PASCAL Boundaries: A Class-Agnostic Semantic Boundary Dataset

Vittal Premachandran      Boyan Bonev      Alan L. Yuille
University of California, Los Angeles,
{vittalp@, bonev@, yuille@stat.}ucla.edu

Abstract

In this paper, we address the boundary detection task motivated by the ambiguities in current definition of edge detection. To this end, we generate a large database consisting of more than 10k images (which is $20\times$ bigger than existing edge detection databases) along with ground truth boundaries between 459 semantic classes including both foreground objects and different types of background, and call it the PASCAL Boundaries dataset, which will be released to the community. In addition, we propose a novel deep network-based multi-scale semantic boundary detector and name it Multi-scale Deep Semantic Boundary Detector (M-DSBD). We provide baselines using models that were trained on edge detection and show that they transfer reasonably to the task of boundary detection. Finally, we point to various important research problems that this dataset can be used for.

1. Introduction

Edge detection has been a fundamental problem in computer vision since the 1970’s [10]. Detecting edges is beneficial for many vision tasks, for example, object detection [30], image segmentation [1], neural circuit reconstruction from brain images [6], and autonomous navigation, among others. This problem is under active research and potential solutions include local filtering-based approaches like the Canny edge detector [5] and the zero-crossing algorithm [26], to pixel-level classification methods that use features obtained by careful manual design like $gPb$ [1], to patch-based clustering algorithms such as Structured Edges (SE) [9], to the more-recent deep learning based approaches such as the N4-network [6] or HED [28].

However, edge detection is an ambiguous task making it difficult to evaluate. There is no clear answer to the question, ‘What is an edge?’ An accepted definition of an edge is those sets of pixels with strong gradients. Existing edge detection databases such as the BSDS300 [21] and BSDS500 [1] were generated by asking the annotators to divide the image into multiple segments resulting in different annotators dividing the images into different segments. This lack of consistency arises because of the fact that edges can occur at different levels of granularity; i) just the exterior boundaries of objects, which divide the image into different object instances (car, road, tree, etc.), ii) interior boundaries dividing an object into its constituent parts (head, neck, torso, etc.), or iii) non-semantic contours emanating from texture (stripes on a tiger) or artificial design (writings on clothes). Hou et al. [13] discuss the ambiguities in these datasets in more detail. In addition to the ambiguity, the BSDS500 dataset has only 500 images and cannot be considered as a large database. This motivates us to construct a new, large, class-agnostic semantic boundary dataset, that is not only large in comparison to BSDS500, but is also without the ambiguity of edge detection.

In this work, we wish to eliminate this ambiguity by restricting ourselves to the coarsest level of granularity, i.e., semantic instance-level object boundaries. Further extensions of our work is possible by introducing edges of other levels of granularity. Figure 1 shows the differences between the BSDS500 annotations (top row) and our annota-
This paper makes the following contributions: i) We define a precise task, namely, boundary detection. To enable progress on this problem, we construct a large dataset with \(\sim 10k\) images taken from the PASCAL VOC2010 challenge and provide boundary annotations between 459 semantic classes including both foreground objects and different types of background. These boundary annotations will be released publicly. The dataset generation process is described in more detail in Section 3. ii) We propose a novel multi-scale deep network-based class-agnostic semantic boundary detector (M-DSBD) to solve the boundary detection task. This is described in detail in Section 4.

The rest of the paper is organized as follows. In Section 5, we provide baselines on this new dataset using two well-performing edge detectors, i.e., Structured Edges (SE) [9] and Holistically-Nested edge detector (HED) [28]. We also provide results obtained using M-DSBD, and its single-scale counterpart DSBD, using existing evaluation methodologies like the F-measure to enable fair comparisons with current and future approaches. Finally, we conclude the paper in Section 6 by pointing to various future directions that this dataset paves way to, which would enable progress in many computer vision problems.

2. Related Work

One of the first databases for edge detection was the Sowerby database [4], which consisted of 100 color images from the streets of Sowerby. At about the same time, there was the South Florida dataset [4], which was simpler than the Sowerby dataset, and consisted of 50 gray scale images. These datasets enabled the identification of the fact that low-level filtering techniques such as the Canny edge detector [5] were limited in many ways. Moreover, the first real statistical techniques for edge detection [14, 15] were developed and tested on these datasets.

