Recurrent Neural Networks for Semantic Instance Segmentation

Amaia Salvador[1], Míriam Bellver[2], Manel Baradad[1], Ferran Marqués[1], Jordi Torres[2], Xavier Giro-i-Nieto[1]
[1]Universitat Politècnica de Catalunya [2]Barcelona Supercomputing Center
{amaia.salvador,xavier.giro,ferran.marques}@upc.edu, {miriam.bellver,jordi.torres}@bsc.es, manel.baradad@alu-etsetb.upc.edu

Abstract

We present a recurrent model for semantic instance segmentation that sequentially generates pairs of masks and their associated class probabilities for every object in an image. Our proposed system is trainable end-to-end, does not require post-processing steps on its output and is conceptually simpler than current methods relying on object proposals. We observe that our model learns to follow a consistent pattern to generate object sequences, which correlates with the activations learned in the encoder part of our network. We achieve competitive results on three different instance segmentation benchmarks (Pascal VOC 2012, Cityscapes and CVPPP Plant Leaf Segmentation). Code is available at `https://imatge-upc.github.io/rsis`.

1. Introduction

Recurrent Neural Networks (RNNs) have been widely used in computer vision problems in which it is necessary to deal with sequential data (e.g. video action classification [13, 43], image captioning [40, 21] or visual question answering [2]). Although images are rarely treated as sequential data, the human exploration of static visual inputs is actually a sequential process [32, 1] that involves reasoning about objects that compose the scene and their relationships.

In this work, we postulate that the *semantic instance segmentation* task can be learned end-to-end with a recurrent neural network. Semantic instance segmentation aims at assigning a binary mask and a categorical label to each object in an image. It is often understood as an extension of object detection where, instead of bounding boxes, accurate binary masks must be predicted. Semantic instance segmentation can benefit from recurrent methods because, thanks to their ability to retain previous information, they can deal with complex object distributions, handle occluded instances and make predictions that are coherent with each other based on what objects have been found and what objects are yet to be discovered.

Current state of the art methods for semantic instance segmentation are, however, not based on recurrent architectures. Most methods [17, 18, 24, 23, 11, 19] extend object detection pipelines based on object proposals (e.g. Faster R-CNN [35]). Typically, these proposals outnumber the actual objects that appear in the image by a significant margin, meaning that most of them are redundant and therefore require a post-processing step such as non-maximum suppression (NMS) to discard potential duplicates. Other works [6, 12, 34] involve training deep neural networks to produce a reduced set of binary masks per image, hence avoiding redundant predictions. However, categorical labels for each mask are obtained by mapping them onto a semantic segmentation of the image obtained with a separate network that introduces an additional computation overhead.

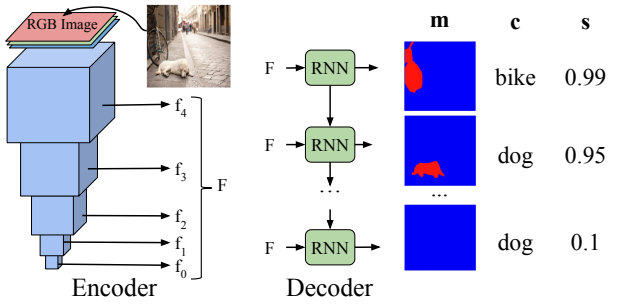

Figure 1. Recurrent model for semantic instance segmentation. Given an RGB image, an encoder extracts relevant features and a recurrent decoder learns to decompose the image by segmenting one instance at a time.

In this paper, we argue that both ranking thousands of object-prominent regions [17, 18, 24, 23, 11, 19] or decomposing the problem into two independent modules [6, 12, 34] are unnatural approaches that are far away from the sequential exploration that we as humans follow to solve related tasks such as visual search [1] or counting [33]. Existing solutions of recurrent instance segmentation [31, 36, 34] generate sequences of class-agnostic masks, from which we take a step further and develop a truly end-to-end recurrent system that *directly* provides a sequence of semantic instances as an output (i.e. both binary masks and categorical labels for all objects in the image). In addition, our model incorporates skip connections, that have been shown beneficial for the semantic segmentation task

[27, 37], but are rarely used for the related task of semantic instance segmentation.

The contributions of this work are threefold: (a) we present an end-to-end recurrent model for semantic instance segmentation that does not require further post-processing and is conceptually simpler than current methods, (b) we show its competitive performance in three instance segmentation benchmarks, and (c) we thoroughly analyze its behavior in terms of the object discovery patterns that it follows.

2. Related Work

Most works on semantic instance segmentation inherit their foundations from object detection solutions. These typically involve two-stage pipelines where a set of object-prominent locations are predicted and subsequently classified and ranked. Hariharan et al. [17] trained a two-branch network to simultaneously learn to classify bounding-box and segment proposals from [3]. This work was further extended in [18], where local activations in all convolutional layers in the network are used to refine the input segments. Chen et al. [8] present a post-processing stage for occlusion handling to refine the predictions from [17].

