Jayne.who (post);

<BACK

Textboxes (2016): Image Text Detection

offline

deeplearning | 21 July 2017

Tags | CNN python tensorflow

In this article, we ***present a*** recent article on Deep Learning ***(2016)*** , ***Textboxes: A Fast Text Detector with a single Deep Neural Network***

Let's put it lightly.

Reference link

1. Original textboxes article (arxiv.org)
2. textboxes with tensorflow (I) my code (github)

1. Introduction

Effectiveness of scene text detection

Scene text is a visual object that can be encountered in everyday situations.

Emphasize high utility.

scene text detection difficulty of development

Despite having a task very similar to traditional traditional OCR (traditional optical character recognition)

The scene text detection was experiencing a slow development du

offline

These difficulties include 'too many backgrounds and forms of letters', diversity of letter states due to light conditions' and others.

Existing complex and slow scene text detection model

Conventional scene text detection

It is divided into several stages such as "character / word candidate creation" -> "candidate filtering" -> "grouping"

Tuning during model learning was very difficult,

The speed of the finished model is also very slow, so it's hard to apply it to real-time detection.

Textboxes overcoming limitations

The author of Textboxes is based on the SSD (2015)

Since it is composed of single network, it shows remarkably faster performance than the existing models,

Accuracy is also significantly improved from previous models.

Similarity with SSD

The Textboxes model is very similar in structure to the SSD (sigle shot mulitbox detector)

Only a few hyper-parameters inside the model are said to be adapted to text recognition.

(that is, increasing the aspect ratio of the default box and convolutional kernel (filter) horizontally)

Improved detection performance through feedback of word recognition

In general, the field of *scene text reading* is divided into two tasks: text detection and text recognition.

The Textboxes model performs double text detection.

However, Textboxes can be linked with various text recognition models after detection,

Given a specific given set (lexicon), the feedback of text recognition improves the learning performance of the detection model

The paper says.

offline

2. Related Works

A precedent

The scene text reading field is divided into detection and recognition,

Detection is once again divided into charicter based and word based.

1. charicter based (character unit)

Lukas Neumann, Jiri Matas, real time scene textlocalization and recognition (2012)

2. word based (word unit)

<R-CNN based>

Max Jaderberg, Karen Simonyan, Andrea Vedaldi, Andrew Zisserman's Reading Text in the Wild with Convolutional Neural Networks (2015)

<YOLO based>

Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman's Synthetic Data for Text Localization in Natural Images (2016)

Textboxes are word based models

Textboxes are word based models that detect by word.

Textboxes inspired by SSD

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg

Single Shot Mulitbox Detector (2015)

The model structure of the Textboxes is inspired by the Single Shot Mulitbox detector (SSD).

(In fact, we use the model structure, box matching scheme, and loss function used in SSD almost literally.)

Textboxes (detection) -> Connection of CRNN (recognition) is implemented.

Shi, Baoguang; Bai, Xiang; Yao, Cong's An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recogniti offline

We used Textboxes model and text recognition model through CRNN for end-to-end (detection to recognition) model learning.

The paper concludes that the advantage of textboxes is that end-to-end training is possible by attaching various recognition models to the back of the paper.

