

It’s All About The Scale - Efficient Text Detection Using Adaptive Scaling

Elad Richardson*, Yaniv Azar, Or Avioz, Niv Geron, Tomer Ronen, Zach Avraham, Stav Shapiro
Penta-AI

Abstract

“Text can appear anywhere”. This property requires us to carefully process all the pixels in an image in order to accurately localize all text instances. In particular, for the more difficult task of localizing small text regions, many methods use an enlarged image or even several rescaled ones as their input. This significantly increases the processing time of the entire image and needlessly enlarges background regions. If we were to have a prior telling us the coarse location of text instances in the image and their approximate scale, we could have adaptively chosen which regions to process and how to rescale them, thus significantly reducing the processing time. To estimate this prior we propose a segmentation-based network with an additional “scale predictor”, an output channel that predicts the scale of each text segment. The network is applied on a scaled down image to efficiently approximate the desired prior, without processing all the pixels of the original image. The approximated prior is then used to create a compact image containing **only text regions**, resized to a **canonical scale**, which is fed again to the segmentation network for fine-grained detection. We show that our approach offers a powerful alternative to fixed scaling schemes, achieving an equivalent accuracy to larger input scales while processing far fewer pixels. Qualitative and quantitative results are presented on the ICDAR15 and ICDAR17 MLT benchmarks to validate our approach.

1. Introduction

Reading text from natural images is a long-standing problem in the field of computer vision. Usually, the problem involves two stages: (1) A text detection mechanism, whose purpose is to localize the individual words in the image, and (2) A text recognition mechanism, whose purpose is to take each detected text region and parse it into a single word.

When it comes to the detection stage, recent methods have made impressive leaps in terms of performance [8,

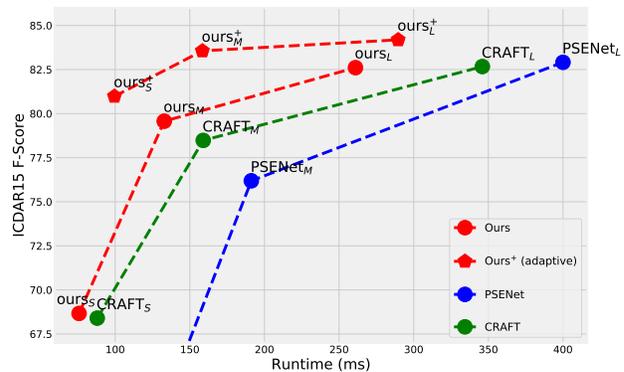


Figure 1: The “performance vs accuracy” trade-off. S , M and L denote an input with a long side of 720, 1024 and 1440 accordingly. Ours is the proposed single-scale method, while Ours⁺ is boosted using the proposed scaling scheme.

16, 15, 3, 40, 35, 1, 21, 17]. These methods can most often be classified into two distinct types. The first type is anchor-based approaches [8, 16, 15, 21, 17], which build upon popular object detection CNN architectures, such as SSD [20], Yolo [32], or Faster R-CNN [33], and directly predict a bounding box or quadrilateral around the text. While efficient, they are less suited for detecting rotated or irregular text. The second type is segmentation-based methods [3, 40, 35, 1], which usually predict, for each pixel, a text/no-text semantic mask from which bounding boxes are extracted using an additional post-processing stage. While this representation is more flexible, it struggles with small text instances which are close to one another and cannot be easily separated.

Indeed, one of the major difficulties in text detection in general, and specifically with the segmentation approaches, lies in detecting small text instances. While much effort has been put into the problem by creating better post-processing schemes [3, 35, 24], the problem of finding better scaling schemes is somewhat overlooked. Instead, most methods simply resort to fixed scaling schemes which are applied on top of their proposed baseline. That is, feeding the same image into the baseline network in an enlarged scale, or even

* Corresponding author. Email: elad.richardson@gmail.com

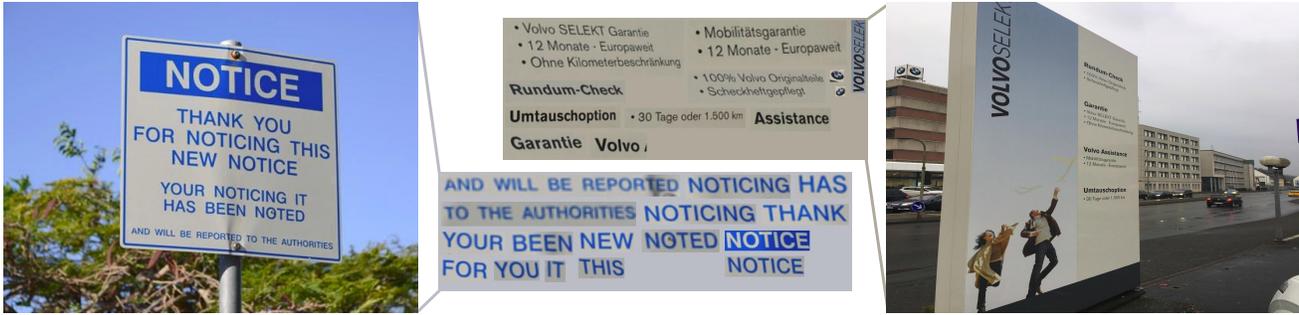


Figure 2: Canonical representations, “knapsacks”, created using the proposed approach on ICDAR17 images. In the left image, note how scaling is applied adaptively to create a uniform scale. In the right image, see how all background regions are removed before rescaling.

multiple ones. Although effective in terms of recall, these schemes are wasteful both in terms of runtime and memory. This can be attributed to two main factors:

1. In many cases, **text occupies only small regions of the original image**. Thus, when enlarging the image to capture small text instances we also redundantly process many more background pixels.
2. Enlarging the entire image changes the scale of all text regions, even though **many text regions might already be in an appropriate scale for detection** and do not need to be enlarged. Large text might even be harder to detect when further enlarged. To mitigate that some methods choose to run multiple fixed scales, but this also increases the processing cost.

