Superpixel Graph Label Transfer with Learned Distance Metric

Stephen Gould[1], Jiecheng Zhao[1], Xuming He[1,2], and Yuhang Zhang[1,3]

[1] Research School of Computer Science, ANU, Australia
[2] NICTA, Australia
[3] Chalmers University of Technology, Sweden

Abstract. We present a fast approximate nearest neighbor algorithm for semantic segmentation. Our algorithm builds a graph over superpixels from an annotated set of training images. Edges in the graph represent approximate nearest neighbors in feature space. At test time we match superpixels from a novel image to the training images by adding the novel image to the graph. A move-making search algorithm allows us to leverage the graph and image structure for finding matches. We then transfer labels from the training images to the image under test. To promote good matches between superpixels we propose to learn a distance metric that weights the edges in our graph. Our approach is evaluated on four standard semantic segmentation datasets and achieves results comparable with the state-of-the-art.

1 Introduction

Semantic segmentation, or multi-class pixel labeling, is a fundamental step in understanding images. In this task every pixel in the image is annotated with a category label, such as "sky", "tree", "water", "person", etc. Traditional methods learn a per-pixel classifier for each category of interest and combine these together with pairwise and higher-order constraints using a conditional Markov random field (CRF) [1, 2]. The difficulty with such methods is that they need re-training whenever new data becomes available or the set of categories of interest is changed. Furthermore, the per-pixel classifier evaluation is an expensive operation that must be performed for each test image.

The latter problem can be addressed by moving to a superpixel representation of the image. Here a bottom-up over-segmentation algorithm (e.g., [3, 4]) is used to divide the image into small contiguous regions of similar appearance. The number of superpixels is significantly smaller than the number of pixels but still capture important image characteristics (such as object boundaries). Thus inference time is dramatically reduced with negligible cost to accuracy.

Recently, nearest neighbor methods have been proposed to address the problem of growing datasets and changing categories [5–9]. These data driven approaches work by first matching regions in the image under test to regions from a large database of hand labelled images. Matches are found based on region and image-level appearance cues. Since the regions in the database are labelled,

D. Fleet et al. (Eds.): ECCV 2014, Part I, LNCS 8689, pp. 632–647, 2014.

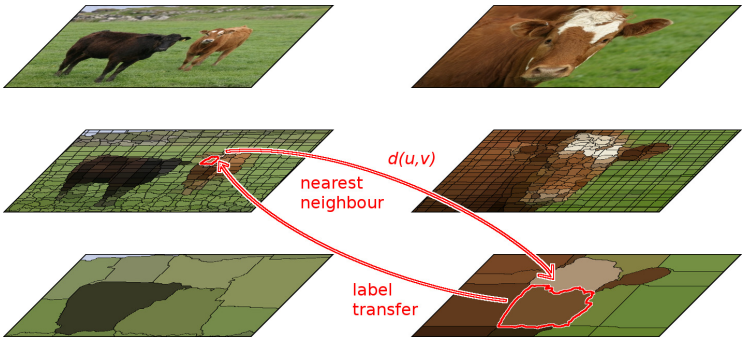


Fig. 1. Illustration of the idea of label transfer on a Superpixel Graph. The graph is built by finding good matches for each superpixel. These are represented by directed edges. Labels are then transferred backwards along the edges.

the target image can be annotated by transferring the labels of the matched regions. Clearly such methods can seamlessly integrate new data but they require a finely tuned distance function for finding matches—ideally one that correlates category labels (the thing we want) with appearance (the thing we observe).

In this paper we propose a nearest neighbor method for semantic segmentation that uses a learned distance function. Our method represents an image by a set of overlapping superpixels of different sizes. Associated with each superpixel is a feature vector that summarizes its appearance. By finding superpixels with similar features in a database of labelled images we can apply label transfer to annotate a new image. The idea is illustrated in Figure 1. Importantly, nearest neighbor matching is done per superpixel but annotation is done at the pixel level allowing pixel labels to be determined from overlapping superpixels.

We present an algorithm for rapidly finding good matches by adaptively constructing a graph where nodes represent superpixels and edges represent matches. Our algorithm can be viewed as an approximate nearest neighbor method, but one where we encourage spatial continuity during matching. This is in contrast to other methods that rely on matching at two different levels: global (image) and local (superpixel). To improve the accuracy of the annotation we learn a distance metric so that superpixels with the same label appear closer in feature space than those with different labels. Our key contribution is the integration of metric learning with graph construction. Specifically, we interleave graph construction and metric learning, and derive the metric learning problem based on the structure of the graph at hand.

We run extensive experiments on four standard datasets and compare our method with and without learning the distance metric. We also evaluate the effect of varying other algorithm parameters such as the number of superpixels and the database size. Our results show that metric learning helps to improve accuracy but as expected other factors, such as dataset size, are also important for achieving good performance.

2 Related Work

Semantic segmentation is a well studied topic in computer vision with a large number of methods proposed. Most closely related to our approach are the so-called label transfer methods. Liu et al. [9] first proposed such an approach with an innovative algorithm that uses SIFT descriptors to align scenes and then overlay labels from a subset of training images onto the target image. The subset of training images is selected by matching global image descriptors.

Zhang et al. [7] use a similar approach. However, instead of aligning scenes by matching SIFT descriptors they directly match pairs of image regions and use a Markov random field (MRF) to smooth the matches. Like our method they use a superpixel representation of the image to reduce complexity and provide spatial support for computing features. However, our method does not require matching (and hence label transfer from) an entire image. Moreover, we use multiple sets of different sized superpixels for representing a single image.

