EASTER: Simplifying Text Recognition using only 1D Convolutions

Kartik Chaudhary[†,*], Raghav Bali[†,*]
[†] Optum Global Solutions, India

Abstract

Recurrent units and complex gated layers are key components of most text recognition models. Their sequential nature and complex mechanisms require large labelled training datasets, high computational requirements and lead to slower inference times. In this paper, we present an **E**fficient **A**nd **S**calable **TE**xt **R**ecognizer (EASTER) to perform optical character recognition on both machine printed and handwritten text. Our model utilises only 1-D convolutional layers without any recurrence or complex gating mechanisms. Our proposed architecture achieves performance similar to best performing recurrent architectures by using only 4% of training data for offline handwritten text recognition task. We present results of our model on different machine printed text recognition datasets as well. We also showcase improvements over the current best results on line level offline handwritten text recognition task. Our work presents a highly scalable and deployable model for real-world settings while being highly performant.

Keywords: OCR, Convolutions, RNNs

1. Introduction

Text is a ubiquitous entity in natural images and most real world datasets like scanned documents, restaurant menu cards, receipts, tax forms, license plates, etc. These datasets may contain text in both printed as well as handwritten formats. Extracting text information from such datasets is a complex task due to variety of writing styles and more so due to limitation of ground truth. Optical Character Recognition (OCR) systems have been in existence for quite sometime now [1, 2]. The improvements and research in Deep Learning, CNN and LSTM based OCR solutions [3–10] and [11] have taken the field by storm. The results from these solutions are leaps and bounds ahead of traditional solutions like Tesseract[12]. The downside of these Deep Learning based solutions is their dependence on huge amounts of labelled training data and compute.

Handwritten Text Recognition or HTR is an even more involved process with countless variation of writing styles. While OCR for printed text has seen good improvements, HTR still remains a challenge. Moreover, the models trained for printed text do not generalise well (even with transfer learning) onto HTR tasks.

This paper makes the following contributions:

- A computationally efficient, simple and scalable, fully convolutional neural network architecture for the task of OCR and HTR that uses only 1D convolutions. We also present ways of generating synthetic training data that eases the task of building an efficient text extraction engine from natural text images.
- We improve upon the state-of-the-art (SOTA) word error rate (WER) on IAM-offline Handwritten text line recognition task and achieve SOTA WER for OCR task on IIIT-5K test data as well.
- We conclude the study by highlighting the limitations and practical considerations of EASTER.

[*]kartik@optum.com, raghavbali@optum.com

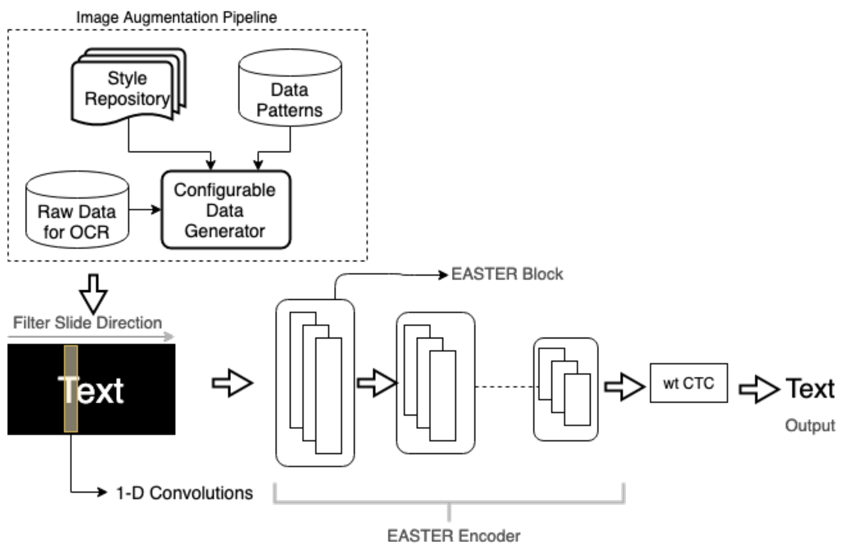


Figure 1. EASTER Pipeline consists of a configurable data generator, an image augmentation pipeline and recurrence free EASTER architecture. Given a list of data patterns like names, addresses, phone numbers, etc. along with a list of fonts, styles, etc. the configurable data generator can generate training datasets of required size. Augmentation pipeline helps add perturbations to generate realistic samples

