IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 25, Issue 1, Ser. I (Jan. – Feb. 2023), PP 37-49
www.iosrjournals.org

Revisiting TextFuseNet: Text Context Enhanced Attention Networks For Scene Text Localization

Hitesh Hinduja
University of Mumbai, India

Mohammed Abbas Ansari
Jamia Millia Islamia (JMI), University, India
**Corresponding Author: Hitesh Hinduja*

Abstract:
The task of localizing text in scene images is a challenging task due to varying possible orientations, fonts, and backgrounds. Methods that can model a richer context of text instances outperform models that only consider limited feature representation of the text instance. TextFuseNet is one such method that uses three levels of hierarchical feature representations - character-level, word-level, and global-level features of text instances present in the images and uses them to be fused with text proposal's feature maps to make richer fused features. To further improve the richness of the context of a text instance, we use efficient attention to model the long-range dependencies across the feature maps in several locations of the network. We conduct experiments on the SynthText dataset and achieved a 9.3% improvement in average precision with attention added after the third and fourth ResNet blocks, after each FPN layer, and within the global segmentation mask compared to the baseline of no attention added to the TextFuseNet architecture. Hence demonstrating the effectiveness of adding efficient attention to the TextFuseNet framework.

Keywords:
Efficient Attention, Text Localization, Feature Fusing, TextFuseNet, Mask R-CNN, SynthText dataset.

Date of Submission: 15-01-2023 Date of Acceptance: 31-01-2023

I. Introduction

Text localization in images is the task of detecting any text instances that might be present in the scene image along with their locations. Generally, object localization involves drawing bounding boxes around the object to be detected. Accurate text localization remains an open problem of active research. The problem is challenging since text can be present in any orientation with varying fonts and possibly obscured in a scene image. Deep learning techniques have been the most effective method to solve the problem of text localization [Yao et al, 2016][Zhang et al, 2018][Lyu et al, 2018].

The TextFuseNet framework [Lyu et al, 2018] demonstrated state-of-the-art accuracies for text localization. TextFuseNet is a modification of the Mask R-CNN framework [He et al, 2017]. The Mask R-CNN framework involves creating a feature pyramid on which candidate regions are proposed which are then classified and refined through bounding box regression. Also, a mask is generated for instance segmentation. One of the limitations of just using the mask R-CNN framework is that text is detected based on a single region of interest (RoI) without considering the global context. Thus, the RoI pooling acts as a bottleneck for semantic information to be used for accurate text detection. To remedy this, TextFuseNet modifies the original pipeline to generate three levels of feature representations: character-level, word-level, and global-level. The newer features are fused with the pooled feature maps to create richer fused features. These representations provide the required semantic information to guide the detection process, thus improving localization accuracy.

Therefore the goal of recent research has been to essentially solve the bottleneck of semantic information caused by pooling layers and the kernel sizes of convolutions. The goal is to provide the model with as much semantic context as possible to detect texts accurately. Understanding what is present around the text area is essential for accurately detecting the location of text instances. Thus the goal is to model long-range dependencies across the image, or within the feature maps of a deep learning network.

Dot-product attention is a popular method to model long-range dependencies and has been pivotal to the field of natural language processing[Bengio et al, 2015][Vaswani et al, 2017]. It has also been used to try to solve tasks in computer vision [Wang et al, 2018] but found limited success due to its quadratic memory and computational complexities. Efficient attention [Zhouran et al, 2021] is mathematically equivalent but a

modification of dot-product attention and has linear memory and computational complexities. It also demonstrated state-of-the-art accuracies for the task of stereo depth estimation. Thus the efficient attention module allows freer use of dot-product attention within several locations of a network.

In this paper, we demonstrate the effectiveness of adding efficient attention modules within the TextFuseNet framework. We conduct an ablation study for the addition of attention into different locations of the network trained on the SynthText dataset [Gupta et al, 2016]. We also find that the addition of attention after the 3rd and 4th ResNet blocks, after each FPN layer, and within the global segmentation mask of the TextFuseNet architecture yields the best localization accuracy.

II. Background and Scope:

Localizing text in scene images has been an area of active interest due to its wide applications in many practical applications such as robot navigation, reading road-signs by self-driving cars, and text-based image retrieval. It has been an active area of interest since the last decade.

Pre-deeplearning era, the work by the computer vision community went in two parallel streams of approaches to solve the problem of text localization. These approaches were connected components analysis (CCA) and sliding windows classification. CCA methods extract candidate components using several techniques and then filter out non-text components using classifiers trained on hand-crafted features. [Jain & Yu, 1998] made use of several pre-processing of the original image and performed connected component analysis to extract several connected parts and then edge features are computed for these parts which are used to classify the text region and obtain location coordinates using SVM. [Ephtein et al, 2010] presented Stroke width transform (SWT) which is based on grouping pixels based on assuming consistent stroke width within each character. [Neumann & Matas. 2010] made use of Maximally stable Extremal Regions (MSERSs) of text as features. [Yi & Tian, 2011] proposed two algorithms for detecting text strings based on grouping adjacent characters and grouping text-lines together. [Huang et al, 2013] proposed a modified SWT called Stroke Feature Transform (SFT) which had better performance, SFT's output was classified into text components and text-lines. [Yin et al, 2014] extracted MSERs as character candidates which were grouped into text candidates using a single-link clustering algorithm which was learned by a self-training distance metric learning algorithm.

The deep learning era began with the seminal work of using a deep CNN architecture called AlexNet for the classification of objects in images from the Imagenet dataset [Krizhevsky et al, 2012], achieving the best classification accuracies. It demonstrated the effectiveness of multiple stacked convolutional blocks for extracting useful representations from visual data i.e. images. Convolutional neural networks also showed their efficacy in the task of end-to-end text recognition [T. Wang et al, 2012] and scene text detection [Huang et al, 2014], making them the basic building blocks of all future text recognition and detection architectures.

The convolution operation restricts the field of view of the network to the kernel size of the filter, hence cannot capture global dependencies for localizing text. To tackle this limitation, [Yao et al, 2016] first produces global, pixel-wise prediction maps to capture global context, on which text detection is performed. Methods casting text localization as an object detection problem emerged with the prominent work of [Jaderberg et al, 2016] which proposed a framework similar to R-CNN [Girshik et al, 2014], where word candidate bounding box proposals were quickly generated with high recall which was classified by a random forest classifier and the boxes were refined using CNNs. [Zhang et al, 2018] modified the R-FCN framework [Dai et al, 2016] with a feature enhance network which essentially fused lower and higher-level features from the Resnet-101 backbone [He et al, 2016], and also introduced a modified ROI-pooling layer which was position-sensitive as compared to position-agnostic normal ROI pooling, thus improving localization accuracies.

