REDN: A Recursive Encoder-Decoder Network for Edge Detection

TRUC LE and YE DUAN
Department of EECS, University of Missouri, Columbia, MO 65105, USA
Corresponding author: Ye Duan (duanye@missouri.edu)

ABSTRACT In this paper, we introduce REDN: A Recursive Encoder-Decoder Network with Skip-Connections for edge detection in natural images. The proposed network is a novel integration of a Recursive Neural Network with an Encoder-Decoder architecture. The recursive network enables iterative refinement of the edges using a single network model. Adding skip-connections between encoder and decoder helps the gradients reach all the layers of a network more easily and allows information related to finer details in the early stage of the encoder to be fully utilized in the decoder. Based on our extensive experiments on popular boundary detection datasets including BSDS500 [1], NYUD [2] and Pascal Context [3], REDN significantly advances the state-of-the-art on edge detection regarding standard evaluation metrics such as Optimal Dataset Scale (ODS) F-measure, Optimal Image Scale (OIS) F-measure, and Average Precision (AP).

INDEX TERMS Edge detection, deep learning, encoder-decoder network, recursive network.

I. INTRODUCTION
Edge detection has been a cornerstone and long-standing problem in computer vision since the early 1970’s [4]–[6] and is essential for a variety of tasks such as object recognition [7], [8], segmentation [1], [9]–[11], etc. Initially considered as a low-level task, researchers now generally agree that high-level visual context such as the perception of objects play an important role in edge detection [1].

Inspired by the success of deep convolutional neural networks (DCNN) in computer vision problems such as image classification [12]–[14], object detection [15], image segmentation [16]–[21], normal estimation [22], [23], image captioning [24], etc., researchers have begun to utilize DCNN for low-level tasks such as edge detection [25]–[30]. For example, Xie and Tu [25] developed a HED network built upon the VGG-16 network [13] which hierarchically obtains edge images at multiple scales. Edges obtained from the initial levels are more localized while those from the deeper levels are more global. The final edge is a linear combination of all edge images at different scales. Later, Kokkinos [10] explicitly applied HED [25] on the image pyramid. Yang et al. [26] developed a fully convolutional encoder-decoder network (CEDN) similar to Noh et al. [21]. The main drawback of these approaches is that the salient edges are obtained at the deeper layers with relatively lower resolution. Thus, the upsampled edge image tends to be blurry and less localized.

To overcome this limitation, several researchers recently proposed the use of a refinement network for fusing edge images at multiple scales to achieve better edge detection results. For instance, Wang et al. [27] proposed a refinement module that fuses a top-down feature map from the backward pathway with the feature map from the current layer in the forward pathway, and further up-samples the map by a small factor of two, which is then passed down the pathway. Liu et al. [28] designed another type of refinement module, which uses all convolution layers at the same hierarchy to predict edge image at that level, to achieve a similar goal. He et al. [31] introduced a bi-directional cascade structure to enforce each layer to focus on a specific scale by training each network layer with a layer-specific supervision and utilizes dilated convolution to generate multi-scale features. Poma et al. [32] proposed DexiNed that uses the Xception network [33] as the main building block whose output is fed into a up-sampling block to produce an intermediate edge scale space, which is then used to compose a final fused edge-map.

The multi-scale edge fusion approaches used by the aforementioned papers [27], [28], [31], [32] have significantly improved the state-of-arts in edge detection by conducting some kind of spatial enhancement via predicting edges at
different scales and then fusing them together by different techniques. However, one potential deficiency of these approaches is that after fusing the outputs from the upsampled top-level layers with the lower level outputs, the final results usually contain some noise.

In this paper, we propose a novel temporal enhancement approach with a feedback loop, which in contrast to the spatial enhancement approaches of [27], [28], [31], [32], does not explicitly combine multiple level edges (which could result in magnifying the noise). More specifically, we propose a novel Recursive Encoder-Decoder Network with Skip Connections (REDN) for edge detection in natural images. Our encoder-decoder network is formed by DenseNet blocks [34], which are used to alleviate the vanishing-gradient problem, strengthen feature propagation and encourage feature reuse. The encoder network performs convolutions and poolings to produce a set of feature maps of different visual levels. The deeper the layer is, the higher level, more abstract, and less localized the features are. The encoder tends to learn more global and high-level features, and could ignore some finer information. The decoder network, which is topologically symmetric with the encoder network, first upsamples the feature maps by transposed convolutions (i.e. deconvolutions) followed by convolutions and finally returns the edge image with the same size as the input image. Nevertheless, since information related to finer details might be lost during the encoding stage, the decoded outputs are generally less detailed. As a result, edges generated by the encoder-decoder network are usually blurry and less localized [26].

To overcome this limitation, in this paper, we propose to add skip-connections [35] that connect one layer in the encoder to the corresponding layer in the decoder of the same level of hierarchy. Since features from the early encoder are forwarded to the later decoder, skip-connections provide sharper visual details. Skip-connections have been widely used in deep learning community such as U-Net [36], Deep Reflectance Map (DRM) [37], ResNet [38] and DenseNet [34]. According to [34], [35], skip-connections greatly improve gradient flow by allowing more even weight update in all of the layers.

We further enhance the network by adding a feedback loop between the output edge map and the input [39], [40]. The purpose of this is to enable iterative refinement of the edges using a single network model. Increasing recursion depth can improve performance without introducing new parameters for additional convolutions and deconvolutions. The whole network can be modeled jointly with shared parameters and optimized in an end-to-end manner.