More recent works build upon Faster R-CNN [35], a state of the art object detection architecture that learns to predict box-level object proposals and subsequently produces classification scores for each one of them. Building from this architecture, Dai et al. [11, 10] propose multi-task network cascades where box coordinates, binary masks and class scores are predicted in chain, allowing the network to reuse information learned in previous stages. Liang et al. [24] extend the architecture in [11] with an iterative binary mask refinement. In contrast with cascade-based methods [11, 24, 10], He et al. [19] design an architecture that predicts bonding boxes, segments and class scores in parallel given the output of a fully convolutional network (hence, no chain reliance is imposed).

While proposal-based methods have shown impressive performance, they generate a large set of predictions that are independent from each other and thus highly redundant. In practice, this means that most of the predictions are thrown away in post-processing (e.g. NMS). In contrast, by using a recurrent approach, the outputs of our network are dependent from each other, which allows us to solve the task with a small number of predictions.

Other works in the literature have presented alternative methods to the proposal-based pipelines by treating the image holistically. Arnab & Torr [5] combine the outputs of an object detection and a semantic segmentation network with Conditional Random Fields (CRFs) to output a class and an instance label for each pixel in the image. Bai & Urtasun [6] learn a watershed transform using the output of a semantic segmentation network, where the obtained energy basis constitute the different object instances in the image. Brabandere et al. [12] use a metric learning objective that projects each pixel into a high-dimensional space where pixels that belong to the same object are mapped close together (or far apart otherwise). However, this method requires a post-processing stage to compose masks based on the clustering of similar pixels.

Our model is closer to recent proposals that formulate the problem of instance segmentation with recurrent approaches, where different object instances are predicted one at a time. Park & Berg [31] present a model composed of different RNN units (one per class) that receive a spatial class score map and learn to separate the different instances that are included in the input. However, their method does not scale well for datasets with a large number of categories, since a separate RNN needs to be trained for each of them. Romera-Paredes & Torr [36] train a model composed of Convolutional LSTMs [41] that receives convolutional features from a pretrained FCN [27] and outputs the separate object segments for the image. A post-processing based on CRFs is applied to their final masks. Ren & Zemel [34] propose a complex multi-task recurrent pipeline for instance segmentation that predicts the box coordinates for a different object at each time step. These object coordinates are then used to extract a sub-region of the image from which a binary mask is predicted. Both [36, 34] are class-agnostic methods and, while [34] reports results for semantic instance segmentation benchmarks, class probabilities for their predicted segments are obtained from the output of a FCN [27] trained for semantic segmentation. To the best of our knowledge, our proposed method is the first to directly tackle semantic instance segmentation with a fully end-to-end recurrent approach.

3. Model

We design an encoder-decoder architecture for semantic instance segmentation, which we depict in Figure 1. Our architecture resembles typical ones from the state of the art for semantic segmentation [27, 37], where skip connections from the layers in the encoder are used to recover low level features that are helpful to obtain accurate segmentation outputs. The main difference between these works and ours is that our decoder is recurrent, enabling the prediction of one instance at a time instead of a single semantic segmentation map where all objects are present.

First, the encoder extracts relevant features from the image, which are used in the decoder to generate accurate binary masks and their class probabilities. The decoder is a recurrent neural network that learns to decompose the different objects in the image and outputs them one at a time.

3.1. Encoder

We use a ResNet-101 [20] model pretrained with ImageNet [38] for image classification as the encoder. We trun-

cate the network at the last convolutional layer, thus removing the last pooling layer and the final classification layer. The encoder takes an RGB image $x \in \mathbb{R}^{h \times w \times 3}$ and extracts features from the different convolutional blocks of the base network $F = \text{encoder}(x)$. F contains the output of the different convolutional blocks $F = [f_0, f_1, f_2, f_3, f_4]$, where f_0 corresponds to the output of the deepest block, and f_4 is the output of the block whose input is the image.

3.2. Decoder

The decoder receives as input the convolutional features F and outputs a set of T_o predictions, being T_o variable for each input image. At any given time step t, the prediction is of the form $\{o_{m,t}, o_{c,t}, o_{s,t}\}$, where $o_{m,t} \in [0,1]^{h \times w}$ is the binary mask, $o_{c,t} \in [0,1]^C$ are the probabilities for the C different categories, and $o_{s,t} \in [0,1]$ represents the objectness score, which is the stopping criterion at test time.

Similarly to [36], we use the Convolutional Long Short-term Memory Units [41] as the basic block of our decoder, in order to naturally handle 3-dimensional convolutional features as input and preserve spatial information. We design an upsampling network composed of a series of ConvLSTM layers, whose outputs are subsequently merged with the side outputs F from the encoder. This merging can be seen as a form of skip connection that bypasses the previous recurrent layers. Such architecture allows the decoder to reuse low level features from the encoder to refine the final segmentation. Additionally, since we are using a recurrent decoder, the reliance on these features can change across different time steps. Intuitively, the decoder might ignore low-level features in the input to segment an object that shares features with previously segmented objects (e.g. an object of the same category as the previous one), while it might choose to use them to segment objects that do not share similarities with earlier predictions.

For a time step t, $h_{i,t}$ represents the output of the i^{th} ConvLSTM layer, which is obtained using the equation:

$$h_{i,t} = \text{ConvLSTM}_\text{i}(\ [\ B_2(h_{i-1,t}) \mid S_i\ ]\ ) \tag{1}$$

where B_2 is the bilinear upsampling operator by a factor of 2, $h_{i-1,t}$ is the hidden state of the previous ConvLSTM layer and S_i is the result of projecting f_i to the same dimension of $h_{i-1,t}$ via a convolutional layer.

