IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 14 May 2022, accepted 20 June 2022, date of publication 27 June 2022, date of current version 1 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3186344

METHODS

LDC: Lightweight Dense CNN for Edge Detection

XAVIER SORIA[1], GONZALO POMBOZA-JUNEZ[1], (Member, IEEE), AND ANGEL DOMINGO SAPPA[2,3], (Senior Member, IEEE)

[1]Faculty of Educational Science, Humanities, and Technology, National University of Chimborazo, Riobamba 060111, Ecuador
[2]CIDIS, ESPOL Polytechnic University, Guayaquil EC090112, Ecuador
[3]Computer Vision Center, Autonomous University of Barcelona, 08193 Barcelona, Spain

Corresponding author: Xavier Soria (xavier.soria@unach.edu.ec)

This work was supported in part by the ESPOL Project TICs4CI under Grant FIEC-16-2018, in part by the Physical Distancing under Grant CIDIS-56-2020, in part by the "CERCA Program/Generalitat de Catalunya," in part by the Spanish Government under Grant PID2021-128945NB-I00, and in part by the CYTED Network "Ibero-American Thematic Network on ICT Applications for Smart Cities" under Grant 518RT0559.

ABSTRACT This paper presents a Lightweight Dense Convolutional (LDC) neural network for edge detection. The proposed model is an adaptation of two state-of-the-art approaches, but it requires less than 4% of parameters in comparison with these approaches. The proposed architecture generates thin edge maps and reaches the highest score (i.e., ODS) when compared with lightweight models (models with less than 1 million parameters), and reaches a similar performance when compare with heavy architectures (models with about 35 million parameters). Both quantitative and qualitative results and comparisons with state-of-the-art models, using different edge detection datasets, are provided. The proposed LDC does not use pre-trained weights and requires straightforward hyper-parameter settings. The source code is released at https://github.com/xavysp/LDC.

INDEX TERMS Edge detection, deep learning, boundary detection.

I. INTRODUCTION

The edge detection task is still considered as the core work on several computer vision and image processing tasks, for instance it is used in medical image segmentation [5] and sketch image retrieval [6], just to mention a few. In the last decades, a large number of Deep Learning (DL) [7] based approaches have been proposed (e.g., [2], [8]–[10]) overcoming the state-of-the-art results from classical learning and non-learning based approaches. Due to the rapid development of deep learning methods, recent years have witnessed an explosive spread of Convolutional Neural Network (CNN) models to perform image edge detection. The obtained performance has been consistently improved by designing new architectures or introducing new loss functions. Though significant advances have been made, most of the works in edge detection were dedicated to achieve higher metrics (i.e., ODS, OIS, AP) with the design of a very deep network, which causes the increase in the numbers of computational operations. Regrettably, these edge detection approaches are not practical for low-capacity devices neither for real-world applications.

Having in mind the aforementioned drawbacks, in this paper a novel lightweight architecture named LDC—Lightweight Dense CNN model for edge detection—is presented. It is intended to be a practical network for real-world applications, which adaptively learns most valuable features focusing on high-frequency information. The proposed architecture is based on DexiNed [2], but in order to seek for a better trade-off between performance and applicability smaller filter size and compact modules are considered. As a result from the proposed modification, an model with less than 1M parameters is obtained, which is fifty time smaller than [2] as well as lighter than most of state-of-the-art approaches.

The proposed network gives cleaner edge-maps with less than 2% of parameters comparing to DexiNed [2]—see results from LDC in Fig. 1(*4th column*) and compare them with results from DexiNed [2] in Fig.1 (*3rd column*). On the other hand, LDC has 4% of parameters

The associate editor coordinating the review of this manuscript and approving it for publication was Ikramullah Lali.

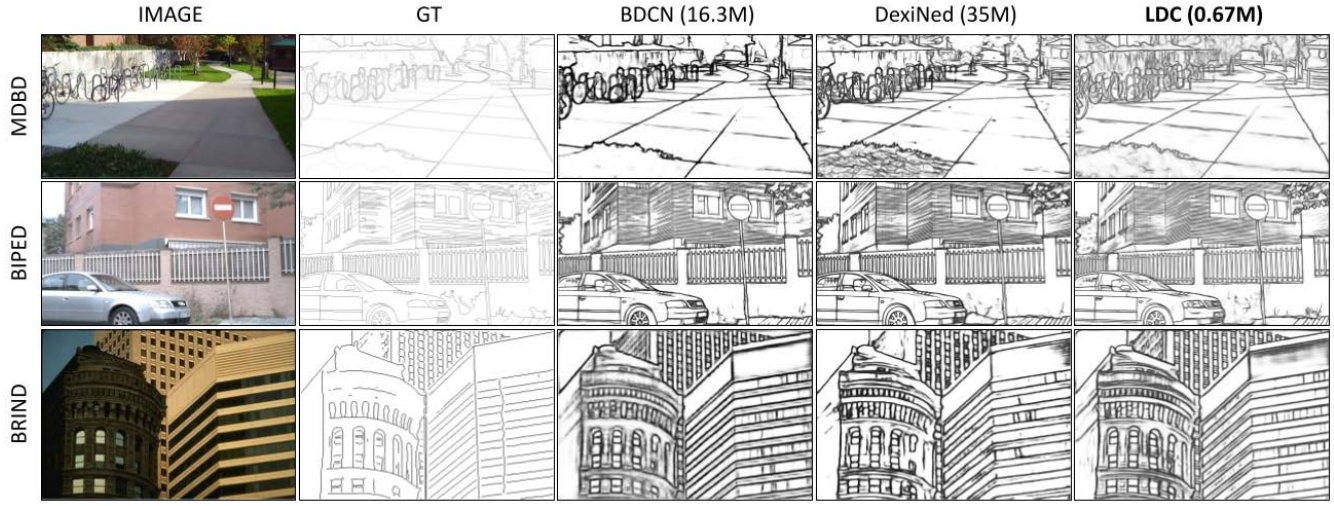

FIGURE 1. This illustration presents three sample images, one from each datasets annotated in edge level. MDBD, first row, is the Multicule dataset for Boundary detection—the edge part [1]. The second row, BIPED is the Barcelona Images for Perceptual Edge Detection, the last version, [2]. The last row, BRIND— is the BSDS dataset [3] re-annotated in edge level by [4]. Two best deep learning based models are selected to compare with the proposed LDC. Each prediction is trained in the same dataset used for evaluation. That is, the prediction from LDC in the first row is trained in MDBD and presents one sample from the same dataset but from the test part.

comparing to BDCN [8]; additionaly, the results from LDC and BDCN ((*2nd column*) indicate that LDC detects more edges than BDCN. Comparisons with state-of-the-art-approaches in the following datasets are also provided: Multicue Dataset for Boundary Detection—the edge annotation part—(MDBD) [1], Barcelona Images for Perceptual Edge Detection (BIPED) [2], [11], and the Berkeley Segmentation Dataset (BSDS) [3] re-annotated by [4] taking care on Reflectance, Illuminance, Normal, and Depth edges that is renamed as BRIND in this proposal. It should be highlighted that the proposed architecture, with such a small number of parameters, reaches in most of these edge detection datasets the best score.

