A Deep Learning Method for Mathematical Formulas Detection in PDF Documents

Nghia Vo Trong
Faculty of Information Technology
University of Science, VNU−HCM
Ho Chi Minh City, Vietnam
20120536@student.hcmus.edu.vn

Van-Loc Nguyen
Faculty of Information Technology
University of Science, VNU−HCM
Ho Chi Minh City, Vietnam
20120131@student.hcmus.edu.vn
ORCID: 0000-0001-9351-3750

Minh-Tam Nguyen Kieu
Faculty of Information Technology
University of Science, VNU−HCM
Ho Chi Minh City, Vietnam
20120572@student.hcmus.edu.vn

Dang Nguyen Hai
Faculty of Information Technology
University of Science, VNU−HCM
Ho Chi Minh City, Vietnam
nhdang@selab.hcmus.edu.vn

***Abstract*—Write the abstract here.**
***Index Terms*—**

I. Introduction

II. Related Works

III. Method

We use the Faster R-CNN model with ResNet50 as the backbone for our model. The Faster R-CNN is a real-time object detection model, which consists of 2 modules. The first module of the Faster R-CNN model is a deep fully convolutional network that proposes regions. The second module is a detector that uses proposed regions from the first one [2]. This is a single, unified network for object detection. By using the recently popular terminology of neural networks as the 'attention' mechanisms, the Region Proposal Networks (RPN) tells the Fast R-CNN where to look.

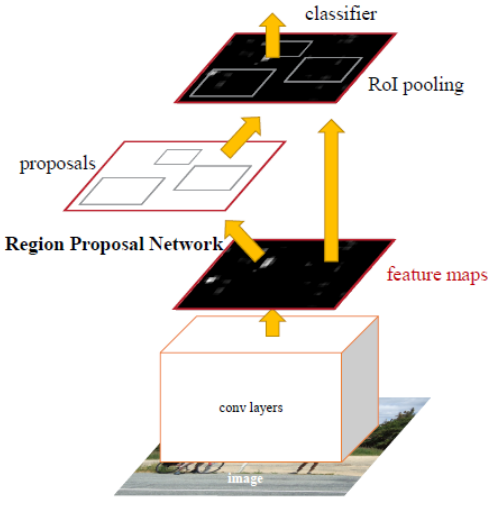

Fig. 1. Faster R-CNN is a single, unified network for object detection. The RPN module plays the role of the 'attention' of this network.

A. Region Proposal Networks

An RPN takes an image as input and outputs a set of rectangular object proposals each with an objectness score, which measures membership to a set of object classes. This process is modeled with a fully convolutional network. The structure of the RPN is shown in Figure 2, and figure 3 shows some examples of object detection using RPN proposals, on the PASCAL VOC 2007 test. The method introduced in [2] detects objects in a wide range of scales and aspect ratios.

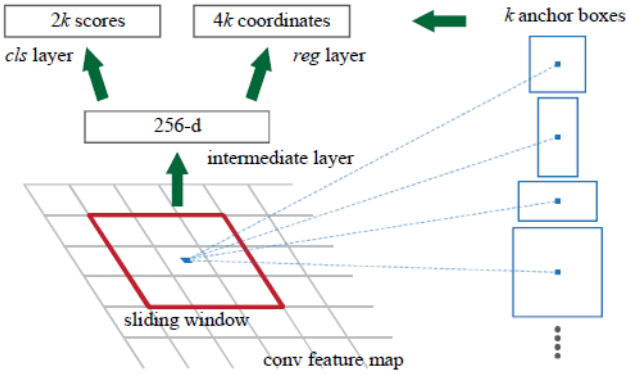

Fig. 2. Region Proposal Network (RPN)

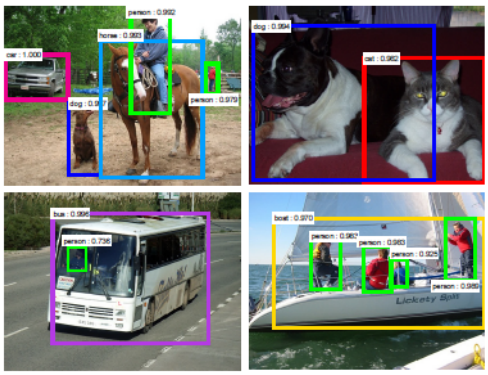

Fig. 3. Examples of object detection using RPN proposals

Loss function: From [2], the loss function of the RPN is:

$$L\left(\{p_i\},\{t_i\}\right)=\frac{1}{N_{cls}}\sum_i L_{cls}\left(p_i,p_i^*\right)$$

$$+\lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}\left(t_i,t_i^*\right).$$

B. Sharing Features for RPN and Fast R-CNN

In the Faster R-CNN model, they use a 4-step training algorithm to learn shared features via alternating optimization. In the first step, the RPN is trained end-to-end by back-propagation and stochastic gradient descent (SGD). In the second step, they train a separate detection network by Fast R-CNN. In the third step, they use the detector network to initialize RPN training, and they let the two networks share convolutional layers. Finally, they keep the shared convolutional layers fixed, they fine-tune the unique layers of Fast R-CNN.