The Sowerby dataset being too small motivated Martin et al. [20] to start creating a public dataset of image segmentations. A set of 300 images from this dataset was then used in [21], who cast the problem of edge detection as a per-pixel classification problem and evaluated their results using a precision-recall curve, which can be summarized by the now-standard F-Measure. This set of 300 images later came to be known as the BSDS300 dataset. The BSDS300 dataset enabled the development of some notable edge detectors such as BEL [8].

Recently, the BSDS300 dataset was extended to incorporate 200 additional images [1] and the new superset dataset was named as the BSDS500 dataset. The BSDS500 dataset has since been heavily worked upon producing significant efforts on edge detection algorithms such as the gPb-edge detector [1], Sketch Tokens [17], SE [9] and, the now state-of-the-art method, HED [28]. Over the last couple of years, many deep network-based edge detection methods, such as the N4-network [6], Deep Edge [3], Deep Contour [24] and HED [28], have shown significant improvements in terms of the F-Measure on the BSDS500 dataset. While these algorithmic improvements are welcome, we feel that further jumps in performance will be limited by the size of the BSDS500 dataset.

The issue with regards to the scale of the BSDS500 dataset and its ambiguity was addressed to a certain extent (though not consciously) by Hariharan et al. [11]. They built an instance-level segmentation dataset for the 20 PASCAL object categories in about 10,000 images corresponding to the trainval set of the PASCAL VOC2010 challenge. However, their aim in building this dataset was to tackle the problem of obtaining class-specific object boundaries, thus requiring them to train \(O(N)\) boundary detectors corresponding to \(N\) object classes. Uijlings and Ferrari [27] go even more extreme by subdividing each object class into \(K\) subclasses, which they call “situations”, thus requiring them to train \(O(NK)\) situational object boundary detectors. Clearly, neither of these approaches are scalable for large \(N\) and \(K\). Therefore, we adopt a class-agnostic boundary detection strategy by going back to building a strong monolithic boundary detector, thus requiring us to train a \(O(1)\) boundary detector. Such an approach allows for the sharing of the computation involved in performing the mid-level task across various high-level vision tasks. Sharing of mid-level computations enables seamless scalability across multiple high-level visual tasks.

A similar dataset is the MS-COCO dataset [18], which contains instance-level masks for 80 object categories and is much larger than the PASCAL dataset. However, this dataset also contains masks only for foreground objects. In comparison, we consider all objects in the image and an initial taxonomy established 459 semantic classes that includes both foreground objects and different types of backgrounds (e.g. sky, water, grass, etc.). Finally, Zhu et al. [29] recently proposed an amodal segmentation dataset, where they label the complete extents of an object (even if they are occluded) on the 500 images of the BSDS500 dataset. We restrict ourselves to the unoccluded parts of the object since labeling the occluded regions of objects would again lead to ambiguity in the task.

3. Dataset Description

We propose a new dataset for boundary detection tasks and call it PASCAL Boundaries. It is fundamentally different from the well-known BSDS500 [1] dataset. BSDS500 allows annotators to divide the images into multiple segments without providing a precise definition of an edge. The annotations thus consists of edges from multiple levels of segmentation hierarchy.

PASCAL Boundaries follows a different approach. The
labels are obtained using clear instructions and unequivocal criteria, so that there are no disagreements. We restrict the labels to be only those edges that separate one object instance from another object instance of the same class, or another object instance from a different class, or, from a background type. This ensures that our boundaries are consistent since the visible extent of a class is well-defined.

The annotators were asked to label all the pixels belonging to the same object (without sub-pixel precision). This annotation produced the PASCAL Context [22] annotated dataset, which uses the images of the PASCAL VOC2010 challenge (10,103 trainval images). To minimize human errors, the images were reviewed two times by different subjects. The boundary annotations in the proposed PASCAL Boundaries dataset are obtained by an automatic post-processing of the PASCAL Context region-based annotations. The boundaries are localized exactly between pixels having different category, or instance, labels. Ideally, they would be 0 width: right between the objects. In practice, we label the boundaries of both objects, which produces two pixel wide boundary annotations. This can be useful for some setups, but in our experiments we thinned them using morphological operations to have boundaries of one pixel width. We do not use sub-pixel precision in our annotations because we found that annotating at such levels of precision would be beyond the abilities of human annotators. Rows 1 and 3 in Figure 5 shows multiple examples of image-boundary pairs. Row 1 contains the original images, row 3 is the class-agnostic semantic boundary map that we obtain from the PASCAL Context annotations (shown in row 2 of Figure 5).