In this paper, an approach is presented to tackle these problems. The core idea behind our approach is that localizing regions of text is much easier than localizing individual words. Hence, we propose to utilize a coarse forward evaluation on a downsized image to locate text regions while simultaneously approximating the scale of each such region. This information is used to create a compact representation containing **only text regions**, where each region is resized to a **canonical scale**, as shown in Figure 2. The compact representation can then be processed using a single forward pass, resulting in a much more efficient evaluation process.

In practice, this is achieved by taking a semantic segmentation method and adding an output channel that represents the height of each text instance. While this information is redundant when the detections are well separated, it is crucial when several text instances are merged. That is because the raw segmentation mask cannot tell us whether a text region contains a single large-scale line or several merged lines of smaller text. The height channel, however, allows us to easily retrieve the scale of each such region and scale it as needed, assuring that the text will be well separated in the

second pass. These scaled text regions are then packed together into a single image, or “knapsack”, that is fed again into the same segmentation network. The full process is shown in Figure 3.

To validate our approach we propose and implement a new semantic segmentation baseline, based on recent state-of-the-art approaches, which uses a simple and efficient post-processing scheme. Our scale channel is added to the proposed baseline to create the final network. The approach is validated under varying conditions and benchmarks, showing that our adaptive method is indeed a powerful alternative to fixed scaling schemes.

The main contributions of this paper are:

- A novel scheme for adaptively scaling text images resulting in a far more efficient process compared to fixed scaling schemes.
- An improved semantic segmentation approach for text detection requiring only a simple post-processing step.

2. Related Work

As mentioned above, current methods can be roughly divided to anchor-based approaches [8, 16, 15, 21, 26, 17] and segmentation-based ones [3, 40, 35, 9, 36, 39], where some recent methods try to fuse the two types together [14, 19, 10, 25]. Our proposed pipeline is based on recent segmentation-based text detection methods, which are discussed next. Details on other approaches which are not covered in this work are presented in [23].

Segmentation-based text detection approaches have gained significant attention in recent years, starting from the seminal works of Yao *et al.* [36] and Zhang *et al.* [39]. These works solve the problem of text detection by reformulating it as a semantic segmentation scheme, which is then solved by a Fully Convolutional Network (FCN) [22]. It was shown that these approaches are better suited for

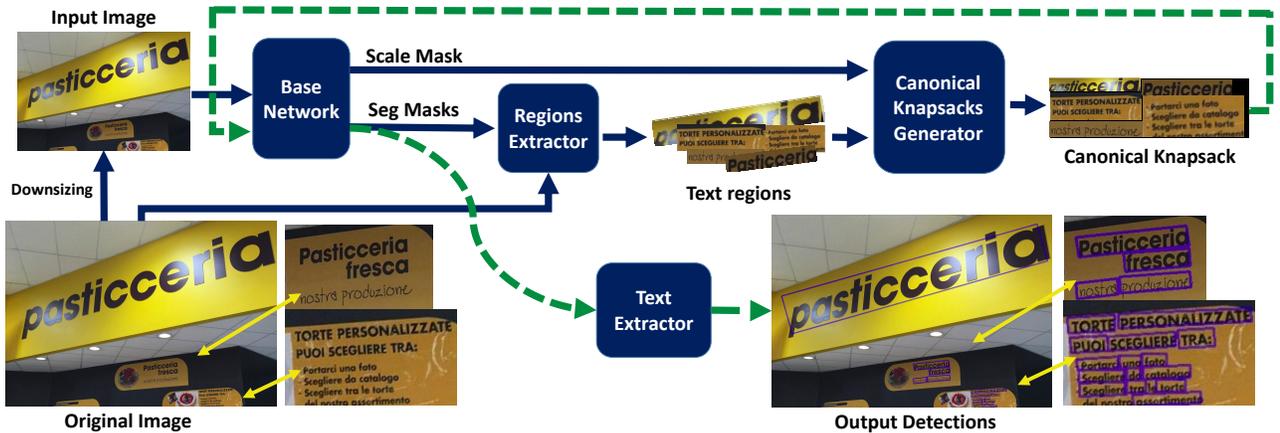


Figure 3: The proposed pipeline. First, a downsized image is fed into our base network to get initial segmentation and scale masks. These masks are then used to create a canonical knapsack, containing only text regions in a uniform scale. This knapsack is then fed again through the baseline network where the segmentation mask is then used to create the refined localization of text instances.

rotated and irregular text and interest in them has subsequently emerged. These methods, however, share a common problem where adjacent word instances tend to connect. This problem is inherent in nearly all segmentation based methods, and recent segmentation based approaches for text detection put a large emphasis on mitigating this problem [31, 3, 35, 1].