The idea of backbone feature hierarchies (FPN) and an improved ROI pooling layer (ROIAlign) was best introduced with the Mask R-CNN framework by [He et al, 2017]. It involved four components: a feature pyramid network (FPN) [Lin, Dollar, et al, 2017] formed out of a backbone feature extractor, a region proposal network (RPN) [Ren et al, 2015], a Fast R-CNN for bounding box regression, and a mask branch for instance segmentation. [Lyu et al, 2018] used the Mask R-CNN framework for spotting text with arbitrary shapes, achieving a state-of-the-art performance. The effort to capture more context-aware features within the Mask R-CNN framework carried on with the work of [Lyu et al, 2018] where they were able to extract three levels of feature representations: character-level, word-level, and global-level and fused them to create richer features for text detection. It also introduced a global segmentation map to create the global-level features to guide text detection. By capturing better global context, TextFuseNet demonstrated state-of-the-art localization accuracies.

The idea of capturing global context to improve localization is essentially trying to model long-range dependencies across the feature maps or regions in the image. A popular method to capture such dependencies is the attention mechanism. It has found rampant usage in the natural language processing domain [Bengio et al, 2015][Vaswani et al, 2017]. [Pan et al, 2017] made use of regional attention using aggregated features of the inception modules for creating a single shot text detector. The dot-product attention [Bengio et al, 2015] was

adapted for computer vision in the non-local module introduced by [Wang et al, 2018]. The memory and computation cost of dot-product attention grows quadratically with the input size, hence limiting its usage. Efficient Attention [Zhouran et al, 2021] introduced a mathematically equivalent but more efficient implementation of dot product attention with linear complexities. Thus efficient attention modules can be added flexibly in different locations of the network architecture to model long-range dependencies.

III. Materials and Methodology:

Our proposed work is inspired by [Zhouran et al, 2021] use of efficient attention within a mask RCNN framework [He et al, 2018] for the task of localizing text in images. We incorporate attention to the TextFuseNet [Ye et al, 2020] architecture which involves extracting multi-level text feature representations in the detection and mask branch of the Mask-RCNN framework. TextFuseNet extracts global-level, word-level, and character-level features and fuses them to form richer features for the task of bounding box regression, classification, and instance segmentation. In this section, we shall describe how we can localize text using TextFuseNet, the efficient attention mechanism, and where it can be incorporated into the network architecture.

Our hypothesis was that the incorporation of attention modules in different locations of the TextFuseNet architecture will yield better localization accuracy.

3.1. TextFuseNet:

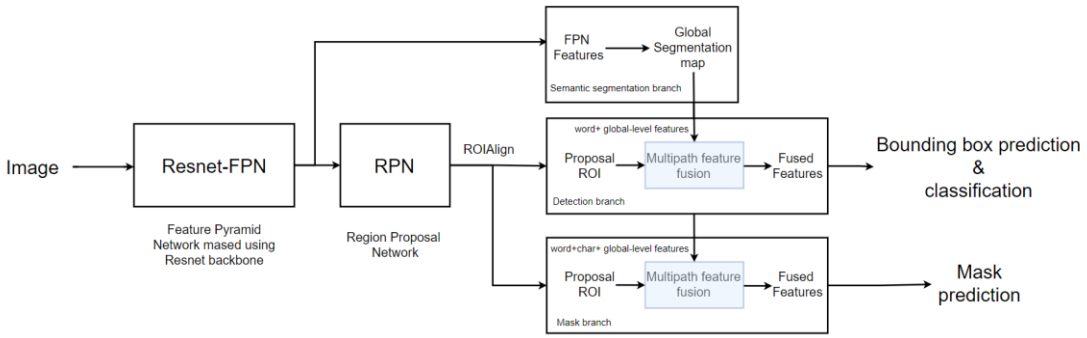


Figure 1: The overall pipeline of the TextFuseNet architecture. It follows the mask RCNN framework but also involves multi-level feature fusion of word-level, character-level, and global-level features using a multi-path feature fusion module as shown in the figure.

3.1.1. Framework

The Textfusenet uses ResNet [He et al, 2016] to extract features from the input image. ResNets has repeating blocks whose output feature maps of varying scales are used to create a feature pyramid network (FPN). Then a region proposal network (RPN) generates proposals at all the levels of the feature pyramid using anchors. The proposals are classified as objects (text) or backgrounds by the RPN. For each proposal, ROIAlign pooling is performed. The pooled proposals then pass through two branches i.e. detection branch and mask branch. The detection branch classifies the object in the proposal and performs bounding box regression. The mask branch performs instance segmentation on the text in the pooled proposal.

The features of FPN are also used to create a third branch i.e. segmentation branch which generates a segmentation map for all objects i.e. text in the image. The segmentation map is used to provide global context to create richer fused features within the detection and mask branch. Within the detection branch, the pooled proposals' feature maps are fused with the global segmentation map using a multi-path feature fusion module to give fused features on which bounding box regression and classification occurs. Within the mask branch, the pooled proposals are classified into character-proposals and word-proposals. The character-level, word-level, and global-level (segmentation map) features are then fused using the multi-path fusion module to give richer fused features for instance segmentation.

Therefore, the overall objective of the TextFuseNet architecture for text localization is

$$L = L_r + L_{seg} + L_{det} + L_{mask} \quad (1)$$

Where L is the total loss for our objective which is a sum of the loss function of RPN, semantic segmentation branch, detection branch, and mask branch.

3.1.2. Segmentation branch:

All the levels of the FPN are upsampled and fused by elementwise summation to produce a single feature map. And then this feature map is upsampled to give a feature map of the same shape as the input image to produce a binary mask of where text is present in the image i.e. segmentation map. This provides the global contextual information necessary for guiding regression of boxes and instance segmentation in detection and mask branch respectively.

After pooling, the field of view of the network is restricted to the region of interest, hence losing important information that might lie beyond the region of interest that can help in classifying the proposal and performing instance segmentation for the object in the proposal. Therefore a global context map is computed to solve this issue which will be fused within the features in the detection and mask branch.

