In *European Conference on Computer Vision*, (pp. 720–733).

Efros, A. A., & Freeman, W. T. (2001). Image quilting for texture synthesis and transfer. In *SIGGRAPH*.

Efros, A. A., & Leung, T. K. (1999). Texture synthesis by non-parametric sampling. In *ICCV*.

Endres, I., & Hoiem, D. (2014a). Category-independent object proposals with diverse ranking. *IEEE TPAMI*, *36*(2), 222–234.

Endres, I., & Hoiem, D. (2014b). Category-independent object proposals with diverse ranking. *IEEE TPAMI*, *36*(2), 222–234.

Everingham, M., Van Gool, L., Williams, C., Winn, J., & Zisserman, A. (2010a). The pascal visual object classes (voc) challenge. *IJCV*, *88*(2), 303–338.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010b). The pascal visual object classes (voc) challenge. *IJCV*, *88*(2), 303–338.

Faktor, A., & Irani, M. (2012). "clustering by composition" - unsupervised discovery of image categories. In *ECCV*.

Fan, R.-E., Chang, K. W., Hsieh, C. J., Wang, X. R., & Lin, C. J. (2008). Liblinear: A library for large linear classification. *JMLR*, *9*(6/1/2008), 1871–1874.
URL http://portal.acm.org/citation.cfm?id=1442794

Fan, R.-E., Chen, P.-H., & Lin., C.-J. (2005). Working set selection using second order information for training SVM. *JMLR*, *6*, 1889–1918.

Farhadi, A., Endres, I., Hoiem, D., & Forsyth, D. (2009). Describing objects by their attributes. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Fei-Fei, L., Fergus, R., & Perona, P. (2004). Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. In *Workshop on Generative-Model Based Vision*.

Felzenszwalb, P. F., Girshick, R. B., & McAllester, D. (2010). Cascade object detection with deformable part models. In *CVPR*.

Felzenszwalb, P. F., & Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *IJCV*, *59*(2), 167–181.

Fergus, R., Perona, P., & Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. vol. 2, (pp. 264–271).

Fergus, R., Weiss, Y., & Torralba, A. (2009). Semi-supervised learning in gigantic image collections. In *NIPS*.

Fernando, B., Habrard, A., Sebban, M., & Tuytelaars, T. (2013). Unsupervised visual domain adaptation using subspace alignment. In *ICCV*.

Feyereisl, J., Kwak, S., Son, J., & Han, B. (2015). Object localization based on structural SVM using privileged information. In *NIPS*.

Fouad, S., Tino, P., Raychaudhury, S., & Schneider, P. (2013). Incorporating privileged information through metric learning. *T-NNLS*, *24*(7), 1086–1098.

Freedman, G., & Fattal, R. (2011). Image and video upscaling from local self-examples. *ACM Trans. Graph.*, *30*(2), 12:1–12:11.
URL http://doi.acm.org/10.1145/1944846.1944852

Freeman, W. T., Jones, T. R., & Pasztor, E. C. (2002). Example-based super-resolution. *IEEE Computer Graphics and Applications*, *22*(2), 56–65.

Frey, B. J., & Dueck, D. (2007). Clustering by passing messages between data points. *Science*, *315*(5814), 972–976.
URL http://www.ncbi.nlm.nih.gov/pubmed/17218491

Freytag, A., Rodner, E., & Denzler, J. (2014). Selecting influential examples: Active learning with expected model output changes. In *ECCV*.

Frome, A., Singer, Y., Sha, F., & Malik, J. (2007). Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *ICCV*.

Galerne, B., Gousseau, Y., & Morel, J. M. (2011). Random phase textures: Theory and synthesis. *IEEE Trans. Image Process*, *20*(1), 257–267.

Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*.

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

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014b). Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Glasner, D., Bagon, S., & Irani, M. (2009). Super-resolution from a single image. In *ICCV*.

Globerson, A., & Roweis, S. T. (2005). Metric learning by collapsing classes. In *NIPS*.

Goldberger, J., Roweis, S., Hinton, G., & Salakhutdinov, R. (2004). Neighbourhood components analysis. In *NIPS*.

Gong, Y., Kumar, S., Rowley, H. A., & Lazebnik, S. (2013). Learning binary codes for high-dimensional data using bilinear projections. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gong, Y., & Lazebnik, S. (2011). Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*.

Gool, L. V., Dewaele, P., & Oosterlinck, A. (1985). Texture analysis anno 1983. *COMPUT VISION GRAPH*, *29*(3), 336 – 357.

Gopalan, R., Li, R., & Chellappa, R. (2011). Domain adaptation for object recognition: An unsupervised approach. In *ICCV*.

Gould, S., Zhao, J., He, X., & Zhang, Y. (2014). Superpixel graph label transfer with learned distance metric. In *ECCV 2014*.

Grauman, K., & Darrell, T. (2006). Unsupervised learning of categories from sets of partially matching image features.

Graves, A., & Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In *ICML*.

Gregor, K., Danihelka, I., Graves, A., & Wierstra, D. (2015). Draw: A recurrent neural network for image generation. In *ICML*.

