PUSHING THE BOUNDARIES OF BOUNDARY DETECTION USING DEEP LEARNING

Iasonas Kokkinos
Center for Visual Computing
CentraleSupélec and INRIA
Chatenay-Malabry, 92095, France
{iasonas.kokkinos}@ecp.fr

ABSTRACT

In this work we show that adapting Deep Convolutional Neural Network training to the task of boundary detection can result in substantial improvements over the current state-of-the-art in boundary detection.

Our contributions consist firstly in combining a careful design of the loss for boundary detection training, a multi-resolution architecture and training with external data to improve the detection accuracy of the current state of the art. When measured on the standard Berkeley Segmentation Dataset, we improve theoptimal dataset scale F-measure from 0.780 to 0.808 - while human performance is at 0.803. We further improve performance to 0.813 by combining deep learning with grouping, integrating the Normalized Cuts technique within a deep network.

We also examine the potential of our boundary detector in conjunction with the task of semantic segmentation and demonstrate clear improvements over state-of-the-art systems. Our detector is fully integrated in the popular Caffe framework and processes a 320x420 image in less than a second.

1 INTRODUCTION

Over the past three years Deep Convolutional Neural Networks (DCNNs) LeCun et al. (1998) have delivered compelling results in high-level vision tasks, such as image classification (Krizhevsky et al., 2013; Sermanet et al., 2013; Simonyan & Zisserman, 2014; Szegedy et al., 2014; Papandreou et al., 2015b) or object detection (Girshick et al., 2014). Recent works have also shown that DCNNs can equally well apply to pixel-level labelling tasks, including semantic segmentation (Long et al., 2014; Chen et al., 2015) or normal estimation (Sermanet et al., 2014; Eigen et al., 2014). A convenient component of such works is that the inherently convolutional nature of DCNNs allows for simple and efficient 'fully convolutional' implementations (Sermanet et al., 2014; Eigen et al., 2014; Oquab et al., 2015; Long et al., 2014; Chen et al., 2015).

Our focus on this work is the low-level task of boundary detection, which is one of the cornerstone problems of computer vision. Segmentation can be considered to be an ill-posed problem, and multiple solutions can be considered plausible, depending on the task at hand - for instance when playing chess we think of a checker board in terms of 64 regions, but when carrying it we treat it as a single object. This is reflected in the inconsistency of human segmentations, illustrated in Fig. 1.

As detailed in Arbelaez et al. (2011) we can 'benchmark' humans against each other, by comparing every annotator to the 'committee' formed by the rest: if a user provides details that no committee member has provided these count as false positives, while if a user misses details provided by a committee member, these count as misses. Repeating over users and combining the results yields the recall and precision of humans, which are in turn summarized in terms of their f-measure, namely their geometric mean. When evaluated on the test set of Berkeley Segmentation Dataset (BSD) humans have an F-measure of 0.803, which is indicative of the difficulty of the task.

Progress in boundary detection has been consistently narrowing the gap between human and machine performance, as measured Our system yields a higher F-measure than humans: when using a common threshold for the whole dataset (Optimal Dataset Scale -ODS) our system's F-measure

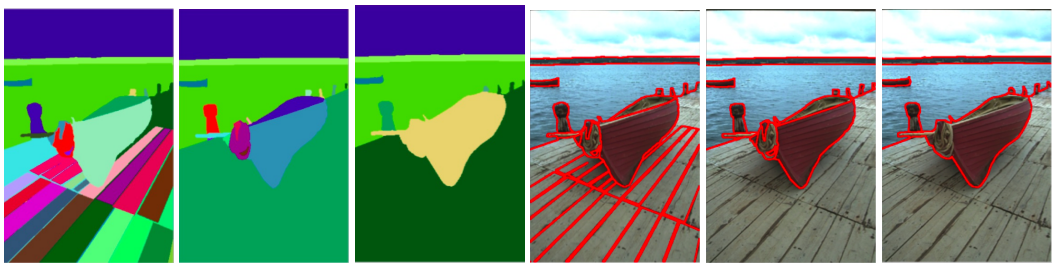


Figure 1: Ground-truth segmentations provided by different annotators for an image from the BSD dataset, and associated boundary maps. The evident lack of agreement among humans is reflected in a low F-measure of human annotators on the task, $F = 0.803$. Our system delivers $F = 0.813$.

equals $F = 0.813$, while when an oracle sets the threshold per image (Optimal Image Scale -OIS) we obtain $F = 0.8308$. Outperforming humans according to this performance measure

As in all works following the introduction of human-annotated datasets (Konishi et al., 2003; Martin et al., 2004), e.g. (Dollar et al., 2006; Arbelaez et al., 2011; Ren, 2008; Kokkinos, 2010a; Ren & Bo, 2012; Dollár & Zitnick, 2015), we use machine learning to optimize the performance of our boundary detector. Recent works (Bertasius et al., 2015; Kivinen et al., 2014; Hwang & Liu, 2015) have shown hat DCNNs yield substantial improvements over flat classifiers; the Holistic Edge Detection approach of Xie & Tu (2015) recently achieved dramatic improvements over the previous state-of-the-art, from an F-measure of 0.75 to 0.78, while keeping computation efficient, requiring 0.4 seconds on the GPU; additional dataset augmentation yielded an F-measure of 0.79.

In this work we make contributions in three fronts: firstly we improve the deep learning algorithms used for boundary detection, secondly we incorporate classical ideas from grouping into the problem and thirdly we exploit our detector to improve the higher-level tasks of semantic segmentation and region proposal generation. We detail these three advances in the following three sections.