2. Related Work

Ingle et. al.[13], showcase improvements on limited datasets with architectures that have recurrent connections. They present Gated Recurrent Convolutional Layers (GRCLs) as specialised convolutional layers to perform recurrence along depth as compared to LSTMs which do the same along time dimension. In this paper we address the problem of training data volume and compute requirements together by presenting variants of our fully convolutional architecture devoid of recurrent connections. Our proposed architecture works on both handwritten and machine printed texts. Being fully convolutional (using only 1-dimensional convolutions) enables development of smaller, faster and parallel trainable models. This further reduces the barrier for deployment and scalability.

Coquenet et. al. [14] challenge the notion of recurrent networks by presenting gated convolutional networks. They present results which are an improvement or are on par in comparison to CNN-BiLSTM based networks. Works by Baoguang et. al.[15], Jaderberg et. al.[16] and Lee et. al.[17] improve further by utilising specialised attention mechanisms, complex recurrent convolutions and other enhancement techniques to solve the tasks of OCR and HTR.

Works cited in this section optimize upon performance without much consideration on the amount of training data required (with the exception of Ingle et. al. [13]). With `EASTER`, the aim was to achieve high performance with minimal training data requirements. This added objective was based upon practical and deployment related considerations of our application. In section 5, we present experiments showcasing `EASTER`'s improved performance against GRCL[13] using just a fraction (only 4%) of their training data volume.

Our work is inspired by research in the field of Automatic Speech Recognition (ASR). Similar to text recognition, the task of speech recognition works upon a sequential input where label alignment is not a trivial task. ASR related work by [18] and [19] rely on using non-recurrent architectures. These works rely on multiple repeating block structures composed of different subcomponents. They also utilise residual connections and experiment with different depths to achieve state of the art performance. To the best of our knowledge, this is the first work that leverages only 1-D convolutions for the task of offline handwriting recognition without any recurrence. `EASTER`, being a simple architecture, outperforms more complex models in terms of training time, volume of training data and performance (word and character error rates). Figure 1 presents the overall setup.

3. Probing Why 1-D Convolutions work for OCR/HTR tasks

RNN based architectures are the go to solution when it comes to datasets which have temporal aspect associated with them. Collobert et. al. [20] presented convolutional networks capable of handling textual inputs for different Natural Language Processing(NLP) tasks. They discussed how convolutional filters capture local context in initial layers and use that to prepare global features in deeper layers. This idea of leveraging convolutions for capturing temporal aspects was also explored in the automatic speech recognition domain by Collobert et. al. [19]. This idea was further extended by Li et. al. [18] where they only used 1-dimensional convolutions (or 1D-CNN) for the task of ASR.

The usage of the term 1-D CNN here is synonymous with the way it is popularly used in the NLP domain. In the NLP domain, a 1-D CNN is a filter with same width as that of input token's embedding vector and moves across multiple tokens with each stride. Thus, performing convolutions in only one direction. This is different from the 2-dimensional convolutions used in computer vision use-cases where kernels move across height and width of an input image.

The input to an OCR/HTR system is typically an image which consists of a single word or a single line of text. Assuming a gray-scale input image (without channels) of height (H) and width (W) with all characters present within these dimensions. To identify a character within this image, a crop of height H and width w (where w is very small as compared to W) would be required. A convolutional filter of $H * w$ dimension can be leveraged in this setup. Such a filter moves across the width (W) to capture local context i.e., moves only in one direction. This is similar to the NLP setting, where we make use of simpler 1-D CNNs to extract local context and deeper layers extract global features. This refined formulation of OCR/HTR tasks along with success of convolutional networks in other temporal use cases motivated us to explore 1-D CNNs.

4. Data Preparation

To prepare training data for both handwritten and machine printed text, we applied different methodologies. The task of preparing training data for handwritten text is a bit more difficult as compared to machine printed. First difficulty is the availability of annotated handwritten text followed by the large variations in handwriting styles. Manually preparing such datasets is time consuming and cost ineffective. In the following subsections we will cover data preparation methodologies for both tasks in detail.