Guadarrama, S., Krishnamoorthy, N., Malkarnenkar, G., Venugopalan, S., Mooney, R., Darrell, T., & Saenko, K. (2013). Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *ICCV*.

Guillaumin, M., Mensink, T., Verbeek, J., & Schmid, C. (2009a). Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *ICCV*.

Guillaumin, M., Verbeek, J., & Schmid, C. (2009b). Is that you? metric learning approaches for face identification. In *ICCV*.

Guillaumin, M., Verbeek, J. J., & Schmid, C. (2010). Multimodal semi-supervised learning for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 902–909).

Gulshan, V., Rother, C., Criminisi, A., Blake, A., & Zisserman, A. (2010). Geodesic star convexity for interactive image segmentation. In *CVPR*.

Gupta, A., & Davis, L. S. (2008). Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers. In *ECCV*, (pp. 16–29). Springer.

Gupta, S., Hoffman, J., & Malik, J. (2015). Cross modal distillation for supervision transfer. *arXiv:1507.00448*.

Gygli, M., Grabner, H., Riemenschneider, H., Nater, F., & Van Gool, L. (2013). The interestingness of images. *ICCV*.

Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., & Malik, J. (2011). Semantic contours from inverse detectors. In *ICCV*.

Hartmann, W., Havlena, M., & Schindler, K. (2014). Predicting matchability. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 9–16).

Hazen, T. J., Sherry, B., & Adler, M. (2007). Speech-based annotation and retrieval of digital photographs. In *INTERSPEECH*.

He, K., & Sun, J. (2012). Statistics of patch offsets for image completion. In *ECCV*.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

He, X., Cai, D., Yan, S., & Zhang, H.-J. (2005). Neighborhood preserving embedding. In *ICCV*.

Heeger, D. J., & Bergen, J. R. (1995). Pyramid-based texture analysis/synthesis. In *SIGGRAPH*.

Hennings-Yeomans, P., Baker, S., & Kumar, B. V. (2008). Simultaneous super-resolution and feature extraction for recognition of low resolution faces. In *CVPR*.

Hinton, G., Vinyals, O., & Dean, J. (2014). Distilling the knowledge in a neural network. In *Deep Learning and Representation Learning Workshop, NIPS*.

Hoffman, J., Guadarrama, S., Tzeng, E., Hu, R., Donahue, J., Girshick, R., Darrell, T., & Saenko, K. (2014). LSDA: Large scale detection through adaptation. In *NIPS*.

Hoiem, D., Efros, A. A., & Hebert, M. (2005). Geometric context from a single image. In *ICCV*, vol. 1, (pp. 654–661). IEEE.

Hoiem, D., Efros, A. A., & Hebert, M. (2008). Putting objects in perspective. *International Journal of Computer Vision*, *80*(1), 3–15.

Hoiem, D., Efros, A. A., & Hebert, M. (2011). Recovering occlusion boundaries from an image. *IJCV*, *91*(3), 328–346.

Hoogs, A., Rittscher, J., Stein, G., & Schmiederer, J. (2003). Video content annotation using visual analysis and a large semantic knowledgebase. In *CVPR*.

Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S. S., & Sundararajan, S. (2008). A dual coordinate descent method for large-scale linear SVM. In *ICML*.

Huang, J.-B., Singh, A., & Ahuja, N. (2015). Single image super-resolution from transformed self-exemplars. In *CVPR*.

Isola, P., Xiao, J., Torralba, A., & Oliva, A. (2011). What makes an image memorable? In *CVPR*.

Isola, P., Zoran, D., Krishnan, D., & Adelson, E. H. (2014). Crisp boundary detection using pointwise mutual information. In *ECCV*, (pp. 799–814). Springer.

Jain, M., van Gemert, J. C., & Snoek, C. G. (2015). What do 15,000 object categories tell us about classifying and localizing actions? In *Computer Vision and Pattern Recognition (CVPR)*, (pp. 46–55).

Jain, P., & Kapoor, A. (2009). Active learning for large multi-class problems. In *IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 762–769).

Jegou, H., Douze, M., & Schmid, C. (2008). Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*.

Ji, Y., Sun, S., & Lu, Y. (2012). Multitask multiclass privileged information support vector machines. In *ICPR*.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., & Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *ACM International Conference on Multimedia*.

Jing, X.-Y., Zhu, X., Wu, F., You, X., Liu, Q., Yue, D., Hu, R., & Xu, B. (2015). Super-resolution person re-identification with semi-coupled low-rank discriminant dictionary learning. In *CVPR*.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning*, (pp. 200–209).

Joshi, A. J., Porikli, F., & Papanikolopoulos, N. (2009a). Multi-class active learning for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, (pp. 2372–2379). IEEE.

Joshi, A. J., Porikli, F., & Papanikolopoulos, N. (2009b). Multi-class active learning for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 2372–2379).

Kalashnikov, D., Mehrotra, S., Xu, J., & Venkatasubramanian, N. (2011). A semantics-based approach for speech annotation of images. *IEEE Trans. Knowl. Data Eng.*, *23*(9), 1373–1387.

