Supplemental Material for Textual Alchemy: CoFormer for Scene Text Understanding

Gayatri Deshmukh
Independent Researcher
dgayatri9850@gmail.com

Onkar Susladkar
Independent Researcher
onkarsus13@gmail.com

Dhruv Makwana
Independent Researcher
dmakwana503@gmail.com

Sparsh Mittal
IIT Roorkee
sparsh.mittal@mfs.iitr.ac.in

Sai Chandra Teja R
Green PMU Semi Pvt Ltd
saichandrateja@gmail.com

SUPPLEMENTARY CONTENT

In this supplementary material, we present the following:

- We present five novel datasets: Textverse10M-E, Textverse10M-H, TST500k, CSTR2.5M and Akshara550. (Section S.1).
- We present the details of multi-headed channel attention (Section S.2).
- We present the details of decoder used for downstream tasks (Section S.3).
- We discuss the details of experimental platform (Section S.4).
- We present additional qualitative results (Section S.6).
- We present ablation results on STR task (Section S.7).

S.1. Our Proposed Five Datasets

In this paper, we propose and release five novel datasets that can be used for pre-training, scene text editing, and scene text recognition tasks. Out of the five datasets, four datasets (Textverse10M-E, Textverse10M-H, TST500K and CSTR2.5M) are created using synthetic data generation techniques, and one (Akshara550) contains real-world images. The datasets are diverse and contain a variety of backgrounds, text sizes, and font styles, making them suitable for various deep-learning tasks. Our datasets will be valuable resources for researchers working on scene text recognition, scene text editing, and related tasks. We now describe them.

Pre-training datasets: Textverse10M-E and Textverse10M-H: Textverse10M-E and Textverse10M-H are designed for pre-training deep learning models for English and Hindi scene text recognition, respectively. We used Synthtiger [11], a synthetic data generation tool, to generate 10 million images for each dataset. In the English dataset (Textverse10M-E), we used 109 fonts and a 1,20,000-word vocabulary, including lowercase, uppercase, and alphanumeric characters. Similarly, in the Hindi dataset (Textverse10M-H), we used 53 fonts and a 67,000 word vocabulary. The images generated in these datasets are diverse, and they contain a variety of backgrounds, text sizes, and font styles, making them suitable for pre-training deep learning models for scene text recognition tasks. Figure S.1 shows sample images from this dataset.

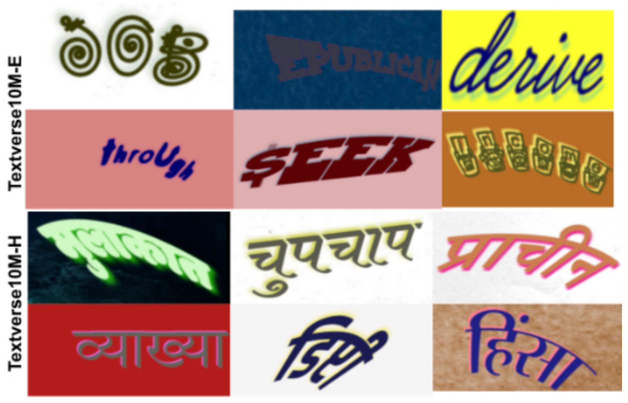

Figure S.1. Sample images from Textverse10M-E and Textverse10M-H datasets

Text Style Transfer 500k (TST500K): Text Style Transfer 500k (TST500K) is a dataset designed for bilingual text editing between two languages, for example, from English to Hindi. Figure S.2 shows sample images from this dataset. This is the first dataset available for scene text editing in 5 language pairs: English-to-Hindi, English-to-Chinese, English-to-Tamil, Hindi-to-English and English-to-Bengali. The dataset contains

100,000 images for each language pair, making a total of 500,000 images. To ensure the quality and diversity of the dataset, we used an 80k-10k-10k train-validation-test split for each language pair, resulting in a total split of 400k-50k-50k. As shown in Figure Figure S.2, each image in the TST500K dataset includes all the necessary information, including the input style image, target text image, background image, style image, and final style-transferred image. We used an image-magic site to determine the texture and size of the typefaces in the images, ensuring consistency and accuracy in the dataset.

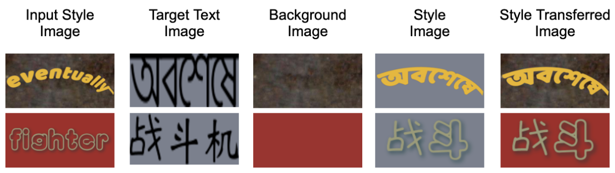

Figure S.2. Sample images from TST500K dataset

Complex Scene Text Recognition 2.5M (CSTR2.5M): The Complex Scene Text Recognition 2.5M (CSTR2.5M) dataset is designed to fine-tune and evaluate deep learning models for scene text recognition tasks. This dataset contains 2.5 million scene text images, each annotated with SynthTiger [11]. We used 187 distinct typefaces to generate this dataset. These typefaces are quite complicated, making it challenging for deep-learning models to recognize characters accurately. This dataset has 47 characters corresponding to 26 English alphabets, ten numerals, and 11 special characters. We split the dataset into 2 million images for training, 250 thousand for validation, and another 250 thousand for testing. This dataset is an excellent resource for researchers working on deep learning models for complex scene text recognition. Figure S.3 shows some images from this dataset.

