Nonparametric Image Parsing using Adaptive Neighbor Sets

David Eigen and Rob Fergus
Dept. of Computer Science, Courant Institute, New York University
{deigen,fergus}@cs.nyu.edu

Abstract

This paper proposes a non-parametric approach to scene parsing inspired by the work of Tighe and Lazebnik [22]. In their approach, a simple kNN scheme with multiple descriptor types is used to classify super-pixels. We add two novel mechanisms: (i) a principled and efficient method for learning per-descriptor weights that minimizes classification error, and (ii) a context-driven adaptation of the training set used for each query, which conditions on common classes (which are relatively easy to classify) to improve performance on rare ones. The first technique helps to remove extraneous descriptors that result from the imperfect distance metrics/representations of each super-pixel. The second contribution re-balances the class frequencies, away from the highly-skewed distribution found in real-world scenes. Both methods give a significant performance boost over [22] and the overall system achieves state-of-the-art performance on the SIFT-Flow dataset.

1. Introduction

Densely labeling a scene is a challenging recognition task which is the focus of much recent work [7, 13, 20, 26, 27]. The difficulty stems from several factors. First, the incredible diversity of the visual world means that each region can potentially take on one of hundreds of different labels. Second, the distribution of classes in a typical scene is far from uniform, following a power-law (as illustrated in Fig. 8). Consequently, many classes will have a small number of instances even in a large dataset, making it hard to train good classifiers. Third, as noted by Frome *et al.* [3], the use of single global distance metric for all descriptors is insufficient to handle the large degree of variation found in a given class. For example, the position within the image may sometimes be an important cue for finding people (e.g. when they are walking on a street), but on other occasions position may be irrelevant and color a much better feature (e.g. the person is close and facing the camera).

In this paper we propose a non-parametric approach to scene parsing that addresses the latter two of these factors. Our method is inspired by the simple and effective method of Tighe and Lazebnik for scene parsing [22]. They show excellent performance using nearest-neighbor methods on image super-pixels, represented by a variety of feature types which are combined in a naive-Bayes framework. We build on their approach, introducing two novel ideas:

1. In an off-line training phase, we learn a set of weights for each descriptor type of every segment in the training set. The weights are trained to minimize classification error in a weighted nearest-neighbor (NN) scheme. Individually weighting each descriptor has the effect of introducing a distance metric that varies throughout the descriptor space. This enables it to overcome the limitations of a global metric, as outlined above. It also allows us to discard outlier descriptors that would otherwise hurt performance (e.g. from segmentation errors).

2. At query-time, we adapt the set of points used by the weighted-NN classification based on context from the query image. We first remove segments based on a global context match. Crucially, we then add back previously discarded segments from rare classes. Here we use the local context of segments to look up rare class examples from the training set. This boosts the representation of rare classes within the NN sets, giving a more even class distribution that improves classification accuracy.

The overall theme of our work is the customization of the dataset for a particular query to improve performance.

1.1. Related Work

Apart from Tighe and Lazebnik [22], other related non-parametric approaches to recognition include: the SIFT-Flow scene parsing method of Liu *et al.* [13, 14]; scene classification using Tiny Images by Torralba *et al.* [23] and the Naive-Bayes NN approach from Boiman *et al.* [1]. However, none of these involve re-weighting of the data, and context is limited to a CRF at most.

Our re-weighting approach has interesting similarities to Frome *et al.* [3] (and related work from Malisiewicz &

Efros [15, 16]). Motivated by the inadequacies of a single global distance metric, they use a different metric for each exemplar in their training set, which is integrated into an SVM framework. The main drawback to this is that the evaluation of a query is slow ($\sim$minutes/image). The weights learned by our scheme are equivalent to a local modulation of the distance metric, with a large weight moving the point closer to a query, and vice-versa. Furthermore, the context-based training set adaptation in our method also effects a query-dependent metric on the data.

The re-weighting scheme we propose has connections to a traditional machine learning approach called editing [2, 11]. These approaches are usually binary in that they either keep or completely remove each training point. Of this family, the most similar to ours is Paredes and Vidal [18], who also use real-valued weights on the points. However, their approach does not handle multiple descriptor types and is demonstrated on a range of small text classification datasets.

There is extensive work on using context to help recognition [10, 17, 24, 25]; the most relevant approaches being those of Gould *et al.* [5, 6] and in particular Heitz & Koller [9] who use "stuff" to help find "things." Heitz *et al.* [8] use similar ideas in a sophisticated graphical model that reasons about objects, regions and geometry. These works have similar goals regarding the use of context but quite different methods. Our approach is simpler, relying on NN lookups and standard gradient descent for learning the weights.

Our work also has similar goals to multiple kernel learning approaches (e.g. [4]) which combine weighted feature kernels, but the underlying mechanisms are quite different: we do not use SVMs, and our weights are per-descriptor. By contrast, the weights used in these methods are constant across all descriptors of a given type. Finally, Boosting [19] is an approach that weights each datapoint individually, as we do, but it is based on parametric models rather than non-parametric ones.

2. Approach

Our approach builds on the nearest-neighbor voting framework of Tighe and Lazebnik [22] and uses three distinct stages to classify image segments: (i) global context selection; (ii) learning descriptor weights; (iii) adding local context segments. Stages (i) and (ii) are used in off-line training, while (i) and (iii) are used during evaluation. While stage (i) is adopted from [22], the other two stages are novel and the main focus of our paper.