The WordFence approach [31] learns an additional border class to force a better separation of word instances. PixelLink [3] tries to predict, for each pixel in an 8-connected neighborhood, whether its neighbors belong to the same text label. The predicted connectivity maps, in addition to the original text/no-text segmentation map, are then used to generate the final detections. The recent PSENet method [35] learns a set of scaled kernels around each text instance, which, in test time, are progressively expanded to generate the complete word instance prediction. The CRAFT method [1] uses character affinity maps to connect character detections into a single word. While both PSENet and CRAFT achieve state-of-the-art results on several competitive benchmarks they require extremely large input images. For example, on the ICDAR15 benchmark [12], images are enlarged from 720×1280 to 1260×2240 , which significantly increases runtime and can present difficulties on platforms with limited resources. While some approaches have tried to apply a two-stage approach, in which rough text regions are first located [9, 38, 6], to the best of our knowledge we are the first to directly learn a text scale channel in order to build an optimized two-stage detection pipeline.

In contrast to previously mentioned works, research into adaptive scaling schemes for object and text instance de-

tection is far less prominent. In [2] it is shown that by learning a single optimal scale for each image, it is possible to improve both the accuracy and speed of object detection. Yuan *et al.* [37] learns scale-adaptive anchors to better handle multi-scale text using fewer anchors. The AutoFocus approach [28] predicts “FocusPixels”, regions which are likely to contain small objects, and applies the multi-scaling process only on these regions, resulting in improvements in terms of runtime and memory efficiency. None of these approaches, however, propose an adaptive scaling scheme specifically suited for text instance segmentation.

3. Proposed Approach

The idea behind our approach is to first use a fast forward pass, over a downsampled image, to predict general text regions and their respective scales. These scales are then used to resize all the text regions into a uniform compact representation which is then forwarded through the same neural network to separate the words.

More specifically, our approach is composed of five steps as can be seen in Figure 3. First, a single-scale detection network is applied to the given input image to detect general text regions and their scales. Secondly, regions are extracted using the segmentation mask. Thirdly, the information extracted from the first stage is used to create a compact knapsack containing only the regions of text, scaled to the same size as can be seen in Figure 2. Fourthly, the knapsack image is forwarded through the same single-scale detection network. Finally, a post-processing mechanism is used to extract the output image.



Figure 4: Training data. From left to right, the input image, the segmentation map, the shrunk map, and the scale map.

3.1. The Single-Scale Method

As shown in many recent works [3, 40, 35], using an architecture that predicts dense pixel-wise output maps grants us the flexibility to learn different forms of mappings, such as the Geometry Map of EAST [40] and the Connectivity Map of PixelLink [3]. Usually, the purpose behind these different representations is to allow effective extraction of well-separated bounding boxes from the output maps. While our adaptive scaling scheme can be used on top of many segmentation architectures, we chose to implement a new baseline as part of our approach. Figure 5 shows an overview of our single scale detection network. The backbone itself is discussed in Section 4.2.

Inspired by the recent work of PSENet [35] and by the shrunk polygons used in the EAST method [40] two output maps are learned. The first one is a simple text/no-text semantic map, while the second one is a shrunk map, where only the inner part of the polygon is classified as text, see Figure 4. The shrunk map is used for an improved distinction between close text instances. As in [35], the shrunk map is created using the Vatti clipping algorithm [34], with the numbers of pixels to clip, d , defined as

$$d = \frac{\text{Area}(P) \times (1 - r^2)}{\text{Perimeter}(P)}. \quad (1)$$

Here P is the initial polygon and the scale ratio, r , is set to 0.4. Similar to [35, 6], the segmentation channels are trained using the dice-loss, which is defined as

$$L(S, G) = 1 - \frac{2 \sum_{x,y} (S_{x,y} \times G_{x,y})}{\sum_{x,y} S_{x,y}^2 + \sum_{x,y} G_{x,y}^2}, \quad (2)$$

where S is the output segmentation map and G is the ground-truth map. As the shrunk map is incorporated into full text map, the loss is applied only on the regions inside the text map. This has the effect of breaking down the learning process into two sub-problems, detecting text regions and localizing words inside each such region. The final loss is set as

$$L_{\text{segment}} = 0.5 \cdot L_c + 0.5 \cdot L_s, \quad (3)$$

where L_c is the dice-loss applied segmentation channel and L_s is the dice-loss on the shrunk segmentation channel. On-line Hard Negative Mining is applied on L_c with a ratio of 3. During post-processing, rotated rectangles are extracted

directly from the shrunk map and expanded in accordance with the shrinking ratio. This results in a simple and efficient post-processing procedure. While the full segmentation map is not used to extract the text instances, it is helpful for better extraction of small text regions, as will be discussed below. Note that unlike [35] our method predicts only a single kernel map, thus avoiding the need for the expansion algorithm proposed there which is more helpful for irregular text shapes.

3.2. Introducing The Scale Channel

The proposed “scale predictor” is created by simply adding another output channel in the last convolutional layer, thus outputting a 3-channel image, containing both the segmentation masks and the scale. The scale of each word is defined by finding the bounding rotated rectangle and taking its height, or smaller axis, as the scale of the entire word. The resulting label is shown in Figure 4. We found that height is a good choice for the scale value, as it is not affected by the number of characters and is closely related to the font size and the spacing between words. For inference we take the average scale inside each segment, weighted by the confidence of the segmentation map, as its scale.

