PASCAL Boundaries: A Semantic Boundary Dataset with A Deep Semantic Boundary Detector

Vittal Premachandran[1], Boyan Bonev[2], Xiaochen Lian[2], and Alan Yuille[1]

[1]Department of Computer Science, Johns Hopkins University
[2]Department of Statistics, University of California, Los Angeles

Abstract

In this paper, we address the task of instance-level semantic boundary detection. To this end, we generate a large database consisting of more than 10k images (which is 20× bigger than existing edge detection databases) along with ground truth boundaries between 459 semantic classes including instances from both foreground objects and different types of background, and call it the PASCAL Boundaries *dataset. This is a timely introduction of the dataset since results on the existing standard dataset for measuring edge detection performance, i.e. BSDS500, have started to saturate. In addition to generating a new dataset, we propose a deep network-based multi-scale semantic boundary detector and name it Multi-scale Deep Semantic Boundary Detector (M-DSBD). We evaluate M-DSBD on PASCAL boundaries and compare it to baselines. We test the transfer capabilities of our model by evaluating on MS-COCO and BSDS500 and show that our model transfers well.*

1. Introduction

Edge detection has been a fundamental problem in computer vision since the 1970's [12]. It is helpful for many vision tasks, e.g., object detection [31], image segmentation [1], among others. Successful methods include the Canny edge detector [5], *gPb* [1], and Structured Edges (SE) [9]. More recently, edge detection has been addressed by deep networks, e.g., HED [30].

Edge detection datasets, such as the BSDS300 [23] and BSDS500[1], helped improve the state of the art. But results on these datasets have started to plateau and may have achieved the upper bound[16]. Also, with a total of only 500 images, BSDS500 is a fairly small dataset by modern standards which restricts the class of deep networks that can be trained on it. A goal of this paper is to develop a multi-scale deep network architecture for edge detection which motivates introducing a larger-scale annotated dataset.

In this paper we choose to target a specific type of edge, namely instance-level semantic boundaries. We make this choice because edge detection is difficult to define due to the many types of edges arising from different levels of granularity; i) 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 arising from texture (stripes on a tiger) or artificial design (writings on clothes). BSDS addressed this difficulty by assigning several labelers to segment each image as they wished, resulting in annotations which could be combined. In many cases the annotations were consistent, but sometimes there were differences [15]. In this work, we restrict ourselves to the coarsest level of edge granularity, i.e., semantic instance-level object boundaries, since we consider it to be the most important and least ambiguous edge category. Figure 1 shows an example BSDS image with its edge

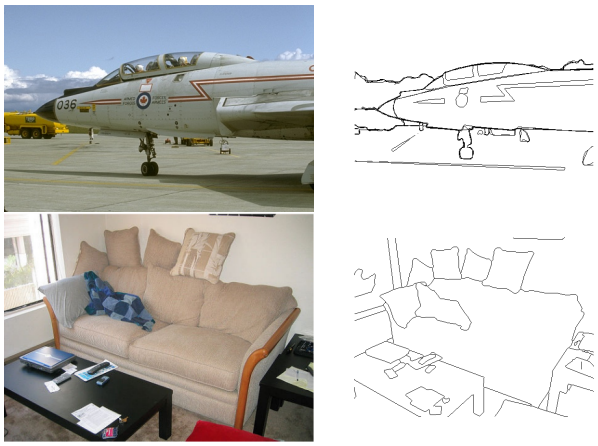


Figure 1: This figure shows the differences between edge annotations from the BSDS500 (top row) and the PASCAL Boundary annotations (bottom row). BSDS500 has multiple annotations that differ with each other. The PASCAL Boundaries annotations are restricted to the highest level of edge granularity, i.e., instance boundaries, providing a single consistent annotation.

annotations (top row) and an example image from PASCAL Boundaries along with its semantic instance-level boundary annotations (bottom row).

To design PASCAL Boundaries we build on PASCAL Context [25] by providing new large-scale instance-level boundary annotations between 459 semantic classes (including both foreground objects and different types of background). PASCAL Context did not include instance-level labels. In addition, annotators cleaned up the boundaries of PASCAL Context. PASCAL Boundaries differs from other instance-level segmentation datasets like SBD [13] and MS-COCO [21] because it gives boundary annotations on a significantly large number of object categories (459), while SBD and MS-COCO restrict their annotations to only 20 and 80 foreground categories, respectively. With these additional labels in the background classes and other foreground objects, PASCAL Boundaries provides twice as many boundary annotations as the SBD dataset.

In addition to building the dataset, we design a deep network-based class-agnostic boundary detector, motivated by the HED architecture [30]. We make some notable changes to the HED architecture by introducing a multi-scale processing pipeline and train each layer in the deep network using a greedy layer-wise approach with deep supervision [19]. We test our model on the PASCAL Boundaries dataset and show that it outperforms well-known edge detectors such as Structured Edges [9] and HED [30]. We also perform experiments on the BSDS500 and MS-COCO datasets to show our model's transfer capability.