C. ResNet50

ResNet50 is a variant of the ResNet model, consisting of 48 convolution layers with 1 MaxPool and 1 Average Pool layer. It has 3.8×10^9 floating points operations. The ResNet50 is a widespread ResNet model.
Figure 4 shows the architecture of the ResNet50.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3,64\\3\times3,64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3,64\\3\times3,64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,64\\3\times3,64\\1\times1,256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,64\\3\times3,64\\1\times1,256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,64\\3\times3,64\\1\times1,256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3,128\\3\times3,128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3,128\\3\times3,128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1,128\\3\times3,128\\1\times1,512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1,128\\3\times3,128\\1\times1,512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1,128\\3\times3,128\\1\times1,512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3,256\\3\times3,256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3,256\\3\times3,256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1,256\\3\times3,256\\1\times1,1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1,256\\3\times3,256\\1\times1,1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1,256\\3\times3,256\\1\times1,1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3,512\\3\times3,512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3,512\\3\times3,512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,512\\3\times3,512\\1\times1,2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,512\\3\times3,512\\1\times1,2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,512\\3\times3,512\\1\times1,2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Fig. 4. ResNet50 architecture

We can see that [3], in a ResNet50 architecture, there are:

- A convolution with a kernel size of 7×7 and 64 different kernels all with a stride of size 2 there is 1 layer.
- Next there is a max pool with also a stride size of 2.
- In the next convolution there is a 1×1, 64 kernel following this a 3×3, 64 kernel and at last a 1×1, 256 kernel. These three layers are repeated in total 3 times so there are 9 layers in this step.
- Next we see a kernel of 1×1, 128 after that a kernel of 3×3, 128 and at last a kernel of 1×1, 512 this step was repeated 4 times so there are 12 layers in this step.
- After that there is a kernel of 1×1, 256 and two more kernels with 3×3, 256 and 1×1, 1024 and this is repeated 6 times there are a total of 18 layers.
- And then again a 1×1, 512 kernel with two more of 3×3, 512 and 1×1, 2048 and this was repeated 3 times there are a total of 9 layers.
- After that we do an average pool and end it with a fully connected layer containing 1000 nodes and at the end a softmax function so this gives us 1 layer.

There are $1+9+12+18+9+1=50$ layers in total.

IV. Experiments

In this section, we will describe the implementation of our mathematical formula detection system and dataset in detail.

A. Dataset

Our data is from the IBEM dataset. This originally comprises 600 documents, with 8273 pages in total. Those documents are parsed from mathematical papers, then each page is annotated with a bounding box of 2 types: isolated and embedded. The dataset is then split into various sets for ICDAR 2021 Competition on Mathematical Formula Detection, including Training, Test, and Validation sets.

Training

- Tr00: 4082 pages.
- Tr01: 760 pages.
- Tr10: 329 pages.

Test

- Ts00: 736 pages.
- Ts01: 380 pages.
- Ts10: 699 pages.
- Ts11: 329 pages.

Validation

- Va00: 577 pages.
- Va01: 380 pages.

Our experiment uses Tr01, Tr10, Ts01 for training, Va01 for validation, and Ts11 for testing with 2178 pages in total ($\sim 26.33\%$ of the original dataset), and an approximate ratio of $4.47 : 1.16 : 1$. The reason for this small subset is for the purpose of evaluating the ability of the model on small subsets, and the performance it gives (F1-score) through time (minutes).

B. Implementation Details

Our baseline model is Faster R-CNN with ResNet50 as the backbone. We have trained on Kaggle with a 4-core CPU, 12GB RAM, and a NVIDIA Tesla P100 GPU [1]. The images are resized to 1447×2048 with the same ratio. The size of the region crops from the image is 1200×1120 to fit the limitation of the machine. They are also flipped and padded for data augmentation. For the feature aggregation, we use FPN (2-6). The loss function for the classifier is Cross-Entropy Loss and for the bouding box is L1 Loss. Test images are resized to 1583×2048 due to the distribution of the test dataset, flip augmentation is also applied. For post-processing, Non-Maximum Suppression (NMS) with 0.5 IoU threshold to remove redundant boxes. All models are trained based on the MMDetection toolbox and config given by Yuxiang Zhong. The optimizer for this baseline is Stochastic Gradient Descent (SGD) with a learning rate of 0.02.