The mathematical formulation of our scale prediction loss draws inspiration from anchor-based object detection methods, specifically [4]. These methods perform bounding-box regression in order to transform default anchor boxes into tight object proposals, predicting 4 transformation parameters that are used for translation and scaling. The parameters are represented as log-space additive offsets which are equivalent to pixel-space multiplications. Having the scale represent multiplication suits our scenario very well, as it grants the same weight to different sized texts in the same image. That is, a 30-pixel high text that was predicted to be 15 pixels high would inflict the same

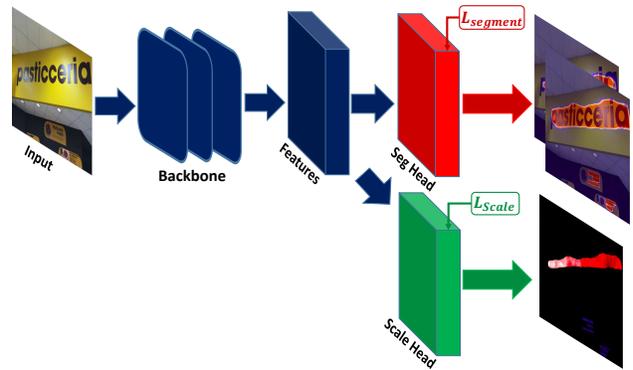


Figure 5: Proposed architecture. Image is fed through a convolutional backbone followed by segmentation and scale layers.

loss as a 300 pixel high text that was predicted to be 150-pixels high. A simpler choice of directly predicting the text height in pixels would result in a drastic over-weighting of large texts, as their additive pixel difference would be much bigger than small texts. This results in the following formulation

$$s_{i,j}^{\hat{}} = \log \left(\frac{s_{i,j}}{s_{ref}} \right), \quad (4)$$

where $s_{i,j}$ is the height of the word the pixel belongs to, and s_{ref} is set to 25. Finally, a Smooth-L1 loss is applied over all regions labeled as text, which means that we do not need to define scale values for background regions. The L_{scale} loss is defined as,

$$L_{scale}(\hat{s}, \hat{s}_{gt}) = \begin{cases} 0.5(\hat{s} - \hat{s}_{gt})^2, & \text{if } |\hat{s} - \hat{s}_{gt}| < 1 \\ |\hat{s} - \hat{s}_{gt}| - 0.5 & \text{otherwise} \end{cases}, \quad (5)$$

where \hat{s} is the normalized scale, and \hat{s}_{gt} is the ground truth normalized scale. L_{scale} is added to $L_{segment}$ for joint training of segmentation and scale,

$$L = L_{segment} + 0.1 \cdot L_{scale}. \quad (6)$$

The values for s_{ref} and the loss weights were set following empirical experiments.

3.3. Refined Inference

The first forward pass over the downsampled image is used to retrieve general text regions, without fine separation between small words, alongside their respective predicted scale. Every text region, or ‘‘blob’’, is then extracted from the original image and resized to the desired scale, which is set as $1.5s_{ref}$. To efficiently process the extracted blobs, all regions are packed together to create a compact knapsack representation. This is done using the Maximal Rectangles Best Short Side Fit algorithm [11]. The knapsacks are then passed through our network to create the refined

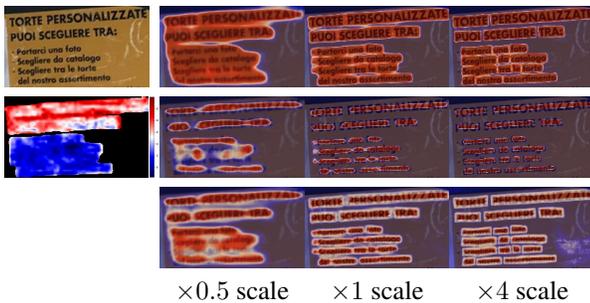


Figure 6: Segmentation outputs under different scales. Each column presents the segmentation map followed by the shrunk segmentation map and the average map. Input image and scale map are shown in the first column.

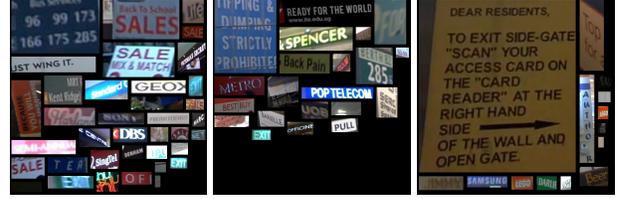


Figure 7: Artificial knapsacks created for augmentation.

segmentation result, from which the final rotated rectangles are extracted.

While text regions can be extracted directly from the shrunk segmentation map, using an averaged map of the two segmentation channels for the first stage results in better performance. This can be attributed to the fact that while the shrunk map gives better separation, it misses some of the smaller text regions, as shown in Figure 6.

Note that while the knapsack images create a compact representation, they are inherently different from the natural images used for training, which can cause the network results to degrade. To mitigate that we simply propose to add a Knapsack Augmentation to the training process. This is done by randomly taking regions of text from different images, resizing them to our reference scale, and packing them using our packing scheme. Some generated knapsacks are shown in Figure 7.

4. Experiments

Here, we evaluate the proposed baseline and adaptive scaling scheme and compare it to fixed scaling approaches. In all experiments, our single scale method is denoted as Ours, while our two-stage solution is denoted as Ours⁺. Methods are evaluated on three different long side scales 720, 1024 and 1440, which are denoted as S , M and L accordingly. Finally, Ours_L denotes that a method was used with an input image resized to scale L . Experiments are conducted on the ICDAR15 and ICDAR17 benchmarks.

