Investigation of Segmentation Based Pooling for Image Quantification

Reid Porter[*a], Neal Harvey[a], Christy Ruggiero[b]
[a]Intelligence and Space Research; [b]Nuclear Engineering and Nonproliferation,
Los Alamos National Laboratory, Los Alamos, New Mexico, USA 87545

ABSTRACT

A key step in many image quantification solutions is feature pooling, where subsets of lower-level features are combined so that higher-level, more invariant predictions can be made. The pooling region which defines the subsets often has a fixed spatial size and geometry, but data adaptive pooling regions have also been used. In this paper we investigate pooling strategies for the data adaptive case and suggest a new framework for pooling that uses multiple sub-regions instead of a single region. We show that this framework can help represent the shape of the pooling region and also produce useful pairwise features for adjacent pooling regions. We demonstrate the utility of the framework in a number of classification tasks relevant to image quantification in digital microscopy.

Keywords: segmentation, semantic segmentation, feature pooling, image quantification, microscopy

1. INTRODUCTION

In material science and bio-medical domains there is great value in automatically detecting, delineating and quantifying particles, grains, cells and neurons within digital microscopy images. The immediate value of automating image quantification is that it frees the application expert, who is all too often performing the analysis manually. The longer term value appears to be even greater. As the volume and throughput of image acquisition systems increases, automating analysis becomes a critical tool for high-volume scientific studies which aim to understand the link between experimental inputs (e.g. gene sequences and chemical processes) and morphological outputs (e.g. phenotype and material structure) that are observable through microscopy. Advancing automated image quantification can therefore enable new scientific discoveries in a number of application domains [2].

Of course, automating image quantification is extremely challenging. The content of interest is specific to the application, it often involves multiple compound objects with multiple parts, and there is large variability in the parts, how parts group into objects, and in the relationship between different objects and content types within the image. For example, in bio-medical imaging, structures such as Glomeruli, are defined by particular configurations of tissue and cell types that are common throughout the Kidney. In material science, particles of interest are often agglomerates of smaller sub-particles that form in specific ways under certain conditions. These analysis challenges are not unique to microscopy, but due to quantitative application-level objectives, microscopy often requires more detailed quantification products than in other domains. In the most general terms, microscopy often requires some, or all, of the following analysis products to be estimated:

1. Assignment of pixels to specific content (or object) categories to estimate areas and volumes.
2. Assignment of pixels to specific object instances to estimate size and shape distributions.
3. Allocation of (mixed) pixels to specific categories and instances to obtain more accurate estimates for 1 and 2.

Figure 1 illustrates some of these analysis products for a bio-medical application. These products are typically more detailed than what is required in computer vision applications that focus on object detection and recognition problems in consumer imagery. However many of the solution methods developed for these applications can be extended, or used as components. In this paper we focus on one small, but important component of many of the solution methods. We investigate a new framework for pooling lower-level features using data-adaptive pooling regions. Our framework is not specific to any particular set of lower-level features, or to specific ways of generating the data-adaptive pooling regions. However it is motivated and designed to support a hierarchical representation that has proven successful in the non-data-adaptive case [3]. In Section 2 we review the computer vision methods that we believe are most relevant to estimating

[*] {rporter, harve, ruggiero}@lanl.gov

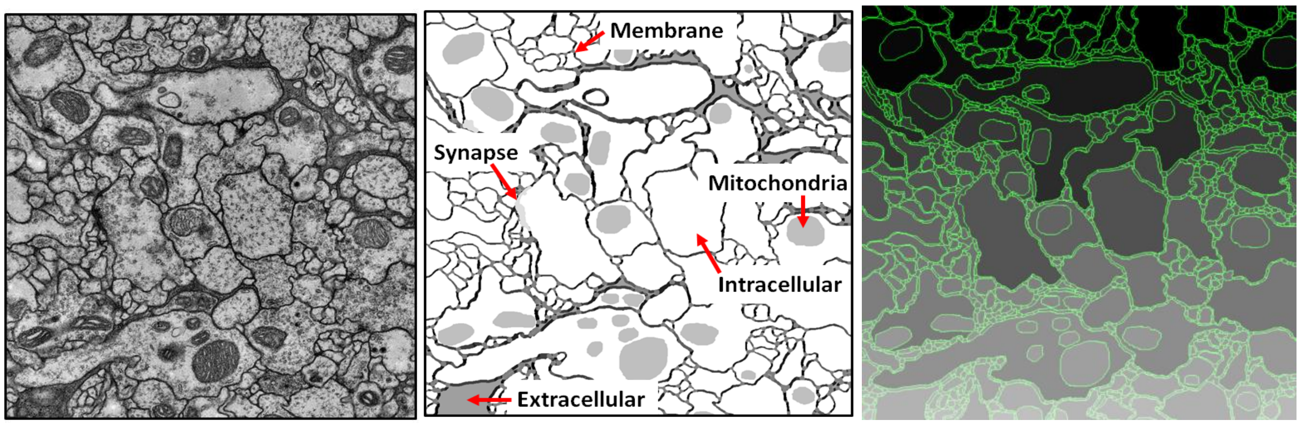


Fig. 1. Example imagery and data products from a publically available transmission electron microscopy (TEM) test-set that includes multiple serial sections of a Drosophila brain neuropile [1]. Left) TEM image Middle) Analysis product 1 – assignment of content labels (coded by gray-value) to pixels. Right) Analysis product 2 – Assignment of instance labels (coded by gray value) to pixels (green outlines delineate instances and are for visualization only).

the three analysis products described and motivate our approach. We present our new pooling framework in Section 3 and discuss how it is evaluated in Section 4. Experimental results are presented in Section 5 and we summarize and conclude in Section 6.

Another somewhat unique aspect of image quantification in microscopy is that complete automation is not always required. In fact, in many applications, experts are willing and able to interact with the quantification system both to focus the analysis on content that is most relevant, and to validate the quantification results. While this is not the focus of this paper, our pooling framework is designed (in part) to support the user interactions that we have observed most useful in a material science microscopy application [4].