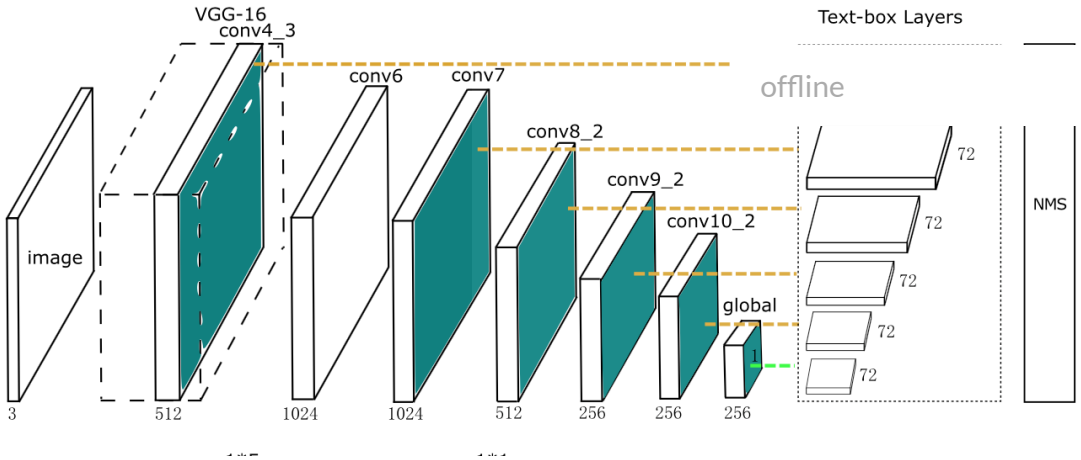
3. Detecting with Textboxes

3-1. Training Phase

Model Structure Architecture

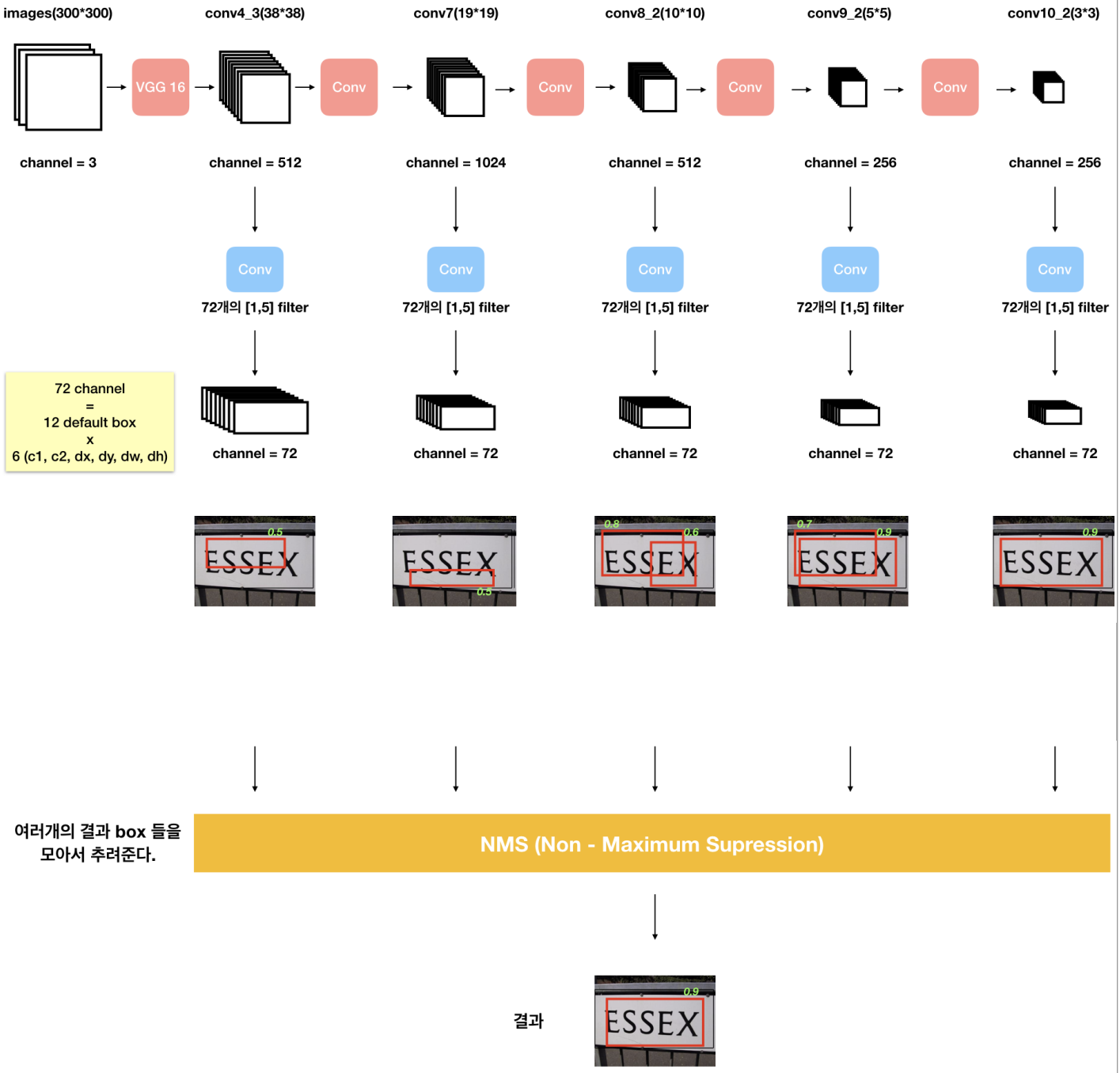
Textboxes are fully-convolutional-networks consisting solely of convolution and pooling.