The contributions of this paper are as follows. i) We provide instance-level boundary annotations on over 10k images from PASCAL VOC2010 to serve as a testbed for boundary detection algorithms.[1]. ii) We propose a multi-scale deep network-based class-agnostic semantic boundary detector (M-DSBD) to solve the boundary detection task, which is described in detail in Section 4.

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 [17] [18] were developed and tested on these datasets.

The small size of Sowerby motivated Martin et al. [23] to start creating a public dataset of image segmentations. A set of 300 images from this dataset was then used in [24] and came to be known as the BSDS300 dataset. More recently, the BSDS300 dataset was extended to incorporate 200 additional images and the superset dataset was named as the BSDS500 dataset [1].

The existence of BSDS has helped calibrate novel edge detectors such as the *gPb*-edge detector [1], Sketch Tokens [20], SE [9] and, HED [30]. Deep net edge detection methods, such as the N4-network [7], Deep Edge [3], Deep Contour [27] and HED [30], have shown significant improvements in terms of the F-measure on the BSDS dataset. But the limited size of BSDS may lead to saturation [16]. The BSDS datasets do not distinguish between different levels of edge granularities.

There are other works that have also restricted themselves to defining boundaries at a single level of granularity. Endres and Hoiem [10] cleaned up the BSDS300 dataset by grouping multiple segments within an object into a single object instance. PASCAL VOC2010 challenge [11] provided instance-level segmentations of 20 object categories in $\sim$ 2.2k images. Hariharan et al. [13] extended the instance-level annotations for the 20 PASCAL object categories to include $\sim$ 10k images corresponding to the `trainval` set of the PASCAL VOC2010 challenge to produce the SBD dataset. A more recent dataset that includes instance-level segments is the MS-COCO dataset [21], which contains instance-level masks for 80 object categories and is much larger than the PASCAL dataset. But all these datasets label instance-level masks only on a restricted set of foreground objects (20 or 80 categories). This makes it difficult to evaluate the boundaries detected between various background categories (recall that we have twice as many annotations as SBD). The NYU dataset [26] provides an answer to this problem by labeling every pixel in an image on $\sim$1.5k images. But all the images in this dataset are indoor scenes and thus cannot capture the variations in real-world (indoor and outdoor) scenes.

3. Dataset Description

We propose a new dataset for the boundary detection task and call it PASCAL Boundaries. PASCAL Boundaries restricts 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.

3.1. Dataset contributions

We use PASCAL Context [25] as a starting point to build the dataset. The PASCAL Context annotations were obtained by asking the labelers to label all the pixels belonging to the same object category (without sub-pixel precision). It goes beyond the original PASCAL semantic segmentation task by providing annotations for the whole scene. The images are obtained from the PASCAL VOC2010 challenge. We make two significant modifications to the PASCAL Context annotations.

[1]The annotations can be found at http://ccvl.jhu.edu/datasets/

1. Instance-level Boundaries: We hired annotators to extend the category-level mask annotations to *instance-level* mask annotations for all the 459 categories identified in PASCAL Context. For certain difficult and ambiguous cases, e.g. two trees next to each other, we separate them into different instances if they are well-separated in the image. If not, they remain grouped into a single mask. An example of the differences between PASCAL Context and PASCAL Boundaries is shown in Figures 2c and 2d. Other instance-level annotation datasets (SBD and COCO) label only a restricted set of foreground object categories (20 and 80, resp.) and do not label any of the background categories. PASCAL Boundaries dataset has twice as many boundary annotations as the SBD dataset (1.45% vs. 0.77% of the pixels), which shows the importance of labeling boundaries between background categories. Figures 2b and 2d clearly reveal the richness of the annotations in PASCAL Boundaries masks in comparison to the annotations in SBD.

2. Higher-precision boundaries: The precision and quality of the boundaries provided by category-level masks in the semantic segmentation datasets (e.g. MSRC, LabelMe etc.), though good enough for semantic segmentation tasks, are usually not satisfactory for the more high-precision boundary detection task. Although PASCAL Context has higher quality masks, some of them needed to be cleaned up. Therefore, we hired annotators to fix the masks at the boundaries of the PASCAL Context dataset that we deemed were not satisfactory for boundary detection.

Mask2Boundaries: The boundary annotations were then obtained by an automatic post-processing of the improved PASCAL Context mask annotations. The boundaries would ideally be of zero-pixel width: right between the object instances. 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 was be beyond the abilities of human annotators. Rows 1 and 3 in Fig. 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 improved PASCAL Context annotations (shown in row 2).