Many works attempt to explain an image by matching parts of it to regions in other images (e.g., [10, 11]). A label transfer approach along these lines, aimed at large scale labeling tasks, is the work of Tighe and Lazebnik [6]. This work also uses a superpixel representation of images but does not enforce spatial continuity during the matching process. Matches are used to construct pseudo-probability vectors for each superpixel in the test image and an MRF produces the final predictions. The approach uses a very large number of features and a pre-selection phase based on global image descriptors like the previous works to prune the images considered at test time. Nevertheless, they are able to label images in under 10s of total processing time.

An extension of this work [5] includes cues derived from per-exemplar object detectors, which have been shown to work well for the object detection task [12]. This improves performance on less abundant classes but comes at a substantial cost in features computed and processing time. Our approach, on the other hand, computes an order of magnitude fewer features and achieves similar results.

Conceptually similar to our work is the PatchMatchGraph method of Gould and Zhang [8] and the PatchWeb method of Barnes [13]. Like our approach, they build a graph over matched image regions. However, instead of using superpixels they use overlapping rectangular image patches. This makes the method expensive both in terms of memory and running time. Our superpixel representation provides a much more compact set of regions without compromising accuracy. To build the graph over image patches, Gould and Zhang [8] and Barnes [13] employ a search strategy motivated by the PatchMatch algorithm of Barnes et al. [14, 15]. We use a similar strategy adapted to superpixels, and also propose a new random projection move.

All of the above methods rely on hand tuned feature vectors for matching regions (patches or superpixels) across images. While this can give high quality results in terms of matching similar appearance there is no guarantee that the features chosen will match regions with similar semantics. Our approach differs by incorporating a learned distance metric. Specifically, we use a variant of the large margin nearest neighbor algorithm of Weinberger and Saul [16]. Eigen and

Fergus [17] also employ a nearest neighbor approach with learned metric but do so by scaling individual training set descriptors. Our metric is a generalized distance capable of arbitrary linear transforms in feature space.

3 Superpixel Graphs

In this section we describe our nearest neighbor algorithm for label transfer. We begin by describing our superpixel-based representation. Next, we introduce a superpixel graph for quickly finding similar superpixels in other images. Our approach is akin to the PatchMatchGraph [8] but rather than using rectangular patches we use superpixels. This gives us a much more compact image representation. Interleaved with our graph construction is the learning of the distance metric used to compare superpixels. At the end of the section we show how our graph can be used for label transfer.

3.1 Superpixel Embedding

We represent an image by a set of superpixels generated by an over-segmentation algorithm. However, a single over-segmentation of an image may fail to capture true object boundaries—too coarse a representation will miss small objects while too fine a representation results in many superpixels that are non-distinctive. Thus we produce many different over-segmentations resulting in an overlapping set of superpixels of different sizes. In our work, we use the superpixel algorithm of Zhang et al. [3] but our approach is not limited to this choice—any other over-segmentation algorithm can be used (e.g., [4]).

We embed our superpixels in a metric space by encoding the superpixels as feature vectors in $\mathbb{R}^n$. We denote the feature vector for superpixel u by $\boldsymbol{x}_u$. In this work we construct the feature vectors by averaging filter responses over the superpixel region. Our filters include the 17-dimensional "texton" filter [2], 13-dimensional dense HOG [18, 19], and 4-connected LBP histograms [20]. We also include the x- and y-location of the superpixel and its size to provide spatial context, and the entropy of each RGB color channel to provide a further measure of texture. To construct $\boldsymbol{x}_u$ we combine these features with the mean and standard deviation of the filter responses from neighboring regions at each of the four compass directions. This gives 510 features for each superpixel. Note that this is much smaller than the feature vectors used in other superpixel-based nearest neighbor methods (e.g., [6]).[1]

During metric learning and image annotation we require superpixel labels $y_u \in \{1, \ldots, C\}$. These are generated from ground truth pixel labels by computing the proportion of each class label within the superpixel. This gives an empirical probability estimate $\hat{P}_u(y)$. Most superpixels only contain pixels with the same label but some larger superpixels can have mixed labels. When a single hard label for superpixel u is required we take it as $y_u = \operatorname{argmax}_y \hat{P}_u(y)$.

[1] We provide full source code for our method including superpixel generation, feature calculation, graph construction, metric learning, and label transfer as part of the Darwin software package [21].

3.2 Building the Superpixel Graph

For finding nearest neighbor matches quickly we construct a graph over superpixels. Our graph is similar to the PatchMatchGraph proposed by Gould and Zhang [8], which uses ideas from Barnes et al. [15] for leveraging existing matches to find better ones. Formally, let $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ be a graph with nodes $\mathcal{V}$ and edges $\mathcal{E}$. Each node $u \in \mathcal{V}$ represents a superpixel and each directed edge $(u, v) \in \mathcal{E}$ represents a match from superpixel u to superpixel v. Note that we do not require superpixel v to also match with u.