Furthermore, in order to force the network to learn more salient edges, we propose a simple but very effective data augmentation scheme by conducting random Gaussian blur to the input images. This also helps to reduce potential over-fitting as the input images are augmented randomly in each iteration during training.

In summary, the main contribution of this paper is to improve the deep learning algorithms used for edge detection by combining skip-connections and feedback loop into an encoder-decoder network, as well as a simple and effective Gaussian blurring based data augmentation. To the best of our knowledge, we are the first group applying the recursive network in low-level tasks such as edge detection. Our REDN experimentally demonstrates state-of-the-art results on popular boundary detection datasets including BSDS500 [1], NYUD [2] and Pascal Context [3].

II. RELATED WORK

The literature of edge detection is very expansive. We will only be able to highlight a few representative works that are closely related to our work.

The early pioneering edge detection methods (e.g. [5], [41]–[47]) focused on low-level cues such as image intensity or color gradients. A complete overview of various low-level edge detectors can be found in [48], [49]. For example, the well-known Canny edge detector [45] finds the peak gradient orthogonal to edge direction. In general, these low-level edge detectors are not very robust and may generate many false positives or false negatives.

In the past decade, people have explored machine learning techniques for more accurate edge detection especially under more challenging conditions [1], [50]–[56]. For example, Dollar et al. [51] used a boosted classifier to independently label each pixel using its surrounding image patch as input. Zheng et al. [52] combined low, mid, and high-level cues to achieve improved results for object-specific edge detection. Arbelaez et al. [1] combined multiple local cues into a global framework based on spectral clustering. Ren and Bo [57] further improved the method of [1] by computing gradients across learned sparse codes of patch gradients. Lim et al. [58] proposed an edge detection approach that classifies edge patches into sketch tokens using random forest classifiers. Sketch tokens are learned using supervised mid-level information in the form of hand drawn contours in images. Dollar and Zitnick [59] learned more subtle variations in edge structure and lead to a more accurate and efficient algorithm. This structured edge detection method was considered one of the best methods for edge detection thanks to its state-of-the-art performance and relatively fast speed.

Recently deep learning approaches become very popular and researchers have attempted to deploy it to edge detection. It is widely believed that accurate detection of edges requires object-level understanding of the image, an area in which deep learning is best known for. Kivinen et al. [60] applied mean-and-covariance restricted Boltzmann machine (mcRBM) architecture [61] to edge detection and obtained competitive results. Starting from candidate contour points from Canny edge detector [45], DeepEdge [62] extracts patches at four different scales and simultaneously run them through the five convolutional layers of the AlexNet [12]. These convolutional layers are connected to two separately-trained network branches. The first branch is trained for classification, while the second branch is trained as a regressor. At testing time, the scalar outputs...
from these two sub-networks are averaged to produce the final score. DeepContour [63] classified image patch of size 45 × 45 into the background or one of the clustered shape classes by a 6-layer convolutional neural network. The disadvantage of both DeepEdge and DeepContour is that at testing time, it operates on the input image in a sliding window fashion (due to the fully-connected layers), which restricts the receptive size of the network to only a small image patch and thus may lose global information.

Inspired by FCN [19], Xie and Tu [25], [64] proposed the HED network which can be trained in an end-to-end manner. An interesting idea of this work is that the final edge map is fused from multiple edge maps obtained at different scales. The multi-scale edge maps are side outputs of a VGG-16 network [13] and hence the shallower edge maps give finer detail edges while the deeper ones capture the more salient edges. The final result is linearly combined from all edge maps at multiple scales. The main drawback of this network is that salient edges are typically learned in the deeper layers, hence they are of low-quality when being up-sampled - edges are blurry and do not stick to actual image boundaries. Later, Kokkinos [10] proposed the Deep-Boundaries network, which is essentially a multi-scale HED [25]. As claimed by Kokkinos [10], the explicit use of multiple scales improves the accuracy of edge detection. However, because being built upon the HED [25] and fed by down-sampled images, Deep-Boundaries also suffers from the same issue as the HED.

To solve the issue of low quality salient edges, Wang et al. [27] and Liu et al. [28] proposed the CED and RCF, respectively. Both papers proposed an extra network to synthesize the high resolution edge maps from low resolution ones instead of trivially using bilinear interpolation. For example, CED’s refinement module fuses a top-down feature map from the backward pathway with the feature map from the current layer in the forward pathway, and further up-samples the map by a small factor (2 ×), which is then passed down the pathway. Maninis et al. [29] proposed the Convolutional Oriented Boundaries (COB) which demonstrated state-of-the-art performance in edge detection. From a single pass of a base convolutional neural network, COB obtains multiscale oriented contours, combines them to build Ultrametric Contour Maps at different scales and finally fuses them into a single hierarchical segmentation structure. He et al. [31] proposed a Bi-Directional Cascade Network to let one layer supervised by labeled edges while adopting dilated convolution to generate multi-scale features.

Poma et al. [32] proposed to integrate the Xception network [33] with a novel upsampling network to fuse features of different scales. Deng et al. [65] proposed to enhance the commonly used weighted cross entropy loss with dice loss to predict sharper boundaries. In [66] an attention model is employed for combining multi-scale features in the context of object contour detection.