Karpathy, A., & Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *CVPR*.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *CVPR*.

Kendall, A., Badrinarayanan, V., & Cipolla, R. (2015). Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*.

Khoreva, A., Benenson, R., Omran, M., Hein, M., & Schiele, B. (2016). Weakly supervised object boundaries. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kim, K. I., & Kwon, Y. (2010). Single-image super-resolution using sparse regression and natural image prior. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *32*(6), 1127–1133.

Kim, T. H., Lee, K. M., & Lee, S. U. (2013). Learning full pairwise affinities for spectral segmentation. *IEEE TPAMI*, *35*(7), 1690–1703.

Kingma, D. P., Mohamed, S., Rezende, D. J., & Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, (pp. 3581–3589).

Klodt, M., Sturm, J., & Cremers, D. (2013). Scale-aware object tracking with convex shape constraints on rgb-d images. In *German Conference on Pattern Recognition (GCPR)*. Saarbrücken, Germany.

Kondermann, C., Mester, R., & Garbe, C. (2008). A statistical confidence measure for optical flows. In *ECCV*.

Kopf, J., Kienzle, W., Drucker, S., & Kang, S. B. (2012). Quality prediction for image completion. *ACM Trans. Graph.*, *31*(6).

Kostinger, M., Hirzer, M., Wohlhart, P., Roth, P., & Bischof, H. (2012). Large scale metric learning from equivalence constraints. In *CVPR*.

Kozakaya, T., Ito, S., & Kubota, S. (2011). Random ensemble metrics for object recognition. In *International Conference on Computer Vision*, (pp. 1959–1966).

Krähenbühl, P., & Koltun, V. (2011). Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012a). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, (pp. 1097–1105).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012b). Imagenet classification with deep convolutional neural networks. In *NIPS*.

Kulis, B. (2012). Metric learning: A survey. *Foundations & Trends in Machine Learning*, *5*(4), 287–364.

Kulis, B., Jain, P., & Grauman, K. (2009). Fast similarity search for learned metrics. *PAMI*, *31*(12), 2143–2157.

Kulis, B., Saenko, K., & Darrell, T. (2011). What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *CVPR*.

Kulkarni, G., Premraj, V., Ordonez, V., Dhar, S., Li, S., Choi, Y., Berg, A. C., & Berg, T. L. (2013). Babytalk: Understanding and generating simple image descriptions. *PAMI*, *35*(12), 2891–2903.

Kumar Mallapragada, P., Jin, R., Jain, A., & Liu, Y. (2009). Semiboost: Boosting for semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(11), 2000–2014.

Kwatra, V., Essa, I., Bobick, A., & Kwatra, N. (2005). Texture optimization for example-based synthesis. *ACM Trans. Graph.*, *24*(3).

Kwatra, V., Schödl, A., Essa, I., Turk, G., & Bobick, A. (2003). Graphcut textures: image and video synthesis using graph cuts. In *SIGGRAPH*.

Lampert, C., Nickisch, H., & Harmeling, S. (2009). Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*.

Lampert, C. H., Nickisch, H., & Harmeling, S. (2013). Attribute-based classification for zero-shot visual object categorization. *T-PAMI*.

Lapin, M., Hein, M., & Schiele, B. (2014). Learning using privileged information: SVM+ and weighted SVM. *Neural Networks*, *53*, 95–108.

Laput, G. P., Dontcheva, M., Wilensky, G., Chang, W., Agarwala, A., Linder, J., & Adar, E. (2013). Pixeltone: a multimodal interface for image editing. In *CHI*.

Lazebnik, S., Schmid, C., & Ponce, J. (2005). A sparse texture representation using local affine regions. *PAMI*, *27*(8), 1265–1278.

Lazebnik, S., Schmid, C., & Ponce, J. (2006a). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*.

Lazebnik, S., Schmid, C., & Ponce, J. (2006b). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*.

Lefebvre, S., & Hoppe, H. (2005). Parallel controllable texture synthesis. In *SIGGRAPH*.

Leistner, C., Saffari, A., Santner, J., & Bischof, H. (2009). Semi-supervised random forests. In *International Conference on Computer Vision*, (pp. 506–513).

Lempitsky, V., Vedaldi, A., & Zisserman, A. (2011). A pylon model for semantic segmentation. In *NIPS*.

Leung, T., & Malik, J. (2001). Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, *43*(1), 29–44.

Li, J., Liang, X., Shen, S., Xu, T., & Yan, S. (2015). Scale-aware fast r-cnn for pedestrian detection. *arXiv preprint arXiv:1510.08160*.

Li, L.-J., & Fei-Fei, L. (2007a). What, where and who? classifying event by scene and object recognition. In *International Conference on Computer Vision*, (pp. 1–8).

Li, L.-J., & Fei-Fei, L. (2007b). What, where and who? classifying event by scene and object recognition.

Li, L.-J., Su, H., Xing, E. P., & Li, F.-F. (2010a). Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *Advances in Neural Information Processing Systems*, (pp. 1378–1386).

