Investigating Efficient and Transferable Multilingual Scene Text Recognition with Varying Image Size and Aspect Ratio

Chenkuan Liu
The University of Texas at Austin
ckliu@utexas.edu

Feihong Hu
The University of Texas at Austin
feihonghu@utexas.edu

Abstract

Scene text recognition (STR) is a subfield in object detection and recognition that specifically deals with text presented in natural images. Recent advancements in deep learning and computer vision have boosted the general performance of STR. However, most of the benchmarks so far are only focused on Latin scripts. In a world that is becoming more cosmopolitan, the need for multilingual STR is steadily growing. In addition, as most of STR algorithms will be eventually deployed in mobile devices, it is also necessary to make the model more efficient in both computation and resource. In this project, we focus on constructing efficient and transferable models that are able to deal with multilingual scene text recognition tasks. The datasets we focus on are CVSI-2015 and MLT-2019. Besides being multilingual, these datasets pose extra challenges with largely varying image size and aspect ratio. With such consideration, we propose several model constructions that are possible to solve these challenges in an efficient and transferable way. These models can extract textual information from million-pixel images without directly running convolution on them, which is time consuming and may easily cause memory explosion. However, due to our time and resource limitation, more experiments are required to further the methodologies we suggest here.

1. Introduction

The goal of scene text recognition is to identify and recognize textual information inside natural images. It can be considered as a subtask of image detection and recognition. Various deep learning models, such as VGG [17], ResNet [6], R-CNN [5], etc., can be used for STR model construction. However, compared to general image detection tasks, text recognition has its own specialty. First, unlike general images, scene text has its own script it belongs to. So far, most of the methods and benchmarks are developed and tested on English scene text images. In a globalized world where people will likely encounter more than one language in their daily life, it is important to let scene text recognition algorithms not only work well on English images but also on other Latin and non-Latin images. This leads to multilingual scene text recognition, which introduces the specific task of script identification besides the normal tasks of text detection and recognition.

Second, since a lot of scene text recognition applications will be deployed in devices such as smartphones, androids and surveillance cameras, it is essential to ensure the efficiency and transferability of the learning algorithm and architecture. Efficiency here are not only measured in conventional sense of training time, model size and inference time, but also in its ability to deal with large-scale, wild scene text information. In wild scene text data, it is common for images and word boxes not having same size or aspect ratio. Directly applying pixel-wise deep neural networks is time-consuming and may easily cause memory explosion on mobile devices. Unlike image classification, rescaling all images into the same size may cause information loss, since some word boxes are already very small in the original image and they may vanish after rescaling. Unlike general object detection, the scene texts in wild images can have various kinds of geometry, ranging from rectangular, parallelogram to more irregular, curved structures. The rectangular word box notations in object detection tasks may not be sufficient to capture text information in STR problems. To make the model efficient, it is necessary to avoid introducing excessive complexities when taking these factors into consideration.

On the other hand, transferability is a specific idea regarding multilingual scene text recognition. Since multilingual STR contains multiple tasks ranging from script identification to text detection and recognition, the specific context may require us to perform only certain tasks inside. Therefore, instead of treating each task independently, a better way is to allow the methodology for one task to be readily adaptable into other tasks. For instance, sometimes we may only need to perform script identification on multilingual scene texts. When it turns out that we need to perform text recognition on the data as well, we can quickly utilize and adapt the methodologies and results from script identification to text recognition, instead of building an entirely new model from scratch.

The key datasets we explore are CVSI-2015 [16] and MLT-2019 [12]. Both datasets are composed by real, multilingual scene text images. CVSI-2015 is a dataset that consists of cropped word images extracted from multi-lingual videos and it contains various subtasks of script identification. MLT-2019 is a data set of natural scene images containing text instances and it contains various subtasks such as text detection, script identification and text recognition. Both of these datasets incorporate the challenges we discussed previously. It is not only favorable but also necessary to come up with efficient and transferable methodologies. However, due to our time and resource limitation, we are only able to address a portion of the concerns discussed previously. Nevertheless, with appropriate extension and adjustment, the methodologies we propose here have the potential to address the challenges we are not able to cover in the scope of this report.

2. Related Work

Chen *et al.* [1] presents a comprehensive survey of almost all aspects associated with scene text recognition so far, including currently available datasets as well as remaining challenges and possible future directions in this field. The topic of multilingual scene text recognition is also discussed inside, which is the key motivation for this report and study.

Regarding the specific dataset and methods, Sharma *et al.* [16] not only introduced the CVSI-2015 dataset but also summarized the methods used during the time of its competition, which includes both conventional machine learning methods and deep learning methods. The deep learning methods presented inside primarily use convolutional neural network with specific handling of image preprocessing and data augmentation.

