ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation
(2016)

Adam Paszke et al.
Resume

March 6, 2019

---

1 Introduction

The authors propose a new neural architecture optimized for fast inference without loosing on accuracy, with such a ligh weight network, we're capable of using it in small embeddded devices with limited memory and computation resources.

2 Model

The architecture of the network is described in the Table below, we have 7 sections in total, delimited by horizental lines:

Name	Type	Output size
initial		$16 \times 256 \times 256$
bottleneck1.0	downsampling	$64 \times 128 \times 128$
$4\times$ bottleneck1.x		$64 \times 128 \times 128$
bottleneck2.0	downsampling	$128 \times 64 \times 64$
bottleneck2.1		$128 \times 64 \times 64$
bottleneck2.2	dilated 2	$128 \times 64 \times 64$
bottleneck2.3	asymmetric 5	$128 \times 64 \times 64$
bottleneck2.4	dilated 4	$128 \times 64 \times 64$
bottleneck2.5		$128 \times 64 \times 64$
bottleneck2.6	dilated 8	$128 \times 64 \times 64$
bottleneck2.7	asymmetric 5	$128 \times 64 \times 64$
bottleneck2.8	dilated 16	$128 \times 64 \times 64$
Repeat section 2, without bottleneck2.0		
bottleneck4.0	upsampling	$64 \times 128 \times 128$
bottleneck4.1		$64 \times 128 \times 128$
bottleneck4.2		$64 \times 128 \times 128$
bottleneck5.0	upsampling	$16 \times 256 \times 256$
bottleneck5.1		$16 \times 256 \times 256$
fullconv		$C \times 512 \times 512$

We have a number of stages each one uses bottleneck blocks, and they only use deconvolution in the last layer given how expensive the operation is.

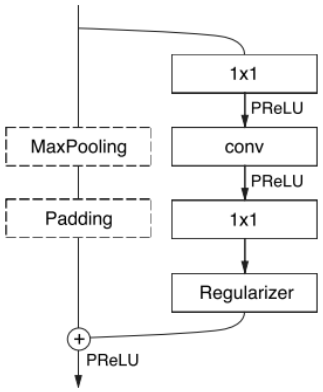
2.1 Bottleneck

Each block consists of three convolutionnal layers: a 1x1 projection that reduces the dimensionalty, a main convolutional layer: which is either regualt conv layer (sometimes a conv 3x3 is replaced by

two convs: 1x5 followed by a 5x1, also called asymetic conv), dilated 3x3 convs or 3x3 deconvolution, and then a 1x1 expansion, each time there conv layer we have BatchNorm followed by PReLU.

For bottleneck2.0, we have a regulized before and after it, the authors experimented with stochastic depth, where we randomly drop whole bottleneck layers, but after using spatial dropout, they obtained better results.

Side note: Spatial Dropout given that in each feature map of size 1xHxW, the adjacent neurones are very similar, if we used regular dropout, we end up with droped neurones in some feature map, but the adjacent neurones in the same features maps of a given droped neurone are likely to be kept intact, and after calculating the gradient, given that they are very similar we end up with only a scaled down value of the gradients, so the authors of "Efficient Object Localization Using Convolutional Networks" proposed to drop the whole feature map, so for a prob = 0.3 and C number of feature maps, we'll drop 30% of the feature maps, so $0.3 \times C \times H \times W$ neurones

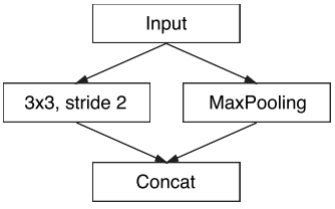


- In the decoder (stages 5 - 6), and max pooling is replaced with max unpool, and padding is replaced with spatial convolution without biases (they generally do not use biases). They also use a small decoder, the idea is that the encoder is responsible for providing the information for processing and filtering, the sole purpose of the decoder is to upsample the output and and finetune the details.

- Usage of PReLU: the authors did not get good results when adding ReLUs to the outputs of batchnorm, so they used PReLUs that learn the negative slope of the non linearities. in the encoder the PReLU in the main branch behaves like a regular relu, but in the bottleneck, they have negative values (inverting the scaling down the negative values), this might be that the network avoid having a indentity given the limited depth of the network and it need to quickly filter out information, in the decodet the weights become positive and we learn a function close to identity (so the decoder only fine tume the encoder outputs).

2.2 Initial block

The initial stage contains only one block, with a parallel maxpool and and a strided conv, generally we have a pooling to reduce the spatial dimensions followed by a convolution, this reduces the computation, but introduces a representation bottlenetck, the alternative is to pool after the conv that double the depth, so the computation is greater, the authors proposed parallel pooling and strided that are concatenated to solve reduce the computation and maintain the representation.



3 Experiments

In the training, they first use only the encoder, train it with a learning rate of $5e-4$, only using a downsampled version of the targets, and then use the decoder to train the whole network with the full size of the masks/targets, this done using batches if 10, with an L2 regulirazation with a weight decay of $2e-4$ and a class weight of $w_{\text{class}} = \frac{1}{\ln(c+p_{\text{class}})}$.

Table 4: Cityscapes test set results

Model	Class IoU	Class iIoU	Category IoU	Category iIoU
SegNet	56.1	34.2	79.8	**66.4**
ENet	**58.3**	**34.4**	**80.4**	64.0

Table 5: Results on CamVid test set of (1) SegNet-Basic, (2) SegNet, and (3) ENet

Model	Building	Tree	Sky	Car	Sign	Road	Pedestrian	Fence	Pole	Sidewalk	Bicyclist	Class avg.	Class IoU
1	75.0	84.6	91.2	**82.7**	36.9	93.3	55.0	47.5	**44.8**	74.1	16.0	62.9	n/a
2	**88.8**	**87.3**	92.4	82.1	20.5	**97.2**	57.1	49.3	27.5	84.4	30.7	65.2	**55.6**
3	74.7	77.8	**95.1**	82.4	**51.0**	95.1	**67.2**	**51.7**	35.4	**86.7**	**34.1**	**68.3**	51.3

Table 6: SUN RGB-D test set results

Model	Global avg.	Class avg.	Mean IoU
SegNet	**70.3**	**35.6**	**26.3**
ENet	59.5	32.6	19.7