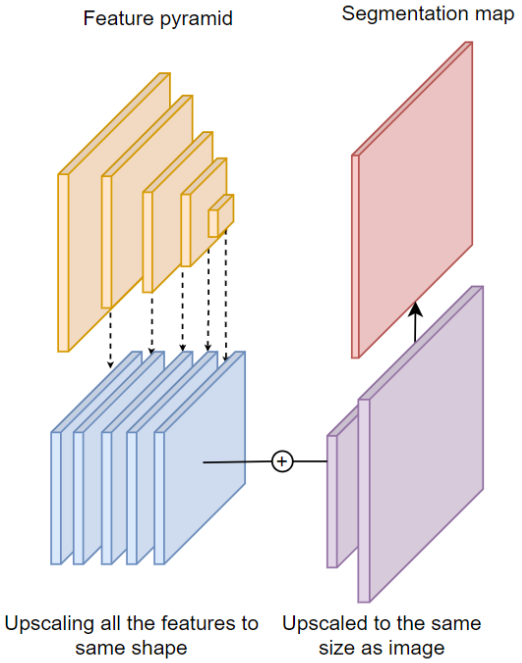


Figure 2: **Illustration of creation of global segmentation map.** All the levels of FPN are upscaled to a common shape, allowing for elementwise summation across the different levels after which the fused feature map is further upscaled to be the same shape of the image to create a segmentation map.

3.1.3. Detection branch

After pooling the proposals using ROIAlign, word-level feature maps are obtained. Generally, these feature maps are further convolved to give layers for computing classes and coordinate logits to perform classification and bounding box regression for each proposal. To guide the process, feature maps can be enriched with global context features using the segmentation map. Thus multi-path fusing is performed within the detection branch between word-level and global-level features.

The global segmentation map is downsampled to the same shape as the pooled proposals and performs elementwise summation of the context to each of them. The multipath fusion module is responsible for downsampling and performing summation, on which further convolutions are applied to bridge the semantic gap among the different features. Residual connections are also provided to ensure a degree of freedom for the pooled features to whether or not to make use of global context features.

3.1.4. Mask branch:

Within the mask branch, the word proposals and character proposals have been identified by the RPN. For each word proposal, its corresponding character proposals are found. Thus character-level features can be created for each word proposal. Also to guide instance segmentation, global context is provided to the proposals using the downsampled global segmentation map features. Hence character-level, word-level, and global-level features are fused to give richer fused features to perform instance segmentation.

For a word proposal $\boldsymbol{r}_i$, a ratio of the area of intersection of character proposals with r over the character's area is determined. A character proposal belong to the word $\boldsymbol{r}_i$ if the ratio is over a threshold (which was 0.8 in our implementation). Thus for input word $\boldsymbol{r}_i$, a set of characters $\boldsymbol{C}_i$ can be identified:

$$C_i = \{c_i \mid (a_i \cap a_j) \div a_j > T\} \quad (2)$$

Where a_i and $a_j$are areas of character proposal's bounding box and word proposal's bounding box respectively, and T is the threshold.

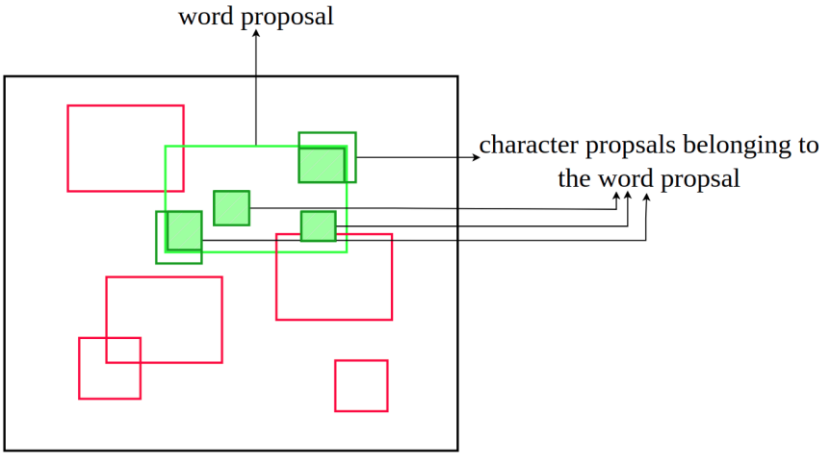


Figure 3: **Illustration of how a character proposal is considered to belong to a word proposal.** The above region represents the proposals generated by the RPN on one of the levels of FPN. A character proposal belongs to a word if the fraction of its area of intersection with the word over its own area is greater than threshold T (0.8 in our implementation).

The character set $\boldsymbol{C}_i$ corresponding to each word will vary in cardinality. Therefore the character proposals are first extracted into 14x14 shapes using ROIAlign and then are fused through elementwise summation. Further convolutions are applied to give character-level features corresponding to each word-level feature. The character-level features provide local context, thus instance segmentation of the word proposal, when fused with its character features, occurs more accurately.

The global feature map is downsampled to the same shape as character-level and word-level feature maps. For each word-level feature map, its corresponding character-level features are added with the global features through elementwise summation. After summation, further convolutions are applied to bridge the semantic gap and residual connections are also provided from the initial word proposal as well. Thus, richer fused features are obtained within the mask branch.

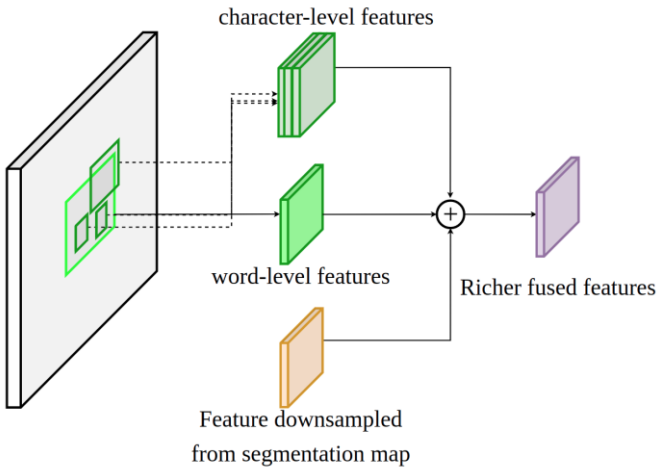


Figure 4: **Illustration of multipath fusion.** On the left are the proposals generated on a level of FPN by the RPN. Character-level features, word-level features, and global-level features are extracted and then fused together through element-wise summation to give richer fused features for instance segmentation.Illustration of multipath fusion. On the left are the proposals generated on a level of FPN by the RPN. Character-level features, word-level features, and global-level features are extracted and then fused together through element-wise summation to give richer fused features for instance segmentation.

