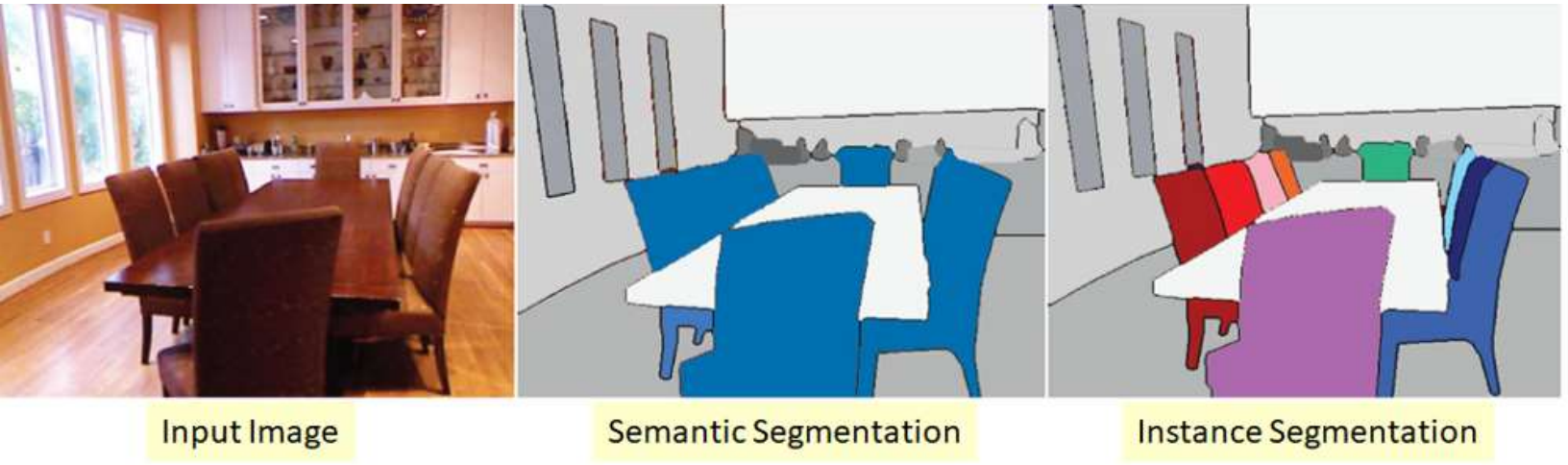
Review: DeepMask (Instance Segmentation)

An Instance Segment Proposal Method Driven by Convolutional Neural Networks

SH Tsang Follow
Dec 19, 2018 · 7 min read

This time, **DeepMask**, by **Facebook AI Research (FAIR)**, is reviewed. Starting from AlexNet, high accuracy is obtained by convolutional neural network (CNN) for image classification, numerous CNN approaches are developed for other tasks such as object detection, semantic segmentation, and instance segmentation. DeepMask is the CNN approach for instance segmentation.



Semantic Segmentation vs Instance Segmentation

- **Image Classification**: Classify the main object category within an image.
- **Object Detection**: Identify the object category and locate the position using a bounding box for every known object within an image.
- **Semantic Segmentation**: Identify the object category of each pixel for every known object within an image. **Labels are class-aware.**
- **Instance Segmentation**: Identify each object instance of each pixel for every known object within an image. **Labels are instance-aware.**

Some Differences from Semantic Segmentation

- **More understanding on the instance individuals.**
- **Reasoning about occlusion.**
- Essential to tasks such as counting the number of objects.

Some Differences from Object Detection

- A bounding box is a very **coarse object boundary**, many pixels irrelevant to the detected object are also included in the bounding box.
- And Non Maximum Suppression (NMS) will **suppress occluded objects or slanted objects.**

Thus, Instance Segmentation is one level increase in difficulty!!!

And DeepMask is the **2015 NIPS** paper with more than **300 citations**. Though it is a paper published in the year of 2015, it is one of the earliest paper using CNN for instance segmentation. It is worth to study it to know the development of deep-learning-based instance segmentation.

Since a region proposal can be generated based on the predicted segmentation mask, object detection task can also be performed.