Equation 1 is applied in chain for $i \in \{1, \ldots, n\}$, being n the number of convolutional blocks in the encoder ($n = 5$ in ResNet). $h_{0,t}$ is obtained by a ConvLSTM with S_0 as input (i.e. no skip connection):

$$h_{0,t} = \text{ConvLSTM}_0(S_0) \tag{2}$$

The number of channels of each ConvLSTM layer is determined based on its depth in the decoder. We set the first two ConvLSTM layers to have dimension D, and set the dimension of the remaining oens to be the one in the previous layer divided by a factor of 2. We set $D = 128$ for Pascal and Cityscapes, and $D = 64$ for CVPPP. All ConvLSTM layers use 3×3 kernels. Finally, a single-kernel 1×1 convolutional layer with sigmoid activation is used to obtain a final binary mask of the same resolution as the input image.

The class and stop prediction branches consist of two fully connected layers: one to predict the category of the segmented object at time step t and one to predict a stopping signal once all objects have been found. These receive the same input $h_{c,t}$, which is given by the concatenation of the max-pooled hidden states of all ConvLSTM layers in the network . Figure 2 shows the details of the recurrent decoder for a single time step.

3.3. Training

The parameters of our model are estimated by adopting a multi-task approach formulated with a cost function composed of three different terms:

Segmentation loss. Similarly to other works [36, 34], we use the soft intersection over union loss (sIoU) as the cost function between the mask predicted by our network m_1 and the ground truth mask m_2:

$$\text{sIoU}(m_1, m_2) = 1 - \frac{\sum_{i=1}^{M} m_{1,i} m_{2,i}}{\sum_{i=1}^{M} m_{1,i} + m_{2,i} - m_{1,i} m_{2,i}} \tag{3}$$

where M is the number of elements in both m_1 and m_2.

We do not impose any specific instance order to match the predictions of our model with the objects in the ground truth. Instead, we let the model decide which output permutation is the best and sort the ground truth accordingly. We assign a prediction for each of the ground truth masks by means of the Hungarian algorithm, using the sIoU as the cost function. Given a sequence of predicted masks $o_m = \{o_{m,1}, ..., o_{m,T_o}\}$ and the list of ground truth masks $g_m = \{g_{m,1}, ..., g_{m,T_g}\}$, the segmentation loss L_m can be expressed as:

$$L_m(o_m, g_m, \delta) = \sum_{t=1}^{T_o} \sum_{t'=1}^{Tg} sIoU(o_{m,t}, g_{m,t'}) \delta_{t,t'} \tag{4}$$

where δ is the matrix of assignments. $\delta_{t,t'}$ is 1 when the predicted and ground truth masks $o_{m,t}$ and $g_{m,t'}$ are matched and 0 otherwise. In the case where $T_o > T_g$, gradients for predictions at $t > T_g$ are ignored.

Classification loss. Our network outputs class probabilities for each of the predicted masks. Given the sequence of class probabilities $o_c = \{o_{c,1}, ..., o_{c,T_o}\}$ and the ground truth one-hot class vectors $g_c = \{g_{c,1}, ..., g_{c,T_g}\}$, the classification loss is computed as the negative log likelihood

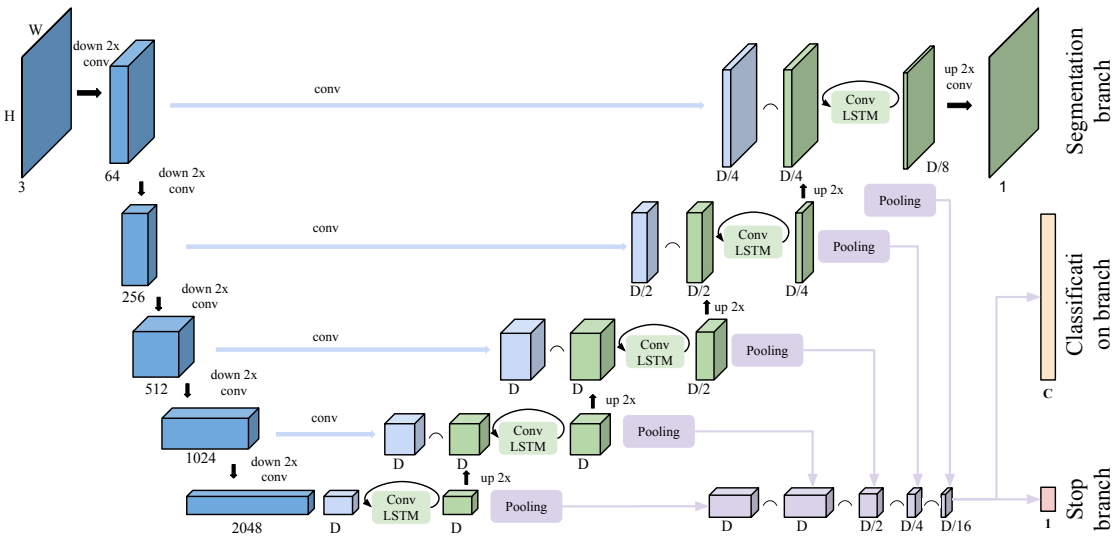

Figure 2. Proposed architecture for end-to-end semantic instance segmentation.