2. BACKGROUND

In this section we outline some of the relevant problems and solutions that have been described in the computer vision literature, with particular emphasis on the methods that motivate our approach. Sections 2.1 and 2.2 are most relevant to analysis product 1. Section 2.3 is most relevant to analysis products 2 and 3.

2.1 What?

An image, or sub-image, is a collection of N pixels, $X = \{\ldots, x_i, x_j, x_k, x_l, \ldots\}$, which we assume to be real-valued, $x \in \mathbb{R}^D$. Pixels can have one (grayscale), three (color) or any number of dimensions (multi-, hyper-spectral). We define y as a discrete random variable that is used to identify the image content type, such as an object category e.g. face, cat, mitochondria etc. The number of distinct labels K could be two (to indicate object / no-object) or quite large depending on the application: $y \in \{1, 2, \ldots, K\}$. In figure 1, $K = 5$. The typical object detection problem in computer vision is to find the label y that minimizes an energy function:

$$y^* = argmin\ E(y, X) \tag{1}$$

Most approaches to this problem replace pixels with a feature descriptor (such as a filter bank response, or a SIFT descriptor [5]) calculated at each pixel location, or at a subset of locations (such as interest points [6]). Different approaches then use different pooling strategies for transforming the (potentially) variable number of features from the sub-image into a fixed length descriptor. Over small sub-images this might be as simple as calculating the mean or maximum of each feature dimension independently. Over larger sub-images a Bag of Words (BOW) representation is often more successful. BOW replaces each feature vector by a coded vector that represents the feature in terms of a fixed set of code-words (e.g. distance to cluster centers). Pooling then proceeds as before by calculating the mean or maximum of each dimension of the coded representation. In many approaches the pooled descriptor is fed into a classifier which performs the minimization in Equation 1.

One way these pooling approaches are extended is to define a hierarchy of pooling regions to better capture the spatial structure between features within the sub-image [7]. Descriptors from each region are typically concatenated to form a larger fixed length descriptor for the image. Using multiple pooling regions to characterize spatial structure is taken perhaps to the most extreme in HOG descriptors [8]. In this case the multiple regions are defined at one scale by tiling the sub-image with a regular grid of sub-sub-images. This leads to a more specialized (and less invariant) descriptor that has proven successful for detecting pedestrians and other relatively compact objects.

Another class of approaches that have proven to be successful in object detection are deep networks [3], [9]. At a high level, these networks implement similar pooling strategies to the previous approaches, but they implement the strategy multiple times, and the output from one stage is used as the input to the next stage. This means pooling is intertwined with feature descriptors and introduced incrementally through the hierarchy, which can potentially lead to a better balance between specificity and invariance.

2.2 Where?

So far we have discussed methods that can predict labels for an image or sub-image. One of the most common ways to locate the object is apply the sub-image detector as a sliding window across a larger image. This produces a set of random variables, $Y = \{\dots, y_i, y_j, y_k, y_l, \dots\}$, where each variable predicts the class label associated with a pixel location $y_i \in \{1, 2, \dots, K\}$. Post-processing steps (such as non-maximal suppression) are often applied to produce the final estimate for the location of the object. It is also common to estimate the bounding box for the object. More recently some of these post-processing steps have been incorporated directly into the detector using methods from structured output prediction [10].

More generally, structured output prediction methods are also being used for solving a problem known as semantic segmentation. The problem defined by semantic segmentation is to assign a label to each location in the image simultaneously by minimizing a 2nd-order energy function [11]:

$$E(Y) = \sum_i f(y_i,\ X) + \sum_{i,j} g(y_i, y_j, X) \tag{2}$$

Much like the methods of the previous section, the Y labels are used to encode content type such as object categories e.g. foreground, background, face, cat, etc. Unlike the previous methods, pairwise terms are used to model statistical dependencies between the labels which can improve detection performance (e.g. object category A is often found next to object category B) [12] as well as help localize objects by replacing bounding boxes with pixel-level masks [13].

We have presented semantic segmentation methods in terms of pixels, but the approach is more general. Pixels are often replaced by other entities such as super-pixels, points, voxels or object locations. In this paper we deal mostly with super-pixels, which can be arbitrarily shaped regions within the image. One extension of the semantic segmentation model introduces higher-order terms [14], [15]. Much like the deep networks of the previous section, another extension proposes hierarchical, or recursively defined, energy functions to better model long range and multi-scale dependencies that exist in image data [16], [17]. In this paper we focus on labeling super-pixels at a single scale, but our approach is partly motivated by future use in segmentation hierarchies.

2.3 How many?

Semantic segmentation provides a framework to integrate what and where, but many of the proposed approaches do not differentiate object instances. This is in contrast to classical image segmentation which (very generally speaking) minimizes a 2nd-order energy function without unary terms:

$$E(Y) = \sum_{i,j} g(y_i, y_j, X) \tag{3}$$

In this case the labels $y_i \in \{\ 1, 2, \dots, N\}$ do not correspond to specific object categories, but are used as generic segment (or cluster) identifiers. The number of distinct labels depends on the image, and can therefore vary from 1 through to the number of pixels N. Unary terms can also be introduced into classical segmentation methods, but in this case they serve as markers (or seeds) and are often used for interactive image segmentation [18].

Some semantic segmentation methods do estimate object instances as well as the object category. One method[*] returns a semantic segmentation along with the locations and bounding boxes for object instances [19]. Another method is called the Layout Consistent Random Field and it generalizes semantic segmentation to produce a pixel level prediction $y_i \in \{\ 1, 2, \dots, N * K\}$ which labels (category, instance) pairs [20].

[*] which inspired the “What”, “Where” and “How Many?” organization of this section

In computer vision, explicit modeling of object instances is often motivated by the need to deal with partially occluded objects. In image quantification applications, experts sometimes want to take this one step further and predict completions for partially occluded objects, so that object area estimates (for example) are more accurate. In the most general sense, this quantification requires multiple labels to be associated with each pixel, and this is not supported by most current models.