Li, L.-J., Su, H., Xing, E. P., & Li, F.-F. (2010b). Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS*.

Li, W., Dai, D., Tan, M., Xu, D., & Van Gool, L. (2016). Fast algorithms for linear and kernel svm+. In *Computer Vision and Pattern Recognition (CVPR)*.

Li, W., Niu, L., & Xu, D. (2014). Exploiting privileged information from web data for image categorization. In *ECCV*, (pp. 437–452).

Li, Y.-F., & Zhou, Z.-H. (2011). Towards making unlabeled data never hurt. In *International Conference on Machine Learning*, (pp. 175–188).

Liang, L., & Cherkassky, V. (2008). Connection between SVM+ and multi-task learning. In *IJCNN*.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014a). Microsoft coco: Common objects in context. In *ECCV*, (pp. 740–755). Springer.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollr, P., & Zitnick, C. (2014b). Microsoft coco: Common objects in context. In *ECCV*.

Liu, B., & He, X. (2015). Multiclass semantic video segmentation with object-level active inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 4286–4294).

Liu, F., & Picard, R. (1996). Periodicity, directionality, and randomness: Wold features for image modeling and retrieval. *PAMI*, *18*(7), 722–733.

Liu, L., Fieguth, P., Kuang, G., & Zha, H. (2011). Sorted random projections for robust texture classification. In *ICCV*.

Liu, W., He, J., & Chang, S.-F. (2010). Large graph construction for scalable semi-supervised learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, (pp. 679–686).

Liu, Y., Lin, W.-C., & Hays, J. (2004). Near-regular texture analysis and manipulation. In *SIGGRAPH*.

Liu, Y., Tsin, Y., & Lin, W.-C. (2005). The promise and perils of near-regular texture. *IJCV*, *62*(1-2), 145–159.

Lockerman, Y. D., Xue, S., Dorsey, J., & Rushmeier, H. (2013). Creating texture exemplars from unconstrained images. *Tech. Report*, (TR1483).

Long, J., Shelhamer, E., & Darrell, T. (2014). Fully convolutional networks for semantic segmentation. In *CVPR*.

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

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *IJCV*, *60*(2), 91–110.

Luo, Y., & Tang, X. (2008). Photo and video quality evaluation: Focusing on the subject. In *ECCV*.

Ma, C., Wei, L.-Y., & Tong, X. (2011). Discrete element textures. In *SIGGRAPH*.

Maas, A. L., Xie, Z., Jurafsky, D., & Ng, A. Y. (2015). Lexicon-free conversational speech recognition with neural networks. In *Proceedings the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Malisiewicz, T., & Efros, A. A. (2007). Improving spatial support for objects via multiple segmentations.

Manjunath, B. S., & Ma, W. (1996). Texture features for browsing and retrieval of image data. *PAMI*, *18*(8), 837–42.

Martin, D., Fowlkes, C., Tal, D., & Malik, J. (2001a). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, (pp. 416–423).

Martin, D., Fowlkes, C., Tal, D., & Malik, J. (2001b). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, (pp. 416–423).

Martinovic, A., Knopp, J., Riemenschneider, H., & Van Gool, L. (2015). 3d all the way: Semantic segmentation of urban scenes from start to end in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 4456–4465).

Matthews, T., Nixon, M. S., & Niranjan, M. (2013). Enriching texture analysis with semantic data. In *CVPR*.

McFee, B., & Lanckriet, G. R. (2010). Metric learning to rank. In *ICML*.

Mottaghi, R., Chen, X., Liu, X., Cho, N.-G., Lee, S.-W., Fidler, S., Urtasun, R., & Yuille, A. (2014). The role of context for object detection and semantic segmentation in the wild. In *CVPR*.

Nasrollahi, K., & Moeslund, T. B. (2014). Super-resolution: a comprehensive survey. *Machine Vision and Applications*, *25*(6), 1423–1468.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop*.

Ni, K., & Nguyen, T. (2007). Image superresolution using support vector regression. *Image Processing, IEEE Transactions on*, *16*(6), 1596–1610.

Nistér, D., & Stewénius, H. (2006). Scalable recognition with a vocabulary tree. In *CVPR*.

Niu, L., Li, W., & Xu, D. (2015). Exploiting privileged information from web data for action and event recognition. *IJCV*.

Ojala, T., Pietikäinen, M., & Mäenpää, T. (2002a). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(7), 971–987.

Ojala, T., Pietikainen, M., & Maenpaa, T. (2002b). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *TPAMI*, *24*(7), 971–987.

Oliva, A., & Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, *42*(3), 145–175.

Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 1717–1724).

Paget, R., & Longstaff, I. (1998). Texture synthesis via a noncausal nonparametric multiscale markov random field. *IEEE Trans. Image Processing*, *7*(6), 925–931.

Palmer, S. E. (1999). *Vision science: Photons to phenomenology*, vol. 1. MIT press Cambridge, MA.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Trans. on Knowl. and Data Eng.*, *22*(10), 1345–1359.

Papadopoulos, D., Clarke, A., Keller, F., & Ferrari, V. (2014). Training object class detectors from eye tracking data. In *ECCV*.