4.1. Datasets

ICDAR15 The ICDAR15 Competition on Robust Reading [12] is a standard benchmark for detecting oriented text in-the-wild. The benchmark is composed of 1,500 images taken using a Google Glass sensor, where 1,000 images are used for training and 500 for testing. Text instances might be rotated and are tagged as quadrilaterals.

ICDAR17 MLT The ICDAR17 Competition on Multi-Lingual Scene Text Detection [29] is a large scale benchmark for text detection in multiple languages. The benchmark contains 7,200 training images and 9,000 test images taken from a diverse set of scenes. The benchmark is deemed challenging both due to the variability in text location and scale, and the need to recognize, and separate, words in different languages.

Method	Recall	Precision	F-Score	Forward	Process	FPS
PSENet _S	46.46%	72.01%	56.48%	66ms	35ms	9.9
PSENet _L	80.79%	83.65%	82.19%	247ms	149ms	2.5
CRAFT _S	60.13%	79.30%	68.40%	77ms	11ms	11
CRAFT _L	80.50%	84.96%	82.67%	301ms	45ms	2.9
Ours _S	58.97%	82.16%	68.67%	66ms	10ms	13
Ours _S ⁺	78.52%	83.60%	80.98%	82ms	16ms	10
Ours _L	80.60%	84.72%	82.61%	244ms	17ms	3.8
Ours _L ⁺	83.05%	85.35%	84.19%	262ms	27ms	3.5

Table 1: Results on the ICDAR15 Benchmark. “Forward” is the GPU runtime, while “Process” includes all the processing done on CPU.

4.2. Implementation Details

For our backbone, we investigate both an FPN module [18], which is a widely used architecture for semantic segmentation, and the ESPNet architecture from [27]. For the FPN module, a pretrained ResNet-50 [7] backbone is used, with an additional feature fusion mechanism, as in [35]. Compared to the lightweight ESPNet, FPN is relatively heavy in terms of runtime and memory efficiency, a distinction that can help understand the effect of the underlying architecture on our method.

For training data, we use both the ICDAR15 and ICDAR17 MLT training images, where both datasets are balanced during training so that every batch is approximately evenly split. A standard augmentation pipeline is used during training, composed of the following steps (1) A photometric distortion process as in [20] (2) Aspect ratio distortion, where the height is scaled by a uniform random factor in the range of [0.6, 1.4] (3) Random scaling by a factor of {0.5, 1, 2, 3} (4) Rotation by an angle between -10° and 10° (5) Random cropping of 640×640 pixels around a labeled text instance, and finally, (6) mirroring is randomly applied with a probability of 0.3, where the cropped image padded as needed. The FPN module is trained with a batch size of 12 using stochastic gradient descent (SGD) with weight decay of $5 \cdot 10^{-4}$ and Nesterov momentum of 0.99. The network is trained for $180 \cdot 10^3$ iterations where the initial learning rate is $1 \cdot 10^{-3}$ and is decayed by a factor of 0.1 every $60 \cdot 10^3$ iterations. ESPNet is trained with a batch size of 16 using the ADAM [13] solver, where the initial learning rate is $1 \cdot 10^{-3}$ and is decayed by a factor of 0.94 every $10 \cdot 10^3$ steps. Training was performed on 4 NVIDIA M60 GPUs, where a single one was used for evaluation.

4.3. Benchmark Results

The proposed approach is evaluated on the ICDAR15 and 17 benchmarks alongside PSENet [35] and CRAFT [1].

Results on PSENet and CRAFT were produced using the official implementations, but with the same input scale as our method, where their hyperparameters were reselected for these scales. This allows us to accurately compare the performance of the method itself decoupled from its input size. Note that all experiments were run on the same NVIDIA M60 machine using Pytorch [30].

ICDAR15 Results Table 1 presents the results on the ICDAR15 benchmark. One can see that, when using the same fixed scales, our single scale method achieves comparable results to state-of-the-art segmentation methods. It is also clear that the *L* scale is consistently more accurate but is also significantly less efficient. Next, looking at the proposed adaptive scaling scheme, Ours_S⁺, one can see that indeed our method is able to stay close to Ours_S in terms of runtime while producing results of much larger scales. This shows that **our method is able to somewhat mitigate the inherent trade-off between runtime and accuracy**. Note that Figure 1 shows these results as a “runtime vs F-Score” graph for better visualization. Qualitative results are presented in Figure 8, showing both the output of Ours_S and Ours_S⁺ alongside the generated knapsacks. This also shows the efficiency of using a compact representation compared to the original image.

ICDAR17 Results Table 2 shows that the results on the challenging ICDAR17 benchmark. Our method uses the same settings, but with a knapsack scale of $1.8s_{ref}$. One can see that CRAFT, which was pretrained on the SynthText dataset [5], is stronger than our baseline in the multi-lingual scenario. Still, our adaptive scheme proves successful. That is Ours_S⁺ is significantly more accurate than Ours_S, getting close to the performance of Ours_L while staying twice as fast. Note that both methods do not reach state-of-the-art results in these configurations, which usually require extremely large scales. PSENet originally rescales each image by a factor of two, while the CRAFT method uses a long side of 2560 pixels, resulting in significantly higher processing time. Quantitative results on ICDAR17 are shown in Figure 10.