<Model of the paper => "Picture =" ">



The above is the model in the paper. Based on this model picture, I tried to draw a model picture that I interpreted below.

<Directly Again => "Green =" "Model =" "Picture =" ">



* Training(Learing) 은 Non-maximum supression 이전의 full Convolutional Model 안에서만 일어난다.
* 즉 training 과정중의 output 은 같은 텍스트에 대해서도 여러 box 를 가진다.

The input images are composed of three channels of RGB.

The size of input images was arbitrarily set to 300 * 300. (Input of different size is also possible)

<Term>

The successive convolution layer is called the **"base network"**

The location from which the intermediate feature map is retrieved from the base network is called **"Map Point"** .

The **output** of this model, created by performing a new convolution of feature maps from the Map Point, is called the **"Textbox Layer"** .

In other words, the Textboxes model is a *Textbox Layer* , not an output value for training (loss function) .

The input images create a number of feature maps through the base network,

Extract feature maps from the intermediate map points and perform a convolution using a horizontally long filter ([1,5]).

Create a Texbox Layer (output) with 72 channels.

Intermediate use of feature maps of various sizes is to detect texts of various sizes.

Base network internal dataflow table

The highlighted area in the table is the **Map Point** .

name	channel(out)	stride	padding	kernel	filter conf	batch	height(out)	width(out)	height(out)	width(out)
data	3	0	0	0		1	300	300	700	700
conv1_1	64	1	1	3x3	[3, 3, 3, 64	1	300	300	700	700
conv1_2	64	1	1	3x3	[3, 3, 64, 6	1	300	300	700	700
pool1	64	2	0	2x2		1	150	150	350	350
conv2_1	128	1	1	3x3	[3, 3, 64, 1	1	150	150	350	350
conv2_2	128	1	1	3x3	[3, 3, 128,	1	150	150	350	350
pool2	128	2	0	2x2		1	75	75	175	175
conv3_1	256	1	1	3x3	[3, 3, 128,	1	75	75	175	175
conv3_2	256	1	1	3x3	[3, 3, 256,	1	75	75	175	175
conv3_3	256	1	1	3x3	[3, 3, 256,	1	75	75	175	175
pool3	256	2	0	2x2		1	38	38	88	88
conv4_1	512	1	1	3x3	[3, 3, 256,	1	38	38	88	88
conv4_2	512	1	1	3x3	[3, 3, 512,	1	38	38	88	88
conv4_3	512	1	1	3x3	[3, 3, 512,	1	38	38	88	88
pool4	512	2	0	2x2		1	19	19	44	44
conv5_1	512	1	1	3x3	[3, 3, 512,	1	19	19	44	44
conv5_2	512	1	1	3x3	[3, 3, 512,	1	19	19	44	44
conv5_3	512	1	1	3x3	[3, 3, 512,	1	19	19	44	44
pool5	512	1	1	3x3		1	19	19	44	44
fc6	1024	1	1	3x3	[3, 3, 512,	1	19	19	44	44
fc7	1024	1	0	1x1	[1, 1, 1024,	1	19	19	44	44
conv6_1	256	1	0	1x1	[1, 1, 1024,	1	19	19	44	44
conv6_2	512	2	1	3x3	[3, 3, 256,	1	10	10	22	22
conv7_1	128	1	0	1x1	[1, 1, 512,	1	10	10	22	22
conv7_2	256	2	1	3x3	[3, 3, 128,	1	5	5	12	12
conv8_1	128	1	0	1x1	[1, 1, 256,	1	5	5	12	12
conv8_2	256	2	1	3x3	[3, 3, 128,	1	3	3	6	6

Why the Textbox Layer has 72 channels

What is the number 72?