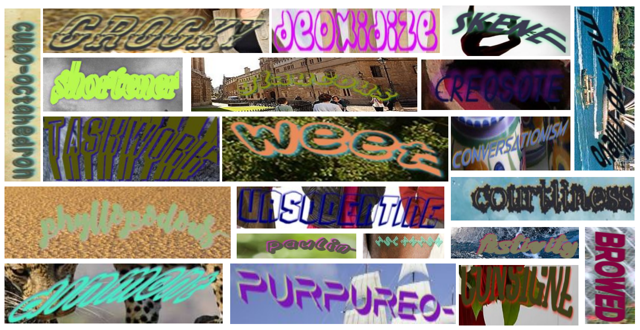

Figure S.3. Sample images from CSTR2.5M dataset

Aksharaa550 One of the main challenges of scene text recognition is dealing with variations in font styles, lighting conditions, and image distortions that can make it difficult to recognize text accurately in natural scenes. To address this challenge, we have developed the Akshara550 dataset, a new dataset of real-scene text images collected from the surrounding environment. The dataset contains a diverse range of text. To create the dataset, we used a smartphone camera to capture images of text in various settings, such as signs, posters, and book covers. We then manually selected and cropped text regions from the images and removed duplicates or low-quality images. The resulting dataset contains 557 high-quality images, each with a corresponding ground truth label indicating the text in the image. Our dataset provides a valuable resource for evaluating STR models, as it contains a diverse range of text styles and sizes representative of real-world text recognition scenarios. Figure S.4 shows sample images from Akshara550.

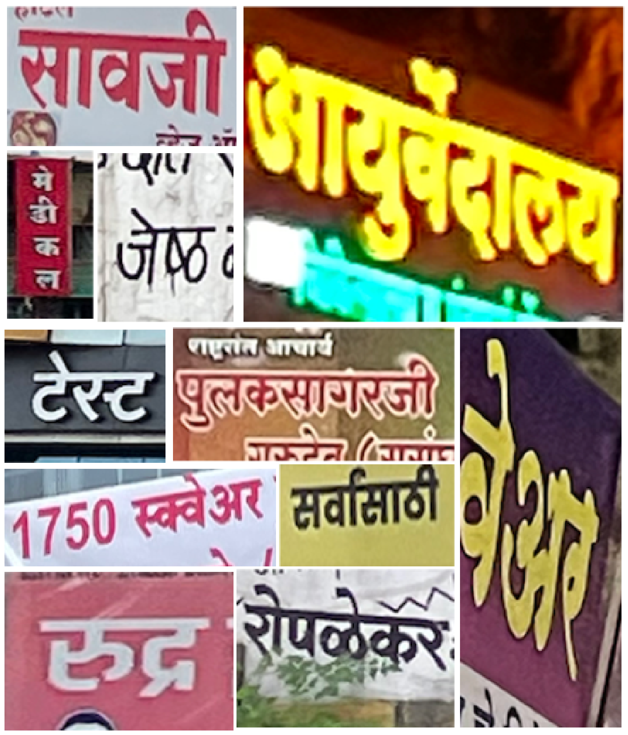

Figure S.4. Sample images from Akshara550 dataset

Tab. S.1 summarizes the characteristics of the proposed dataset.

S.2. Multi-headed Channel Attention (MCA)

The following content presents the mathematical operations involved in multi-headed channel attention (MCA). Figure S.5 shows the implementation of MCA code in the PyTorch framework.

1. Deriving learnable representations of key, query and value: Consider three tensors with dimensions $B \times C \times H \times W$, where B represents the batch size, and C, H,