A weight associated with each edge (u, v) represents the cost of matching u to v. We define this cost to be the (generalized) distance between the feature vectors describing the two superpixels:

$$d_M(u, v) = (\boldsymbol{x}_u - \boldsymbol{x}_v)^T M (\boldsymbol{x}_u - \boldsymbol{x}_v) \tag{1}$$

where $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$ are the feature vectors associated with superpixels u and v, respectively, and $M = LL^T \succeq 0$ is a positive semi-definite matrix that parameterizes the metric. When $M = I$ the metric is the Euclidean norm, and when $M = \hat{\Sigma}^{-1}$ the metric is the Mahalanobis distance, where $\hat{\Sigma}^{-1}$ is the inverse covariance matrix of the data.

The set of nodes adjacent to node u in the graph are $\mathcal{N}_u = \{v : (u, v) \in \mathcal{E}\}$. If the corresponding superpixels are those closest, in feature space, to u out of all $v \in \mathcal{V}$ then $\mathcal{N}_u$ is the set of exact nearest neighbors. Thus to find the set of k nearest neighbors for all superpixels u we need to solve the following optimization problem over the set of edges in the graph.

$$\begin{aligned} &\text{minimize}_{\mathcal{E}} \textstyle\sum_{(u,v)\in\mathcal{E}} d_M(u, v) \\ &\text{subject to } \forall u \in \mathcal{V} \colon \deg(u) = k \\ &\qquad\qquad \forall (u, v) \in \mathcal{E} \colon \text{img}(u) \neq \text{img}(v) \\ &\qquad\qquad \forall (u, v), (u, w) \in \mathcal{E} \colon \text{img}(v) \neq \text{img}(w) \end{aligned} \tag{2}$$

where we constrain the out degree of each node to k and denote such a graph by $\mathcal{G}_k$. Furthermore, since we are interested in using our graph for label transfer there is no use in matching a superpixel to another one within the same image. Thus we add the constraint $\text{img}(u) \neq \text{img}(v)$, meaning superpixels u and v cannot come from the same image. Finally, as in Gould and Zhang [8] we would like to find a diverse set of matches and so restrict each superpixel to matching at most one superpixel from any single image. Consequently we are guaranteed that each superpixel matches to superpixels from k different images. This is encoded by the last constraint.

Equation 2 is a hard optimization problem since we are minimizing over the discrete set of edges in the graph. Therefore, we perform approximate optimization via a move-making algorithm that we describe below. Since finding nearest neighbors requires many distance computations we can accelerate the search by first transforming the features as $\boldsymbol{x}' = L^T \boldsymbol{x}$ so that $d_M(u, v) = \|\boldsymbol{x}'_u - \boldsymbol{x}'_v\|^2$.

Our overall algorithm for building a superpixel graph as described above is summarized in Algorithm 1.

Algorithm 1. Build Superpixel Graph.

input training images (and labels) and k
generate superpixels $\mathcal{V} = \{u\}$
compute superpixel features $\{\boldsymbol{x}_u\}$ and labels $\{y_u\}$
define feature transform matrix

$$L = \begin{cases} I & \text{euclidean} \\ \mathbf{diag}\,(1/\sigma_i) & \text{whitened} \\ \Sigma^{-1/2} & \text{mahalanobis} \\ \text{LMNN}_k(\{\boldsymbol{x}_u\}, \{y_u\}) & \text{learned (see §3.3)} \end{cases}$$

transform features $\forall u \in \mathcal{V}$, $\boldsymbol{x}'_u \leftarrow L^T \boldsymbol{x}_u$
initialize superpixel graph $\mathcal{G}_k = \langle \mathcal{V}, \mathcal{E} \rangle$ on features $\boldsymbol{x}'$
repeat
    attempt search moves for each $u \in \mathcal{V}$
until convergence
return graph $\mathcal{G}_k$ and feature transform matrix L^T

Move Making Optimization. Our search moves are motivated by the PatchMatch algorithm of Barnes et al. [15, 14]. We begin by initializing the graph with random edges that honor the constraints of Equation 2, i.e., each node has exactly k outgoing edges and no two edges emanating from a node terminate at nodes belonging to the same image. We then iterate over a sequence of search moves to incrementally improve the objective. We terminate after a fixed number of iterations or when the objective value (i.e., graph structure) does not change after attempting all moves.

Briefly, each move evaluates a set of candidate matches for one or more superpixels. These manifest as changes to the graph structure. Consider a candidate match v for superpixel u. To decide whether or not to add the edge (u, v) to the graph we examine the current outgoing edges from u. If one of these edges, say (u, w), points to a superpixel from the same image as v and if $d_M(u, v)$ is smaller than $d_M(u, w)$ we replace (u, w) with (u, v). Now, if none of the current outgoing edges from u point to a superpixel from the same image as v then we replace the highest cost outgoing edge with (u, v) if it has a smaller cost. This local update rule ensures that the objective of Equation 2 is non-increasing and the constraints are always satisfied.

An key aspect of our moves is that they are context enriched and encourage spatially coherent matches. That is, they are not simply based on local superpixel feature similarity. Nevertheless the moves can be computed in parallel for each superpixel u. This is important for scaling to larger image datasets as we progress towards solving the scene understanding challenge.

Exhaustive Search. The exhaustive search move is a naive move that finds the best k matches for a given superpixel u by comparing it to all superpixels in all other images. This is computationally expensive so we only apply it to a small number of superpixels per iteration. We randomly choose the superpixels from a distribution weighted by the cost of the current matches for each superpixels—that is, we are more likely to sample a superpixel that currently has bad matches

than a superpixel that already has good ones. Our hope is that this will seed the graph with some very good edges that other search moves can exploit.