A query image Q consists of a set of super-pixel segments q, each of which we need to classify into one of C classes. The training dataset T consists of super-pixel segments s, taken from images I_1 to I_M. The true class c_s^* for each segment in T is known. Each segment is represented by D different types of descriptors (the same set of 19 used in [22].[1] Additionally, each image I_m has a set global context descriptors, $\{g_m\}$ that capture the content of the entire image; these are computed in advance and stored in kd-trees for efficient retrieval.

2.1. Global Context Selection

In this stage, we use overall scene appearance to remove descriptors from scenes bearing little resemblance to the query. For example, the segments taken from a street scene are likely to be distractors when trying to parse a mountain scene. Thus their removal is expected to improve performance. A secondary benefit is that the subsequent two stages need only consider a small subset of the training dataset T, which gives a considerable speed-up for big datasets.

For each query Q we compute global context descriptors $\{g_q\}$, which consists of 4 types: (i) a spatial pyramid of vector quantized SIFT [12]; (ii) a color histogram spatial pyramid and (iii) Gist computed with two different parameter settings [17]. For each of the types, we find the nearest neighbors amongst the training set $\{g_m\}$. The ranks across the four types of context descriptor are averaged to give an overall ranking. We then form a subset G of the segment-level training database T that consists of segments belonging to the top v images from our image-level ranking. We denote the global match set $G = \text{GLOBALMATCHES}(Q, v)$. v is an important parameter whose setting we explore in Section 4.

2.2. Learning Descriptor Weights

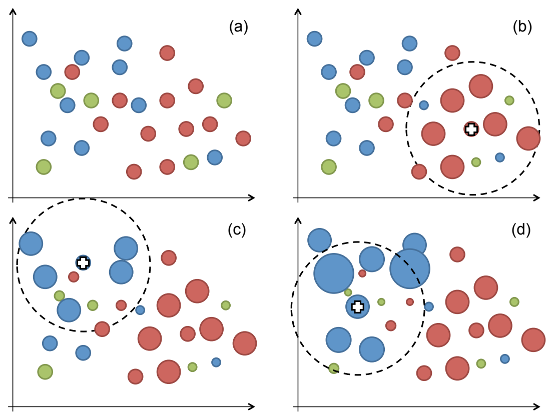

Figure 1. Toy example of our re-weighting scheme. (a): Initially all descriptors have uniform weight. (b), (c) & (d): a probe point is chosen (cross) and points in the neighborhood (black circle) of the same class as the probe have their weights increased. Points of a different class have their weights decreased, so rejecting outlier points. In practice, (i) there are multiple descriptor spaces, one for each descriptor type and (ii) the GLOBALMATCH operation removes some of the descriptors.

[1] These include quantized SIFT, color, position, shape and area features.

To learn the weights, we adopt a leave-one-out strategy, using each segment s (from image I_m) in the training dataset T as probe segment (a pretend query). The weights of the neighbors of s are then adjusted to increase the probability of correctly predicting the class of s.

For a query segment s, we first compute the global match set $G_s = \textsc{GlobalMatches}(I_m, v)$. Let the set of descriptors of s be D_s. Following [22], the predicted class $\hat{c}$ for each segment is the one that maximizes the ratio of posterior probabilities $P(c|D_s)/P(\bar{c}|D_s)$. After the application of Bayes rule using a uniform class prior[2] and making a naive-Bayes assumption for combining descriptor types, this is equivalent to maximizing the product of likelihood ratios for each descriptor type:

$$\hat{c} = \arg\max_c L(s,c) = \arg\max_c \prod_{d \in D_s} \frac{P(d|c)}{P(d|\bar{c})} \tag{1}$$

The probabilities $P(d|c)$ and $P(d|\bar{c})$ are computed using nearest-neighbor lookups in the space of the descriptor type of d, over all segments in the global match set G. In the un-weighted case (i.e. no datapoint weights), this is:

$$P(d|c) \propto p_d(c) = \frac{n_d^N(c)}{n_d(c)}, \quad P(d|\bar{c}) \propto \bar{p}_d(c) = \frac{\bar{n}_d^N(c)}{\bar{n}_d(c)}$$

where $n_d^N(c)$ is the number of points of class c in the nearest neighbor set N of d, determined by taking the closest k neighbors of d[3]. $n_d(c)$ is the total number of points in class c. $\bar{n}_d^N(c)$ is the number of points *not* of class c in the nearest neighbor set N of d (i.e. $\sum_{c' \neq c} n_d^N(c')$), and similarly for $\bar{n}_d$. Conceptually, both $n_d^N(c)$ and $n_d(c)$ should be computed over the match set G; in practice, this sample may be small enough that using G just for $n_d^N(c)$ and estimating $n_d(c)$ over the entire training database T can reduce noise.