Table S.1. Characteristics of our proposed five datasets

S.No.	Dataset Name	Dataset type	Task	Number of Images	Image Resolution	Train-test-validation split
1.	Scene Text Pretraining Text English (Textverse10M-E)	Synthetic	Pre-training	10 million	128X64, 256X128	None
2.	Scene Text Pretraining Text Devanagari (Textverse10M-H)	Synthetic	Pre-training	10 million	128X64, 256X128	None
3.	Text Style Transfer 500k (TST500K)	Synthetic	Scene Text Editing	500k	128X64, 256X128	400k-50k -50k
4.	Complex Scene Text Recognition 2.5M (CSTR2.5M)	Synthetic	Text Recognition	2.5 million	256X128, 512X256	2M-250k-250k
5.	Akshara550	Real	Text Recognition	557	Variable size	Only test (557)

```
1   # Let k q v be the output from c-blocks
2   # einsum is the pytorch function used to perform tensor operations
3   # b = batch_size, n = Number_heads, c = channels, h = hight, w = width
4   # here, ck, cq, cv means channels of key, query and value respectively
5   class MCA(nn.Module):
6     def _init_(self, num_heads, input_channels, ck, cq, cv):
7       super()._init_()
8       self.heads = num_heads
9       self.key = CBlock(input_channels, ck)
10      self.query = CBlock(input_channels, cq)
11      self.value = CBlock(input_channels, cv)
12
13    def forward(self, k, q, v):
14      k = self.key(k)
15      q = self.query(q)
16      v = self.value(v)
17
18      b, ck, hk, wk = k.shape
19      b, cq, hq, wq = q.shape
20      b, cv, hv, wv = v.shape
21
22      k = k.view(b, self.head, ck/self.head, hk, wk)
23      q = q.view(b, self.head, cq/self.head, hq, wq)
24      v = v.view(b, self.head, cv/self.head, hv, wv)
25
26      attn = torch.einsum('bnchw,bnkhw->bnkc', k, q)
27      attn = torch.softmax(attn, axis=2)
28      final_out = torch.einsum('bnkc,bnchw->bnkhw', attn, v)
29      out = final_out.view(b, -1, hv, wv)
30
31      return out
```

Figure S.5. Code snippet of proposed MCA

and W denote the number of channels, height, and width of the feature map, respectively. These three tensors are referred to as key, query, and value and are fed into the multi-headed channel attention (MCA) block. To derive a learnable representation for each input tensor, the tensors are passed through a C-Block, as shown in Figure 3(d) present in the main paper. The output of the C-Blocks provides learnable representations of the key, query, and value tensor, each with a shape of $B \times C \times H \times W$ (line no. 14-16 in Figure S.5).

2. Determining Attention Score: For the sake of clear exposition, we first explain the operation of a single head. In order to determine attention scores, a similar approach to that of conventional MSA [7] is used, starting with the multiplication of the key tensor with the query tensor. However, unlike conventional MSA, where the dot product is calculated between the key and query tensor, in the case of MCA, a batch-wise spatial multiplication is performed with the help of `einsum` operation (line no. 26 in Figure S.5), resulting in a tensor with a shape of $B \times C \times C$. Here, C represents the channel-wise covariance matrices for the key and query vectors.

3. Computing Softmax: After obtaining the resultant matrics through the previous step, we apply the softmax function along the last channel dimension of the vector to generate probabilistic attention scores (line no. 27 in Figure S.5). At this stage, the shape of the vector is $B \times C \times C$.

4. Multiplying scores with values: To obtain the weighted attention values, the output of softmax function (which is the attention score) is multiplied by its corresponding value tensor. This multiplication is performed along the first channel dimension, resulting in a tensor with a final shape of $B \times C \times H \times W$. This multiplication process is referred to as batch-wise channel multiplication, and it is performed using the `einsum` operation (line no. 28 in Figure S.5). The channel attention mechanism implemented in this step enables the network to focus selectively on important channels and suppress less significant ones, ultimately enhancing the feature representations' quality.

Implementing MCA using multiple heads: On using multiple heads, multiple sets of key, query, and value representations are generated. For each head, the steps from 1 to 4 are repeated. The output of each head is concatenated and passed to the subsequent layers of CoFormer blocks for further processing.

S.3. Decoder design

S.3.1. Architecture of each decoder

The decoder block shown in Figure S.6 uses three branches to process the input and apply channel and Fourier attention. We discovered that using channel attention in conjunction with Fourier attention enables the model to concentrate on all the essential characteristics. Channel attention helps the model to understand inter-dependencies across channels, while Fourier attention helps it to identify distinguishing characteristics. Together, they more effectively infer finer channel-wise attention. The decoder block

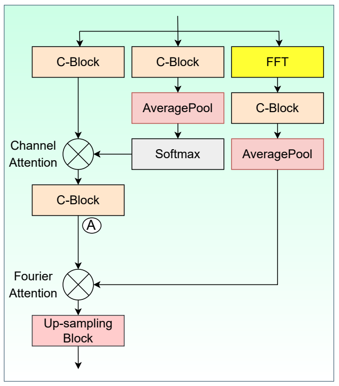

Figure S.6. Decoder block architecture

applies channel attention by calculating channel-specific attention probabilities from the second branch input, which is processed through a C-block, adaptive average pooling, and a softmax activation. The feature maps from the first branch are then multiplied by these attention probabilities to weigh their values. The Fourier attention in the decoder block is applied by processing the third branch input with FFT and a C-block, followed by adaptive average pooling to obtain a 1-D feature map. Later, the resultant 1-D feature map is multiplied by the feature map from location Ⓐ in Figure S.6. The final result is then up-sampled using transposed convolution, BN, and GeLU.