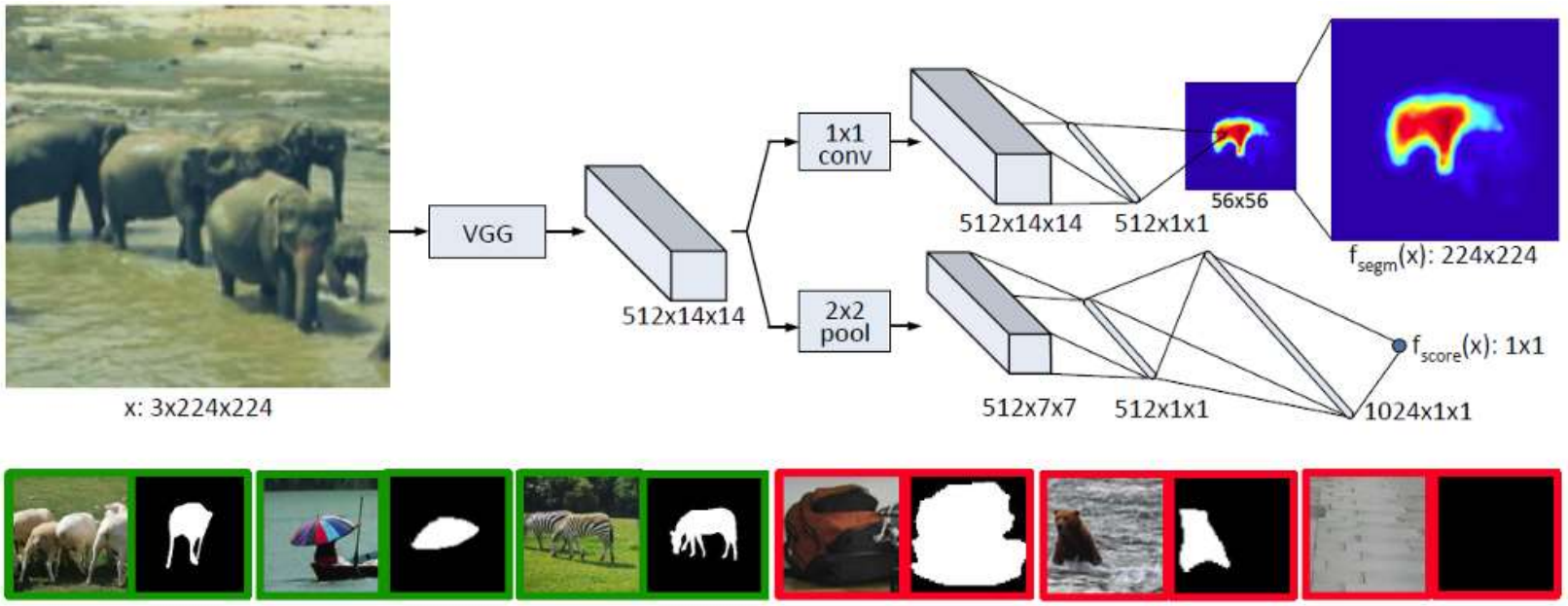
. . .

What Are Covered

1. **Model Architecture**
2. **Joint Learning**
3. **Full Scene Inference**
4. **Results**

. . .

1. Model Architecture



Model Architecture (Top), Positive Samples (Green, Left Bottom), Negative Samples (Red, Right Bottom)

Left Bottom: Positive Samples

A label $yk=1$ is given for k-th positive sample. To be a positive sample, two criteria need to be satisfied:

- **The patch contains an object roughly centered in the input patch**.
- **The object is fully contained in the patch and in a given scale range.**

When $yk=1$, the **ground truth mask** mk **has positive values** for the pixels which belong to the **single object** located in the centre of the image patch.

Right Bottom: Negative Samples

Otherwise, a label $yk=-1$ is given for a negative sample even the object is partially present. **When** $yk=-1$**, the mask is not used.**

Top, Model Architecture: Main Branch

The model as shown above, given the input image patch x, after feature extraction by VGGNet, The fully connected (FC) layers originated in VGGNet are removed. The last max pooling layer in VGGNet is also removed, thus the output before splitting into two paths are of the size of 1/16 of input. For example as above, the input is 224×224 (3 is the number of channels in the input image, i.e. RGB), the output at the end of main branch is $(224/16)\times(224/16)=14\times14$. (512 is the number of feature maps after convolution.)

There are two paths after VGGNet:

- **The first path is to predict the class-agnostic segmentation mask, i.e. *fsegm*(*x*).**
- **The second path is to assign a score corresponding to how likely the patch is to contain an object**, i.e. ***fscore*(*x*)**.