The MLT-2019 introduction paper by Nayef *et al.* [12] presented a baseline method called ”E2E-MLT” [13], which uses ResNet50 [6] as the first step and follows by text proposal and task-specific neural networks. However, there are a lot of rooms to improve on it regarding both model transferability and efficiency. Nayef *et al.* [12] also listed the methods that achieved high performance during the competition, which generally used a combination of techniques including but not limited to CNN, RNN, Seq2Seq and self-attention.

There are many other papers and proposed models beyond the ones presented above. However, the methods above are representative of the trends regarding multilingual STR algorithm: the methodologies that achieve high performance have complex structure and may not be accessible to normal computing machines. In addition, some of them are also very task-specific and are not easily adaptable to other subtasks of multilingual STR or even other datasets.

In this report, we develop a patch-wise method that first devide the scene text images into square patches and then feed these patches into convolutional neural network instead of the original image. The outputs will be further fed into transformers or additional convolutional neural network structures. In this way, our methodology achieve promising results without applying convolution directly on million-level pixel images, and the backbone can be adapted into different subtasks ranging from script identification to text detection and recognition.

3. Methodology

3.1. Script Identification

3.1.1 Proposed Method for Identification

Fig. 1 is our proposed method for multilingual script identification. Given a script image, we first transform it into grayscale, and then resize the height to 64 and rescale the width proportionally to the nearest multiple of 64. For instance, the original script image input on the lower-left of Fig. 1 has size $(38, 104, 3)$. After grayscaling and proportional resizing, it becomes $(64, 192, 1)$. Then, we slice it into several $(64, 64, 1)$ square patches. In this way, the sample image in Fig. 1 is partitioned into three patches as displayed inside. Each patch will pass into a convolutional neural network. For the choice of CNN, we use MobileNet [7] here as it is parameter-saving and computationally faster. Several customizations are made inside: the input is $(64, 64, 1)$ instead of 3 channels for color images; the activation function is leaky Relu with $\alpha = 0.1$ instead of the original Relu, as empirically leaky Relu leads to better performance on this dataset; a few blocks inside are removed compared to the original MobileNet [7] to reduce the total number of parameters, as more parameters will be added to the model from later structure.

After our customized CNN, each square patch will generate a $(512, 1)$ column vector embedding. We will concatenate the embeddings generated from different patches and pad to length 11, so that the embeddings will have shape $(512, 11)$. 11 is chosen as our max length because the largest aspect ratio from CVSI dataset is 10.74:1 and it will lead to 11 square patches. Then, these embeddings will pass into a masked Transformer encoder [19]. At here, we use 8 attention heads with each head generating a 64 dimension output, and we let the key and value share the same matrix. The number of encoders is a hyperparameter and we set it to 2 here. In this way, the output will have the same shape $(512, 11)$. We add skip-connections to the output from the embeddings before the encoder. The skip-connections are in the form of concatenation as suggested

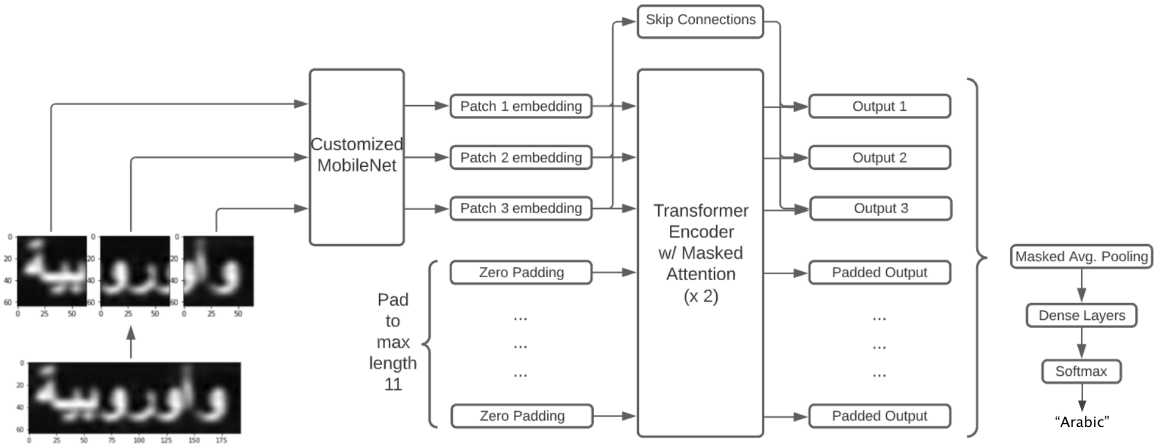


Figure 1. Structure of our method for script identification.

in DenseNet [8] instead of summation in ResNet [6], so that the gradients will be directly passed back into the embeddings before the encoder and make the learning update faster and easier. With such concatenation, the resulted output will have shape $(1024, 11)$. In order to only consider the meaningful embedding columns, a masked average pooling layer will be acted upon the output so that the resulting vector will have shape $(1024, 1)$. It will pass into a few dense layers with softmax at the end to give the script type prediction.