S.3.2. Overall decoder block design in LISTNet

LISTNet uses two decoder blocks; the architecture of each of these decoder blocks is only slightly different from that explained above. The decoder block design in LISTNet is shown in Figure S.7. Rather than using Fourier attention in the same decoder block, this approach uses cross-Fourier attention in the decoder. As seen in Figure S.7, the decoder block of Stream-Bg gets the feature map from position Ⓑ located in the Decoder block of Stream-ST. Similarly, the decoder block in Stream-ST gets the feature map from position Ⓐ in Stream-Bg. This lets information flow from one stream to the other. With the additional context provided by Stream-Bg, Stream-ST is better able to produce text that is stylistically consistent with the input style image (I_{IS}).

Also, in stage-2, the same decoder block shown in Figure Figure S.6 is utilized with one modification; here, we set kernel size to 1 in the transposed convolution of each decoder block. This means there is no up-sampling of the feature map, and the output has the exact spatial dimensions as the input.

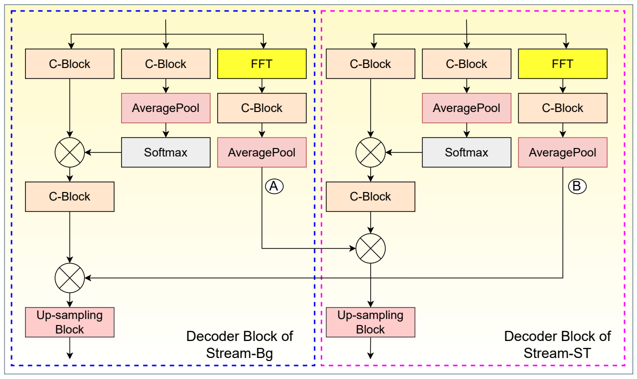

Figure S.7. Decoder block in LISTNet

S.4. Implementation platform and evaluation metrics

Implementation details of downstream tasks: In our experiments, we used PyTorch and CUDA 11.2 to train our models for Scene Text Recognition and Scene Text Editing. For all tasks, we initialized the learning rate at 0.0001 and used the Cosine Annealing Scheduler to decrease the learning rate gradually. We utilized the Adam optimizer for Scene Text Recognition. For training our Scene Text Recognition model, we used an NVIDIA A100 GPU and trained the model for 120,000 steps with a batch size of 256. For training our LISTNet (Scene Text Style Transfer model) model, we used the Adam optimizer for the generator and the RMSProp optimizer for the discriminator. We trained our model on two NVIDIA A5000 GPUs for 1,000,000 steps with a batch size of 128.

Implementation details of pre-training: Two models were pre-trained for pre-training tasks, one for Hindi scene-text images and the other for English scene-text images. Both models were trained using the same approach. Each model was trained for 1 million steps using the AdamW optimizer with a batch size of 128. The initial learning rate was set to 0.0001, and the Cosine Annealing scheduler was used to adapt the learning rate during training. To improve the models' performance and robustness, several data augmentation techniques were employed, such as elastic distortion, random rotation between -40 and $+40$ degrees, optical distortion, color jitter, random affine transformations, and others. These techniques were used to increase the models' ability to handle variations in the input data.

Evaluation metrics: For scene text editing, we use Peak Signal to Noise Ratio (PSNR), Structural SIMilarity (SSIM) and Learned Perceptual Image Patch Similarity (LPIPS) metrics. For text recognition tasks, we employ the word accuracy metric, which considers a prediction as correct if and only if all characters in all positions match. Thirty-six alphanumeric characters in lowercase are used for training and inference.

Parameter count and FLOP count: Table S.2 shows the number of parameters and FLOPs in various networks.

Table S.2. The number of parameters and FLOPs in various networks

Model	Parameters	FLOPs
VGG-16 [5]	138M	31G
Resnet-50 [2]	26M	7.7G
Resnet-152 [2]	60M	11G
Inception V3 [6]	23.5M	11.3G
ResNeXt-101 [10]	84M	14G
ConvNeXt-T [4]	29M	4.5G
ConvNeXt-S [4]	50M	8.7G
Swin-B [3]	88M	15.4G
ConvNeXt-B [4]	89M	15.4G
Swin-L [3]	197M	34.5G
ConvNeXt-L [4]	198M	34.4G
CvT-13 [9]	20M	4.53G
PVT-L [8]	61.4M	9.8G
CoFormer (ours)	48M	3.2G

S.5. Comparison of activation maps of different model

The activation maps produced for a given input image are shown in Figure S.8. Evidently, the previous models struggle to distinctly discern the boundaries of the scene text within the image, as opposed to the remaining elements present. The lack of exact text border identification considerably reduces the overall performance of these models. The LISTNet model, on the other hand, demonstrates a remarkable capacity to properly detect the boundaries of the text as well as interpret the underlying structure of the scene text. This is due to Fourier Transform and attention mechanism present in CoFormer. This combination of techniques not only enhances the LISTNet model's capability to accurately detect text boundaries but also empowers it to grasp the nuanced structural characteristics inherent in scene text. The Fourier Transform enriches the model's representation learning by capturing frequency-domain features, while the attention mechanism refines the model's focus on pertinent regions, thus contributing to its performance in text-related tasks.