Dataset Statistics and Characteristics: PASCAL Boundaries has images of 360×496 pixels on average, from which an average of 1.45% of pixels are annotated as boundaries. This is in comparison to the SBD dataset [13], which has only 0.77% of pixels labeled as boundaries. This clearly shows that labeling the boundaries of only the foreground objects ignores at least 50% of all the existing boundaries in an image. The percentage of pixels labeled as boundary in the PASCAL Boundaries dataset, however, 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 different levels of granularity (e.g. the interiors of objects). This number drops to 0.91% if we consider only those pixels of the BSDS500 annotations that were labeled by all the annotators annotating the image. Fig. 3 shows the top 45 category pairs that contribute to the boundaries in the PASCAL Boundaries dataset. We can see that the top 7 contributors are because of the additional annotations that we provide (e.g. ground/sky), which didn't exist in the original 20 PASCAL object categories.

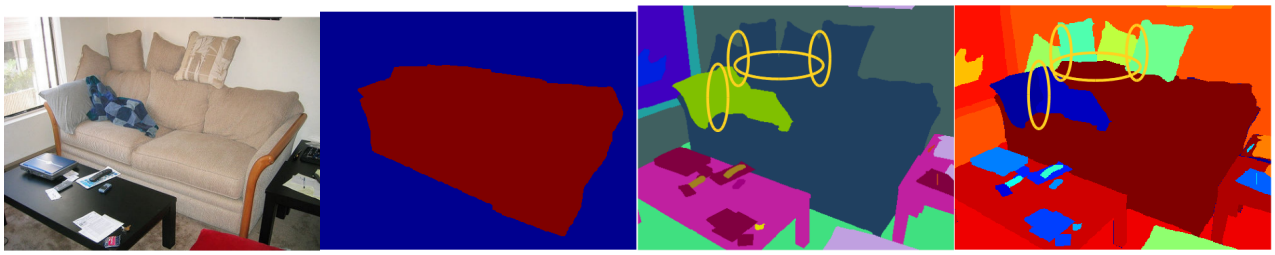
Many of the images in this dataset are non-iconic, which means they contain multiple objects, not necessarily biased towards "photography images" (one salient object in the center with high contrast with respect to the background). Minimizing the bias towards "photography images" is beneficial for computer vision applications that need to work in the real world. We also emphasize that the number of images in the PASCAL Boundaries dataset ($\sim$ 10k) is much larger than in existing edge detection datasets. The increased scale of this dataset provides more variation in the boundary types and is beneficial for learning deep models. Moreover, testing algorithms on thousands of images, as opposed to testing them on a couple hundred images, will provide more credibility to future boundary detection models.

Dataset Split: We split the dataset into three parts; `train`, `val` and `test`. The PASCAL Boundaries `test` set consists of the same images as in the `val` set of PASCAL Context dataset; 5105 images. The `val` set consists of 500 images from PASCAL Context's `train` set, and the `train` set consists of the rest of the images from the PASCAL Context's `train` set.

4. M-DSBD

To complement the PASCAL Boundaries database, we propose a 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. We build upon the HED architecture and use the publicly released code of fully convolutional network (FCN) architecture [22].

Our network architecture is shown in Figure 4. 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 [30], 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, thus leading to a more robust boundary detector (also cor-



(a) Original Image (b) SBD [13] (c) PASCAL Context [25] (d) PASCAL Boundaries Masks

Figure 2: (a) Shows an example image. (b) Shows the instance-level mask from SBD [13]. (c) Shows the category-level masks from PASCAL Context [25]. (d) Shows the instance-level masks in PASCAL Boundaries, which is significantly richer than the annotations in SBD. The yellow ovals (displayed only for illustration) show example regions where instance-level annotations were introduced in the PASCAL Boundaries dataset over PASCAL Context's category-level masks.

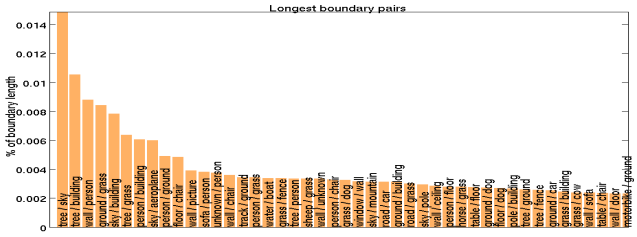


Figure 3: Longest boundaries classified by the pairs of categories which the boundary separates. Only the top 45 out of 105,111 possible combinations are shown.

roborated by the results from our experiments).

More formally, for a given image, $\mathbf{x}$, the network rescales it to multiple scales, $\mathcal{S} \in \{1, 2, ..., |\mathcal{S}|\}$, to produce an image pyramid, $\{\mathbf{x}^s\}_{s=1}^{|\mathcal{S}|}$. Our network acts on each rescaled image in this image pyramid, $\mathbf{x}^s$, and outputs a class-agnostic boundary map for each scale, $\hat{\mathbf{y}}^s (= \sigma(\hat{\mathbf{y}}_a^s))$. The final boundary prediction, $\hat{\mathbf{y}}$, involves taking a linear combination of the scale-specific boundary prediction activations, $\hat{\mathbf{y}}_a^s$ as $\hat{\mathbf{y}}(i) = \sigma(\sum_{s=1}^{|\mathcal{S}|} w_{scale}^s \hat{\mathbf{y}}_a^s(i))$. Here, i is used to index the pixel locations in the image and w_{scale}^s is the linear combination weight associated with scale s, which can be vectorized and written as $\mathbf{w}_{scale}$, and $\sigma(.)$ is used to denote the sigmoid function that maps the boundary prediction activations into the range $[0, 1]$.