2 HED AND DSN TRAINING

We start from a brief presentation of the 'Holistic Edge Detection' (HED) work of Xie & Tu (2015) as it serves as a starting point for our work. HED uses 'Deep Supervised Network' (DSN) (Lee et al., 2015) training to fine-tune the VGG network for the task of boundary detection, illustrated in Fig. 2. The principle behind DSN can be loosely understood as classifier stacking adapted to deep learning and turns out to be practically very successful: if a multi-layer architecture is optimized for a given task, one can anticipate better results by informing each layer about the final objective, rather than relying on the final layer to back-propagate the information to its predecessors. This was shown to systematically improve convergence and test performance, both in generic detection tasks (Lee et al., 2015) and in particular in the context of boundary detection (Xie & Tu, 2015).

In particular, using the notation of Xie & Tu (2015), we have a training set $S = (X_n, Y_n), n = 1, \dots, N$ with X_n being the input image and $Y_n = \{y_j^{(n)}, j = 1, \dots, |X_n|\}, y_j^{(n)} \in \{0, 1\}$ being the predicted labels (we will drop the n subscript for brevity).

We consider a multi-layer network, represented in terms of the union of its individual layer parameters, $\mathbf{W}$, to which we append a set of per-layer 'side' parameters $\mathbf{w}^{(1)}, \dots \mathbf{w}^{(M)}$. These side parameters aim at steering the intermediate layers of the network to extract features that are useful for the classification task even when used on their own. This is reminiscent to classifier stacking, but the difference is that the intermediate layers of the network are not performing classification - but are rather forced to be useful for the side classifier appended to them. This is shown to both improve convergence and test performance (Lee et al., 2015).

The objective function of DSN/HED is phrased as:

$$\mathcal{L}_{side}(\mathbf{W}, \mathbf{w}) = \sum_{m=1}^{M} \alpha_m l^m(\mathbf{W}, \mathbf{w}^{(m)}), \tag{1}$$

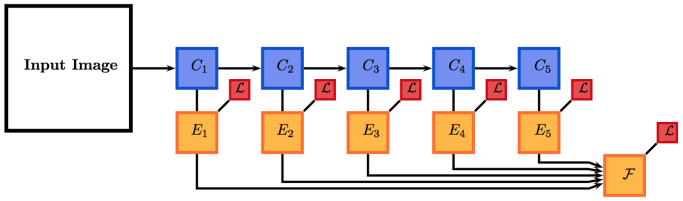


Figure 2: HED/DSN training architecture: every intermediate layer of a DCNN (shown blue) is processed by a side layer (shown in orange) which is penalized by a loss function $\mathcal{L}$. The intermediate results are combined in a late fusion stage, which is again trained with the same loss function.

where l^m are the side-layer losses on the side output of the m-th layer and α_m indicates the importance of the different side layer losses - e.g. setting $\alpha_m = 0, m < M$, which amounts to standard training with a single loss at the top. In HED l^m is a class-balanced cross-entropy loss:

$$l^m(\mathbf{W}, \mathbf{w}^{(m)}) = -\beta \sum_{j \in Y_+} \log P(y_j = 1|X; \mathbf{W}, \mathbf{w}^{(m)}) - (1-\beta) \sum_{j \in Y_-} \log P(y_j = 0|X; \mathbf{W}, \mathbf{w}^{(m)}) \quad (2)$$

$$\doteq \sum_{j \in Y} w_{\hat{y}_j} S(\hat{y}_j, s_j^m), \quad (3)$$

where Eq. 2 Y_+, Y_- are the positive and negative training sample indices respectively, and β is a design parameter set to mitigate the substantially larger number of negative samples in images. The probabilities in Eq. 2 are obtained in terms of a sigmoidal function operating on the inner product $s_j^m = \langle \mathbf{w}^{(m)}, \mathbf{f}_j \rangle$ between the side layer parameters $\mathbf{w}^{(m)}$ and the features $\mathbf{f}_j$ of the DCNN at position j, $P(y_j = 1|X; \mathbf{W}, \mathbf{w}^{(m)}) = \sigma(s_j^m)$. In Eq. 3 we rewrite Eq. 2 in a more general form where we sum over the whole image domain and use the ground truth label $\hat{y}_j$ to indicate which weight and which of the two loss terms is used per pixel j.

An additional processing step of HED is a late fusion stage where the side outputs are combined into a final classification score. This is very meaningful for the task of boundary detection, as it exploits the multi-scale information extracted by the different processing layers of the DCNN. In particular, denoting by S^m the field of values predicted by the m-th side-layer, these are linearly combined into a final score, $S^{fs} = \sum_{m=1}^{M} h_m S^m$; a fusion loss is used to learn the weights $\mathbf{h}$ by calibrating the relative importance of the different side-layers when forming the final prediction:

$$\mathcal{L}_{fuse}(\mathbf{W}, \mathbf{w}, \mathbf{h}) = \sum_{j \in Y} w_{\hat{y}_j} S(\hat{y}_j, \sum_{m=1}^{M} h_m s_j^m) \quad (4)$$

The overall objective function of HED is written as follows:

$$\mathcal{L}_{HED}(\mathbf{W}, \mathbf{w}, \mathbf{h}) = \mathcal{L}_{side}(\mathbf{W}, \mathbf{w}) + \mathcal{L}_{fuse}(\mathbf{W}, \mathbf{w}, \mathbf{h}) \quad (5)$$

and is optimized using common Stochastic Gradient Descent training with momentum.

3 Improved Deep Boundary Detection Training

Having outlined the HED framework, we now turn to our contributions, consisting in (i) Multiple Instance Learning for boundary detection (ii) Graduated Deep Supervision (iii) Multi-Scale training, as well as introducing external data.