Random Projection. Motivated by locality-sensitive hashing (LSH) [22], the random projection move uses the fact that it is much easier to find nearest neighbors in a one-dimensional space than an n-dimensional space. Moreover, points close to each other in n dimensions will remain close to each other when projected onto a line. Of course, points that are distant may also end up close on the line. Nevertheless, the move is effective for finding some good matches especially during early move making iterations.

Concretely, we choose a random direction $\boldsymbol{z}$ in the n-dimensional feature space. We then project the feature vector for each superpixel onto this direction giving $\boldsymbol{z}^T \boldsymbol{x}_u \in \mathbb{R}$. Sorting by these values we can easily find nearby superpixels in direction $\boldsymbol{z}$ (not from the same image). We compute the distance between the superpixels in the full n-dimensional space with those within a fixed horizon h along direction $\boldsymbol{z}$ and perform the update rule described above.

Local Search and Propagate. Local search and propagate moves exploit image smoothness. For local search we consider candidate matches from the image neighborhood of the current match—that is, for edge $(u, v) \in \mathcal{E}$ we consider all superpixels adjacent to v in the image. For the propagation move we consider candidate matches where both superpixels are image-neighbors of a current match—that is, given an edge $(u, v) \in \mathcal{E}$ we consider the match (u', v') where u' and v' are superpixels adjacent to u and v in their respective images.

Enrichment. The last moves are the forward and inverse enrichment moves, which leverage properties of our graph $\mathcal{G}_k$. The forward enrichment move takes pairs of edges (u, v) and (v, w), and considers adding the candidate edge (u, w) to the graph. The inverse enrichment move takes an edge (u, v) and consider adding the reverse edge (v, u) to the graph. Both the forward and inverse enrichment moves tend to rapidly spread good matches across the graph.

3.3 Distance Metric Learning

As discussed above, it is important that our distance metric puts semantically similar superpixels closer than semantically different ones. However, the distance metric only has access to the observed superpixel features not their labels. There is no a priori guarantee that superpixel with similar features have the same category label since this is highly dependent on the features chosen and their relative scaling. Thus we need to learn a metric with the desired property that superpixels of the same label are clustered together. We do so using a variant of the large margin nearest neighbor (LMNN) algorithm of Weinberger and Saul [16].

Formally, let $\mathcal{N}_u^+ \subseteq \{v \in \mathcal{V} : y_v = y_u\}$ with $|\mathcal{N}_u^+| = k$ be the set of target superpixels within the neighborhood set of u, and let $\mathcal{N}_u^- = \{w \in \mathcal{V} : y_w \neq y_u\}$ be the set of so-called imposter superpixels with label differing from that of u. We wish to learn a metric so that all $v \in \mathcal{N}_u^+$ are closer to u than any $w \in \mathcal{N}_u^-$. The

large margin nearest neighbor algorithm aims to find such a metric by solving the following convex optimization problem

$$\begin{array}{ll} \text{minimize}_M & \sum_{uv} d_M(u,v) + C \sum_{uvw} \xi_{uvw} \\ \text{subject to} & \forall uvw: \; d_M(u,w) - d_M(u,v) \geq 1 - \xi_{uvw} \\ & \xi_{uvw} \geq 0 \\ & M \succeq 0 \end{array} \tag{3}$$

where uv iterates over all $u \in \mathcal{V}$ and $v \in \mathcal{N}_u^+$, and uvw iterates over all $u \in \mathcal{V}$, $v \in \mathcal{N}_u^+$ and $w \in \mathcal{N}_u^-$. Here $C > 0$ trades off regularization of M with the margin constraint. The problem is a positive semi-definite program which we solve by the subgradient method on L (see Appendix A).

Our method differs from the implementation of Weinberger and Saul [16] in that the target and imposter nearest neighbors are chosen to satisfy the constraints of Equation 2. This is important because we wish to rule out adjacent superpixels in the same image as target nearest neighbors since these are unhelpful for label transfer. Given an annotated training set and initial distance metric M we run our graph construction algorithm using the following two label-augmented distance functions

$$d_M^+(u,v) = \begin{cases} d_M(u,v) & \text{if } y_u = y_v \\ \infty & \text{otherwise} \end{cases} \quad \text{and} \quad d_M^-(u,v) = \begin{cases} d_M(u,v) & \text{if } y_u \neq y_v \\ \infty & \text{otherwise} \end{cases} \tag{4}$$

to find $\mathcal{N}_u^+$ and $\mathcal{N}_u^-$, respectively. Specifically, the edges in the graph constructed using the first metric contain only target nearest neighbors and the edges in the graph constructed using the second metric contain only imposter nearest neighbors. We then iterate between learning the metric and refining the graph edges based on the newly learned metric, thus giving a principled way of learning the distance metric in our context-enriched nearest neighbor graph.

3.4 Label Transfer

The superpixel graph provides a simple mechanism for performing label transfer for semantic segmentation. First we build a superpixel graph on a set of training images. Then for each novel image, we introduce the image to the graph and run a small number of move-making iterations (50 in our experiments) with the existing edges fixed. We then transfer labels in the following way.