The Textboxes model pre-sets the Default Box for each pixel in each Map Point, with 12 different aspect ratios. (Same as SSD)

(There are a myriad of default boxes.)

A Default box creates a single predicted box.

The predicted box consists of offset 4 values (dx, dy, dw, dh) and confidence 2 values (c1, c2).

There are 12 default boxes per feature map pixel

Total 12 * 6 (4 + 2) = 72 results are displayed.

offset?

The amount of change in shape from the defualt box that the predicted box was calculated.

A default box creates a single predicted box.

If a default box is located at **(x0, y0)** ** and the width is w0 and the height is h0 **

(All boxes have (x, y, w, h) values on maps scaled from 0 to 1, so you can make comparisons between map locations of different sizes.)

And the offset of the predicted prediction box from the box default **(dx, dy, dw, dh)** if

The location information (x, y, w, h) of the predicted box can be recovered as follows.

$$
\begin{aligned}
x &= x_0 + w_0 \Delta x, \\
y &= y_0 + h_0 \Delta y, \\
w &= w_0 \exp(\Delta w), \\
h &= h_0 \exp(\Delta h).
\end{aligned}
$$

How to create a default box

Before training your model, you should first create tens of thousands of default boxes, 12 each for each pixel at each Map Location.

The x0, y0 values (scaled values between 0 and 1) in each default box are different for each pixel in each Map Location

It is not a value of w0, h0.

Control with **aspect ratio** and **scale** .

The 12 Default Boxes in a Map Location are the same for each pixel.

This is because 12 **aspect ratios** are set in advance. (w: h)

However, for each Map Location, the (w0, h0) of the Default Box are different.

This is because we set the w0, h0 by multiplying the specified aspect ratio by the specific **scale** for each of the six map locations .

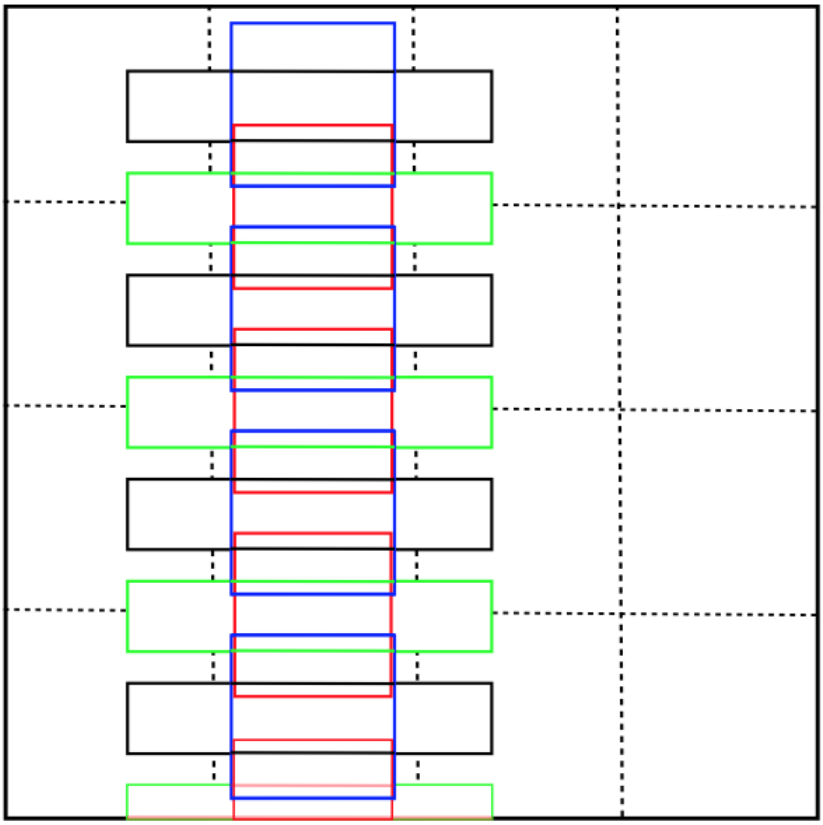
1) Aspect ratio configuration

Because the text is basically long, the aspect ratio of the Default box is also long.