Method	Recall	Precision	F-Score	Forward	Process	FPS
CRAFT _S	42.56%	74.46%	54.16%	91ms	16ms	9.3
CRAFT _L	61.68%	78.80%	69.20%	351ms	65ms	2.4
Ours _S	39.36%	73.10%	51.17%	83ms	10ms	10.9
Ours _S ⁺	51.93%	76.27%	61.79%	106ms	27ms	7.48
Ours _L	56.77%	74.21%	64.33%	311ms	35ms	2.88
Ours _L ⁺	61.82%	72.94%	66.92%	329ms	64ms	2.54

Table 2: Results on the ICDAR17 benchmark.

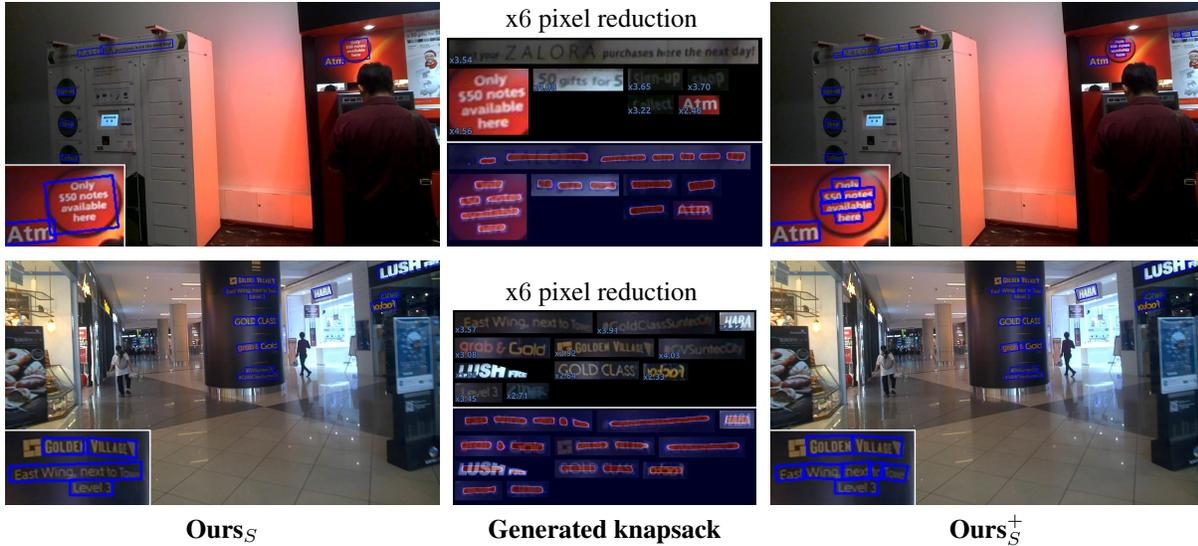


Figure 8: Results on the ICDAR15 benchmark. Scaling factor on every region is shown with respect to the input size for $Ours_S$. The pixel reduction is calculated as the ratio between the original image size and the created knapsack.

Backbone ₁	Backbone ₂	Recall	Precision	F-Score	FPS
FPN	—	58.98%	82.16%	68.66%	13.13
ESPNet	—	46.17%	74.17%	56.91%	43.68
FPN	FPN	78.52%	83.60%	80.98%	10.05
ESPNet	ESPNet	66.49%	78.64%	72.06%	28.09
FPN	ESPNet	72.17%	78.93%	75.40%	11.01
ESPNet	FPN	70.72%	84.57%	77.03%	23.62

Table 3: Backbone hybrids. Different combinations of the FPN and ESPNet backbones on the ICDAR15 benchmark are evaluated. Backbone₁ is the backbone used for the initial segmentation and Backbone₂ is the backbone used over the knapsacks.

4.4. Additional Analysis

On choosing the backbone We now turn to investigate how using different backbone architectures affects our results by comparing the behavior of ESPNet and FPN when used in the first or second segmentation stages. Table 3 shows the results of our method in the S scale over the ICDAR15 benchmark where different backbones were used. One can see that while FPN is indeed more accurate, it is significantly slower than the ESPNet architecture and that both methods benefit from our adaptive solution. In the rest of our experiments we chose to use FPN as our baseline due to its increased accuracy. An interesting configuration is the usage of ESPNet for the first stage, which requires only coarse segmentation, and FPN in the second stage for the refined localization. This composition is almost twice as fast than the single-stage FPN while also increasing the F-Score by 8%.

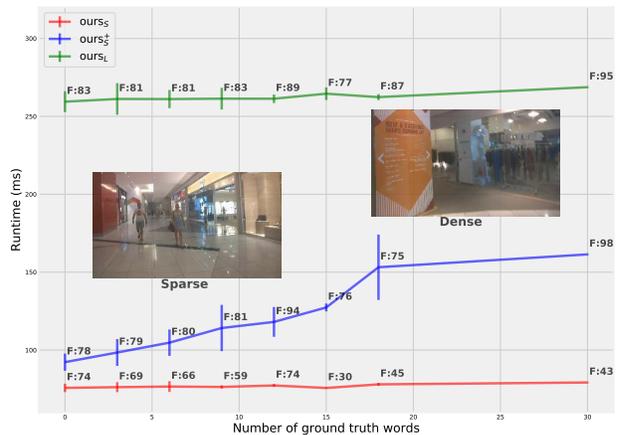


Figure 9: Runtime by number of words in ICDAR15. Average F-Score is shown for each point. Sample images for each region are also shown.

