SegNet: A Deep ConvolutionalEncoder-Decoder Architecture for ImageSegmentation
(2016)

Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla
Resume

February 12, 2019

---

1 Introduction

Segnet is also one of the architecture used in image segmentation relying on encoder-decoder variants, the novelty of segnet lies in the manner in which the decoder upsamples its lower resolution feature maps, this is done using the max pooling indices computed in the max pooling step of the corresponding encoder to perform non linear upsamling, this eliminates the need for applying deconvolution thus reducing the number of parameters to be learned, the upsampled maps are sparse, and to produce dence features, we convolve the umsampled feature maps usign tainable filters.

2 Architecture

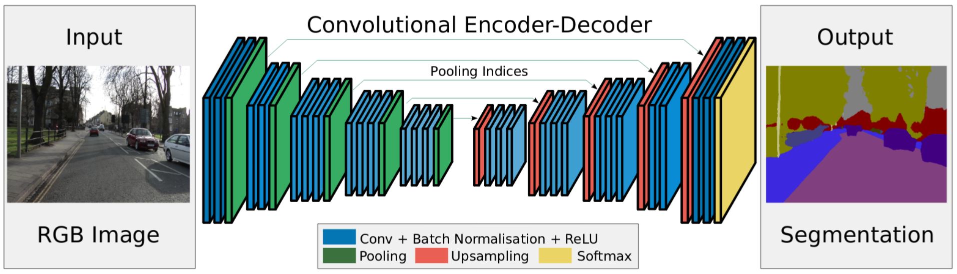

Fig. 2. An illustration of the SegNet architecture. There are no fully connected layers and hence it is only convolutional. A decoder upsamples its input using the transferred pool indices from its encoder to produce a sparse feature map(s). It then performs convolution with a trainable filter bank to densify the feature map. The final decoder output feature maps are fed to a soft-max classifier for pixel-wise classification.

Segnet has an encoder network and a corresponding decoder network, followed by a final pixelwise classification layer, the encoder consists of 13 convolutional layers which corresponds to the VGG16 netword with the FC layers descarded, this reduces the number of parameters from 134M to 14.7M, each encoder layer has a corresponding decoder layer and hence the deocder layer is also of 13 layers, the final decoder output is fed into a multiclass softmax classifier to produce class probabilities for each pixel independently.

- Each encoder in the encoder network performs convolution with a filter bank to produce a set of feature maps. These are then batch normalized. Then an element-wise rectified-linear non-linearity (ReLU) max(0, x) is applied. Following that, max-pooling with a 22 window and stride 2 (non-overlapping window) is performed and the resulting output is sub-sampled by a factor of 2. Max-pooling is used to achieve translation invariance over small spatial shifts in the input image.

- Max pooling helps achieve translation invariance over small spatial shifts, but also reducing the spatial resolution, this lossy representation is not beneficial for segmentation when boundary information is important, to solve this we can eihter capture and store boundary information before sub sampling in the encoder, this is not practical given that this takes a larger amount of memory, one possible solution is to only store the pooling indices and use them in the decoder parts, which is more efficient that storing and memorizing feature maps.

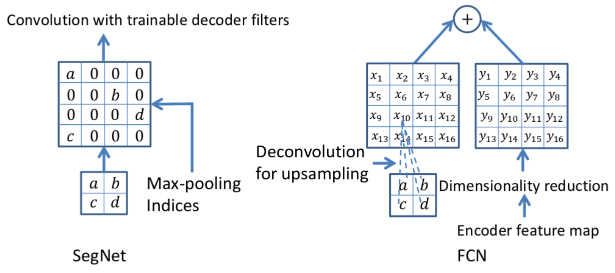

Fig. 3. An illustration of SegNet and FCN [2] decoders. a, b, c, d correspond to values in a feature map. SegNet uses the max pooling indices to upsample (without learning) the feature map(s) and convolves with a trainable decoder filter bank. FCN upsamples by learning to deconvolve the input feature map and adds the corresponding encoder feature map to produce the decoder output. This feature map is the output of the max-pooling layer (includes sub-sampling) in the corresponding encoder. Note that there are no trainable decoder filters in FCN.

The analysis of the network and its performances is done on a smaller variants of segnet.

				Median frequency balancing							Natural frequency balancing						
		Storage	Infer	Test				Train			Test				Train		
Variant	Params (M)	multiplier	time (ms)	G	C	mIoU	BF	G	C	mIoU	G	C	mIoU	BF	G	C	mIoU
Fixed upsampling																	
Bilinear-Interpolation	0.625	0	24.2	77.9	61.1	43.3	20.83	89.1	90.2	82.7	82.7	52.5	43.8	23.08	93.5	74.1	59.9
Upsampling using max-pooling indices																	
SegNet-Basic	1.425	1	52.6	82.7	62.0	47.7	35.78	94.7	96. 2	92.7	84.0	54.6	46.3	36.67	96.1	83.9	73.3
SegNet-Basic-EncoderAddition	1.425	64	53.0	83.4	**63.6**	48.5	35.92	94.3	95.8	92.0	**84.2**	56.5	**47.7**	36.27	95.3	80.9	68.9
SegNet-Basic-SingleChannelDecoder	0.625	1	33.1	81.2	60.7	46.1	31.62	93.2	94.8	90.3	83.5	53.9	45.2	32.45	92.6	68.4	52.8
Learning to upsample (bilinear initialisation)																	
FCN-Basic	0.65	11	24.2	81.7	62.4	47.3	**38.11**	92.8	93.6	88.1	83.9	55.6	45.0	**37.33**	92.0	66.8	50.7
FCN-Basic-NoAddition	0.65	n/a	23.8	80.5	58.6	44.1	31.96	92.5	93.0	87.2	82.3	53.9	44.2	29.43	93.1	72.8	57.6
FCN-Basic-NoDimReduction	1.625	64	44.8	**84.1**	63.4	**50.1**	37.37	**95.1**	**96.5**	**93.2**	83.5	**57.3**	47.0	37.13	**97.2**	**91.7**	**84.8**
FCN-Basic-NoAddition-NoDimReduction	1.625	0	43.9	80.5	61.6	45.9	30.47	92.5	94.6	89.9	83.7	54.8	45.5	33.17	95.0	80.2	67.8

