Scene text detection and Its application on Android

Xu Zhang

***Abstract*—Text detection using deep learning can deal with scene text that has much more complex surroundings and is in various shapes and orientations. However, deep learning models usually has very large model and takes a long time to inference. In this project, I proposed a modified model that combines EAST and residual bottleneck layers in MobileNets. The experiments show that residual bottleneck layers can greatly reduce the parameters and computation without losing too much performance. Finally, the model is implemented on Android. Note: My code is heavily relied on the work from https://github.com/argman/EAST. Please see my readme.md file for more information on my own work.**

I. Introduction

SCENE text detection is a common challenge in many computer vision competitions like COCO, ICDAR etc. Unlike traditional OCR(Optical Character Recognition), scene text has much more complex surroundings. The text are located randomly in the image and can take various shapes and orientation. The rapid development in deep learning provide a solution for this challenge.

Although text detection is also a category of object detection, commonly used object detection models like Faster-RCNN [1] etc. do not perform well on this challenge. Most of these models comprise a Region Proposal Network(RPN) that makes proposals to generate Region of Interest(ROI) for further classification and bounding box regression. Because the text have the property of arbitrary length and the proposals are usually of limited sizes and shapes at each pixel on the feature map, the RPN cannot make good proposals to capture all the text.

The Connectionist Text Proposal Network(CTPN) [2] handles this problem by making sequential fixed-width vertical anchors as proposals. The proposals are connected by a CNN and followed with a RNN, resulting an end to end trainable model. This model exploit the sequential nature of text. EAST [3] is another architecture that provides a light-weight solution for text detection. The author claims that text detection requires both the high level features to determine the existence of text and the low level features to predict the geometry enclosing a small word region. Thus, this model merged four levels of features to predict the score map (existence) and geometry box. Without RPN, EAST is much more efficient while maintaining good performance.

Though being effective, the original EAST model is still too big to be implemented on mobile devices. Recently, MobileNets [4], as well as its better version, MobileNetV2 [5], provide a solution to implement these deep models to mobile devices.

In this project, the original EAST backbone is replaced with MobileNetsV2, which uses residual bottleneck layers. Then, EAST's feature fusion part is also replaced with residual bottleneck layers. The performance of each model is compared and the influence of regular convolutional layers and residual bottleneck layers is discussed.

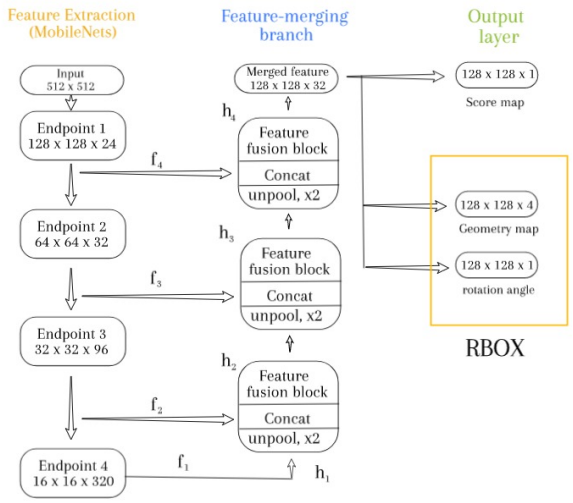

Fig. 1: Mobile-EAST model architecture.

II. Model Architecture

The model is majorly based on EAST proposed by Zhou etc [1]. Unlike most object detection algorithm, the EAST model does not have the complex proposing system, like Regional proposal network, Region of interesting pooling. Thus, it provides another simple but more efficient solution to the text detection task. In the meanwhile, MobileNets [2], a light-weighted model that has comparable results with many popular large scale models like VGG, ResNet makes it possible to even further boost the performance. Thus, in my model, I tried to combine these two together to see if the model can be further improved.

The overview of the architecture is shown in Fig.1. It is majorly based on EAST proposed by Zhou [3]. It can be decomposed into three parts: feature extraction, feature fusion and output layer. The feature extraction part is MobileNetV2 models pretrained on ImageNet data. Four endpoints of MobileNets are extracted, denoted as fi. Appendix shows the architecture of MobileNetV2, the output of layer 8, 6, 4, 3 are chosen as the four endpoints. Each of them has size

of 1/32, 1/16, 1/8, 1/4 of the input image with 320, 96, 32, 24 channels, respectively. Then these 4 feature maps are concatenated together in a U-shape manner. The merge process can be expressed as follows:

Each feature f(i) is unpooled and concatenated with previous feature map f(i+1). A feature fusion block is applied on the concatenated features and then render the output to the next block. In the feature fusion block, two types of strategies are considered. First one is the original method in the EAST paper, which uses a regular conv 1x1 bottleneck to cut down the number of channels and reduces computation, followed by a conv 3x3 that fuses the information to finally produce the output of this merging stage.

