Towards Weakly-Supervised Text Spotting using a Multi-Task Transformer

Supplementary Materials

The following sections provide additional explanations regarding our method. In Sec. A we emphasize the novel aspects of our architecture design. Sec. B portrays a visualization of the self-attention mechanism that is used in our system. Sec. C shows the model performance on random strings and Sec. D contains further details about the recognition heads and the training time.

A. Architecture Comparison

TextTranSpotter (TTS) suggests a text spotting architecture design which is different from existing approaches. Fig. 1 shows a comparison between common text spotting architectures and TTS. The key difference is the use of the joint query embeddings which allows sharing most of the weights between the recognition and detection tasks, and changes the way the recognition and detection heads operate. In most text spotting architectures [1–4] only the convolutional backbone is joint, meaning most of the computation is done in each head separately. In TTS, the transformer encoder-decoder is also shared, **such that about 95% of the model weights are shared between the two tasks**. The detection head in our approach consists of only a few linear layers and has about 1% of all of the model's parameters.

In most previous approaches, the input to the different heads is a spatial feature map describing the image. This feature map is then cropped using a pooling operation and fed into the recognition head. During training, most meth-

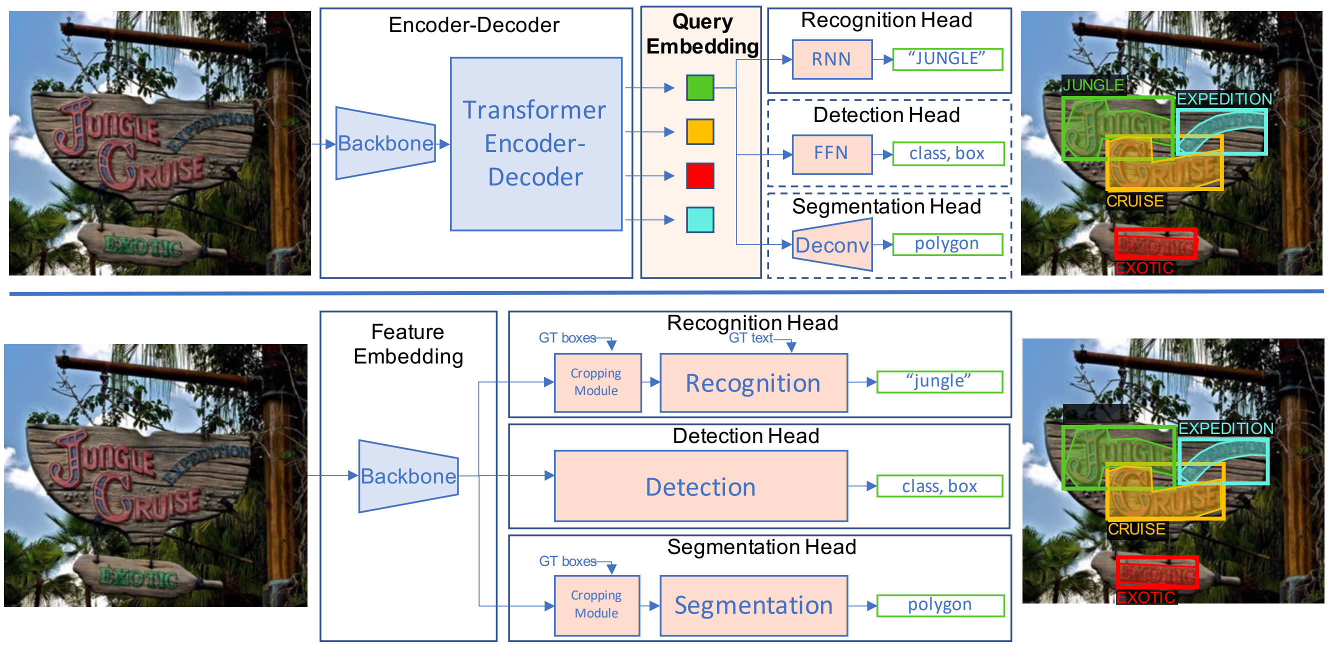

Figure 1. **TextTranSpotter architecture compared to common approaches.** An overview of TTS (top) compared to common text spotting approaches (bottom). The joint components in each architecture are colored in blue and the separate components for each head are colored in orange. In TTS, the main component of the model, the transformer encoder-decoder is shared between the recognition and detection tasks while in most approaches the main components are in the separate heads.

ods use the ground truth bounding boxes for the pooling. During inference, the heads operate sequentially so that the predictions of the detection head are used to perform the pooling before the recognition head. In contrast, in TTS the input to both heads is a one-dimensional feature vector for each query describing its content, including its location and transcription. This enables the recognition and detection to work simultaneously and optimize the query embedding for both tasks. Together with the Text Hungarian Loss described in the paper in Sec. 3.2, it allows training the model without using the ground truth bounding boxes as input to the recognition head.