Papadopoulos, D. P., Uijlings, J. R. R., Keller, F., & Ferrari, V. (2016). We don't need no bounding-boxes: Training object class detectors using only human verification. In *CVPR*.

Papandreou, G., Chen, L.-C., Murphy, K., & Yuille, A. L. (2015). Weakly- and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *ICCV*.

Park, M.-G., & Yoon, K.-J. (2015). Leveraging stereo matching with learning-based confidence measures. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, (pp. 101–109).

Pathak, D., Krähenbühl, P., & Darrell, T. (2015a). Constrained convolutional neural networks for weakly supervised segmentation. In *ICCV*.

Pathak, D., Shelhamer, E., Long, J., & Darrell, T. (2015b). Fully convolutional multi-class multiple instance learning. In *ICLR*.

Paulin, M., Revaud, J., Harchaoui, Z., Perronnin, F., & Schmid, C. (2014). Transformation pursuit for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 3646–3653).

Pechyony, D., Izmailov, R., Vashist, A., & Vapnik, V. (2010). SMO-style algorithms for learning using privileged information. In *DMIN*.

Pechyony, D., & Vapnik, V. (2010). On the theory of learning with privileged information. In *NIPS*.

Peng, B., & Veksler, O. (2008). Parameter selection for graph cut based image segmentation. In *BMVC*, vol. 32, (pp. 42–44).

Pinheiro, P. H. O., & Collobert, R. (2015). From image-level to pixel-level labeling with convolutional networks. In *CVPR*.

Pitelis, N., Russell, C., & Agapito, L. (2014). Semi-supervised learning using an unsupervised atlas. In *Machine Learning and Knowledge Discovery in Databases*, vol. 8725, (pp. 565–580).

Platt, J. C. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines. Tech. rep., Advances in Kernel Methods - Support Vector Learning.

Pont-Tuset, J., & Marques, F. (2012). Supervised assessment of segmentation hierarchies. In *ECCV*, (pp. 814–827). Springer.

Portilla, J., & Simoncelli, E. P. (2000). A parametric texture model based on joint statistics of complex wavelet coefficients. *IJCV*, *40*(1), 49–70.

Prest, A., Leistner, C., Civera, J., Schmid, C., & Ferrari, V. (2012). Learning object class detectors from weakly annotated video. In *IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 3282–3289).

Qin, D., Chen, Y., Guillaumin, M., & Van Gool, L. (2014). Learning to rank bag-of-word histograms for large-scale object retrieval. In *BMVC*.

Quattoni, A., Collins, M., & Darrell, T. (2008). Transfer learning for image classification with sparse prototype representations. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Quattoni, A., & Torralba, A. (2009). Recognizing indoor scenes. In *CVPR*.

Raina, R., Battle, A., Lee, H., Packer, B., & Ng, A. Y. (2007). Self-taught learning: transfer learning from unlabeled data. In *International Conference on Machine Learning*, (pp. 759–766).

Razavian, A. S., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). Cnn features off-the-shelf: An astounding baseline for recognition. In *CVPR Workshop*.

Ren, X., & Malik, J. (2003). Learning a classification model for segmentation. In *ICCV*, (pp. 10–17). IEEE.

Ren, Z., & Shakhnarovich, G. (2013a). Image segmentation by cascaded region agglomeration. In *CVPR*.

Ren, Z., & Shakhnarovich, G. (2013b). Image segmentation by cascaded region agglomeration. In *CVPR*, (pp. 2011–2018). IEEE.

Riegler, G., Schulter, S., Rüther, M., & Bischof, H. (2015). Conditioned regression models for non-blind single image super-resolution. In *ICCV*.

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, *33*(1-2), 1–39.

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., & Bengio, Y. (2014). Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*.

Rosch, E. (1978). Principles of categorization. *Cognition and Categorization*, (pp. 27–48).

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*(5500), 2323–2326.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A., & Fei-Fei, L. (2015a). Imagenet large scale visual recognition challenge. *IJCV*, (pp. 1–42).

Russakovsky, O., Li, L.-J., & Fei-Fei, L. (2015b). Best of both worlds: human-machine collaboration for object annotation. In *CVPR*.

Russell, B. C., Freeman, W. T., Efros, A. A., Sivic, J., & Zisserman, A. (2006). Using multiple segmentations to discover objects and their extent in image collections. In *CVPR*, vol. 2, (pp. 1605–1614). IEEE.

Russell, B. C., Torralba, A., Murphy, K. P., & Freeman, W. T. (2008a). Labelme: A database and web-based tool for image annotation. *IJCV*, *77*(1-3), 157–173.

Russell, B. C., Torralba, A., Murphy, K. P., & Freeman, W. T. (2008b). Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision*, *77*(1-3), 157–173.

Rusu, A. A., Colmenarejo, S. G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., & Hadsell, R. (2016). Policy distillation. In *ICLR*.

Saini, P., & Kaur, P. (2013). Automatic speech recognition: A review. *IJETT*.