S.6. Qualitative results

S.6.1. Attention maps

The attention map visualizations produced by the CoFormer model on scene text images obtained from the internet are shown in Figure S.9. These visualizations were derived from the fifth CoFormer block of the model and demonstrate the model's ability to focus on the relevant parts of an image, which is crucial for accurate scene text recognition.

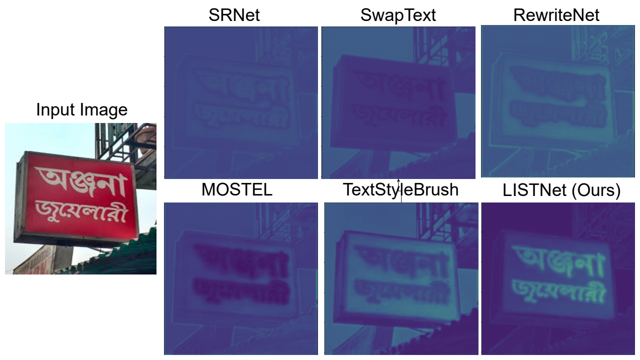

Figure S.8. Comparison of activation maps of different model

As shown in the first row of Figure S.9, the CoFormer model can focus on the smallest text present in the image, indicating its ability to identify and recognize text in complex scenes accurately. These attention maps provide valuable insights into the model's decision-making process and highlight its ability to attend to important regions of an image selectively.

S.6.2. t-SNE results

The t-SNE visualization on the CSTR2.5M dataset is presented in Figure S.10. Compared to the other models, our CoFormer architecture demonstrated superior performance in distinguishing between the 47 characters presented in the dataset. This is evidenced by the clear separation and clustering of character representations in the t-SNE plot. This confirms that our model can better capture and differentiate each character's unique features. These results demonstrate the effectiveness of our approach in improving the discriminative power of character embeddings and highlight the potential of CoFormer.

S.6.3. Qualitative results of text recognition

We present the text recognition results for samples from the CSTR2.5M dataset in Figure S.11. The ABINET and PARSeq-A models performed well on the relatively clear, horizontal, and high-resolution input images, accurately recognizing the text in these samples. In contrast, the CRNN model failed to recognize any characters in most images. Notably, the image with the word 'STATHAOORAPA' presented a significant challenge to all models due to its complex font and style. In fact, this word is not recognized correctly by any model. Overall, our technique demonstrated robustness to text orientation variability and the presence of different backgrounds, as evidenced by the successful recognition of text in most of the samples presented. These results highlight the effectiveness of our technique for text recognition tasks and demonstrate its potential as a valuable tool for real-world applications, particularly in scenarios where text may be presented in a variety

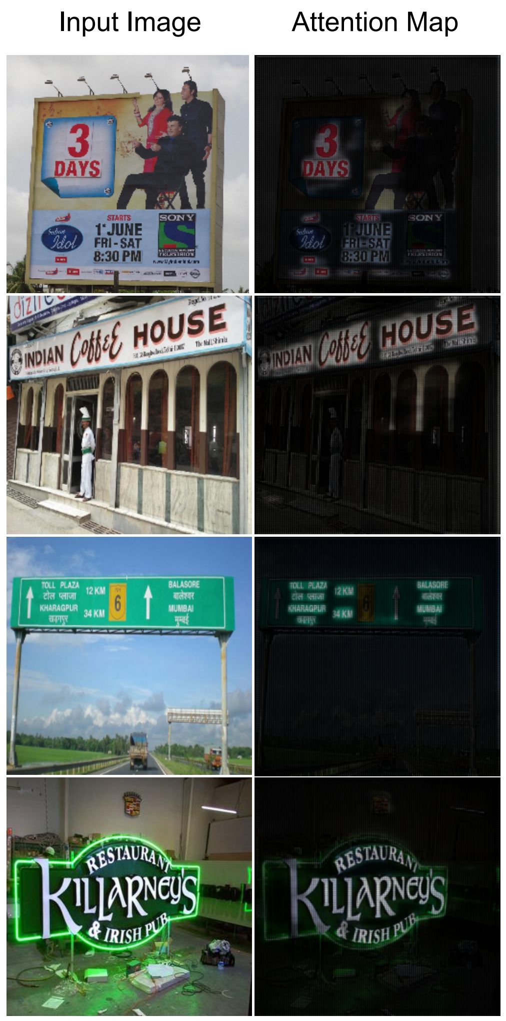

Figure S.9. CoFormer-generated attention maps on real-world images. The first column displays the real-world scene text images, while the second displays the attention map. Evidently, CoFormer has a strong comprehension ability and can pay attention to scene text in images.

of orientations and backgrounds.