To eliminate zeros in $P(d|\bar{c})$, we smooth the above probabilities using a smoothing factor t:

$$q_d(c) = (n_d^N(c) + \bar{n}_d^N(c))^2 \cdot p_d(c) + t \tag{2}$$
$$\bar{q}_d(c) = (n_d^N(c) + \bar{n}_d^N(c))^2 \cdot \bar{p}_d(c) + t \tag{3}$$

and define the smoothed likelihood ratio $L_d(c)$:

$$L_d(c) = \frac{q_d(c)}{\bar{q}_d(c)}$$

We now introduce weights w_{di} for each descriptor d of each segment i. This changes the definitions of n_d and n_d^N:

$$n_d(c) = \sum_{i \in T} w_{di} \delta(c_i^*, c) = W^T \Delta$$

[2]Using the true, highly-skewed, class distribution $P(c)/P(\bar{c})$ dramatically impairs performance for rare classes.

[3]We also include all points at zero distance from d, so $n_d^N(c)$ is occasionally larger than k.

$$n_d^N(c) = \sum_{i \in N} w_{di} \delta(c_i^*, c) = W^T \Delta^N$$

where c_i^* is the true class of point i and T is the training set. Note that when using only the match set G to estimate $n_d(c)$, the sum over T need only be performed over G. In matrix form, W is the vector of weights w_{di}, and Δ is the $|T| \times |C|$ class indicator matrix whose ci-th entry is $\delta(c_i, c)$. For neighbor counts, Δ^N is the restriction of Δ to the neighbor set N — that is, its entries in rows $i \notin N$ are zero.

Similarly, for $\bar{n}_d(c)$ and $\bar{n}_d^N(c)$ we use the complement $\bar{\Delta} = 1 - \Delta$:

$$\bar{n}_d(c) = \sum_{i \in T} w_{di} \delta(c_i^*, \bar{c}) = W^T \bar{\Delta}$$

$$\bar{n}_d^N(c) = \sum_{i \in N} w_{di} \delta(c_i^*, \bar{c}) = W^T \bar{\Delta}^N$$

To train the weights, we choose a negative log-likelihood loss:

$$\begin{aligned} J(W) &= \sum_{s \in T} J_s(W) = \sum_{s \in T} -\log L(s, c^*) + \log \sum_{c \in C} L(s,c) \\ &= \sum_{s \in T} \left(-\sum_{d \in D_s} \log L_d(c^*) + \log \sum_{c \in C} \prod_{d \in D_s} L_d(c) \right) \end{aligned}$$

The derivatives with respect to W are back-propagated through the nearest neighbor probability calculations using 5 chain rule steps. The vector of weights W_d (the weights for all segments on descriptor type d) is updated as follows:

Step 1:

$$\frac{\partial n_d}{\partial W_d} = \Delta, \quad \frac{\partial n_d^N}{\partial W_d} = \Delta^N, \quad \frac{\partial \bar{n}_d}{\partial W_d} = \bar{\Delta}, \quad \frac{\partial \bar{n}_d^N}{\partial W_d} = \bar{\Delta}^N$$

Step 2:

$$\frac{\partial p_d}{\partial W_d} = (\Delta^N - p_d \cdot \Delta)/n_d, \quad \frac{\partial \bar{p}_d}{\partial W_d} = (\bar{\Delta}^N - \bar{p}_d \cdot \bar{\Delta})/\bar{n}_d$$

Step 3:

$$\frac{\partial q_d}{\partial W_d} = 2(n_d^N + \bar{n}_d^N) \cdot p \cdot 1^N + (n_d^N + \bar{n}_d^N)^2 \cdot \frac{\partial p_d}{\partial W_d}$$

$$\frac{\partial \bar{q}_d}{\partial W_d} = 2(n_d^N + \bar{n}_d^N) \cdot \bar{p} \cdot 1^N + (n_d^N + \bar{n}_d^N)^2 \cdot \frac{\partial \bar{p}_d}{\partial W_d}$$

Step 4:

$$\frac{\partial \log L_d}{\partial W_d} = \frac{1}{q_d} \frac{\partial q_d}{\partial W_d} - \frac{1}{\bar{q}_d} \frac{\partial \bar{q}_d}{\partial W_d}$$

Step 5:

$$\frac{\partial J_s}{\partial W_d} = -\frac{\partial \log L_d}{\partial W_d}(c^*) + \frac{1}{\sum_c L(c)} \sum_c L(c) \cdot \frac{\partial \log L_d(c)}{\partial W_d}$$

where $1^N = \Delta^N + \bar{\Delta}^N$, and products and divisions are performed element-wise. The weight matrix is updated using gradient descent:

$$W \leftarrow W - \eta \frac{\partial J_s}{\partial W}$$

where η is the learning rate parameter. In addition, we enforce positivity and upper bound constraints on each weight, so that $0 \leq w_{di} \leq 1$ for all d, i. We initialize the learning with all weights set to 0.5 and η set to 0.1.

The above procedure provides a principled approach to maximizing the classification performance, using the same naive-Bayes framework of [22]. It is also practical to deploy on large datasets: although the the time to compute a single gradient step is $O(|T||C|)$, we found that fixing n_d and $\bar{n}_d$ to their values with the initial weights yields good performance, and limits the time for each step to $O(|G||C|)$.

2.2.1 Effect of the Smoothing Parameter

Aside from smoothing the NN probabilities, the smoothing parameter t also modulates $L_d(c)$ as a function of $n_d(c)$, the number of descriptors of each class. As such, it gives a natural way to bias the algorithm toward common classes or toward rare ones.