4.1. Handwritten Text

We utilised three different approaches to prepare the training dataset for the task of handwritten OCR. The IAM handwriting dataset [21] contains stroke information for handwritten text collected from various contributors in an un-constrained setting. We leverage the images from the offline subset as input samples which have corresponding transcriptions available. This dataset has limited samples yet provides line level data points with different handwriting styles and text patterns.

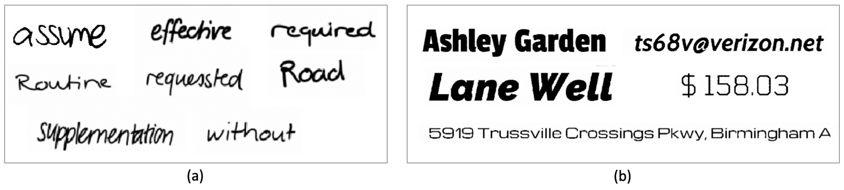


Figure 2. Synthetically Generated Random Samples; (a)Handwritten Text samples, (b) Machine Printed Text Samples for random name, email address, dollar value and street address

The next approach was to synthetically generate handwritten text data, similar to how we prepared the machine printed dataset. For this we leveraged the method presented by Alex Graves [22] to generate handwritten text. Similar to GRCL [13], we enabled the model to generate different styles and variations as well. We used specific hyper-parameters to control the mixture density layer and sampling temperature to generate usable samples. The final approach was to manually write as well as label the samples to make sure we cover a wide variety of text patterns and style. See figure 2(a) for samples from our handwritten text dataset.

4.2. Machine Printed Text

Collecting machine printed dataset from with-in the organisation and from different public sources has certain restrictions and issues. These factors range from datasets being very clean, limited fonts or having only specific type of patterns. Instead of collecting and curating data from such sources, we devised an ingenious method for synthetically generating a dataset for machine printed text.

The first step was to assemble a list of common patterns of text in the real world. These include text for street addresses, names, dollar amounts, etc.. The second step was to prepare a repository of different font styles, strokes, formatting (underline, italics, etc.). The final step was to use the patterns and styles as inputs to a probabilistic synthetic text data generator. The generator was designed to have configurable settings to adjust the variation in styles, patterns, length of text samples and overall dataset size. See figure 2(b) for samples from our machine printed text dataset. Using this method, we virtually have the ability to generate infinite such datasets for training and improving our models.

5. Architecture

OCR and HTR both involve taking images with text as input and generating corresponding text as output. Recurrent architectures make use of LSTMs to capture sequential dependency followed by Connectionist Temporal Loss (CTC)[7] to help train models without specifically aligning inputs and their labels.

The basic architecture we present here is a 1-D Convolutional network inspired by the research in the field of Automatic Speech Recognition (ASR) tasks. Works of Li et. al.[18], Collobert et. al.[19] and Pratap et. al. [23] highlight the effectiveness of convolutional networks in handling sequence to sequence tasks (ASR) without recurrent connections. We extend the similar thought process for the field of OCR and HTR using `EASTER` (see figure 4).

5.1. `EASTER` Encoder

`EASTER` follows a block approach wherein each block consists of multiple repeating sub-blocks. Each sub-block comprises of a 1-D Convolutional layer with multiple filters followed by layers for normalisation, ReLU and dropout. We utilise padding to maintain the dimensions of the input slice. Each `EASTER` architecture has 1 fixed preprocessing block(B1) and 3 fixed post-processing blocks(B2-B4). The pre and post processing blocks also follow similar block structure. In our experiments, we found Batch-Normalisation to outperform other normalisation techniques, this is similar to the findings mentioned in [18]. Figure 3 shows a sample `EASTER` block. Table 1 shows the structure of a 3x3 EASTER model. The table outlines the number of blocks, sub-blocks, number of filters and other hyper-parameters.

Table 1. EASTER 3x3: 9 blocks each consisting of 3 1-D Convolutional sub-blocks, 1 preprocessing block(B1) and 3 post processing blocks(B2-B4). Overall the model contains 14 layers with 1Million trainable parameters.