For each pixel p in the image we construct a distribution $P_p(y_p \mid \mathcal{G})$ over labels by considering all superpixels u that contain p. Formally, we have

$$P_p(y \mid \mathcal{G}) \propto \sum_{u \sim p} \sum_{v:(u,v)\in\mathcal{E}} \lambda_{uv} \hat{P}_v(y) \tag{5}$$

where $u \sim p$ indicates the that pixel p is in superpixel u, and $\hat{P}_v(y)$ is the empirical distribution over labels for superpixel v as described in Section 3.1. Here the term $\lambda_{uv} \geq 0$ controls the relative weight for each matching superpixel. In our work we set λ_{uv} to the inverse rank of v in the sorted list of nearest

neighbors for u. Thus the closest match gets a weight of one, the next closest match a weight of one-half, etc. The inferred label for pixel p is then computed as $y_p^\star = \text{argmax}_y P_p(y \mid \mathcal{G})$.

4 Experimental Results

We conduct extensive experiments on four different scene understanding datasets:

- The Polo dataset [23] is a 6-class dataset comprising images related to the sport of polo. The dataset contains 317 unique images divided into a pre-defined training set of 80 images and test set of 237 images.[2]
- The MSRC dataset [24, 2] is a 21-class dataset consisting of a large variety of images.[3] The dataset contains 591 total images divided into a pre-defined training set of 276 images, validation set of 59 images, and evaluation (test) set of 256 images. We combine training and validation for our experiments.
- The Stanford Background Dataset [25] is an 8-class dataset consisting of 715 images of rural, urban and harbor scenes. Results in the literature report on a training set of size 572 images and test set of 143 images. However, a standard split is not provided.
- The SIFT Flow dataset [9] is a very large 33-class dataset. It consists of 2688 images divided into a pre-defined training set of 2488 images and test set of 200 images. While there are many classes only a few of them dominate.

In all experiments we report the average pixelwise accuracy on the set of test images. We also report the pixelwise accuracy averaged by class. Unless otherwise stated we set the parameters of our over-segmentation algorithm to generate five different sets of superpixels ranging in size from approximately 576 down to 16 superpixels per image.

Efficacy of Search Moves. We first examine the effectiveness of our search moves in constructing a superpixel graph. Figure 2 shows the objective value of Equation 2 as a function of number of iterations for different search strategies. Here we use the subset of training images from the MSRC dataset and do not consider labeling accuracy. The results lead to three interesting observations. First, the objective drops rapidly in the first few iterations with only small improvements in the objective occurring after about 5 or so iterations. Second, the exhaustive search move helps very little in terms of the global objective. This is not surprising given that this move only affects a small number of superpixels. Nevertheless, the move is cheap to compute (on a single superpixel) and does provide a small numerical improvement (barely noticeable in the plots).

The third observation is that propagate, local search, and enrichment moves when used on their own all converge very quickly to a local optimum. However,

[2] We remove the three images from the test set that originally appeared in both training and test sets in [23].

[3] The dataset actually contains 23 classes but standard practice is to remove the "horse" and "mountain" class due to their low occurrence.

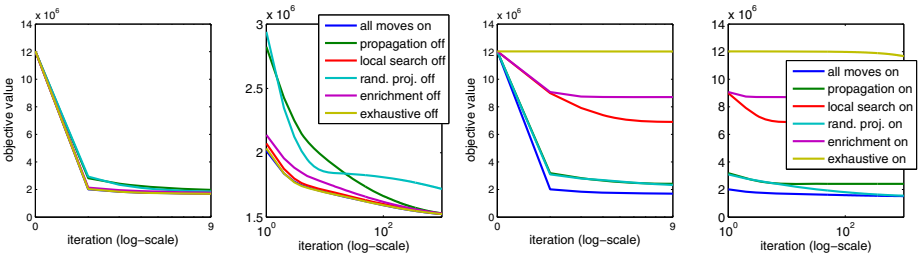


Fig. 2. Objective value as a function of number of iterations for different search strategies: (a) all-but-one strategies from initialization for 10 iterations; (b) all-but-one strategies, initialization not shown, to 1000 iterations; (c) only-one strategy from initialization for 10 iterations; (d) only-one strategy, initialization not shown, to 1000 iterations. Note different vertical scale for (b).

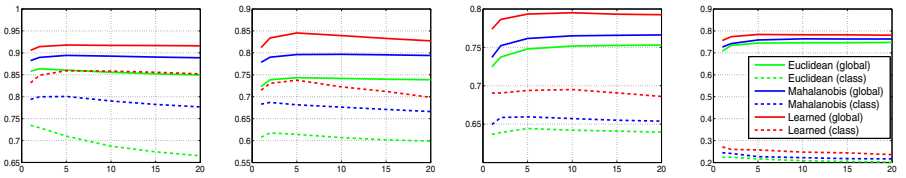
with the other moves they are effective at accelerating the search. Random projection, propagate, and enrichment are all very powerful moves in reducing the objective value, and hence rapidly finding good superpixel matches. This is because these moves exploit the structure of the feature space (random projection), structure of images (propagate), and structure of the graph (enrichment).

Results on Standard Datasets. Our main interest is semantic segmentation accuracy. In this experiment we construct a superpixel graph $\mathcal{G}_k$ over the training set of images using Euclidean distance, diagonal Mahalanobis distance, and the distance metric learned by the large margin nearest neighbor algorithm (see Section 3.3). We then introduce the test images into the graph and evaluate label transfer on these images. Note that labels are only transferred from the training set images to the test set images. Figure 3 shows performance on each dataset as a function of the number of nearest neighbors k. Results for $k = 5$ are listed in Table 1. We conducted repeated runs of our method with different random seeds and found the results varied by less than 0.5%. We also include in the table results obtained by replacing our search method with FLANN [26] (for the learned distance metric), and a comparison to the conceptually similar PatchMatchGraph approach [8] as well as current state-of-the-art methods.