In the paper, we set six aspect ratios (1, 2, 3, 5, 7, 10) and apply it to the center of the pixel once,

Define a total of 12 pixels by default box.

The reason why I dropped six down slightly without putting all the boxes in the center is for fine detection.



In the figure, blue (center, aspect ratio 1), black (center, aspect ratio 5), red (down, aspect ratio 1), and green (down, aspect ratio 5).

2) Scale configuration

Scale is not mentioned in this paper, but it is described in the SSD paper and is actually used in Textboxes implementations.

Each Scale in the six Map locations forms an isometric sequence.

Loss Function

Prepare all the default boxes, prepare the network model

If you put an image in this model and run it

Tens of thousands of default boxes yield values for each (dx, dy, dw, dh, c1, c2).

These values are used to construct a loss function,

We will go through each iteration in a way that minimizes this loss function through the Optimizer,

This process is repeated a myriad of times.

The formula for loss function is shown below.

Briefly,

```
total loss = confidnece loss + location loss
```

is.

For the confidence loss, use the softmax loss function.

For location loss, use the smooth_L1 loss function.

(The formula below is from the SSD paper.)

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \quad (1)$$

where N is the number of matched default boxes. If $N = 0$, wet set the loss to 0. The localization loss is a Smooth L1 loss [6] between the predicted box (l) and the ground truth box (g) parameters. Similar to Faster R-CNN [2], we regress to offsets for the center (cx, cy) of the default bounding box (d) and for its width (w) and height (h).

$$\begin{gathered} L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^{k} \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m) \\ \hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h \\ \hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right) \end{gathered} \quad (2)$$

The confidence loss is the softmax loss over multiple classes confidences (c).

$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} x_{ij}^p log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)} \quad (3)$$

and the weight term α is set to 1 by cross validation.

But the important thing is that we do not use the prediction values from all the default boxes.

** Classify default boxes as positive and negative. **

Jaccard Overlap & Hard Negative Mining

We compare the Jaccard Overlap of **all the default boxes** we made in advance with **all the ground truth boxes in** one image

Jaccard Overlap> 0.5 dafualt boxes were **positive**

The Jaccard Over <0.5 or less Box default **negative** to specify that.

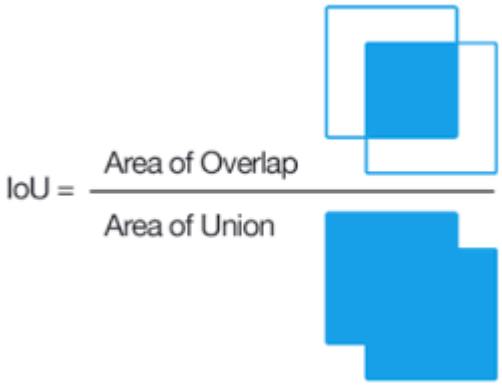
The number of negatives is significantly greater than the number of positive

Negative parts are ignored so that the number of negative parts is three times the number of positive parts.

This process is called Hard Negative Mining.

Jaccard Overlap Thesis (SSD)

Matching strategy During training we need to determine which default boxes correspond to a ground truth detection and train the network accordingly. For each ground truth box we are selecting from default boxes that vary over location, aspect ratio, and scale. We begin by matching each ground truth box to the default box with the best jaccard overlap (as in MultiBox [7]). Unlike MultiBox, we then match default boxes to any ground truth with jaccard overlap higher than a threshold (0.5). This simplifies the learning problem, allowing the network to predict high scores for multiple overlapping default boxes rather than requiring it to pick only the one with maximum overlap.



Hard Negative Mining Thesis (SSD)

Hard negative mining After the matching step, most of the default boxes are negatives, especially when the number of possible default boxes is large. This introduces a significant imbalance between the positive and negative training examples. Instead of using all the negative examples, we sort them using the highest confidence loss for each default box and pick the top ones so that the ratio between the negatives and positives is at most 3:1. We found that this leads to faster optimization and a more stable training.