B. Deformable Attention

One of the key features of our architecture is the ability to read text without any explicit spatial information regarding its location in the image, such as masks or bounding boxes. As opposed to existing two-stage methods, in which an accurate RoIPooling mechanism is crucial for the system to read rotated and curved text, TTS can handle such cases by design. The deformable attention mechanism, which is the main building block of the deformable transformer [5], is able to adjust the structure of its interest area dynamically, ignoring irrelevant or noisy parts of the image, and is therefore invariant to the text orientation and curvature. Since the deformable transformer is shared and optimized for both detection and recognition heads, it learns attention kernels which are meaningful for both tasks.

Fig. 2. shows an illustration of the cross-attention map of the decoder. We draw all six layers of the decoder, with 8 self-attention heads each, on the same image, where each head contains 4 sampling points. The attention weight of each sampling point is illustrated by its color. Even though the bounding-box contains text other than the desired word, the attention maps focus on the the correct word, allowing the recognition head to correctly predict its transcription.

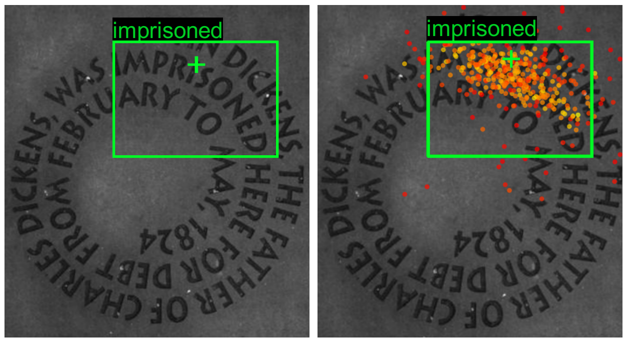

Figure 2. **Deformable attention.** Left: Prediction example in which the bounding box contains irrelevant text. Right: Deformable attention map of the decoder layers. Each sampling point is marked as a filled circle whose color indicates its corresponding attention weight. The reference point is shown as green cross marker, and the predicted bounding box is shown as a green rectangle. The attention clearly segments the word "imprisoned" from rest of the text located in the bounding box.

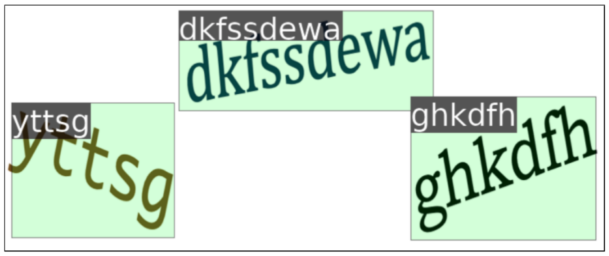

Figure 3. **TextTranSpotter predictions on random strings.** TextTranSpotter is able to predict random strings correctly, even though it was not trained on such data.

C. Performance on Random Strings

In Figure 3 we show the predictions of the model on random strings. Even though such cases are not common in the dataset, TextTranSpotter is able to predict them correctly. In the paper, we also presented cases of out of vocabulary words ("TSINGHUA", Figure 5 in the paper). These cases show that the model does not memorize the training set vocabulary and is able to generalize to unseen strings.

D. Architecture and Training Details

D.1. Recognition Heads

The ablation study included a comparison between two possible recognition heads: an RNN and an MLP (linear) head. The results were presented in Table 5. Here we include an explanation about each option.

The RNN head consists of: (1) an LSTM layer mapping the 1D query embedding into a f_{char} tensor at each step (2) character-level layer, mapping each f_{char} feature into an alphabet-sized softmax. The LSTM receives as input a one-hot encoding of the previous character predicted.

The linear head consists of: (1) word-level layer, mapping the 1D query embedding into a $L_{word} \times f_{char}$ tensor, and (2) character-level layer, mapping each f_{char} feature into an alphabet-sized softmax.

D.2. Training Time

Training was done on 8 A100-SXM4 GPUs. Synthtext pretrain, fully-supervised finetune, and weakly-supervised finetune took 4 days, 25 hours, and 10 hours, respectively.

References

[1] Xiaoxue Chen, Lianwen Jin, Yuanzhi Zhu, Canjie Luo, and Tianwei Wang. Text recognition in the wild: A survey. *ACM Computing Surveys (CSUR)*, 54(2):1–35, 2021. 1

[2] Minghui Liao, Guan Pang, Jing Huang, Tal Hassner, and Xiang Bai. Mask textspotter v3: Segmentation proposal network for robust scene text spotting. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 706–722. Springer, 2020. 1

[3] X. Liu, Ding Liang, Shihan Yan, D. Chen, Y. Qiao, and Junjie Yan. Fots: Fast oriented text spotting with a unified network. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5676–5685, 2018. 1

[4] Yuliang Liu, Chunhua Shen, Lianwen Jin, Tong He, Peng Chen, Chongyu Liu, and Hao Chen. Abcnet v2: Adaptive bezier-curve network for real-time end-to-end text spotting. *arXiv preprint arXiv:2105.03620*, 2021. 1

[5] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 2