Predicting a set of labels for each example is known as multi-label classification and solutions have been developed for object labeling [21]. A more recent work has described a multi-label segmentation problem called super-segmentation [22] which the authors describe as generalizing classical segmentation. From our perspective, and in the language presented here, super-segmentation is most similar to multi-label semantic segmentation, since instances do not appear to be differentiated.

2.4 Integrating what, where and how many with feature pooling

Combing ideas and methods to better integrate "what", "where" and "how many" has fueled a large amount of research. "What" methods that operate on fixed, often rectangular sub-images are currently considered the state-of-the-art for detection. The "where" and "how many" methods define adaptive, data-driven sub-regions that are essential to delineation and other quantification tasks. Perhaps the easiest way to combine the two approaches is to use the "where" regions as pooling masks for the "what" features or predictions. This reduces a variable sized set of "what" descriptors into a fixed length region descriptor which can be used in a variety of different ways in subsequent processing. One potential advantage of this approach is that the region itself characterizes some of the shape information that may be relevant to an object category and this opens the door to a large body of research focused on shape characterization [23].

At one extreme, simple pooling functions, such as the average or the maximum, are applied to each feature dimension independently [24]. In this case the shape information is indirectly encoded. At the other extreme shape characterization features can be calculated on the region mask (or perimeter) explicitly, and used to augment the appearance features. Most relevant to this paper are efforts that attempt to bridge this gap. Carreira et. al. advance the state-of-the-art in two ways [25]. First, they proposes using second-order statistics to characterize dependencies between feature dimensions within a segment, and second, they suggest enhancing input features with spatial information that relates the feature location to the pooling region bounding box. Another approach, that is particularly relevant to this paper, is called Region-based Context Features (RCF) [26]. This approach defines weighted regions of support for multiple histograms based on the distance to the region perimeter. The approach presented in this paper has a very similar approach, but our motivation and design choices lead us to a general framework that can be used to pool features for both unary and pairwise terms.

3. SEGMENTATION BASED FEATURE POOLING

The basic idea we investigate in this paper is to use segments to define multiple pooling regions (referred to as sub-segments) instead of using segments to define a single pooling region. Figure 2 illustrates the idea and we refer to the sub-segments as the "boundary segment", the "inner segment" and the "edge segment". The sub-segments are defined by the geodesic distance between candidate pixel locations and the original segment boundary. This distance could be used to weight the contribution of pixels to the different sub-segment descriptors, but in this paper, we use a hard assignment. This means we can calculate the inner segment by simply eroding the original segment mask with a circular structuring element of width W. The boundary segment is then the difference between the dilated mask (using the same structuring element) and the inner segment. The edge segment is defined for each pair of neighboring segments, and is simply the intersection of the two boundary segments. Neighboring edge segments are defined by overlapping edge segments.

There are several motivations for the proposed sub-segments:

1. Informally, segments are often defined by two opposing forces: inner regions with relatively similar feature values, and boundaries which divide regions with relatively different feature values.
2. The boundary region provides a basis for standard shape based features, such as the perimeter, and aspect ratio.
3. The boundary region provides a convenient way to define edge segments which can define features for the pairwise terms in 2nd-order energy functions.

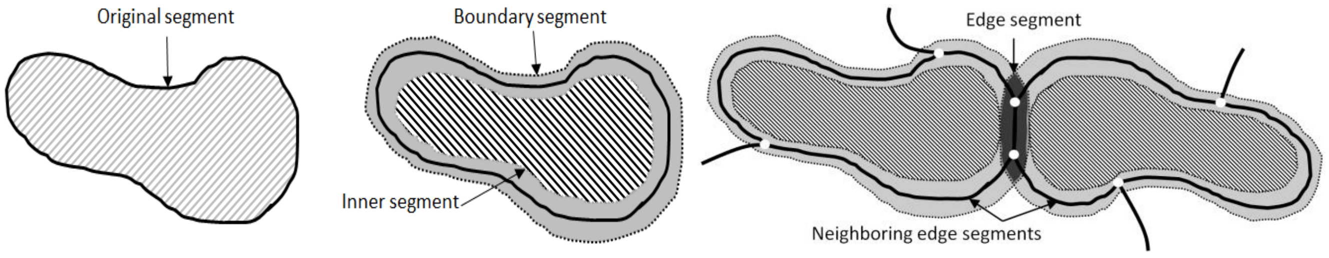


Fig. 2. Left) A segment used for data-adaptive pooling is divided into two main pooling regions called the boundary and inner segments.

Right) Overlapping boundary segments define edge segments and an edge graph for deriving pairwise features.

4. The sub-segments are easily combined into larger segments. For example pooled features for the combined boundary can be estimated by pooling the two boundary segments and removing the twice counted edge segment.

There are many ways the proposed sub-segments could be utilized for feature pooling. In this paper we perform experiments with histograms of pixel intensities and gradient orientations. The details of this implementation are described next.

3.1 Segmentation-based Histogram of Orientated Gradients (SHOG)

We use pixel intensity and gradient orientation as the lower-level features. These features are accumulated into histograms using the segmentation based pooling regions. Our implementation of the gradient histograms follows the one used for HOG descriptors [8]. There are 9 bins over the range $0^{\circ} - 180^{\circ}$ (the sign of the gradient orientation is ignored). Pixel contributions are weighted by the gradient magnitude and linearly interpolated between the closest two bins.

Fig. 3 visualizes these histograms for a number of synthetic images. In these cases it can be seen that the boundary, edge and combined boundary sub-segments provide fairly good characterization of basic shape attributes.