Each scale-specific boundary prediction, $\hat{\mathbf{y}}^s$, is obtained by passing the rescaled image, $\mathbf{x}^s$, though a series of convolutional layers, rectified linear unit (ReLU) layers, and max-pooling layers. We use $CNN(\mathbf{x}^s; \mathbf{W}_{base}, \mathbf{w}_{side}^s)$ to denote the processing done on the rescaled image, $\mathbf{x}^s$, by a convolutional neural network parameterized by two sets of weights, $\mathbf{W}_{base}$ and $\mathbf{w}_{side}^s$, to produce the scale-specific boundary map, $\hat{\mathbf{y}}^s = CNN(\mathbf{x}^s; \mathbf{W}_{base}, \mathbf{w}_{side}^s)$. Note that $\mathbf{W}_{base}$ is independent of the scale of the image and is shared across all image scales, and $\mathbf{w}_{side}^s$ denotes the scale-specific weights. We will explain both these weights in more detail.

Recently, various works have shown that a boost in performance is achievable by using features from the intermediate layers of the deep network [30][22][14][6]. MDSBD 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, $\mathbf{f}^{(s,k)}(i) \in \mathbb{R}^{d_k}$ (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 ($\mathbf{W}_{base}$) that are used to produce the features, $\mathbf{f}^{(.,k)}$, be denoted as $\mathbf{W}_{base}^{1:k}$. We fuse these features into a 1-channel feature map, $\mathbf{f}_{side}^{(s,k)}$, which can be extracted at the side of each intermediate layer, k, using a 1×1 convolution kernel, i.e., $\mathbf{f}_{side}^{(s,k)}(i) = \mathbf{w}_{feat}^{(s,k)\top} \mathbf{f}^{(s,k)}(i)$, where, $\mathbf{f}_{side}^{(s,k)}(i) \in \mathbb{R}$ is the 1-d feature at the spatial location, i, and $\mathbf{w}_{feat}^{(s,k)}$ are the linear weights used to combine the intermediate layer features.

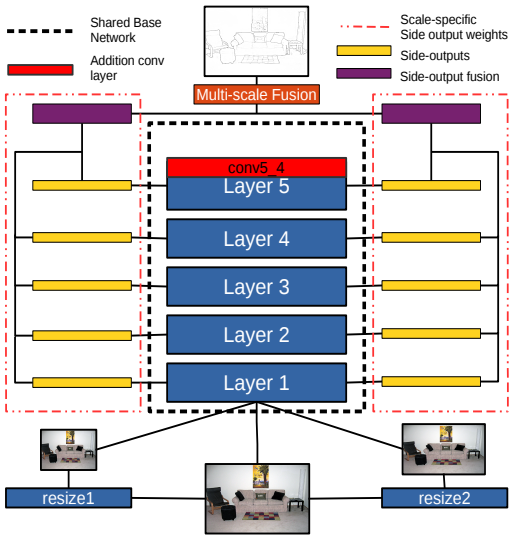


Figure 4: This figure shows our multi-scale deep network architecture. The base network weights are shared across all scales. The figure only shows two side-output connections, while in practice, the multi-scale fusion layer fuses the predictions from three different scales.

Due to the max-pooling layers, the spatial resolution of the side-output features, $\mathbf{f}_{side}^{(s,k)}$, will not be the same as the spatial resolution of the image, $\mathbf{x}^s$. So, we upsample the side-output features, using a deconvolution layer with an appropriately sized kernel, $\mathbf{w}_{up}^{(s,k)}$, before taking a linear combination of these side output features to

produce the scale-specific boundary prediction activation, $\hat{\mathbf{y}}_a^s(i) = \sum_{k=1}^{K} w_{fuse}^{(s,k)} \mathbf{f}_{(side,up)}^{(s,k)}(i)$. Here, $\mathbf{f}_{(side,up)}^{(s,k)} = UP(\mathbf{f}_{side}^{(s,k)}; \mathbf{w}_{up}^{(s,k)})$ is the upsampled feature map, $\mathbf{w}_{up}^{(s,k)}$ are the weights corresponding to the interpolation kernel, and $w_{fuse}^{(s,k)} \in \mathbb{R}$ 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, $\mathbf{w}_{fuse}^s \in \mathbb{R}^K$, where K is the total number of layers in the deep network. We concatenate all the side-output weights and denote the concatenated vector as $\mathbf{w}_{side}^s = [[\mathbf{w}_{feat}^{(s,k)}]_{k=1}^K, [\mathbf{w}_{up}^{(s,k)}]_{k=1}^K, [\mathbf{w}_{fuse}^s]]$, where $[.]_{k=1}^K$ is used to denote the concatenation of the layer-specific weights.