On text sparsity and runtime One of the interesting properties of the proposed method is its adaptive runtime. Where fixed scaling schemes would use approximately the same runtime for different images, our method adaptively changes the processing time according to the amount of text in the image. Figure 9 shows the average processing time as a function of the number of words in the image, on the ICDAR15 benchmark. As expected one can see that while $Ours_S$ is constantly fast and $Ours_L$ is constantly slower, $Ours_S^+$ processes sparse images with almost no overhead while staying faster than $Ours_L$ even on the more dense images.

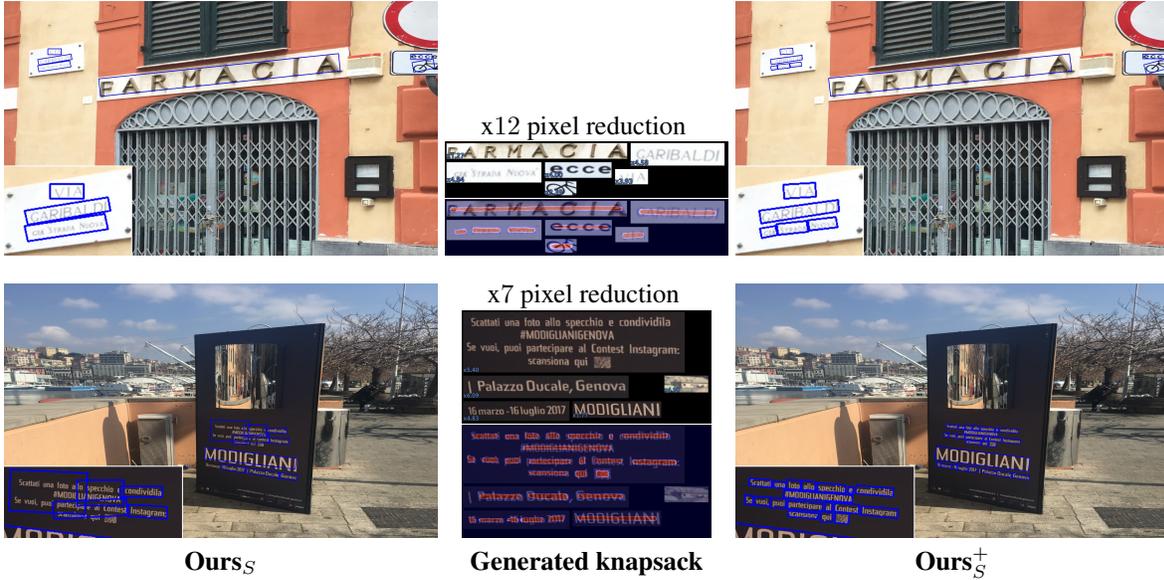


Figure 10: Results on the ICDAR17 MLT benchmark. Some images were cropped for better visualization.

On processed pixels While previous figures analyze our results in terms of runtime, another interesting evaluation is the number of input pixels fed to the network. While correlated with runtime, this analysis brings us another perspective as it is independent on the backbone and hardware used by the method. In Figure 11 we show the overall input area fed into the network of our different configurations alongside state-of-the-art methods, where the best reported single scale configuration is shown. Note how our adaptive method can significantly increase the F-score with only a small amount of extra area from the second pass. This can be attributed to the fact that indeed the knapsacks are significantly smaller than the original input image. One can see that our $Ours_S^+$ configuration is close to those reported by PSENet and CRAFT on 1260×2240 input images while processing fewer pixels. Interestingly, recent methods such as the Pyramid Mask Text Detector [19] can get impressive results while processing 1080×1920 images. We believe that our adaptive scaling could be a powerful extension for these methods as well.

5. Limitations

As shown in our experiments, our approach achieves results that are on-par with larger scales while reducing the runtime. Still, we note that there are some limitations to the proposed approach. Mainly, when choosing an extremely small scale for the first stage, the baseline might fail to segment some of the smaller text regions, resulting in them not appearing in the final result. Furthermore our method builds upon the fact that text is usually sparse, for dense images such as documents or newspapers our method might not be as effective.

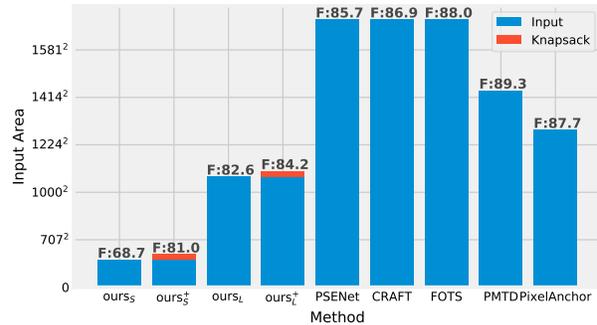


Figure 11: Input pixels for ICDAR15. For each method, the amount of pixels fed into the network is shown, for our method we include the area of compact representation as well. F-scores are shown on top of each bar according to the reported results of PSENet [35], CRAFT [1], FOTS [21], PMTD [19] and Pixel-Anchor [14].

6. Conclusion

We presented a novel adaptive scaling scheme for efficient text detection. Our approach uses a semantic segmentation network to detect coarse text regions while simultaneously predicting their scale. This information is then used to create a compact representation containing only the scaled text regions, from which refined word instances are extracted using an additional segmentation stage. Our approach is shown to be a powerful alternative to fixed scaling schemes, achieving the accuracy of larger scales in a more efficient manner.