3.2 Efficient Attention (EA):

Attention is a mechanism for modeling long-range dependencies within neural networks. In a computer vision task, it allows each pixel within a feature map to attend to all the other feature maps. Hence the field of view increases with the use of attention mechanisms. Generally in an attention mechanism, pairwise similarity between queries $\boldsymbol{Q}$ and keys $\boldsymbol{K}$ is measured for each position and use the similarities as weights to sum over the values $\boldsymbol{V}$ for all positions [Poupart, 2019].

$$attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \sum_i similarity(\boldsymbol{Q}, \boldsymbol{k_i}) \times \boldsymbol{v_i} \tag{3}$$

The similarity function is what differs among the different attention mechanisms. Consider that we have an input feature map $\mathbb{R}^{h \times w \times c}$ and flatten it to $n = hw \Rightarrow \mathbb{R}^{n \times c}$. Hence we have n positions each of depth c. This feature map of size $n \times c$ is passed through convolutions to produce 3 features: Queries $\boldsymbol{Q} \in \mathbb{R}^{n \times d_k}$, Keys $\boldsymbol{K} \in \mathbb{R}^{n \times d_k}$ and Values $\boldsymbol{V} \in \mathbb{R}^{n \times d_v}$. Then the efficient attention mechanism [Zhouran et al, 2021] works as per the following equation:

$$\boldsymbol{E}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \rho_q(\boldsymbol{Q}) \, (\rho_k(\boldsymbol{K})^{\mathrm{T}} \, \boldsymbol{V}) \tag{4}$$

where ρ_q and ρ_k are normalization functions for $\boldsymbol{Q}$ and $\boldsymbol{K}$ respectively. In practice, we use the softmax function for normalization.

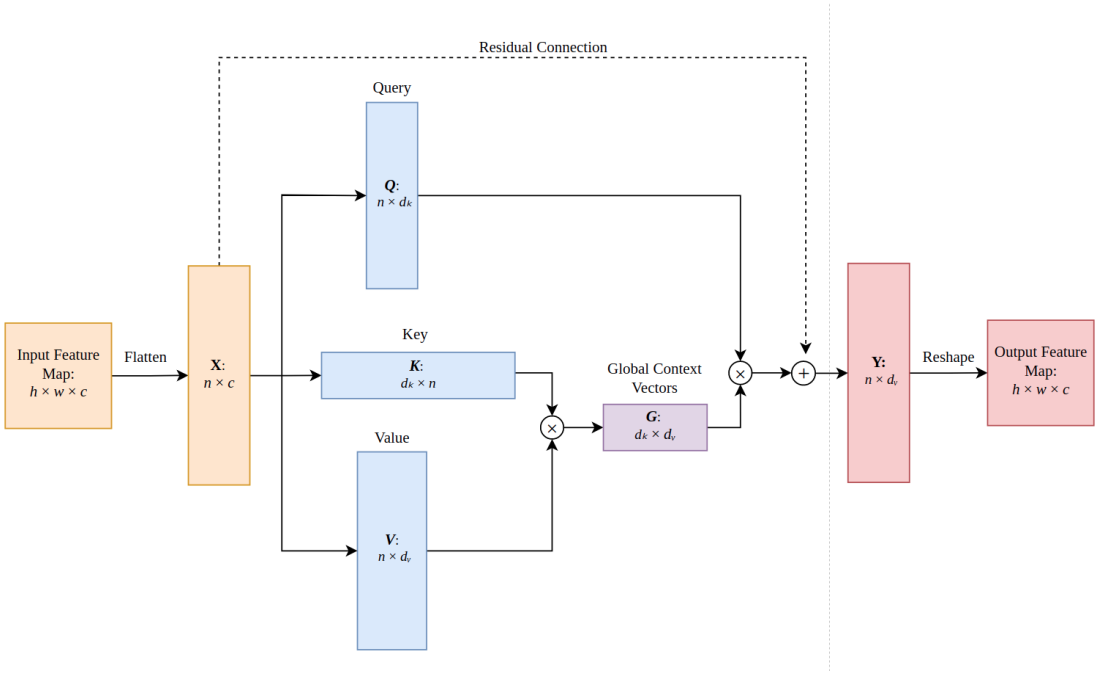


Figure 5: Efficient attention mechanism illustration

Efficient attention essentially uses the keys $\boldsymbol{K} \in \mathbb{R}^{n \times d_k}$ as d_k template attention maps. Each template attention map is capable of capturing the different semantic aspects of the input feature map. These semantic aspects can be the foreground, background, center of the text, edges of text, etc. The values $\boldsymbol{V} \in \mathbb{R}^{n \times d_v}$, which depend on the input, are aggregated using the keys in $\boldsymbol{K}$ through the linear combination to produce global context vectors $\boldsymbol{G} \in \mathbb{R}^{d_k \times d_v}$. Vectors in $\boldsymbol{G}$ also summarize a global semantic aspect of the input feature map. For each position, i in the input feature, each query in $\boldsymbol{Q}$ i.e. $\boldsymbol{q}_i \in \mathbb{R}^{1 \times d_k}$ is used as coefficients over the sum of the global context vectors $\{\boldsymbol{g}_0, \boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_{d_k - 1}\}$ where $\boldsymbol{g}_j \in \mathbb{R}^{1 \times d_v}$:

$$\boldsymbol{e}_i = \boldsymbol{q}_{i,0}\boldsymbol{g}_0 + \boldsymbol{q}_{i,1}\boldsymbol{g}_1 + \boldsymbol{q}_{i,2}\boldsymbol{g}_2 + \boldsymbol{q}_{i,3}\boldsymbol{g}_3 + \ldots + \boldsymbol{q}_{i,d_k - 1}\boldsymbol{g}_{d_k - 1} \tag{5}$$

where $\boldsymbol{e}_i$ is the i^{th} position's efficient attention output. For example, if there is a text pixel at position i, $\boldsymbol{q}_i$ would be such that the weightage of global context vectors $\boldsymbol{g}$ capturing text would be greater. Thus allowing the representation at i to be refined using attention.

The total output feature map is $\boldsymbol{E} \in \mathbb{R}^{n \times d_v}$. If the dimension of values $d_v \neq c$ i.e. to the number of channels in the input feature, 1×1 convolutions are applied to restore the original dimensions. A residual connection from the input feature is also added to the output of the Efficient attention block to ensure that the neural network doesn't lose important representations during the computation of attention.

3.3. Details of adding Efficient Attention modules (EA):

The efficient attention module can be added into several locations of the TextFuseNet architecture. These modules can help model long-range dependencies across the feature map in different locations to help improve localization accuracy. Here we shall discuss these locations and how the EA module has been incorporated in detail.

At each location, the dimensions of keys and values have been both set to 64. The output feature shape of the EA block will be the same as the input feature shape. Adding the EA module cannot decrease the localization accuracy compared to the baseline since residual connections are provided as well.

3.3.1. ResNet backbone:

The ResNet backbone is built using blocks that are repeated multiple times. The blocks are Res2, Res3, Res4, and Res5 which are repeated depending upon the resnet depth, for example in ResNet-50, they are repeated 3, 4, 6, 3 times respectively. As we move along the depth of the backbone, the outputs of each block capture more and more abstract representations of the input image. We can add EA modules in between each block to allow the network to attend to specific locations of the input image where text can be present.