Block #	# of Sub-Blocks	Kernel	# of Filters	Dropout	Dilation	Stride
Preprocess-I (B1)	2	3	64	0.2	1	2
E1	3	3	128	0.2	1	1
E2	3	4	128	0.3	1	1
E3	3	6	128	0.3	1	1
Postprocess-I(B2)	1	7	256	0.4	2	1
Postprocess-II (B3)	1	1	512	0.4	1	1
Postprocess-III (B4)	1	1	|Vocab|	0	1	1

5.2. CTC and Weighed CTC Decoder

The Connectionist Temporal Classification (CTC) method [7] is used to train as well as infer results from our models. The characters (or vocabulary for our task of OCR/HTR) in the input image vary in width and the spacing. CTC enables us to handle such a task without the need to align input images and ground truth.

We denote the training dataset as $D = \{X, Y\}$, where X is the input image for transcription and Y is the label or ground truth. Assuming we have a vocabulary set L, then $Y = L^s$, where s represents label length. CTC generates outputs at every time step t. The final output contains a sequence of repeating consecutive characters with ϵ (denoting blank space) in between. Thus, we add another symbol ϵ representing a blank to the vocabulary set L.

$$L^+ = \{L \cup \epsilon\} \tag{5.1}$$

The objective is to minimise the negative log probability of obtaining Y given an input X, i.e.

$$Objective = -\Sigma_{(X,Y)\epsilon D} \log p(Y|X) \tag{5.2}$$

The final output is obtained by merging consecutive repeating characters delimited by ϵ. For instance, an output sequence like $1\epsilon bb\epsilon\epsilon a$ maps to $1ba$. To obtain such an output, we define a function γ which squeezes repeating characters to single occurrence and removes blanks (ϵ). Thus, γ is a function which maps the intermediate repeating sequence output (denoted as π) to the final output y.

$$p(y|X) = \Sigma_{\gamma(\pi)=y} \log p(\pi|X) \tag{5.3}$$

$$p(\pi|X) = \Pi_{t=1}^{T} y_{\pi_t}^{t} \tag{5.4}$$

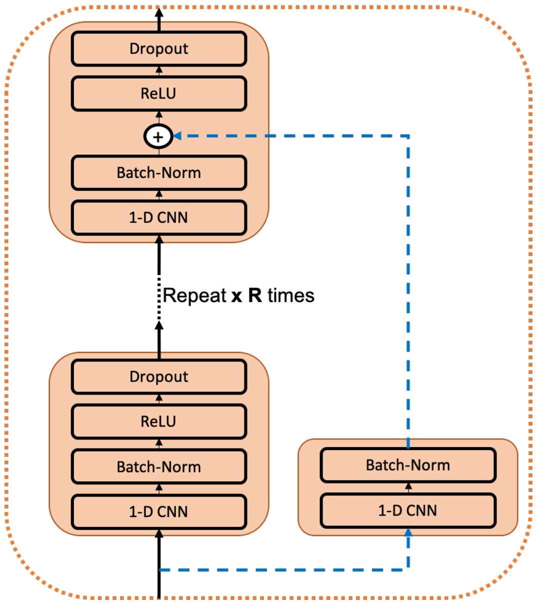


Figure 3. Components of EASTER block. Each block contains multiple repeating sub-blocks consisting of layers for 1-D Conv, Batch Normalisation, ReLU and Dropout. We also experiment with residual connections denoted with dotted blue line in our deepest 20-layer variant (EASTER 5x3). The R in figure denotes the number of sub-blocks used.

where $y^t_{\pi_t}$ is the probability of generating label π_t at time t. Thus, the predicted label y for input X is given as:

$$y = \gamma(argmax_{\pi} p(\pi|X)) \tag{5.5}$$

Due to the way this function is designed, the model generates far too many ϵ's as compared to actual characters. This leads to the model being biased towards the blank class (ϵ). To address this problem, [24] show multiple ways of adjusting the class weights for CTC. These weighting strategies address the problem of class imbalance and result in fast convergence. The class weighted CTC method is denoted as:

$$Class\_Weighted\_CTC(y|X) = -\Sigma_t \Sigma_k \alpha_k y^t_k \log y^t_k \tag{5.6}$$

where y^t_k is the generated output at time t and,

$$\alpha_k = \begin{cases} 1-\alpha & \text{if k} = \epsilon \\ \alpha & \text{otherwise} \end{cases} \tag{5.7}$$

In our experiments we saw significant improvement in performance with weighted CTC while training on small datasets. The most basic architecture for EASTER is shown in figure 4. It is a ExR architecture with separate preprocessing and post-processing blocks where E is the number of encoding blocks and R is the number of sub-blocks within each encoding block