Acknowledgments

The authors would like to thank Mr. Alon Palombo for his support and insights during the early stages of this research.

References

- [1] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee. Character region awareness for text detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9365–9374, 2019.
- [2] T.-W. Chin, R. Ding, and D. Marculescu. Adascale: Towards real-time video object detection using adaptive scaling. *arXiv preprint arXiv:1902.02910*, 2019.
- [3] D. Deng, H. Liu, X. Li, and D. Cai. Pixellink: Detecting scene text via instance segmentation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [4] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014.
- [5] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2315–2324, 2016.
- [6] D. He, X. Yang, D. Kifer, and L. Giles. Textcontournet: a flexible and effective framework for improving scene text detection architecture with a multi-task cascade. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 676–685. IEEE, 2019.
- [7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [8] P. He, W. Huang, T. He, Q. Zhu, Y. Qiao, and X. Li. Single shot text detector with regional attention. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3047–3055, 2017.
- [9] T. He, W. Huang, Y. Qiao, and J. Yao. Accurate text localization in natural image with cascaded convolutional text network. *arXiv preprint arXiv:1603.09423*, 2016.
- [10] Z. Huang, Z. Zhong, L. Sun, and Q. Huo. Mask r-cnn with pyramid attention network for scene text detection. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 764–772. IEEE, 2019.
- [11] J. Jylänki. A thousand ways to pack the bin—a practical approach to two-dimensional rectangle bin packing. 2010.
- [12] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, et al. Icdar 2015 competition on robust reading. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1156–1160. IEEE, 2015.
- [13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [14] Y. Li, Y. Yu, Z. Li, Y. Lin, M. Xu, J. Li, and X. Zhou. Pixel-anchor: A fast oriented scene text detector with combined networks. *arXiv preprint arXiv:1811.07432*, 2018.
- [15] M. Liao, B. Shi, and X. Bai. Textboxes++: A single-shot oriented scene text detector. *IEEE transactions on image processing*, 27(8):3676–3690, 2018.
- [16] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu. Textboxes: A fast text detector with a single deep neural network. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
- [17] M. Liao, Z. Zhu, B. Shi, G.-s. Xia, and X. Bai. Rotation-sensitive regression for oriented scene text detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5909–5918, 2018.
- [18] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.
- [19] J. Liu, X. Liu, J. Sheng, D. Liang, X. Li, and Q. Liu. Pyramid mask text detector. *arXiv preprint arXiv:1903.11800*, 2019.
- [20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.
- [21] X. Liu, D. Liang, S. Yan, D. Chen, Y. Qiao, and J. Yan. Fots: Fast oriented text spotting with a unified network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5676–5685, 2018.
- [22] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.
- [23] S. Long, X. He, and C. Ya. Scene text detection and recognition: The deep learning era. *arXiv preprint arXiv:1811.04256*, 2018.
- [24] S. Long, J. Ruan, W. Zhang, X. He, W. Wu, and C. Yao. Textsnake: A flexible representation for detecting text of arbitrary shapes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 20–36, 2018.
- [25] P. Lyu, M. Liao, C. Yao, W. Wu, and X. Bai. Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–83, 2018.
- [26] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, and X. Xue. Arbitrary-oriented scene text detection via rotation proposals. *IEEE Transactions on Multimedia*, 20(11):3111–3122, 2018.
- [27] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 552–568, 2018.
- [28] M. Najibi, B. Singh, and L. S. Davis. Autofocus: Efficient multi-scale inference. *arXiv preprint arXiv:1812.01600*, 2018.
- [29] N. Nayef, F. Yin, I. Bizid, H. Choi, Y. Feng, D. Karatzas, Z. Luo, U. Pal, C. Rigaud, J. Chazalon, et al. Icdar2017 robust reading challenge on multi-lingual scene text detection and script identification-trc-mlt. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1454–1459. IEEE, 2017.
- [30] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

- [31] A. Polzounov, A. Ablavatski, S. Escalera, S. Lu, and J. Cai. Wordfence: Text detection in natural images with border awareness. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 1222–1226. IEEE, 2017.
- [32] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.
- [33] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
- [34] B. R. Vati. A generic solution to polygon clipping. *Communications of the ACM*, 35(7):56–64, 1992.
- [35] W. Wang, E. Xie, X. Li, W. Hou, T. Lu, G. Yu, and S. Shao. Shape robust text detection with progressive scale expansion network. *arXiv preprint arXiv:1903.12473*, 2019.
- [36] C. Yao, X. Bai, N. Sang, X. Zhou, S. Zhou, and Z. Cao. Scene text detection via holistic, multi-channel prediction. *arXiv preprint arXiv:1606.09002*, 2016.
- [37] Q. Yuan, B. Zhang, H. Li, Z. Wang, and Z. Luo. A single shot text detector with scale-adaptive anchors. *arXiv preprint arXiv:1807.01884*, 2018.
- [38] X. Yue, Z. Kuang, Z. Zhang, Z. Chen, P. He, Y. Qiao, and W. Zhang. Boosting up scene text detectors with guided cnn. *arXiv preprint arXiv:1805.04132*, 2018.
- [39] Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu, and X. Bai. Multi-oriented text detection with fully convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4159–4167, 2016.
- [40] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. East: an efficient and accurate scene text detector. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 5551–5560, 2017.