S.6.4. Qualitative results on scene text editing

Scene text editing has important implications for various applications, including graphic design, advertising, and multimedia production. By transforming the style of the text in images, it can enhance the visual appeal and impact of the content, making it more engaging and memorable. Our results demonstrate the potential of LISTNet in these applications by transforming the text style of various types of style images to target images. We applied STE on three different types of images, including random images, PPT slides, and social media memes, in multiple languages.

STE on random images: Figure S.12 shows scene-text editing applied on random images. As shown in the figure, the model has successfully transformed the text style of the input style image, providing a different look and feel to the target text. The transformed text style is aligned with the intended style, enhancing the images' overall aesthetics.

STE on PPT slides: The Figure Figure S.13 shows the STE applied on PPT slides. In this case, the model has transformed the style of the text present on the slide to the target text, making it more visually appealing and engaging. The transformed text style is consistent with the slide's theme, which helps deliver the message more effectively. The application of STE on PPT slides has many advantages. It can improve the overall aesthetics of the slide and enhance its visual appeal, making it more engaging for the audience. In addition, STE can be useful for creating multilingual presentations. Using STE, the slides' text style can be transformed to match the target language, enabling the creation of presentations in multiple languages without the need for separate design efforts for each language. Visual results of our model LISTNet show its usefulness in this case.

STE on social media memes: Figure S.14 shows the STE applied on social media memes. In this case, the model has successfully transformed the text style of the meme, making it more humorous and relatable. The transformed text style matches the context of the meme, enhancing its overall impact.

Table S.3. STR ablation results.

Experiments	Word Accuracy
1. Without Fourier	84.03
2. Without Fourier and without pretraining	80.98
3. Replacing MCA with MSA	83.92
4. Different model size (Default uses 13 CoFormer blocks)	
4a. Four CoFormer Blocks	83.19
4b. Eight CoFormer Blocks	84.27
5. Different backbones (Default uses CoFormer)	
5a. ViT	83.98
5b. Swin Transformer	84.92
5c. PVT-L	82.47
5d. CvT-13	83.29

S.7. Ablation results on STR

Table S.3 shows the ablation results for STR task on ArT (Real images) [1] dataset in auto-regressive fashion. All the experiments are performed on pre-trained model.

(1,2) It is evident from the results that the removal of the Fourier Transform decreased accuracy by 2.54%. Also, the accuracy dropped by 5.89% when neither Fourier Transform nor pretraining was used, showing how important these parts are to the total performance of the model.

(3) Similar to above, the performance degrades when MSA block is used in place of MCA.

(4) We carried out experiments with varying block counts. Our default model uses 13 CoFormer blocks. A

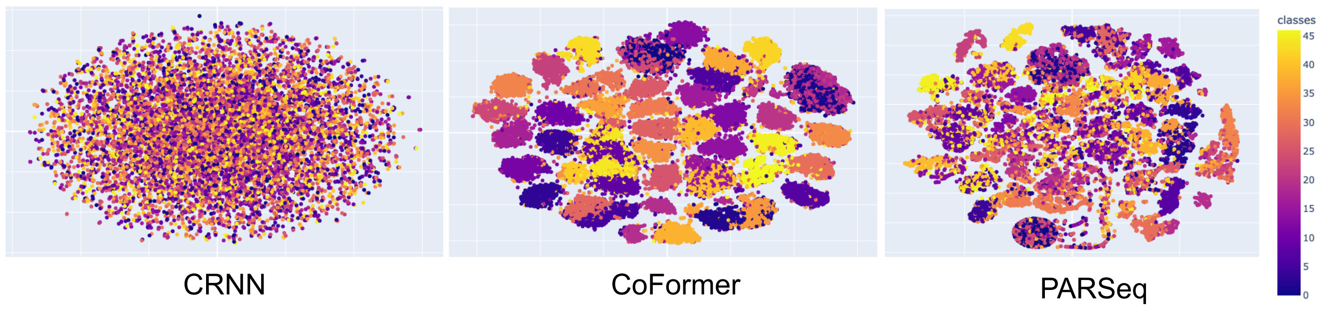

Figure S.10. t-SNE representation of the last layer activations of a network before applying softmax for the CSTR2.5M dataset. The classes of these data points are shown on the right-hand side, with shades of color ranging from 0 to 25 representing the English alphabet, shades of color ranging from 25 to 33 representing numbers, and shades of color ranging from 34 to 47 representing various special symbols.