We will add EA after the Res3 block [28×28×128], Res4 block [14×14×256], Res5 block [7×7×512]. The outputs of each block are used as input to the next block and also used to create the feature pyramid network (FPN).

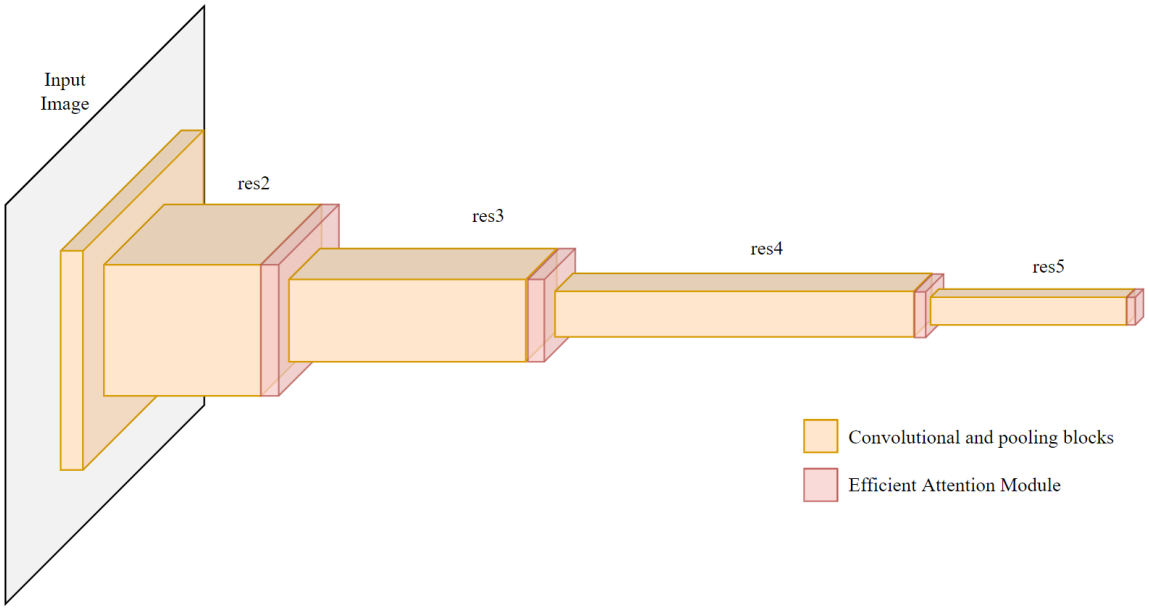


Figure 6: **Illustration of addition of efficient attention to the ResNet backbone.** ResNet is made up of repeating blocks such as Res1. Res2, Res3 and Res4, between which efficient attention modules can be added. In the above figure, red areas depict the possible locations for adding efficient attention modules.

3.3.2. Feature Pyramid Network (FPN):

As the layers of the pyramid are built in a top-down approach, where the top (smaller) layer of FPN is taken and interpolate it to be the same size as the next ResNet input. The next ResNet input is known as the lateral feature. Then the lateral features and top-down features are added element-wise to produce the next level of FPN. After the element-wise summation to create the level of FPN, we added the efficient attention module with the same channels as the level with dimensions of keys and values as 64.

We add the EA module similarly for each level of FPN. Adding these EA layers makes it possible for the model to be able to attend to specific regions of features at each FPN level. Therefore it can improve the accuracies of

the region proposal network and similarly for the global segmentation mask which makes use of the features of FPN.

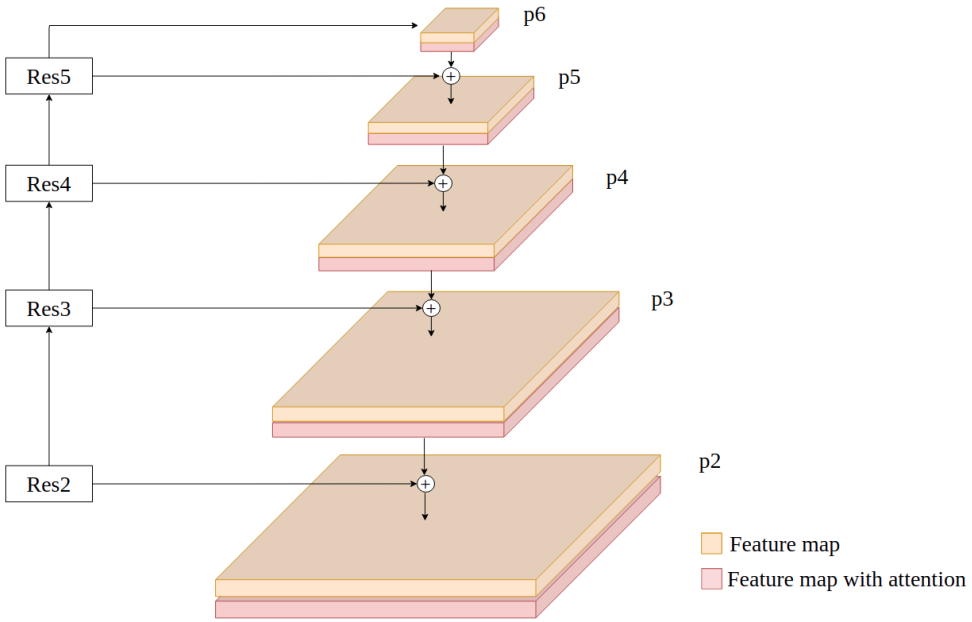


Figure 7: **Illustration of addition of efficient attention to the feature pyramid network.** Efficient attention can be added between each layer of the FPN where each feature map goes through the attention module and then is combined with the lateral features from the ResNet backbone through elementwise summation to make the next layer of the feature pyramid.

3.3.3. Multi-path-fusion module:

Multipath fusion module takes multiple level features i.e. global, word, and character level, and fuses them through element-wise summation to give richer fused features. We add EA just before global mask features are used as input to the multipath fusion module, thus we send global mask features with attention as input for multipath fusion. We can also add EA after the fusion of all the multi-level features i.e. we add attention to the richer fused features.