between the matched pairs determined by δ:

$$L_c(o_c, g_c, \delta) = -\sum_{t=1}^{T_o}\sum_{t'=1}^{Tg} g_{c,t'} log(o_{c,t})\delta_{t,t'} \quad (5)$$

Stop loss. Finally, we optimize our model to learn to stop once all objects have been found in the image.

$$L_s(o_s, g_s) = -\sum_{t=1}^{T_g+1} g_{s,t} log(o_{s,t}) + (1-g_{s,t}) log(o_{s,t}) \quad (6)$$

where $t_{s,t}$ = 1 for $t \in \{1, ..., T_t\}$ and 0 otherwise.

The total loss is the weighted sum of the three terms: $L_t = L_m + \lambda L_c + \gamma L_s$.

4. Experiments

In this section we introduce the datasets and metrics used to evaluate our method, describe the implementation details and discuss the results of our proposed approach.

4.1. Datasets and metrics

We evaluate our models on three benchmarks previously used for semantic instance segmentation:

CVPPP Plant Leaf Segmentation [29]: Small dataset of images of different plants. We follow the same scheme as in [36, 34], using only 128 images from the A1 subset for training. Results are evaluated by the dataset authors over 33 test images.

Pascal VOC 2012 [14]: Contains objects of 20 different categories. We use the additional annotations from [16] to train our models and we evaluate on the original VOC 2012 validation set, composed of 1,449 images.

Cityscapes [9]: 5,000 street-view images containing objects of 8 different categories. The dataset is split in 2,975 images for training, 500 for validation and 1,525 for testing.

Models trained for the CVPPP dataset are evaluated with the symmetric best dice (SBD) and the difference in count (DiC) metric, as in [29]. For Cityscapes and Pascal VOC, we use common evaluation practices in [17] and use the average precision AP at different IoU thresholds.

4.2. Experimental setup

We use the Adam optimizer [22] with a learning rate of 10^{-3} for all layers in the decoder, 10^{-6} for the layers in the encoder, and a weight decay of 10^{-6} in all layers in the network. Our models are implemented in PyTorch.

We train our multi-task model by subsequently adding penalty terms to the loss function one at a time as training progresses. In our experiments we observe that while the penalty term for instance classification L_c quickly converges, the task of segmenting one object at a time is much more challenging to learn. We hypothesize that this is mainly due to the fact that the encoder we use is pretrained for image classification and not segmentation. To facilitate convergence, we first train the network for 20 epochs with $\lambda = 0$ and set it to 0.1 after 20 epochs. Similarly, the penalty term L_s also converges quickly, therefore we set it to 0 and activate it after the model converges for $L_t = L_m + 0.1L_c$. Finally we add the stopping loss term L_s to the cost function with $\gamma = 0.5$ for Pascal, $\gamma = 0.1$ for CVPPP and $\gamma = 1.0$ for Cityscapes and resume training until convergence.

We resize images to 256×256 pixels for Pascal, 256×512 for Cityscapes and 400×400 for CVPPP. We use typical data augmentation strategies during training.

When training for datasets with a high number of objects per image (i.e. Cityscapes and CVPPP) we use curriculum

	R	$SBD \uparrow$	$DiC \downarrow$
MSU [39]		66.7 ($\pm$7.6)	2.3 ($\pm$1.6)
Nottingham [39]		68.3 ($\pm$6.3)	3.8 ($\pm$2.0)
Wageningnen [42]		71.1 ($\pm$6.2)	2.2 ($\pm$1.6)
IPK [30]		74.4 ($\pm$4.3)	2.6 ($\pm$1.8)
PRIAn [15]		-	1.3 ($\pm$1.2)
Brabandere et al. [12]		84.2 (-)	**0.8** (-)
RIS + CRF [36]	✓	66.6 ($\pm$8.7)	1.1 ($\pm$0.9)
Ren & Zemel [34]	✓	**84.9** ($\pm$4.8)	**0.8** ($\pm$1.0)
Ours	✓	70.6 ($\pm$11.2)	1.2 ($\pm$1.1)

Table 1. Results on the CVPPP plant leaf segmentation benchmark. We report values for both SBD (the higher the better) and DiC (the lower the better). The R column indicates whether the method is recurrent.

learning [7] to guide the optimization process, where we begin optimizing the model to predict only two objects and increase this value by one once the validation loss plateaus. In practice, we have found this approach to be effective for these two datasets, but unnecessary in the case of the Pascal dataset where images have fewer objects.

4.3. Single-class instance segmentation

We first evaluate our models in the CVPPP Plant Leaf Segmentation dataset. We compare with participants to the CVPPP challenge [39, 42, 30, 15], the two recurrent models in [36, 34] and the metric learning based approach from [12]. Table 1 summarizes the results. Our method outperforms the one by Romera-Paredes & Torr [36], whose approach is also recurrent and their network is the most similar to ours (i.e. they use a fully convolutional neural network whose input is the raw image and the output are the consecutive masks). The recurrent model in [34] obtains better results in this challenge, especially in terms of SBD. However, [34] uses a complex pipeline consisting of several modules, including a foreground prediction, detection of instances, and finally their segmentations. In contrast, our method directly predicts binary masks from image features without using bounding box supervision. In Figure 3 we show examples of predictions obtained by our model for images in the dataset. In the first row, some good examples from the test set are depicted, and in the second row we show some typical mistakes of our network, such as predicting the same leaf twice or merging two leaves in the output of the same time step.