The second method is using a stride 1 residual bottleneck layer as the feature fusion block, which is build upon Depthwise Separable Convolutional layer [4]. Depthwise Separable Convolutional layer factorizes the original convolution process into two parts: depthwise convolution that operates on each channel of input and pointwise convolution that generates a linear combination of the output of depthwise layer. On top of Depthwise Separable Convolution, residual bottleneck layer [5] takes a low dimensional input, expand it into high dimension and apply depthwise separable convolutions. In the final step, the tensor is compressed back to the low dimension. The combination of this inverted residual structure and depthwise separable convolutions enables the layer to preserve information while saving a lot of computations. The detail of these two blocks are shown in Fig.2.

The final output layers are several conv 1x1 operations to project 32 channels of concatenated feature maps into 1 channel of score map and a 5 channel geometry map. The geometry map used RBOX representation of the bounding box. It is comprised of 4 distances from the current pixel location of geometry map to the top, right, bottom, left boundaries of the rectangle respectively. The last channel of the geometry map is the rotation angle of the rectangle.

III. Label generation

The data I used for training is from the ICDAR2015 Incidental text challenge. It contains 1000 images with all the text in images being annotated by 8 coordinates. These 8 coordinates represents the polygon that covers the text area.

A. Score map generation

The score map of an image is filled with 0s and 1s, where 1 indicates positive area that contains text. One way to generate the score map is fill the area within the 8 coordinates with 1s. However, because it is hard to tell the real boundary of the text area, a shrunk version of the original one is used.

B. Geometry map generation

The original EAST model proposed two kinds of geometry, RBOX and QUAD. In this project, only RBOX is used. RBOX represents the rectangle by the four distances of the current pixel to the four boundaries and the rotation angle. To obtain the RBOX from 8 polygon coordinates, I first generated a rotated rectangle that covers the text region with minimal area. Then for each pixel within the rectangle, we calculated the 4 distances to the 4 boundaries of the text box. Along with the rotation angle, the geometry map labels are generated.

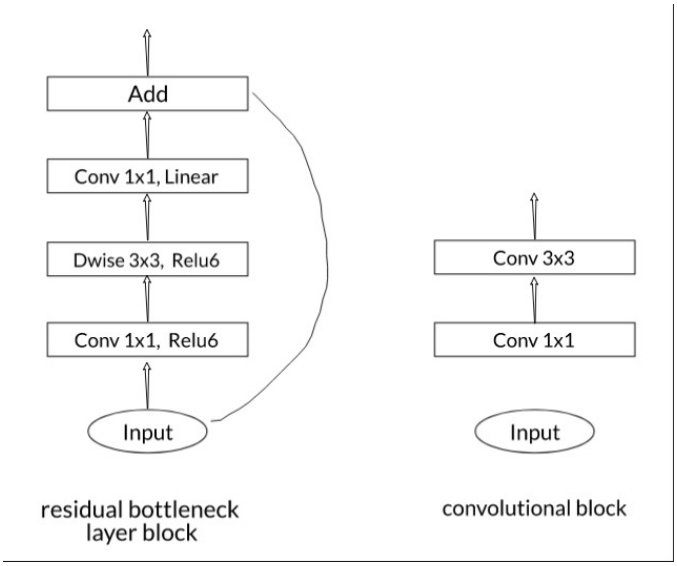

Fig. 2: Different blocks used in feature fusion.

IV. Loss functions

The total loss is the weighted sum of the loss of score map and the loss of geometry map:

$$L = Ls + \lambda g Lg$$

Where λg is the weight to balance the loss of score map and the loss of geometry. I use λg = 0.01 in the training.

A. Loss of score map

Unlike the original paper, which adopt the balanced cross-entropy as the loss, my project used diced score coefficient(DSC) which is a measure of overlap widely used to assess segmentation performance [6]. It is widely used because it can deal with situations of unbalanced data and in object detection most pixels are background which is highly unbalanced. The 2-class variant of dice loss, can be expressed as

$$DL_2 = 1 - \frac{\sum_{n=1}^{N} p_n r_n + \epsilon}{\sum_{n=1}^{N} p_n + r_n + \epsilon} - \frac{\sum_{n=1}^{N} (1-p_n)(1-r_n) + \epsilon}{\sum_{n=1}^{N} 2 - p_n - r_n + \epsilon}$$

where R is the reference foreground segmentation (gold standard) with values r_n, and P is the predicted probabilistic map for the foreground label over N image elements p_n, with the background class probability being $1 - P$. The ϵ term is used here to ensure the loss function stability by avoiding the numerical issue of dividing by 0,