Overall, the contribution in the manuscript can be summarized as follow:

- A lightweight CNN architecture is proposed from DexiNed [2], which has just 674K parameters compared to 35M in DexiNed.
- A lost function has been slightly modified from [9].
- An extensive comparison study is provided with the state-of-the-art edge detectors, those that have less than 1M parameters. This paper is entirely dedicated to edge detection, that is, all the models considered for quantitative comparison use BIPED, MDBD, and the new BRIND datasets, for training and validating the model. Additionally, each model is cross-validated using the other datasets.
- An in depth ablation study is conducted till reaching the proposed robust LDC. This contribution is one of the first manuscripts entirely dedicated for edge detection based on CNN that has no pre-training weights, high hyper-parameter settings, nor extensive time consumption in the training process.

The remainder of this paper is organized as follows. Section II presents the related work highlighting the number of parameters required for each approach. Then, Section III details the proposed architecture; experimental results are presented in Section IV. Finally, conclusions are provided in Section V.

II. RELATED WORK

This section briefly review algorithms and datasets for edge detection. Since the inception of Sobel edge detector [12], back in the 60s of the last century, a large number of approaches have been proposed for tackling this challenging task. For an in depth review, the following edge detection surveys can be considered [13]–[17]. Edge detectors can be broadly categorized into four groups: *i*) driven by low-level features; *ii*) brain-inspired; *iii*) classical learning-based; and *iv*) deep learning as stated in [2]. Bellow, a short description of DL based edge detectors is presented.

A. DEEP LEARNING BASED EDGE DETECTORS

The proliferation of CNN models started with the ImageNet challenge [18]. Since then, a numerous deep learning based models have been proposed for edge detection. One of the most used algorithm is HED [19] that is a deep supervised model based on the VGG16 architecture [20], the cross-entropy loss function used in HED facilitated to the community to design models with the end to end edge training procedure. Based on HED architecture several models (e.g., CED [21], RCF [10], BDCN [8]) have been proposed. Most of them use VGG16 [20] as backbone architecture, which is the architecture used by HED; there are a few of them that use ResNet [22]. The state-of-the-art

models mentioned above propose different intermediate edge fusion and slightly loss function modifications. Those models still use pre-training weights and layer level hyper-parameters settings. To mitigate these drawbacks, DexiNed, initially proposed in [11] and extended in [2], proposes to use a different CNN architecture partially based on [23] as well as a novel dataset for training. Although DexiNed has about 35M parameters, it can be trained from scratch with less hyper-parameter tuning; it reaches the best results on datasets intended for edge detection. Finally, a novel algorithm for edge detection, named CATS, is proposed in [9]. It is based on the usage of a new type of loss function, which uses cross-entropy additional to boundary tracing and texture suppression loss. As a result from this loss function, more cleaner and thinner edge-maps are obtained.

In addition to the approaches mentioned above, there are also contributions based on the usage of GAN models [24] termed as ContourGan [25], and more recently Transformers [26]. Although substantial improvements have been made by designing new architectures or introducing new loss functions [8], [9], [21], [27], most of the approaches require an increase on the number of computational operations. Hence they are not practical for low-capacity devices neither for real-world applications. This drawback has motivated the development of lightweight models (in general with less than 1M parameters) reaching appealing results. As examples of these lightweight models we can mention TIN [28] and PiDiNet [29] architectures; the main features of these models are the reduction on kernel size as well as the manual kernel setting. In the case of PiDiNet, it still uses CNN layer level of hyper-parameter tuning.

B. DATASETS FOR EDGE DETECTION

In recent years, different datasets have been proposed for training edge detection architectures. The first dataset is CID [30], which is not intended for edge detection but for the close related problem of contour detection; it consists of a set of 40 gray-scale images. Since this dataset is intended for contour detection there are some edges without annotations. Another widely used dataset is BSDS300, which has been presented in [31] and then extended to BSDS500 in [3]; although the annotations in BSDS are intended for boundary detection there are some images annotated at the edge level, this fact makes it difficult to train learning-based algorithms as stated in [2], [11], [32], [33]. Also related with the edge detection, we can mention the New York University Indoor dataset [34], which contains annotations for the semantic segmentation problem, conditioned for edge detection. Related with semantic segmentation, there are other datasets that have been also considered for tackling the edge detection problem (e.g., PASCAL-Context [35], CITYSCAPES [36], to name a few).

On the contrary to previous approaches, which were intended for contour and boundary detection, or to extract annotations for semantic segmentation, in 2016, Multicue Dataset for Boundary Detection (MDBD) has been proposed [1]. It consists of multiple annotations for boundary and edge level. More recently, [11] presents the Barcelona Images for Perceptual Edge Detection (BIPED) dataset; which is extended in [2]. Finally, [4] presents BRIND, which uses BSDS500 images for the edge level annotations. These three datasets (i.e., MDBD, BIPED and BRIND) are especially devoted for edge detection, hence they are going to be used for training LDC.

III. PROPOSED APPROACH

This section introduces the proposed Lightweight Dense Convolutional neural network for edge detection (LDC) architecture and gives details on the different modules. As mentioned in Section I, the proposed LDC model is based on DexiNed [2] and CATS [9], so below we are going to present in details the proposed modifications.