4.4. Multi-class instance segmentation

Next we evaluate our approach in two challenging benchmarks for semantic instance segmentation: Cityscapes and Pascal VOC 2012.

Table 2 shows the results for Cityscapes. Although our accuracy does not meet state of the art methods [19, 12, 6], we find that for the *car* class category we are comparable to

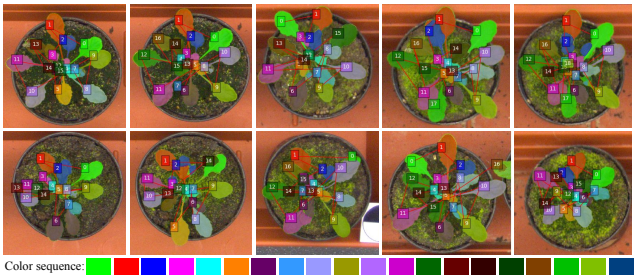

Figure 3. Example predictions for CVPPP dataset.

	R	AP	AP_{50}	AP_{100m}	AP_{50m}
Mask R-CNN [19]		**32.0**	**58.1**	**45.8**	**49.5**
DWT [6]		19.4	35.3	31.4	36.8
Brabandere et al. [12]		17.5	35.9	27.8	31.0
MCG + RCNN [9]		4.6	12.9	7.7	10.3
Ren & Zemel [34]	✓	9.5	18.9	16.8	20.9
Ours	✓	5.8	13.8	10.2	11.7
Mask R-CNN [19]		**49.1**	**71.8**	**69.9**	**67.9**
DWT [6]		31.5	48.5	53.5	50.8
Brabandere et al. [12]		24.4	43.2	40.0	44.0
MCG + RCNN [9]		10.5	26.0	21.2	17.5
Ren & Zemel [34]	✓	27.5	41.9	54.2	46.8
Ours	✓	23.8	42.6	39.3	44.7

Table 2. Results for the Cityscapes dataset. We compare against state-of-the art methods, reporting the average for all categories and separately for cars. The R column indicates whether the approach is recurrent or not.

[34], which is the only other work that has assessed a recurrent method on this dataset. However, their classification scores are provided by a separate module trained for the task of semantic instance segmentation. Our method predicts both the binary mask and the categorical label for each instance. In Figure 4 we show some sample predictions of our model for this dataset. While our model works well for cars and other simpler and dominant objects such as people, it misses most bicycles or motorbikes, which turn out to be the most difficult categories of this dataset. We believe that this behavior can be explained with the fact that [34] relies on a more complex model that adds more supervision with both bounding box and mask penalty terms.

Finally, we train and evaluate our model with the Pascal dataset. While images in this dataset do not have as many object instances as the other two, they are far more complex and different from each other. This poses an additional challenge to our model, which must adapt the scanning pattern to the image layout. Table 3 summarizes the results. In general, our approach achieves competitive performance against state of the art methods. Our approach outperforms early proposal-based methods in [17, 8] by a significant margin across all IoU thresholds. Compared to more recent works [24, 4, 25, 5], our method falls behind

Figure 4. Sample predictions for the Cityscapes dataset.

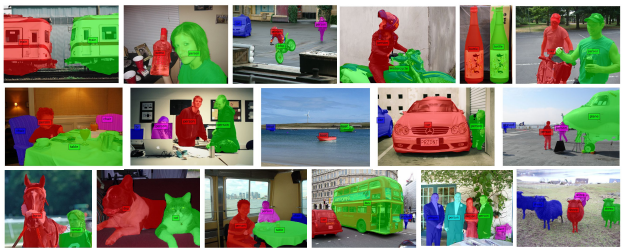

Figure 5. Examples of predicted sequences for Pascal.

for lower thresholds, but remains competitive and even superior in some cases for higher thresholds, thanks to the use of skip-connections in our decoder.

We also compare our method with the recurrent model in [36], which is trained as a class-agnostic instance segmentation algorithm. Since they train and evaluate their method only on the person category in the Pascal VOC 2012 validation set, we report the results for this category separately despite our model is trained for all 20 categories. We outperform their results by a significant margin when using a deeper encoder such as ResNet-101. Our method also performs better when trained using a VGG-16 encoder, which resembles the encoder they use in terms of number of parameters. Figure 5 shows examples of predicted object sequences for Pascal images, which show that our model can deal with occlusions and objects of different scales and shapes. Consistently with the previous two datasets, our model sometimes predicts a single mask for two object instances in the image, which causes ambiguity for the classification branch in cases where the two objects belong to different categories (e.g. the two second man in the last image from the first row is segmented together with a bicycle).

