Multi-scale Processing of Noisy Images using Edge Preservation Losses

Nati Ofir
BIU
Ramat Gan, Israel
Email: natiofir@gmail.com

Yosi Keller
BIU
Ramat Gan, Israel
Email: yosi.keller@gmail.com

***Abstract*—Noisy image processing is a fundamental task of computer vision. The first example is the detection of faint edges in noisy images, a challenging problem studied in the last decades. A recent study introduced a fast method to detect faint edges in the highest accuracy among all the existing approaches. Their complexity is nearly linear in the image's pixels and their runtime is seconds for a noisy image. Their approach utilizes a multi-scale binary partitioning of the image. By utilizing the multi-scale U-net architecture, we show in this paper that their method can be dramatically improved in both aspects of run time and accuracy. By training the network on a dataset of binary images, we developed an approach for faint edge detection that works in linear complexity. Our runtime of a noisy image is milliseconds on a GPU. Even though our method is orders of magnitude faster, we still achieve higher accuracy of detection under many challenging scenarios. In addition, we show that our approach to performing multi-scale preprocessing of noisy images using U-net improves the ability to perform other vision tasks under the presence of noise. We prove it on the problems of noisy objects classification and classical image denoising. We show that multi-scale denoising can be carried out by a novel edge preservation loss. As our experiments show, we achieve high-quality results in the three aspects of faint edge detection, noisy image classification, and natural image denoising.**

I. Introduction

Edge detection is one of the fundamental problems of computer vision. Many works addressed this problem and introduced a variety of solutions. Unfortunately, some imaging domains suffer from faint edges and noisy images, such as medical, satellite, and even real natural images. The detection of edges under such challenging conditions should be implemented by methods geared to that end. Existing approaches that deal with a high level of noise are all relatively slow (runtime of seconds for an image).

This work is the first to use deep learning to detect faint edges, denoted as Faint-Edges-Detection CNN (FED-CNN). By training the FED-CNN [23] on a simulated faint-edges dataset [16], we developed a novel approach for edge detection in noisy images. Since a forward pass of a network can be optimized on a GPU, our algorithm is real-time and is orders of magnitude faster than existing approaches. Denote by N the image pixels, then our runtime is $O(N)$, while the FastEdges state-of-the-art approach [22] runs in $O(NlogN)$. Even though, our experiments demonstrate that it is yet more accurate in the task of binary edge detection at low Signal-to-Noise-Ratios (SNR), where the SNR is the ratio of the edge contrast c to the noise level σ, such that $SNR = \frac{c}{\sigma}$. Edge detection of a medical noisy image is depicted in Figure 1.

(a) (b) (c)

Fig. 1. Example of a medical image with many curved edges. (a) The original image. (b) The proposed FED-CNN approach results. (c) FastEdges [22] results. Both methods achieve high quality of detection while ours run in milliseconds and FastEdges runtime is more than seconds.

(a) (b) (c)

Fig. 2. Denoising result at additive noise of 50 standard deviation, of the proposed multi-scale network trained by our edge preservation loss. (a) The noisy input image. (b) The results of the proposed scheme. (c) Denoising results of the state-of-the-art DnCNN [34] approach. Our method achieves the highest SSIM [31] scores in our experiments at all the noise levels.

The similarity between the classical vision approach of FastEdges [22] and our method FED-CNN (Faint-Edge-Detection-Convolutional-Neural-Network), is that both tackle noisy edge detection by utilizing multi-scale preprocessing of the image. Ofir et al. [22] utilize a binary partitioning tree of the image, into sub-areas, and compute edge-filter responses at every sub-rectangle of image pixels and concatenate curves from each sub-rectangle using dynamic programming like approach. The proposed denoising scheme aims to mimic its multi-scale filters using a convolutional neural network (CNN). We show that the multi-scale processing of the image, using a CNN can be carried out by the U-Net architecture. In addition to faint edge detection, we show that our approach can be applied to additional computer vision tasks, such as to improves the performance of a classifier trained for noisy image classification. Specifically, we show that the accuracy of the Resnet20 [28] classifier, on a noisy CIFAR10 [15] dataset, is increased. This emphasizes the importance of multi-scale filters and preprocessing to perform vision tasks at low SNRs.

We also apply an edge detector as an auxiliary loss for image denoising. We train U-Net to perform denoising, and by that developing a deep-multi-scale algorithm at a state-of-the-art level. We use for training a novel architecture that utilizes an edge preservation auxiliary loss. Our results of denoising are excellent in the perceptual measurement of Structure-of-Similarity (SSIM) and Peak-Signal-to-Noise-Ratio (PSNR). See Figure 2 for example of our denoising approach relative to the state-of-the-art DnCNN [34]. We managed to remove the noise and preserve the signals in the highest quality among the existing denoising algorithms.

II. Previous Work

Edge detection is a fundamental problem in image processing and computer vision with a plethora of related works. Marr and Hildreth [19] studied edge detection using the zero crossings of the 2D Laplacian applied to an image, while Sobel [10] proposed to applying a 3×3 derivative filter on an image, and computing the gradients. Canny [5] extends Sobel by hysteresis thresholding of the local gradients. These classical approaches are very fast, but unfortunately very sensitive to image noise and cannot accurately detect faint edges. The advanced group of works is focused on the problem of boundary detection and segmentation [1], [8], [13], achieving accurate results when applied to the Berkeley Segmentations Dataset (BSDS500) [20]. A recent class of works is optimized and trained on this dataset utilizing deep learning tools [18], [32], [33]. Even though such approaches perform well for boundary detection, their accuracy degrades in the presence of noise as was shown by Ofir at al. [22].