A. CONVOLUTIONAL NEURAL NETWORK ARCHITECTURE

The DexiNed architecture [2] consists of two subnets, Dexi and USNet. On the one hand, Dexi is composed by 6 blocks; each block, from the third block, is connected with two types of skip-connections. On the other hand, USNet is a conditional CNN used to upscale and convert feature maps in edge-maps with the same size as the input image of Dexi subnet. This architecture results in a model with 35M of parameters. Having in mind the lightweight focus of current work, the following modifications to DexiNed are proposed to generate the LDC architecture:

- Just four blocks from DexiNed are used in LDC.
- Instead of using the same filter size as DexiNed, LDC, with the purpose of light-weighting the number of parameters, drastically reduce Dexi filters' size (see labels inside blue rectangles in Fig. 2).
- As shown in Fig. 2, LDC has four intermediate edge-map predictions, hence, the final result comes from the fusion of these predictions. The strategy used for fusing these edge-maps is inspired in CATS [9]; it is referred to as Context-aware Fusion block or just CoFusion. This set of operations are also slightly modified to reduce the number of parameters without compromising the robustness. The modifications proposed in LDC are as follows. Instead of using 3 convolutional layers and 2 group of normalization with a kernel size of 64, LDC reduce the CoFusion to 2 convolutional layers and 1 group of normalization with the kernel size of 32. The other configurations are the same as in CATS [9].
- For the intermediate edge-maps formation in LDC, USNet is used with the same configuration as in DexiNed.
- Finally, to train LDC, the loss function from CATS [9] is also slightly modified as detailed in Section III-B.

With the modifications mentioned above, LDC results in a model with just 674K parameters. That is less than 2% of parameters compared to about 35M of parameters from DexiNed. For the efficient and robust training of the proposed

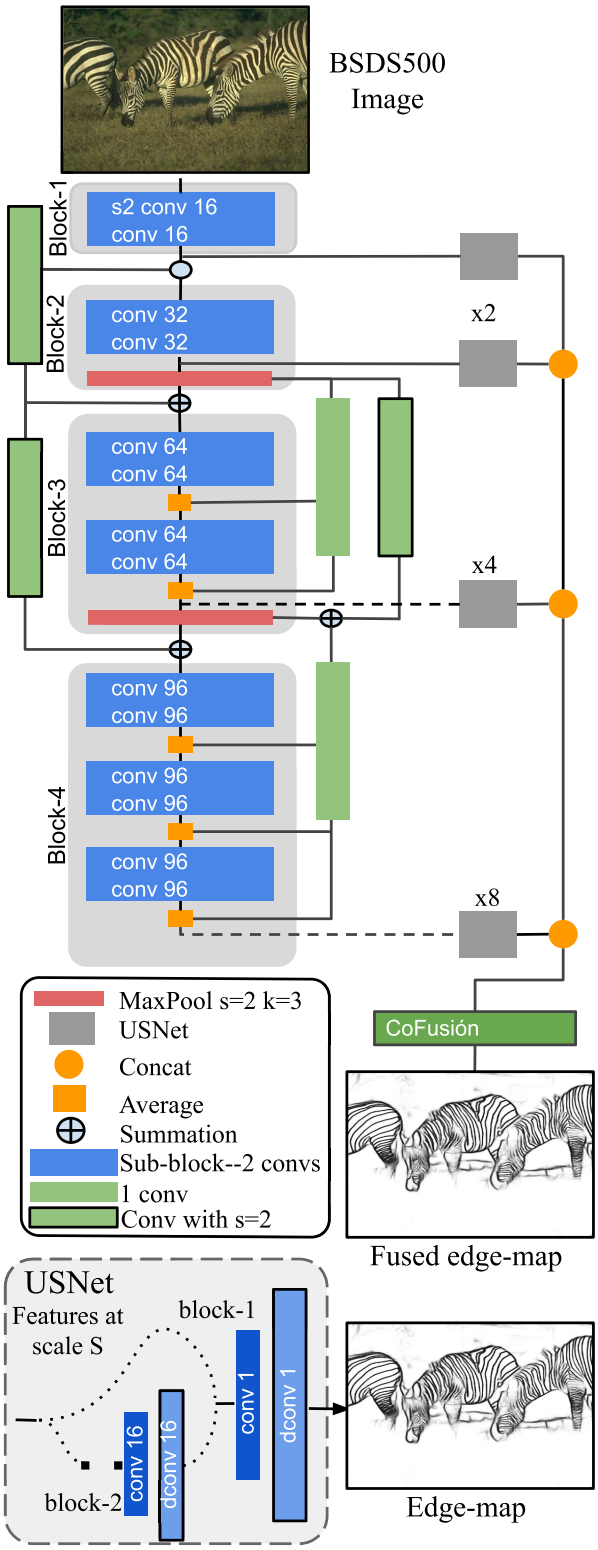

FIGURE 2. Architecture of the proposed LDC model.

model, different hyper-parameters have been set, which are going to be presented in Section IV. The LDC training and testing process is a straightforward procedure.

B. LOSS FUNCTION

The loss function from CATS [9] has been slightly modified to train LDC, which will be termed as CASTloss2. Overall, LDC return a set of edge-map predictions, $\hat{Y} \in R^{m\times n\times p}$, from a given RGB image $X \in R^{m\times n\times 3}$, validated with the corresponding ground truth, Y. The $\hat{Y}_p$ represents the last output from the proposed model (i.e., the fifth output), which corresponds to the prediction from the CoFusion stage, see Fig. 2 for more details. The $\hat{Y}_p$ output is considered for qualitative and quantitative comparison in the current work. The CATSloss2 is applied to every $\hat{Y}_i$. CATSloss2 is composed by three losses, tracing (cross-entropy) loss l_t, boundary tracing loss l_{bt}, and texture suppression loss l_{txs}. Hence the resulting CATSloss2 (l) is computed as follows:

$$l = l_t + \alpha_{bt} \times l_{bt} + \alpha_{txs} \times l_{txs} \tag{1}$$

where α_{bt} is a weight to regularize boundary tracing loss and α_{txs} is for texture suppression loss for each LDC prediction. The final loss is the sum of the l losses computed from each $\hat{Y}_i$—there are five predictions. Regarding the l_t loss, it is defined as follow:

$$\begin{aligned} l_t(\hat{Y}_i, Y) &= \frac{1}{m \times n} - w(Y \cdot \log \hat{Y}_i + (1 - Y) \cdot \log(1 - \hat{Y}_i)); \\ w_{(Y=1)} &= 1 * \frac{Y_-}{Y_+ + Y_-}, w_{(Y=0)} = 1.1 * \frac{Y_+}{Y_+ + Y_-}, w_{(Y=2)} = 0 \end{aligned} \tag{2}$$

where w is the weight of tracing loss, Y_- and Y_+ denote negative and positive edge samples in the given Y, respectively. Concerning to boundary tracing loss (l_{bt}), it is defined as follow:

$$l_{bt}(\hat{Y}_i, Y) = \frac{1}{m \cdot n} - \sum_{p \in E} \log \left(\sum_{j \in D_p} \frac{\hat{Y}_j}{\sum_{j \in R_p^e \setminus D_p} \hat{Y}_j + \sum_{j \in D_p} \hat{Y}_j} \right) \tag{3}$$

where E, as stated in [9], are the edge points of a given Y. R_p^e denotes a edge-map patch from $\hat{Y}_j$ that contains edge fragments, the center of the R_p^e is p. The edge points in R_p^e are represented in D_p. Finally, the texture loss (l_{txs}) is defined as follow:

$$l_{txs}(\hat{Y}_i, Y) = \frac{1}{m \cdot n} - \sum_{p \in Y \setminus \hat{E}} \log \left(1 - \sum_{j \in R_p^t} \frac{\hat{Y}_i^j}{|R_p^t|}\right), \tag{4}$$

where R_p^t is the edge-map patch that centers at a non-edge point p, and $\hat{E}$ is a set that includes all edges and their confusion pixels that have been used in l_{bt}. For a more detailed description see [9].