The improvements due to these contributions are summarized in Table. 1, where we report our ODS- and OIS-based F-measures on the BSD test set, alongside with the average precision (AP). We compare to our own HED-type baseline that yields a performance marginally below that of the original HED system of Xie & Tu (2015); the latest system of Xie & Tu (2015) has an improved F-measure of $F = 0.79$, due to additional dataset augmentation, which we have not performed yet. We anticipate that this could further boost our already substantially better performance of $F = 0.813$. Further comparisons can be found in Table. 2.

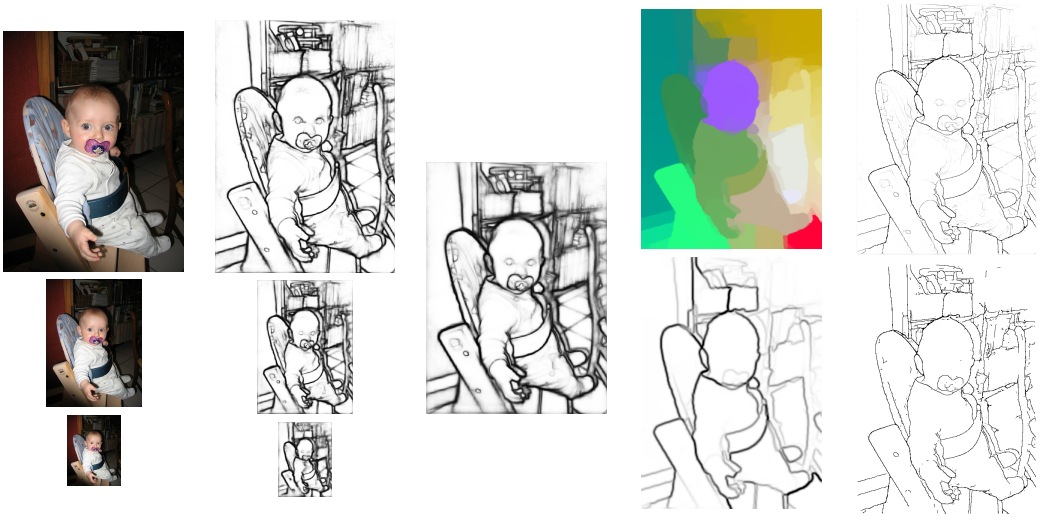


Image Pyramid | Tied CNN outputs | Scale fusion | NCuts & boundaries | Final outputs

Figure 3: Overview of the main computation stages in our system: an input image is processed at three different scales in order to obtain multi-scale information. The the three scales are fused and sent as input to the Normalized Cuts algorithm, that delivers eigenvectors (we show the first three of eight dimensions as an RGB image) and the resulting 'Spectral Boundaries'. The latter are fused with the original boundary map, nonmaximum suppressed, and optionally thresholded (bottom row). All stages are implemented in Caffe, requiring less than a second on an Nvidia Titan GPU.

3.1 Dealing with annotation inconsistencies

The first of our contributions aims at dealing with the inconsistency of human annotations in the BSD, illustrated in Fig. 4. As can be seen, even if the two annotators agree about the semantics (a tiger in water), they may not place the boundaries at a common location. This makes it challenging to define 'positive' and 'negative' training samples in the vincinity of boundaries.

This problem has already been acknowledged in the literature; for instance Sironi et al. (2015) turn boundary detection into a regression problem, by explicitly manipulating the ground truth to become smoother - which however may come at the cost of localization accuracy. In Xie & Tu (2015) a heuristic that was used was to only consider a pixel as positive if it is annotated consistently by more than three annotators. It is however unclear why other pixels should be labelled as negatives.

Our approach builds on Kokkinos (2010a), where Multiple Instance Learning (MIL) (Dietterich et al., 1997) is used to accommodate orientation inconsistencies during the learning of an orientation-sensitive boundary detector. That work was aimed at learning orientation-sensitive classifiers in the presence of orientation ambiguity in the annotations - we take a similar approach in order to deal with positional ambiguity in the annotations while learning a position-sensitive detector.

Standard, 'single instance' learning assumes training samples come in feature-label pairs -or, as in HED above, every pixel is either a boundary or not. Instead, MIL takes as a training sample a set

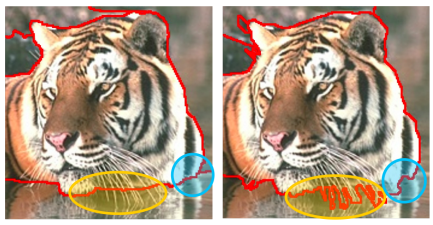


Figure 4: Location uncertainty of human annotations in the BSD dataset: even if annotators agree on the semantics, the boundaries positions remain uncertain. As shown by the blue circle, the precise position is unclear, while as shown by the orange ellipse, even the overall boundary shape may vary.

Method	Baseline	MIL	G-DSN	M-Scale	VOC	Grouping
ODS	0.7781	0.7863	0.7892	0.8033	0.8086	0.8134
OIS	0.7961	0.8083	0.8106	0.8196	0.8268	0.8308
AP	0.804	0.802	0.789	0.8483	0.861	0.866

Table 1: Improvements obtained in this work over our own reproduction of a HED-type baseline : each column corresponds to a Section (MIL: 3.1, G-DSN: 3.2, Multi-Scale: 3.3, VOC: 3.4, Grouping: 4). Each improvement builds on top of the directly previous one.

of features ('bag') and its label. A bag should be labelled positive if at least one of its features is classified as positive, and negative otherwise.