Images						
GT	Weet	TWINNING	BIUREIDE	deoxidize	HADHBUIP	STATHAOORAPA
CRNN	weotc	TuHHLUOI	DIUBELDE	dcoxiliZe	ADHuH	NNNOOOO
VITSTR-S	Weet	TUHHHLHU	BRUREIBE	deoxilize	tnBBBuIK	STATNSSOAOPH
TRBA	weet	TWHNNHMG	DUKEiBE	deoxldize	HABNBUiP	STOTOAOROPH
ABINET	weet	TWINHHNU	BIUKBiBC	deoXidize	KHAHDUIP	STAHOOROPH
PARSeq-A	Weet	TWINNHNG	DIUREIDE	deoxidize	HADDBuIP	STAHAROORAPH
CoFormer	Weet	TWINNING	BIUREIDE	deoxidize	HADDBUIF	STAHAROORAPH
Images						
GT	POLYONYN	enacTmenT	infringer	LInsTOCK	shay	bewreck
CRNN	PolyCNUN	endomnenT	infnnoer	LIosIOOR	8pall	heurcclc
VITSTR-S	POCyOWN	enacTnneT	infrnoger	LInSTOCR	slyay	berreek
TRBA	POLWCWN	enacTMenT	infringer	LInsTOCR	shall	buuweck
ABINET	POLyCNYN	enacHmenT	infrmnger	LNSTOCR	shay	beureck
PARSeq-A	POLYCNYN	enacTmmnT	infringer	LInSToCK	shay	beuureck
CoFormer	POLYONYN	enacTmenT	infringer	LInSTOCK	shay	bewrcck

Figure S.11. Comparison of SOTA models for text recognition on CSTR2.5M

model containing four CoFormer blocks, in contrast, had an accuracy of 83.19, while eight blocks produced an accuracy of 84.27. The accuracy of the default mode was 86.57 percent. These results illustrate the importance of architectural depth.

(5) We test the STR model with a transformer architecture other than CoFormer. Based on resuts, it is clear that CoFormer is an improvement over other transformer backbones.

References

[1] Chee Kheng Chng, Yuliang Liu, Yipeng Sun, Chun Chet Ng, Canjie Luo, Zihan Ni, ChuanMing Fang, Shuaitao Zhang, Junyu Han, Errui Ding, et al. Icdar2019 robust reading challenge on arbitrary-shaped text-rrc-art. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1571–1576. IEEE, 2019. 6

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[3] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 5

[4] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Com-

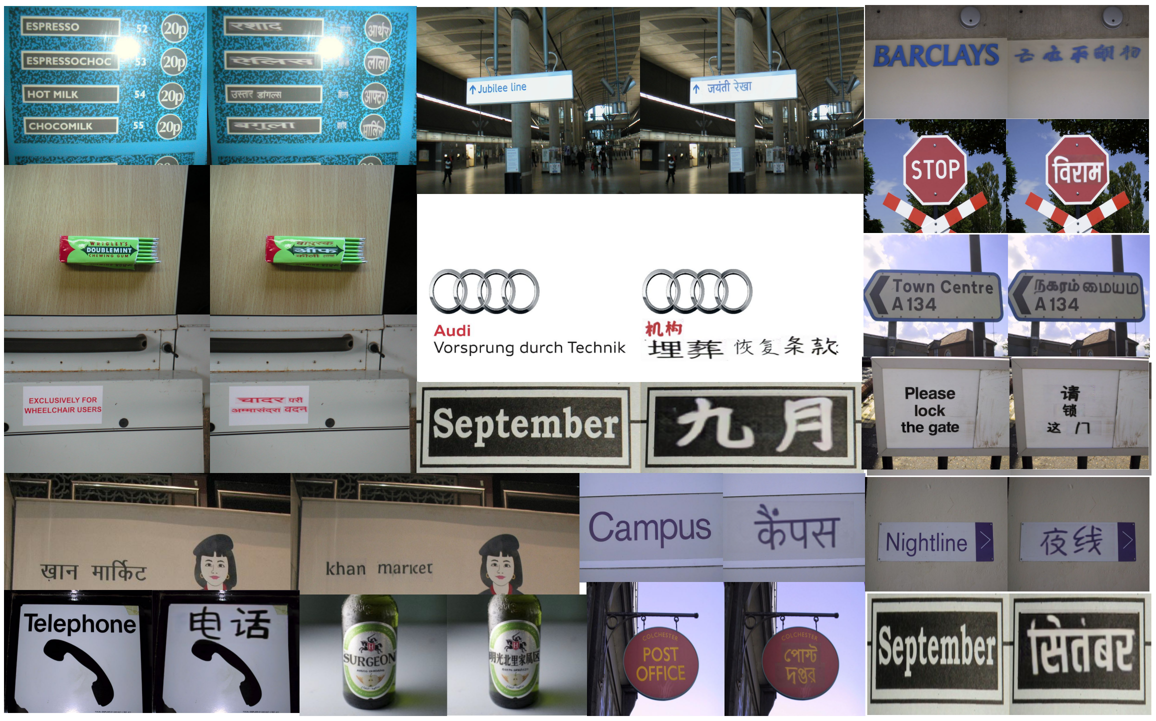

Figure S.12. Results of LISTNet for STE on sample images (some images are from the internet, whereas others are from the datasets used for experiments). On the left is the scene text image with the input style image, and on the right is the style transferred image (output of LISTNet). Note that style transfer does not need to be on the translated word; it can be on any word. For some of the images shown here, the style transfer is done on the translated word; in other images, it is done on a random (or user-supplied) word.