To see this, let us assume $n_d^N(c) + \bar{n}_d^N(c) = k$ (which is usually the case; see footnote 3). This lets us rearrange $L_d(c)$ to obtain (omitting d for brevity and defining $u = t/k^2$):

$$L(c) = \frac{n^N(c)\bar{n}(c) + u \cdot n(c)\bar{n}(c)}{\bar{n}^N(c)n(c) + u \cdot n(c)\bar{n}(c)}$$

Note that $n(c)\bar{n}(c)$ depends only on the frequency of class c in the dataset, not on the NN lookup. The influence of t therefore becomes larger for progressively more common classes. So by increasing t we bias the algorithm toward rare classes, an effect we systematically explore in Section 4.

2.3. Adding Segments

The global context selection procedure discards a large fraction of segments from the training set T, leaving a significantly smaller match set G. This restriction means that rare classes may have very few examples in G — and sometimes none at all. Consequently, (i) the sample resolution of rare classes is too small to accurately represent their density, and (ii) for NN classifiers that use only a single lookup among points of all classes (as ours does), common points may fill a search window before any rare ones are reached. We seek to remedy this by explicitly adding more segments of rare classes back into G.

To decide which points to add, we index rare classes using a descriptor based on semantic context. Since the classifier is already fairly accurate at common background classes, we can use its existing output to find probable background labels around a given segment. The context descriptor of a segment is the normalized histogram of class labels in the 50 pixel dilated region around it (excluding the segment region itself). See Fig. 2(a) & (b) for an illustration of this operation, which we call MAKECONTEXTDESCRIPTOR.

To generate the index, we perform leave-one-out classification on each image in the training set, and index each super-pixel whose class occurs below a threshold of r times in its image's match set G. In this way, the definition of a rare class adapts naturally according to the query image. This the BUILDCONTEXTINDEX operation.

When classifying a test image, we first classify the image without any extra segments. These labels are used to generate the context descriptors as described above. For each super-pixel, we look up the nearest r points in the rare segments index, and add these to the set of points G used to classify that super-pixel. See Algorithm 2 for more details.

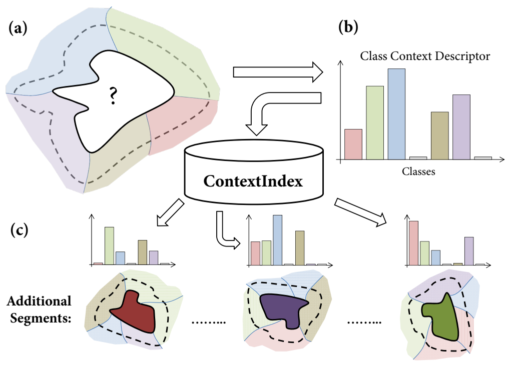

Figure 2. Context-based addition of segments to the global match set G. (a): Segment in the query image, surrounded by an initial label map. (b): Histogram of class labels, built by dilating the segment over the label map, which captures the semantic context of the region. This is matched with histograms built in the same manner from the training set T. (c): Segments in T with a similar surrounding class distribution are added to G.

3. Algorithm Overview

The overall training procedure is summarized in Algorithm 1. We first learn the weights for each segment/descriptor, before building the context index that will be used to add segments at test time. Note that we do not rely on ground truth labels for constructing this index, since not all segments in T are necessarily labeled. Instead, we use the predictions from our weighted NN classifier. NN algorithms work better with more data, so to boost performance we make a horizontally flipped copy of each training image and add it to the training set.

The evaluation procedure, shown in Algorithm 2, involves two distinct classifications. The first uses the weighted NN scheme to give an initial label set for the query image. Then we lookup each segment in the ContextIndex structure to augment G with more segments from rare classes. We then run a second weighted classification using this extended match set to give the final label map.

Algorithm 1 Training Procedure

```
1:  procedure LearnWeights($T$)
2:      Parameters: $v, k$
3:      $W_{di} = 0.5$
4:      for all segments $s \in T$ do
5:          $G$ = GlobalMatches($I_m, v$)
6:          NN-lookup to obtain $\Delta^N, \bar{\Delta}^N$
7:          Compute $\frac{\partial J_s}{\partial W_d}$
8:          $W_d \leftarrow W_d - \eta \frac{\partial J_s}{\partial W_d}$
9:      end for
10: end procedure

11: procedure BuildContextIndex($T, W$)
12:     Parameters: $v, k$
13:     ContextIndex = $\varnothing$
14:     for all $I \in T$ do
15:         $G$ = GlobalMatches($I, v$)
16:         label_map = Classify($I, G, W, k$)
17:         for all Segments $s$ in $I$ with rare $\hat{c}_s$ in $G$ do
18:           desc = MakeContextDescriptor($s$, label_map)
19:           Add (desc $\rightarrow I, s$) to ContextIndex
20:         end for
21:     end for
22: end procedure

23: function Classify($I, G, W, k$)
24:     for all segments $s \in$ image $I$ do
25:         $k$NN-lookup in $G$ to obtain $\Delta^N, \bar{\Delta}^N$
26:         Use weights $W$ to compute $n_d^N(c), \bar{n}_d^N(c)$ and $L_d(c)$
27:         $\hat{c}_s = \underset{c}{\text{argmax}} \prod_d L_d(c)$
28:     end for
29:     return label_map $\hat{c}$
30: end function
```