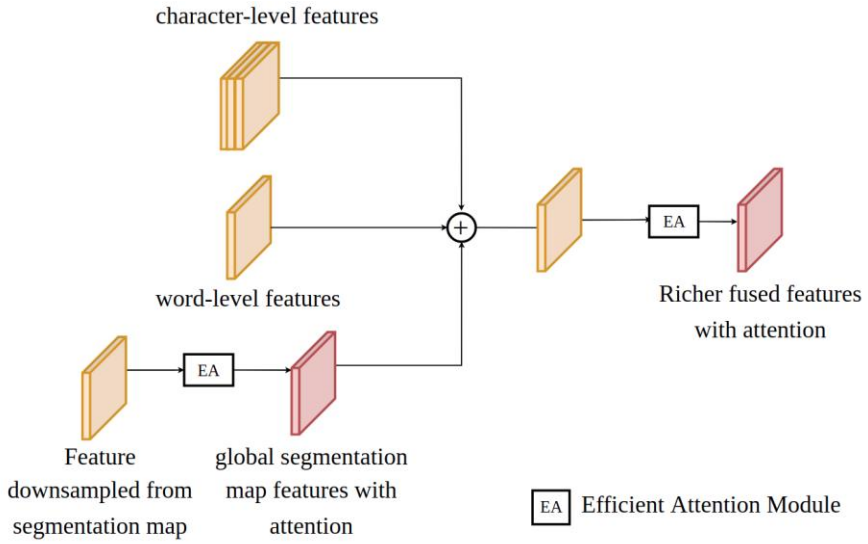


Figure 8: **Illustration of addition of efficient attention to the multipath fusion module.** The efficient attention module has been after the downsampled features from the segmentation map and also after the fused feature map.Red blocks represent feature maps after attention.

IV. Experiments:

To test our hypotheses, we shall describe the dataset that was used to train and test and the experiments that were conducted with different locations of attention incorporation into the TextFuseNet architecture.

4.1. Infrastructure Limitations:

We were restricted due to the limited infrastructure available to us for training on large datasets. We only had access to GPUs through free google colab access. Google colab provides access to only one GPU for a limited time.

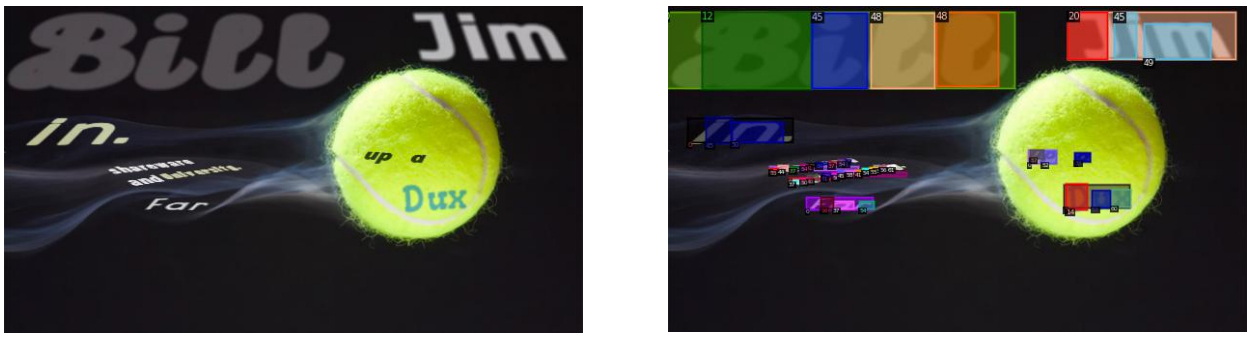
Therefore we opted for conducting small-scale experiments where we shall take a smaller dataset and train for a limited number of iterations. We ensured that hyperparameters of training remain the same for all the different modified architectures we are trying to compare. All of the models were trained on the same training set and then we inferred on the same test set and computed localization accuracy for the bounding box and mask using Average precision as our metric.

The fixed hyperparameters, same train set, and test set served as a *control* of the experiment, enabling us to compare localization accuracy of different modified architectures and deduce observations from the results.

4.2. Dataset:

TextFuseNet architecture requires word-level and character-level annotations for text present in images to be trained. We decided to test our hypothesis using a dataset that did have character-level annotation i.e.SynthText dataset.

SynthText dataset was curated by [Gupta et al] in 2016, where they used an engine to overlay synthetic (generated) text to existing background images in a natural way, accounting for the local 3D-scene geometry. They generated 800,000 synthetic text images using their engine where each image had about 10-word instances



annotated with character and word-level bounding boxes.

Figure 9: Text is overlaid on images using a generator that takes into account 3-D geometry to ensure realism. On the left, we have an image from the SynthText dataset, and on the right are its annotations.

Using the existing annotations, we created a segmentation mask using polygon coordinates of the bounding boxes themselves. Thus each object (character or word) in the dataset had the following labels:

1. bounding box [***x***, ***y***, ***w***, ***h***] where (***x***,***y***) are the coordinates of the top-left corner and ***w*** and ***h*** are the absolute width and height of the box.
2. segmentation mask [***x1***, ***y1***, ***x2***, ***y2***, ***x3***, ***y3***, ***x4***, ***y4***] where (x_i, y_i) are coordinates of the polygon mask in clockwise order starting from the top-left corner.
3. class label out of the following classes:

{*"text", 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z*}

Therefore we have a total of 63 categories of objects where 1 is for text and 62 for alphanumeric characters.

The training set contained 2428 images sampled randomly from the SynthText dataset, similarly the test set contained 122 images which were also sampled randomly from the SynthText dataset.

Training Set						Test Set					
category	# instances	category	# instances	category	# instances	category	# instances	category	# instances	category	# instances
text	20776	*k*	578	*F*	512	*text*	883	*k*	29	*F*	12
0	305	*l*	2415	*G*	159	*0*	6	*l*	96	*G*	6
1	488	*m*	1506	*H*	228	*1*	21	*m*	58	*H*	8
2	322	*n*	4746	*I*	689	*2*	12	*n*	196	*I*	30
3	335	*o*	5455	*J*	124	*3*	18	*o*	221	*J*	9
4	146	*p*	1203	*K*	100	*4*	7	*p*	50	*K*	1
5	181	*q*	32	*L*	505	*5*	9	*q*	2	*L*	15
6	143	*r*	4109	*M*	399	*6*	4	*r*	177	*M*	19
7	104	*s*	4163	*N*	358	*7*	5	*s*	195	*N*	12
8	243	*t*	7043	*O*	237	*8*	4	*t*	303	*O*	14
9	325	*u*	2111	*P*	207	*9*	19	*u*	95	*P*	10
a	5857	*v*	620	*Q*	15	*a*	275	*v*	26	*Q*	1
b	2438	*w*	1583	*R*	389	*b*	41	*w*	52	*R*	13
c	1449	*x*	144	*S*	553	*c*	63	*x*	5	*S*	13
d	2438	*y*	1419	*T*	704	*d*	90	*y*	64	*T*	39
e	9356	*z*	78	*U*	126	*e*	391	*z*	5	*U*	3
f	1319	*A*	648	*V*	71	*f*	57	*A*	27	*V*	3
g	1022	*B*	243	*W*	184	*g*	49	*B*	11	*W*	7
h	4639	*C*	240	*X*	93	*h*	194	*C*	9	*X*	2
i	3828	*D*	582	*Y*	88	*i*	191	*D*	35	*Y*	3
j	151	*E*	222	*Z*	9	*j*	2	*E*	12	*Z*	0
Total number of instances: 99112						Total number of instances: 4229					

Table 1: Distribution of instances among the different classes in the sampled training set and test set. There were 99112 total instances in the training set and 4229 instances in the test set.