5.3. Training

For a typical training input, we first transform the input image into grayscale followed by scaling it down to a height of 40-pixels. The network is able to handle variable width inputs and requires no additional transformations. Individual characters have specific local structures and we utilise overlapping 1-D convolutions to exploit the same. 1-D Convolutions also capture short term sequential dependencies across sliding frames and further assist

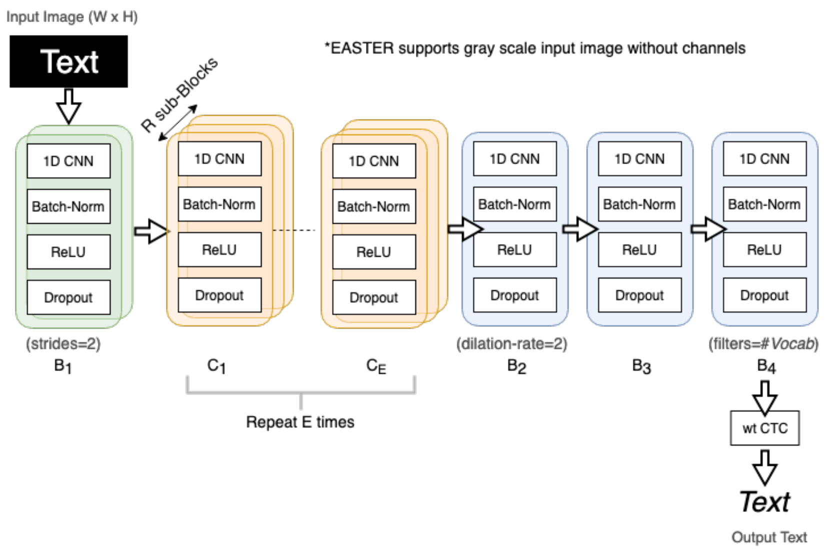


Figure 4. Basic ExR `EASTER` Architecture with 1 fixed preprocessing(B1) block and 3 fixed post-processing blocks(B2-B4)

in capturing the underlying temporal aspects of the sequence. `EASTER` block architecture enables it to learn higher level features without the need of recurrence or any specialised gated mechanism. We trained our models using Keras [25] with Tensorflow [26] backend.

The smallest model with half a million (0.5M) parameters achieves best results in under 1 hour of training.This enables faster turn-around time for retraining, experiments and ease of deployment. We experimented with two more variants, one with increased number of filters and the other with more depth. The deepest variant has 28M parameters and outperforms RNN based models with a good margin on our internal dataset.

6. Experiments and Results

6.1. Handwriting Task

We evaluate `EASTER`'s performance on IAM dataset for the task of HTR. To enable comparison with GRCL [13], we follow the same process of training our models on different combinations of the IAM dataset. [13] firstly train GRCL with IAM-offline train dataset which contains only 6161 lines of handwritten text and report results on the IAM-offline test dataset. We repeat the same experiment with `EASTER`-5x3 (20 layer variant) with residual connections and showcase improvements on both WER and CER by a good margin. We also went a step ahead and trained our model on only first 3000 samples out of 6161 samples i.e., only 50% of the actual training data. `EASTER`-5x3's performance on 3k samples was observed to be better than GRCL with 6161 training samples.

In the second experiment performed by [13], they concatenated smaller input lines in different combination to form a larger training dataset. The resulting training dataset has 511,524 training samples. This is a significantly larger training dataset as compared to the first experiment. We performed this experiment without the augmentation pipeline and observe a better performance from `EASTER`. Table 2 refers to the two experiments.

We present results on IAM-offline test data without any language/lexicon model to fairly compare with different network architectures. Our model shows lower CER than MDLSTM [11], similar to dilated temporal convolutions [27] and much lower than hybrid CNN-RNN

Table 2. Line-level recognition results on IAM Offline test Dataset as measured using Word Error Rate (WER) and Character Error Rate (CER). GRCL refers to Gated Recurrent Convolutional Layers as presented by Ingle et. al.[13]. All the results presented here are based on greedy decoding without any language/lexicon model.