We initialize the base network weights, $\mathbf{W}_{base}$, from the five convolutional layers of the VGG16 network [28], which was pretrained on the ImageNet database. We encourage the reader to refer to [28] for the architecture of the base network. From our experiments, we found that augmenting the VGG16 convolutional weights, with an additional convolutional layer (`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, `conv5_4`, which consists of 512 filters of size 3×3. The 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, $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ..., (\mathbf{x}_{|\mathcal{D}|}, \mathbf{y}_{|\mathcal{D}|})\}$, where $\mathbf{x}_i$'s are the images and $\mathbf{y}_i$'s are the boundary labels. We employ batch-stochastic gradient descent to update the initialized weights. We use a class-balanced cross-entropy loss,

$$\begin{aligned}\Delta(\mathbf{y}, \hat{\mathbf{y}}; \mathbf{W}) = -\beta \sum_{j \in \{i|\mathbf{y}(i)=1\}} \log P(\hat{\mathbf{y}}(j) = 1|\mathbf{x}; \mathbf{W}) \\ -(1-\beta) \sum_{j \in \{i|\mathbf{y}(i)=0\}} \log P(\hat{\mathbf{y}}(j) = 0|\mathbf{x}; \mathbf{W}),\end{aligned} \tag{1}$$

as the loss function. Here, $\mathbf{y}$ is the groundtruth label map, $\hat{\mathbf{y}}$ is the predicted label map, $\mathbf{W}$ are the weights used to make the prediction, $\hat{\mathbf{y}}$, and β is the class-balancing weight, which is fixed to 0.9 in all our experiments. Class-balancing is needed because of the severe imbalance in between the number of boundary pixels and non-boundary pixels.

We make use of a layer-by-layer deep supervision [19] to warm-start the training procedure. We greedily update the weights [2] in each layer by backpropagating the gradients from a side-output loss, $\Delta_k(\mathbf{y}, \hat{\mathbf{y}}^k; \mathbf{W}_{(\Delta,k)})$, which is computed between the ground truth boundary map, $\mathbf{y}$, and the side output, $\hat{\mathbf{y}}^k (= \sigma(\mathbf{f}_{(side,up)}^{(s,k)}))$, obtained from the intermediate features out of layer k using the weights $\mathbf{W}_{(\Delta,k)} = [\mathbf{W}_{base}^{1:k}, \mathbf{w}_{fuse}^{(s,k)}, \mathbf{w}_{up}^{(s,k)}]$. The layer-by-layer deep supervision procedure uses a side-output loss, $\Delta_k(\mathbf{y}, \hat{\mathbf{y}}^k)$, to update only the weights corresponding to that layer. The weights of all other layers are not changed. For example, while backpropagating from $\Delta_k(\mathbf{y}, \hat{\mathbf{y}}^k)$, only the weights, $[\mathbf{W}_{base}^k, \mathbf{w}_{fuse}^{(s,k)}, \mathbf{w}_{up}^{(s,k)}]$ are updated; $\mathbf{W}_{base}^k$ 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 upto 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 finetune the network using a scale-specific boundary detection loss, $\Delta_s(\mathbf{y}, \hat{\mathbf{y}}^s; \mathbf{W}_{(\Delta,s)})$ where $\mathbf{W}_{(\Delta,s)} = [\mathbf{W}_{base}, \mathbf{w}_{side}^s]$. This is different from the training procedure in [30], 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 off 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.

We are now left with the task to learn the optimal weights to fuse the various scale-specific predictions. To this end, we use the final boundary detection loss, $\Delta_b(\mathbf{y}, \hat{\mathbf{y}}; \mathbf{W}_{(\Delta,b)})$, where $\mathbf{W}_{(\Delta,b)} = [\mathbf{W}_{base}, [\mathbf{w}_{side}^s]_{s=1}^{|\mathcal{S}|}, \mathbf{w}_{scale}]$. In this final stage of learning, we switched off the gradients from the side-output and scale-specific losses, and backpropagated the gradients only from the boundary detection loss. The base network weights $\mathbf{W}_{base}$ were not updated during this final stage; only $[\mathbf{w}_{side}^s]_{s=1}^{|\mathcal{S}|}$ and $\mathbf{w}_{scale}$ were updated.

5. Experiments

We perform experiments on three datasets; PASCAL Boundaries, BSDS500 and MS-COCO. We train our deep network on the `train` set of PASCAL Boundaries dataset. Hyperparameters, like the number of iterations, β, etc., were tuned on the PASCAL Boundaries `val` set. For the experiments on BSDS500 and MS-COCO, we used the model that was trained on PASCAL Boundaries `train` set and show its transfer capabilities on the `test` set of the respective datasets.