In particular, since human annotations come with some positional uncertainty, the standard evaluation protocol of Martin et al. (2004) allows for some slack in the predicted position of a pixel (a fixed fraction of the image diagonal). One therefore does not need to label every positive pixel as a positive, but rather give a large score to a pixel in its vicinity - and to be more precise, a set of pixels in the line perpendicular to its orientation. This set of pixels forms the bag associated to every positive pixel annotation. A pixel is declared negative if it is not contained in any positive bag.

More specifically, we associate every ground-truth boundary position j with a set of N_j positions and an associated feature bag, $\mathcal{X}_j = \{X_{j,1}, \ldots, X_{j,N_j}\}$. These positions are estimated by identifying the image positions that (i) lie closer to i than any other ground-truth pixel and (ii) have a distance below a threshold d.

For each feature $X_{j,k}$ of the j-th bag our classifier provides a probability $p_{j,k}$ of it being positive, exactly as described inEq. 2 but now the decision is taken by maximizing over instance probabilities:

$$p_{\mathcal{X}_j} = P(y_j = 1|\mathcal{X}_j) = \max_{k \in [1,\ldots N_j]} p_{j,k} \tag{6}$$

The cost function now writes:

$$l^m(\mathbf{W}, \mathbf{w}^{(m)}) = \sum_{j \in Y_-} w_{\hat{y}_j} S(-1, s_j^m) + \sum_{j \in Y_+} w_{\hat{y}_j} S(1, \max_{j \in \mathcal{B}_i} s_j^m) \tag{7}$$

where $\mathcal{B}_i$ is the 'bag' of pixel indices associated with sample i; this allows positive samples to select the neighbours that most support them while forcing all negatives to be negative. In terms of optimization, the $\max$ operation in Eq. 6 is not differentiable, but we can use a subdifferential of p_j:

$$\partial p_j = \frac{dp_{j,k^*}}{df(X_{j,k^*})}, \quad \text{where } k^* = \arg\max_k p_{j,k}. \tag{8}$$

The 'MIL' column of Table. 1 reports improvements over the baseline obtained by setting the distance, d to 1; setting $d = 2$ yields similar improvements.

3.2 Graduated DSN Training

The two terms in the objective function of HED, Eq. 5:

$$\mathcal{L}(\mathbf{W}, \mathbf{w}, \mathbf{h}) = \mathcal{L}_{side}(\mathbf{W}, \mathbf{w}) + \mathcal{L}_{fuse}(\mathbf{W}, \mathbf{w}, \mathbf{h}) \tag{9}$$

play a complementary role: the first, side-layer, terms force the intermediate layers to be discriminative and also extract some preliminary classification information; the second, fusion-layer, term calibrates the importance of the intermediate classifications delivered by the side-layers.

As discussed in Lee et al. (2015), DSN can be understood as simplifying the associated learning problem in terms of optimization. But once the network parameters are in the right regime, we can discard any simplifications that were required to get us there. This is a strategy used in the classical Graduated Non-Convexity technique (Blake & Zisserman, 1987), and here we show that it also helps improve DSN when applied to boundary detection.

For this we modify the training objective by associating the 'side' term with a temporally decreasing weight while keeping the second term's weight fixed:

$$\mathcal{L}^{(t)}(\mathbf{W}, \mathbf{w}, \mathbf{h}) = (1 - \frac{t}{T})\mathcal{L}_{side}(\mathbf{W}, \mathbf{w}) + \mathcal{L}_{fuse}(\mathbf{W}, \mathbf{w}, \mathbf{h}),$$

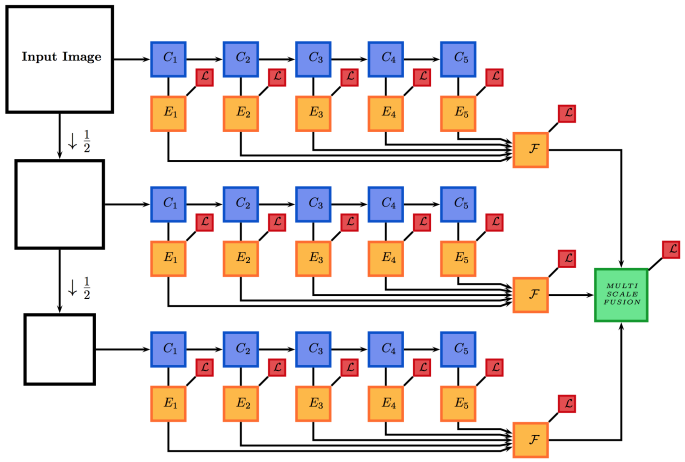


Figure 5: Network architecture used for multi-resolution HED training: three differently scaled versions of the input image are provided as inputs to three FCNN networks that share weights - their multi-resolution outputs are fused in a late fusion stage, extending DSN to multi-resolution training.

where t is the current training epoch and T is the total number of epochs. Our training criterion starts from DSN, where every intermediate layer is trained for classification, and eventually leads to a skip-layer architecture, where the early layers are handled exclusively by the final fusion criterion. By the end the fusion-layer can use the side-layers at will, without the compromises needed to keep the side losses low. The improvements are reported in the G-DSN column of Table. 1.

3.3 Multi-Resolution Architecture

The authors of HED use 'Deep Supervised Network' (DSN) (Lee et al., 2015) training to fine-tune the VGG network for the task of boundary detection, illustrated in Fig. 5. However, image boundaries reside in multiple image resolutions (Witkin, 1983) and it has repeatedly been shown that fusing information from multiple resolutions improves boundary detection, e.g. in Dollár & Zitnick (2015); Arbelaez et al. (2011). Even though the authors of HED use information from multiple scales by fusing the outputs of many layers, multi-resolution boundary detection can still help.