Thus, PASCAL Boundaries is the first dataset which comprehensively annotates unoccluded image boundaries with an unequivocal criterion. Many of the images in this dataset are non-iconic, which means they contain multiple objects, not necessarily biased towards “photography” images (one salient object in the center with high contrast with respect to the background). Minimizing this kind of bias is beneficial for realistic computer vision applications. BSDS500, on the other hand, consists of images with a dominant foreground object and without much clutter in the images. We also emphasize that the number of images in the PASCAL Boundaries dataset (~ 10k) is much larger than in existing datasets. The increased scale of the dataset provides more variation in the boundary types and is beneficial for learning deep models. Moreover evaluations on ~ 5k images ensures a stricter test than evaluations performed on just a couple hundred images.

Dataset Statistics: PASCAL Boundaries has images of 360×496 pixels on average, from which an average of 1.45% of pixels are annotated as boundaries. This percentage is slightly lower than the 1.81% of pixels annotated as edges in BSDS500, on images of 321×481 pixels size. This is understandable since the BSDS annotations consisted of edges from the interiors of objects. This number drops to 0.91% if we consider only those pixels that were labeled by all the annotators annotating the image.

Extensions: Many extensions of this dataset are possible. It is easy to annotate junctions in the image, i.e., regions in the image where there is a confluence of more than two contiguous objects. In some types of junctions, for example, in the case of T-junctions, these boundary confluences could act as cues for occlusion.

Another extension that we believe is useful is using the PASCAL Context class information in conjunction with the boundary information. In this way, the local appearances of boundaries can be analyzed and clustered based on pairs of classes on either side of a boundary. In Fig. 2 we show the most common shared boundaries, classified by pairs of categories, and sorted by boundary length. Note that the boundary length is influenced by the size of the regions in the image, not only by their number of instances.

4. Multi-scale Deep Semantic Boundary Detector (M-DSBD)

To complement the PASCAL Boundaries database, we propose a novel multi-scale deep network-based semantic boundary detector (M-DSBD). As an overview, our network takes an RGB image as input and outputs a prediction map that provides the confidence for the presence of a class-agnostic object boundary at each pixel location. To this end, we build upon the fully convolutional network (FCN) architecture [19].

Our network architecture is shown in Figure 3. M-DSBD works on multiple scales of input images, which is a common practice in many computer vision algorithms. Since the objects in our images occur at different scales, we try to provide invariance to it by explicitly detecting object boundaries at various scales during both the training and testing phases. Note that this is different from HED [28], where the authors use multi-scale only while training the network. Combining the predictions from multiple scales of the same image allows the deep network model to be scale-invariant.
max-pooling layers. We use convolutional layers, rectified linear unit (ReLU) layers, and by passing the rescaled image, class-agnostic boundary map for each scale, \( \hat{x} \) is the rescaled image in this image pyramid, \( x \) boundary map, \( y \) is the linear combination weight associated with \( s \), which can be vectorized and written as \( w_{\text{scale}} \), \( \sigma(\cdot) \) is used to denote the sigmoid function that maps the boundary prediction activations into the range \([0,1]\). The final boundary prediction, \( \hat{y} \), involves taking a linear combination of the scale-specific boundary prediction activations, \( \hat{y} \),

\[
\hat{y}(i) = \sigma\left( \sum_{s=1}^{\vert S \vert} w_{\text{scale}}^s \hat{y}^s_a(i) \right). \tag{1}
\]

where, \( i \) is used to index the pixel locations in the image and \( w_{\text{scale}}^s \) is the linear combination weight associated with scale \( s \), which can be vectorized and written as \( w_{\text{scale}} \), and \( \sigma(\cdot) \) is used to denote the sigmoid function that maps the boundary prediction activations into the range \([0,1]\). The final boundary prediction, \( \hat{y} \), is obtained by passing the rescaled image, \( x^s \), though a series of convolutional layers, rectified linear unit (ReLU) layers, and max-pooling layers. We use \( CNN(x^s; W_{\text{base}}, w_{\text{side}}) \) to denote the processing done on the rescaled image, \( x^s \), by a convolutional neural network parameterized by two sets of weights, \( W_{\text{base}} \) and \( w_{\text{side}} \), to produce the scale-specific boundary map,

\[
\hat{y}^s = CNN(x^s; W_{\text{base}}, w_{\text{side}}^s). \tag{2}
\]

Note that \( W_{\text{base}} \) is independent of the scale of the image and is shared across all image scales, and \( w_{\text{side}}^s \) denotes the scale-specific weights. We will explain both these weights in more detail, shortly.