The results show some interesting trends. First, the learned distance metric outperforms the Euclidean norm and Mahalanobis distance, as expected. Moreover, the results are competitive with the state-of-the-art. For example, Ladicky et al. [27] report 87.0% (78.0%) on the MSRC dataset and Tighe and Lazebnik [5] achieve 78.6% (39.2%) on the SIFT Flow dataset compared to our 84.5% (73.8%) and 78.4% (25.7%), respectively. Our class-averaged pixelwise accuracy on the SIFT Flow dataset is low, which we attribute in part to our equal weighting of category labels. The state-of-the-art approach employ per-exemplar detectors to boost performance on less abundant classes, and it would be interesting to see if such an approach could improve our results too.

Table 1. Quantitative experimental results showing percentage pixelwise accuracy and percentage class-averaged accuracy in parentheses on test set images for different datasets at $k = 5$ nearest neighbors per superpixel. Table includes results from (i) replacing our search algorithm with FLANN [26] using the learned metric, (ii) the PatchMatchGraph approach [8], and (iii) state-of-the-art methods.

	Euclidean	Mahal.	Learned	FLANN [26]	PMG [8]	S-of-the-A
Polo	86.1 (71.0)	89.4 (80.1)	91.8 (85.9)	91.7 (85.8)	94.2 (91.7)	94.2 (91.7) [8]
MSRC	74.3 (61.4)	79.6 (68.2)	84.5 (73.8)	82.3 (70.4)	79.0 (72.8)	87.0 (78.0) [27]
Stanford	74.8 (64.4)	76.2 (66.0)	79.3 (69.4)	78.8 (69.1)	73.4 (62.0)	82.9 (74.5) [28]
SIFT Flow	74.5 (21.6)	75.9 (22.7)	78.4 (25.7)	77.5 (24.2)	65.2 (14.9)	78.6 (39.2) [5]



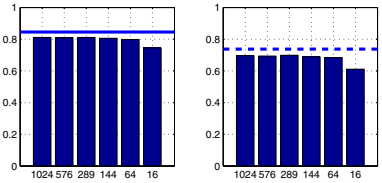
(a) Polo (b) MSRC (c) Stanford (d) SIFT Flow

Fig. 3. Pixelwise semantic segmentation accuracy as a function of the number of nearest neighbors. Shown are global averaged (solid lines) and class averaged (dashed lines) results for different distance metrics.

Second, it is clear from Figure 3 that accuracy saturates at about five nearest neighbors. Interestingly there is a drop in class-averaged accuracy (dashed line) for all metrics as we increase the number of nearest neighbors. This is most pronounced in the Polo dataset. We surmise that the drop is due to the class imbalance in the datasets—as k increases it is more likely that instances with abundant class labels appear in the nearest neighbor set adversely affecting the less abundant classes. The effect can also be seen to a lesser extent for the learned metric in the global-averaged results (solid line). We attribute this to the fact that satisfying the margin constraints is more difficult with larger k and that some classes only appear in a small number of images. For example, only 24 images in the MSRC dataset contain the class "cat".

A handful of qualitative results are shown in Figure 5. Observe that the annotations are quite blocky in parts. This is an artifact of using superpixels as our base representation. Note, however, we have not performed any post processing on the transferred labels. The application of a Markov random field (MRF) or bilateral filter to the results should remove many of these artifacts, but that is not the focus of our investigation here.

Despite the block artifacts, the annotations are generally very good. For example, the horses in the Polo dataset can be reliably detected at different scales, and in the SIFT Flow dataset we are able to correctly label quite difficult categories such as the sidewalk in the bottom right example. Note, however, that our method uses only local context and sometimes makes mistakes. The park



(a) global-averaged (b) class-averaged

Fig. 4. Results on the MSRC dataset showing global and class averaged accuracy for a single over-segmentation of the image as a function of (approximate) number of superpixels. Solid and dashed lines show performance from using all over-segmentations for global and class averaged accuracy, respectively.

bench in the bottom right MSRC example is partially mislabelled as building due to similarity in appearance. The bottom left SIFT Flow example also shows confusion in the transferred labels due to the unusual viewpoint of the scene. We believe that these types of mistakes can be corrected with larger datasets.

Effect of Superpixel Size. Next we evaluate the effect of superpixel size on the label transfer results. Here we generate six sets of over-segmentations where we choose the parameters to give between approximately 1024 superpixels to approximately 16 superpixels per image. We build a superpixel graph with learned distance metric on each set of over-segmentations and apply label transfer as described above. Figure 4 shows results on the MSRC dataset. Other datasets exhibit similar behavior.

The results clearly show that smaller superpixels (more per image) give better accuracy than larger ones (less per image). However, combining multiple over-segmentations leads to even better performance than any single over-segmentation. Moreover, beyond about 289 superpixels per image the accuracy tapers off. Note that in our experiments with combined over-segmentations we did not include the set of 1024 superpixels.