We first observed that simply averaging the results of the network applied to differently scaled versions of the image improved performance substantially. We then turned to a more accurate way of doing the multi-resolution detection: we consider a DSN-type multi-resolution architecture with tied weights, meaning that layers that operate at different resolutions share weights with each other. Parameter sharing across layers both accelerates convergence and also avoids over-fitting. We initialize the weights from a single-resolution architecture and fine-tune with a smaller set of iterations. In order to capture fine-level boundaries the top-resolution image is an upsampled version of the original - e.g. for a 381×421 image from the BSD dataset we use a 577×865 upsampled version, from which we compute a three-level pyramid by downsampling by a factor of 2 and 4.

The multi-resolution results are fused through an additional fusion layer that combines the fused results of the individual resolutions. The improvements are reported in the $S = 3$ column of Table. 1.

3.4 Training with External Data

Even though HED uses the pre-trained VGG network as initialization, dataset augmentation was reported to give substantial improvements. The authors in Xie & Tu (2015) originally used 32 geometric transformations (16 rotations and flipping) of the 300 images used in the BSD trainval set, resulting in a total of roughly 10000 training images - in a recent version the authors consider two additional transformations are considered, resulting in roughly 30000 training images and pushing performance from $F = 0.78$ to $F = 0.79$.

We have not used these additional scalings in our experiments due to time constraints, but have considered the use of boundaries from the PASCAL-Context dataset (Mottaghi et al., 2014), where all objects and 'stuff' present in the scene are manually segmented. Our sole modification to those boundaries has been to label the interiors of houses as 'don't care' regions that are ignored by the

loss, since all of the windows, doors, or balconies that are missed by the annotators seemed to us as being legitimate boundaries. We only apply flipping to these images, resulting in roughly 20000 images, which are appended to the 10000 images we had originally used. As can be seen from the 'VOC' column of Table. 1, this yields a substantial improvement.

4 USING GROUPING IN A DEEP ARCHITECTURE

The combination of the techniques outlined above already help boundary detection outperform humans on the task of boundary detection - but still do not use any grouping information when delivering the probability of having boundaries. The boundary detector only implicitly exploits grouping cues such as closedness or continuity that can often yield improvements in the high-precision regime (Zhu et al., 2007; Kokkinos, 2010b).

To capture such information we use the Normalized Cuts (NCuts) technique of Shi & Malik (1997); Arbelaez et al. (2011). We treat the image as a weighted graph, where nodes corresponding to pixels and weights correspond to low-level affinity between pixels measured in terms of the Intervening Contour cue (Shi & Malik, 1997), where the contours are now estimated by our boundary detector. The NCut technique considers a relaxation of the discrete normalized cut optimization problem, which results in a generalized eigenvector problem (Shi & Malik, 1997):

$$(D - W)\mathbf{v} = \lambda D\mathbf{v}, \tag{10}$$

where D is the graph degree matrix and W is the affinity. The solutions to this generalized eigenvector problem can be understood (Belkin & Niyogi, 2001) as euclidean embeddings of the inter-node distances - so nodes that have similar embeddings are likely to belong together and vice versa.

One of the main impediments to the application of this technique has been computation time, requiring roughly 60 seconds on the CPU for a 321×481 image for 10 eigenvectors. Even though accelerations exist, e.g. Cour et al. (2005), we found it simpler to harness the computational power of GPUs and integrate the Damascene system of (Catanzaro et al., 2009) with the Caffe deep learning framework; when integrated with our boundary detector Damascene yields 8 eigenvectors for a 577×865 image in less that 0.2 seconds. It is also straightforward to use a downsampled version of the boundary map to yield further accelerations.

These embeddings can be used for boundary detection in terms of their directional derivatives, in order to provide some 'global' evidence for the presence of a boundary, known as the 'spectral probability of boundary' cue (Arbelaez et al., 2011). This further improves the performance of our detector, yielding an F-measure of 0.813, which is substantially better than our earlier performance of 0.807, and humans, who operate at 0.803. Due to time constraints we have used a very simple fusion scheme (addition of the posterior with SpectralPB) - we anticipate that adding a few processing layers can further improve performance.

We summarize the impact of the different steps described above in Fig. 6 - starting from a baseline (that performs slightly worse than the HED system of Xie & Tu (2015) we have introduced a series of changes that resulted in a system that performs boundary detection with an F-measure that exceeds that of humans. When compared to the current state-of-the-art method of Xie & Tu (2015) our method clearly dominates in terms of all typical performance measures, as shown in Table 2.

Indicative qualitative results are included in the supplemental material.

5 SYNERGIES WITH HIGHER-LEVEL TASKS

Having pushed the performance of boundary detection to a good level, we now turn to using it in conjunction with other, higher-level tasks. We consider semantic segmentation, and object proposal generation, and observe clear improvements over the currently leading approaches.

5.1 SEMANTIC SEGMENTATION

Since our model is fully-convolutional we can easily combine it with the recent line of works around FCNN-based semantic segmentation(Long et al., 2014; Chen et al., 2015; Papandreou et al., 2015a;

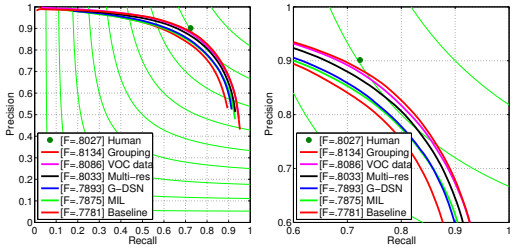


Figure 6: Impact of the different improvements described in Section 2: starting from a baseline that performs only slightly worse than the HED system of (Xie & Tu, 2015) we end up with a detector that largely surpasses human F-measure, illustrated in terms of green isocontours. On the right we zoom into the high-F measure regime.