At the end of the classification,

** location loss only uses positive boxes **

** confidence loss uses c1 for positive boxes and c2 for negative boxes. **

The total loss is (location loss + confidence loss) / N.

N is the number of boxes detected as positive.

This value can be reduced through the Optimizer to learn the model.

3-2. Validation / Test Phase

Non Maximum Supression (NMS)

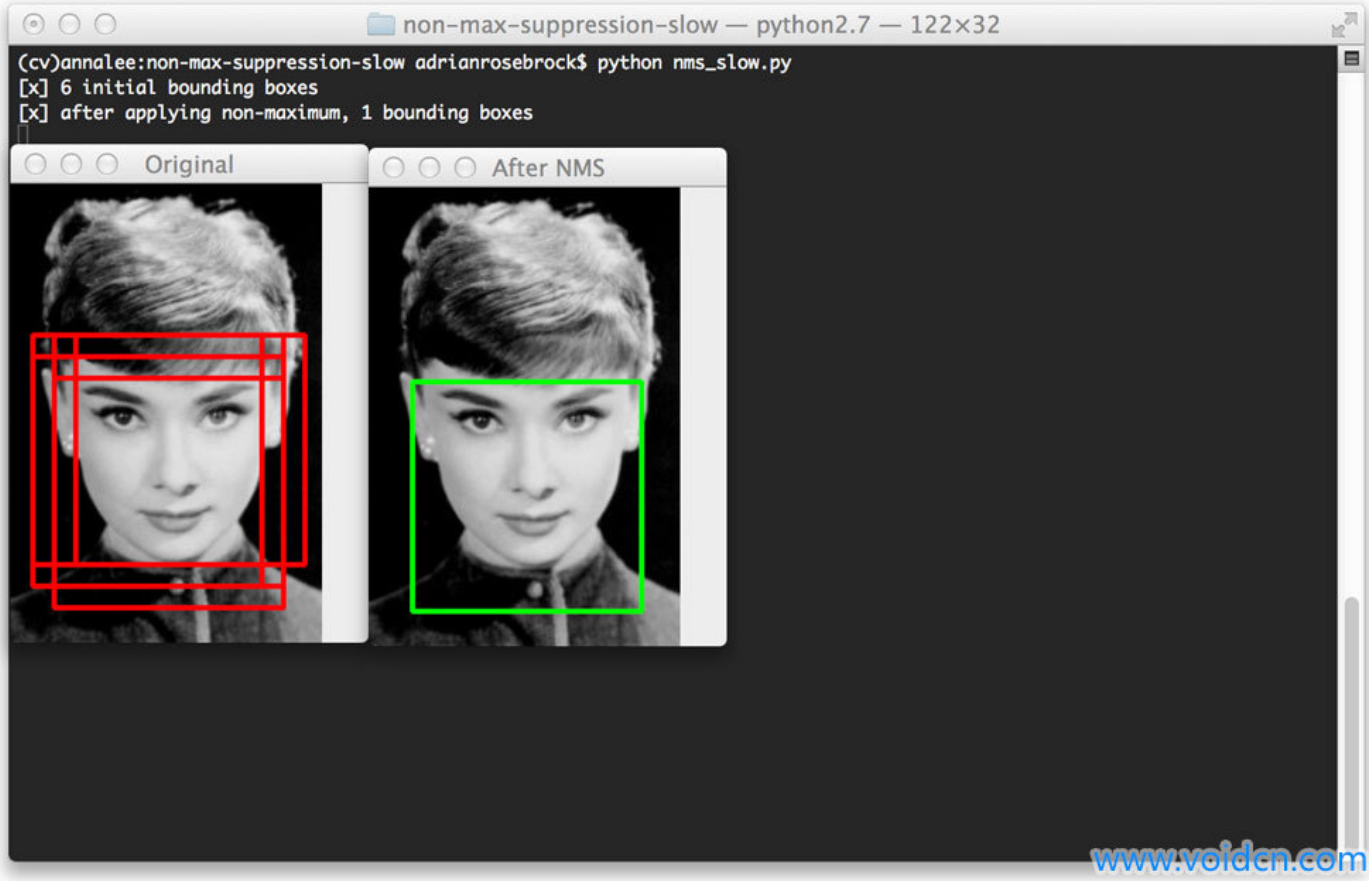
When you put the input image in the training finished model, the Textbox Layer comes out as output.