4.3. Evaluation Metric:

The task of our model is to draw bounding boxes and polygons for localization and segmentation respectively of characters/text in an image. It also classifies each identified object into 63 classes as defined earlier. To evaluate the effectiveness of the model, we must compare the predicted labels with the ground-truth labels.

We consider a predicted bounding box/polygon to be a true-positive(TP) if its intersection over union (IoU) with the ground truth bounding box/polygon is above a certain threshold. A prediction would be a false-positive(FP) if its IoU is below the threshold. If a ground-truth bounding box/polygon doesn't have an IoU with any predictions over the threshold, we consider it to be a false-negative(FN) since our model has not been able to predict it. For the different thresholds of IoU, we will have different numbers of true-positives, false-positives,and false-negatives. Precision is defined as the fraction of true positives overall predictions for bounding boxes/polygons made by the model. Whereas, recall is defined as the fraction of true positives overall ground truths i.e. actual bounding-boxes/polygons that exist in the image.

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN} \quad (6)$$

Average Precision (AP) is defined as the area under the precision-recall (PR) curve. For the 63 different possible classes of objects, we calculate AP separately for each class and then take the mean of all APs to give mean average precision (mAP). We also calculate mAP at different IoUs, mAP calculated at 0.5 IoU is termed AP50, similarly, mAP calculated at 0.75 IoU is termed AP75. The mAP is also calculated separately for objects at different scales, giving APs, APm, and APl where they are mAP for small, medium, and large objects respectively. Using the metrics such as AP50, AP75, APs, APm, and APl, we can judge the effectiveness of a model at localizing text in images.

4.4. Implementation Details:

All of our experiments were conducted on a single NVIDIA Tesla P100 (16G) provided through Google Colab. Training and inference were done on a single GPU. Everything was done using the Detectron2 [Wu et al, 2019] framework built using PyTorch.

4.4.1. Architecture:
We use the ResNet-50 backbone and build an FPN that takes in [res2, res3, res4, res5] block's outputs to produce a 5-layer FPN [p2, p3, p4, p5, p6]. In the RPN, we use anchors of sizes [32, 64, 128, 256, 512] and aspect ratios [0.5, 1.0, 2.0]. RPN generates 2000 proposals and chooses the top 1000 proposals after non-maximal suppression (NMS) on each level of FPN. ROI Heads selects a batch of 512 proposals per image from proposals on levels [p2, p3, p4, p5] to perform ROIAlign on. The number of ROI heads is 63 i.e. equal to the number of classes. For the detection branch, ROI head with pooler resolution 7 is used whereas, for mask branch, pooler resolution 14 is used. The segmentation head takes in [p2, p3, p4, p5] levels of FPN and fuses them to produce a segmentation map of the input image. Multipath fusion module was added in the mask branch. Efficient attention can be added in different locations as stated earlier in section 3.3.

4.4.2. Training:
Weights pretrained on Microsoft COCO dataset [Lin et al, 2014] for instance segmentation were used for the Mask RCNN components of our architecture. The added components' weights i.e. EA and multi-path fusion modules were initialized using the Kaiming uniform method [He et al, 2015]. The network was trained using stochastic gradient descent (SGD) with 1 image per batch for 2000 iterations with the momentum equal to 0.9. The base learning rate was 0.001 which was decreased by dividing by 0.1 after the 1000th iteration. Data Augmentation techniques such as multi-scaling and random flipping were also applied during training.

4.4.3. Inference:
During inference, the shortest edge of images in the test set were all resized to 800. We then make predictions on each image and calculate AP50, AP75, APs, APm, and APl for bounding box and segmentation mask prediction separately.

4.5 Ablation Study & Results:
To validate our hypothesis, we first trained the TextFuseNet architecture, without any attention added, to the subset of the SynthText dataset. We then inferred on the test set and obtained localization accuracies for text detection and segmentation respectively. This served as the baseline for our experiments, against which we compared the subsequent additions of attention to understand their utility in improving localization accuracy.

Attention was added in the several locations of the network as per section 3.3, and the network was trained with the same initial configuration as the baseline on the training set. Then localization accuracies on the same test set were recorded for the different combinations of attention incorporation to the network. Table 2 contains all of our observations.

We observed that certain combinations of attention addition did increase the localization accuracy compared to the baseline. Out of which, the combination of attention added to res3, res4, fpn, and global segmentation mask yielded the best improvement of 9.3% in overall average precision. The combination of res3, res4, and fpn demonstrated better results for AP75 and APm, which means that this network makes much tighter predictions around the ground truth for medium-sized text with the incorporation of attention. Hence our hypothesis has been validated.

Location of EA	Box AP						Mask AP					
	AP	AP50	AP75	APs	APm	APl	AP	AP50	AP75	APs	APm	APl
None	1.307	3.217	0.828	1.282	2.203	24.852	1.320	3.209	0.792	1.293	2.216	25.369
res3	1.291	2.993	0.920	1.269	1.910	**28.123**	1.309	2.988	0.976	1.288	1.965	27.451
res3+res4	1.302	3.052	0.807	1.270	1.997	26.950	1.314	3.043	0.925	1.287	2.045	27.572
fpn	1.339	3.041	0.911	1.314	1.842	27.073	1.362	3.015	1.020	1.335	1.919	**27.802**
res3+res4+fpn	1.423	3.354	**0.946**	1.373	**2.569**	26.250	1.442	3.354	**1.028**	1.395	**2.686**	26.854
res3+res4+res5+fpn	1.266	3.065	0.706	1.242	2.434	26.241	1.281	3.068	0.767	1.255	2.486	27.139
fpn+seg	1.352	3.395	0.816	1.341	2.403	24.214	1.367	3.384	0.906	1.354	2.403	23.633
res3+res4+fpn+seg	**1.429**	**3.427**	0.911	**1.425**	2.292	25.241	**1.460**	**3.460**	0.992	**1.459**	2.351	26.370
res3+res4+fpn+seg+ m(g)	1.278	3.286	0.693	1.246	2.362	25.225	1.295	3.297	0.743	1.263	2.439	25.274
res3+res4+fpn+seg+ m(f)	1.212	2.988	0.757	1.197	1.684	25.216	1.237	3.015	0.837	1.217	1.762	26.011
res3+res4+fpn+seg+m(f+g)	1.320	3.246	0.869	1.293	2.040	25.845	1.321	3.211	0.861	1.293	2.145	26.152

Table 2: Localization Accuracies of different experiments on the subset of SynthText dataset for the task of text detection (Box AP) and segmentation (Mask AP). 'seg' stands for attention added to global segmentation mask, 'm' stands for multipath fusion module and attention added after downsampling of global segmentation map (g) and after feature fusion (f). Please refer to section 3.3 for implementation details.