Recently there have been new researches focusing on semantic edge detection which aims to simultaneously detect both the object boundaries as well as classify the objects at the same time with deep neural networks. For example, CASENET [30] proposes a category aware semantic edge detection algorithm based on a novel multi-label learning framework where each boundary pixel is labeled into categories of adjacent objects that fuses the category-wise edge activations at the top convolution layer with the bottom layer features using a multi-label loss function. Dynamic Feature Fusion (DFF) of [67] proposes a novel way to leverage multi-scale features. The multi-scale features are fused by weighted summation with fusion weights generated dynamically for each image and each pixel. Meanwhile, Simultaneous Edge Alignment and Learning (SEAL) of [68] deals with severe annotation noise of the existing edge dataset [69]. SEAL treats edge labels as latent variables and jointly trains them to align noisy misaligned boundary annotations. Semantically Thinned Edge Alignment Learning (STEAL) of [70] improves the computation efficiency of edge label alignment through a lightweight level set formulation. In addition, STEAL optimizes the model for non-maximum suppression (NMS) during training while previous works use NMS at the post-processing step. Besides edge detection on single image, recently there have also been researches that utilize the temporal coherence within multi-frames of images or videos [71], [72] to provide better cues for tasks such as edge detection.

Our REDN architecture is based on an encoder-decoder network with significant improvements. Firstly, we use DenseNet blocks within each convolution group. Secondly, we add skip-connections between encoder and decoder, which helps the gradient to more easily reach all the deep layers of a network. Additionally, finer details in the early stage of the encoder are preserved to be used in the decoder. Thirdly, our recursive network is used with convolutions to further increase the network depth with the same number of parameters. In the next section, we will describe our network architecture in depth followed by evaluation results.

III. RECURSIVE ENCODER–DECODER NETWORK WITH SKIP-CONNECTIONS

Fig. 1 shows the architecture of our REDN. Our network takes as input an RGB image and a recursive edge image, concatenates them (in the depth channel) and passes through an encoder-decoder network. The encoder consists of 5 blocks of DenseNet [34]. The decoder is symmetric with the encoder with max-pooling replaced by transposed convolution (i.e. deconvolution). Skip-connections connects corresponding layers of encoder and decoder at the same hierarchy. The decoder outputs an edge image of the same resolution as the input image, which serves as a recursive input to replace the edge image in the network (feedback loop). There are $L$ iterations and $L = 0$ indicates no feedback loop at all. In contrast to DeepEdge [62] and DeepContour [63] which can only be applied on image patch of fixed size due to the use of fully-connected layers, our REDN does not contain any fully-connected layer, and can consume images of any size.
In the following sections, we elaborate the REDN in more details and discuss the training and testing procedures.

### A. TRAINING FORMULATION

We denote our input training dataset by \( S = \{(X_i, Y_i)\}_{i=1}^N \), where \( X_i \) denotes raw input image patch (we use patch size of 256 \( \times \) 256 during all experiments) and \( Y_i \) denotes the corresponding binary ground truth edge map for image patch \( X_i \). The goal of the network is to produce edge maps approaching the ground truth. Let \( W \) be the collection of all network parameters for simplicity. The network runs through \( L \) iterations, each of which produces an edge map \( f^{(l)}(X_i|W) \) \( (l = 0, \ldots, L) \). Thus, \( f^{(L)}(X_i|W) \) is the final output of the REDN. Consequently, the ultimate goal is to minimize the loss between the final edge map and the ground truth, or

\[
\min_W L \left( f^{(L)}(X_i|W), Y_i \right)
\]

where \( L \) is the loss function, a weighted cross-entropy that will be discussed later.

Nevertheless, training such a deep network is not trivial when \( L \geq 1 \). Adapting the idea of deeply supervised network training [25], [64], we also regularize the network by adding multiple losses for all \( f^{(l)}(X_i|W) \). The goal now is to minimize the following.

\[
\min_W \sum_{l=0}^{L} \alpha_l L \left( f^{(l)}(X_i|W), Y_i \right)
\]

where \( \{\alpha_l\}_{l=0}^{L} \) are weights for edge maps at each iteration. We set \( \alpha_l = l + 1 \) to force the network to focus on the edge maps at later iterations.

### B. TESTING FORMULATION

During testing, given image \( X \), we obtain the edge map predictions at all iterations of REDN, i.e. \( f^{(l)}(X|W) \), \( l = 0, \ldots, L \). The final edge map is defined as the last one.

\[
\hat{Y}_{RED} = f^{(L)}(X_i|W)
\]

Alternatively, one may define the final edge map as a weighted combination of all edge maps with learnable weights \( \gamma \) as follows.

\[
\hat{Y}_{REDN} = \frac{\sum_{l=0}^{L} \gamma_l L \left( f^{(l)}(X_i|W), Y_i \right)}{\sum_{l=0}^{L} \gamma_l}
\]

Empirically, when the network is trained properly, we do not notice a significant difference between these two formulations (3) and (4) both visually and quantitatively. Therefore, we opt to use (3) for simplicity.
C. NETWORK ARCHITECTURE

1) ENCODER
The encoder extracts features from the input image, so we need an architecture that is deep and can efficiently generate perceptually multi-level features. Inspired from the recent success of DenseNet [34] on image classification, we design our encoder by stacking 5 DenseNet blocks. The first block consists of two $5 \times 5$ convolution layers with 64 kernels for each, followed by a similar second block with max-pooling layer in between which downsamples the feature maps and hence forces the network to learn good global features. Starting from the third block, we double the number of kernels for each successive block, which results in a 512-dimension feature map after the fifth block. Moreover, we also increase the number of convolution layers to 3, 3 and 4 for the third, fourth and fifth blocks, respectively for more powerful architecture. Every convolution layer in the encoder composes of a convolution layer, a batch normalization layer [73] and a leaky rectified unit activation [74] (with leaking coefficient of 0.1) in this order.