Recently, various works have shown that a boost in performance is achievable by using features from the intermediate layers of the deep network \([19, 12, 28]\). M-DSBD also uses features from the intermediate layers of the base network, which we combine using a linear combination to produce a scale-specific boundary prediction map. Let, \( f(s,k)(i) \in \mathbb{R}^{d_k} \) \( d_k \) is the number of convolutional filters in layer \( k \) denote the feature vector at a spatial location, \( i \), obtained from an intermediate layer, \( k \), and, let the subset of the weights of the base network \( (W_{\text{base}}) \) that are used to produce the features, \( f(s,k) \), be denoted as \( W_{\text{base}}^{1:k} \). We fuse these features into a 1-channel feature map, \( f_{\text{side}}(s,k) \), which can be extracted at the side of each intermediate layer, \( k \), using a \( 1 \times 1 \) convolution kernel, i.e.,

\[
f_{\text{side}}(s,k)(i) = W_{\text{fuse}}^{s,k} f(s,k)(i) \tag{3}
\]

where, \( f_{\text{side}}(s,k)(i) \) is the 1-channel feature at the spatial location, \( i \), and \( W_{\text{fuse}}^{s,k} \) are the linear weights used to combine the intermediate layer features.

Due to the max-pooling at end of the intermediate layers, the spatial resolution of the side-output features, \( f_{\text{side}}(s,k) \), will not the same as the spatial resolution of the image, \( x \). So, we upsample the side-output features, using a deconvolution layer with an appropriately sized kernel, \( w_{\text{up}}(s,k) \), before taking a linear combination of these side output features to produce the scale-specific boundary prediction activation,

\[
\hat{y}^s_a(i) = \sum_{k=1}^{K} w_{\text{fuse}}^{s,k} f_{\text{side}}(s,k)(i). \tag{4}
\]

Here, \( f_{\text{side},up}^{s,k} = UP(f_{\text{side}}^{s,k}, w_{\text{up}}^{s,k}) \) is the upsampled feature map, \( w_{\text{up}}^{s,k} \) are the weights corresponding to the interpolation kernel, and \( w_{\text{fuse}}^{s,k} \in \mathbb{R}^K \) is the weight associated with the \( k \)-th layer side output for performing the linear fusion. We combine all linear fusion weights into a vector notation, \( w_{\text{fuse}} \in \mathbb{R}^K \), where \( K \) is the total number of layers in the deep network. We group all the side-output weights and denote the set as \( w_{\text{side}}^s = \{ w_{\text{fuse}}^{s,k} \}_{k=1}^{K} \cup \{ w_{\text{up}}^{s,k} \}_{k=1}^{K} \cup \{ w_{\text{fuse}}^{s,k} \} \).

We initialize the base network weights, \( W_{\text{base}} \), from the five convolutional layers of the VGG16 network \([25]\), which was pretrained on the ImageNet database. We encourage the reader to refer to \([25]\) for the architecture of the base network. From our experiments, we found that augmenting the VGG16 convolutional weights, with an additional convolutional layer \((\text{conv5}_4)\), improved the performance of the boundary detection task. Therefore, our base network architecture consists of the original convolutional layers from the VGG16 architecture and an additional convolutional layer, \text{conv5}_4, which consists of 512 filters of size \(3 \times 3\).
weights for this new conv5_4 layer were initialized randomly by drawing from a Gaussian distribution.

4.1. Training Procedure

We now describe the training procedure that was employed to train the weights in our deep network. As mentioned above, we build on the Fully Convolutional Network architecture, which allows us to backpropagate the gradients computed at each pixel location.

Our training set consists of the image-boundary label pairs, \( D = \{(x_1, y_1), (x_2, y_2), ..., (x_d, y_d)\} \), where \( x_i \)’s are the images and \( y_i \)’s are the boundary labels. We employ batch-stochastic gradient descent to update the initial weights in each layer starting from layer 1.