TABLE 1
Comparison of decoder variants. We quantify the performance using global (G), class average (C), mean of intersection over union (mIoU) and a semantic contour measure (BF). The testing and training accuracies are shown as percentages for both natural frequency and median frequency balanced training loss function. SegNet-Basic performs at the same level as FCN-Basic but requires only storing max-pooling indices and is therefore more memory efficient during inference. Note that the theoretical memory requirement reported is based only on the size of the first layer encoder feature map. FCN-Basic, SegNet-Basic, SegNet-Basic-EncoderAddition all have high BF scores indicating the need to use information in encoder feature maps for better class contour delineation. Networks with larger decoders and those using the encoder feature maps in full perform best, although they are least efficient in terms of inference time and memory.

After analyzing the performaces of each variants, these general points can be deduced:

1. The best performance is achieved when encoder feature maps are stored in full. This is reflected in the semantic contour delineation metric (BF) most clearly.
2. When memory during inference is constrained, then compressed forms of encoder feature maps (dimensionality reduction, max-pooling indices) can be stored and used with an appropriate decoder (e.g. SegNet type) to improve performance.
3. Larger decoders increase performance for a given encoder network.

3 Training

All the encoder and decoder weights are initialized using He initialization, all variants are trained using SGD with momentum of 0.9 with a LR of 0.1 with a batch if 12 images. for the loss, they use cross-entropy loss as the objective function for training the network. The loss is summed up over all the pixels in a mini-batch. When there is large variation in the number of pixels in each class in the training set (e.g road, sky and building pixels dominate the CamVid dataset) then there is a need to weight the loss differently based on the true class. This is termed class balancing.

They usemedian frequency balancing[13] where theweight assigned to a class in the loss function is the ratio of themedian of class frequencies computed on the entire training setdivided by the class frequency. This implies that larger classes in the training set have a weight smaller than 1 and the weight sof the smallest classes are the highest. They also experimented with training the different variants without class balancing or equivalently using natural frequency balancing.

3.1 Metrics

To compare the quantitative performance of the different decodervariants, they use three commonly used three performance measures: global accuracy (G), class average accuracy (C), mean IoU (mIoU), and also the Berkeley contour matching score commonly used to evaluate unsupervised image segmentation quality

3.2 Results

Network/Iterations	40K				80K				>80K				Max iter
	G	C	mIoU	BF	G	C	mIoU	BF	G	C	mIoU	BF	
SegNet	88.81	59.93	50.02	35.78	89.68	69.82	57.18	42.08	90.40	71.20	60.10	46.84	140K
DeepLab-LargeFOV [3]	85.95	60.41	50.18	26.25	87.76	62.57	53.34	32.04	88.20	62.53	53.88	32.77	140K
DeepLab-LargeFOV-denseCRF [3]	not computed								89.71	60.67	54.74	40.79	140K
FCN	81.97	54.38	46.59	22.86	82.71	56.22	47.95	24.76	83.27	59.56	49.83	27.99	200K
FCN (learnt deconv) [2]	83.21	56.05	48.68	27.40	83.71	59.64	50.80	31.01	83.14	64.21	51.96	33.18	160K
DeconvNet [4]	85.26	46.40	39.69	27.36	85.19	54.08	43.74	29.33	89.58	70.24	59.77	52.23	260K

TABLE 3
Quantitative comparison of deep networks for semantic segmentation on the CamVid test set when trained on a corpus of 3433 road scenes *without class balancing*. When end-to-end training is performed with the same and fixed learning rate, smaller networks like SegNet learn to perform better in a shorter time. The BF score which measures the accuracy of inter-class boundary delineation is significantly higher for SegNet, DeconvNet as compared to other competing models. DeconvNet matches the metrics for SegNet but at a much larger computational cost. Also see Table 2 for individual class accuracies for SegNet.

Network/Iterations	80K				140K				>140K				Max iter
	G	C	mIoU	BF	G	C	mIoU	BF	G	C	mIoU	BF	
SegNet	70.73	30.82	22.52	9.16	71.66	37.60	27.46	11.33	72.63	44.76	31.84	12.66	240K
DeepLab-LargeFOV [3]	70.70	41.75	30.67	7.28	71.16	42.71	31.29	7.57	71.90	42.21	32.08	8.26	240K
DeepLab-LargeFOV-denseCRF [3]	not computed								66.96	33.06	24.13	9.41	240K
FCN (learnt deconv) [2]	67.31	34.32	24.05	7.88	68.04	37.2	26.33	9.0	68.18	38.41	27.39	9.68	200K
DeconvNet [4]	59.62	12.93	8.35	6.50	63.28	22.53	15.14	7.86	66.13	32.28	22.57	10.47	380K

TABLE 4
Quantitative comparison of deep architectures on the SUNRGB-D dataset when trained on a corpus of 5250 indoor scenes. Note that only the RGB modality was used in these experiments. In this complex task with 37 classes all the architectures perform poorly, particularly because of the smaller sized classes and skew in the class distribution. DeepLab-Large FOV, the smallest and most efficient model has a slightly higher mIoU but SegNet has a better G,C,BF score. Also note that when SegNet was trained with *median frequency class balancing* it obtained 71.75, 44.85, 32.08, 14.06 (180K) as the metrics.