2) DECODER
The decoder maps the learned features to another space and eventually reaches the edge image. This network is symmetric with the encoder with 5 DenseNet blocks. We use transposed convolutions (or deconvolutions) to upsample the feature maps corresponding to max-pooling in the encoder. The transposed convolutions are initialized as bilinear filters which purely serve as upsample filters. At the last layer, the decoder returns an edge prediction from the 64-channel layer via convolution. To facilitate the training, we also use batch normalization and leaky rectified units in the same way as in the encoder except for the last layer which only consists of a convolution followed by a sigmoid activation.

3) SKIP-CONNECTIONS
The encoder progressively extracts and down-samples features, while the decoder upsamples and combines them to construct the output. The sizes of feature maps are exactly mirrored in our network. We concatenate early encoded features (from the encoder) to the corresponding decoded features (from the decoder) at the same spatial resolution, in order to obtain local sharp details preserved in early encoder layers. There are four of such skip-connections corresponding to four different levels of hierarchies, which are called mirror-links. Mirror-link is a form of skip connection which has been proven effective in many deep networks such as ResNet [38] and DenseNet [34]. Besides the sharpness, these skip-connections could also regulate gradient flow and allow better trained networks.

4) FEEDBACK LOOP
This is a recursive connection similar to the Recurrent Neural Network (RNN). In contrast to RNN in which recurrence targets temporal sequence and tries to learn temporal changes, our feedback loop refines the edge map progressively without introducing more network parameters. At the beginning, there is no edge image generated, so the initial edge image is set as a blank image (i.e. zero-image). After the first pass through the encoder-decoder network, the output edge map is recursively fed back to the input and repeatedly processed through the shared encoder-decoder network. The whole REDN is jointly optimized in an end-to-end manner. Due to the memory limit, we only conduct experiments for $L = 2$.

D. LOSS FUNCTION
We use weighted sigmoid cross-entropy function $L$ compute the loss between our predicted edge $\hat{Y}_{\text{REDN}}$ (or other intermediate edge images $f^{(l)}(X_l|W), l = 0, \ldots, L$) and the ground truth edge image $Y$ as follows.

$$L = -(1 - \beta) \sum_{j \in Y_+} \log \hat{Y}_{\text{REDN}} - \beta \sum_{j \in Y_-} \log (1 - \hat{Y}_{\text{REDN}})$$ (5)

where $Y_+$ and $Y_-$ denote edge and non-edge pixels, respectively and $\beta = \frac{|Y_+|}{|Y|}$ to balance the relative importance of these two classes.

E. IMPLEMENTATION
We implement our framework using the publicly available TensorFlow [75].

1) HYPER-PARAMETERS
In contrast to fine-tuning CNN for image classification, adapting CNN for pixel-wise output requires special care. Even with the proper initialization or a pre-trained model, sparse ground truth distributions coupled with conventional loss functions lead to difficulties in network convergence. Through experimentation, we choose the following hyper-parameters: mini-batch size of 8, convolutional filters randomly initialized by Gaussian distribution with zero-mean and standard deviation of 0.01, convolutional biases all zero-initialized, deconvolutions initialized as bilinear filters, weight decay of $10^{-6}$, training epochs equal 500. Furthermore, we use Adam optimizer [76] with an initial learning rate of $10^{-4}$. As mentioned earlier, we extract image patches of size $256 \times 256$ for training but use the whole image during testing.

2) DATA AUGMENTATION
Data augmentation has proven to be a crucial technique in training deep neural networks. For each training image, we randomly sample 500 patches, each of size $256 \times 256$, which is a kind of image cropping. We further randomly flip the training image horizontally. These together lead to an augmented training set that is a factor of 500 times larger than the unaugmented set.

In addition, we add random Gaussian noise (black-and-white noise) to the training images by sampling from a Gaussian distribution with zero-mean and standard deviation of 20 (assuming image intensities are within $[0, 255]$).
FIGURE 2. Original images (top row) are augmented with Gaussian noise to force the network to extract stronger edges.

TABLE 1. Datasets and parameters.

| Dataset          | # train | # test | maxDist |
|------------------|---------|--------|---------|
| BSDS500 [1]      | 300     | 200    | 0.0075  |
| NYUD-v2 [2]      | 795     | 654    | 0.011   |
| Pascal Context [3]| 7605    | 2498   | 0.0075  |

TABLE 2. BSDS500 [1] test evaluation.