The particular problem of faint edge detection in noisy images was addressed by Galun et al. [9] by detecting faint edges using the difference of oriented means. They applied a matched filter that averages along each side of the edge and maximizes the contrast across the edge. Their method is limited to straight lines, with a computational complexity of $O(NlogN)$ where N is the number of pixels in an image. Ofir et al. [22] extended this work to curved edges utilizing dynamic programming and approximations, to achieves better accuracy at a complexity of $O(NlogN)$. In practice, the run time of these methods on a noisy image is seconds. Sub-linear approaches [12], [30] were introduced for detecting straight and curved edges.

The proposed scheme utilizes deep-learning to improve these results, and we introduce a $O(N)$ algorithm, whose actual runtime is negligible due to GPU acceleration. Although our method is faster, we achieve even more accurate results when detecting faint edges in noisy images. We utilize the U-Net architecture [23] that was first derived for biomedical image segmentation. We show that due to its multi-scale processing, it allows us to perform other vision tasks under hard conditions of low signals and high noise.

Classification is a fundamental task in machine learning and computer vision. Early methods to train a classifiers utilized Support-Vector-Machines (SVM) [27] and logistic-regression [11]. Deep learning approaches proved superior in classification accuracy. The Resnet CNN architecture [28] emphasized the importance of residual connections in classification neural networks. These networks are also the foundation of object detection and localization as described in Single-Shot-Multibox-Detector (SSD) [17]. All the above approaches suffer from the presence of noise and objects at low SNRs. We use as an example the Resnet20 [28] classifier and CIFAR10 dataset [15] to exemplify the importance of a multi-scale preprocessing of the noisy image to produce clean heat-maps. As done with faint edges, we use the U-Net [23] architecture for this preprocessing.

Image denoising is one of the most studied areas of image processing and computer vision. Early methods rely on the Wavelet Transform [21]. Advanced methods are based on patch repetitions in the image like Non-Local-Means (NLM) [3] and Block-Matching and 3D-filtering (BM3D) [7]. Recent methods utilize CNNs for denoising [34], [25]. In this work, we show that the multi-scale processing of U-Net [23] is useful for handling noisy images, and to produce competitive denoising results on noisy natural images. It is carried out by a novel architecture that utilizes edge preservation as an auxiliary loss.

III. Faint Edges Detection in Noisy Images

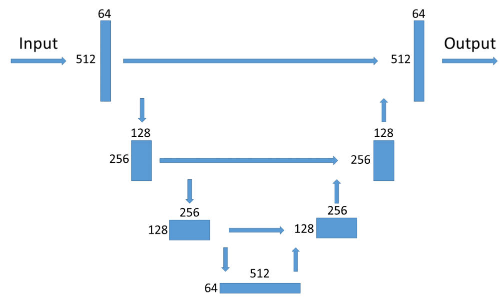

Fig. 3. The multi-scale architecture of U-net used in this work. The network downscaled the activation map by a series of max-pooling layers in the first half, and upsampled by interpolation layers in its second half. There are concatenation connections between the first half and the second half for every two layers having the same number of channels.

We propose the Faint-Edges-Detection CNN (FED-CNN) to formulate the detection of faint edges in noisy images as binary image segmentation, based on the architecture that is depicted in Figure 3 and described in Table I. This FED-CNN allows creating a multi-scale algorithm for detection and segmentation, where sigmoid activations are appended on top of the last layer. The complexity analysis of FED-CNN is linear $O(N)$ in image pixels, since a convolution with a 3×3 filter is linear, and the number of such convolutions of any input image is constant. The speedup of FED-CNN over FastEdges [22] is mostly because of hardware optimization on a GPU since the theoretical complexity is similar: linear for FED-CNN and nearly linear for FastEdges [22]. The proposed FED-CNN was trained using the Dice coefficients. Denote by

y' the binary labels of the edges, and by y the output of the network for a given input x, the Dice coefficient is given by

$$Di(y, y') = -\frac{\sum_p y'(p) \cdot y(p)}{\sum_p y'(p) + \sum_p y(p)}, \quad (1)$$

where p is an image pixel. This loss encourages accurate detections, while penalizes false alarms.

We initialize the FED-CNN using random weights, and used a dataset of 1406 binary images to train the network [16]. For each ground truth image we apply a Canny edge detector [5] to extract the labels Y', and used to create a set of noisy images having different signal to noise ratios (SNR's)

$$I = clip(0.1 * (snr \cdot I_c + I_n) + 0.45). \quad (2)$$

I is the noisy image, input to our network, I_c is the original binary image, I_n is a random Gaussian noise image with standard deviation of 1, $\forall p : I_n(p) \sim N(0, 1)$. The snr is the measure of the faintness of the edges, each binary image creates six training images $snr = [1, 1.2, ..., 2]$. The $clip(\cdot)$ clips the pixel values to $[0, 1]$.