The method here is inspired by previous works in different fields. The patch-wise slicing on the input image and Transformer [19] processing is motivated by the practice used in Vision Transformer [4]. However, instead of the costly linear projection, we use our customized MobileNet [7] to generate the embeddings for each patch. In particular, the batch number of the patches feeding into the customized MobileNet is dependent on the width of the original image. For example, suppose we use a batch size of 2. The first image is the one displayed in Fig. 1, which leads to 3 square patches and hence can be formulated as a $(3, 64, 64, 1)$ input. Suppose the second image has 9 square patches, then it will be formulated as a $(9, 64, 64, 1)$ input. These inputs will be concatenated into a $(12, 64, 64, 1)$ block and pass into our customized MobileNet as a single batch with batch number 12 with associated batch normalization [9] inside. This is particularly helpful in terms of training and inference efficiency. When the images get much larger, as we will demonstrate later, the only increasing number is the batch number for the MobileNet instead of the height and width of the input image patches. In other words, the efficiency of our proposed method mainly relies on our specific application of batching. Since the scene text images have varying sizes and aspect ratio, the number of patches feeding into the MobileNet can be ranging from less than 10 to more than a hundred when passing forward, but each patch will always have the size of $(64, 64, 1)$.

3.1.2 Updates for Identification

Our proposed method is mainly motivated by two major practices: the practice to pad sequences into the same maximum length in many natural language processing problems, and the practice of applying Transformer [19] to vision setting as established by Visual Transformer [4]. However, many updates and modifications can be done to the model we proposed in Fig. 1.

During our actual implementations, the positional encoding of Transformer do not bring any noticeable difference to the performance. This is partially due to the fact that if one image belongs to a certain script, each patch will also belong to the exact same script and hence the positional ordering does not matter as much as in other task settings. As a result, a more simplified attempt is to remove the entire Transformer encoder and apply masked average pooling right after our customized MobileNet (this can also be considered as using 0 stack of Transformer encoder). In this way, there is also no residual concatenation inside. As a result, the simplified model without Transformer encoder can be considered exactly as a vision variant of deep averaging network (DAN) [10]. Empirically, as we will demonstrate in the Experiments section, this simplified model actually gives better performance on the CVSI [16] dataset with fewer parameters and faster training time. However, this does not mean the simplified method will always prevail. In certain settings suggested by Cheng *et al.* [2], such as Latin vs. Russian and Chinese vs. Japanese, the scripts of interest have a large amount of shared characters in com-

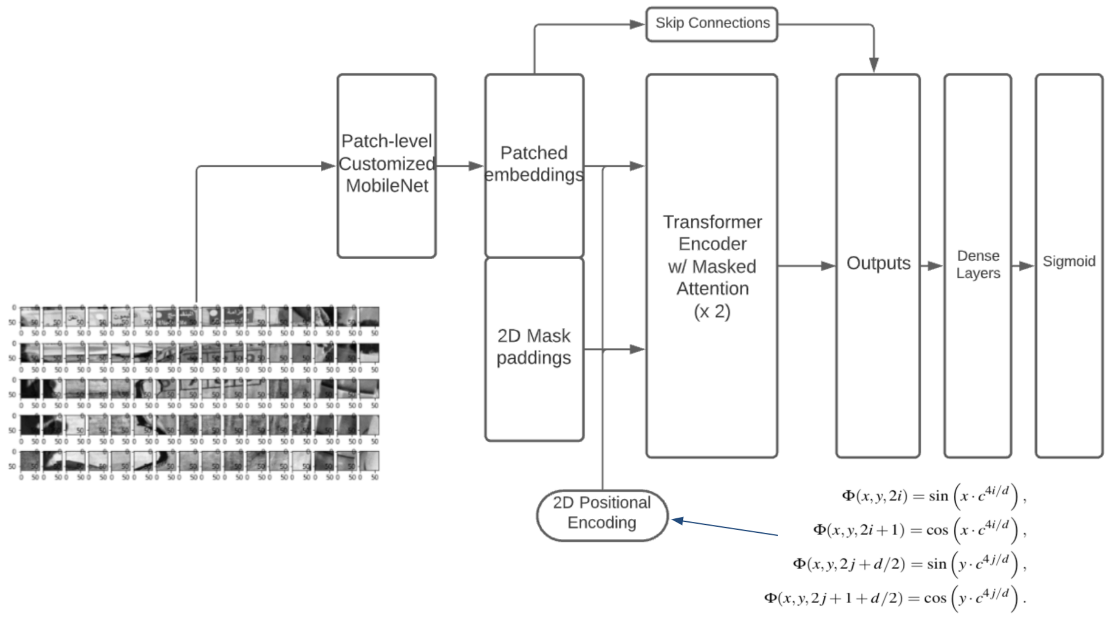

Skip Connections
Patch-level Customized MobileNet
Patched embeddings
2D Mask paddings
Transformer Encoder w/ Masked Attention (x 2)
Outputs
Dense Layers
Sigmoid
2D Positional Encoding
$\Phi(x, y, 2i) = \sin\left(x \cdot c^{4i/d}\right)$,
$\Phi(x, y, 2i+1) = \cos\left(x \cdot c^{4i/d}\right)$,
$\Phi(x, y, 2j+d/2) = \sin\left(y \cdot c^{4j/d}\right)$,
$\Phi(x, y, 2j+1+d/2) = \cos\left(y \cdot c^{4j/d}\right)$.