IV. EXPERIMENTS

A. DATASETS USED FOR EVALUATION

Three datasets have been used for training the proposed LDC architecture: MDBD [1], BIPED [2], and BRIND [4]

and compute quantitative evaluations—note these datasets are intended for edge detection. Furthermore, in order to evaluate the generalization of LDC trained with a given image set, the aforementioned three datasets have been used for cross-validation. Finally, qualitative evaluations have been performed by including CID [30], NYUD [34], and CITYSCAPES [36] datasets. Bellow, more details about each one of these datasets are provided.

1) MDBD

The Multicue Dataset for Boundary Detection [1] is a set of 100 images in high definition. These images have multiple annotations for boundary and edge level. In this manuscript, just the edge part of MDBD is considered. Generally, 80% of the images are selected for the training, and the remaining 20% are considered for testing. For a fair evaluation, LDC uses the same configuration as in DexiNed [2].

2) BIPED

The Barcelona Images for Perceptual Edge Detection (the last version) [2] has 250 images in high definition. 200 images are considered for training and the remaining 50 for testing. This dataset has just one annotation in the edge level that has been carefully validated. The same augmentation and processing operations used in DexiNed are applied in LDC.

3) BRIND

The Berkeley Reflectance, Illuminance, Normal and Depth edge dataset is presented in [4], it is a re-annotation of BSDS500 images [3] at the edge level. For the evaluation purpose, annotations from all type of edges are mixed before implementing the augmentation process. As in BSDS500, 300 images are considered for training and the remaining for testing. The augmentation procedure applied to BIPED is also implemented to BRIND.

4) CID

The Contour Image Database [30] contains 40 gray-scale images annotated in contour level. This dataset is used for qualitative LDC's judgement.

5) NYUD

The New York University Dataset [34] is used for the qualitative judgement. Therefore, just the test part of NYUD is used. This dataset is not considered for the models training due to the GT are generated for segmentation purposes and this bias the DL model training.

6) CITYSCAPES

The Cityscapes Dataset [36] is considered since those images have rich semantic segmentation annotations and challenging scenes. This dataset is also used only for visual judgement of the edge-map predictions.

B. IMPLEMENTATION DETAILS

The LDC is implemented in PyTorch [37], using a TITAN X 12 GB GPU. The final version of LDC is set with Adam optimizer, weight decay 0., initial learning rate $5e-5$; its updates are in 6, 12, 18 epochs with the following $25e-4$, $5e-4$, $1e-5$ learning rates, respectively. The stable training reach in the 17th epoch that take around 10 hours with a batch size of 8. In the majority of CNN layers Xavier normal initializer is applied, just in the last convolutional and deconvolutional layers in USNet are set with random normal distribution, mean 0. and standard deviation 0.1, respectively.

The metrics considered for the quantitative evaluation are Optimal Dataset Scale (ODS), Optimal Image Scale (OIS) and the Average Precision (AP). The GT for BIPED and BRIND is clipped in 0 and 1 after summing 0.6 to all values of GT greater than 0.2. For the MDBD GT, 0.2 is summed to all values greater than 0.1 and then clipped.

C. QUANTITATIVE COMPARISON

This section presents quantitative comparison of LDC with the state-of-the-art lightweight approaches. The selection of lightweight models for comparisons is based on the number of parameters, only models with less than 1 million parameters are considered: BDCN-B2 (i.e., 2 blocks of BDCN [8]) with 268K parameters, TIN [28] with 244K parameters, and PiDiNet [29] with 710K parameters. All those models are trained and evaluated in the following datasets: MDBD [1], BIPED [2], and BRIND [4]. The trained lightweight models have been also cross-validated, that is, PiDiNet trained on BIPED is evaluated in the test set of BIPED, MDBD, and BRIND—the same for the remaining lightweight models. For a fair comparison, all models are trained with the same data augmentation procedure as in LDC. In other words, MDBD used the same 80% of the images to train the whole models compared in Table 1, and the same images are considered to evaluate the effectiveness of the models in comparison. With respect to the edge-map predictions from all the models considered in the evaluation, a non-maximum suppression is applied before evaluating in the aforementioned metrics (see Section IV-B).

Table 1 presents ODS, OIS, and AP for the setup mentioned above (lightweight models and datasets). In addition to the lightweight models mentioned above, Table 1 also shows as a reference, the best results from the state-of-the-art for each dataset. These results corresponds to heavy models (16.3M and 35M parameters) and are depicted in the first row of each dataset section. The column **Trained** refers to the models trained on the dataset shown in that column, while **Tested** column refers to the dataset used to evaluate on the three metrics coming in the following columns. For example, the last row in Table 1 corresponds to the results of LDC trained on BIPED and evaluated in BRIND, MDBD, and BIPED, in that order. The last column shows the training epochs (*Ep*). The number of parameters (#*P*) for each model is provided in the second column.

As it can be appreciated from Table 1, the proposed model reaches the best result in ODS when it is trained and evaluated in the same dataset (e.g., trained and tested in BIPED, it score 0.889 ODS), that is just 0.6% below to the best result scored by DexiNed, which has about 35M parameters. In other words, with just less than 2% of parameters LDC reaches almost the same score as the state of the art. Considering BRIND, our approach get a better score than BDCN, which has about 16M parameters. Finally, with respect to the proposed cross-validation, to evaluate the model generalization, in most of the comparisons LDC gets better than its counterparts, losing just in three (i.e., LDC trained in BRIND and tested in BIPED; trained in BIPED and tested in BRIND; and trained in MDBD and tested in BIPED). It should be mentioned that in those cased where LDC did not reach the best result it remain as the second best score.