We trained the FED-CNN for 100 epochs and augmented the dataset [16] by different SNRs and horizontal and vertical flipping of the images. Moreover, we added samples of a pure noise image with no labels. Our dataset after augmentations contains $\sim$17,000 examples. We split the dataset to 90% training and 10% testing.

IV. Classification of Noisy Images

In this section, we apply the proposed FED-CNN architecture to the classification of noisy images, using a Resnet20 CNN [28]. We apply noisy image classification by two steps: denoising the natural images by IDCNN (Image-Denoisnig-CNN) and then classifying using Resnet20. We found out that the best way to train this scheme is by an end-to-end approach. Denote by x an input images, in CIFAR10 dataset [15] $x_i \in \mathbb{R}^{32\times32\times3}$. Given the Resnet classifier, the label is given by $y_i = resnet(x_i)$. We trained this model using the CIFAR10 dataset and achieved a classification accuracy of 91.66%. Denote the Resnet network trained on the regular clean CIFAR10 as $resnet_c$. We created a noisy version of CIFAR10, by adding Gaussian noise of different standard deviations to the image. The accuracy of $resnet_c$ on noisy CIFAR10 is only 34.1%. This shows that the accuracy of conventional classification CNNs, that achieve excellent results when applied to clean images, dramatically degrade due to the presence of noise.

The straightforward approach is to train Resnet using a noisy dataset, denote this network by $resnet_n$. The accuracy of this network on noisy CIFAR10 increases significantly to 77.5%. However, we aim to filter out the noise using the proposed IDCNN. Given a noisy image x_i, our preprocessing will produce a noise-free heat map for classification purposes such that $h = IDCNN(x_i)$. Then we classify with Resnet using this heatmap such that $y_i = resnet(IDCNN(x_i))$. Note that this replica of Resnet is trained using the noisy CIFAR10, and we train end-to-end, such that the heat-maps for classification might differ from those for edge-detection or regular image denoising. We found that the end-to-end approach was the best way to train this architecture, in comparison to training firstly the IDCNN, and only then the classification network on top of it. Figure 4 shows the outline of the proposed classification process.

TABLE I
The multi-scale architecture of U-net. The network has a 'U' shape, it downscales the image by a series of max-pooling layers in the first part, and up samples by interpolation layers in the second part. There are concatenation connections between the first part to the second for every two layers in the same image dimensions. Due to its architecture, the network applies multi-scale filters that maximize the signals and average the noise. Therefore it allows detection of edges and objects at low SNRs.

#	Type	Out Dim	Kernel	Stride	Pad
1	convolution	64	3×3	1	1
2	ReLU	64	-	1	0
3	convolution	64	3×3	1	1
4	ReLU	64	-	1	0
5	max-pooling	64	2×2	2	0
6	convolution	128	3×3	1	1
7	ReLU	128	-	1	0
8	convolution	128	3×3	1	1
9	ReLU	128	-	1	0
10	max-pooling	128	2×2	2	0
11	convolution	256	3×3	1	1
12	ReLU	256	-	1	0
13	convolution	256	3×3	1	1
14	ReLU	256	-	1	0
15	max-pooling	256	2×2	2	0
16	convolution	512	3×3	1	1
17	ReLU	512	-	1	0
18	convolution	512	3×3	1	1
19	ReLU	512	-	1	0
20	UpSample	512	-	0.5	0
21	cat(20,14)	768	-	-	0
22	convolution	256	3×3	1	1
23	ReLU	256	-	1	0
24	convolution	256	3×3	1	1
25	ReLU	256	-	1	0
26	UpSample	256	-	0.5	0
27	cat(26,9)	384	-	-	0
28	convolution	128	3×3	1	1
29	ReLU	128	-	1	0
30	convolution	128	3×3	1	1
31	ReLU	128	-	1	0
32	UpSample	128	-	0.5	0
33	cat(32,4)	192	-	-	0
34	convolution	64	3×3	1	1
35	ReLU	64	-	1	0
36	convolution	64	3×3	1	1
37	ReLU	64	-	1	0
38	convolution	1	1×1	1	0
39	Sigmoid	1	-	1	0

V. Image Denoising using an Edges-Guided Auxiliary Loss

In this section, we continue to exemplify the use of edge detection in image denoising. We introduce a novel approach to train image denoising with an auxiliary loss of edge preservation. To that end, we train an Image Denoising CNN (IDCNN) for images denoising. We use the images in the

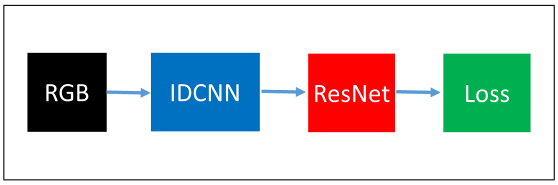

Fig. 4. Visualization of our classification process. The input is an RGB image, typically with a high level of noise. Then, it is processed by the multi-scale filters of IDCNN to produce meaningful heat-maps. Resnet 20 classifier process the heat-map and outputs the classification label. Finally, the whole scheme is trained using classification cross-entropy loss.