| Method          | Year | ODS   | OIS   | AP    |
|-----------------|------|-------|-------|-------|
| Canny [45]      | 1986 | 0.600 | 0.640 | 0.580 |
| MShift [77]     | 2002 | 0.601 | 0.644 | 0.493 |
| EGB [78]        | 2004 | 0.610 | 0.640 | 0.560 |
| gPb-ovt-ucm [1] | 2011 | 0.726 | 0.757 | 0.696 |
| SCG [57]        | 2012 | 0.739 | 0.758 | 0.773 |
| ISICRA [79]     | 2013 | 0.724 | 0.752 | 0.783 |
| Sketch Tokens [58]| 2013 | 0.727 | 0.746 | 0.780 |
| DeepNets [60]   | 2014 | 0.738 | 0.759 | 0.758 |
| MCG [80]        | 2014 | 0.747 | 0.779 | 0.759 |
| SE [59]         | 2015 | 0.746 | 0.767 | 0.803 |
| OEF [81]        | 2015 | 0.749 | 0.772 | 0.817 |
| LEP [82]        | 2015 | 0.757 | 0.793 | 0.828 |
| N4-Fields [83]  | 2015 | 0.753 | 0.769 | 0.784 |
| DeepEdge [62]   | 2015 | 0.753 | 0.772 | 0.807 |
| CSCNN [84]      | 2015 | 0.756 | 0.775 | 0.798 |
| DeepContour [63]| 2015 | 0.756 | 0.773 | 0.797 |
| CEDN [26]       | 2016 | 0.788 | 0.804 | 0.834 |
| HED-fusion [64] | 2017 | 0.782 | 0.804 | 0.833 |
| HED-late-merging [64]| 2017 | 0.788 | 0.808 | 0.840 |
| COB [29]        | 2017 | 0.793 | 0.820 | 0.859 |
| CED [27]        | 2017 | 0.803 | 0.820 | 0.871 |
| RCF [28]        | 2017 | 0.806 | 0.823 | –     |
| AMH-Net [66]    | 2017 | 0.798 | 0.829 | 0.869 |
| LPCB [65]       | 2018 | 0.800 | 0.816 | –     |
| BDCN [31]       | 2019 | 0.806 | 0.826 | 0.847 |
| DextINed [32]   | 2020 | 0.729 | 0.745 | 0.583 |
| REDN (ours)     | 2020 | 0.808 | 0.828 | 0.827 |

(Fig. 2). This data augmentation forces the network to learn the stronger edges such as object contours over finer texture ones. This augmentation also helps combat over-fitting because each training image is augmented differently in each iteration.

3) RUNNING TIME

Training ranges from 4 hours for the BSDS500 dataset with 300 images to 50 hours for the Pascal Context dataset with 7605 images on a single Titan-X GPU. REDN produces an edge response for an image of size 512 × 512 in about 270 milliseconds including interface overhead (e.g. image loading), which is approximately 3.4 frames/second. This is significantly more efficient than existing CNNs such as DeepEdge [62], DeepContour [63] and COB [29].

IV. EVALUATION

This section presents the performance of our REDN on the well-known datasets for edge detection such as BSDS500 [1], NYUD [2] and Pascal Context [3] (see Table 1). We adopt three standard evaluation metrics commonly used for edge detection, fixed contour threshold Optimal Dataset Scale (ODS) F-measure, per-image best threshold Optimal Image Scale (OIS) F-measure, and Average Precision (AP) [1]. Optimal Dataset Scale (ODS) F-measure is the best F-measure on the dataset for a fixed scale. It corresponds to the most outward point on the precision-recall curve. Optimal Image Scale (OIS) F-measure is the aggregate F-measure on the dataset for the best scale in each image. The OIS generally reports a better performance because the F-measure is computed by aggregating per-image counts that correspond to those that give the best F-measure for each image. Average Precision (AP) is the Average Precision on the full recall range (equivalently, the area under the precision-recall curve). F-measure is the harmonic mean of precision and recall and is defined as $\frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$. We compare our method against popular state-of-the-art methods including both the non-deep learning and deep learning approaches. For a fair quantitative comparison, we apply a standard non-maximal suppression technique [59] to all edge maps generated by all methods to obtain thinned edges before evaluation.

A. DATASETS

We evaluate our algorithm on BSDS500 [1], NYUD [2] and Pascal Context [3] datasets using standard metrics such as
ODS/OIS F-measure and AP. The BSDS500 dataset has edge annotation ground truth while the others do not. The NYUD and Pascal Context datasets are primarily for semantic segmentation. To obtain the ground truth edges, we first identify all the boundary pixels, treat them as a binary image and then apply image thinning using MATLAB function bwmorph (see examples in Fig. 3).

1) BSDS500
The Berkeley Segmentation Dataset and Benchmark (BSDS500) [1] consists of 200 training, 100 validation and 200 testing images. We use the training and validation sets (300 images) for training our REDN. Each colored image is of size $481 \times 321$ or $321 \times 481$ and is manually annotated ground truth contours. We simply overlay all annotations followed by image thinning to obtain a single ground truth image. Unlike other image-to-image deep learning frameworks such as HED [25] which resizes the input image to a fixed size of $400 \times 400$, our REDN runs on the original image without resizing. We use padding to make image dimension fit after convolutional, pooling and deconvolutional layers and crop the output to get the result of the original dimension.

2) NYUD
The NYUD dataset [2], was used for edge detection in [57], [85], has 1449 RGB-D images of indoor scenes (which are quite different from outdoor scenes of the BSDS500 [1]). As a result, it is more challenging because the edges are more cluttered and there are more variations. Here we use the setting described in [59] and evaluate our REDN on data processed by [85]. The NYUD dataset is split into 795 training and 654 testing images. These splits are carefully selected such that images from the same scene are only in one of these sets. All images are of size $640 \times 480$. This dataset also has depth image and although our REDN is easily extensible to RGB-D image, we do not use this information for our experiment. HED [64] has three networks accepting RGB, depth encoded HHA [85] and RGB-HHA, respectively. Consequently, we include the results of all these three network versions in Table 3. For a fair comparison, during evaluation we increase the maximum tolerance allowed for correct
matches of edge predictions to ground truth from 0.0075 to 0.011 as used in [59], [64], [85].