Algorithm 2 Evaluation Procedure

```
1:  procedure EvaluateTestImage($Q$)
2:      Parameters: $v, k, r$
3:      $G$ = GlobalMatches($Q, v$)
4:      init_label_map = Classify($Q, G, W, k$)
5:      for all segments $s \in Q$ do
6:       desc = MakeContextDescriptor($s$, init_label_map)
7:       $H_s$ = ContextMatches(desc,ContextIndex,r)
8:      end for
9:      final_label_map = Classify($Q, G \cup H, W, k$)
10: end procedure
```

4. Experiments

We evaluate our approach on two datasets: (i) Stanford background [5] (572/143 training/test images, 8 classes) and (ii) the larger SIFT-Flow [13] dataset (2488/200 training/test images, densely labeled with 33 object classes).

In evaluating sense parsing algorithms there are two metrics that are commonly used: per-*pixel* classification rate and per-*class* classification rate. If the class distribution were uniform then the two would be the same, but this is not the case for real-world scenes. A problem with optimizing pixel error alone is that rare classes are ignored since they occupy only a few percent of image pixels. Consequently, the mean class error is a more useful metric for applications that require performance on all classes, not just the common ones. Our algorithm is able to smoothly trade off between the two performance measures by varying the smoothing parameter t at evaluation time. Using a 2D plot for the pair of metrics, the curve produced by varying t gives the full performance picture for our algorithm.

Our baseline is the system described in Section 2, but with no image flips, no learned weights (i.e. they are uniform) and no added segments. It is essentially the same as the Tighe and Lazebnik [22], but with a slightly different smoothing of the NN counts. Our method relies on the same set of 19 super-pixel descriptors used by [22]. As other authors do, we compare the performance without an additional CRF layer so that any differences in local classification performance can be seen clearly. Our algorithm uses the following parameters for all experiments (unless otherwise stated): $v = 200$, $k = 10$, $r = 200$.

4.1. Stanford Background Dataset

Fig. 3 shows the performance curve of our algorithm on the Stanford Background dataset, along with the baseline system. Also shown is the result from Gould *et al.* [5], but since they do not measure per-class performance, we show an estimated range on the x-axis. While we convincingly beat the baseline and do better than Gould *et al.* [4], our best per-pixel performance of 75.3% fall short of the current state-of-the-art on the dataset, 78.1% by Socher *et al.* [21]. The small size of the training set is problematic for our algorithm, since it relies on good density estimates from the NN lookup. Indeed, the limited size of the dataset means that the global match set is most of the dataset (i.e. $|G|$ is close to $|T|$), so the global context stage is not effective. Furthermore, since there are only 8 classes, adding segments using contextual cues gave no performance gain either. We therefore focus on the SIFT-Flow dataset which is larger and better suited to our algorithm.

4.2. SIFT-Flow Dataset

The results of our algorithm on the SIFT-Flow dataset are shown in Fig. 4, where we compare to other approaches using local labeling only. Both the trained weights and

[4]Assuming some a per-class performance consistent with their per-pixel performance.

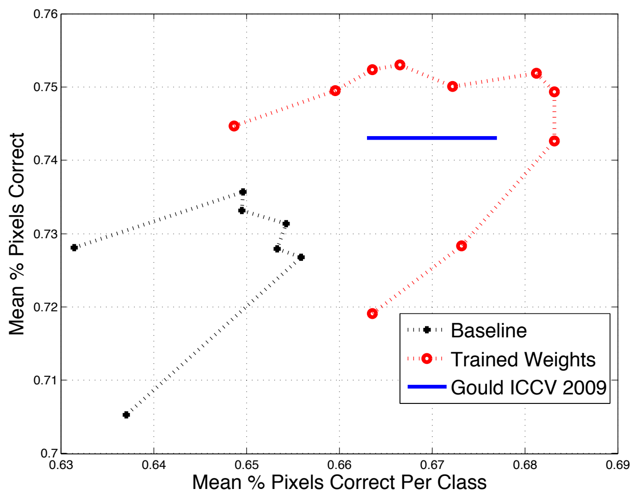

Figure 3. Evaluation of our algorithm on the Stanford background dataset, using local labeling only. x-axis is mean per-class classification rate, y-axis is mean per-pixel classification rate. Better performance corresponds to the top right corner. Black = Our version of [22]; Red = Our algorithm (without added segments step); Blue = Gould *et al.* [5] (estimated range).

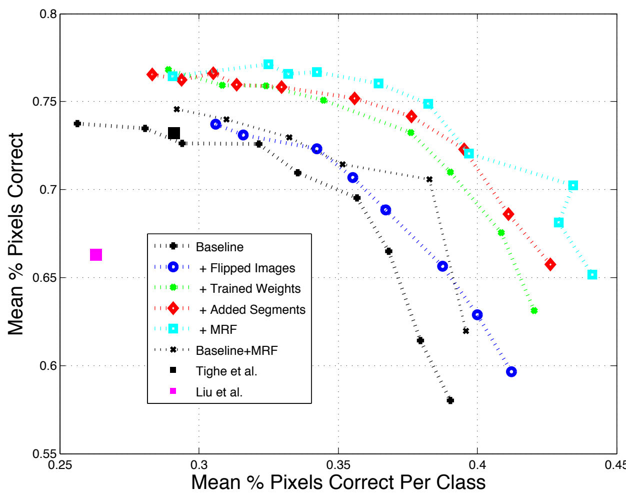

Figure 4. Evaluation of our algorithm on the SIFT-Flow dataset. Better performance is in the top right corner. Our implementation of [22] (black + curve) closely matches their published result (black square). Adding flipped versions of the images to the training set improves the baseline a small amount (blue). A more significant gain is seen when after training the NN weights (green). Refining our classification after adding segments (red) gives a further gain in per-class performance. Adding an MRF (cyan) also gives further gain. Also shown is Liu *et al.* [13] (magenta). Not shown is Shotton *et al.* [20]: 0.13 class, 0.52 pixel.