Top, First Path: Predicting Segmentation Map

1×1 convolution is performed first without changing the number of feature maps, **non-linear mapping without dimension reduction** is done here. After that, **two FC layers** are performed. (It is notice that there is **no ReLU in between these two FC layers!**)

Unlike in semantic segmentation, the network must output a mask for a single object even when multiple objects are present. (Just like the elephant in the centre of the input image as shown above.)

Finally, **a 56×56 segmentation map is generated**. And a simple **bilinear interpolation** is to upsample the segmentation map to **224×224**.

Top, Second Path: Predicting Object Score

2×2 max pooling followed by two FC layers. Finally, one single value Predicted Object Score, *fscore*(*x*), is obtained. Since positive samples are given based on the two criteria mentioned above, *fscore*(*x*) is to predict whether the input image has satisfied these two criteria.

. . .

2. Joint Learning

2.1. Loss Function

The network is trained to jointly learn the pixel-wise segmentation map ***fsegm*(*xk*)** at each location (i,j) and the predicted object score ***fscore*(*xk*).** The loss function is shown as below:

$$\mathcal{L}(\theta) = \sum_k \left(\frac{1+y_k}{2w^o h^o} \sum_{ij} \log(1 + e^{-m_k^{ij} f_{segm}^{ij}(x_k)}) + \lambda \log(1 + e^{-y_k f_{score}(x_k)}) \right)$$

To be brief, the loss function is a sum of binary logistic regression losses, one for each location of the segmentation network ***fsegm*(*xk*)** and one for the object score ***fscore*(*xk*)**. The first term implies that we only backpropagate the error over the segmentation path if $yk=1$.

If $yk=-1$, i.e. the negative sample, the first term become 0 and will not contribute to the loss. Only the second term contributes the loss.

For the sake of data balance, equal number of positive and negative samples is used.

2.2. Other Details

Batch size of 32 is used. **Pretrained ImageNet model** is used. There are **75M parameters** in total. The model takes around **5 days** to train on a Nvidia Tesla K40m.

. . .

3. Full Scene Inference

3.1. Multiple Locations and Scales

During inference (testing), the model is applied densely at **multiple locations with a stride of 16 pixels**, and **multiple scales from 1/4 to 2 with a step size of square root of 2**. This ensures that there is at least one tested image patch that fully contains each object in the image.

3.2. Fine Stride Max Pooling

Since the input test image is larger than the training input patch size, we need a corresponding **2D scoring map** as an output rather than one single scoring value. An interleaving trick is used before the last max pooling layer for the scoring branch, i.e. the **Fine Stride Max Pooling** proposed in OverFeat.

To be brief, multiple max pooling is done on the feature map. A pixel shift is performed before each max pooling.

. . .

4. Results

4.1. MS COCO (Boxes & Segmentation Masks)

80,000 images and a total of nearly 500,000 segmented objects, are used for training. And the first 5000 images of the MS COCO 2014 are used for validated.

	Box Proposals				Segmentation Proposals						
	AR@10	AR@100	AR@1000	AUC	AR@10	AR@100	AR@1000	AUC^S	AUC^M	AUC^L	AUC
EdgeBoxes [34]	.074	.178	.338	.139	-	-	-	-	-	-	-
Geodesic [16]	.040	.180	.359	.126	.023	.123	.253	.013	.086	.205	.085
Rigor [14]	-	.133	.337	.101	-	.094	.253	.022	.060	.178	.074
SelectiveSearch [31]	.052	.163	.357	.126	.025	.095	.230	.006	.055	.214	.074
MCG [24]	.101	.246	.398	.180	.077	.186	.299	.031	.129	.324	.137
DeepMask20	.139	.286	.431	.217	.109	.215	.314	.020	.227	.317	.164
DeepMask20*	.152	.306	.432	.228	.123	.233	.314	.020	.257	.321	.175
DeepMaskZoom	.150	**.326**	**.482**	**.242**	**.127**	**.261**	**.366**	**.068**	**.263**	.308	**.194**
DeepMaskFull	.149	.310	.442	.231	.118	.235	.323	.020	.244	**.342**	.176
DeepMask	**.153**	.313	.446	.233	.126	.245	.331	.023	.266	.336	.183

Average Recall (AR) Detection Boxes (Left) and Segmentation Masks (Right) on MS COCO Validation Set (AR@n: the AR when n region proposals are generated. AUCx: x is the size of objects)

- **DeepMask20**: Trained only with objects belonging to one of the 20 PASCAL categories. AR is low compared to DeepMask which means the network is not generalized to unseen classes. (low scores for unseen classes.)
- **DeepMask20***: Similar to DeepMask but the scoring path uses the original DeepMask.
- **DeepMaskZoom**: Additional smaller scale to boost AR but with the cost of increased inference time.
- **DeepMaskFull**: The two FC layers at the path for predicting the segmentation mask are replaced by one FC layer directly mapped from the $512\times14\times14$ feature maps to the 56×56 segmentation maps. The whole architecture are with **over 300M parameters**. It is slightly inferior to DeepMask and much slower.