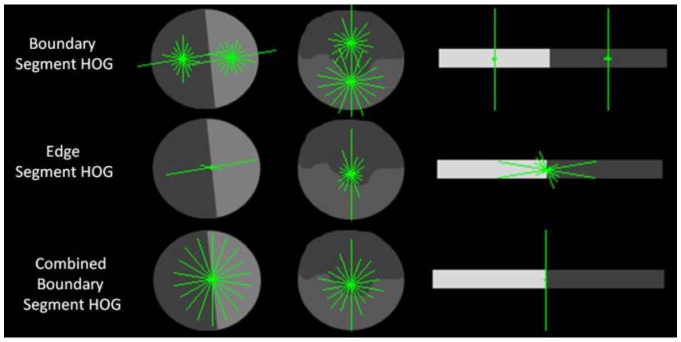


Fig. 3. Example of SHOG pooling using the boundary, edge and combined boundary sub-segments. Green lines indicate relative counts in the 9 gradient orientation bins.

3.2 Normalization

We note that multiple normalizations of the pooled regions with respect to neighbors proved to be a key design choice with the HOG descriptors [8]. In this paper we investigate a number of different approaches for normalization of sub-segment histograms. For a B bin histogram $\vec{h}$ where bin b has contribution $h(b)$, we divide each bin by the total contribution: $\vec{n} = \vec{h} / N_h$ where $N_h = \sum_{b=1:B} h(b)$. We use the normalized histogram counts as well as the normalization constant as features for the segment: $s = [\, N_h \; \vec{n}\,]$. As will be discussed in Section 3.4 we maintain the histogram in subsequent processing to make use of histogram distance functions which means we think of this concatenated vector as a 2-dimensional feature space[*]. When we calculate this feature space using the original segment, it is called **Single** in Section 5.

The most straightforward normalization multi-segment pooling is to normalize each sub-segment independently. This leads to a 4-dimensional feature space for describing each segment, where we simply concatenate normalized features from the inner and boundary segments, and we call this the **Split** strategy in our experiments. An alternative is to jointly normalize the inner and boundary segments. This means we concatenate the histograms prior to normalization which

[*] Note that we perform the same normalization for both gradient and intensity histograms throughout, which doubles the final feature vector dimensions with respect to numbers discussed here.

doubles the number of histogram bins, but reduces the number of features in the final representation back to 2-dimensions. In our experiments we call this the **Split-Joint** strategy.

3.3 Neighborhoods

Segments define spatial regions of support for features and, in an ideal case, encapsulate the spatial information required in subsequent processing. However, it is unlikely that segments are exact and in many cases neighborhood information is required to help inform local decisions for each segment. For example, in convolutional networks, the spatial support for features is typically larger than the subsequent pooling stage. Neighborhoods can also be defined on segmentations using connectivity in the image plane, however, the number of neighbors varies from segment to segment. To incorporate neighborhood information and still generate a fixed length feature vector we propose a cross-histogram (or competitive) normalization strategy. Specifically, if $\overrightarrow{h_i}$ is the histogram of the i^{th} segment, and $L = \{\dots, h, i, j, \dots\}$ are the segments associated within a local neighborhood of i then we normalize each bin, $l(b) = h(b)/\, N_b$, where $N_b = \sum_{s \in L} h_s(b)$. We then normalize $\overrightarrow{l_i}$ as before, so that the integrated histogram sums to 1 and we again have a 2-dimensional feature vector. This strategy is indicated with **Neighbor** modifiers in out experiments. Note, in Figure 2 we also defined a neighborhood for the edge segments, and can therefore use the same approach to incorporate neighborhood information into the edge segment descriptors.

3.4 Feature Space Expansion

In our experiments we use a linear classifier to evaluate the utility of the various pooling strategies. Instead of feeding the pooled features directly into the classifier, we expand the representation using exemplars. This could be done with coding strategies similar to BOW representations, or it could be done implicitly with a Kernel classifier. For ease of implementation, our expansion was explicit and similar to Fisher Vectors [27]. Given a training set with (segment feature vector, label) pairs, $\{(s(1), y(1)), (s(2), y(2)), \dots, (s(N), y(1))\}$, we randomly select K exemplars. The expanded descriptor for a segment is then the distance between each of its feature dimensions with the corresponding dimension of each exemplar. If the original segment descriptor had D dimensions then the expanded descriptor has $K * D$ dimensions. Note, we treat histograms as a single dimension and we use a Chi-Square distance function to calculate the distance.

4. EVALUATION FRAMEWORK

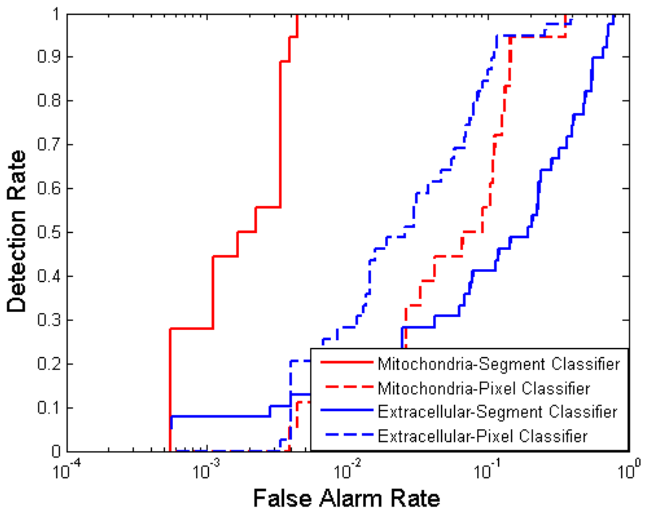
Ideally we would evaluate the proposed approach in terms of the final application objectives, and in Section 2 we briefly described some of the application-level models that are relevant. Pooled features are incorporated into these models through the unary and pairwise terms in Equation 2. These terms are chosen for the application, but they are typically parameterized so that they can be tailored to the application using data. The most common parameterization for these functions is linear:

$$f(y_i, X) = \overrightarrow{W_u}.\overrightarrow{\Phi}_u(y_i, X) \qquad\qquad g(y_i, y_j, X) = \overrightarrow{W_p}.\overrightarrow{\Phi}_p(y_i, y_j, X) \tag{4}$$

Generally speaking, learning the weights from training data is a challenging problem, and many additional design choices that must be made with respect to inference and learning that affect application-level performance [28]. For this reason, we use simpler, surrogate learning problems to evaluate our approach, and consider each term in turn.