adding segments procedures give a significant jump in performance. The latter procedure only gives a per-class improvement, consistent with its goal of helping the rare classes (see Fig. 8 for the class distribution).

To the best of our knowledge, Tighe and Lazebnik [22] is the current state-of-the-art method on this dataset (Fig. 4, black square). For local labeling, our overall system outperforms their approach by **10.1%** (29.1% vs 39.2%) in per-class accuracy, for the same per-pixel performance, a 35% relative improvement. The gain in per-pixel accuracy is **3.6%** (73.2% vs 76.8%).

Adding an MRF to our approach (Fig. 4, cyan curve) gives 77.1% per-pixel and 32.5% per-class accuracy, outperforming the best published result of Tighe and Lazebnik [22] (76.9% per-pixel and 29.4% per-class). Note that their result uses geometric features not used by our approach. Adding an MRF to our implementation of their system gives a small improvement over the baseline which is significantly outperformed by our approach + an MRF.

Sample images classified by our algorithm are shown in Fig. 9. We also demonstrate the significance of our results by re-running our methods on a different train/test split of the SIFT-Flow dataset. The results obtained are very similar to the original split and are shown in Fig. 5.

In Fig. 6, we explore the role of the global context selection by varying the number of image-level matches, controlled by the v parameter which dictates $|G|$. For small values performance is poor. Intermediate v gives improved performance under both metrics. But if v is too large, G contains many unrelated descriptors and the per-class performance is decreased. This demonstrates the value of the global context selection procedure, since without it $G = T$, and the per-class performance would be poor.

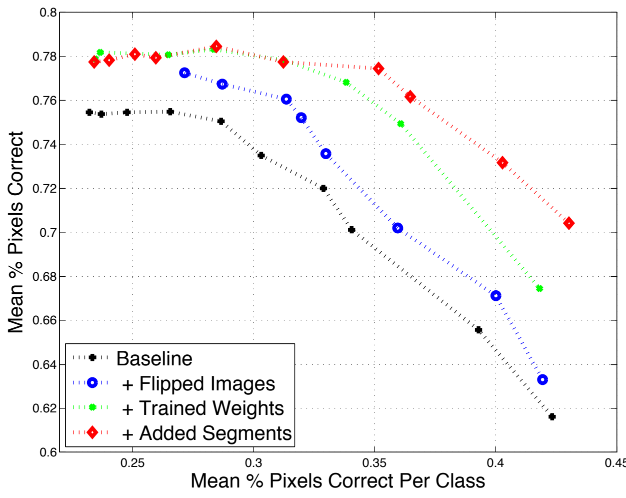

Figure 5. Results for a different train/test split of the SIFT-Flow dataset to one standard one used in Fig. 4. Similar results are obtained on both test sets.

In Fig. 7 we visualize the descriptor weights, showing how they vary across class and descriptor type (by averaging them over all instances of each class, since they differ for each segment). Note how the weights jointly vary across both class and descriptor. For example, the min_height descriptor usually has high weight, except for some spa-

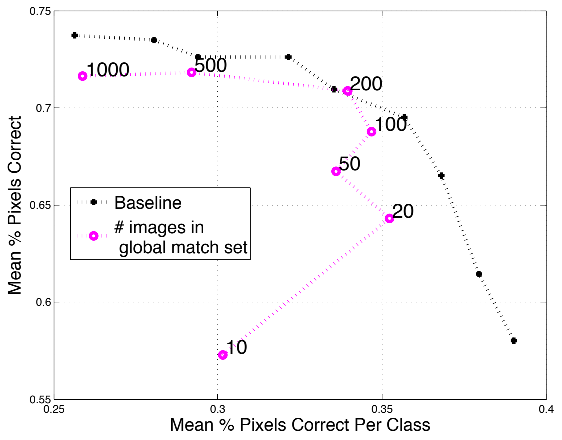

Figure 6. The global context selection procedure. Changing the parameter v (value at each magenta dot) affects both types of error. See text for details. For comparison, the baseline approach using a fixed $v = 200$ (and varying the smoothing t) is shown.