Implementation details: We used the publicly available FCN code [22], which is built on top of the Caffe frame-

work to train our deep network. 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 `train` set to a standard resolution of 400×400. 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 `train` set has ~ 4500 training images[2].

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 [30]. We use the helper evaluation functions provided in the SE Detection Toolbox [8] to obtain all the numbers we report in this paper.

5.1. Baselines

Transfer from Edge Detection: We tested the baseline edge detection methods, SE [9] and HED [30] that were trained on BSDS, on the 5105 images present in the `test` set of PASCAL Boundaries. The precision/recall curves are shown in Fig. 6 and the plots are labeled as 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 from texture. Our database, in comparison, deals only with object-level boundaries.

Retraining HED on PASCAL Boundaries: To provide a fair comparison to M-DSBD, we retrained 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' `train` set. To account for an increase in the complexity of the PASCAL Boundaries dataset, we allowed the training to continue until the results started to plateau, which is $\sim$20k iterations. HED, when trained on the PASCAL Boundaries dataset, yields an F-score of 0.60. This plot is labeled as HED-PB in Fig. 6.

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.

[2]Source code can be found at https://github.com/VittalP/M-DSBD

Training Strategy: To test our training strategy, we replaced HED's training method with a greedy layer-by-layer training strategy to warm-start the training process. We then used these updated weights as an initialization to finetune 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 [30]. This approach of training HED provided similar results to HED-PB, yielding an F-score of 0.598. However, this result was obtained in just 5k iterations (3k for greedy layer-wise pretraining + 2k for finetuning). Since this method uses the HED architecture but a different training strategy (i.e., the greedy layer-wise pretraining), 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 3×3. 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, $\Delta_k(\mathbf{y}, \hat{\mathbf{y}}^k; \mathbf{W}_{(\Delta,k)})$, and backpropagated only from the scale-specific boundary detection loss[3], $\Delta_s(\mathbf{y}, \hat{\mathbf{y}}^s; \mathbf{W}_{(\Delta,s)})$. Doing so, improved the 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, $\mathcal{S} = \{1, 0.8, 0.5\}$, 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

[3]Please note that this experiment was done on a single scale. When we use the term "scale-specific loss", the gradients were backpropagated from the loss computed using the original-sized images.

