# CEDL Homework01 - Introduce a New NN with Memory

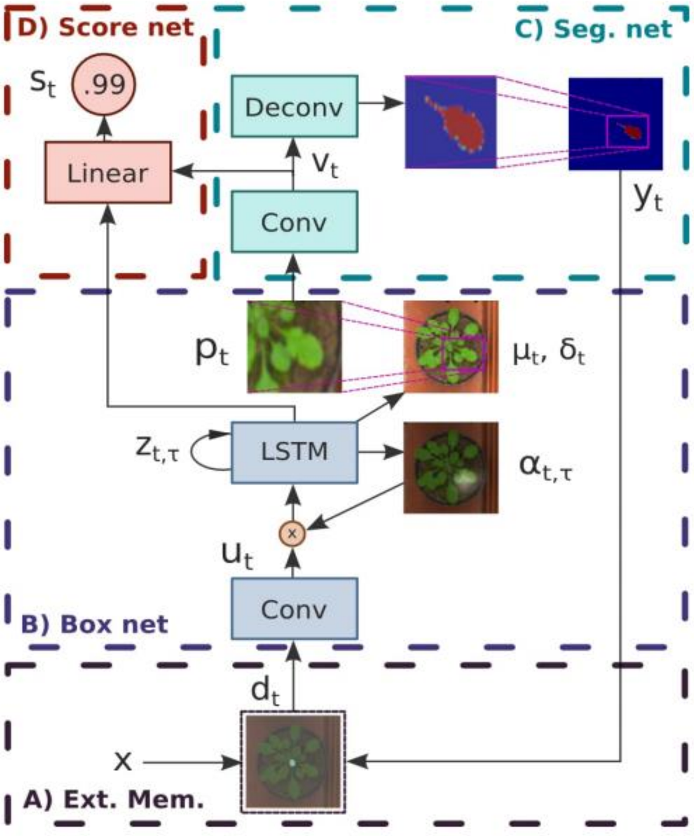
## End-to-End Instance Segmentation and Counting with Recurrent

arXiv:1605.09410v2 [cs.LG] 6 Sep 2016 ***Reference*** : https://arxiv.org/abs/1605.09410

105062536 劉祐欣 105065514 柯子逸

This paper addresses one of the main challenges of instance segmentation, that is, the object occlusion. Because the non-maximal suppression (NMS) may suppress the detection results for a heavily occluded object, too much overlap with foreground objects in cluttered scenes, the proposed method performs dynamic NMS to reason about occlusion. Counting the instances of an object class was considered jointly with instance segmentation.

The proposed model based on a recurrent neural network (RNN) that utilizes visual attention to perform instance segmentation and also performs dynamic NMS, using an object that is already segmented to aid in the discovery of an occluded object later in the sequence.



The model of this paper has four major components, illustrated in the above figure: (A) External memory (B) Box proposal network (C) Segmentation network (D) Scoring network, we will briefly illustrate the model below.

## Part A: External memory

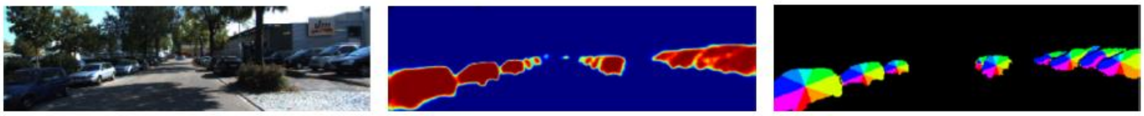
External memory is used to track the state of the already segmented objects, decide where to look next. It provides object boundary details from all previous steps.

The formulation is the cumulative canvas: it stores the full history of segmentation outputs. Canvas has 10 channels in total: the first channel of the canvas keeps adding new pixels from the output of the previous time step, and the other channels store the preprocessed input image.

$$\boldsymbol{c}_t = \begin{cases} \boldsymbol{0}, if \;\; t = 0 \\ \max(\boldsymbol{c}_{t-1}, \boldsymbol{y}_{t-1}) \, , otherwise \end{cases}$$

$$\boldsymbol{d}_t = [\boldsymbol{c}_t, \boldsymbol{x}]$$

$\mathrm{c}_\mathrm{t}$ is the channel of the canvas from the current time step, $\mathrm{y}_{\mathrm{t}-1}$ is the output segmentation sequences from the previous time step. The figure below illustrates the preprocessed input image, which has one channel for foreground segmentation and 8 channels for quantized angle maps.



## Part B: Box network

Part B localizes objects of interest. The CNN outputs a $\mathrm{H}' \times \mathrm{W}' \times \mathrm{L}$ feature map $\mathrm{u}_\mathrm{t}$. A single glimpse is indexed by $\tau$.The proposed method allows the glimpse LSTM to look at different locations by feeding a dimension $\mathrm{L}$ vector each time. In this part, dynamic pooling is employed to extract useful information along spatial dimensions, weighted by $\alpha_\mathrm{t}^{\mathrm{h,w}}$, initialized to be uniform over all locations.

$$\boldsymbol{u}_t = \mathrm{CNN}(\boldsymbol{d}_t)$$

$$\boldsymbol{z}_{t,\tau} = \mathrm{LSTM}(\boldsymbol{z}_{t,\tau-1}, \sum_{h,w} \alpha_{t,\tau}^{h,w} \boldsymbol{u}_t^{h,w,l})$$

$$\alpha_{t,\tau} = \begin{cases} 1/(H' \times W'), if \; \tau = 0 \\ \mathrm{MLP}(\boldsymbol{z}_{t,\tau}), otherwise \end{cases}$$

$\mathrm{LSTM}()$ denotes unrolling the long short-term memory by one time-step with the previous hidden state $\mathrm{z}_{\mathrm{t}-1}$ and current input $\mathrm{u}_\mathrm{t}$, and returning the current hidden state $\mathrm{z}_{\mathrm{t},\tau}$ ; $\mathrm{MLP}()$ denotes passing an input $\mathrm{z}_{\mathrm{t},\tau}$ through a multi-layer perceptron and returning the hidden state $\alpha$.