4.5. Class-agnostic instance segmentation

We also evaluate our model as a class-agnostic object proposal algorithm. We compare the masks generated by our method against those generated by object proposal techniques, including those that provide binary masks and not only bounding boxes (namely SCG [3], MCG [3] and COB [28]). For this experiment, we simply ignore output class probabilities and keep the binary mask obtained at each time step as an object proposal. Table 4 shows the Average Recall at different IoU thresholds. Compared to [3, 28], our method achieves higher recall with a smaller number of proposals (i.e. $N = 10$). Although for higher values of N our performance does not increase in the same proportion as [3, 28], our proposals are still competitive, and achieve higher recall for high IoU thresholds. This demonstrates the suitability of using a recurrent approach for semantic instance segmentation, which facilitates the prediction of a small set of objects by design and can still achieve competitive performance.

	R	mean AP				person AP
IoU threshold:		0.5	0.6	0.7	0.8	0.5
SDS [17]		43.8	34.5	21.3	8.7	47.9
Chen et al. [8]		46.3	38.2	27	13.5	48.3
PFN [25]		58.7	51.3	42.5	31.2	48.4
R2-IOS [24]		**66.7**	**58.1**	46.2	-	60.4
Arnab et al. [4]		58.3	52.4	45.4	34.9	58.6
Arnab et al. [5]		61.7	55.5	**48.6**	**39.5**	65.6
MPA [26]		60.3	54.6	45.9	34.3	**67.6**
RIS [36]	✓	-	-	-	-	46.6
RIS + CRF [36]	✓	-	-	-	-	50.1
Ours (VGG16)	✓	46.2	39.8	33.7	26.2	51.7
Ours (R101)	✓	57.6	52.5	43.3	38.1	60.9

Table 3. Performance comparison on Pascal VOC 2012 with state of the art methods for different intersection over union thresholds. The R column indicates whether the approach is recurrent or not.

4.6. Ablation Studies

We perform ablation studies to quantify the effect of each of the components in our network (encoder, skip-connections and number of recurrent layers). Table 5 presents the results of these experiments for Pascal.

First, we compare the performance of different pre-trained image encoders. We find that a deeper encoder yields better performance, with a 23,87% relative increase in performance from VGG-16 to ResNet-101. Further, we analyze the effect of using different skip connection modes (i.e. summation, concatenation and multiplication), as well as removing them completely. While there is little difference between the different skip-connection modes, concatenation has better performance and completely removing skip connections causes a drop of performance of 6.6%, which demonstrates the effectiveness of using skip-connections to obtain accurate segmentation masks.

We also quantify the effect of reducing the number of ConvLSTM layers in the decoder. To remove ConvLSTM layers, we simply truncate the decoder chain and the output of the last ConvLSTM is upsampled to match the image dimensions and becomes the input to the last convolutional layer that outputs the final mask. Removing ConvL-

	AR@50		AR@70		AR@85	
	10	50	10	50	10	50
COB [28]	52.4	**71.9**	36.0	52.8	22.9	35.0
MCG [3]	41.3	63.2	24.4	43.2	12.7	22.6
SCG [3]	41.2	60.8	22.5	38.9	11.2	19.7
Ours	**68.2**	70.7	**55.9**	**57.6**	**41.9**	**42.9**

Table 4. Comparison of the Average Recall at different IoU thresholds with state of the art object proposal techniques.

Encoder	skip	N	AP@50	AP@50 person
VGG16	concat	5	46.5	51.7
R50	concat	5	53.0	53.9
R101	concat	5	**57.6**	**60.9**
R101	sum	5	56.7	57.8
R101	mult	5	56.1	59.2
R101	none	5	53.8	51.3
R101	concat	4	56.0	59.0
R101	concat	3	56.1	59.5
R101	concat	2	54.5	54.0
R101	-	1	53.3	50.6

Table 5. Ablation studies on Pascal VOC 2012 validation set.

STM layers also means removing the corresponding skip connection. Results in table 5 show a decrease in performance as we remove layers from the decoder, which indicates that both the depth of the decoder and the skip connections coming from the encoder contribute to the final result. Notably, keeping the original 5 ConvLSTM layers in the decoder, but removing the skip connections gives a similar performance as using a single ConvLSTM layer without skip-connections (AP of 53.3 against 53.2). This indicates that a deeper recurrent module composed of several ConvLSTM layers can only improve performance if the side outputs from the encoder are used as additional inputs.

4.7. Object Sorting Patterns

We analyze the sorting patterns learned by the network to segment one instance at a time by computing their correlation with arbitrary sorting strategies. We define arbitrary orders, namely: right to left (r2l), bottom to top (b2t) and large to small (l2s) and check whether our network follows any of these patterns when predicting objects in a sequential manner. We take the center of mass of each object to represent its location, and the sum of the pixels that compose it as the measure for its size.

We sort the sequence of predicted masks according to one of the strategies and the obtained permutation indices are compared with the original ones with the Kendall tau correlation metric: $\tau = \frac{P-Q}{N(N-1)/2}$. Given a sequence of masks $x \in \{x_1, ..., x_N\}$ and its permutation $y \in \{y_1, ..., y_N\}$, P is the number of concordant pairs (i.e. pairs that appear in the same order in the two lists) and Q is the number of discordant pairs. $\tau \in [-1, 1]$, where 1 indicates complete correlation, -1 inverse correlation and 0 means there is no correlation between sequences. Table 6 presents the results for this experiment. For simplicity, we do not show the results for the opposite sorting criteria in the table (i.e. left to right, small to large and top to bottom), since their τ value would be the same but with the opposite sign. In the case of Cityscapes and Pascal, we find a strong correlation with the right to left sorting strategy, indicating that most predictions for images in this dataset follow this pattern. However, in the case of Pascal, the value is lower than the one for Cityscapes, and correlation values for bottom to top and large to small sorting strategies are also high. For CVPPP images, we observe that predicted objects correlate the most with a top to bottom scanning strategy.