Model	Training Dataset	# of Training Samples	Augmentation	WER	CER
`EASTER 5x3` (Ours)	IAM-Off	**3,000**	No	**33.3**	**14.0**
GRCL [13]	IAM-Off	6,161	No	35.2	14.1
`EASTER 5x3` (Ours)	IAM-Off	6,161	No	**25.4**	**9.8**
GRCL [13]	IAM-Off + IAM-On-Long	511,524	No	22.3	8.8
`EASTER 5x3` (Ours)	IAM-Off + IAM-On-Long	**24,481**	**No**	**20.6**	**7.9**
GRCL [13]	IAM-Off + IAM-On-Long	511,524	Yes	17.0	**6.7**

Table 3. Comparison of Line-level recognition results on IAM Offline test Dataset as measured using Word Error Rate (WER) and Character Error Rate (CER). All the results presented here are based on greedy decoding without any language/lexicon model.

Model	Augmentation	WER	CER
Chowdhury and Vig (2018) [28]	Yes	21.1	11.4
Krishnan et al. (2018) [29]	Yes	32.8	9.7
Voigtlaender et al. (2016) [11]	Yes	27.5	8.3
Sharma et al. (2021) [27]	Yes	23.0	**7.9**
Ingle et el. (2019) [13]	**No**	22.3	8.8
`EASTER 5x3` (Ours)	**No**	**20.6**	**7.9**

([28] and [29]). All our results (Table 3) are based on limited training data and without any augmentations as described above.

We observed similar performance boost on internal datasets. The improvements in WER and CER along with need for lesser training data and a lighter model (in terms of trainable parameters and memory footprint) helped us meet production requirements as well.

6.2. **Machine Printed Tasks**

We performed multiple separate experiments using `EASTER` for the task of OCR as well. The first experiment was based on the dataset prepared using our data generation pipeline. We utilised this dataset to benchmark our performance for internal datasets/use cases.

The second set of experiments involved preparing our model for performance on some of the benchmarking datasets for text extraction from natural images. Popular benchmarking datasets we used for our experiments are as follows:

- **IIIT-5k** [30]: This dataset consists of 2000 training images collected from the internet. There are about 3000 test samples. The dataset also consists of 50 and 100 word lexicon which we do not use during our experiments.
- **Google-SVT** [31]: is another interesting dataset consisting of only 257 training images and 647 images for test. Unlike [32], we do not use the lexicon for our experiments.

The models with best results on the benchmark datasets utilise a range of techniques to train. Since the volume of training data in these datasets is not enough, **Synth-90k** [33] dataset is used as a training set. As the name implies, this dataset generates realistic synthetic images. It consists of 900k test images and about 7million for training. Other techniques like preprocessing input crops to handle skew, rotation, contrast, super-resolution etc. are also applied by some of the works. For our benchmarking experiments we only make use of **Synth-90k** dataset for training our models. We do not apply any preprocessing

Table 4. Machine printed natural text recognition word accuracies. All results are in unconstrained setting, i.e. models do not use lexicons/language models during decoding. `EASTER` 5x3 in particular uses only greedy decoding in contrast to others in the table

Model	SVT	IIIT-5k
Jaderberg et.al.[16]	71.1%	-
Lee et. al.[17]	80.7%	78.4%
Baoguang et. al.[15]	80.8%	78.2%
Wang et. al.[32]	**81.5%**	80.8%
`EASTER`-5x3	78.5%	**86.76%**

techniques other than resizing. Our experimental setup consists of a 20 layer deep variant, i.e. `EASTER`-5x3), model with residual connections. This model has 28million trainable parameters and a vocabulary of 62 characters from the English language (26 lower-case + 26 Upper-case + 10 numerals).

We did not apply any augmentations to our training dataset and performed greedy decoding. No additional post processing steps, like usage of language models, etc. were applied. There were two major reasons behind experimenting without a language model. Firstly, training language model requires additional time and data which is not always available. Secondly, and more importantly, the usage of language model slows down the inference pipeline in practice. Since our aim was to develop a deployable and usable OCR/HTR model, we experimented without a language model for `EASTER`.

Compared to [15–17], `EASTER` comfortably improves upon the WER and CER performance against the **IIIT-5k** dataset. Our model is able to achieve 86.76% word accuracy with 4.56% character error rate. For case of **Google-SVT**, our model achieves a word accuracy of 78.51% with a 9.7% character error rate. We did not observe any improvements while using the training images from **Google-SVT** for fine-tuning. Our model improves upon the best in case of **IIIT-5k** while nearly achieves benchmark results on **Google-SVT**. It is important to note that in both cases, the inference was done with a greedy decoder without relying on the lexicon. The results and comparison are shown in detail in table 4 for reference.