3) PASCAL CONTEXT
The Pascal Context dataset [3] contains carefully localized pixel-wise semantic annotations for the entire image on the PASCAL VOC 2010 detection trainval set. It contains 10103 images, which is approximately 20 times larger than the BSDS500 dataset, spanning over 459 semantic categories. Images in this dataset have various sizes and are quite challenging due to the increased scene complexities.
FIGURE 5. The recursive network ($L = 2$) improves the results of the encoder-decoder network with skip-connections ($L = 0$) by cleaning noisy edges and enhancing stronger ones.

TABLE 3. NYUD-v2 [2] test evaluation.

| Method         | Year | ODS | OIS | AP  |
|----------------|------|-----|-----|-----|
| gPb-owt-ucm [1]| 2011 | 0.726 | 0.757 | 0.696 |
| Silberman et al. [2] | 2012 | 0.658 | 0.661 | n/a  |
| MCG-B [80]      | 2014 | 0.652 | 0.681 | 0.613 |
| SE [59]         | 2015 | 0.685 | 0.699 | 0.679 |
| HED-RGB [64]    | 2017 | 0.720 | 0.734 | 0.734 |
| HED-HHA [64]    | 2017 | 0.682 | 0.695 | 0.702 |
| HED-RGB-HHA [64]| 2017 | 0.746 | 0.761 | 0.786 |
| ResNet50-RGB-HHA [29] | 2017 | 0.745 | 0.762 | 0.792 |
| ResNet50-RGB [29] | 2017 | 0.746 | 0.761 | 0.789 |
| ResNet50-RGB-D [29] | 2017 | 0.683 | 0.699 | 0.681 |
| RCF-RGB [28]    | 2017 | 0.729 | 0.742 | 0.794 |
| RCF-RGB-HHA [29] | 2017 | 0.757 | 0.771 | 0.800 |
| COB-PC [29]     | 2017 | 0.710 | 0.735 | 0.734 |
| COB-RGB [29]    | 2017 | 0.778 | 0.799 | 0.814 |
| COB-RGB-HHA [29]| 2017 | 0.784 | 0.805 | 0.825 |
| AMH-Net-RGB [66]| 2017 | 0.744 | 0.758 | 0.765 |
| AMH-Net-HHA [66]| 2017 | 0.716 | 0.729 | 0.734 |
| AMH-Net-RGB-HHA [66]| 2017 | 0.771 | 0.786 | 0.802 |
| LPCB-RGB [65]   | 2018 | 0.739 | 0.754 | 0.765 |
| LPCB-HHA [65]   | 2018 | 0.707 | 0.719 | 0.730 |
| LPCB-RGB-HHA [65]| 2018 | 0.762 | 0.778 | 0.792 |
| BDCN-RGB [31]   | 2019 | 0.748 | 0.763 | 0.770 |
| BDCN-HHA [31]   | 2019 | 0.707 | 0.719 | 0.731 |
| BDCN-RGB-HHA [31]| 2019 | 0.765 | 0.781 | 0.813 |
| DexNed [32]     | 2020 | 0.658 | 0.674 | 0.556 |
| REDN (ours)     | 2020 | 0.793 | 0.813 | 0.832 |

B. VISUAL COMPARISON

Fig. 4 shows side-by-side comparison between different boundary detection algorithms. As we can see, the non-deep learning methods such as SE [59] and gPb-owt-ucm [1] produce sharp and clean edges in areas with high-contrast but fail at low-contrast regions because they only use local features and thus do not have a object-level understanding.

The HED [25], which uses the features from VGG-16 network [13], performs much better and is able to capture objects even in low-contrast cases and it is not easily confused by object’s interior boundary. Its weakness remains in the blurry and less localized edge responses, which may prevent it from recovering the sharp details.

Our REDN’s results are generally cleaner, sharper and more accurate. Additionally, our results capture more global boundaries. For example, in the airplane image (second to last row of Fig. 4), only the most salient edges of the plane are retained.

Fig. 5 illustrates the benefits of the feedback loop in our network. In a pure encoder-decoder network without feedback loop (i.e. $L = 0$), the results are blurry in the fine texture regions. However, with the recursive network, these errors are cleaned up and salient edges are enhanced.

C. QUANTITATIVE COMPARISON

For numerical comparison, Table 2 shows the F-measure of various edge detection algorithms on BSDS500 dataset. It is obvious that our REDN is better than other methods regarding ODS/OIS F-measure with $ODS = 0.808$, $OIS = 0.828$ while providing reasonable $AP = 0.827$. Table 3 provides the numerical statistics of the tested algorithms on the NYUD dataset. As we can see, although our REDN only takes as input RGB image and ignores depth information, it is still better than HED [64], RCF [28] and COB [29] network which relies on both RGB and depth information. We set a new state-of-the-art edge detection on the NYUD dataset at $ODS = 0.793$, $OIS = 0.818$ and $AP = 0.832$. 
The Pascal Context dataset is significantly larger and more challenging than the first two. From Table 4, without recursive network (i.e. $L = 0$), our REDN will be similar to CEDN [26], except the skip-connections. As we can see from the table, skip-connections boost the ODS F-measure from 0.702 to 0.744, which is a huge improvement from CEDN even though it is still marginally behind COB [29]. However, with feedback loop (i.e. $L = 2$), REDN edges out COB to achieve ODS F-measure of 0.759. Furthermore, with our novel data augmentation of adding random Gaussian noise, REDN manages to push the results a little further at ODS = 0.761, OIS = 0.785 and AP = 0.787.