4.1 Learning to Label Segments

In the first set of experiments, we test the impact of different pooling strategies on labeling segments independently. The segments are defined by the ground-truth segmentation shown on the right in Figure 1. The task is to design a classifier to predict the semantic labels shown in the middle of Figure 1. This experiment is in some sense, the best case scenario for segmentation based pooling. Objects of interest are perfectly delineated and therefore, in principle, the classifier has all the shape information it might need available to it. These experiments are clearly artificial (perfect segmentations are rarely available) but they do illuminate design choices. Note also that even with perfect segments, the utility of segmentation based pooling depends on the application and content of interest. On the left in Figure 4, pooling gradient orientations and intensity over the original segment (**Single**) provides a significant advantage for features with relatively well defined shape characteristics (the circularity of mitochondria) compared to a pixel-level classifier (the GENIE pixel classifier [29]). However for more amorphous content, such as the extracellular regions, the segmentation based pooling appears to make the problem more difficult compared to the pixel-level problem.



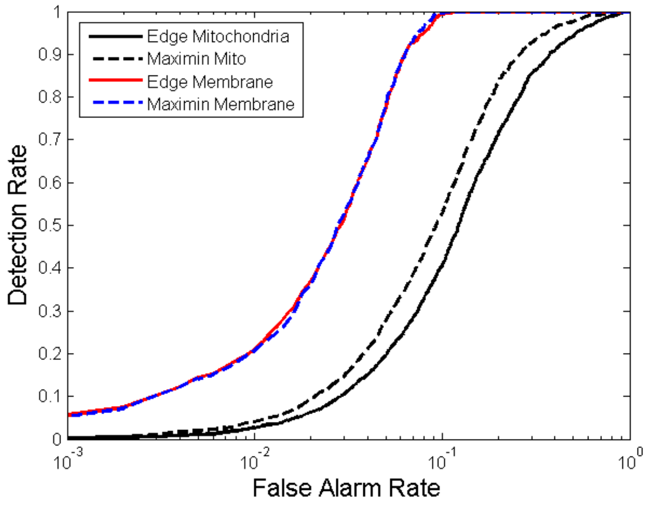


Fig. 4. Left) Segment labeling performance for two different features using a segment classifier (solid line) and a pixel classifier (dashed line) that classifies segments with a majority vote.

Right) Segmentation performance for two different features using a pairwise edge classifier (solid line) and a maximin classifier (dashed line).

4.2 Learning to Merge Segments

In the second set of experiments we evaluate the impact of pooling strategies on learning pairwise terms. We start with an over-segmentation of the feature of interest, and the task is to learn an affinity function that groups (or merges) segments into larger components. In previous work, we called this problem learning to merge [30]. We use the Drosphillia TEM test-set in these experiments as well. One test problem focuses on the membrane feature, and it is shown on the left in Figure 5. In the ground-truth provided for this feature, there are in fact 5 sub-categories of membrane defined – four different orientations and one category for "junctions". These sub-categories define a natural over-segmentation for the larger "membrane" feature. Figure 5 illustrates the over-segmentation in green and the red highlight shows a subset of the segments that have been identified as belonging to the larger class, and hence should be merged.

For other features in the test-set, we do not have over-segmentations, and so we generate them ourselves. The next three images in Figure 5 show 3 different segmentations for the mitochondria feature at various levels of over-segmentation. These segmentations were generated using a hierarchical watershed algorithm at 3 different segmentation levels. We intersected the watershed segmentation with the ground truth segmentation to produce the final over-segmentation. This means that the over-segmentation does not bleed into the background as we move up the hierarchy.

To evaluate the pooling strategies on these problems, we learn a pairwise classifier to predict a binary label $y_{i,j} \in \{+1,-1\}$ for each pair of neighboring segments (i,j). The positive examples are generated from pairs of segments that both belong to the same instance of the feature of interest (e.g. yellow edges on the right in Figure 5). Negative examples are generated from the pairs of segments that include one segment from the feature instance and the other segment from a different feature or different instance.

Pairwise classifiers are sometimes a poor substitute for minimizing 2nd-order energy functions since pairs can be highly dependent. We therefore experimented with a maximin classifier [31] which includes a connected component inference step as part of the learning process. As shown on the right in Figure 4 we observed some variability between the pairwise and maximin classifier performance depending on the feature of interest. We suggest this variability is explained largely by the different in connectedness between the two features. However we did not observe any significant difference in the relative performance of different pooling strategies with the two different objectives, and therefore for the remainder of the paper we report results from the pairwise classifier.

4.3 Fixed Features

In previous work, we developed a number of more traditional shape features that were of particular interest in a material science application. For the most part these shape features can be calculated on the binary mask defined by the segment. We also calculated a number of pairwise features based on a vectorized representation of the segmentation edges.

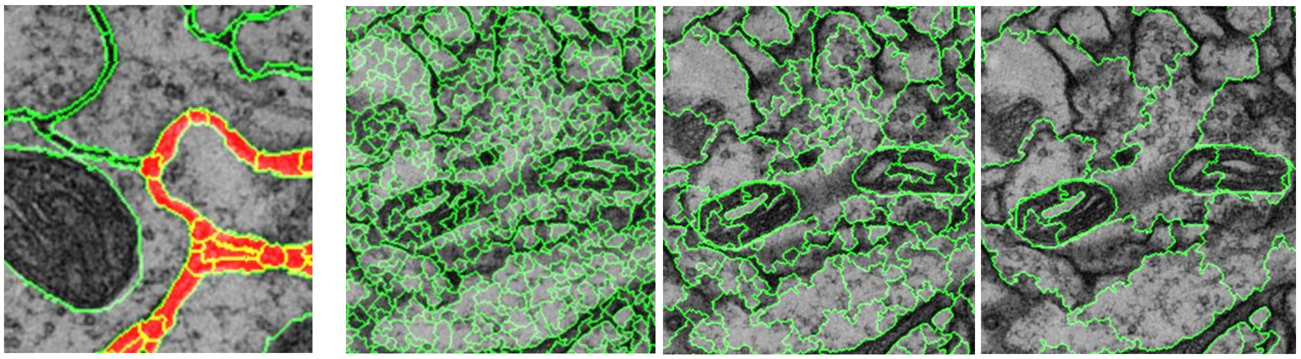


Fig. 5. Experiments to evaluate pooling strategies for pairwise features include Left) Merging sub-categories of the membrane feature and from Left to Right) **Base**, **Quarter** and **Half** - level segmentations produced for the Mitochondria feature.