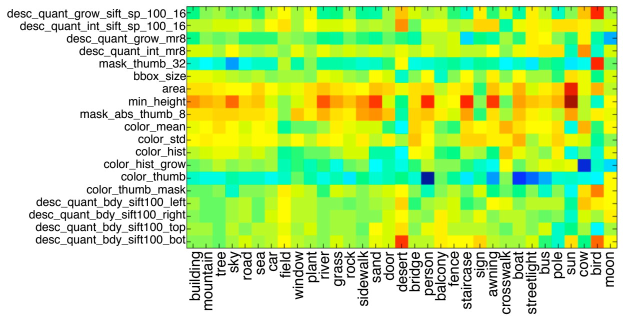

Figure 7. A visualization of the mean weight for different classes by descriptor type. Red/Blue corresponds to high/low weights. See text for details.

tially diffuse classes (e.g. desert,field) where its weight is low.

Fig. 8 shows the expected class distribution of super-pixels in G for the SIFT-Flow dataset before and after the adding segments procedure, demonstrating its efficacy. The increase in rare segments is important in improving per-class accuracy (see Fig. 4).

In Table 1, we list the timings for each stage of our algorithm running on the SIFT-Flow dataset, implemented in Matlab. Note that a substantial fraction of the time is just taken up with descriptor computations. The search parts of our algorithm run in a comparable time to other non-parametric approaches [22], being considerably faster than methods that use per-exemplar distance measures (e.g. Frome *et al.* [3] which takes 300s per image).

5. Discussion

We have described two novel mechanisms for enhancing the performance of non-parametric scene parsing based on NN methods. Both share the underlying idea of customizing the dataset for each NN query. Rather than assuming that the full training set is optimally discriminative, adapting the dataset allows for better use of imperfectly generated descriptors with limited power. Learning weights focuses the classifier on more discriminative features and removes outlier points. Likewise, context-based adaptation uses information beyond local descriptors to remove distractor super-pixels whose appearances are indistinguishable from those of relevant classes. Reintroducing rare class examples improves density lost in the initial global pruning. On sufficiently large datasets, both contributions give a significant performance gain, with our best performance exceeding the current state-of-the-art on the SIFT-Flow dataset. Our code has been made available at
http://www.cs.nyu.edu/~deigen/adaptnn/.

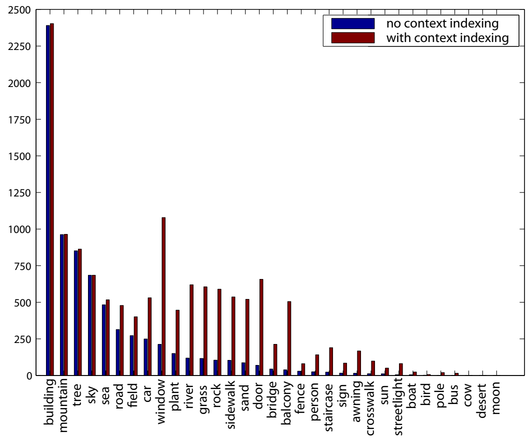

Figure 8. Expected number of super-pixels in G with the same true class c_s^* of a query segment, ordered by frequency (blue). Note the power-law distribution of frequencies, with many classes having fewer than 50 counts. Following the Adding Segments procedure, counts of rare classes are significantly boosted while those for common classes are unaltered (red). Queries were performed using the SIFT-Flow dataset.

	Learned Weights	Full
Global Descriptors	2.8	2.8
Segment Descriptors	3.0	3.0
GLOBALMATCH	0.9	0.9
CLASSIFY	3.5	3.5
CONTEXTMATCHES	-	0.4
CLASSIFY	-	6.1
Total	10.3	16.6

Table 1. Timing breakdown (seconds) for the evaluation of a single query image using the full system and our system without adding segments (just global context match + learning weights). Note the descriptor computation makes up around half of the time.

Acknowledgments

This research was supported by the NSF CAREER award IIS-1149633. The authors would like to thank David Sontag for helpful discussions.