Once the weights have been fine-tuned using our greedy layer-by-layer update procedure, we switch off the side-output loss, \( \Delta_b(y, \hat{y}^k) \), which is computed between the side output, \( \hat{y}^k = \sigma(\mathbf{w}_{(\text{side, up})}^{(s,k)}) \), obtained from the intermediate features out of layer \( k \), and the ground truth boundary map, \( y \). The side-output loss is the sum of the weighted cross-entropy loss at each pixel location, i.e.,

\[
\Delta_b(y, \hat{y}) = -\beta \sum_{j \in \{i | y(i) = 1\}} \log P(y_j = 1 | x; \mathbf{W}_{(\Delta,b)}) - (1 - \beta) \sum_{j \in \{i | y(i) = 0\}} \log P(y_j = 0 | x; \mathbf{W}_{(\Delta,b)})
\]

where \( \mathbf{W}_{(\Delta,b)} = \{ \mathbf{W}_{\text{base}} \} \cup \{ \mathbf{w}_{\text{side}}^{(s,k)} \} \cup \{ \mathbf{w}_{\text{up}}^{(s,k)} \} \), and \( \beta \) is the class-balancing weight. Class-balancing is needed because of the severe imbalance in between the number of boundary pixels and non-boundary pixels. We fixed \( \beta = 0.9 \), which we found to work well in our experiments.

The layer-by-layer deep supervision network uses a side-output loss, \( \Delta_b(y, \hat{y}^k) \), to update only the weights corresponding to that layer. The weights of all other layers are not changed. For example, while backpropagating from \( \Delta_b(y, \hat{y}^k) \), only the weights, \( \{ \mathbf{w}_{\text{base}}^{(k)} \} \cup \{ \mathbf{w}_{\text{fuse}}^{(s,k)} \} \cup \{ \mathbf{w}_{\text{up}}^{(s,k)} \} \) are updated; \( \mathbf{W}_{\text{base}} \) corresponds to the weights in the \( k \)-th layer of the base network. The rest of the weights are untouched. We sequentially update the weights in each layer starting from layer 1 and ending at layer \( K \).

Once the weights have been fine-tuned using our greedy layer-by-layer update procedure, we switch off the side-output losses and fine-tune the network using a scale-specific boundary detection loss,

\[
\Delta_s(y, \hat{y}) = -\beta \sum_{j \in \{i | y(i) = 1\}} \log P(y_j = 1 | x; \mathbf{W}_{(\Delta,s)}) - (1 - \beta) \sum_{j \in \{i | y(i) = 0\}} \log P(y_j = 0 | x; \mathbf{W}_{(\Delta,s)})
\]

where \( \mathbf{W}_{(\Delta,s)} = \{ \mathbf{W}_{\text{base}} \} \cup \{ \mathbf{w}_{\text{side}}^{(s,k)} \} \). This is different from the training procedure in [28], where the authors employ deep supervision and force each side-output prediction to be a boundary map. We, on the other hand, only use deep supervision to warm-start the training procedure and switch off the gradients from the side-output loss while updating the fusion weights. In other words, we do not enforce each side output to correspond to a boundary prediction map, but use these side outputs as features for the scale-specific boundary map. Enforcing each side output to be a boundary predictor of its own right prevents the fusion layer from providing the best performance. Allowing the side outputs to only act as features for the fusion layer, by switching of the gradients from the side-output loss, enables a layer’s features to be complementary to other layers’ features, thus permitting the fusion weights to extract the best possible performance.

All that is left is to learn the optimal weights to fuse the various scale-specific predictions. To this end, we define the final boundary detection loss, \( \Delta_s(y, \hat{y}) \) as,

\[
\Delta_s(y, \hat{y}) = -\beta \sum_{j \in \{i | y(i) = 1\}} \log P(y_j = 1 | x; \mathbf{W}_{(\Delta,s)}) - (1 - \beta) \sum_{j \in \{i | y(i) = 0\}} \log P(y_j = 0 | x; \mathbf{W}_{(\Delta,s)})
\]

where \( \mathbf{W}_{(\Delta,s)} = \{ \mathbf{W}_{\text{base}} \} \cup \{ \mathbf{w}_{\text{side}}^{(s,k)} \} \cup \{ \mathbf{w}_{\text{scale}} \} \). In this final stage of learning, we switched off the gradients from the side-output losses and the scale-specific losses, and backpropagated the gradients only from the boundary detection loss. Moreover, the base network weights \( \mathbf{W}_{\text{base}} \) were not updated during this final stage, and only the side-output weights, \( \mathbf{w}_{\text{side}}^{(s,k)} \), and the scale-fusion weights, \( \mathbf{w}_{\text{scale}} \), were updated.