Although far from exhaustive, we use these shape features as a representative example of the more traditional approach to shape characterization, and refer to them as **Fixed Features** in our experiments. The two sets of features used are:

- **Segment features:** Area, Perimeter, Circularity, Convex Hull Area, Convex Hull Perimeter, Area Convexity (ratio of area and convex hull area), Perimeter Convexity (ratio of perimeter to convex hull perimeter), Aspect Ratio, Mean Intensity.
- **Edge (pairwise) features:** Average Area, Difference in Area, Average Perimeter, Difference in Perimeter, Average Mean Intensity, Difference in Mean Intensity, Edge Length, Edge Straightness, Edge Connection Angle (Average angle between edge and neighboring edges), Distance (between segment centroids).

5. EXPERIMENTAL RESULTS

5.1 Learning to Label Segments

We implemented 4 different experiments with each of the 4 main features (mitochondria, membrane, intracellular and extracellular). In each experiment we train a segment-level classifier which predicts a binary label $y \in \{+1, -1\}$ for each segment. We use one image for training (e.g. image 00 in stack 1), and one image for evaluating performance (e.g. image 10 in stack 1). We repeat this experiment 10 times (there are a total of 20 images in the annotated image stack) and average the ROC curves over the 10 trials to produce Figure 6. For each trial we use the $L1$ regularized logistic regression classifier provided in LibLinear [32] as the classifier, although we did not observe as significant difference in performance to the standard linear SVM formulation. We chose the regularization parameter with 5-fold cross-validation and then retrained with the entire training image. We use a fixed value $W = 1$ which means the boundary region is approximately two-pixels wide. We use $K = 256$ exemplars, however the number of training examples (and hence the number of exemplars) is often less than this.

We make a number of observations about the results in Figure 6. For 3 of the 4 experiments, splitting the pooling region into boundary and inner segments (**Split**) leads to better performance than using a single pooled region (**Single**), but joint normalizing the two segments (**Split-Joint**) leads to better performance than the single pooled region for all 4 experiments. We observe also that joint-normalizing the two segments generally outperforms the independent normalization strategy, and for Mitochondria this is particularly pronounced.

Adding the additional neighborhood normalized histograms improved performance in all cases, and often significantly. We suggest this improvement is particularly large in these experiments since segment labels are highly dependent on their context (e.g. Mitochondria appear inside Intracellular objects) and the ground truth segment sizes and shape vary greatly depending on the semantic labels (e.g. Mitochondria are relatively small and compact and Intracellular are large and irregular).

The performance of the **Fixed Features** varies greatly over the four experiments: it has the best performance for mitochondria, but amongst the worst performance for other features. We suggest this is because the fixed features represent various different ways of quantifying the circularity of segments – a characteristic of particular interest to our

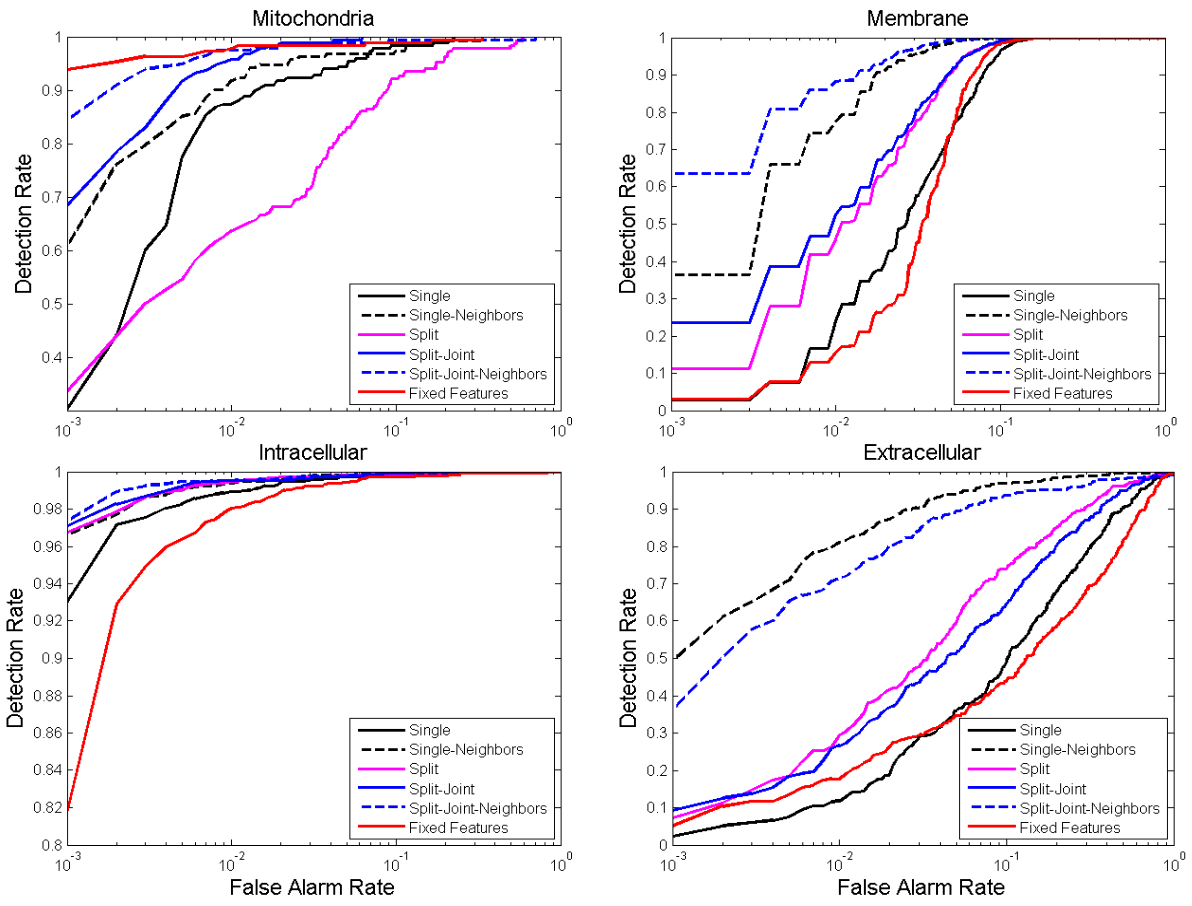


Fig. 6. Segment label performance for the various pooling strategies for the four main features of interest in Figure 1.