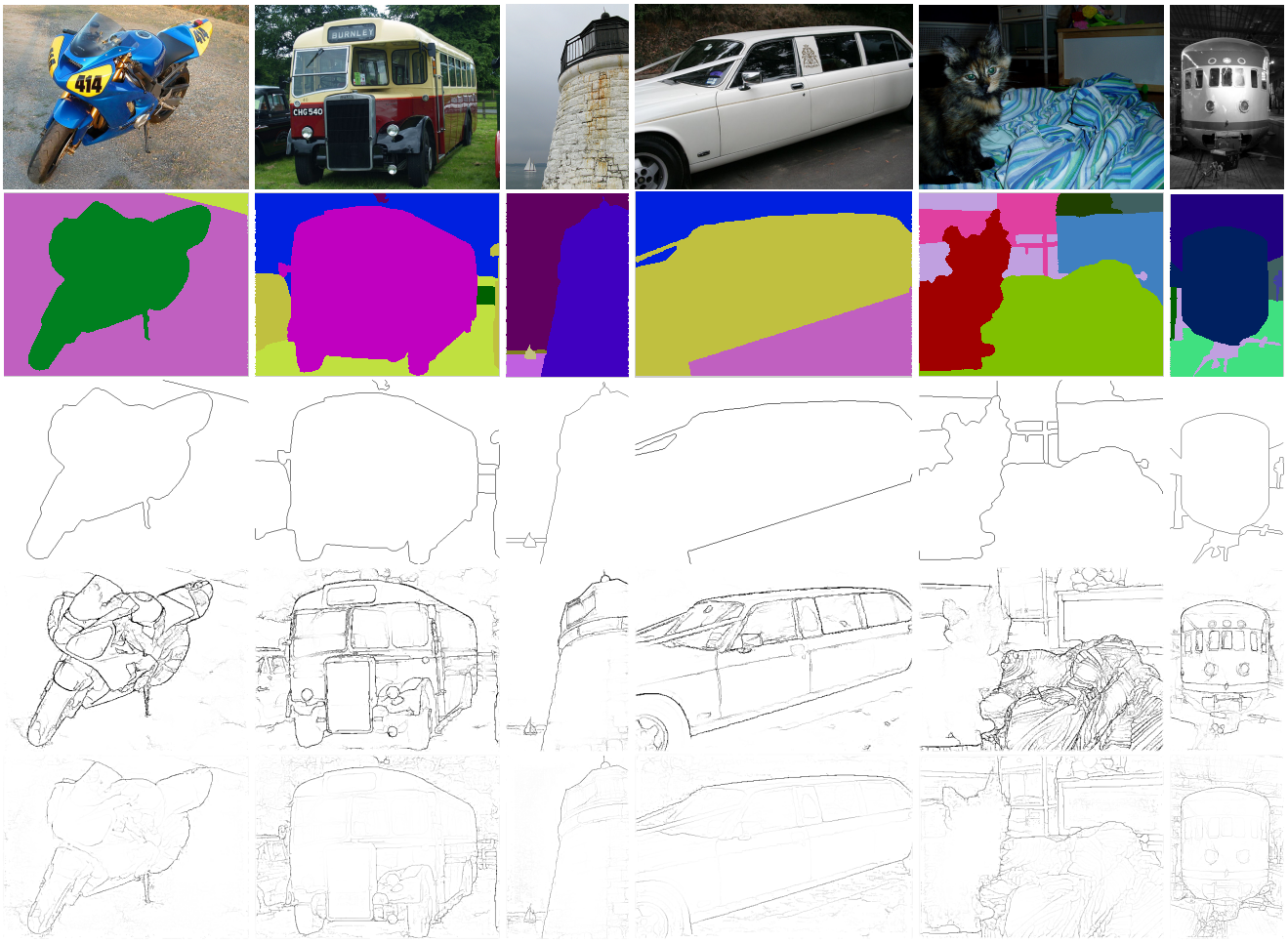


Figure 5: This figures shows some qualitative results. Row 1 shows example images, row 2 shows the respective per-pixel instance-level masks, which is used to generate the class-agnostic boundary maps of PASCAL Boundaries shown in row 3. Row 4 shows results from HED [30]. The last row shows the results 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.

were backpropagated from the boundary detection loss, $\Delta_b(\mathbf{y}, \hat{\mathbf{y}}; \mathbf{W}_{(\Delta,b)})$. 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 PASCAL Boundaries. 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).

5.3. Transfer to Other Datasets

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.

Method	ODS	OIS	AP
SE-BSDS [9]	0.541	0.570	0.486
HED-BSDS [30]	0.553	0.585	0.518
HED-arch-greedy	0.598	-	-
HED-PB	0.60	0.62	0.60
HED-arch+`conv5_4`-greedy	0.62	-	-
DSBD	0.643	0.663	0.650
M-DSBD	**0.652**	**0.678**	**0.674**

Table 1: This table tabulates the results on PASCAL Boundaries `test` set. Note that M-DSBD clearly outperforms HED-PB.

MS-COCO: We also performed experiments on the images from the MS-COCO dataset [21]. This dataset provides instance-level masks for 80 foreground object categories. Some of these categories are not present in the PASCAL dataset. We use the M-DSBD model that was trained on the `train` set of PASCAL Boundaries and, to enable comparison with [29], we test it on the first 5000 images of the

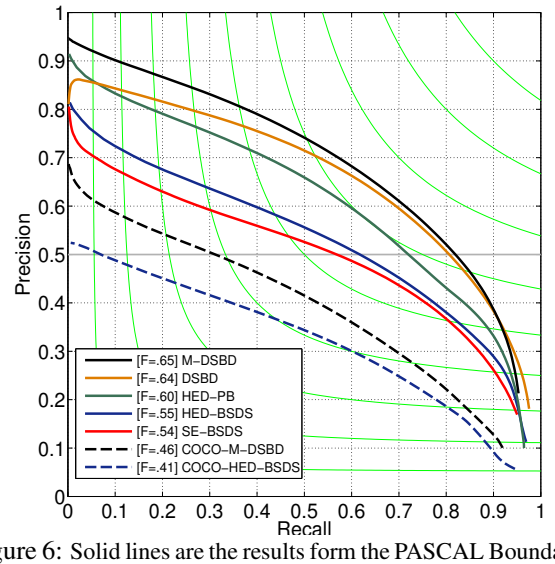
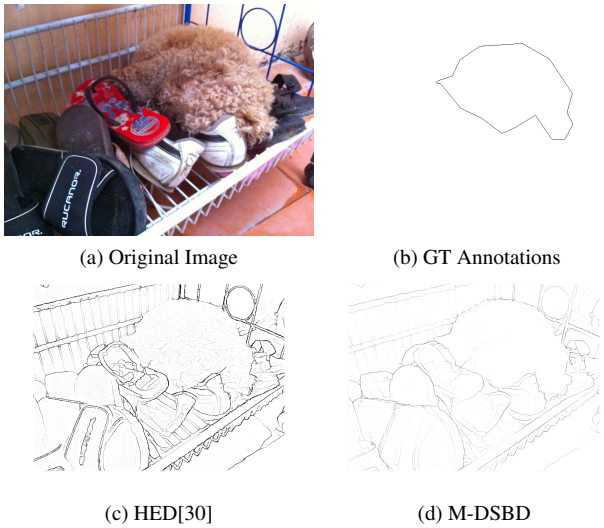


Figure 6: Solid lines are the results form the PASCAL Boundaries `test` set and the dashed lines are the results from the first 5000 images from the `val` set of the MS-COCO. *-BSDS curves were obtained using models trained on BSDS500. M-DSBD curves were obtained by training on the PASCAL Boundaries `train` set, which outperforms HED-PB, which was obtained by training HED on PASCAL Boundaries' `train` set.