4.2. PASCAL VOC 2007 (Boxes)

PASCAL VOC07	AR@10	AR@100	AR@1000	AUC
EdgeBoxes [34]	.203	.407	.601	.309
Geodesic [16]	.121	.364	.596	.230
Rigor [14]	.164	.321	.589	.239
SelectiveSearch [31]	.085	.347	.618	.241
MCG [24]	.232	.462	.634	.344
DeepMask	**.337**	**.561**	**.690**	**.433**

Average Recall (AR) for Detection Boxes on PASCAL VOC 2007 Test Set

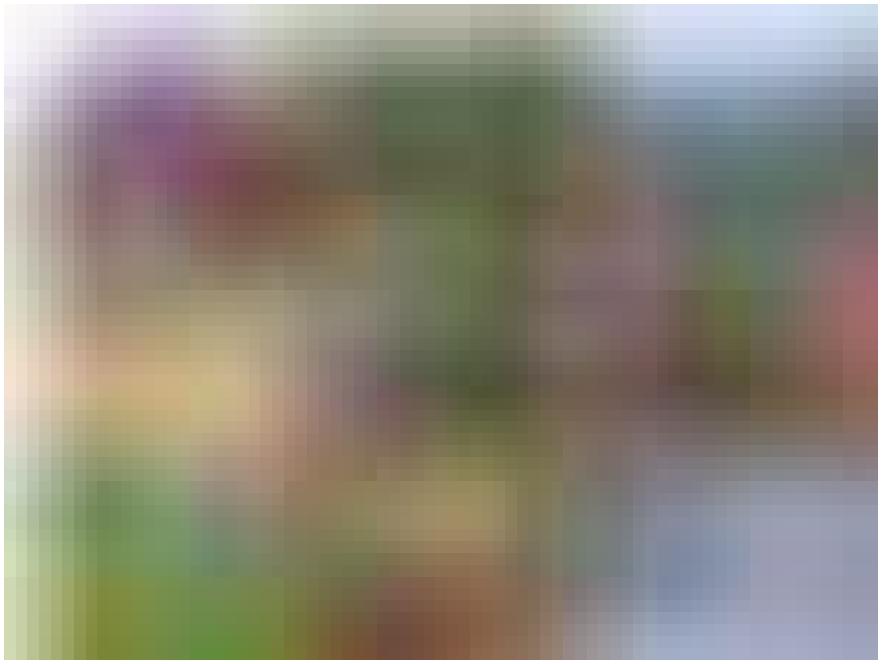
- Region proposals are generated based on the predicted segmentation masks, which can be used as the first step of object detection task.

- **Fast R-CNN using DeepMask** outperforms original Fast R-CNN using Selective Search as well as other state-of-the-art approaches.