However, Detecting is not completed by the Textbox Layer itself.

You have to choose which of the many values is a real Predicted Box.

NMS is used in this part.

References: Non-Maximum Suppression for Object Detection in Python



When you perform the NMS, you will recall several candidate predicted boxes into a single most predicted box as above.

Model Accuracy Measurement

It is very important how you measure Accuracy to see the performance of the model you have learned.

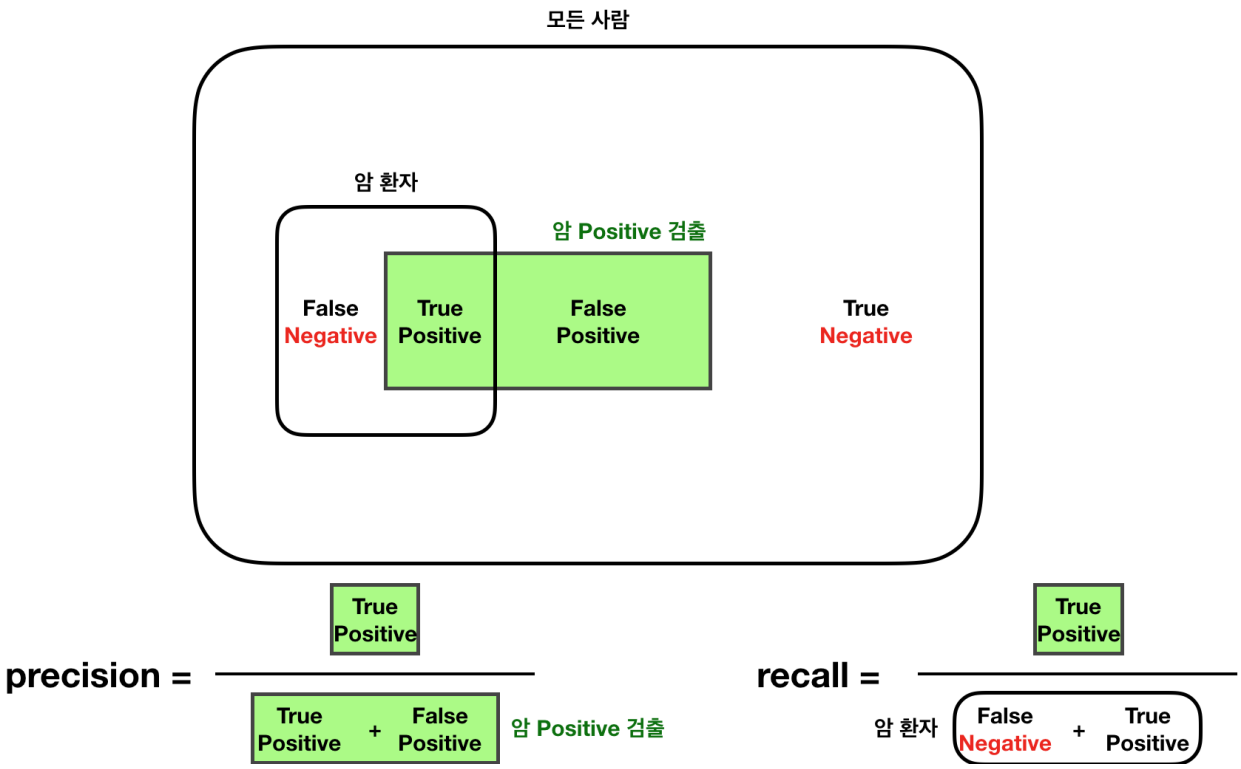
F-Mesure is used for Accuracy measurement .

F-Mesure

F-measure is very often used and important for measuring accuracy.

Two of the most important indicators for evaluating accuracy in machine learning are **precision** and **recall** .

I tried to draw a picture to help you understand the accuracy and recall rate.



Accuracy (precision) we **would expect that Positive** one **will fit** the ratio.