Salembier, P., & Garrido, L. (2000). Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *IEEE TIP*, *9*(4), 561–576.

Salvador, J., & Perez-Pellitero, E. (2015). Naive bayes super-resolution forest. In *ICCV*.

Schmid, C. (2001). Constructing models for content-based image retrieval. In *CVPR*.

Schulter, S., Leistner, C., & Bischof, H. (2015). Fast and accurate image upscaling with super-resolution forests. In *CVPR*.

Sharmanska, V., Quadrianto, N., & Lampert, C. (2012). Augmented attribute representations. In *European Conference on Computer Vision*, (pp. 242–255).

Sharmanska, V., Quadrianto, N., & Lampert, C. H. (2013). Learning to rank using privileged information. In *ICCV*.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE TPAMI*, *22*(8), 888–905.

Shotton, J., Winn, J., Rother, C., & Criminisi, A. (2006). Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV*.

Shotton, J., Winn, J., Rother, C., & Criminisi, A. (2009). Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *IJCV*.

Shotton, M. C. R. R. C. T. C. K., J.; Johnson (2008). Semantic texton forests for image categorization and segmentation.

Shrivastava, A., Singh, S., & Gupta, A. (2012). Constrained semi-supervised learning using attributes and comparative attributes. In *European Conference on Computer Vision*, (pp. 369–383).

Silberman, N., Hoiem, D., Kohli, P., & Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *ECCV*.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Learning local feature descriptors using convex optimisation. *TPAMI*, *36*(8), 1573–1585.

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

Singh, S., Gupta, A., & Efros, A. A. (2012). Unsupervised discovery of mid-level discriminative patches. In *European Conference on Computer Vision*, (pp. 73–86).

Sivic, J., Russell, B. C., Efros, A. A., Zisserman, A., & Freeman, W. T. (2005). Discovering objects and their location in images. In *ICCV*.

Sivic, J., Russell, B. C., Zisserman, A., Freeman, W. T., & Efros, A. A. (2008). Unsupervised discovery of visual object class hierarchies.

Srihari, R., & Zhang, Z. (2000). Show&tell: a semi-automated image annotation system. *MultiMedia, IEEE*, *7*(3), 61–71.

Srivastava, N., Mansimov, E., & Salakhudinov, R. (2015). Unsupervised learning of video representations using lstms. In *International Conference on Machine Learning*.

Srivastava, N., & Salakhutdinov, R. (2012). Multimodal learning with deep boltzmann machines. In *NIPS*.

Sun, J., & Ling, H. (2013). Scale and object aware image thumbnailing. *IJCV*, *104*(2), 135–153.

Sun, J., Xu, Z., & Shum, H.-Y. (2008). Image super-resolution using gradient profile prior. In *CVPR*.

Sun, L., & Hays, J. (2012). Super-resolution from internet-scale scene matching. In *ICCP*.

Tappen, M. F., Russell, B. C., & Freeman, W. T. (2003). Exploiting the sparse derivative prior for super-resolution and image demosaicing. In *In IEEE Workshop on Statistical and Computational Theories of Vision*.

Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*(5500), 2319–2323.

Thévenaz, P., Blu, T., & Unser, M. (2000). Image interpolation and resampling.

Thomason, J., Venugopalan, S., Guadarrama, S., Saenko, K., & Mooney, R. (2015). Integrating language and vision to generate natural language descriptions of videos in the wild. In *COLING*.

Timofte, R., De Smet, V., & Van Gool, L. (2013). Anchored neighborhood regression for fast example-based super resolution. In *ICCV*.

Timofte, R., De Smet, V., & Van Gool, L. (2014). A+: Adjusted anchored neighborhood regression for fast super-resolution. In *ACCV*.

Timofte, R., Rothe, R., & Van Gool, L. (2015). Seven ways to improve example-based single image super resolution. *arXiv:1511.02228*.

Torresani, L., Szummer, M., & Fitzgibbon, A. (2010). Efficient object category recognition using classemes. In *European Conference on Computer Vision*, (pp. 776–789).

Tschumperle, D., & Deriche, R. (2005). Vector-valued image regularization with pdes: a common framework for different applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *27*(4), 506–517.

Tuceryan, M., & Jain, A. K. (1993). Texture analysis. In *Handbook of Pattern Recognition and Computer Vision (2nd Ed.)*, (pp. 235–276).

Tuytelaars, T., Lampert, C. H., Blaschko, M. B., & Buntine, W. (2009). Unsupervised object discovery: A comparison. *IJCV*.

Tuytelaars, T., Lampert, C. H., Blaschko, M. B., & Buntine, W. (2010). Unsupervised object discovery: A comparison. *International Journal of Computer Vision*, *88*(2), 284–302.

van der Maaten, L. J., Postma, E. O., & van den Herik, H. J. (2009). Dimensionality reduction: A comparative review. *Tech. Rep., Tilburg Uni.*.

Vapnik, V., & Izmailov, R. (2015). Learning using privileged information: Similarity control and knowledge transfer. *JMLR*, *16*, 20232049.

Vapnik, V., & Vashist, A. (2009). A new learning paradigm: Learning using privileged information. *Neural Networks*, *22*(56), 544–557.