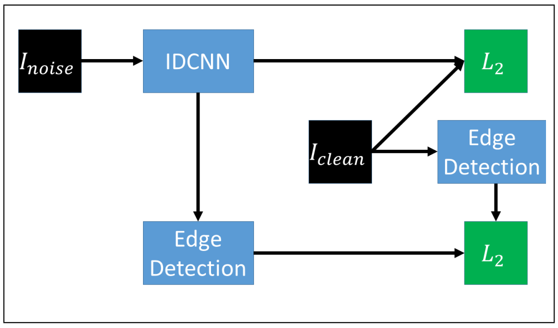

Fig. 5. Architecture of our training of IDCNN and IDCNN-E for natural image denoising. The input noisy image I_{noise} is the input for the U-Net denoising. Then, the output is connected to L_2 loss compared to the ground truth I_{clean}. In addition, we add an edge preserving auxiliary loss. By another L_2 loss we compare the Sobel edge detection [10] of I_{clean} to the edges map of the denoised image: $Edges(IDCNN(I_{noise}))$. This scheme improves the performance of our denoising over the regular L_2 loss training in both aspects of PSNR and SSIM.

Berkeley-Segmentation-Dataset (BSDS) [20], where each image is first converted to gray-scale, to create the pairs $\{I_c, I_n\}$ of clean and corresponding noisy images. The noisy images were computed by adding white Gaussian noise to each pixel using the noise $n \sim N(0, \sigma^2 = 15^2, 25^2, 50^2)$.

The input for training IDCNN is the noisy image I_n and the output is trained to be as closest to I_c using a L_2 regression loss. The IDCNN is trained for 200 epochs. To improve the quality of the results we train the same architecture called IDCNN-E by and edge preservation auxiliary loss, and the resulting architecture is depicted in Figure 5. The overall loss is the sum of the L_2 loss and the edge loss. The label for the loss is the Sobel edge detector [10] applied to the clean image. The labels are compared to the result of the same edge maps of IDDCNN

$$L_E = ||\frac{\partial}{\partial x} I_c - \frac{\partial}{\partial x} IDCNN(I_n)||_2^2. \tag{3}$$

In conclusion, IDCNN-E is trained by $L_2 + L_E$ while IDCNN is trained only by L_2 loss.

Our experiments show that our auxiliary loss improves the performance of the denoising PSNR and SSIM. In comparison to the excellent deep approach of DnCNN [34], we achieve a state-of-the-art level of image denoising. We show that the proposed IDCNN-Edges (IDCNN-E) is able to denoise the image while preserving the image details and it is the best approach for maximizing the SSIM score of the denoising result.

VI. Experimental Results

We experimentally verified the proposed scheme using simulated and real images. In edge detection we compare to FastEdges [22], classic Canny edge detector [5] and to the CNN-based Holistically-Edge-Detection (HED) [32]. We test these methods in both aspects of quality and run time. To extract a measure of similarity between the binary results and the ground truth we use a strict version of F-measure, that applies pixel-wise accuracy and does not allow the matching of neighboring pixels. The F-measure is the harmonic mean of the precision and recall:

$$F = \frac{2 \cdot precision \cdot recall}{precision + recall}. \tag{4}$$

Denote by Y' the labels and by Y the results, the precision is

$$precision = \frac{TruePositive}{TruePositive + FalsePositive} = \frac{\sum Y \cdot Y'}{\sum Y}, \tag{5}$$

while the recall is given by

$$recall = \frac{TruePositive}{TruePositive + FalseNegative} = \frac{\sum Y \cdot Y'}{\sum Y'}. \tag{6}$$

We compared the algorithms using a challenging binary pattern shown in Figure 7. This pattern contains a triangle, straight lines, 'S' shape, and concentric circles. We used it to create a set of noisy images with $SNR = [1, 1.2, ..., 2]$ by Gaussian additive noise. For each SNR, we compute the F-measure for every method for 100 iterations of different random noises as in Eq. 2. Then we take the average F-score of all the iterations. It can be seen in Figure 6 that the graph of the methods geared to detect faint edges in noisy images, ours and [22], are superior to Canny [5]. Moreover, the proposed FED-CNN outperforms the FastEdges approach.

The results are summarized in Table II, and it follows that for $SNR = \{1, 2\}$, the proposed FED-CNN approach achieves the highest F-score. Figure 7 shows the simulation images of $SNR = 2$. Compared to FastEdges [22], both approaches achieve a good accuracy of detection on this image, but the FED-CNN yields less false detections.

Even though the FED-CNN is the most accurate, it achieves a real-time running time as shown in Table III. Its runtime is 10 millisecond, similar to Canny's run time, and it is orders of magnitude faster that FastEdges [22] that runs in seconds. We computed these run times on a single machine with i7 CPU, 32 GB of RAM, and geforce gtx 1070 GPU. Note that the Canny and FastEdges implementations run on the CPU while our new method utilized the parallelism of the GPU.