Method	ODS	OIS	AP
gPb-owt-ucm (Arbelaez et al., 2011)	0.726	0.757	0.696
SE-Var (Dollár & Zitnick, 2015)	0.746	0.767	0.803
DeepNets (Kivinen et al., 2014)	0.738	0.759	0.758
N4-Fields (Ganin & Lempitsky, 2014)	0.753	0.769	0.784
DeepEdge (Bertasius et al., 2015)	0.753	0.772	0.807
CSCNN (Hwang & Liu, 2015)	0.756	0.775	0.798
DeepContour (Shen et al., 2015)	0.756	0.773	0.797
HED-fusion (Xie & Tu, 2015)	0.790	0.808	0.811
HED-late merging (Xie & Tu, 2015)	0.788	0.808	0.840
Ours (DCNN + sPb)	0.8134	0.8308	0.866

Table 2: Comparison to the state-of-the-art in boundary detection, including the latest version of HED, trained with its most recent dataset augmentation (Xie & Tu, 2015). We clearly outperform HED across all performance measures, while maintaining the computational speed above 1 frame per second.

Zheng et al., 2015). These have delivered excellent results, and in particular the use of the Dense Conditional Random Field (DenseCRF) of Krähenbühl & Koltun (2011) by Chen et al. (2015); Papandreou et al. (2015a); Zheng et al. (2015), has enhanced the discriminative power of FCNNs with local evidence gathered by the image intensity.

Following Chen et al. (2015) we define the CRF distribution as:

$$P(\boldsymbol{x}) = \frac{1}{Z}\exp\Big(-E(\boldsymbol{x})\Big), \quad E(\boldsymbol{x}) = \sum_i \phi_i(x_i) + \sum_{ij}\theta_{ij}(x_i, x_j). \tag{11}$$

where $\boldsymbol{x}$ is the pixel-label assignment and $E(\boldsymbol{x})$ is the energy function. In Eq. 11 $\phi_i(x_i) = -\log P(x_i)$ with $P(x_i)$ being the CNN-based probability of assigning label j to pixel i, and $\theta_{ij}(x_i, x_j)$ is a bilateral filter-like image-based pairwise potential between i and j:

$$\theta_{ij}(x_i, x_j) = w^1 \exp\Big(-\frac{|p_i - p_j|^2}{2\sigma_\alpha^2} - \frac{|I_i - I_j|^2}{2\sigma_\beta^2}\Big) + w^2 \exp\Big(-\frac{|p_i - p_j|^2}{2\sigma_\gamma^2}\Big). \tag{12}$$

The first kernel in Eq. 12 depends on both pixel positions (denoted as p) and pixel color intensities (denoted as I), while the second kernel only depends on pixel positions - the hyper-parameters σ_α, σ_β and σ_γ control the Gaussian kernels. Mean-field Inference for this form of pairwise terms can be efficiently implemented with high-dimensional filtering (Adams et al., 2010).

Our modifications are very simple: firstly, we adapt the multi-resolution architecture outlined in the previous section to semantic segmentation. Using multi-resolution processing with tied-weights and performing late score fusion yielded substantially better results than using a single-resolution network: as shown in Table. **??** when combining the multi-scale network's output with DenseCRF inference, performance increases from 72.7 (single-scale counterpart of Chen et al. (2015)) or 73.9 (skip-layer multi-scale counterpart of Chen et al. (2015)) to 74.8 (our multi-scale) in mean accuracy.

Secondly, we integrate the boundary information extracted by our detector into the DenseCRF by using the eigenvectors computed by normalized Cuts to augment the RGB color features of Eq. 12, thereby conveying boundary-based proximity into DenseCRF inference. In particular we augment the dimensionality of I_i in Eq. 12 from 3 to 6, by concatenating the 3 eigenvectors delivered by NCuts with the RGB values. We observe that introducing the Normalized Cut eigenvectors into DenseCRF inference yields a clear improvement over an already high-performing system (from 74.8 to 75.4), while a small additional improvement was obtained we performing graph-cut inference with pairwise terms that depend on the boundary strength (from 75.4 to 75.7). Further improvements can be anticipated though an end-to-end training using the recursive CNN framework of Zheng et al. (2015) as in the currently leading works - we will explore this in future work.

Indicative qualitative results are included in the supplemental material.

6 CONCLUSION

We have proposed a method to substantially improve deep learning-based boundary detection performance. Our system is fully integrated in the Caffe framework and operates in less than one second per frame. Its F-measure, as measured on the standard BSD dataset is higher than that of humans.