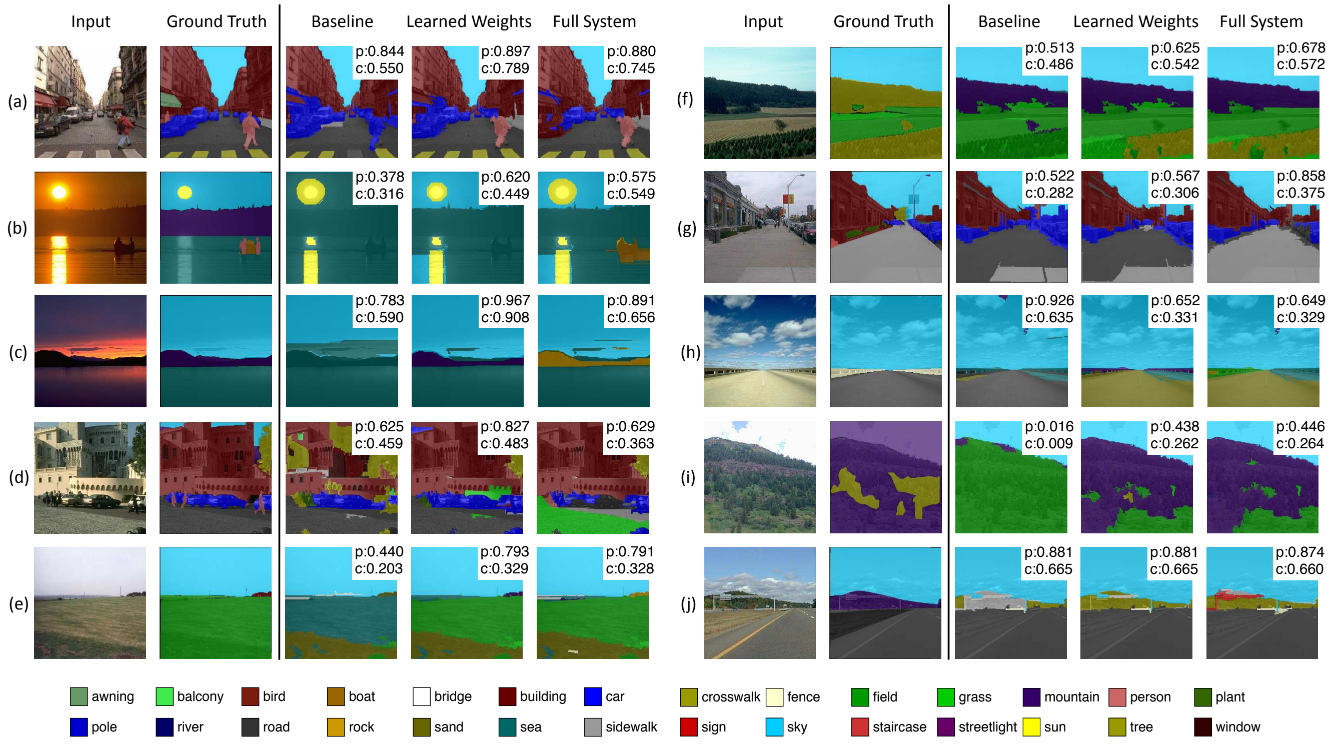

Figure 9. Example images from the SIFT-Flow dataset, annotated with classification rates using per-pixel ("p") and per-class ("c") metrics. Learning weights improves overall performance. Adding rare class examples improves classification of less common classes, like the boat in (b) and sidewalk in (g). Failures include labeling the road as sand in (h) and the mountain as rock (a rarer class) in (c).

References

[1] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008. 1

[2] P. Devijver and J. Kittler. *Pattern Recognition. A Statistical Approach*. Prentice Hall, 1992. 2

[3] A. Frome, Y. Singer, and J. Malik. Image retrieval and classification using local distance functions. In *NIPS*, 2006. 1, 7

[4] P. V. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *ICCV*, 2009. 2

[5] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semnatically consistent regions. In *CVPR*, 2009. 2, 5, 6

[6] S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller. Multi-class segmentation with relative location prior. *IJCV*, 200. 2

[7] X. He, R. Zemel, and M. Carreira-Perpinan. Multiscale crfs fro image labeling. In *CVPR*, 2004. 1

[8] G. Heitz, S. Gould, A. Saxena, and D. Koller. Cascaded classifcation models: Combining models for holistic scene understanding. In *NIPS*, 2008. 2

[9] G. Heitz and D. Koller. Learning spatial context: using stuff to find things. In *CVPR*, 2008. 2

[10] D. Hoiem, A. Efros, and M. Hebert. Closing the loop on scene interpretation. In *CVPR*, 2008. 2

[11] J. Koplowitz and T. Brown. On the relation of the performance to editing in nearest neighbor rules. *Pattern Recognition*, 13(3):251–255, 1981. 2

[12] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, pages 2169–2178, 2006. 2

[13] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing: label transfer via dense scene alignment. In *CVPR*, 2009. 1, 5, 6

[14] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. Freeman. Sift flow: dense correspondence across difference scenes. In *ECCV*, 2008. 1

[15] T. Malisiewicz and A. Efros. Recognition by association via learning per-exemplar distances. In *CVPR*, 2008. 2

[16] T. Malisiewicz, A. Gupta, and A. Efros. Ensemble of exemplar-svms for object detection. In *ICCV*, 2011. 2

[17] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 42:145–175, 2001. 2

[18] R. Paredes and E. Vidal. Learning weighted metrics to minimize nearest-neighbor classification error. *IEEE PAMI*, 28(7):1100–1100, 2006. 2

[19] R. Schapire. The boosting approach to machine learning: An overview. In D. D. Denison, M. H. Hansen, C. Holmes, B. Mallick, and B. Yu, editors, *Nonlinear Estimation and Classification*. Springer, 2003. 2

[20] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008. 1, 6

[21] R. Socher, C. Lin, A. Y. Ng, and C. D. Manning. Parsing natural scenes and natural language using recursive neural networks. In *ICML*, 2001. 5

[22] J. Tighe and S. Lazebnik. Superparsing: Scalable nonparametric image parsing with superpixels. In *ECCV*, 2010. 1, 2, 3, 4, 5, 6, 7

[23] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: a large database for non-parametric object and scene recognition. *IEEE PAMI*, 30(11):1958–1970, November 2008. 1

[24] A. Torralba, K. P. Murphy, and W. T. Freeman. Contextual models for object detection using brfs. In *NIPS*, 2005. 2

[25] Z. Tu. Auto-context and its application to high-level vision tasks. In *CVPR*, 2008. 2

[26] Z. Tu, X. Chen, A. L. Yuille, and S.-C. Zhu. Image parsing: Unifying segmentation, detection, and recognition. In *ICCV*, 2003. 1

[27] J. Winn and J. Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects.auto-context and its application to high-level vision tasks. In *CVPR*, 2006. 1