Traditional method inspired Deep Neural Network for Edge Detection
Jan Kristanto Wibisono and Hsueh-Ming Hang
28 May 2020

Vladislav Panferov

Novosibirsk State University

v.panferov@g.nsu.ru

June 4, 2020

Overview

Introduction

The aim of the work is to implement DNN based edge prediction with minimal complexity. Usually researchers use VGGNet as their feature-based extractor which was designed for image classification problem.
So, the authors of the paper have opinion that edge detection is much simpler job, and VGGNet is an overkill. The authors simplify the network architecture to include:

1. Feature extractor
2. Enrichment
3. Summarizer

1. Gradient
2. Low Pass Filter
3. Pixel Connection

The proposed method can effectively reduce the complexity and retain the edge prediction quality. The authors propose TIN1 and TIN2 (Traditional Inspired Network), which has an accuracy higher than the recent BDCN2 (Bi-Directional Cascade Network) but with much smaller model.

Feature extractor

We start from a simple feature extractor, which is a 3x3 convolutional neural network layer. We settle 16 feature channels

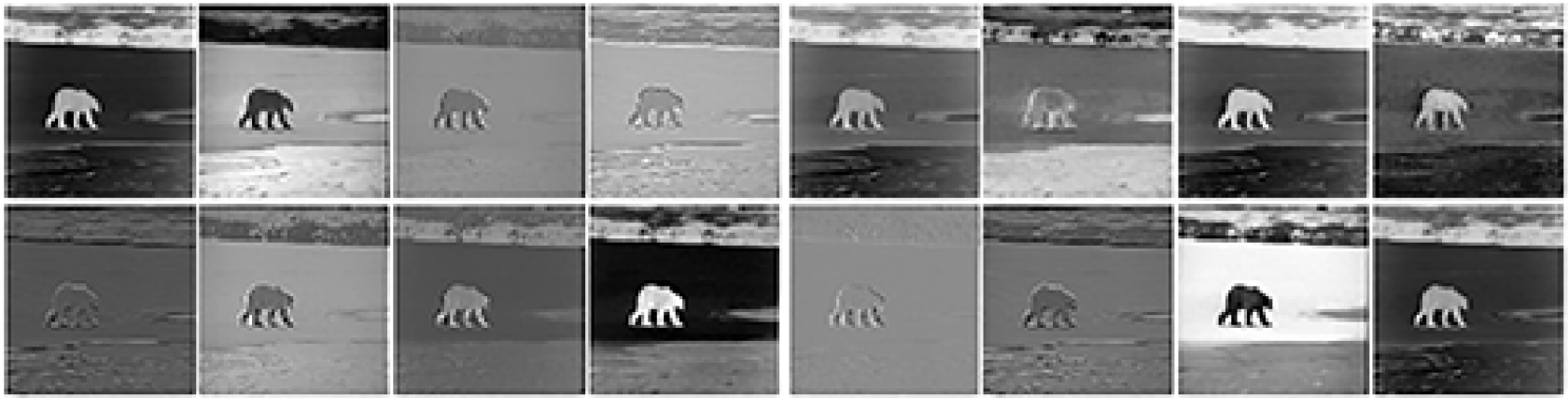

Figure: 16 feature maps from the 1st Feature Extractor

Enrichment

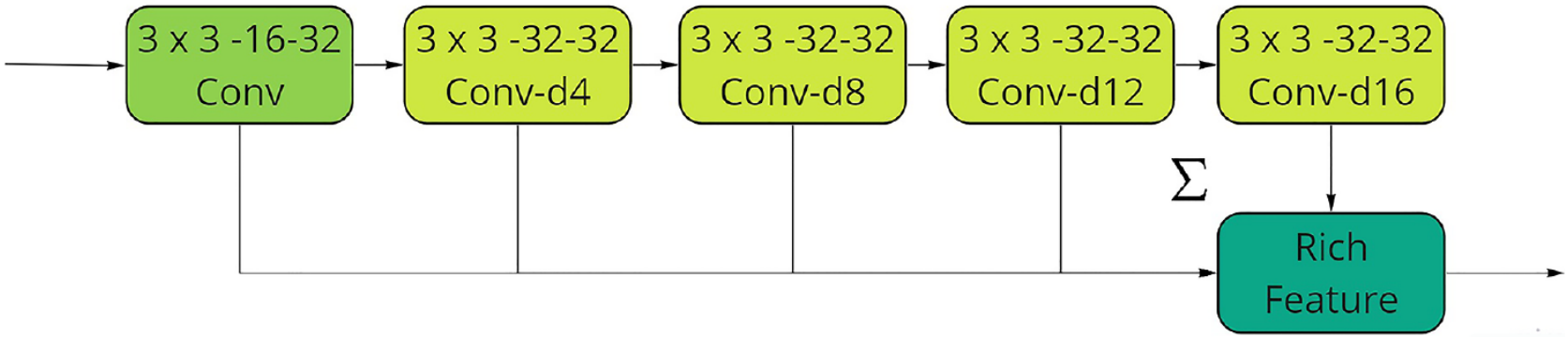

Figure: Enrichment module

Enrichment

Figure: Enrichment output

Summarizer

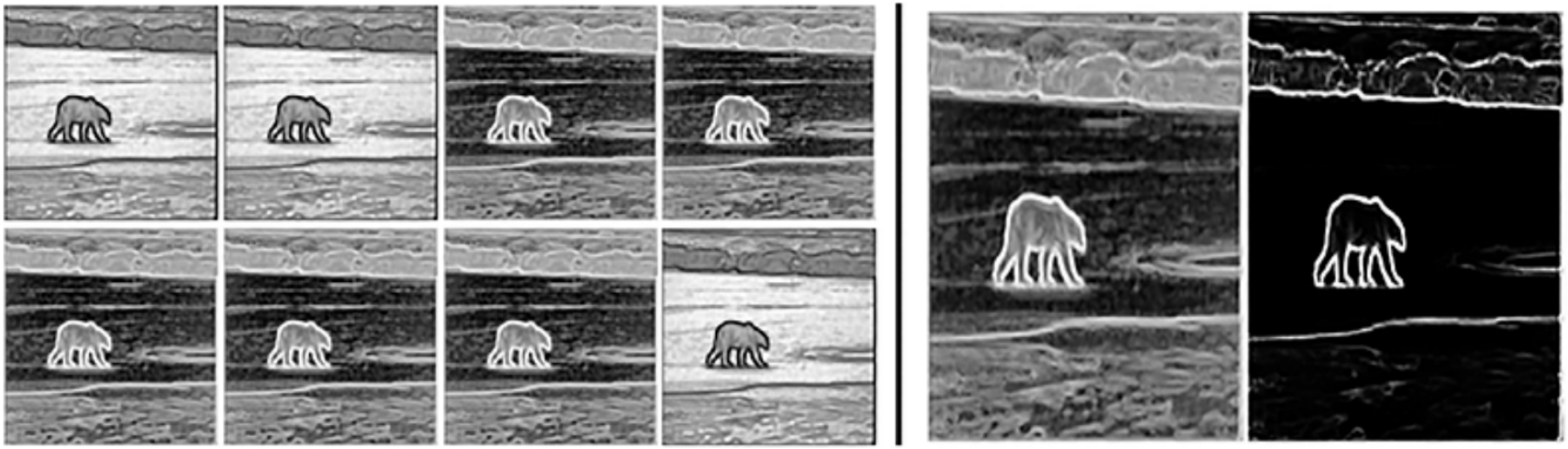

Figure: (a) Outputs of Summarizer 1 (b) Final Output

Loss Function

Loss Function

$$L(X_i; W) = \begin{cases} \alpha.log(1 - P(X_i; W)), & \text{if } y_i = 0, \\ 0, & \text{if } 0 < y_i < th, \\ \beta.log(P(X_i; W)), & \text{otherwise.} \end{cases} \quad (1)$$

y_i - ground-truth, $P(X)$ - sigmoid function, th - threshold (64),
W - weights, X_i - input