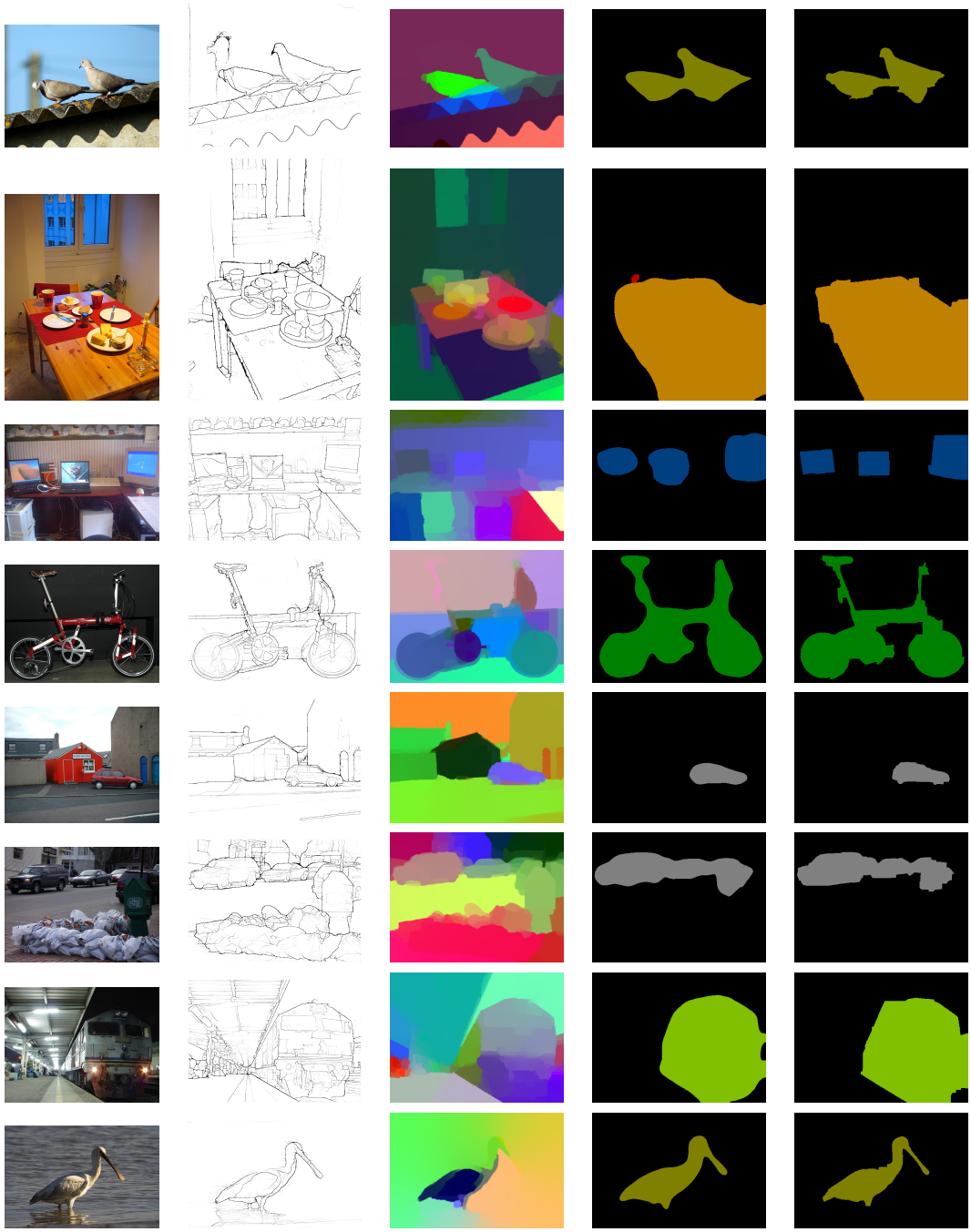
We anticipate that further improvements can be gained through a joint treatment of other low-level cues, such as symmetry (Tsogkas & Kokkinos, 2012) or surface orientation, and depth (Eigen & Fergus, 2014). We also intend to further explore the merit of our detector in the context of high-level tasks, such as object detection and recognition.

7 ACKNOWLEDGEMENTS

This work was supported by FP7-RECONFIG and equipment donated by NVIDIA. I thank the authors of Xie & Tu (2015) for inspiration, Alp Guler for illustrations and tables, Kostas Papazafeiropoulos for help with porting Damascene to Caffe, George Papandreou for guidance on Caffe and Pierre-André Savalle for teaching me to handle prototxt files like a professional `sed`user.

8 SUPPLEMENTAL MATERIAL

We provide below qualitative results on images from the Pascal VOC test set.



(a) (b) (c) (d) (e)

Figure 7: Indicative results on the PASCAL VOC 2012 test set: for each image we show in (b) the final estimate of the probability of boundary, in (c) three leading eigenvectors delivered by Normalized Cuts (d) the semantic segmentation that would be obtained by our multi-scale DCNN variant of DeepLab, prior to DenseCRF inference and (e) the improved result obtained by combining DenseCRF inference with the normalized Cut embeddings and the image boundaries.

REFERENCES

Adams, Andrew, Baek, Jongmin, and Davis, Myers Abraham. Fast high-dimensional filtering using the permutohedral lattice. In *Computer Graphics Forum*, 2010.

Arbelaez, Pablo, Maire, Michael, Fowlkes, Charless, and Malik, Jitendra. Contour detection and hierarchical image segmentation. *PAMI*, 2011.

Belkin, Mikhail and Niyogi, Partha. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, 2001.

Bertasius, Gedas, Shi, Jianbo, and Torresani, Lorenzo. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *Proc. CVPR*, 2015.

Blake, Andrew and Zisserman, Andrew. *Visual Reconstruction*. MIT Press, 1987.

Catanzaro, Bryan C., Su, Bor-Yiing, Sundaram, Narayanan, Lee, Yunsup, Murphy, Mark, and Keutzer, Kurt. Efficient, high-quality image contour detection. In *Proc. ICCV*, 2009.

Chen, Liang-Chieh, Papandreou, George, Kokkinos, Iasonas, Murphy, Kevin, and Yuille, Alan L. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.

Cour, Timothée, Bénézit, Florence, and Shi, Jianbo. Spectral segmentation with multiscale graph decomposition. In *Proc. CVPR*, 2005.

Dai, Jifeng, He, Kaiming, and Sun, Jian. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. *arXiv preprint arXiv:1503.01640*, 2015.