5. Experiments

We predominantly experimented on the newly collected PASCAL Boundaries dataset that was described in Section 3. This database consists of images from the trainval set of the PASCAL VOC2010 challenge. There are a total of 10,103 images that have been labeled. We train our deep network on the train set of the dataset and test on the test set. Note that since we label only the images from the trainval set of the PASCAL VOC2010 challenge, the test set of the PASCAL Boundaries dataset corresponds to the val set of the PASCAL VOC2010 challenge.

Implementation details: We used the publicly available FCN code [19], which is built on top of the Caffe framework to train our deep network. We modified the sigmoid_cross_entropy_loss_layer to compute the weighted cross-entropy loss. In addition, we provide functionalities within the Caffe framework that resizes (downsample and upsample) data blobs to arbitrary resize...
Factors\(^1\). Weight updates were performed using batch-SGD with a batch size of 5 images. To enable batch training on a GPU, we resized all images from the training set to a standard resolution of \(400 \times 400\). The learning rate was fixed to \(1e^{-7}\), and weight decay was set to \(0.0002\). We did not augment our training data since the PASCAL Boundaries dataset has \(\sim 5000\) training images.

**Evaluation Criterion:** The standard evaluation criterion for evaluating edge detection algorithms is the F-score. We also use the same evaluation criterion for evaluating our boundary detector. In addition, we provide baselines on the new dataset using two other well-known edge detection algorithms; SE\(^9\) and HED\(^{28}\). We use the helper evaluation functions provided in the SE Detection Toolbox\(^7\) to obtain all the numbers we report in this paper.

### 5.1. Transfer from Edge Detection

We tested the baseline edge detection methods, SE\(^9\) and HED\(^{28}\), on the 5105 images present in the test set of the PASCAL Boundaries dataset, and Figure 4 shows the precision/recall curves. A more detailed, and exhaustive comparison is provided in Table 1. SE and HED models were trained on the BSDS dataset and were released by the respective authors. To make this explicit, we call them SE-BSDS and HED-BSDS, respectively.

We see that both SE-BSDS and HED-BSDS transfer reasonably on to the PASCAL Boundaries dataset; SE-BSDS achieves an F-score of 0.541, while HED-BSDS achieves an F-score of 0.553. The ranking order of SE’s and HED’s performance when tested on the BSDS500 dataset also transfers over when tested on the PASCAL Boundaries dataset. This shows that BSDS500 edges are not entirely different from our definition of segment boundaries. The BSDS500 boundaries constitute object-level boundaries, object part-level boundaries, and boundaries emanating texture. Our database, in comparison, deals only with object-level boundaries.

**Retraining HED on PASCAL Boundaries:** To provide a fair comparison, we tried training HED using their publicly-released training code. We retained all the parameters that were set by the authors. We only replaced the training set from the BSDS500’s augmented training set (which HED uses) to PASCAL Boundaries’ training set. To account for an increase in the complexity of the PASCAL Boundaries dataset (in comparison to the BSDS500 dataset), we trained HED for a total of 100k iterations (as opposed to the 10k iterations that the authors report in [28]). We snapshotted the model every 1000 iterations and used a validation set of 25 images (randomly chosen from the training set) to select the best model. Surprisingly, we found the performance of the best model when tested on the PASCAL Boundaries’ test set to be worse by several % points compared to the performance obtained using their released model (HED-BSDS)\(^2\).

We believe that the optimal parameters set by the authors of HED to train on the BSDS500 dataset might not be the optimal parameters to train on the PASCAL Boundaries dataset. We did not experiment with different parameter settings.

### 5.2. Accretion Study and (M-)DSBD Results

We present our results in a step-by-step modular fashion to show the improvements that the respective components in our model provide.

**Training Strategy:** To test our training strategy, we replaced HED’s training method with greedy layer-by-layer training strategy to warm-start the training process. We then used these updated weights as an initialization to train the HED architecture by backpropagating the gradients from the losses computed at each of the five side-output predictions and the fusion layer prediction, simultaneously, as was done in [28]. This approach of training the HED architecture provided an improvement of 3\% over the results that were obtained while testing with the publicly-released pretrained model; we were able to obtain an F-score of 0.59.

---

\(^1\)Source code will be released for public use.

\(^2\)We obtained an F-score of 0.36 when we trained HED using the authors’ training code on the PASCAL Boundaries dataset. Moreover, we also noticed a drop in performance when we tried replicating HED’s results on the BSDS500 dataset.
Figure 5: This figure shows some qualitative results. Row 1 shows example images, row 2 shows the respective per-pixel class annotations from the PASCAL Context dataset [22], which is used to generate the class-agnostic boundary maps of PASCAL Boundaries shown in row 3, rows 4 and 5 show results from SE [9] and HED [28], respectively, and the final row shows the results from M-DSBD. Notice how M-DSBD is able to identify object-level boundaries and outputs far less number of internal edges. The edge detection techniques, on the other hand, detect edges across multiple levels of hierarchies.