Figure 2. Structure of our directly extended method for text detection in simplified setting.

mon and the difference is relatively subtle. In such case, merely averaging the embeddings may not be sufficient and further aggregations are needed on the patches.

In general, our simplified method without Transformer encoder leads to better performance compared to the original method in Fig. 1, but more datasets and tests are needed to verify the necessity of the inclusion of Transformer encoder as well as the positional encoding inside. Nevertheless, since the simplified method can also be considered as a special case with 0 stacks of Transformer encoder inside, we present it as a whole as Fig. 1 displays.

3.2. Text Detection

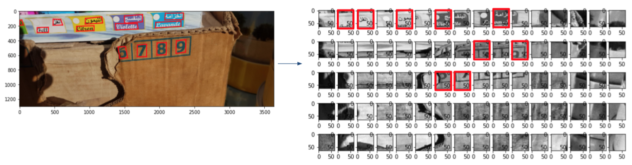


Figure 3. Demonstration of the simplified text detection task (the left image is enlarged and displayed later in Fig. 7).

Text detection is a much heavier task compared to script identification, since the images are generally much larger and we need to predict coordinates instead of categorical outputs. In consideration of our time and resource limitation, in this report, we simplify the text detection task to predict which image patch contains the center of at least one text box. Fig. 3 provides a visual demonstration of how the original text box prediction is simplified to patch-wise text center prediction in our setting. The center coordinates are computed as the mean value of the four corners of the quadrangle. They are divided by 64 using integer division and the resulting (x, y) quotient value represents the corresponding patch with positive label.

3.2.1 Proposed Method for Detection

Fig. 2 demonstrates our proposed method for text detection in the simplified setting. This method is directly extended from our previous method for script identification in Fig. 1. There are only three minor differences inside. The first one is that we use 2D positional embeddings instead of 1D to account for the spatial position of the image. The formulas for 2D positional embeddings are from Raisi *et al.* [14]. The second one is that we pad the max length to 256 instead of 11. This is because we are mainly experimenting text detection on MLT-2019 [12] dataset, which contains relatively large images and we rescale the longer side to 1024 and resize the shorter side accordingly to the nearest multiple of 64 (if the original image is less than 1024×1024, then we will keep the size unchanged and only rescale the height and width to the nearest multiple of 64). In this way, the largest image will have size 1024×1024, which will give 16×16 patches and hence lead to a length 256 after flattening. Each patch will output a 512 dimension embedding as before. The third one is that we use sigmoid activation function at the end to give binary predictions of whether a

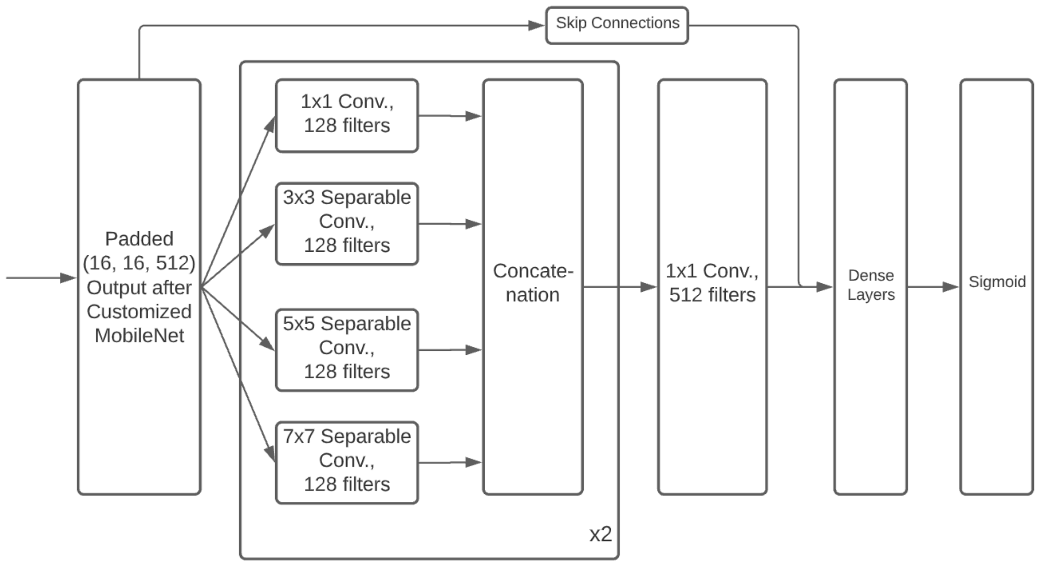

Skip Connections
Padded (16, 16, 512) Output after Customized MobileNet
1x1 Conv., 128 filters
3x3 Separable Conv., 128 filters
5x5 Separable Conv., 128 filters
7x7 Separable Conv., 128 filters
Concatenation
x2
1x1 Conv., 512 filters
Dense Layers
Sigmoid