	Pascal VOC	Cityscapes	CVPPP
r2l	**0.4916**	**0.9499**	-0.3601
b2t	0.2788	-0.0408	**-0.7212**
l2s	0.2739	0.1373	0.0801

Table 6. Kendall tau coefficient τ with arbitrary patterns.

Figure 6. Examples of different sorting patterns. Examples of predicted object sequences for images in Pascal VOC 2012 validation set that highly correlate with the different sorting strategies.

Strong correlation values are coherent with the prediction examples for CVPPP and Cityscapes in Figures 3 and 4, which show a scanning pattern to find objects that is consistent across different images. In the case of Pascal images, this pattern can vary from one image to the other. Figure 6 shows examples of images that present high correlation with each of the three sorting strategies. Interestingly, our model adapts its scanning pattern based on the image contents, choosing to start from one side when objects are next to each other, or starting from the biggest one when the remaining objects are much smaller.

Further, we quantify the number of object pairs that are predicted in each of the arbitrary orders. For a pair of objects o_1 and o_2 that are predicted consecutively, we can say they are sorted in a particular order if their difference in the axis of interest is greater than 15% (e.g. a pair of consecutive objects follows a right to left pattern if the second object is to the left of the first by more than $0.15 \times W$ pixels, being

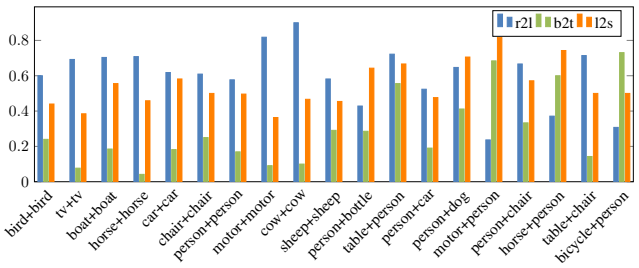

Figure 7. Percentage of consecutive object pairs of different categories that follow a particular sorting pattern.

W the image width). Figure 7 shows the results for object pairs separated by category. For clarity, only pairs of objects that are predicted together at least 20 times are displayed.

We observe a substantial difference between pairs of instances from the same category and pairs of objects of different classes. While same-class pairs seem to be consistently predicted following a horizontal pattern (right to left), pairs of objects from different categories are found following other patterns reflecting the relationships between them. For example, the pairs motorcycle+person, bicycle+person or horse+person are often predicted following the vertical axis, from the bottom to the top of the image, which is coherent with the usual spatial distribution of objects of these categories in Pascal images (horses, bikes and motorbikes are normally ridden by people).

We also check whether the order of the predicted object sequences correlates with the features from the encoder. Since these are the inputs to the recurrent layers in the decoder (which do not change across different time steps), the network must learn to encode the information of the object order in these activations. To test whether this is true, we permute the object sequence based on the activations in each of the convolutional layers in the encoder. Table 7 shows the Kendall tau correlation values of predicted sequences with these activations, before and after training the model. We observe that correlation increases after training the model for our task. The predicted sequences correlate the most with the activations in the last convolutional block in the encoder both for Pascal and Cityscapes. This is a reasonable behavior, since those features are the input to the first ConvLSTM layer in the decoder. In the case of images from the CVPPP dataset, we find that the predicted object sequences inversely correlate with the activations in the last convolutional layer in the encoder, indicating that the sequence starts with the least active object. In Figure 8 we display the most and least active object in the last convolutional layer in the encoder before and after training the model. For Pascal images, we observe a shift of the most active objects from the center of the image to the bottom-right part of the image, while the least active objects are located in the left part of the image. In the case of Cityscapes, objects with the highest activation move from the center to right-most part of the image after training. Regarding CVPPP, we observe that the network learns a specific route to predict leaves which is consistent accross different images. In this case, the most active object correlates with the last object of the route, that tends to be in the center of the image, whereas the least active correlates with the first object of the sequence that tends to be in the top-most part of the image.

	Cityscapes		Pascal VOC		CVPPP	
	before	after	before	after	before	after
f_4	-0.091	-0.105	-0.048	-0.062	-0.132	-0.245
f_3	0.090	0.034	0.014	-0.005	0.179	0.060
f_2	-0.028	0.002	-0.088	-0.125	0.031	-0.070
f_1	0.045	0.056	0.008	0.286	0.124	0.232
f_0	**0.071**	**0.629**	**0.274**	**0.634**	**-0.152**	**-0.414**

Table 7. Correlation with convolutional activations. Kendall tau correlation with activations from the different blocks in the encoder, before and after training.