C. Remarks

We have tested on 3 configs: Faster R-CNN with schedule 1x (12 epochs), Dynamic R-CNN with schedule 1x (12 epochs) to check if it is better than the faster one and Faster R-CNN with schedule 2x (24 epochs) to check if the model is underfitting with low epochs.
The results are given in the figures below.

[1] https://www.kaggle.com/docs/notebooks

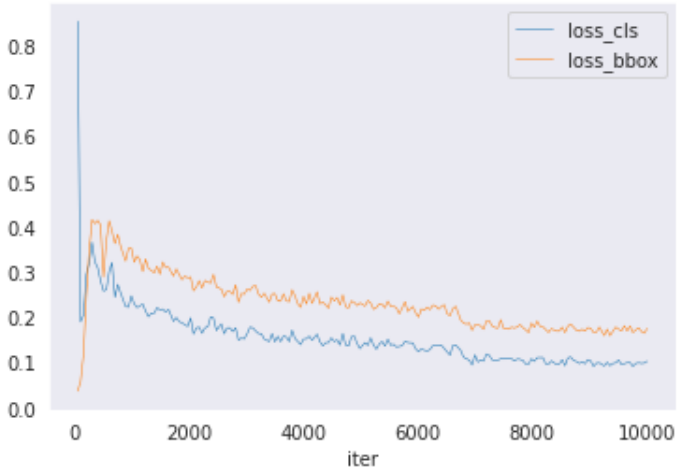

Fig. 5. Faster R-CNN with schedule 1x

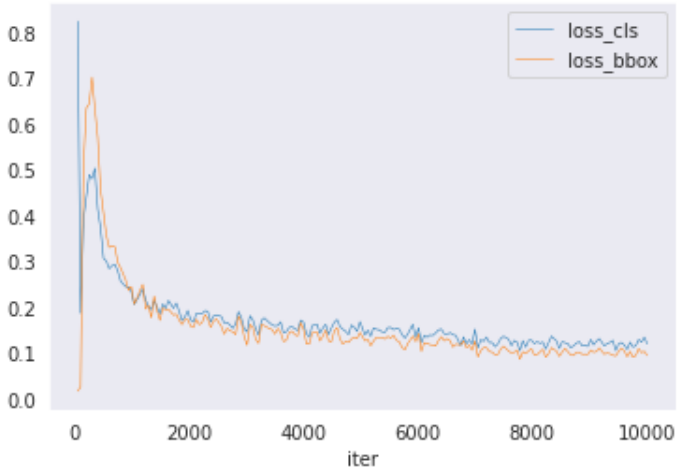

Fig. 6. Dynamic R-CNN with schedule 1x

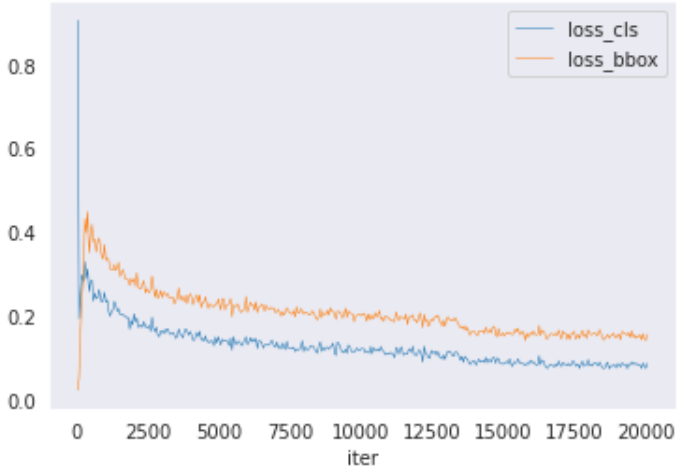

Fig. 7. Faster R-CNN with schedule 2x

The F1-score gained from the model is as follow.