Recall (recall) the **actual ground truth** of **it matched** the rate.

Both of these indicators are important.

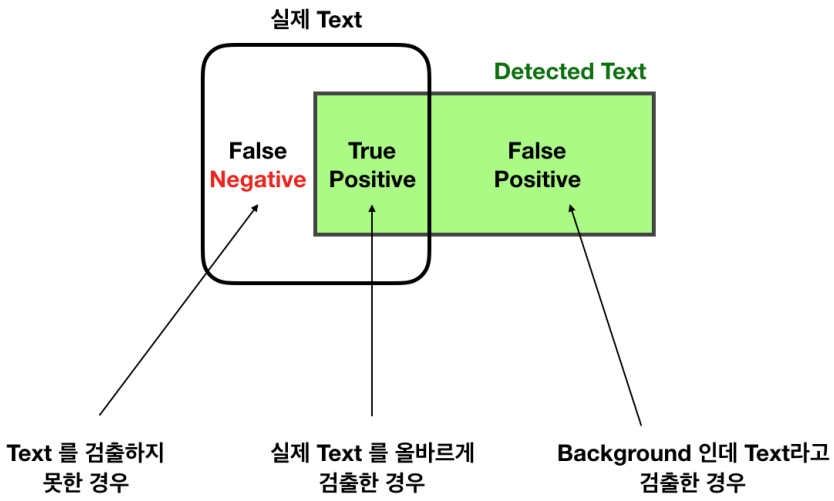
F-Mesure is a precision value that mixes the proportion of precision and recall appropriately.

<F-Mesure formula>

$$F_1 = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

Harmonization average of precision and recall

To obtain the accuracy of the Textboxes model using F-Mesure:



The paper also describes the performance of the actual Textboxes detection model that they trained for various data sets.

P (Precision) / R (Recall) / F (F-Measure)

We calculated the accuracy by three indicators and boasted it.

We have also compared it with various text detection models.

Table 1: Text localization on ICDAR 2011 and ICDAR 2013. P, R and F refer to precision, recall and F-measure respectively. FCRNall+filts reported a time consumption of 1.27 seconds excluding its regression step so we assume it takes more than 1.27 seconds.

Datasets	ICDAR 2011						ICDAR 2013						Time/s
Evaluation protocol	IC13 Eval			DetEval			IC13 Eval			DetEval			
Methods	P	R	F	P	R	F	P	R	F	P	R	F	
Jaderberg (Jaderberg et al. 2016)	–	–	–	–	–	–	–	–	–	–	–	–	7.3
MSERs-CNN (Huang, Qiao, and Tang 2014)	0.88	0.71	0.78	–	–	–	–	–	–	–	–	–	–
MMser (Zamberletti, Noce, and Gallo 2014)	–	–	–	–	–	–	0.86	0.70	0.77	–	–	–	0.75
TextFlow (Tian et al. 2015)	0.86	0.76	0.81	–	–	–	0.85	0.76	0.80	–	–	–	1.4
FCRNall+filts (Gupta, Vedaldi, and Zisserman 2016)	–	–	–	**0.92**	0.75	0.82	–	–	–	**0.92**	0.76	0.83	>1.27
FCN (Zhang et al. 2016)	–	–	–	–	–	–	0.88	0.78	0.83	–	–	–	2.1
SSD (Liu et al. 2016)	–	–	–	–	–	–	0.80	0.60	0.68	0.80	0.60	0.69	0.1
Fast TextBoxes	0.86	0.74	0.80	0.88	0.74	0.80	0.86	0.74	0.80	0.88	0.74	0.81	**0.09**
TextBoxes	**0.88**	**0.82**	**0.85**	0.89	**0.82**	**0.86**	**0.88**	**0.83**	**0.85**	0.89	**0.83**	**0.86**	0.73

Finish the paper review

In fact, Textboxes have the advantage of being able to connect with a Recognition model such as C_RNN to learn at once.

If you have a chance, I will try to review that connected model.

The actual implementation of Textboxes is implemented in python tensorflow and will be posted soon.

If you are interested

I would appreciate it if you look around the blog.

If you have any questions or mistakes, please feel free to comment.