Figure 8 shows the FED-CNN results when applied to noisy images from the binary images dataset [16] used for training and testing. It follows that we manage to detect and track high curvatures edges, being very similar to the ground truth labels. Figure 9 show the FED-CNN and FastEdges [22] results on a group of real images. Both methods obtains high quality of

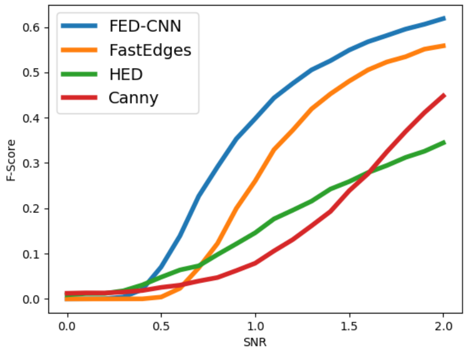

Fig. 6. Simulation of faint edges detection in noisy image. The strict F-score graph along the different signal-to-noise ratios from 0 to 2. The methods geared for faint edges detection achieve the highest accuracy. Our method FED-CNN obtains a slightly higher graph from the previous approach of FastEdges [22].

TABLE II
F-SCORE OF THE METHODS AT SNR 1 AND 2. AT BOTH SNR'S OUR METHOD ACHIEVES THE HIGHEST SCORE.

Algorithm	SNR = 1	SNR = 2
FED-CNN	**0.4**	**0.62**
FastEdges	0.28	0.56
HED	0.14	0.34
Canny	0.08	0.45

detection. However, since our network is fully convolutional, its run-time does not scale significantly with size, and our run time on these images is much faster than FastEdges.

A. Noisy Image Classification

We also examined the applicability of the proposed architecture of IDCNN (Image-Denoising) to image classification, following Section IV. For that we studied the following, we can classify images using the basic $resnet_c$, we can train on the noisy dataset and use $resnet_{noisy}$, or add preprocessing by IDCNN and to use $resnet(IDCNN)$.

We evaluated the performance of these three variants empirically. Table IV shows the overall accuracy of classification to 10 or 100 classes. Our approach is the only one that crosses the 80% gap and achieves an accuracy of 82.7% on noisy CIFAR10. In addition, Figure 10 shows the accuracy of each approach at every noise level. The basic network of $resnet_c$ gains the highest score of clean images, while our approach

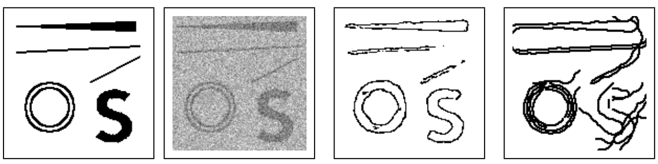

Fig. 7. Results on the simulation noisy image. From left to right: the clean binary pattern, the original images at SNR=2, our network result, FastEdges [22] result. Both methods produce good quality results while ours contains less false detections.

TABLE III
RUN TIME IN MILLI-SECONDS OF THE DIFFERENT METHODS OF EDGE DETECTION. OUR RUNTIME IS VERY CLOSE TO CANNY'S TIME AND IS ORDER OF MAGNITUDE FASTER THAN FASTEDGES. WE ACHIEVE THIS IMPROVEMENT MAINLY BY RUNNING OUR NETWORK ON A GPU, WHICH IS A HARDWARE OPTIMIZATION. THE ADVANTAGE OF CNN APPROACHES OVER CLASSIC METHODS LIKE FASTEDGES [22] IS THAT THEY ARE EASILY IMPLEMENTED AND ACCELERATED ON A GPU. IN ADDITION, DUE TO THE SIMPLICITY OF FED-CNN, ITS RUNTIME ON A CPU IS ALSO FASTER THAN FASTEDGES [22].

Algorithm	Run-Time (milliseconds)
FED-CNN	**10**
FED-CNN-CPU	**800**
FastEdges	2600
Canny	3

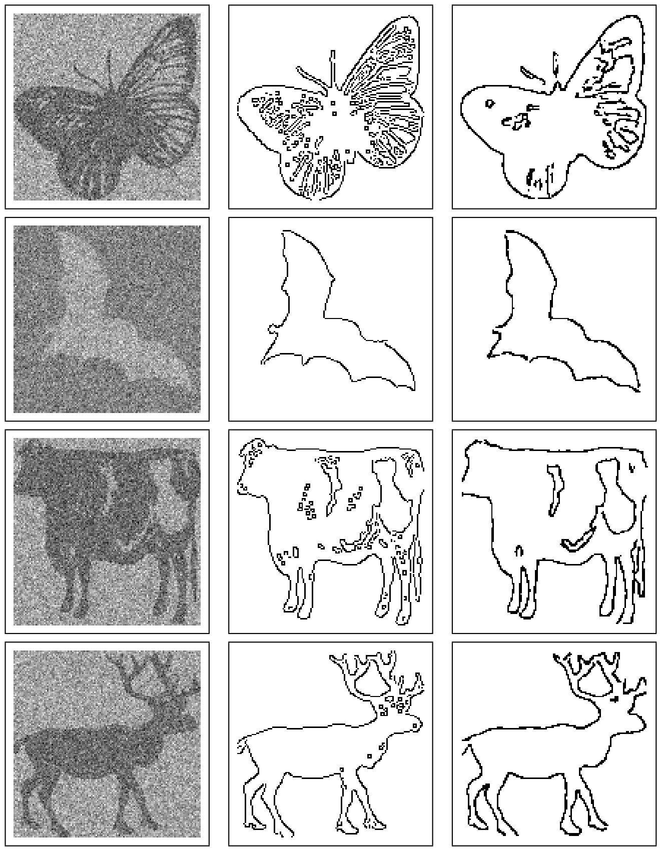

Fig. 8. Result on image from the binary images dataset [16] that we used to train and test our network. Left: the input noisy images with a binary pattern. Middle: the ground truth labels. Right: our detections. FED-CNN result is very similar to the ground truth and we manage to detect and track edges even at high curvatures.