Figure 4. Structure of our updated method for text detection in simplified setting.

particular patch contains at least one text box center or not.

3.2.2 Updates for Detection

Since transformers are heavier to train than convolutional neural networks in general sense, and our script identification method gives better performance when excluding the transformer structure inside, it becomes a reasonable approach to find substitutes for our 2D-transformer based method in Fig. 2 as well. However, different from our previous script identification, the relative positional information is necessary here, as whether an patch contains text box centers is dependent on the information in its neighboring patches. With such consideration, we propose an updated method that applies ideas from Inception network [18] to account for different receptive fields during convolution.

Fig 4. demonstrates the structure of the updated method. After we get the padded $(16, 16, 512)$ embeddings as before, instead of using transformer encoder, we pass the embeddings into 4 separate convolution blocks with kernel sizes 1×1, 3×3, 5×5 and 7×7 respectively. We use same padding here to preserve the 16×16 spatial dimension. Each convolution block contains 128 filters and thus will output 128 channels. The 1×1 convolution uses the standard convolution procedure; the 3×3, 5×5 and 7×7 convolutions use the depthwise separable convolution procedure introduced by Chollet [3], which consists of a depthwise convolution followed by a pointwise convolution without non-linearities inside. Such procedure largely saves the total number of parameters compared to standard convolution. The outputs from each convolution block will concatenate together to form a new $(16, 16, 512)$ embedding, and this embedding will go through the exact same procedure again. The results then will go through a 1×1 convolution with 512 filters. Its output will be skip-concatenated with the original $(16, 16, 512)$ embeddings to form a $(16, 16, 1024)$ embedding. This concatenated embedding will go through dense layers with sigmoid activation at the end to output binary predictions.

The reason to use convolution blocks with different kernel sizes is because some text boxes are very small and only locate inside 1 or 2 patches, while others might be very large and can across half of the image. With different kernel sizes, we can get receptive fields with different sizes and the text boxes with varying shapes will all get covered during feature extraction.

4. Experiments

4.1. Datasets

CVSI-2015 CVSI-2015 [16] is a multi-lingual dataset that consists of 10,866 cropped word images extracted from multi-lingual videos. These images are composed of three south Indian scripts (i.e. Kannada, Tamil and Telugu), six north Indian scripts (i.e. Hindi, Bengali, Oriya, Gujrathi, Punjabi and Arabic), and English scripts. Fig. 5 shows examples of the ten scripts in CVSI-2015 dataset. Fig. 6 shows the detailed statistics of image numbers inside training set, validation set and test set. The task of CVSI-2015 is to perform script identification from the combination of all ten scripts.

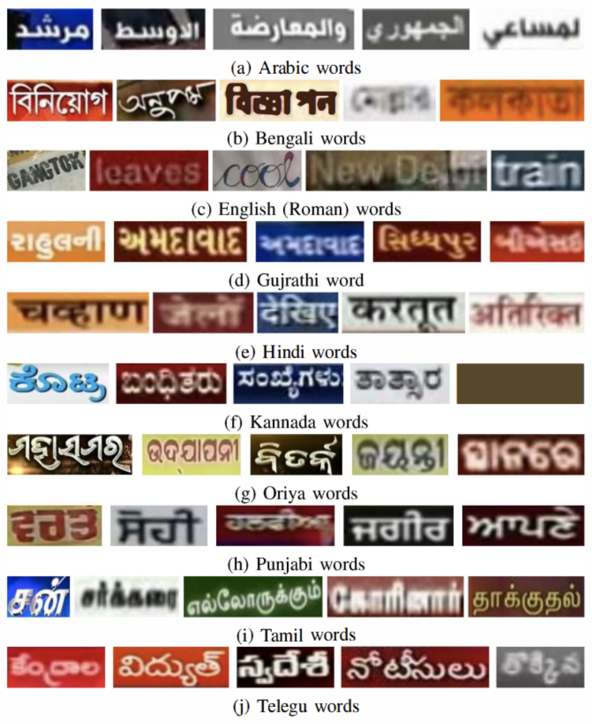


Figure 5. Examples of the ten scripts in CVSI-2015 dataset.