Effect of Dataset Size. An interesting question for all machine learning approaches is whether performance at the task is limited by the dataset size. The nature of our algorithm allows us to go some way in experimentally answering this question. Here we build a superpixel graph over all images in the dataset and perform label transfer, one at a time, to each image in the test set. This is akin to leave-one-out cross-validation, which is expensive to perform on methods with long training phases but easy to do with our approach. To keep our analysis compatible with the previous experiments we use the same distance metric learned on the training set of images with k set to 5 and report average accuracies only on the test set images. Results are shown in Table 2.

For the SIFT Flow dataset and the Stanford Background dataset the effect of increased dataset size is negligible. This can be explained by the already large dataset size in the case of SIFT Flow and the dominance of easy to label background classes in the Stanford Background dataset. More interesting are

Table 2. Quantitative results showing percentage pixelwise accuracy and percentage class-averaged accuracy on test set images for $k = 5$ nearest neighbors with matching against training set images only (standard) versus matching against entire dataset (large). The last column shows the improvement. Note that the distance metric is learned using the training images only.

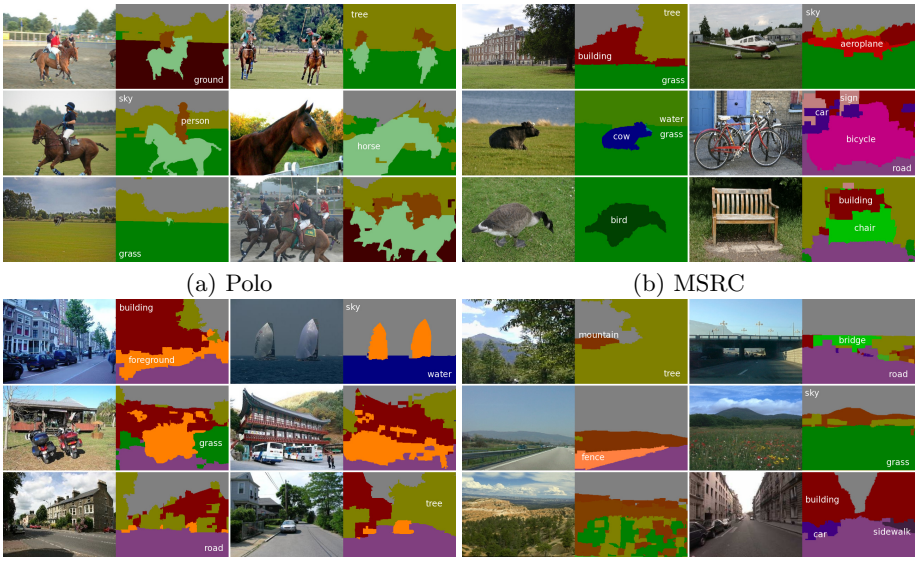
	Standard	Large	Δ
Polo [23]	91.8 (85.9)	92.5 (87.4)	0.7 (1.5)
MSRC [24, 2]	84.5 (73.8)	86.3 (76.6)	1.8 (2.8)
Stanford [25]	79.3 (69.4)	79.6 (69.4)	0.3 (0.0)
SIFT Flow [9]	78.4 (25.7)	78.4 (25.4)	0.0 (-0.3)

the results for the other two datasets. The Polo dataset is a relatively easy dataset with only a small number of categories. However, the training set is small and accuracy is improved when additional images are available at test time. The MSRC dataset shows the greatest improvement in accuracy with the larger dataset, which can be explained by the diversity and difficulty of the classes in the MSRC dataset and the relatively small training set size. These results confirm the intuition that, at least for nearest neighbor techniques, larger datasets are required for recognizing a large and diverse set of classes.

Running Time. Finally, we evaluate the running time of the different steps in our algorithm. All running times were measured on a 3.4GHz Intel Xeon processor with eight cores. Most of the steps are very fast. It takes approximately 0.5s per image to compute the superpixels, 0.3s per image to compute the features, 1.5s per image to build the superpixel graph on the set of training images (for $k = 5$), and 0.4s per image to perform label transfer. Thus at test time it takes around 2.8s to label a novel image. The time depends on the number of superpixels in the test image but is robust to the size of the graph—only the cost of the random projection move scales with graph size but its overhead is small as long as we pre-compute the projections. Our graphs range in size from 320 thousand nodes for the Polo dataset to 2.6 million for the SIFT Flow dataset. Replacing our nearest neighbor search with FLANN reduced processing time per image to 1.2s but accuracy suffers (see Table 1). By far the most expensive step was the metric learning, taking over an hour. Fortunately, this step only needs to be done during training and is quick to apply on each test instance.

5 Discussion and Future Work

This paper has presented a novel approach to semantic segmentation by label transfer. The approach involves two key contribution. The first is the proposal of a method for rapidly building a graph over superpixels that can be thought of as an approximate nearest neighbor algorithm that is context enriched. The second is the inclusion of a learning step to provide a meaningful metric, which relates observed features to unobservable semantics. We performed extensive experiments on four standard datasets and showed that our approach is competitive



(a) Polo (b) MSRC

(c) Stanford (d) SIFT Flow

Fig. 5. Example annotation results on four standard semantic segmentation datasets using our superpixel graph label transfer method. Best viewed in color.

with state-of-the-art. Nevertheless, there is still room for improvement and we are excited about future opportunities suggested by our work.