gets the highest scores for all the other noise levels. We also emphasize our heat-map contribution visually. Figure 11 show an example of an image from the noisy CIFAR10 test set. For each image, we show the corresponding heat map produced by IDCNN. As can be seen, we succeed to produce meaningful heat-maps for every noise level such that the structure of the object is preserved and the noise is removed. As FED-CNN maximized the SNR for edges, the architecture of IDCNN does the same for maximizing objects' visibility and SNR due to

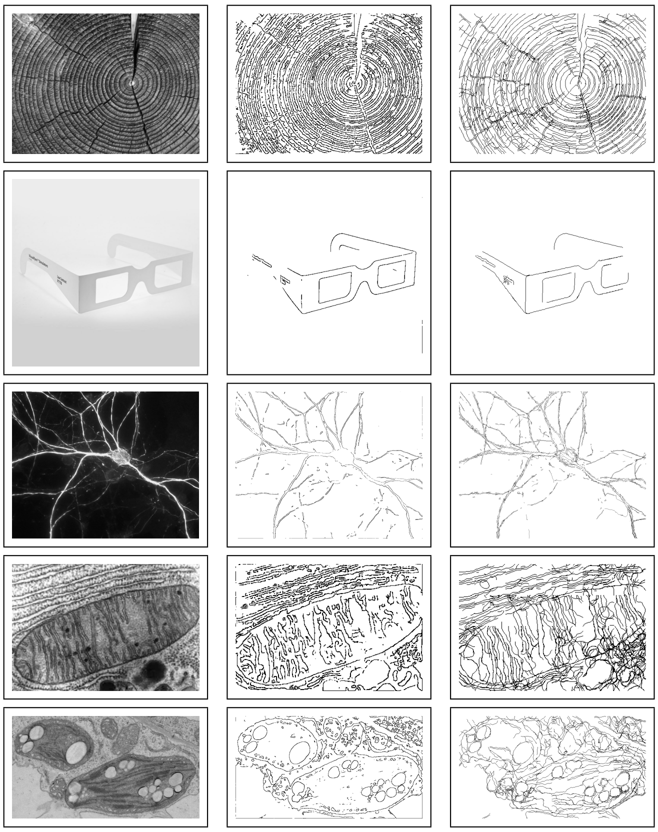

Fig. 9. Examples of real images. Left: the original gray scale images. Middle: our results. Right: FastEdges [22] results. Both methods achieve high quality of detections.

its multi-scale set of filters.

TABLE IV
CLASSIFICATION ACCURACY OF THE DIFFERENT METHODS AND ARCHITECTURES AVERAGED ON ALL NOISY LEVELS. OUR APPROACH OF MULTI-SCALE PREPROCESSING BY IDCNN [29] AND CLASSIFICATION BY RESNET 20 [28] ACHIEVES THE HIGHEST ACCURACY.

Algorithm	**CIFAR10**	**CIFAR100**
$resnet(IDCNN)$	**82.7**	**53.3**
$resnet_{noisy}$	77.5	46.0
$resnet_c$	34.1	16.9

B. Image Denoising

We evaluated the proposed IDCNN trained by L_2 regression loss and IDCNN-Edges (IDCNN-E) approach trained by an edge preservation auxiliary loss for image denoising using the Berkeley-Segmentation-Dataset (BSDS) [20]. Zero-mean Gaussian noise with $\sigma = 15, 25, 50$ was added to each image. We used the 200 train images to train the IDCNN, and the 200 test images to evaluate our scheme. We compared our approach to the classical methods of BM3D [7] and to the state-of-the-art deep-approach DnCNN [34]. For quantitative comparison we measured the Peak-Signal-to-Noise-Ratio (PSNR) and Structure-of-Similarity (SSIM) [31]. As shown in Table V, the IDCNN-E achieves the highest perceptual SSIM score, and we gain competitive PSNR. In particular, following [2], there is a known perception-distortion trade-off, and it is difficult to achieve both.

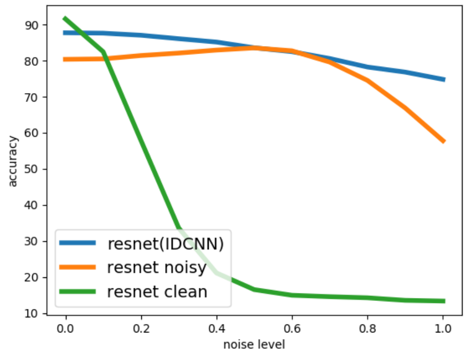

Fig. 10. Accuracy of classifiers along different noise levels. Our approach, training of IDCNN [23] and resnet20 [28] , $resnet(IDCNN)$, on a noisy dataset archives the highest accuracy at all levels greater than 0. Training of Resnet 20 on a noisy dataset, without the multi-scale preprocessing of IDCNN, archives lower scores. The regular Resnet 20 classifier, trained on a clean dataset, achieves the highest score on noise free images, but poor accuracy on all other noise levels.