Script	Total word	Training set	Validation set	Testing set
Arabic	1011	607	101	303
Bengali	1032	619	103	310
English (Roman)	1135	681	113	341
Gujrathi	1086	651	108	327
Hindi (Devnagari)	1088	653	109	326
Kannada	1047	628	105	314
Oriya	1087	652	109	326
Punjabi (Gurumukhi)	1055	633	106	316
Tamil	1070	642	107	321
Telegu	1077	646	108	323
Total	10688	6412	1069	3207

Figure 6. Detailed statistics of CVSI-2015 dataset.

MLT-2019 MLT-2019 [12] is a dataset of 20,000 natural scene images containing text instances from Arabic, Bangla, Chinese, Devanagari, English, French, German, Italian, Japanese and Korean, (2,000 images per language). Every image is real and the text in the image is annotated at work level. Inside the 20,000 images, 10,000 images are available as training set with ground truths given, while the other 10,000 images are kept as validation and the results can only be seen after submitted to the competition. Since the images of MLT-2019 are relatively large, we only use 5,000 images (500 from each language) from the 10,000 train images here. We further divide the 5,000 images into 4,000 train images (400 from each language) and 1,000 validation images (100 from each language) in our study. Fig. 7 and Fig. 8 are sample images from the dataset.

MLT-2019 contains four tasks in total: multi-lingual text detection, cropped word script identification, joint-text detection and script identification and end-to-end text detection and recognition. In our experiments, we only focus on the first multi-lingual text detection task with simplified patchwise binary prediction as demonstrated in Fig. 3.

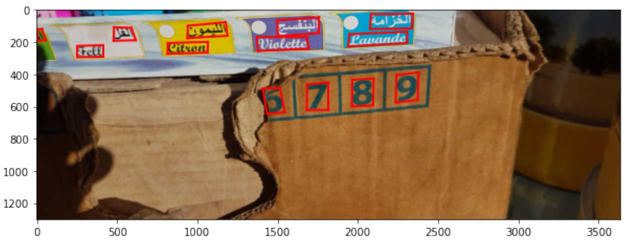


Figure 7. The left image in Fig. 3, which is a multi-lingual sample in MLT-2019. The word boxes in MLT-2019 are annotated by quadrangles instead of rectangular boxes.

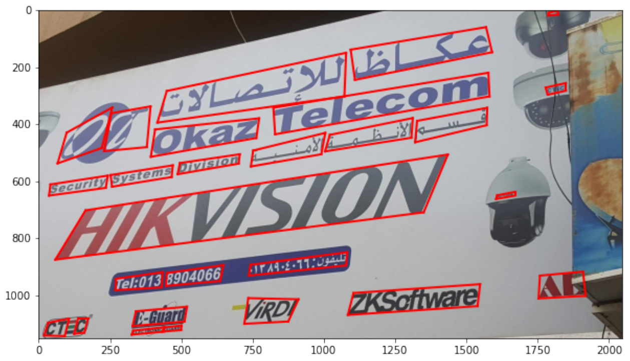


Figure 8. Another multi-lingual sample in MLT-2019. We can observe that some word boxes are relatively small while others may span across half of the image.

4.2. Implementation Detail

We use Tensorflow and Keras for this study. All of our implementations are run using Google Colab's free provided Tesla K80 GPU. All methods for both datasets will first change the original color image input into grayscale.

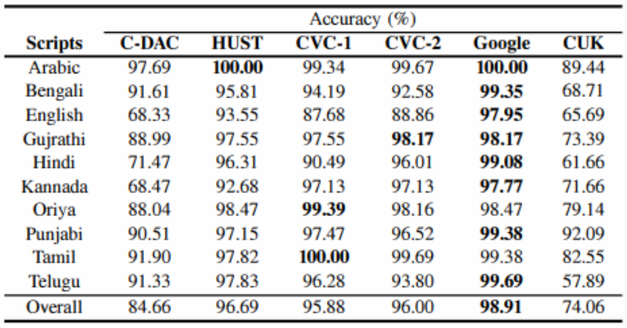
For CVSI-2015 [16], we implement three methods: the first method rescales each image into fixed $(64, 96, 1)$ shape and passes it into a full-scaled MobileNet [7] with Leaky Relu as the choice of activation function inside; the second method uses the model displayed in Fig. 1, which uses employs padding, Transformer encoder [19] and masked averaging as we discussed in Section 3.1.1; the third method performs masked averaging right after MobileNet output as discussed in Section 3.1.2. All methods use Adam optimizer [11] with initial learning rate 8×10^{-4} and batch size 16 for training and each epoch performs random shuffling on training data. We use categorical cross-entropy as our loss function. All methods are implemented for at least 24 epochs and after that the epoch with best validation performance will be saved as our choice of the weights for the model.