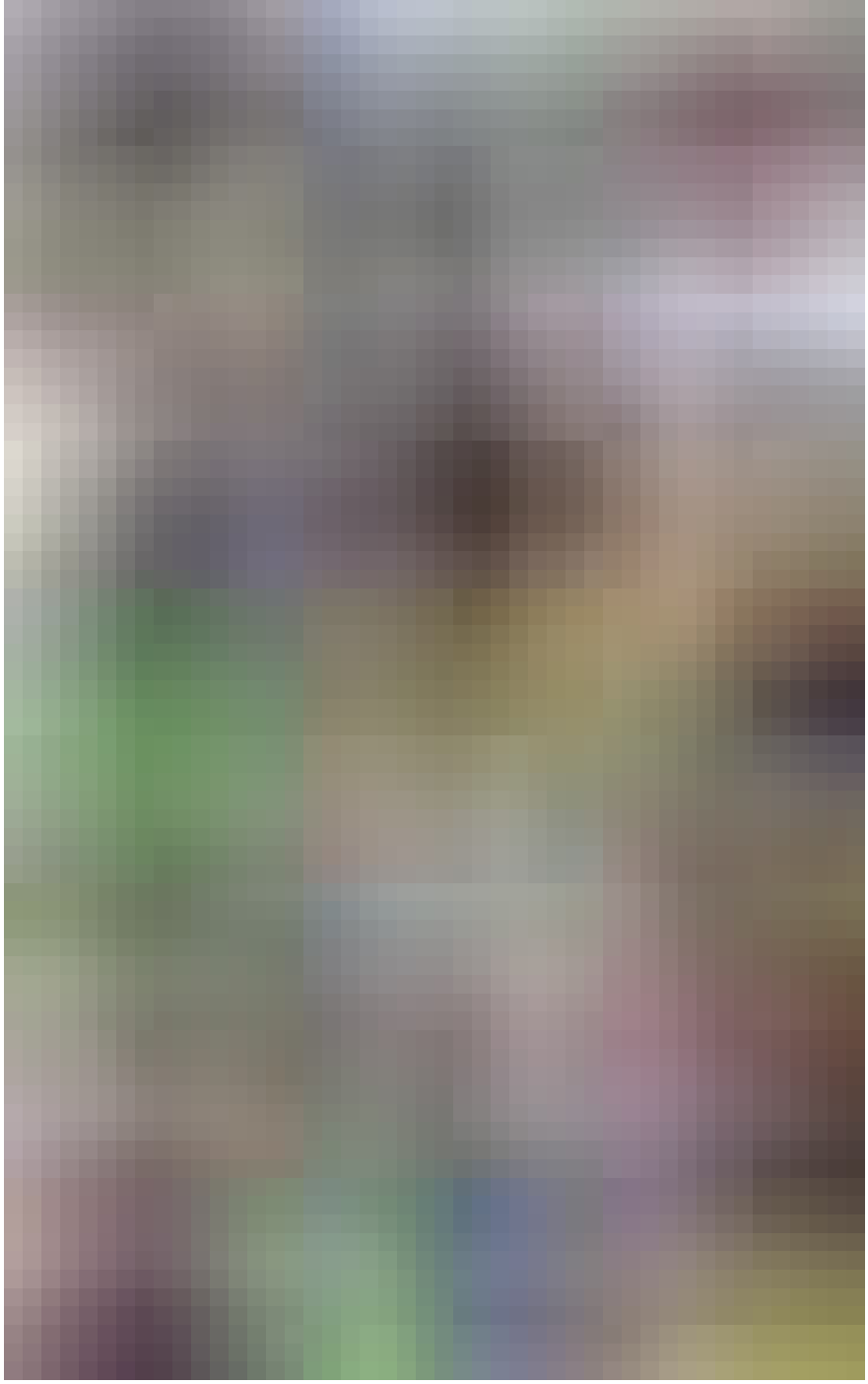
4.3. Inference Time

- The inference time in MS COCO is 1.6s per image.
- The inference time in PASCAL VOC 2007 is 1.2s per image.
- Inference time can be further dropped by about 30% by parallelizing all scales in a single batch.

4.4. Qualitative Results



DeepMask proposals with highest IoU to the ground truth on selected images from COCO. Missed objects (no matching proposals with IoU > 0.5) are marked with a red outline.



More Results from COCO. Missed objects (no matching proposals with IoU > 0.5) are marked with a red outline.

. . .

DeepMask has been updated that the VGGNet backbone is replaced by ResNet in GitHub.

After DeepMask, FAIR also invented SharpMask. Hope I can cover it later as well.

. . .

References

[2015 NIPS] [DeepMask]
Learning to Segment Object Candidates

My Related Reviews

Image Classification
[LeNet] [AlexNet] [ZFNet] [VGGNet] [SPPNet] [PReLU-Net] [DeepImage] [GoogLeNet / Inception-v1] [BN-Inception / Inception-v2] [Inception-v3] [Inception-v4] [Xception] [MobileNetV1] [ResNet] [Pre-Activation ResNet] [RiR] [RoR] [Stochastic Depth] [WRN] [FractalNet] [Trimps-Soushen] [PolyNet] [ResNeXt] [DenseNet]

Object Detection
[OverFeat] [R-CNN] [Fast R-CNN] [Faster R-CNN] [DeepID-Net] [R-FCN] [YOLOv1] [SSD] [YOLOv2 / YOLO9000]

Semantic Segmentation
[FCN] [DeconvNet] [DeepLabv1 & DeepLabv2] [ParseNet] [DilatedNet] [PSPNet]

Biomedical Image Segmentation
[CUMedVision1] [CUMedVision2 / DCAN] [U-Net] [CFS-FCN]