Dietterich, Thomas G., Lathrop, Richard H., and Lozano-perez, Tomas. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31–71, 1997.

Dollar, P., Tu, Z., and Belongie, S. Supervised Learning of Edges and Object Boundaries. In *Proc. CVPR*, 2006.

Dollár, Piotr and Zitnick, C. Lawrence. Fast edge detection using structured forests. *PAMI*, 37(8):1558–1570, 2015.

Eigen, David and Fergus, Rob. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *arXiv:1411.4734*, 2014.

Eigen, David, Puhrsch, Christian, and Fergus, Rob. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014.

Ganin, Yaroslav and Lempitsky, Victor. Nˆ 4-fields: Neural network nearest neighbor fields for image transforms. In *Computer Vision–ACCV 2014*, pp. 536–551. Springer, 2014.

Girshick, Ross, Donahue, Jeff, Darrell, Trevor, and Malik, Jitendra. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

Hwang, J.-J. and Liu, T.-L. Pixel-wise deep learning for contour detection. In *ICLR*, 2015.

Kivinen, Jyri J., Williams, Christopher K. I., and Heess, Nicolas. Visual boundary prediction: A deep neural prediction network and quality dissection. In *AISTATS*, 2014.

Kokkinos, Iasonas. Boundary detection using f-measure-, filter- and feature- (f^3) boost. In *ECCV*, 2010a.

Kokkinos, Iasonas. Highly accurate boundary detection and grouping. In *Proc. CVPR*, 2010b.

Konishi, S., Yuille, A., Coughlan, J., and Zhu, S.-C. Statistical edge detection: Learning and evaluating edge cues. *IEEE Trans. PAMI*, 25(1):57–74, 2003.

Krähenbühl, Philipp and Koltun, Vladlen. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, 2011.

Krähenbühl, Philipp and Koltun, Vladlen. Learning to propose objects. In *Proc. CVPR*, 2015.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2013.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. In *Proc. IEEE*, 1998.

Lee, Chen-Yu, Xie, Saining, Gallagher, Patrick W., Zhang, Zhengyou, and Tu, Zhuowen. Deeply-supervised nets. In *Proc. AISTATS*, 2015.

Lin, Guosheng, Shen, Chunhua, Reid, Ian, et al. Efficient piecewise training of deep structured models for semantic segmentation. *arXiv preprint arXiv:1504.01013*, 2015.

Liu, Ziwei, Li, Xiaoxiao, Luo, Ping, Loy, Chen Change, and Tang, Xiaoou. Semantic image segmentation via deep parsing network. *arXiv preprint arXiv:1509.02634*, 2015.

Long, Jonathan, Shelhamer, Evan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. URL `http://arxiv.org/abs/1411.4038`.

Martin, D., Fowlkes, C., and Malik, J. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans. PAMI*, 26(5):530–549, 2004.

Mottaghi, Roozbeh, Chen, Xianjie, Liu, Xiaobai, Cho, Nam-Gyu, Lee, Seong-Whan, Fidler, Sanja, Urtasun, Raquel, and Yuille, Alan. The role of context for object detection and semantic segmentation in the wild. In *Proc. CVPR*, 2014.

Oquab, Maxime, Bottou, Léon, Laptev, Ivan, and Sivic, Josef. Is object localization for free? - weakly-supervised learning with convolutional neural networks. In *Proc. CVPR*, 2015.

Papandreou, George, Chen, Liang-Chieh, Murphy, Kevin, and Yuille, Alan L. Weakly- and semi-supervised learning of a DCNN for semantic image segmentation. In *Proc. ICCV*, 2015a.

Papandreou, George, Kokkinos, Iasonas, and Savalle, Pierre-André. Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection. In *Proc. CVPR*, 2015b.

Ren, Xiaofeng. Multiscale helps boundary detection. In *ECCV*, 2008.

Ren, Xiaofeng and Bo, Liefeng. Discriminatively trained sparse code gradients for contour detection. In *NIPS*, 2012.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Yann. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014.

Sermanet, Pierre, Eigen, David, Zhang, Xiang, Mathieu, Michaël, Fergus, Rob, and LeCun, Yann. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv:1312.6229*, 2013.

Shen, Wei, Wang, Xinggang, Wang, Yan, Bai, Xiang, and Zhang, Zhijiang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3982–3991, 2015.

Shi, Jianbo and Malik, Jitendra. Normalized cuts and image segmentation. In *Proc. CVPR*, 1997.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

Sironi, A., Turetken, E., Lepetit, V., and Fua., P. Multiscale centerline detection. *PAMI*, 2015.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going deeper with convolutions. *arXiv:1409.4842*, 2014.

Tsogkas, Stavros and Kokkinos, Iasonas. Learning-based symmetry detection in natural images. In *Proc. ECCV*, 2012.

Uijlings, Jasper R. R., van de Sande, Koen E. A., Gevers, Theo, and Smeulders, Arnold W. M. Selective search for object recognition. *IJCV*, 2013.

Witkin, A.P. Scale-space filtering. In *Proc. Int. Joint Conf. on Artificial Intel.*, pp. 1019–1022, 1983.

Xie, Saining and Tu, Zhuowen. Holistically-nested edge detection. In *Proc. ICCV*, 2015.

Zheng, Shuai, Jayasumana, Sadeep, Romera-Paredes, Bernardino, Vineet, Vibhav, Su, Zhizhong, Du, Dalong, Huang, Chang, and Torr, Philip H. S. Conditional random fields as recurrent neural networks. In *Proc. ICCV*, 2015.

Zhu, Qihui, Song, Gang, and Shi, Jianbo. Untangling cycles for contour grouping. In *Proc. CVPR*, 2007.