For MLT-2019 [12], we implement two methods: the 2D masked transformer version as described in Section 3.2.1 and Fig. 2, and the updated block convolution version as described in Section 3.2.2 and Fig. 4. Both methods use Adam optimizer [11] with initial learning rate 2×10^{-4} and batch size 4 for training. Since the images are relatively large, for each epoch, 512 images will be randomly selected from the 4,000 training set for learning and batched gradient descent. We use weighted binary cross-entropy as our loss function. Since only a small portion of all patches contains scene texts information, we use a weight value of 5.0 to reduce the potential false negatives of the model prediction. Both methods are implemented for 40 epochs in total before hitting Google Colab's usage limit.

4.3. Evaluation

Methods	# of params	Validation and test accuracy range (%)	Highest validation accuracy (%)	Highest test accuracy (%)
Fixed Size Mobile	4.3 M	83.0∼85.5	85.87	85.10
Mobile+Transformer	5.7 M	89.0∼91.0	91.86	90.86
Mobile+Direct Avg.	4.1 M	91.0∼93.5	**93.82**	**91.80**

Table 1. Performance of our methods on CVSI-2015.



	Accuracy (%)					
Scripts	**C-DAC**	**HUST**	**CVC-1**	**CVC-2**	**Google**	**CUK**
Arabic	97.69	**100.00**	99.34	99.67	**100.00**	89.44
Bengali	91.61	95.81	94.19	92.58	**99.35**	68.71
English	68.33	93.55	87.68	88.86	**97.95**	65.69
Gujrathi	88.99	97.55	97.55	**98.17**	**98.17**	73.39
Hindi	71.47	96.31	90.49	96.01	**99.08**	61.66
Kannada	68.47	92.68	97.13	97.13	**97.77**	71.66
Oriya	88.04	98.47	**99.39**	98.16	98.47	79.14
Punjabi	90.51	97.15	97.47	96.52	**99.38**	92.09
Tamil	91.90	97.82	**100.00**	99.69	99.38	82.55
Telugu	91.33	97.83	96.28	93.80	**99.69**	57.89
Overall	84.66	96.69	95.88	96.00	**98.91**	74.06

Figure 9. Result of CVSI-2015 in the original competition.

CVSI-2015 We use accuracy score to measure the performance of our three methods mentioned previously. Table 1 shows the number of parameters, general range and highest values of validation and test accuracy of the three methods. Fig. 9 displays the original competition results of CVSI-2015 [16]. The highest accuracy score is 98.91%, which is achieved using significant data augmentation and sliding windows on wide images with highest confidence score chosen.

In our experiments, the third method, with only 4.1 million parameters, achieves approximately 92-93% accuracy. However, our method is not saturated yet. We can add more convolution blocks inside our customized MobileNet and apply data augmentation as well. A reasonable guess is that data augmentation is essential to boost our performance from current stage to more than 95% accuracy or even higher. There are many ways to perform data augmentation, such as treating each patch (or a combination of two or three patches) as a single image, or rotating the images to certain degrees. The future works on this dataset is to experiment with different kinds of data augmentation and observe their influence to the performance.

Methods	# of params	Precision (%)	Recall (%)	F1-score (%)
Random Assign Labels	/	4.48	50.56	8.23
Mobile+2D Transformer	5.6 M	46.65	48.18	47.40
Mobile+Inception Blocks	4.4 M	47.85	52.66	50.14

Table 2. Precision, recall and F1-score on our 1,000 validation images from MLT-2019.

MLT-2019 We use F1-score to measure our performance on the simplified text detection task with the two methods aforementioned, as illustrated in Fig. 2 and Fig. 4 respectively. Table 2 displays the number of parameters and the precision, recall and F1-score of each method. As it still takes many more epochs to converge, we compare our results with randomly assigning labels to the image patches. The results in the table show that both of our methods truly learn the patterns of the tasks and perform much better than random assignment, which has relatively high recall but very low precision as random guessing would lead to many false positives. In particular, the last method with customized inception blocks achieves 50.14 F1 score with a 4.4 million parameter light-weight model and limited training resource.

5. Conclusion and Future Work

5.1. Summary and Insights

In this study, we propose several patch-wise methods on both script identification and text detection tasks that can handle scene text images with varying image size and aspect ratio. All methods use patch-wise MobileNet [7] as their backbone. The patch-wise processing avoids applying convolutions directly on million-pixel images. When the image size gets increasingly large, each patch will keep the same dimension and the only value that is increasing is the batch size feeding into the patched MobileNet, namely the total number of patches within our designated training batch. In other words, our methods are utilizing the fact that modern computing devices are able to handle the inputs in an efficient way during learning when it is composed of a large batch size but each single element inside is relatively small (which is just $64 \times 64 \times 1$ in our setup).