$\alpha and \beta$

$$\alpha = \gamma.\frac{Y_+}{Y}, \ \beta = \frac{Y_-}{Y} \quad (2)$$

Y_+ and Y_- denote the number of edge and non-edge ground truth pixels, respectively,
γ - is used to balance between edges and non-edges

Loss Function

Total loss function

$$LL(W) = \sum_{k=1}^{K} \sum_{i=1}^{I} L(X_i^k; W) + \sum_{i=1}^{I} L(X_i^{fuse}; W) \tag{3}$$

K - number of stages

TIN1 Architecture

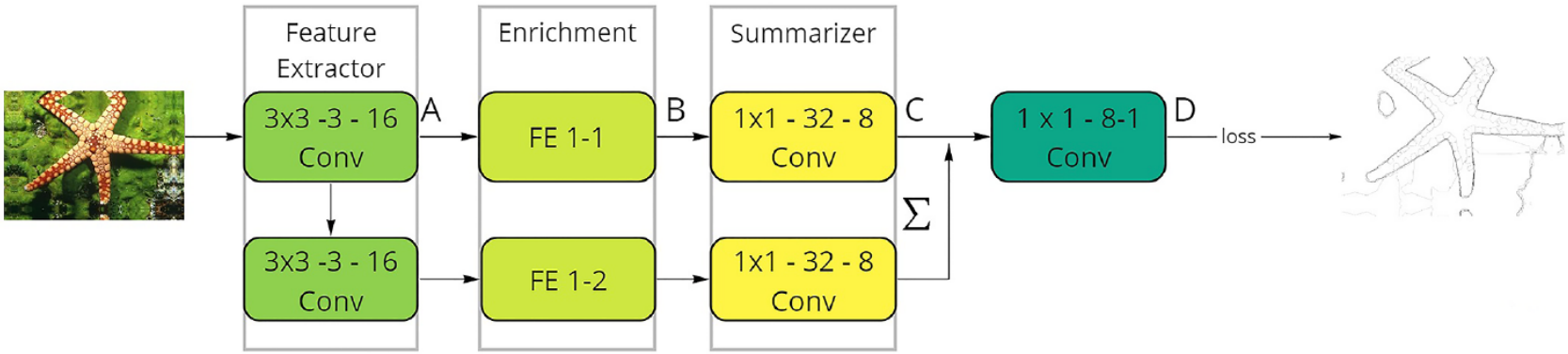

Figure: TIN1 architecture

TIN2 Architecture

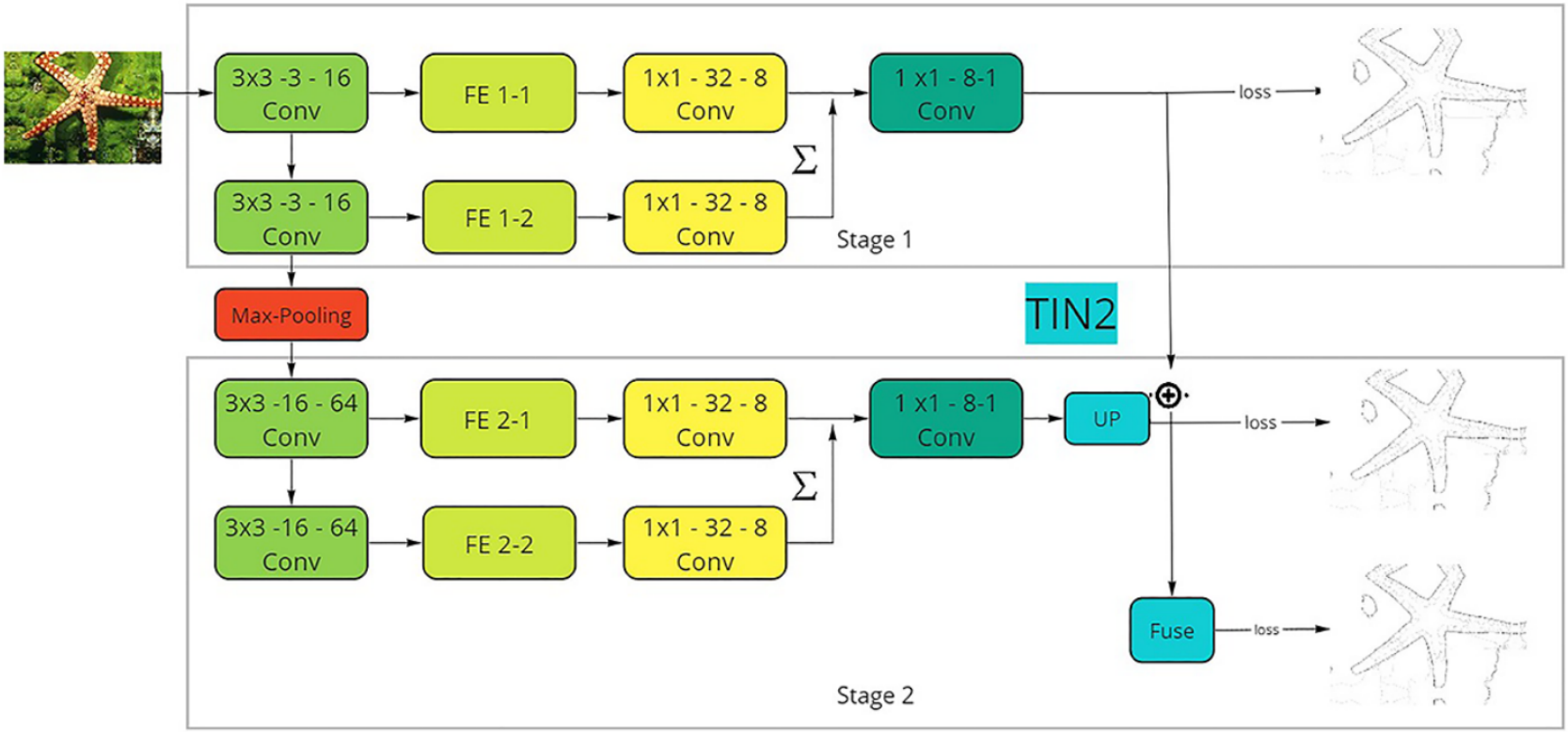