Since this method uses the HED architecture but a different training strategy (i.e., the greedy layer-wise training), we use the term ‘HED-arch-greedy’ to indicate this model.

**More Convolutional Layers:** Since the PASCAL Boundaries dataset is more complex than the BSDS dataset, we experimented with adding more layers to the models so that it could capture the dataset’s complexity. We began by adding an additional convolution layer, conv5_4. We built the layer conv5_4 with 512 filters, each with a kernel size of $3 \times 3$. We also added a ReLU layer to rectify the output of conv5_4. This enhanced architecture was able to further improve the results by $3\%$ over the previous model by producing an F-score of 0.62 on the test set. We experimented with adding more layers to the network, but found that they did not improve the performance of the model. We use the term ‘HED-arch+conv5_4-greedy’ for this model.

**Switching deep supervision off:** An interesting outcome was observed when we used deep supervision just to warm-start the training process. Upon completion of the greedy layer-by-layer training process, we switched off the backpropagation of the gradient from the side-output losses (Eq. 5) and backpropagated only from the scale-specific boundary detection loss$^3$ (Eq. 6). Doing so, improved the performance of the model.

---

$^3$Please note that the above experiment was done on a single scale.
performance of the model by another 2%. We call this version as the single scale Deep Semantic Boundary Detector (DSBD). We believe that the improvement in performance was achievable because we no longer force the side-output predictions to be boundary detectors of their own right, but use them as features for the fusion layer. That said, we do acknowledge the importance of deep supervision for warm-starting the training process.

**Multi-scale Boundary Detection:** Finally, we experimented with the M-DSBD architecture that was described in Section 4. We used three scales, \( S = \{1, 0.8, 0.5\} \), for training and testing. The base network weights were not updated at this stage. Only the scale-specific side output weights, and the multi-scale fusion weights were updated during this final training procedure. The gradients were backpropagated from the boundary detection loss (Eq. 7). Our experiments supported our hypothesis that multi-scale processing would improve the task of boundary detection by providing a further improvement of 1% on the test set of the PASCAL Boundaries dataset. Our model and training procedure produced a final F-score of 0.652, which is significantly more than the other baselines.

We tabulate all the numbers described above in Table 1. ‘BSDS’ is used to indicate that the model was trained on the BSDS500 dataset. We also show some qualitative results in Fig. 5. Notice that our boundary detector is capable of identifying the semantic boundaries confidently and detects far less number of internal edges. On the other hand, the edge detectors identify edges across various levels of granularity (which they were trained to detect).

**BSDS500:** For completeness, we report the performance of M-DSBD on the BSDS500 dataset. Table 2 tabulates the results. Note that M-DSBD was trained on the PASCAL Boundaries’ train set, but tested on the BSDS500’s test set. The numbers show that our model transfers to a different dataset while producing competitive results. Fig. 6 shows an example image from the BSDS500 dataset along with the edge and boundary detections. We can see from the figure M-DSBD transfers on to the BSDS500 dataset and is successful in providing high confidence for object boundaries and low confidence for internal edges.

**6. Conclusion and Future Work**

In this paper, we pointed to the ambiguity in the definition of edge detection, and, defined a precise task, namely class-agnostic boundary detection. To facilitate progress in solving this problem, we release a large dataset of \( \sim 10k \) images with labeled boundaries, which is 20 times bigger than the widely-used BSDS500 dataset, and without any ambiguity in the annotations. In addition, we proposed a novel multi-scale deep semantic boundary detector and showed that it performs well on the boundary detection task.

We now conclude the paper by pointing to various new research directions that can emerge out of this dataset. Firstly, since boundaries are complementary to pixel-level semantic labeling, it would be interesting to develop joint techniques that can exploit the advantages of each of these respective tasks. Secondly, state-of-the-art object proposal generators are based on edge grouping. It will be interesting to study the effect that instance-level semantic boundary predictions have on object proposals. And, finally, this dataset allows easy access to regions of occlusions because of the presence of occlusion cues (triple points). This dataset provides a good starting point to work on the hard task of occlusion-handling.