After the common patch-wise backbone, we experiment with different approaches that includes either Transformer [19] or different ways of convolution aggregating. From the evaluation, we can observe that, for both tasks, the methods with Transformer do not perform better than the methods without it. However, this does not mean Transformer is not a viable option in scene text recognition problems. Due to our time and resource limitation, all methods here have not reached their full potential. In addition, even within our limited setting, Transformer demonstrates a high degree of transferability. When we move from script identification to text detection, the Transformer-based method can seamlessly transform from Fig. 1 to Fig. 2. For convolution-based methods, we need more customizations regarding each sub-problem. In addition, Transformer-based method also has high adaptability within a specific task. For example, if we want to make the text detection method more fine-grained and use 32×32 patches instead of 16×16, the Transformer-based method can stay exactly the same, while for convolution blocks in Fig. 4, we might need to add convolution filters with more kernel size options, such as 9×9, 11×11 or even larger, to account for the change of our patch size. Moreover, due to our resource limit, we do not extend the methods here into full-scaled scene text recognition problems, in which a lot of synthetic images are needed as additional data to perform multilingual text decoding, as suggested in MLT-2019's official paper [12]. During scene text recognition, a decoder is needed, and with Transformer decoder [19] in that setting, it means that we are using the Transformer architecture for all different subtasks of scene text recognition. Such same modality will be helpful for editing and tuning the models and free us from handcrafting separate model for each problem.

On the other hand, Transformer-based methods are mainly inspired by their original applications in natural language processing. When adapted to vision setting, it is necessary to notice the difference between vision problems (scene text problems in particular in our context) and language processing problems. In NLP, long range dependency can happen quite often in tasks such as co-referencing and translation. In scene text detection problems, even though the text boxes may have varying sizes and shapes, each of them is located in a specific region of the image. As a result, only a certain number of patches are needed to determine the center (or actual text box coordinates in future extensions) of a text, and a globalwise self-attention on the image might be an "overkill." This is observed in our implementations and results in Table 2, where the "Mobile+Inception Blocks" method achieves better performance with fewer parameters compared to the "Mobile+2D Transformer" method.

5.2. Extension and Future Work

Future work on our study includes data augmentation and continuing training our methods with more computing resource. Moreover, we can extend the simplified text detection method into a full-scaled detection of quadrangle word boxes. During such extension, we can utilize the results we have got so far and make the final layer predict 8 coordinate labels of the four corner points beside the binary center prediction. After that, during anchor box matching, we can only consider the corner coordinates predictions where the associated binary center prediction label is true. In other words, for efficient extension, it is also necessary to avoid performing anchor box matching and non-max suppression on the entire image. For non-max suppression, since we are dealing with quadrangles instead of rectangular boxes, it is also necessary to come up with substitute computations, such as calculating the sum of distances between each corresponding corner, instead of directly calculate the overlapping area of two arbitrary quadrangles. With certain tuning, extensions and modifications as suggested here, it is possible to extend our backbone to build a patch-wise based variant of YOLO algorithm [15] for scene text detection, which only looks forward and is end-to-end differentiable.

After that, we can add Transformer decoder [19] with customizations to perform multilingual scene text decoding. However, even with modern computing resources, both text detection and text recognition are computationally heavy. Even our models are relatively light-weight in terms of number of parameters, they will still take a significant amount of time to train for full-scaled scene text detection and recognition. In order to make these tasks efficient to learn, more methodologies beyond the current data-centric regime are needed to be discovered.

References

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

[2] Changxu Cheng, Qiuhui Huang, Xiang Bai, Bin Feng, and Wenyu Liu. Patch aggregator for scene text script identification. *ArXiv*, 1912.03818, 2019.

[3] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):142–158, 2016.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[7] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco An-

dreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *ArXiv*, 1704.04861, 2017.

[8] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456. PMLR, 2015.

[10] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691. ACL, 2015.

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv*, 1412.6980, 2017.

[12] Nibal Nayef, Yash Patel, Michal Busta, Pinaki Nath Chowdhury, Dimosthenis Karatzas, Wafa Khlif, Jiri Matas, Umapada Pal, Jean-Christophe Burie, Cheng-lin Liu, and Jean-Marc Ogier. Icdar2019 robust reading challenge on multi-lingual scene text detection and recognition — rrc-mlt-2019. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1582–1587, 2019.

[13] Yash Patel, Michal Busta, and Jiri Matas. E2e-mlt - an unconstrained end-to-end method for multi-language scene text. *ArXiv*, 1801.09919, 2018.

[14] Zobeir Raisi, Mohamed A. Naiel, Paul Fieguth, Steven Wardell, and John Zelek. 2d positional embedding-based transformer for scene text recognition. *Journal of Computational Vision and Imaging Systems*, 6(1):1–4, Jan. 2021.

[15] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[16] Nabin Sharma, Ranju Mandal, Rabi Sharma, Umapada Pal, and Michael Blumenstein. Icdar2015 competition on video script identification (cvsi 2015). In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1196–1200, 2015.

[17] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR 2015 : International Conference on Learning Representations*, 2015.

[18] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv*, 1409.4842, 2014.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.