### TABLE 4. Pascal context [3] test evaluation.

| Method | Year | ODS  | OIS  | AP  |
|--------|------|------|------|-----|
| MCG-B [80] | 2014 | 0.554 | 0.609 | 0.528 |
| SE [59] | 2015 | 0.533 | 0.568 | 0.496 |
| IEP-B [82] | 2015 | 0.570 | 0.636 | 0.547 |
| HED [25] | 2015 | 0.688 | 0.707 | 0.704 |
| CEDN [26] | 2016 | 0.702 | 0.718 | 0.744 |
| COB [29] | 2017 | 0.750 | 0.781 | 0.773 |

**REDN ($L = 0$) (no data aug.)**

| Method | Year | ODS  | OIS  | AP  |
|--------|------|------|------|-----|
| REDN ($L = 0$) (no data aug.) | 2020 | 0.744 | 0.769 | 0.771 |
| REDN ($L = 2$) (no data aug.) | 2020 | 0.759 | 0.784 | 0.784 |
| REDN ($L = 2$) (with data aug.) | 2020 | 0.761 | 0.785 | 0.787 |

### TABLE 5. Cross-dataset performance of REDN.

| Train | Test | ODS  | OIS  | AP  |
|-------|------|------|------|-----|
| Pascal Context | BSDS500 | 0.755 | 0.781 | 0.828 |
|             | NYUD  | 0.732 | 0.766 | 0.783 |
| BSDS500 | Pascal Context | 0.643 | 0.653 | 0.681 |
|             | NYUD  | 0.627 | 0.653 | 0.703 |

### D. CROSS-DATASET EVALUATION

To further demonstrate the generalization capability of our network, we train our model with one dataset and test it with another dataset. Table 5 shows the performance of our method on BSDS500, NYUD and Pascal Context datasets. The performance of a pretrained model is expected to be lower than that of a fine-tuned one. Our pretrained model yields a high precision but a low recall due to its object-selective nature between any two datasets. Furthermore, since the BSDS500 dataset is pretty small and less diversified than the other two, the model trained on it results in bigger drops in performance when tested on NYUD and Pascal Context.

### V. CONCLUSION

We have proposed a method to substantially improve deep learning-based boundary detection performance. Our REDN adds skip-connections into the encoder-decoder network, which sharpens and preserves more details at the later layers, and a feedback loop, which allows progressive improvement of the edge image. We propose a novel data augmentation scheme using Gaussian blurring that can force the network to learn more salient edges as well as reduce the potential overfitting. Our system is fully end-to-end trainable and operates in approximately 3.4 frames per second, a speed of practical relevance. As measured on the standard datasets such as BSDS500, NYUD and Pascal Context, our REDN performs very well in comparison with the state-of-the-art approaches.

The recurrent network enables us to increase the network depth without increasing the number of parameters, however since we need to store all intermediate results during feedback loops it does require a higher memory consumption. To overcome this limitation, in the future we plan to integrate the recently proposed light weight memory efficient networks such as MobileNetV1 [86], MobileNetV2 [87], SqueezeNet [88], ShuffleNet [89], etc into our framework. Another future direction we would pursue is currently we are predicting a single edge probability per pixel. We would like to extend our framework to automatically predict the edge probabilities per pixel for 8 directions, as is done in [1] and [90], but using a deep learning framework instead. To further improve the performance, we could follow [31] by conducting transfer learning to first pretrain the network with a large-scale dataset then fine-tune it with the benchmark dataset. We could also make use of multi-scale edge detection during testing as is done in [65] by resizing an input image to several different resolutions and feed them into the network, then resize the outputs back to the original size and average them to obtain the final prediction. Beside edge detection, we would also like to apply the proposed framework for other image processing and computer vision tasks such as semantic edge detection and image segmentation.

### ACKNOWLEDGMENT

The authors would like to thank the authors of BSDS500 data set [1], NYUD data set [2], and Pascal Context data set [3] for providing the benchmark data as well as evaluation toolbox. They would also like to thank the authors of all edge detection algorithms [1], [10], [45], [57]–[60], [62]–[64], [78], [81], [83], [84] for providing source code and/or benchmark results for comparison.

### REFERENCES

[1] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, “Contour detection and hierarchical image segmentation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, no. 5, pp. 898–916, May 2011.

[2] P. K. Nathan Silberman, D. Hoiem, and R. Fergus, “Indoor segmentation and support inference from RGBD images,” in Proc. Eur. Conf. Comput. Vis., 2012, pp. 746–760.

[3] R. Motlagh, 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 Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2014, pp. 891–898.

[4] Y. Chien, “Pattern classification and scene analysis,” IEEE Trans. Autom. Control, vol. AC-19, no. 4, pp. 462–463, Aug. 1974.

[5] J. R. Fram and E. S. Deutsch, “On the quantitative evaluation of edge detection schemes and their comparison with human performance,” IEEE Trans. Comput., vols. C-24, no. 6, pp. 616–628, Jun. 1975.

[6] G. S. Robinson, “Color edge detection,” Opt. Eng., vol. 16, no. 5, 1977, Art. no. 165479.

[7] S. Ullman and R. Basri, “Recognition by linear combinations of models,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 13, no. 10, pp. 992–1006, Oct. 1991.

[8] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid, “Groups of adjacent contour segments for object detection,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 30, no. 1, pp. 36–51, Jan. 2008.
[9] J. Malik, S. Belongie, T. Leung, and J. Shi, “Contour and texture analysis for image segmentation,” Int. J. Comput. Vis., vol. 43, no. 1, pp. 7–27, 6 2001.

[10] I. Kokkinos, “Pushing the boundaries of boundary detection using deep learning,” in Proc. Int. Conf. Learn. Represent., 2016, pp. 1–12.