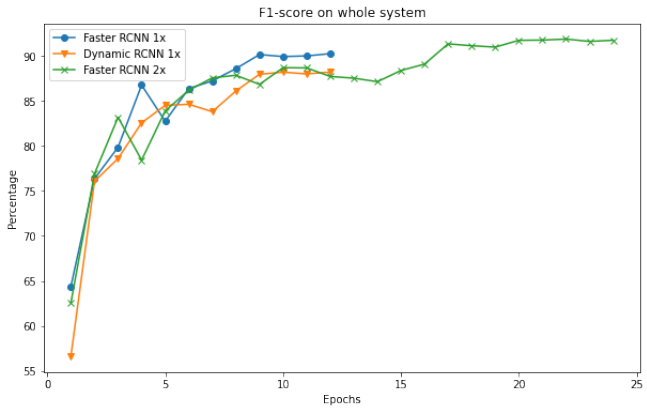

Fig. 8. F1-score on whole system

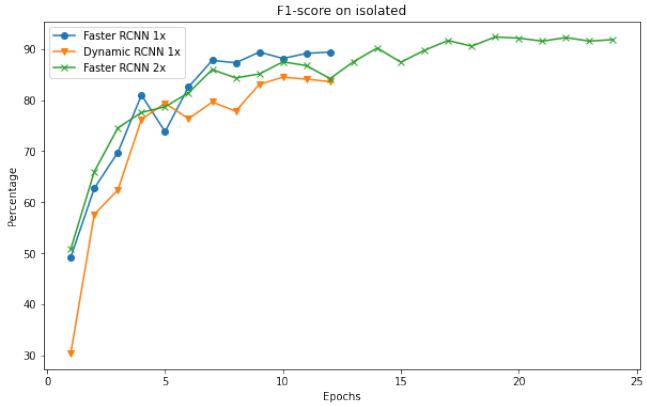

Fig. 9. F1-score with isolated bounding box

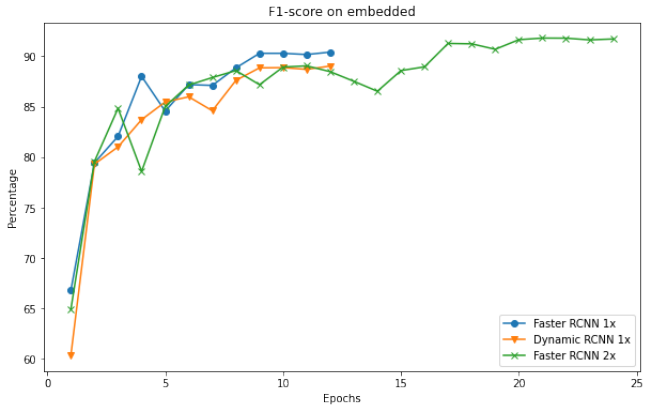

Fig. 10. F1-score with embedded bounding box

It can be seen from the graphs that on the whole system, with the same schedule 1x, the F1-scores given by the Faster R-CNN model are higher than the one by Dynamic R-CNN if we use the same number of epochs, except in the case of 5 epochs. The difference gets higher when we increase the number of epochs. Compared to the scores by Faster R-CNN with schedule 2x (24 epochs), although it gives a lower percentage when trained with a small number of epochs, the score becomes increasing to around 90%. Moreover, on the isolated bounding box, the Faster R-CNN model shows its benefit when compared with the number of Dynamic R-CNN, the F1-score of Faster R-CNN is nearly 90% while the one of Dynamic R-CNN reaches about 80% when they are both trained with 12 epochs. Considering the Faster R-CNN with schedule 2x, it gives the same F1-score with Dynamic R-CNN 1x at the point of 12 epochs, however, the score is about 90% at the point of 24 epochs. Besides that, it can be inferred from the figures of the embedded bounding box that with the same number of epochs (12 epochs), the Faster R-CNN model always provides better results than the Dynamic R-CNN, in spite of the fact that the difference is not large. When we increase the number of epochs to 24, we can observe that the F1-score of Faster R-CNN can reach the milestone of nearly 95%.

From the result given above, we can conclude that the Faster R-CNN model gives better F1-score than the Dynamic R-CNN model.

V. Future Works

VI. Conclusion

Acknowledgment

References

[1] Zhong, Y., Qi, X., Li, S., Gu, D., Chen, Y., Ning, P., and Xiao, R. (2021). 1st Place Solution for ICDAR 2021 Competition on Mathematical Formula Detection. Available: http://arxiv.org/abs/2107.05534.

[2] Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), 1137–1149. Available: https://doi.org/10.1109/TPAMI.2016.2577031

[3] Kaushik, A. (n.d.). Understanding ResNet50 architecture. https://iq.opengenus.org/resnet50-architecture/