material scientists – and as a result it performs particularly well for mitochondria.

5.2 Learning to Merge Segments

For the merge experiments we use the same experimental parameters as the previous section, and the results are summarized in Figure 7. Unlike the labeling experiments, using multiple pooling regions (boundary and inner segments) for the combined features did not significantly improve the performance over using combined features with a single region. However, adding the additional **Edge Segment** features (Joint/Edge) appears to consistently improve performance. As before, adding the neighborhood normalized features appears to improve performance. As might be expected, this was most significant for the membrane problem, where some fraction of the edges to be merged have a specific geometric relationship to neighboring edges (i.e. short sides of a rectangle). We also observe that the fixed features are again highly variable, with some of the best, and some of the worst, results across experiments.

6. SUMMARY

We have presented a novel framework to pool lower-level features using a data-driven segmentation. The framework breaks each segment into a number of sub-segments and uses them in various combinations to generate pooled features for unary and pairwise energy functions. For unary features, our experiments indicate that joint normalization of sub-segments leads to consistently better performance compared to using the original segment, while maintaining the same descriptor length. We also proposed a cross-histogram (competitive) normalization scheme for incorporating information from variable sized neighborhoods into the pooled representation. This neighborhood information also consistently improved performance. For pairwise features, our experiments indicate that adding dedicated features for the edge segment improved the performance compared to just combing unary features. For both unary and pairwise terms, the performance of the proposed approach is competitive with hand crafted features on problems for which the

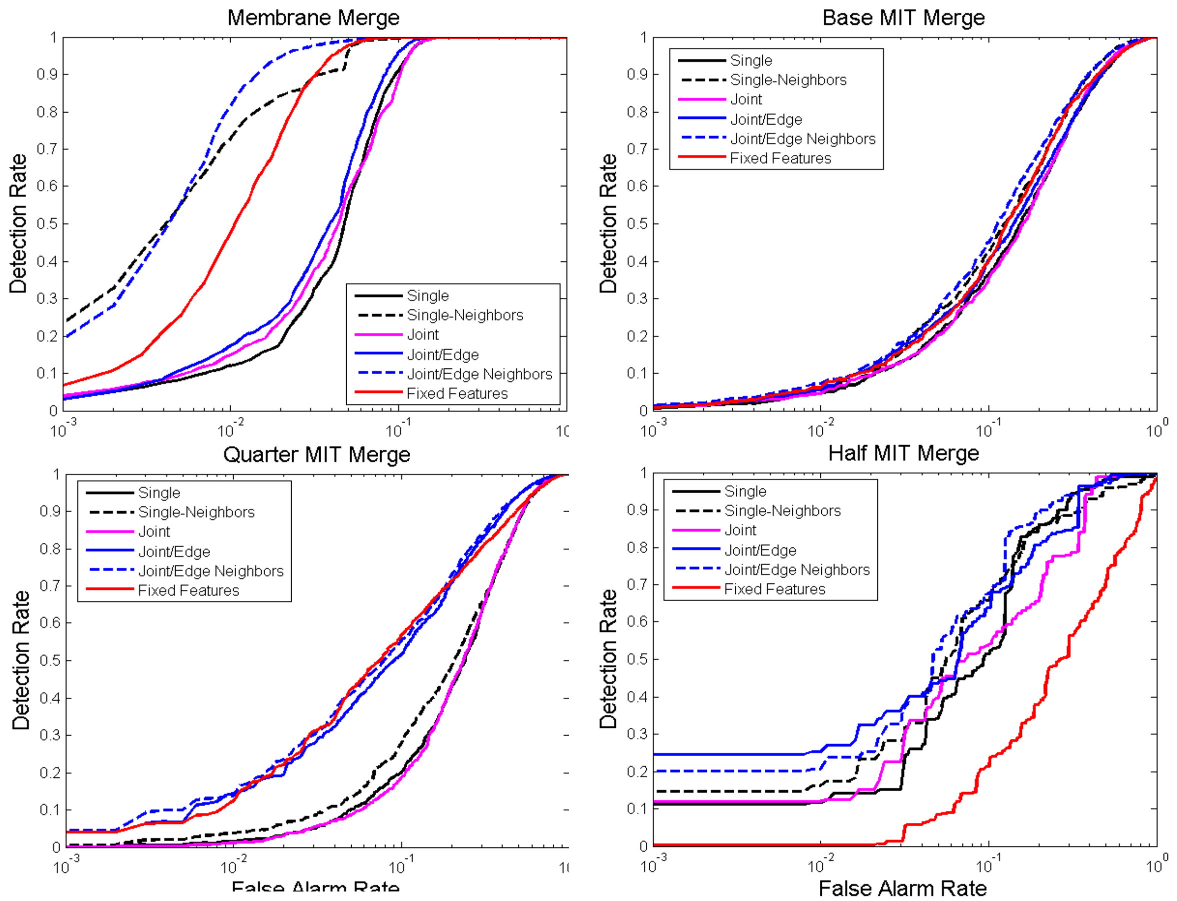


Fig. 7. Pairwise classifier performance for the various pooling strategies for the four experiments shown in Figure 5. Top-left is the sub-category membrane merge problem defined by the ground-truth. Base, Quarter and Half MIT represent decreasing levels of over-segmentation for the mitochondria experiment.

hand crafted features appear ideally suited, and significantly better on other problems. This indicates that the proposed framework may provide a general purpose solution for a wide range of applications.