Varas, D., Alfaro, M., & Marques, F. (2015). Multiresolution hierarchy co-clustering for semantic segmentation in sequences with small variations. In *ICCV*.
URL http://arxiv.org/abs/1510.04842

Varma, M., & Zisserman, A. (2009). A statistical approach to material classification using image patch exemplars. *PAMI*, *31*(11), 2032–2047.

Vedaldi, A., & Fulkerson, B. (2008). VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/.

Vedaldi, A., & Lenc, K. (2014). Matconvnet – convolutional neural networks for matlab. *CoRR*, *abs/1412.4564*.

Verbeek, J., & Triggs, B. (2007). Region classification with markov field aspect models. In *CVPR*.

Vezhnevets, A., Buhmann, J., & Ferrari, V. (2012). Active learning for semantic segmentation with expected change. In *CVPR*.

Vijayanarasimhan, S., & Grauman, K. (2012). Active frame selection for label propagation in videos. In *ECCV*.

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. In *CVPR*.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *CVPR*.

Von Ahn, L. (2006). Games with a purpose. *Computer*, *39*(6), 92–94.

Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., & Gong, Y. (2010). Locality-constrained linear coding for image classification. In *CVPR*.

Wang, X., & Gupta, A. (2015). Unsupervised learning of visual representations using videos. In *International Conference on Computer Vision*, (pp. 2794–2802).

Wang, Z., & Ji, Q. (2015). Classifier learning with hidden information. In *CVPR*.

Wang, Z., Liu, D., Yang, J., Han, W., & Huang, T. (2015). Deeply improved sparse coding for image super-resolution. In *ICCV*.

Wei, L.-Y., Lefebvre, S., Kwatra, V., & Turk, G. (2009). State of the art in example-based texture synthesis. In *EG-STAR*.

Weinberger, K. Q., & Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.*, *10*, 207–244.

Weston, J., Ratle, F., & Collobert, R. (2008). Deep learning via semi-supervised embedding. In *International Conference on Machine Learning*, (pp. 639–655).

Weston, J., Ratle, F., Mobahi, H., & Collobert, R. (2012). Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, (pp. 639–655). Springer.

Wu, L., Hoi, S., Jin, R., Zhu, J., & Yu, N. (2012). Learning bregman distance functions for semi-supervised clustering. *IEEE Trans. Knowl. Data Eng.*, *24*(3), 478–491.

Xia, G.-S., Delon, J., & Gousseau, Y. (2010). Shape-based invariant texture indexing. *IJCV*, *88*(3), 382–403.

Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., & Torralba, A. (2010). Sun database: Large-scale scene recognition from abbey to zoo.

Xing, E. P., Ng, A. Y., Jordan, M. I., & Russell, S. (2003). Distance metric learning, with application to clustering with side-information. In *NIPS*.

Xu, C., Whitt, S., & Corso, J. J. (2013). Flattening supervoxel hierarchies by the uniform entropy slice. In *ICCV*, (pp. 2240–2247). IEEE.

Xu, J., Schwing, A. G., & Urtasun, R. (2015a). Learning to segment under various forms of weak supervision. In *CVPR*.

Xu, L., Ren, J., Yan, Q., Liao, R., & Jia, J. (2015b). Deep edge-aware filters. In *ICML*.

Xu, X., Li, W., & Xu, D. (2015c). Distance metric learning using privileged information for face verification and person re-identification. *T-NNLS*, *26*, 3150–3162.

Xu, Z., Tao, D., Zhang, Y., Wu, J., & Tsoi, A. (2014). Architectural style classification using multinomial latent logistic regression. In *European Conference on Computer Vision*, (pp. 600–615).

Yang, C.-Y., Huang, J.-B., & Yang, M.-H. (2011). Exploiting self-similarities for single frame super-resolution. In *ACCV*.

Yang, C.-Y., Ma, C., & Yang, M.-H. (2014a). Single-image super-resolution: A benchmark. In *ECCV*.

Yang, C.-Y., & Yang, M.-H. (2013). Fast direct super-resolution by simple functions. In *The IEEE International Conference on Computer Vision (ICCV)*.

Yang, J., Wright, J., Huang, T., & Ma, Y. (2010). Image super-resolution via sparse representation. *Image Processing, IEEE Transactions on*, *19*(11), 2861–2873.

Yang, M., Dai, D., Shen, L., & Van Gool, L. (2014b). Latent dictionary learning for sparse representation based classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 4138–4145).

Yang, Y., & Newsam, S. (2010). Bag-of-visual-words and spatial extensions for land-use classification. In *International Conference on Advances in Geographic Information Systems*, (pp. 270–279). ACM.

Yao, A., Gall, J., Fanelli, G., & Gool, L. V. (2011). Does human action recognition benefit from pose estimation? In *British Machine Vision Conference (BMVC)*.

Yao, A., Gall, J., Leistner, C., & Van Gool, L. (2012). Interactive object detection. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, (pp. 3242–3249). IEEE.