---

When we use the term "scale-specific loss", the gradients were backpropagated from the loss computed using the original-sized images.

| Method       | ODS | OIS | AP  |
|--------------|-----|-----|-----|
| SE - BSDS    | 0.541 | 0.570 | 0.486 |
| HED - BSDS   | 0.553 | 0.585 | 0.518 |
| HED-arch-greedy | 0.59  | -   | -   |
| HED-arch+conv5-4-greedy | 0.62  | -   | -   |
| DSBD         | 0.643 | 0.663 | 0.650 |
| M-DSBD       | 0.652 | 0.678 | 0.674 |

Table 1: Results on the PASCAL Boundaries dataset. SE’s and HED’s results are from the models that were trained on the BSDS500. The results from M-DSBD shows that multi-scale does improve performance over single scale.

| Method       | ODS | OIS | AP  |
|--------------|-----|-----|-----|
| SE [9]       | 0.746 | 0.767 | 0.803 |
| HED [28]     | 0.782 | 0.804 | 0.833 |
| M-DSBD-PASCAL-B | 0.751 | 0.773 | 0.789 |

Table 2: Results on the BSDS500 dataset. The M-DSBD model was trained on the PASCAL Boundaries dataset. The results show that methods trained on a boundary detection task perform fairly well on the edge detection task.
References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011. 1, 2
[2] Y. Bengio, P. Lamblin, D. Popovici, and H. a. Larochelle. Greedy layer-wise training of deep networks. 2007. 5
[3] G. Bertasius, J. Shi, and L. Torresani. Deeepedge: A multi-scale bifurcated deep network for top-down contour detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2
[4] K. Bowyer and P. J. Phillips. Empirical evaluation techniques in computer vision. *IEEE Computer Society Press*, 1998. 2
[5] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 679–698, 1986. 1, 2
[6] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems*, pages 2843–2851, 2012. 1, 2
[7] P. Dollar. Structured edge detection toolbox. https://github.com/pdollar/edges. Accessed: 2015-10-06. 6
[8] P. Dollar, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, volume 2, pages 1964–1971. IEEE, 2006. 2
[9] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014. 1, 2, 6, 7, 8
[10] J. R. Fram and E. S. Deutsch. On the quantitative evaluation of edge detection schemes and their comparison with human performance. *IEEE Transactions on Computers*, 100(6):616–628, 1975. 1
[11] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *International Conference on Computer Vision (ICCV)*, pages 991–998. IEEE, 2011. 2
[12] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015. 4
[13] X. Hou, A. Yuille, and C. Koch. Boundary detection benchmarking: Beyond f-measures. In *Computer Vision and Pattern Recognition (CVPR)*, 2013 *IEEE Conference on*, pages 2123–2130. IEEE, 2013. 1
[14] S. Konishi, A. L. Yuille, J. Coughlan, and S. C. Zhu. Fundamental bounds on edge detection: An information theoretic evaluation of different edge cues. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on*, volume 1. IEEE, 1999. 2
[15] S. Konishi, A. L. Yuille, J. M. Coughlan, and S. C. Zhu. Statistical edge detection: Learning and evaluating edge cues. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(1):57–74, 2003. 2
[16] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. *arXiv preprint arXiv:1409.5185*, 2014. 5
[17] J. J. Lim, C. L. Zitnick, and P. Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3158–3165. IEEE, 2013. 2
[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014. 2
[19] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 3, 4, 5
[20] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *International Conference on Computer Vision (ICCV)*, volume 2, pages 416–423. IEEE, 2001. 2
[21] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 26(5):530–549, 2004. 1, 2
[22] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 3, 7
[23] X. Ren. Multi-scale improves boundary detection in natural images. In *European Conference on Computer Vision (ECCV)*, pages 533–545. Springer, 2008.
[24] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang. Deep-contour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*., 2015. 2
[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4
[26] V. Torre and T. A. Poggio. On edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 147–163, 1986. 1
[27] J. Uijlings and V. Ferrari. Situational object boundary detection. 2015. 2
[28] S. Xie and Z. Tu. Holistically-nested edge detection. In *International Conference on Computer Vision (ICCV)*, 2015. 1, 2, 3, 4, 5, 6, 7, 8
[29] Y. Zhu, Y. Tian, D. Mexatas, and P. Dollár. Semantic amodal segmentation. *arXiv preprint arXiv:1509.01329*, 2015. 2
[30] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *Computer Vision–ECCV 2014*, pages 391–405. Springer, 2014. 1