In order to evaluate the lightness of the proposed approach, and compare it with state-of-the-art lightweight architectures, the Frame Per Second (FPS) is computed on the BSDS500 [3] test images. The FPS evaluation procedure used in [28], [29] is considered. It consists in evaluating all the BSDS500 test images and computing an average of the amount of images processed in a second. Figure 3(*a*) presents comparisons between the different versions of LDC and the models in Table 1 in term of FPS. In order to highlight the lightness of the proposed approach, a laptop based on i5-10210U CPU processor is considered. As it can be appreciated the full version of the proposed LDC architecture (LDC-B6), due to its straightforward procedures, reaches the highest FPS score compared with the state-of-the-art approaches, eventhough some of the models, TIN and BDCN-B2, have less than half parameters of LDC-B6. This results is even better when the LDC-B4 version of the proposed approach is considered, reaching an average of more than 4 images per second—see Fig. 3(*a*). Finally, the lighter version, LDC-B2, can process more than 12 images each second on such a cheap hardware. Concerning to the training stage, as shown in the Fig. 3 (*b*), LDC can "stabilize" the training in less time than its counterparts, from the 12 epoch.

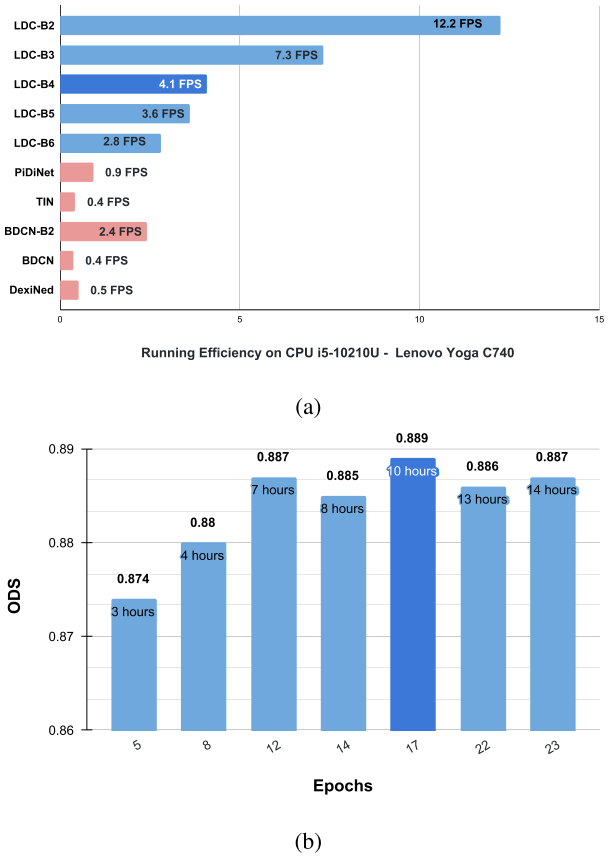

FIGURE 3. (*a*) **Efficiency of the models presented in Table 1 and Table 2 computed as Frame-Per-Second (FPS) in BSDS500 [3] test images—evaluation performed on a Lenovo Y740 laptop with an i5-10210U CPU processor.** (*b*) **Results on ODS from different epochs of LDC. The training procedure takes almost 14 hours till reaching 23 epochs on a TITAN X 12GB GPU.**

D. ABLATION STUDY

This section presents an ablation study on different configurations of LDC to show the robustness and straightforwardness, which with less than 2% parameters of DexiNed and 4% parameters of BDCN, reaches similar results. Different settings on loss functions, edge-map fusion strategies, and LDC blocks, are presented in Table 2. With this settings, LDC model does not need, manual kernel settings, layer level LR settings, nor pre-training weights. The proposal trains from scratch. This section presents the final setting of the LDC trained and evaluated in BIPED dataset. That is, the weight decay is set 0., initial learning rate is $5e-5$, the set of LR is $[25e-4, 5e-4, 1e-5]$, which is updated in [6, 12, 18] epochs. The predictions considered for quantitative evaluations are from the 17th epoch.

As mentioned above, LDC architecture is based on DexiNed [2], additionally, a slightly modified loss function and final prediction fusion, CoFusion, is considered from CATS [9]. The first section (five top rows) of Table 2 presents an empirical evaluation using different loss functions and edge-map fusion strategies. The best configuration for LDC comes from the fifth row, which corresponds to 4 blocks, CATSLoss2 loss function and the outputs fusion from CATS [9].

Once the best hyper-parameters are set, from the block 2 till the block 6 of LDC are evaluated. With the hyper-parameters setting stated as indicated above and as shown in the second section (five bottom rows) of Table 2, the best result is the prediction from the 4 blocks of LDC. It should be highlighted that LDC with 3 block has just 156K parameters, less than any models compared in Table 1, furthermore it reaches better results that BDCN-B2 or TIN. Finally, Fig. 3 (*b*) presents an illustration of results from LDC predictions on different epochs. The best results is the one obtained in epoch 17th with just 10 hours of training on a TITAN X 12GB GPU.

TABLE 1. Comparison of results from the proposed lightweight model (LDC) and state-of-the-art approaches. Three edge based datasets have been considered for the evaluation.

Methods	#P	Trained	Tested	ODS	OIS	AP	Tested	ODS	OIS	AP	Tested	ODS	OIS	AP	Ep
REF BDCN [8]	*16.3M*	BRIND [4]	BRIND [4]	*.789*	*.808*	*.795*	MDBD [1]	–	–	–	BIPED [2]	–	–	–	*20K*
BDCN-B2 [8]	268K			.733	.762	.708		.858	.870	.793		.826	.842	.855	20K
TIN [28]	244K			.717	.743	.658		.812	.842	.833		.756	.773	.726	20
PiDiNet [29]	710K			.789	.801	**.830**		.859	.867	.886		**.848**	**.855**	**.870**	20
LDC (Ours)	674K			**.790**	**.805**	.823		**.875**	**.885**	**.891**		.841	.846	.867	12
REF DexiNed-a [2]	35M	MDBD [1]	BRIND [4]	–	–	–	MDBD [1]	*.894*	*.902*	*.951*	BIPED [2]	–	–	–	*12*
BDCN-B2 [8]	268K			.722	.754	.692		.866	.877	.842		**.822**	**.836**	.867	20K
TIN [28]	244K			.686	.714	.685		.857	.866	.872		.742	.769	.745	20
PiDiNet [29]	710K			.670	.697	.688		.879	.882	.899		.791	.806	.801	20
LDC (Ours)	674K			**.741**	**.764**	**.787**		**.880**	**.891**	**.937**		.820	.830	**.888**	6
REF DexiNed [2]	35M	BIPED [2]	BRIND [4]	–	–	–	MDBD [1]	–	–	–	BIPED [2]	*.895*	*.900*	*.927*	*11*
BDCN-B2 [8]	268K			.667	.703	.438		.843	.857	.724		.849	.864	.902	20K
TIN [28]	244K			.588	.652	.612		.812	.842	.833		.806	.820	.845	20
PiDiNet [29]	710K			**.749**	**.771**	.768		.852	.859	.876		.887	**.895**	.924	20
LDC (Ours)	674K			.745	.763	.785		**.869**	**.882**	**.910**		**.889**	.894	**.932**	17

TABLE 2. Different settings evaluated till reaching the best LDC configuration. The LDC versions are trained and evaluated on BIPED dataset.