Yoo, D., Park, S., Lee, J.-Y., & Kweon, I. S. (2015). Multi-scale pyramid pooling for deep convolutional representation. In *CVPR Workshop*.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, (pp. 3320–3328).

Yu, F. X., Cao, L., Feris, R. S., Smith, J. R., & Chang, S.-F. (2013). Designing category-level attributes for discriminative visual recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 771–778).

Yu, Y., Fang, C., & Liao, Z. (2015). Piecewise flat embedding for image segmentation.

Zalesny, A., Ferrari, V., Caenen, G., & Van Gool, L. (2005). Composite texture synthesis. *IJCV*, *62*(1-2).

Zelnik-Manor, L., & Perona, P. (2004). Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems*, (pp. 1601–1608).

Zeyde, R., Elad, M., & Protter, M. (2012). On single image scale-up using sparse-representations. In *Curves and Surfaces*, (pp. 711–730).

Zhang, P., Wang, J., Farhadi, A., Hebert, M., & Parikh, D. (2014a). Predicting failures of vision systems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 3566–3573).

Zhang, Q., Hua, G., Liu, W., Liu, Z., & Zhang, Z. (2014b). Can visual recognition benefi from auxiliary information in training? In *ACCV*.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., & Torr, P. (2015). Conditional random fields as recurrent neural networks. In *ICCV*.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., & Schlkopf, B. (2004). Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, (pp. 321–328).

Zhu, S. C., Wu, Y., & Mumford, D. (1998). Filters, random fields and maximum entropy (frame): Towards a unified theory for texture modeling. *IJCV*, *27*(2), 107–126.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *International Conference on Machine Learning*, (pp. 912–919).

Zhu, X., & Goldberg, A. B. (2009). *Introduction to Semi-Supervised Learning*.

Zitnick, C. L., & Doll, P. (2014). Edge Boxes : Locating Object Proposals from Edges. In *ECCV*.

List of Publications

Journal Publications

1. A. Yao, J. Gall, L. Van Gool. Coupled Action Recognition and Pose Estimation from Multiple Views. *International Journal of Computer Vision (IJCV)*, 2012. 100(1), 16-37.

2. J. Gall, A. Yao, N. Razavi, L. Van Gool and V. Lempitsky. Hough Forests for Object Detection, Tracking and Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2011. 33(11), 2188-2202.

3. A.Y.J. Yao and W. Einhaeuser. Colour aids late but not early stages of rapid natural scene recognition. *Journal of Vision*, 2008. 8(16):12, 1-13.

Refereed Conference Proceedings

1. A. Yao, J. Gall, C. Leistner and L. Van Gool. Interactive Object Detection. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

2. A. Yao, J. Gall, L. Van Gool, and R. Urtasun. Learning Probabilistic Non-Linear Latent Variable Models for Tracking Complex Activities. *Neural Information Processing Systems (NIPS)*, 2011.

3. A. Yao, J. Gall, G. Fanelli and L. Van Gool. Does Human Action Recognition Benefit from Pose Estimation? In *Proceedings British Machine Vision Conference (BMVC)*, 2011.

4. A. Yao, D. Uebersax, J. Gall and L. Van Gool. Tracking People in Broadcast Sports. In *Proceedings German Association for Pattern Recognition (DAGM)*, 2010.

5. J. Gall, A. Yao and L. Van Gool. 2D Action Recognition Serves 3D Human Pose Estimation. In *Proceedings European Conference on Computer Vision (ECCV)*, 2010.

6. A. Yao, J. Gall and L. Van Gool. a Hough Transform-Based Voting Framework for Action Recognition. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

Others

1. D. Waltisberg, A. Yao, J. Gall and L. Van Gool. Variations of a Hough-Voting Action Recognition System. *Proceedings International Conference on Pattern Recognition (ICPR) Contests*, 2010.

2. G. Fanelli, A. Yao, P.L. Noel, J. Gall and L. Van Gool. Hough Forest-Based Facial Expression Recognition from Video Sequences. *International Workshop on Sign, Gesture and Activity (SGA)*, 2010.

Curriculum Vitae

Personal Data

Name	Stefan Saur
Date of birth	1^{st} December 1979
Place of birth	Buchen (Odenwald), Germany
Citizenship	German

Education

2005 – 2009	*ETH Zurich, Computer Vision Laboratory, Switzerland*
	Doctoral studies
2004	*National University of Singapore, Singapore*
	Semester abroad
2000 – 2005	*University of Karlsruhe, Germany*
	Studies of Electrical Engineering and Information Technology
	Graduation with the degree Dipl.-Ing.
1990 – 1999	*Ganztagsgymnasium Osterburken, Germany*

Work Experience

2005 – 2008	*ETH Zurich, Computer Vision Laboratory, Switzerland*
	Teaching and research assistant
2000 – 2005	*Siemens AG, Germany*
	Several internships, semester project, and master thesis
2001 – 2009	*Webdesign, self-employed*

Awards

2004 *Baden-Württemberg Stipendium*, Landesstiftung Baden-Württemberg
2000 *IPP award*, University of Karlsruhe, Germany