For example, we would like to investigate learning a non-linear distance metric, which we believe would lead to even better results. This may be achieved, for example, via a kernelized metric learning approach or deep feature architecture (e.g., [29]). More interesting would be combining the metric learning with the superpixel graph construction, which may improve the efficiency of the metric learning algorithm and also facilitate adaptive metric learning on a long-lived and evolving superpixel graph. As dataset sizes grow we believe that nearest neighbor approaches using techniques such as those discussed in this paper will become more and more relevant for scene understanding.

A Derivation of Subgradient Update on L

Eliminating ξ_{uvw} we write Eqn. 3 as the minimization over $M = LL^T \succeq 0$ of

$$\sum_{uv} d_M(u,v) + C \sum_{uvw} \Big[1 - d_M(u,w) + d_M(u,v)\Big]_+ \tag{6}$$

where $[\cdot]_+ = \max\{\cdot, 0\}$. Now let $\mathcal{A} \subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{V}$ be the set of uvw triplets with violated margin constraints, i.e., where $d_M(u,w) - d_M(u,v) < 1$ for $v \in \mathcal{N}_u^+$ and

$w \in \mathcal{N}_u^-$. Then

$$g = \sum_{uv} \nabla_L d_M(u,v) + C \sum_{uvw \in \mathcal{A}} \left(\nabla_L d_M(u,v) - \nabla_L d_M(u,w) \right)$$

is a subgradient of Equation 6, where $\nabla_L d_M(u,v) = 2L^T(\boldsymbol{x}_u - \boldsymbol{x}_v)(\boldsymbol{x}_u - \boldsymbol{x}_v)^T$.

References

1. He, X., Zemel, R.S., Carreira-Perpinan, M.: Multiscale conditional random fields for image labeling. In: CVPR (2004)
2. Shotton, J., Winn, J.M., Rother, C., Criminisi, A.: *TextonBoost*: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part I. LNCS, vol. 3951, pp. 1–15. Springer, Heidelberg (2006)
3. Zhang, Y., Hartley, R., Mashford, J., Burn, S.: Superpixels via pseudo-boolean optimization. In: ICCV (2011)
4. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Susstrunk, S.: SLIC superpixels. Technical Report 149300, EPFL (2010)
5. Tighe, J., Lazebnik, S.: Finding things: Image parsing with regions and per-exemplar detectors. In: CVPR (2013)
6. Tighe, J., Lazebnik, S.: SuperParsing: Scalable nonparametric image parsing with superpixels. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 352–365. Springer, Heidelberg (2010)
7. Zhang, H., Xiao, J., Quan, L.: Supervised label transfer for semantic segmentation of street scenes. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 561–574. Springer, Heidelberg (2010)
8. Gould, S., Zhang, Y.: PatchMatchGraph: Building a graph of dense patch correspondences for label transfer. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part V. LNCS, vol. 7576, pp. 439–452. Springer, Heidelberg (2012)
9. Liu, C., Yuen, J., Torralba, A.: Nonparametric scene parsing: Label transfer via dense scene alignment. In: CVPR (2009)
10. Faktor, A., Irani, M.: "Clustering by composition" – unsupervised discovery of image categories. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part VII. LNCS, vol. 7578, pp. 474–487. Springer, Heidelberg (2012)
11. Malisiewicz, T., Efros, A.A.: Recognition by association via learning per-exemplar distances. In: CVPR (2008)
12. Malisiewicz, T., Gupta, A., Efros, A.A.: Ensemble of exemplar-svms for object detection and beyond. In: ICCV (2011)
13. Barnes, C.: PatchMatch: A Fast Randomized Matching Algorithm with Application to Image and Video. PhD thesis, Princeton University (2011)
14. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: PatchMatch: A randomized correspondence algorithm for structural image editing. In: SIGGRAPH (2009)
15. Barnes, C., Shechtman, E., Goldman, D.B., Finkelstein, A.: The generalized PatchMatch correspondence algorithm. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part III. LNCS, vol. 6313, pp. 29–43. Springer, Heidelberg (2010)

16. Weinberger, K., Saul, L.: Distance metric learning for large margin nearest neighbor classification. JMLR 10, 207–244 (2009)
17. Eigen, D., Fergus, R.: Nonparametric image parsing using adaptive neighbor sets. In: CVPR (2012)
18. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (2005)
19. Felzenszwalb, P., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. PAMI (2010)
20. Ojala, T., Pietikainen, M., Maenpaa, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. PAMI 24, 971–987 (2002)
21. Gould, S.: DARWIN: A framework for machine learning and computer vision research and development. JMLR 13, 3533–3537 (2012)
22. Indyk, P., Motwani, R.: Approximate nearest neighbor—towards removing the curse of dimensionality. In: 30th Symp. of Theory of Comp., pp. 604–613 (1998)
23. Zhang, H., Quan, L.: Partial similarity based nonparametric scene parsing in certain environment. In: CVPR (2011)
24. Criminisi, A.: Microsoft Research Cambridge (MSRC) object recognition pixel-wise labeled image database (version 2) (2004)
25. Gould, S., Fulton, R., Koller, D.: Decomposing a scene into geometric and semantically consistent regions. In: ICCV (2009)
26. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. In: VISSAPP (2009)
27. Ladicky, L., Russell, C., Kohli, P., Torr, P.H.: Associative hierarchical random fields. PAMI (2013)
28. Ren, X., Bo, L., Fox, D.: RGB-(D) scene labeling: Features and algorithms. In: CVPR (2012)
29. Hu, J., Lu, J., Tan, Y.P.: Discriminative deep metric learning for face verification in the wild. In: CVPR (2014)