V. Conclusion:

In this paper, we demonstrated that adding efficient attention modules in several locations of the TextFuseNet architecture will improve localization accuracies. We also discovered through our experiments that the best combination of addition of attention to the network is after the 3rd and 4th ResNet blocks, after each FPN layer, and within the global segmentation mask. A better state-of-the-art model can be achieved with the use of efficient attention in TextFuseNet, hence moving closer to solving the problem of text localization in images.

References:

[1]. [Jain & Yu, 1998] Jain, A.K. and Yu, B. Automatic Text Location in Images and Video Frames. Pattern Recognition (1998) , 31, pp. 2055-2076

[2]. [Ephtein et al, 2010] B. Epshtein, E. Ofek, and Y. Wexler. Detecting Text in Natural Scenes with Stroke Width Transform. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2010), pp. 2963–2970

[3]. [Neumann & Matas, 2010] Neumann L., Matas J. A Method for Text Localization and Recognition in real-world images. Proceedings of the Asian Conference on Computer Vision (2010), pp. 770-783

[4]. [Yi & Tian, 2011] Chucai Yi, Yingli Tian. Text String Detection From Natural Scenes by Structure-Based Partition and Grouping. IEEE Transactions on Image Processing 20(9) (2011), pp. 2594-2605

[5]. [Krizhevsky et al, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet Classification with Deep Convolutional Neural Networks. Proceedings of the 25th International Conference on Neural Information Processing Systems (2012), pp. 1097–1105

[6]. [T. Wang et al, 2012] Wang, Tao, David J. Wu, Adam Coates, and A. Ng. End-to-end text recognition with convolutional neural networks. Proceedings of the 21st International Conference on Pattern Recognition (2012), pp. 3304-3308.

[7]. [Huang et al, 2013] Huang, Weilin, Zhe L. Lin, Jianchao Yang, and Jue Wang. Text Localization in Natural Images Using Stroke Feature Transform and Text Covariance Descriptors. IEEE International Conference on Computer Vision (2013), pp. 1241-1248

[8]. [Yin et al, 2014] Xu-Cheng Yin, Xuwang Yin, Kaizhu Huang, and Hong-Wei Hao. Robust Text Detection in Natural Scene Images. IEEE Transactions on Pattern Analysis and Machine Intelligence 36(5) (2014), pp. 970-9831

[9]. [Huang et al, 2014] Weilin Huang, Yu Qiao, and Xiaoou Tang. Robust Scene Text Detection with Convolution Neural Network Induced MSER Trees. Proceedings of the European Conference on Computer Vision (2014), pp. 497-511

[10]. [Girshick et al, 2014] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2014), pp. 580-587

[11]. [Lin et al, 2014] Lin, Tsung-Yi, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. Proceedings of the European Conference on Computer Vision (2014)

[12]. [Bengio et al, 2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. Proceedings of the International Conference on Learning Representations (2015)

[13]. [He et al, 2015] He, Kaiming, X. Zhang, Shaoqing Ren and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. Proceedings of the IEEE International Conference on Computer Vision (2015), pp. 1026-1034.

[14]. [Yao et al, 2016] Cong Yao, Xiang Bai, Nong Sang, Xinyu Zhou, Shuchang Zhou, and Zhimin Cao. Scene Text Detection via Holistic, Multi-Channel Prediction. arXiv preprint (2016) arXiv:1606.09002

[15]. [Jaderberg et al, 2016] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading Text in the Wild with Convolutional Neural Networks. International Journal of Computer Vision 116 (2015), pp. 1-20

[16]. [Dai et al, 2016] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks. Proceedings of the International Conference on Neural Information Processing Systems (2016), pp. 379–387

[17]. [He et al, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016), pp. 770-778

[18]. [Gupta et al, 2016] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic Data for Text Localisation in Natural Images. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016)

[19]. [He et al, 2017] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. Proceedings of the IEEE International Conference on Computer Vision (2017), pp. 2980-2988

[20]. [Lin, Dollar et al 2017] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017), pp. 936-944

[21]. [Ren et al, 2015] Shaoqing Ren, Kaiming He, and Ross Girshick. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (2015), pp. 1137-1149

[22]. [Vaswani et al, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All You Need. arXiv preprint (2017) arXiv:1706.03762v5

[23]. [Pan et al, 2017] Pan He, Weilin Huang, Tong He, Qile Zhu, Yu Qiao, and Xiaolin Li. Single Shot Text Detector with Regional Attention. Proceedings of the IEEE International Conference on Computer Vision (2017), pp. 3066-3074

[24]. [Zhang et al, 2018] Sheng Zhang, Yuliang Liu, LianwenJin, and Canjie Luo. Feature Enhancement Network: A Refined Scene Text Detector. Proceedings of the AAAI Conference on Artificial Intelligence (2018), pp. 2612–2619

[25]. [Lyu et al, 2018] Pengyuan Lyu, Minghui Liao, Cong Yao, Wenhao Wu, and Xiang Bai. Mask TextSpotter: An End-to-End Trainable Neural Network for Spotting Text with Arbitrary Shapes. Proceedings of the European Conference on Computer Vision (2018)

[26]. [Wang et al, 2018] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2018), pp. 7794-7803

[27]. [Poupart, 2019] Pascal Poupart. Attention and Transformer Networks. Lecture 19 in CS480/680, 2019.

[28]. [Wu et al, 2019] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2

[29]. [Ye et al, 2020] Jian Ye, Zhe Chan, Juhua Liu, and Bo Du. TextFuseNet: Scene Text Detection with Richer Fused Features. Proceedings of the International Joint Conference on Artificial Intelligence (2020), pp. 516-522
[30]. [Zhouran et al, 2021] Shen Zhuoran, Zhang Mingyuan, Zhao Haiyu, Yi Shuai, and Li Hongsheng. Efficient Attention: Attention with Linear Complexities. Proceedings of the IEEE Winter Conference on Applications of Computer Vision (2021)

Hitesh Hinduja, et. al. "Revisiting Text Fuse Net: Text Context Enhanced Attention Networks for Scene Text Localization." *IOSR Journal of Computer Engineering (IOSR-JCE)*, 25(1), 2023, pp. 37-49.