In conclusion, we have presented a general purpose strategy for characterizing and integrating the pooling region shape information with the information characterized in lower-level features. In future work we plan to use this framework to investigate hierarchical learning of both features and pooling regions to better address image quantification needs in microscopy. We also plan to investigate how user-interaction can help in the design and learning of such hierarchies

REFERENCES

1. Cardona, A., et al., *An integrated micro- and macroarchitectural analysis of the Drosophila brain by computer-assisted serial section electron microscopy.* PLoS Biol, 2010. **8**(10).
2. Jain, V., H.S. Seung, and S.C. Turaga, *Machines that learn to segment images: a crucial technology for connectomics.* Curr Opin Neurobiol, 2010. **20**(5): p. 653-66.
3. LeCun, Y., et al., *Gradient-based learning applied to document recognition.* Proceedings of the IEEE, 1998. **86**(11): p. 2278-2324.
4. Porter, R., et al. *Interactive image quantification tools in nuclear material forensics*. 2011.
5. Lowe, D.G. *Object recognition from local scale-invariant features*. in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. 1999.
6. Rosten, E., R. Porter, and T. Drummond, *Faster and better: A machine learning approach to corner detection.* IEEE Trans. Pattern Analysis and Machine Intelligence (to appear), 2009.

7. Lazebnik, S., C. Schmid, and J. Ponce. *Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories*. in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. 2006.
8. Dalal, N. and B. Triggs. *Histograms of oriented gradients for human detection*. in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. 2005.
9. Simonyan, K., A. Vedaldi, and A. Zisserman, *Deep Fisher Networks for Large-Scale Image Classification,* in *Advances in Neural Information Processing Systems 26*2013. p. 163-171.
10. Blaschko, M. and C. Lampert, *Learning to Localize Objects with Structured Output Regression*, in *Computer Vision – ECCV 2008*, D. Forsyth, P. Torr, and A. Zisserman, Editors. 2008, Springer Berlin Heidelberg. p. 2-15.
11. Kumar, S. and M. Hebert. *Discriminative random fields: a discriminative framework for contextual interaction in classification*. in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. 2003.
12. Lafferty, J.D., A. McCallum, and F.C.N. Pereira, *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*, in *Proceedings of the Eighteenth International Conference on Machine Learning*2001, Morgan Kaufmann Publishers Inc. p. 282-289.
13. Larlus, D. and F. Jurie. *Combining appearance models and Markov Random Fields for category level object segmentation*. in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. 2008.
14. Boix, X., et al., *Harmony Potentials.* International Journal of Computer Vision, 2012. **96**(1): p. 83-102.
15. Kohli, P., M.P. Kumar, and P.H.S. Torr. *P3 & Beyond: Solving Energies with Higher Order Cliques*. in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*. 2007.
16. Tu, Z., et al., *Image Parsing: Unifying Segmentation, Detection, and Recognition.* International Journal of Computer Vision, 2005. **63**(2): p. 113-140.
17. Corso, J.J., A. Yuille, and Z. Tu. *Graph-Shifts: Natural Image Labeling by Dynamic Hierarchical Computing*. in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2008.
18. Couprie, C., et al., *Power Watershed: A Unifying Graph-Based Optimization Framework.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2011. **33**(7): p. 1384-1399.
19. Ladický, Ľ., et al., *What, Where and How Many? Combining Object Detectors and CRFs*, in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Editors. 2010, Springer Berlin Heidelberg. p. 424-437.
20. Winn, J. and J. Shotton. *The Layout Consistent Random Field for Recognizing and Segmenting Partially Occluded Objects*. in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. 2006.
21. Boutell, M.R., et al., *Learning multi-label scene classification.* Pattern Recognition, 2004. **37**(9): p. 1757-1771.
22. Marques, J.S. and M.A.T. Figueiredo. *Image super-segmentation: Segmentation with multiple labels from shuffled observations*. in *Image Processing (ICIP), 2011 18th IEEE International Conference on*. 2011.
23. Yang, M., K. Kpalma, and J. Ronsin, *A survey of shape feature extraction techniques.* Pattern Recognition, 2008: p. 43-90.
24. Farabet, C., et al., *Learning Hierarchical Features for Scene Labeling.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2013. **35**(8): p. 1915-1929.
25. Carreira, J., et al., *Semantic segmentation with second-order pooling*, in *Proceedings of the 12th European conference on Computer Vision - Volume Part VII*2012, Springer-Verlag: Florence, Italy. p. 430-443.
26. Pantofaru, C., et al. *Combining Regions and Patches for Object Class Localization*. in *The Beyond Patches Workshop in conjunction with the IEEE conference on Computer Vision and Pattern Recognition*. 2006.
27. Perronnin, F., J. Sánchez, and T. Mensink, *Improving the Fisher Kernel for Large-Scale Image Classification*, in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Editors. 2010, Springer Berlin Heidelberg. p. 143-156.
28. Nowozin, S. and C.H. Lampert, *Structured Learning and Prediction in Computer Vision.* Foundations and Trends® in Computer Graphics and Vision, 2011. **6**(3–4): p. 185-365.
29. Harvey, N.R., et al., *Comparison of GENIE and conventional supervised classifiers for multispectral image feature extraction.* Geoscience and Remote Sensing, IEEE Transactions on, 2002. **40**(2): p. 393-404.
30. Porter, R.B., S. Lundquist, and C. Ruggiero, *Learning to merge: a new tool for interactive mapping.* Proc. SPIE, 2013: p. 87431F-87431F.
31. Turaga, S.C., et al. *Maximin affinity learning of image segmentation*. in *NIPS*. 2009.
32. Fan, R.-E., et al., *LIBLINEAR: A Library for Large Linear Classification.* Journal of Machine Learning Research, 2008. **9**: p. 1871-1874.