B2	B3	B4	B5	B6	Loss	Fusion	ODS	OIS	AP	#P
		✓			BDCNloss2	HEDfusion	.887	.893	.929	**672K**
		✓			BDCNloss2	Cofusion	**.889**	.893	.930	674K
		✓			CATSloss	Cofusion	.886	.893	**.935**	713K
		✓			CATSloss2	HEDfusion	.885	.891	.930	**672K**
		✓			CATSloss2	Cofusion	**.889**	**.894**	.932	674K
✓					CATSloss2	Cofusion	.843	.856	.912	**18K**
	✓				CATSloss2	Cofusion	.874	.882	.928	156K
		✓			CATSloss2	Cofusion	**.889**	**.894**	**.932**	674K
			✓		CATSloss2	Cofusion	.887	.893	.931	792K
				✓	CATSloss2	Cofusion	.882	.889	.928	888K

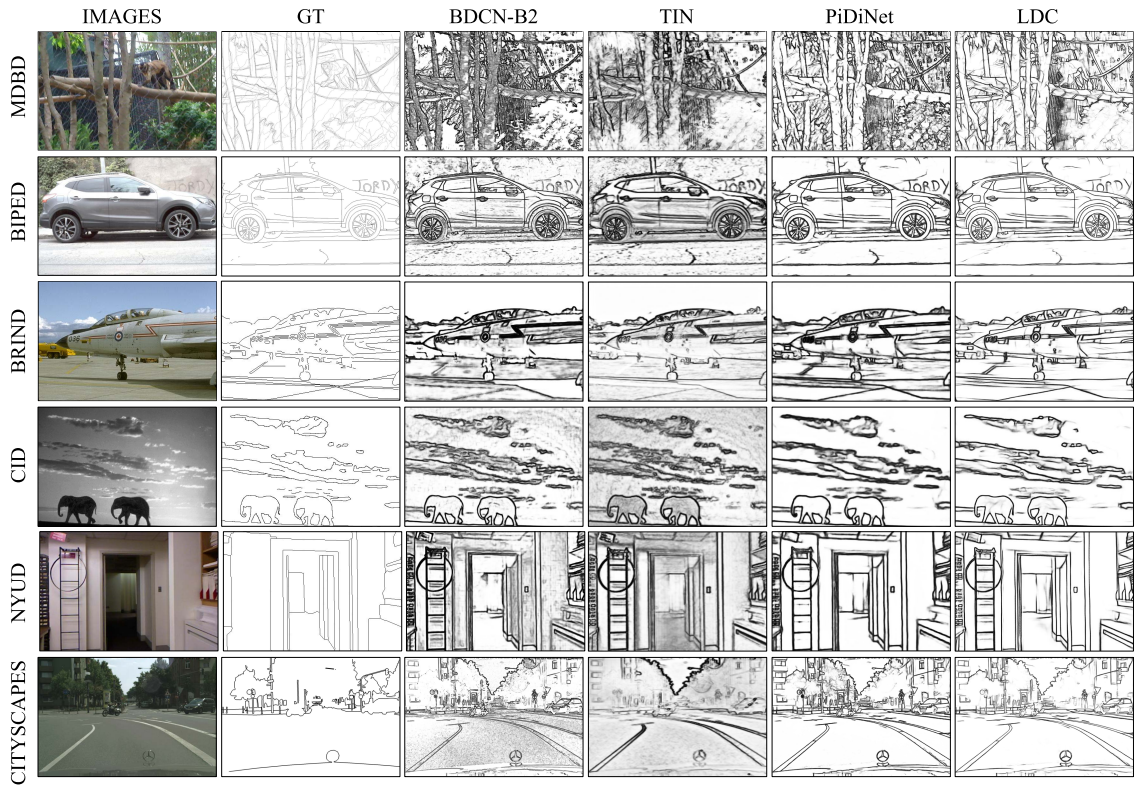

FIGURE 4. Edge-map predictions from the four lightweight models trained on BIPED.