Figure: TIN2 architecture

Results

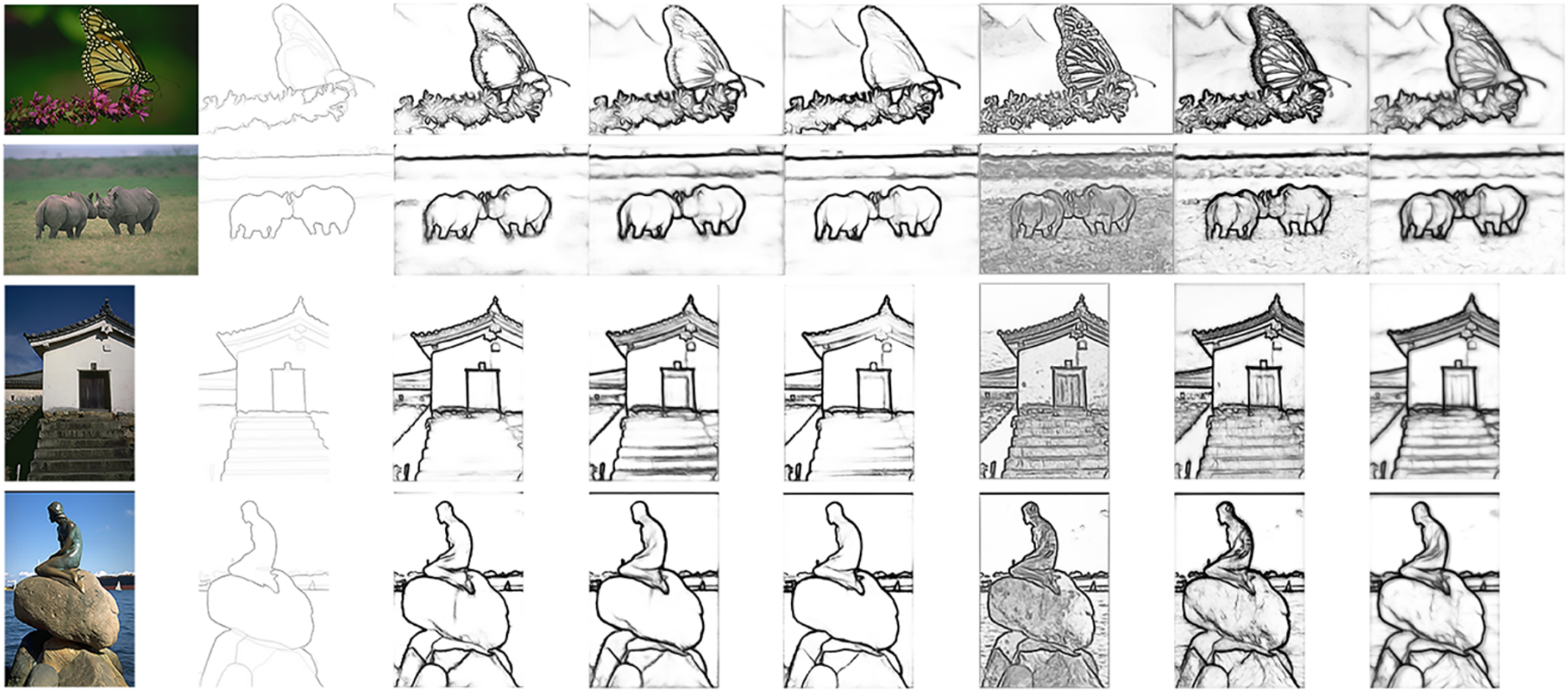

Figure: Edge Detection Comparison on BSDS500. From left to right: input, ground truth, HED, RCF, BDCN5, BDCN2, TIN1(Ours), TIN2(Ours)

Results

Method	ODS	IOS	No Params	FPS
Canny [6]	0.611	0.676		28
gPb-UCM [1]	0.729	0.755		1/240
SCG[10]	0.739	0.758		1/18
SF[12]	0.743	0.763		2.5
OEF [13]	0.749	0.772		2/3
DeepEdge [14]	0.753	0.772	-	1/1000
DeepContour[17]	0.757	0.776	-	1/30
HED [3]	0.782	0.804	14.7	30
CEDN [4]	0.788	0.804	119.6	10
RCF [17]	0.806	0.823	14.8	30
RCF-MS [17]	0.811	0.830	14.8	10
CED [16]	0.794	0.811	21.4	30
CED-MS [16]	0.815	0.834	21.4	10
LPCB [18]	0.808	0.824	-	30
LPCB-MS [18]	0.815	0.834	-	10
BDCN2 [2]	0.766	-	0.28	
BDCN3 [2]	0.796	-	2.26	
BDCN4 [2]	0.812	-	8.69	
BDCN [2]	0.820	0.838	16.3	
BDCN-MS [2]	0.828	0.844	16.3	
TIN1 (ours)	0.749	0.772	0.08	30
TIN2 (ours)	0.772	0.795	0.24	30

References

Jan Kristanto Wibisono and Hsueh-Ming Hang (28 May 2020)

Traditional method inspired Deep Neural Network for Edge Detection

The End