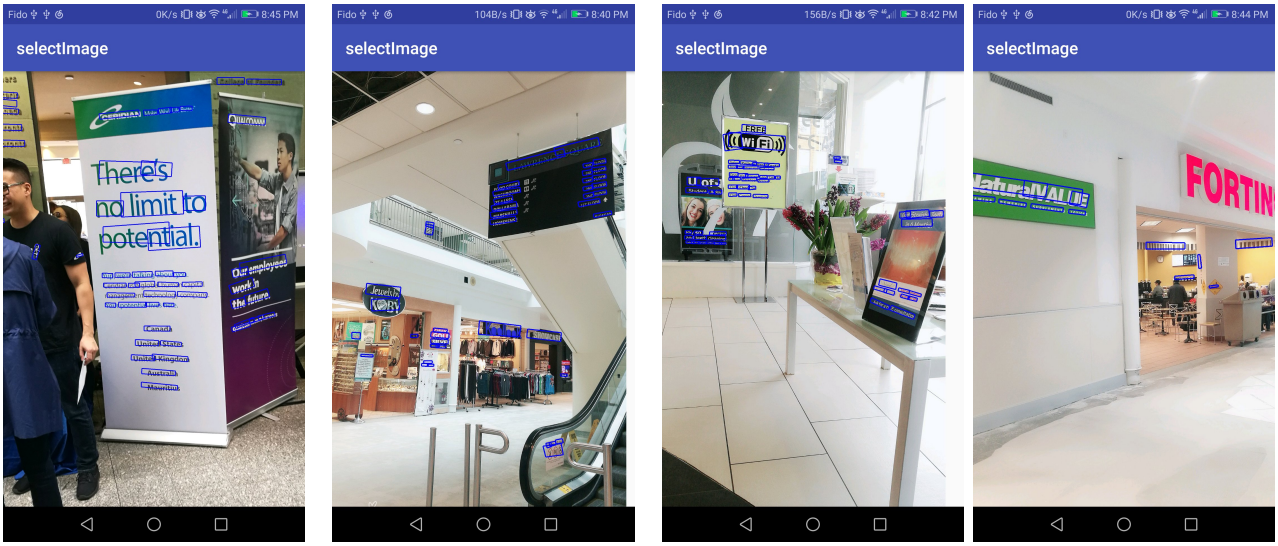

Fig. 3: Text detection examples with Android

B. Loss of geometry map

First, for the AABB part, IoU loss is adopted

$$L_{AABB} = -logIoU(\hat{R}, R^*) = -log(\frac{\hat{R} \cap R^*}{\hat{R} \cup R^*})$$

Where $\hat{R}$ is the predicted geometry and R^* is its corresponding ground truth geometry. Then, the loss of rotation angle is obtained from

$$L_\theta(\hat{\theta}, \theta^*) = 1 - \cos(\hat{\theta} - \theta^*)$$

where $\hat{\theta}$ is the prediction to the rotation angle and θ^* represents the ground truth. Finally, the overall geometry loss is the weighted sum of AABB loss and the rotation angle loss.

$$L_g = L_{AABB} + \lambda_\theta L_\theta$$

where λ_θ is the weight and is set to 10 in the training.

V. Experiments

Two different models derived from the architecture are trained and evaluated. First model extracts features from MobileNets and uses feature fusion block from original EAST model. Second model has the same feature extraction but adopts residual bottleneck layer in its feature fusion block. These two models all trained in the same hyper-parameter configuration. These two models are named Mobile-EAST1 and Mobile EAST2.

To speed up the training process, the feature extraction part of each model is initialized from the pre-trained ImageNet model. I used Adams optimizer with learning rate of 1e-4, decay rate of 0.94 in every 10000 step. Each model is trained on a 4 k80 GPU machine for 12 hours. Although the loss is still decreasing, out of the consideration of the cost. I stopped the training at step 18000. I believe with more training time, the results could be better.

VI. Evaluation

A. Locality-Aware NMS

The score map and geometry map produced by the model have to go through a Non-maximum suppression to get the final results. The geometry map are filtered by their corresponding score map. Those geometries survived from the filtering threshold are merged row by row iteratively. For more information on the NMS method, please see [3].

B. Evaluation metrics

The evaluation followed the evaluation script of ICDAR2015 challenge 4. It is based on a single Intersection-over-Union criterion, with a threshold of 50%. Geometries with over 50% IoU is classified as True otherwise False. The number of Truth Positive, False Positive, False Negative can then be counted. The final evaluation metrics is the Precision, Recall and F1 given the number of TP, TN, FN above.