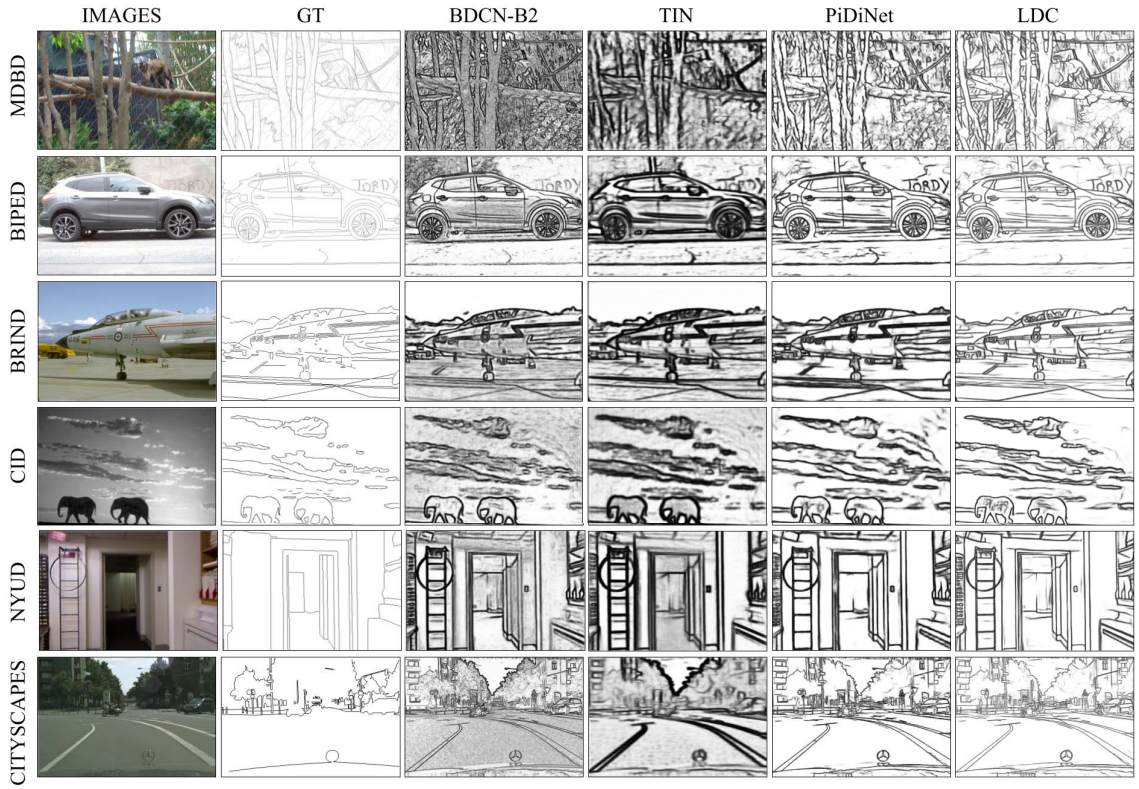

FIGURE 5. Edge-map predictions from the four lightweight models trained on BRIND.

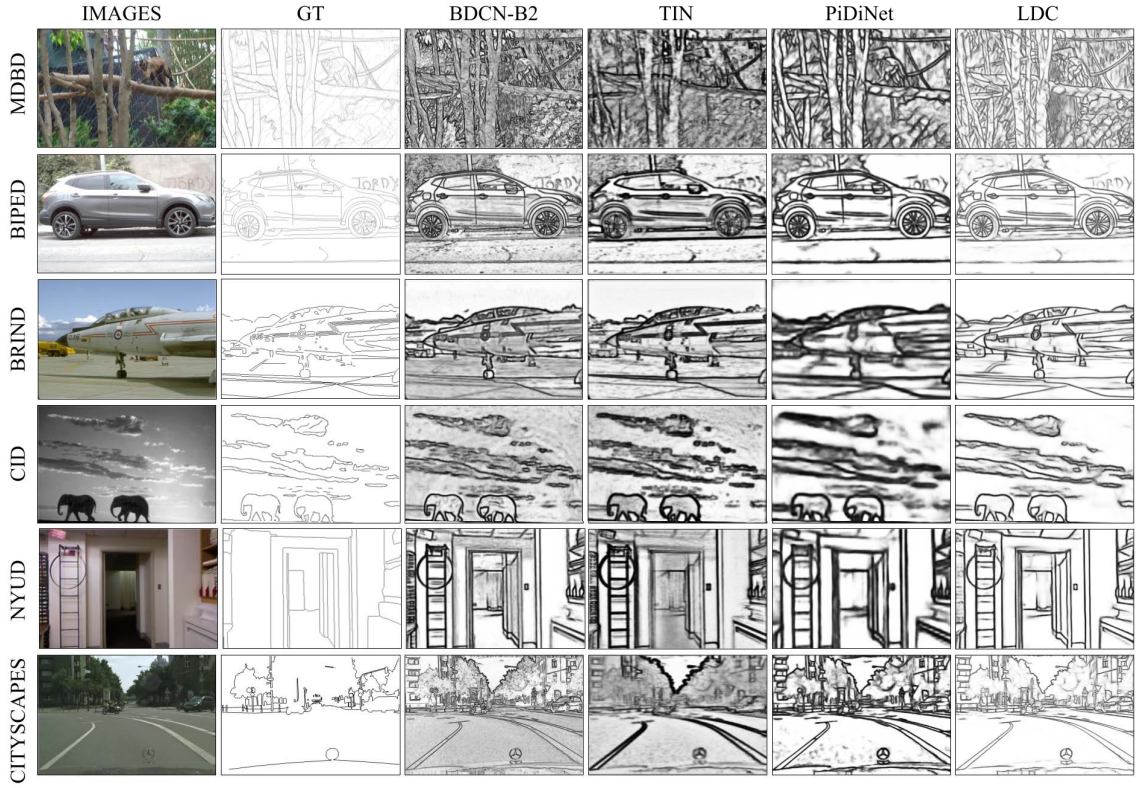

FIGURE 6. Edge-map predictions from the four lightweight models trained on MDBD.

E. QUALITATIVE RESULTS

This section presents qualitative results from all the datasets presented in Sec. IV-A. All the models are trained in BIPED/BRIND/MDBD and evaluated in the corresponding dataset. Figure 4 shows results when all models considered for evaluation are trained in BIPED; Figure 5 presents results on the same set of images when the models are trained on BRIND and finally, Fig. 6 shows results trained on MDBD.

Figures 4, 5 and 6 illustrate one sample for each dataset considered in this qualitative evaluation. It can be appreciated that predictions from LDC are cleaner and sharper, independently if the images are captured in indoor, outdoor, or gray-scale scenes; furthermore, it does not depend on the dataset used for training. In all the cases LDC is able to rightly predict edges. The second best result comes from PiDiNet. Focusing on CID dataset, the four models are able to predict edges from grayscale images, in spite of the fact these kind images are not considered in the training stage.

V. CONCLUSION

This manuscript presents a practical, straightforward, and also robust CNN model for edge detection. This model has less than 0.7 million of parameters and is able to reach the state-of-the-art results on edge based datasets. The availability of three datasets intended for edge detection opens the scope for the development of new learning-based algorithms in the edge detection domain.

ACKNOWLEDGMENT

The authors would like to thank NVIDIA for GPU donations.

REFERENCES

[1] D. A. Mély, J. Kim, M. McGill, Y. Guo, and T. Serre, "A systematic comparison between visual cues for boundary detection," *Vis. Res.*, vol. 120, pp. 93–107, Mar. 2016.

[2] X. S. Poma, A. D. Sappa, P. Humanante, and A. Akbarinia, "Dense Extreme Inception Network for Edge Detection," 2021, *arXiv:2112.02250*.

[3] 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.

[4] M. Pu, Y. Huang, Q. Guan, and H. Ling, "RINDNet: Edge detection for discontinuity in reflectance, illumination, normal and depth," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 6879–6888.

[5] X. Du, Y. Nie, F. Wang, T. Lei, S. Wang, and X. Zhang, "AL-Net: Asymmetric lightweight network for medical image segmentation," *Frontiers Signal Process.*, vol. 2, May 2022, doi: 10.3389/frsip.2022.842925.