Figure 12 depicts the denoising results using the BSDS dataset [20]. It follows that we achieve similar denoising as the DnCNN [34], while better preserving the details of the texture of the input images. Table VI shows the results of different approaches on the BSD68 dataset [24], where it follows that we achieve state-of-the-art perceptual SSIM score results.

TABLE V
QUANTITATIVE PSNR(DB) AND SSIM RESULTS OF DENOISING ON THE NOISY NATURAL IMAGES FROM THE TESTSET OF BSDS500 [20]. OUR APPROACH, USING IDCNN FOR DENOISING ACHIEVES THE HIGH PERCEPTUAL SCORE OF SSIM [31]. IN ADDITION, OUR DISTORTION SCORE OF PSNR IS ALSO COMPETITIVE RELATIVE TO THE STATE-OF THE ART APPROACH OF DNCNN [34]. OUR EDGE PRESERVING AUXILIARY LOSS IMPROVE THE PERFORMANCE OF IDCNN IN DENOISING IN BOTH MEASUREMENTS. THE HIGHEST SSIM SCORES ARE HIGHLIGHTED.

Algorithm	$\sigma = 15$	$\sigma = 25$	$\sigma = 50$
IDCNN-E	31.00/**0.9**	28.86/**0.85**	25.95/**0.75**
IDCNN	30.80/0.89	28.73/0.84	25.93/**0.75**
DnCNN	31.74/**0.9**	29.89/**0.85**	25.69/0.71
BM3D	31.07/0.88	28.26/0.81	24.57/0.67

TABLE VI
THE AVERAGE PSNR(DB) AND SSIM RESULTS OF DIFFERENT METHODS ON THE BSD68 DATASET [24]. THE HIGHEST SSIM SCORES ARE HIGHLIGHTED.

Algorithm	$\sigma = 15$	$\sigma = 25$	$\sigma = 50$
IDCNN-E	30.78/**0.9**	28.61/**0.84**	25.78/**0.74**
IDCNN	30.51/0.89	28.53/**0.84**	25.73/**0.74**
DnCNN	31.73/**0.9**	29.16/**0.84**	26.23/0.71
DeepAM [14]	31.68/0.89	29.21/0.82	26.24/0.72
TRD [6]	31.42/0.88	28.91/0.81	25.96/0.70
MLP [4]	-	28.91/0.81	26.00/0.71
CSF [26]	31.24/0.87	28.91/0.81	-
BM3D	31.12/0.87	28.91/0.81	25.65/0.69

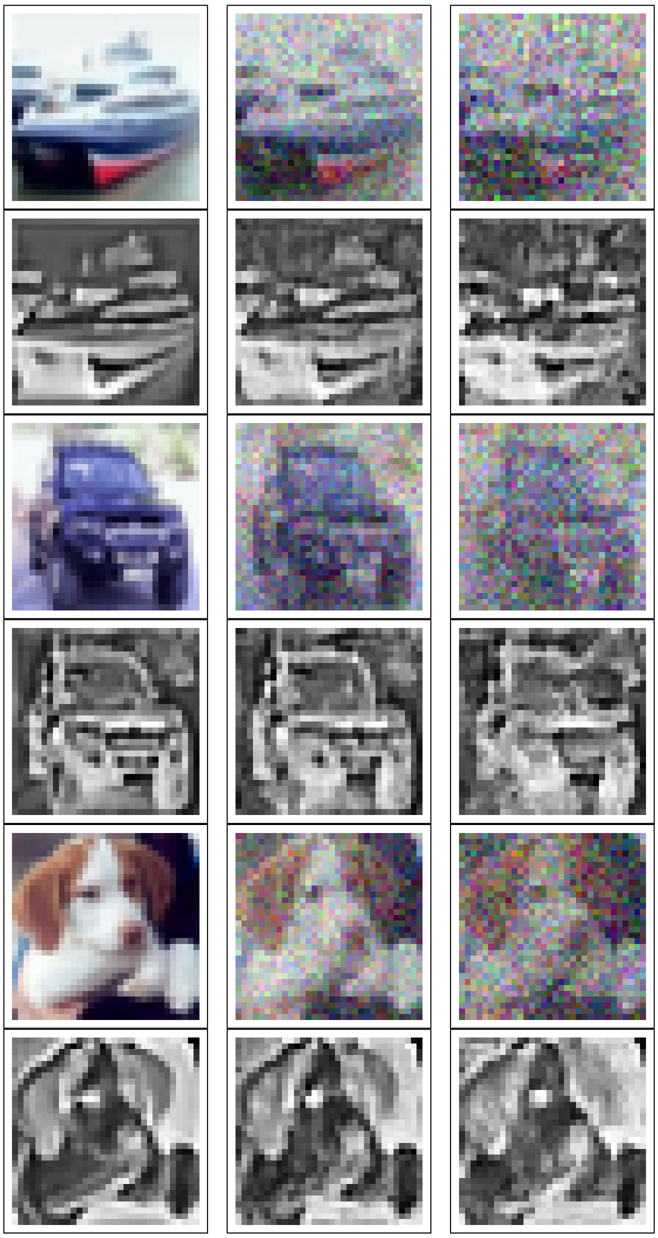

Fig. 11. Example of noisy object from the CIFAR 10 [15] dataset and their heat-maps produced by IDCNN trained for noisy image classification. Odd rows show the 32×32 images of the object in 3 noisy levels: clean, moderate and high. Even rows show the corresponding heat-map of the images produced my the multi-scale preprocessing of IDCNN architecture. These heat-maps allows classification of objects under high levels of noise.