C. Evaluation results

The evaluation is conducted using the ICDAR2015 test dataset. Two models using different feature fusion blocks are evaluated. These two models are then transfered to frozen models and the results are shown in Table1. In Table1, time is the average inference time when evaluating the 500 image test. Size is the froze model's .pb file's storage size in Mb.

TABLE I: Comparison of text detection models

Model	Precision	Recall	F1	Time(ms)	Size(Mb)
CTPN [2]	0.74	0.52	0.61	*	*
EAST [3]	0.85	0.77	0.81	176	96
Mobile-EAST1	0.66	0.59	0.62	94	8.4
Mobile-EAST2	0.69	0.70	0.70	95	7.7

It can be seen that the Mobile-EAST models are much smaller than the original EAST model, smaller than $^1/_{10}$ of

the size of the ResNet EAST model. The inference time of Mobile-EAST model is also nearly half the time of the ResNet EAST model.However, with this much reduction in model size, the models perform decently with both of them having F1 score higher than 0.60.

Within these two Mobile-EAST models, the second one has considerably better performance than the first one. As these two models both extract feature from the same endpoints, the only difference between these two models is the feature fusion part. The first one used regular convolution layers while the second used residual bottleneck layers. The difference in performance could be resulted from 2 reasons:

1) The residual bottleneck layer's intermediate expansion layer has a high dimension and this high dimension tensor's information is better preserved by the depthwise convolution. Even after projected back to low dimension, the information loss caused by the non-linearity of activation function is minimized compared to regular convolutional layer.
2) Both model used MobileNetV2 as the feature extraction backbone. As the 4 endpoints I chose are all outputs of residual bottleneck layers, a regular convolutional layer could cause inconsistency and lose information.

VII. Implementation on Android

Tensorflow provides a java interface that could run frozen tensorflow models on Android devices. As shown in Fig.4, The input of the input must be compressed into 1D and its output is also 1D array. To use the output, the array must be reshaped to the desired shape.

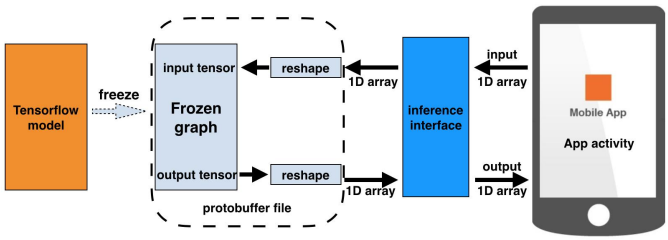

Fig. 4: Android tensorflow interface architecture

Fig.3 are some examples tested on my phone. It can detect most of the text in the image regardless of the complex surroundings. The detector can also detect text with some rotation angles.

However, it also fails in many cases. It will mistakenly classify those well aligned items, such as some decorations as text. As shown in the last image of Fig.3, the detector misclassified a well aligned horizontal area as text. Also, the bounding box is not very compact with the text area. The boundaries between words are overlapped in a lot of cases.

VIII. Conclusions

From this project, it is shown that MobileNets, as well as its major component, residual bottleneck layer have the ability to greatly reduce the model size and computation complexity without too much loss of performance. As deep learning models get lighter and lighter, I believe we will see much more deep learning applications being implanted to mobile devices.

Appendix

TABLE II: The architecture of MobileNetV2

Layer	Input	Operator
1	$512^2 * 3$	conv2d
2	$256^2 * 32$	bottleneck
3	$256^2 * 16$	bottleneck
4	$128^2 * 24$	bottleneck
5	$64^2 * 32$	bottleneck
6	$32^2 * 64$	bottleneck
7	$32^2 * 96$	bottleneck
8	$16^2 * 160$	bottleneck
9	$16^2 * 320$	bottleneck
10	$16^2 * 1280$	bottleneck
11	$1 * 1 * 1280$	bottleneck

References

[1] Girshick, R., 2015. Fast r-cnn. arXiv preprint arXiv:1504.08083.

[2] Tian, Z., Huang, W., He, T., He, P. and Qiao, Y., 2016, October. Detecting text in natural image with connectionist text proposal network. *In European Conference on Computer Vision* (pp. 56-72). Springer, Cham.

[3] Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W. and Liang, J., 2017. EAST: an efficient and accurate scene text detector. arXiv preprint arXiv:1704.03155.

[4] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[5] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.C., 2018. Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation. arXiv preprint arXiv:1801.04381.

[6] Sudre, C.H., Li, W., Vercauteren, T., Ourselin, S. and Cardoso, M.J., 2017. Generalised Dice overlap as a deep learning loss function for highly unbalanced segmentations. *In Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support* (pp. 240-248). Springer, Cham.

[7] ICDAR 2015. http://rrc.cvc.uab.es/?ch=4&com=introduction