[6] Q. Yu, F. Liu, Y.-Z. Song, T. Xiang, T. M. Hospedales, and C. C. Loy, "Sketch me that shoe," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 799–807.

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[8] J. He, S. Zhang, M. Yang, Y. Shan, and T. Huang, "BDCN: Bi-directional cascade network for perceptual edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 100–113, Jan. 2022.

[9] L. Huan, N. Xue, X. Zheng, W. He, J. Gong, and G.-S. Xia, "Unmixing convolutional features for crisp edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, May 27, 2021, doi: 10.1109/TPAMI.2021.3084197.

[10] Y. Liu, M. Cheng, X. Hu, J. Bian, L. Zhang, X. Bai, and J. Tang, "Richer convolutional features for edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 8, pp. 1939–1946, Aug. 2019.

[11] X. Soria, E. Riba, and A. Sappa, "Dense extreme inception network: Towards a robust CNN model for edge detection," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 1923–1932.

[12] I. Sobel, "Camera models and machine perception," Comput. Sci. Dept., Technion, Tech. Rep, 1970.

[13] D. Ziou and S. Tabbone, "Edge detection techniques—An overview," *Pattern Recognit. Image Anal. C/C Raspoznavaniye Obrazov I Analiz Izobrazhenii*, vol. 8, pp. 537–559, Dec. 1998.

[14] M. Basu, "Gaussian-based edge-detection methods—A survey," *IEEE Trans. Syst., Man Cybern., C, Appl. Rev.*, vol. 32, no. 3, pp. 252–260, Aug. 2002.

[15] R. Maini and H. Aggarwal, "Study and comparison of various image edge detection techniques," *Int. J. Image Process.*, vol. 3, no. 1, pp. 1–11, 2009.

[16] X.-Y. Gong, H. Su, D. Xu, Z.-T. Zhang, F. Shen, and H.-B. Yang, "An overview of contour detection approaches," *Int. J. Autom. Comput.*, vol. 15, no. 6, pp. 656–672, Jun. 2018.

[17] D. Yang, B. Peng, Z. Al-Huda, A. Malik, and D. Zhai, "An overview of edge and object contour detection," *Neurocomputing*, vol. 488, pp. 470–493, Jun. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092523122200248X

[18] O. Russakovsky, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[19] S. Xie and Z. Tu, "Holistically-nested edge detection," *Int. J. Comput. Vis.*, vol. 125, nos. 1–3, pp. 3–18, Dec. 2017.

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

[21] Y. Wang, X. Zhao, Y. Li, and K. Huang, "Deep crisp boundaries: From boundaries to higher-level tasks," *IEEE Trans. Image Process.*, vol. 28, no. 3, pp. 1285–1298, Mar. 2019.

[22] K. 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.

[23] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.

[24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1–47.

[25] H. Yang, Y. Li, X. Yan, and F. Cao, "ContourGAN: Image contour detection with generative adversarial network," *Knowl.-Based Syst.*, vol. 164, pp. 21–28, Jan. 2019.

[26] M. Pu, Y. Huang, Y. Liu, Q. Guan, and H. Ling, "EDTER: Edge detection with transformer," 2022, *arXiv:2203.08566*.

[27] R. Deng, C. Shen, S. Liu, H. Wang, and X. Liu, "Learning to predict crisp boundaries," in *Proc. Eur. Conf. Comput. Vis.*, Cham, Switzerland: Springer, 2018, pp. 562–578.

[28] J. K. Wibisono and H.-M. Hang, "Traditional method inspired deep neural network for edge detection," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2020, pp. 678–682.

[29] Z. Su, W. Liu, Z. Yu, D. Hu, Q. Liao, Q. Tian, M. Pietikainen, and L. Liu, "Pixel difference networks for efficient edge detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 5097–5107.

[30] C. Grigorescu, N. Petkov, and M. A. Westenberg, "Contour detection based on nonclassical receptive field inhibition," *IEEE Trans. Image Process.*, vol. 12, no. 7, pp. 729–739, Jul. 2003.

[31] 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 *Proc. 8th IEEE Int. Conf. Comput. Vis. (ICCV)*, Jul. 2001, pp. 416–423.

[32] X. Hou, A. Yuille, and C. Koch, "Boundary detection benchmarking: Beyond F-measures," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2123–2130.

[33] N. Wang, W. Wang, and W. Hu, "Thangka mural line drawing based on cross dense residual architecture and hard pixel balancing," *IEEE Access*, vol. 9, pp. 48841–48850, 2021.

[34] S. Gupta, P. Arbelaez, and J. Malik, "Perceptual organization and recognition of indoor scenes from RGB-D images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 564–571.

[35] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 891–898.
[36] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3213–3223.
[37] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, and A. Desmaison, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–12.

XAVIER SORIA received the Ph.D. degree in computer science from the Autonomous University of Barcelona, Spain, in 2019. He is currently an Adjunct Professor with the National University of Chimborazo. His research interests include machine learning and its applications on computer vision and image processing tasks.

GONZALO POMBOZA-JUNEZ (Member, IEEE) received the B.S. degree in electronic and computer engineering from the Polytechnic School of Chimborazo, Ecuador, in 2000, the M.S. degree in software development from the University of Granada, the M.S. degree in software engineering from the Autonomous Regional University of the Andes, and the Ph.D. degree in information and communication technologies from the University of Granada, Spain. Since 1996, he has been a Professor with the National University of Chimborazo, Ecuador. His research interests include advanced monitoring and control systems, natural user interface systems, ubiquitous home automation and ambient intelligence systems, and software platforms for embedded and mobile devices, works with machine learning algorithms.

ANGEL DOMINGO SAPPA (Senior Member, IEEE) received the degree in electromechanical engineering from the National University of La Pampa, General Pico, Argentina, in 1995, and the Ph.D. degree in industrial engineering from the Polytechnic University of Catalonia, Barcelona, Spain, in 1999. In 2003, after he holding research positions in France, U.K., and Greece, he joined the Computer Vision Center, Barcelona, where he is currently holds a senior scientist position. Since 2016, he has been a Full Professor with ESPOL Polytechnic University, Guayaquil, Ecuador, where he leads the CIDIS Research Center, Computer Vision Team. He is also the Director of the Electrical Engineering Ph.D. Program. His research interests include cross-spectral image processing and representation, 3D data acquisition, processing, modeling, and computer vision applications.