VII. Conclusions

We introduced a novel work for multi-scale processing of noisy images using edge preservation losses. Our work is the first to solve the detection of faint edges in noisy images by using deep learning techniques. We compared our method to FastEdges [22] which is the state-of-the-art in faint edge detection. We showed experimentally that we succeed to improve this method in both aspects of run time and quality. We achieved similar and better results on simulation and real images while improving the run times in orders of magnitude. FastEdges needed seconds to process an image whereas our algorithm requires only milli-seconds by utilizing a fully convolutional network running on a GPU. Moreover, we showed that our approach to overcoming noise using deep multi-scale preprocessing of the image also improves the robustness of classifiers to noisy objects. The accuracy of noisy objects classification increases dramatically when applying the same preprocessing of our faint-edges detector. We emphasize the robustness of our CNN to noise also by the classical problem of image denoising. Our approach to training this network for denoising, using edge preservation auxiliary loss, achieves state-of-the-art scores of noise removal in natural images.

Fig. 12. Denoising results at $\sigma = 25$ on our noisy natural images simulated on the BSDS500 testset [20]. The input noisy image on the left, our results in the middle and DnCNN [34] on the right. As can be seen, both methods achieve similar quality of denoising while ours approach preserve more details of the image texture and edges.

References

[1] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 328–335, 2014.

[2] Y. Blau and T. Michaeli. The perception-distortion tradeoff. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6228–6237, 2018.

[3] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 60–65. IEEE, 2005.

[4] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2392–2399. IEEE, 2012.

[5] J. Canny. A computational approach to edge detection. In *Readings in Computer Vision*, pages 184–203. Elsevier, 1987.

[6] Y. Chen, W. Yu, and T. Pock. On learning optimized reaction diffusion processes for effective image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5261–5269, 2015.

[7] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising with block-matching and 3d filtering. In *Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning*, volume 6064, page 606414. International Society for Optics and Photonics, 2006.

[8] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1841–1848. IEEE, 2013.

[9] M. Galun, R. Basri, and A. Brandt. Multiscale edge detection and fiber enhancement using differences of oriented means. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[10] W. Gao, X. Zhang, L. Yang, and H. Liu. An improved sobel edge detection. In *Computer Science and Information Technology (ICCSIT),*

2010 3rd IEEE International Conference on, volume 5, pages 67–71. IEEE, 2010.
[11] F. E. Harrell. Ordinal logistic regression. In *Regression modeling strategies*, pages 311–325. Springer, 2015.
[12] I. Horev, B. Nadler, E. Arias-Castro, M. Galun, and R. Basri. Detection of long edges on a computational budget: A sublinear approach. *SIAM Journal on Imaging Sciences*, 8(1):458–483, 2015.
[13] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson. Crisp boundary detection using pointwise mutual information. In *European Conference on Computer Vision*, pages 799–814. Springer, 2014.
[14] Y. Kim, H. Jung, D. Min, and K. Sohn. Deeply aggregated alternating minimization for image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6419–6427, 2017.
[15] A. Krizhevsky, V. Nair, and G. Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.
[16] L. J. Latecki, R. Lakamper, and T. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 424–429. IEEE, 2000.
[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.
[18] K.-K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. Van Gool. Convolutional oriented boundaries: From image segmentation to high-level tasks. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):819–833, 2018.
[19] D. Marr and E. Hildreth. Theory of edge detection. *Proc. R. Soc. Lond. B*, 207(1167):187–217, 1980.
[20] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423. IEEE, 2001.
[21] M. K. Mihcak, I. Kozintsev, K. Ramchandran, and P. Moulin. Low-complexity image denoising based on statistical modeling of wavelet coefficients. *IEEE Signal Processing Letters*, 6(12):300–303, 1999.
[22] N. Ofir, M. Galun, B. Nadler, and R. Basri. Fast detection of curved edges at low snr. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 213–221, 2016.
[23] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.
[24] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 860–867. IEEE, 2005.
[25] U. Schmidt and S. Roth. Shrinkage fields for effective image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2774–2781, 2014.
[26] U. Schmidt and S. Roth. Shrinkage fields for effective image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2774–2781, 2014.
[27] J. A. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.
[28] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.
[29] R. Unnikrishnan, C. Pantofaru, and M. Hebert. A measure for objective evaluation of image segmentation algorithms. In *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, pages 34–34. IEEE, 2005.
[30] Y.-Q. Wang, A. Trouvé, Y. Amit, and B. Nadler. Detecting curved edges in noisy images in sublinear time. *Journal of Mathematical Imaging and Vision*, 59(3):373–393, 2017.
[31] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.
[32] S. Xie and Z. Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015.
[33] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang. Object contour detection with a fully convolutional encoder-decoder network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 193–202, 2016.
[34] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.