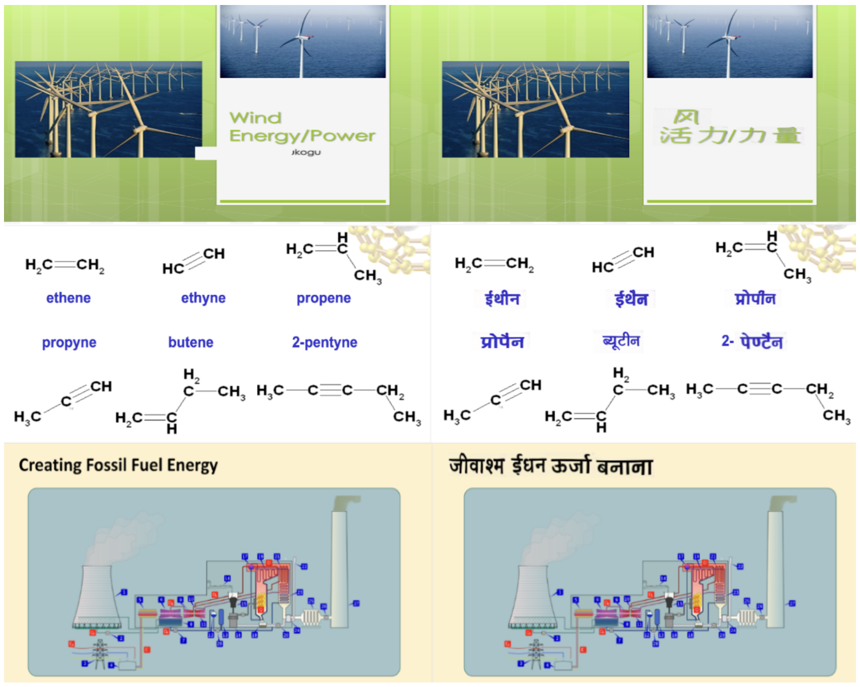

Figure S.13. Results of LISTNet for STE on educational PPTs. The same comments made for Figure S.12 are applicable here.

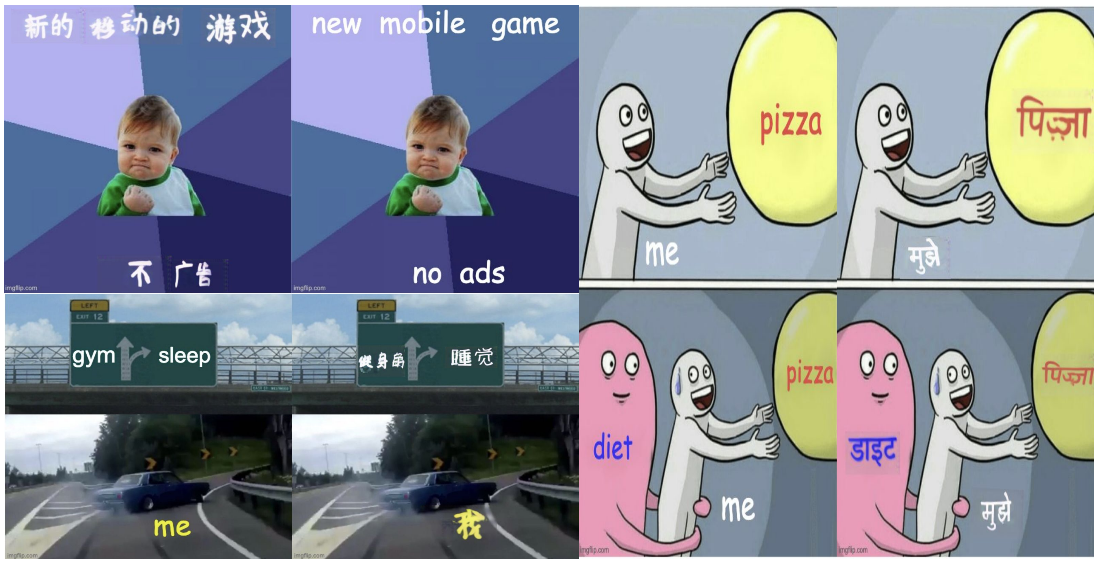

Figure S.14. Results of LISTNet for STE on social media memes. The same comments made for Figure S.12 are applicable here.

puter Vision and Pattern Recognition, pages 11976–11986, 2022. 5

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5

[6] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 5

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 3

[8] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 568–578, 2021. 5

[9] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22–31, 2021. 5

[10] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 5

[11] Moonbin Yim, Yoonsik Kim, Han-Cheol Cho, and Sungrae Park. Synthtiger: Synthetic text image generator towards better text recognition models. In *International Conference on Document Analysis and Recognition*, pages 109–124. Springer, 2021. 1, 2