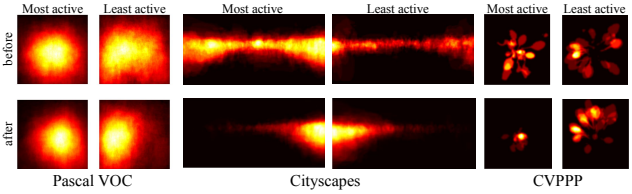

Figure 8. Most and least active objects in last convolutional layer from the encoder before and after training.

5. Conclusion

We have presented a recurrent method for end to end semantic instance segmentation that is able to produce competitive results on three different benchmarks. Thanks to its ability to retain information from previous predictions, our approach is able to produce a variable number of objects for every image. Such ability removes the need for post-processing on the output predictions. We observe that our model is able to learn coherent scanning patterns to follow while generating object predictions that can vary across different images depending on their spatial structure.

6. Acknowledgements

This work was partially supported by the Spanish Ministry of Economy and Competitivity under contracts TIN2012-34557 by the BSCCNS Severo Ochoa program (SEV-2011-00067), and contracts TEC2013-43935-R and TEC2016-75976-R. It has also been supported by grants 2014-SGR-1051 and 2014-SGR-1421 by the Government of Catalonia, and the European Regional Development Fund (ERDF).

References

[1] T. A. Amor, S. D. Reis, D. Campos, H. J. Herrmann, and J. S. Andrade Jr. Persistence in eye movement during visual search. *Scientific reports*, 6:20815, 2016.

[2] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh. Vqa: Visual question answering. In *CVPR*, 2015.

[3] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*, 2014.

[4] A. Arnab and P. H. Torr. Bottom-up instance segmentation using deep higher-order crfs. In *BMVC*, 2016.

[5] A. Arnab and P. H. Torr. Pixelwise instance segmentation with a dynamically instantiated network. In *CVPR*, 2017.

[6] M. Bai and R. Urtasun. Deep watershed transform for instance segmentation. In *CVPR*, 2017.

[7] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML*, 2009.

[8] Y.-T. Chen, X. Liu, and M.-H. Yang. Multi-instance object segmentation with occlusion handling. In *CVPR*, 2015.

[9] 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 *CVPR*, 2016.

[10] J. Dai, K. He, Y. Li, S. Ren, and J. Sun. Instance-sensitive fully convolutional networks. In *ECCV*, 2016.

[11] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, 2016.

[12] B. De Brabandere, D. Neven, and L. Van Gool. Semantic instance segmentation with a discriminative loss function. In *CVPRW*, 2017.

[13] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015.

[14] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge. *IJCV*, 2010.

[15] M. V. Giuffrida, M. Minervini, and S. A. Tsaftaris. Learning to count leaves in rosette plants. 2016.

[16] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011.

[17] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *ECCV*, 2014.

[18] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.

[19] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*, 2017.

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[21] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015.

[22] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. Fully convolutional instance-aware semantic segmentation. In *CVPR*, 2017.

[24] X. Liang, Y. Wei, X. Shen, Z. Jie, J. Feng, L. Lin, and S. Yan. Reversible recursive instance-level object segmentation. In *CVPR*, 2016.

[25] X. Liang, Y. Wei, X. Shen, J. Yang, L. Lin, and S. Yan. Proposal-free network for instance-level object segmentation. *arXiv preprint arXiv:1509.02636*, 2015.

[26] S. Liu, X. Qi, J. Shi, H. Zhang, and J. Jia. Multi-scale patch aggregation (mpa) for simultaneous detection and segmentation. In *CVPR*, 2016.

[27] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[28] K.-K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. Van Gool. Convolutional oriented boundaries. In *ECCV*, 2016.

[29] M. Minervini, A. Fischbach, H. Scharr, and S. A. Tsaftaris. Finely-grained annotated datasets for image-based plant phenotyping. *Pattern recognition letters*, 81:80–89, 2016.

[30] J.-M. Pape and C. Klukas. 3-d histogram-based segmentation and leaf detection for rosette plants. In *ECCVW*, 2014.

[31] E. Park and A. C. Berg. Learning to decompose for object detection and instance segmentation. In *ICLRW*, 2016.

[32] G. Porter, T. Troscianko, and I. D. Gilchrist. Effort during visual search and counting: Insights from pupillometry. *The Quarterly Journal of Experimental Psychology*, 2007.

[33] G. Porter, T. Troscianko, and I. D. Gilchrist. Effort during visual search and counting: Insights from pupillometry. *The Quarterly Journal of Experimental Psychology*, 60(2):211–229, 2007.

[34] M. Ren and R. S. Zemel. End-to-end instance segmentation with recurrent attention. In *CVPR*, 2017.

[35] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[36] B. Romera-Paredes and P. H. S. Torr. Recurrent instance segmentation. In *ECCV*, 2016.

[37] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

[39] H. Scharr, M. Minervini, A. P. French, C. Klukas, D. M. Kramer, X. Liu, I. Luengo, J.-M. Pape, G. Polder, D. Vukadinovic, et al. Leaf segmentation in plant phenotyping: a collation study. *Machine vision and applications*, 2016.

[40] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015.

[41] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional LSTM network: A machine learning approach for precipitation. In *NIPS*, 2015.

[42] X. Yin, X. Liu, J. Chen, and D. M. Kramer. Multi-leaf tracking from fluorescence plant videos. In *ICIP*, 2014.

[43] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015.