`val` set in MS-COCO. These experiments show the class-agnostic transfer capabilities of our deep network since we are testing on object categories that were unseen during training. We also report the results of running HED on the same test set.

The dotted lines in Fig. 6 show the precision-recall curves, and Tab. 3 provides other numbers. Looking at the F-score and AP, we can see that M-DSBD again outperforms HED at the task of boundary detection ([29] do not provide the F-score). Note that the AP values reported in [29] are obtained by using a detector that was trained on MS-COCO, while we are showing the transfer capabilities of our class-agnostic detector that was trained on PASCAL Boundaries. Moreover, the numbers reported in [29] are a result of using multiple situational boundary detectors, while ours is a monolithic detector. Finally, realize that M-DSBD's results (and HED's) are *underestimates* because our method detects boundaries that are not labeled in MS-COCO (e.g. background classes), which we have shown (on the PASCAL dataset) to constitute to about 50% of the boundary annotations. Figure 7 shows the severity of this problem. The ground truth boundary annotations provided by MS-COCO (Fig. 7b) does not include the boundaries of various other objects that are present in the image (Fig. 7a). Also, as shown on PASCAL images, HED (Fig. 7c) detects many internal edges while M-DSBD (Fig. 7d) is better at restricting itself to object instance boundaries.



(a) Original Image (b) GT Annotations

(c) HED[30] (d) M-DSBD

Figure 7: (a) Shows an image from the MS-COCO dataset, (b) shows the groundtruth boundary mask (c) shows the edge output from HED and (d) shows the boundary output from M-DSBD. Notice the lack of richness in the boundary annotations of the MS-COCO dataset.

Method	ODS	OIS	AP
SE [9]	0.746	0.767	0.803
HED [30]	0.782	0.804	0.833
M-DSBD-PASCAL-B	0.751	0.773	0.789

Table 2: This table tabulates results on BSDS500 `test` set. The numbers show that our model transfers reasonably to the edge detection task.

Method	ODS	OIS	AP
Situational-CNN [29]	-	-	0.43
HED-BSDS [30]	0.41	0.42	0.31
M-DSBD-PASCAL-B	0.46	0.47	0.38

Table 3: MS-COCO results: note that the M-DSBD (and HED) numbers are underestimates due to the lack of boundary annotations of background objects in the COCO dataset (Fig. 7).

6. Conclusion and Future Work

In this paper, we addressed the task of instance-level semantic boundary detection. We built upon the PASCAL Context [25] dataset by providing instance-level boundaries to 459 object categories, during which we also improved the quality of certain masks. The introduction of this dataset is timely since the results on the BSDS500 dataset have started to saturate. Moreover, PASCAL Boundaries has annotations on $\sim$ 10k images, thereby making it 20$\times$ bigger than BSDS500. In addition, we developed a multi-scale deep network based boundary detector (M-DSBD) which achieves better performance at the task of boundary detection compared to two other well-known edge detection algorithms. We tested the transfer capabilities of our model on BSDS500 and MS-COCO datasets and showed that the model transfers reasonably well.

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.

[2] Y. Bengio, P. Lamblin, D. Popovici, and H. a. Larochelle. Greedy layer-wise training of deep networks, 2007.

[3] G. Bertasius, J. Shi, and L. Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *CVPR*, 2015.

[4] K. Bowyer and P. J. Phillips. *Empirical evaluation techniques in computer vision*. IEEE Computer Society Press, 1998.

[5] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 679–698, 1986.

[6] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.

[7] 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.

[8] P. Dollar. Structured edge detection toolbox. https://github.com/pdollar/edges. Accessed: 2015-10-06.

[9] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[10] I. Endres and D. Hoiem. Category independent object proposals. In *ECCV*. Springer, 2010.

[11] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[12] 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.

[13] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, pages 991–998. IEEE, 2011.

[14] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.

[15] X. Hou, A. Yuille, and C. Koch. Boundary detection benchmarking: Beyond f-measures. In *CVPR*. IEEE, 2013.

[16] I. Kokkinos. Surpassing humans in boundary detection using deep learning. In *International Conference on Learning Representations*, 2016.

[17] 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 *CVPR*, volume 1. IEEE, 1999.

[18] 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.

[19] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. *arXiv preprint arXiv:1409.5185*, 2014.

[20] J. J. Lim, C. L. Zitnick, and P. Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In *CVPR*, pages 3158–3165. IEEE, 2013.

[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollr, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[22] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[23] 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 *ICCV*, volume 2, pages 416–423. IEEE, 2001.

[24] 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.

[25] 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 *CVPR*, 2014.

[26] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[27] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *CVPR*, 2015.

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[29] J. Uijlings and V. Ferrari. Situational object boundary detection. In *CVPR*, 2015.

[30] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015.

[31] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*. Springer, 2014.