Then pass $\mathrm{LSTM}()$'s hidden state through a linear layer to obtain predicted box coordinates. The box is parameterized by its normalized center ($\hat{\mathrm{g}}\mathrm{X}$,$\hat{\mathrm{g}}\mathrm{Y}$), and size ($\log\hat{\delta}\mathrm{X}, \log\hat{\delta}\mathrm{Y}$). $\gamma$ is used when re-projecting the patch to the original image size.

$$\left[\hat{g}X, \hat{g}\mathrm{Y}, \log\hat{\delta}X, \log\hat{\delta}Y, \log\sigma X, \log\sigma Y, \gamma\right] = \boldsymbol{w}_b^{\mathsf{T}}\boldsymbol{z}_{t,end} + \boldsymbol{w}_{b0}$$

$$gX = (\hat{g}X + 1)W/2$$

$$gY = (\hat{g}Y + 1)H/2$$

$$\delta X = \hat{\delta}XW$$

$$\delta Y = \hat{\delta}YH$$

Gaussian kernel is used to extract an $\widehat{\mathrm{H}} \times \widehat{\mathrm{W}}$ patch from the $\hat{\mathrm{x}}$, a concatenation of the original image with $\mathrm{d_t}$.

$\mathrm{i, j}$ index the location in the patch of dimension $\widehat{\mathrm{H}} \times \widehat{\mathrm{W}}$, and $\mathrm{a, b}$ index the location in the original image. $\mathrm{F_X}$and $\mathrm{F_Y}$ indicate the contribution of the location $(\mathrm{a, b})$ in the original image towards the location $(\mathrm{i, j})$ in the extracted patch.

$$\mu_X^i = gX + (\delta X + 1) \cdot (i - \frac{\widehat{W}}{2} + 0.5)/\widehat{W}$$

$$\mu_Y^j = gY + (\delta Y + 1) \cdot (j - \frac{\widehat{H}}{2} + 0.5)/\widehat{H}$$

$$F_X^{a,i} = \frac{1}{\sqrt{2\pi}\sigma X}\exp(-\frac{\left(a-\mu^i X\right)^2}{2\sigma^2 X})$$

$$F_Y^{b,j} = \frac{1}{\sqrt{2\pi}\sigma Y}\exp(-\frac{\left(b-\mu^j Y\right)^2}{2\sigma^2 Y})$$

$$\hat{\boldsymbol{x}}_{\boldsymbol{t}} = [\boldsymbol{x}_{\boldsymbol{0}}, \boldsymbol{d}_t]$$

$$\boldsymbol{p}_t = \mathrm{Extract}(\, \hat{\boldsymbol{x}}_{\boldsymbol{t}}, F_Y, F_X) \equiv F_Y^{\mathsf{T}}\hat{\boldsymbol{x}}_{\boldsymbol{t}}F_X$$

### Part C: Segmentation network

The segmentation network utilizes a convolution network to produce a feature map $\mathrm{v_t}$, then input into a deconvolution layers to upsample the low-resolution feature map. Finally produce a patch-level segmentation prediction heat map $\hat{\mathrm{y}}_{\mathrm{t}}$ and re-project this patch prediction to the original image using the transpose of $\mathrm{F_Y}$ and $\mathrm{F_X}$.

$$\boldsymbol{v}_t = \mathrm{CNN}(\boldsymbol{p}_t)$$

$$\hat{\boldsymbol{y}}_{\boldsymbol{t}} = \mathrm{D} - \mathrm{CNN}(\boldsymbol{v}_t)$$

$$\boldsymbol{y_t} = \mathrm{sigmoid}(\gamma \cdot \mathrm{Extract}(\ \boldsymbol{\hat{y}_t}, F_Y^{\intercal}, F_X^{\intercal}) - \beta)$$

### Part D: Scoring network

Part D takes information from the box network $(\mathrm{z_t})$ and the segmentation network ($\mathrm{v_t}$) to produce a score between 0 and 1 to estimate the number of objects in the image and terminate the sequential process.

$$s_t = \mathrm{sigmoid}(\boldsymbol{w}_{zs}^{\intercal}\boldsymbol{z}_{t,end} + \boldsymbol{w}_{vs}^{\intercal}\boldsymbol{v_t} + \boldsymbol{w}_{s0})$$

### Loss functions

The total loss function is a sum of three losses: the segmentation matching IoU loss$\mathcal{L}_\mathrm{y}$; the box IoU loss $\mathcal{L}_\mathrm{b}$; and the score cross-entropy loss $\mathcal{L}_\mathrm{s}$:

$$\mathcal{L}(\mathrm{y, b, s}) = \mathcal{L}_y(y, y^*) + \mathcal{L}_b(b, b^*) + \mathcal{L}_s(s, s^*)$$

## Unique properties of the new NN:

--The next fine-grained part is decided to segment based on the segmentation output from the previous time step and the decision of interest region from multiple glimpses in the box network.

--The external memory addresses the segmentation in order, that is, the counting is addressed from the smaller segmentation to the bigger one.

## Applications of taking advantage of the properties:

--Detection of cancer cells, observation of cells growth, and other kinds of applications in the biomedical field.

--The warehousing management, which automatically check the quantity of the products in storage.

--Traffic flow detection and management.