[11] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes: Active contour models,” Int. J. Comput. Vis., vol. 1, no. 4, pp. 321–331, Jan. 1988.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2012, pp. 1097–1105.

[13] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in Proc. Int. Conf. Learn. Represent., 2014, pp. 1–14.

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 1–9.

[15] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[16] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, “Learning hierarchical features for scene labeling,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 8, pp. 1915–1929, Aug. 2013.

[17] A. Sharma, O. Tuzel, and D. W. Jacobs, “Deep hierarchical parsing for semantic segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 530–538.

[18] 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 Proc. Int. Conf. Learn. Represent., 2015, pp. 1–14.

[19] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 3431–3440.

[20] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A deep convolutional encoder-decoder architecture for image segmentation,” 2015, arXiv:1511.00561. [Online]. Available: http://arxiv.org/abs/1511.00561

[21] H. Noh, S. Hong, and B. Han, “Learning deconvolution network for semantic segmentation,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 1520–1528.

[22] D. Eigen and R. Fergus, “Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 1463–1472.

[23] X. Wang, D. F. Fouhey, and A. Gupta, “Designing deep networks for surface normal estimation,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 2650–2658.

[24] X. Wang, D. F. Fouhey, and A. Gupta, “Designing deep networks for surface normal estimation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 539–547.

[25] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell, “Long-term recurrent convolutional networks for visual recognition and description,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 7, pp. 1713–1726, Jul. 2013.

[26] S. Xie and Z. Tu, “Hierarchical edge detection,” in Proc. IEEE Trans. Pattern Anal. Mach. Intell., vol. 38, no. 2, pp. 261–274, Feb. 2016.

[27] S. Zheng, Z. Tu, and A. L. Yuille, “Detecting depth, surface normals and semantic labels with a common multi-scale convolutional architecture,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 1463–1472.

[28] T. Le, Y. Duan: REDN: A Recursive Encoder-Decoder Network for Edge Detection, 2016, pp. 4508–4516.

[29] J. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.

[30] P. Dollar and P. Hart, Pattern Classification Scene Analysis. Hoboken, NJ, USA: Wiley, 1973.

[31] D. Marr and E. Hildreth, “Theory of edge detection,” Proc. Roy. Soc. London, B, Biol. Sci., vol. 207, pp. 187–217, Feb. 1980.

[32] D. R. Martin, C. C. Fowlkes, and J. Malik, “Learning to detect natural image boundaries using local brightness, color, and texture cues,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 26, no. 5, pp. 530–549, May 2004.

[33] P. Dollar, Z. Tu, and S. Belongie, “Supervised learning of edges and object boundaries,” in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR), vol. 2, Jun. 2006, pp. 1964–1971.

[34] S. Zheng, Z. Tu, and A. L. Yuille, “Detecting object boundaries using low-, mid-, and high-level information,” in Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit., Jun. 2007, pp. 1–8.

[35] J. Mairal, M. Leordeanu, B. Bach, M. Hebert, and J. Ponce, “Discriminative sparse image models for class-specific edge detection and image interpretation,” in Proc. IEEE Eur. Conf. Comput. Vis., Oct. 2012, pp. 178–193.

[36] N.立面ski and M. Mignotte, “A particle filter framework for contour detection,” in Proc. IEEE Eur. Conf. Comput. Vis., Oct. 2012, pp. 780–783.

[37] M. Leordeanu, R. Sukthankar, and C. Schtincheuz, “Generalized boundaries from multiple image interpretations,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 36, no. 7, pp. 1312–1324, Jul. 2014.

[38] X. Ren and L. Bo, “Discriminatively trained sparse code gradients for edge detection,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2012, pp. 584–592.

[39] J. J. Lim, C. L. Zitnick, and P. Dollar, “Sketch tokens: A learned mid-level representation for contour and object detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2013, pp. 3158–3165.
P. Lei, F. Li, and S. Todorovic, “Boundary flow: A siamese network that predicts boundary motion without training on motion,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 1732–1740.

Q. Zhao, “Segmenting natural images with the least effort as humans,” in Proc. Brit. Mach. Vis. Conf., Mar. 2015, p. 110.

Y. Ganin and V. Lempitsky, “N4-fields: Neural network nearest neighbor fields for image transforms,” in Proc. IEEE Asian Conf. Comput. Vis., May 2015, pp. 536–551.

J.-J. Hwang and T.-L. Liu, “Pixel-wise deep learning for contour detection,” in Proc. IEEE Int. Conf. Learn. Represent., 2015, pp. 1–2.

S. Gupta, R. Girshick, P. Arbelaez, and J. Malik, “Learning rich features from RGB-D images for object detection and segmentation,” in Proc. IEEE Eur. Conf. Comput. Vis., Oct. 2014, pp. 345–360.

A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “MobileNets: Efficient convolutional neural networks for mobile vision applications,” 2017, arXiv:1704.04861. [Online]. Available: http://arxiv.org/abs/1704.04861

M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 4510–4520.

F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size,” 2016, arXiv:1602.07360. [Online]. Available: http://arxiv.org/abs/1602.07360

X. Zhang, X. Zhou, M. Lin, and J. Sun, “ShuffleNet: An extremely efficient convolutional neural network for mobile devices,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 6848–6856.

N. Alpaslan and K. Hanbay, “Multi-resolution intrinsic texture geometry-based local binary pattern for texture classification,”IEEE Access, vol. 8, pp. 54415–54430, 2020.