7. Limitations and Discussion

Although our work can achieve compelling results in many cases, the results are not so favourable in some scenarios. Figure 5 highlights some of the failure modes for IIIT-5k and Google-SVT datasets. We observed that even though the transcriptions do not match ground truth, the outputs seem to be honest mistakes. These mistakes can be largely attributed to issues like bad image quality along with distortions which are completely transforming certain characters. For instance, consider the distortions of characters "f" and "G" in figure 5(b) and figure 5(e) respectively. The characters have been distorted to such an extent that it is nearly impossible to pick the correct character without extensive post-processing. There are also cases where specific fonts seem to confuse the model (see figure 5(f) and 5(g)). While distortions are tricky to handle, font related issues can be tackled using more training examples.

The handwritten text recognition task is slightly more complex than the machine printed case. Though `EASTER` outperforms benchmarks both in terms of performance as well as training and compute requirements, it does face challenges when presented with overlapping and highly cursive handwritings. Figure 6 presents a few failure modes. Most issues in the highlighted examples can be attributed to unintelligible scribbles or at times hard to distinguish shapes. For instance, in the second example in figure 6, the word "our" is misread

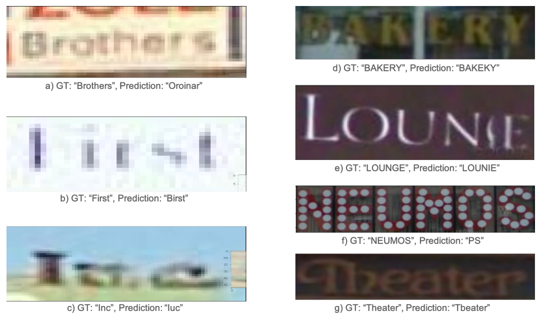
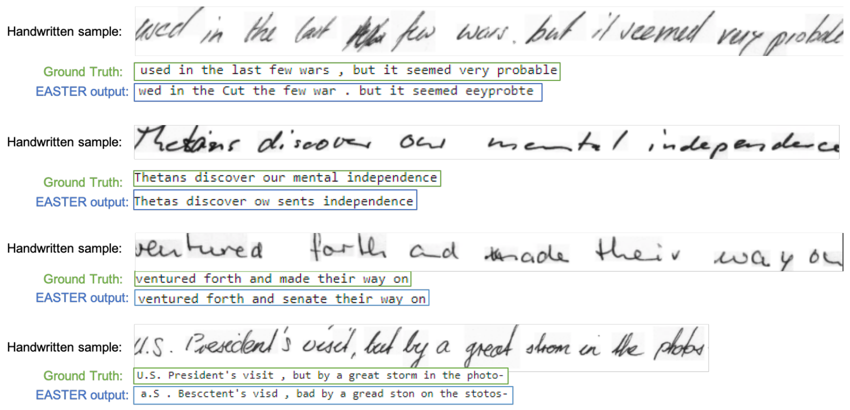

a) GT: “Brothers”, Prediction: “Oroinar”
b) GT: “First”, Prediction: “Birst”
c) GT: “Inc”, Prediction: “Iuc”
d) GT: “BAKERY”, Prediction: “BAKEKY”
e) GT: “LOUNGE”, Prediction: “LOUNIE”
f) GT: “NEUMOS”, Prediction: “PS”
g) GT: “Theater”, Prediction: “Tbeater”

Figure 5. Samples where `EASTER` fails to transcribe correctly. Each example consists of ground truth followed by model output. Samples (b) and (e) can be attributed to distortions while samples (f) and (g) are associated with font related issues.)



```
Ground Truth:  used in the last few wars , but it seemed very probable
EASTER output: wed in the Cut the few war . but it seemed eeyprobte

Ground Truth:  Thetans discover our mental independence
EASTER output: Thetas discover ow sents independence

Ground Truth:  ventured forth and made their way on
EASTER output: ventured forth and senate their way on

Ground Truth:  U.S. President's visit , but by a great storm in the photo-
EASTER output: a.S . Bescctent's visd , bad by a gread ston on the stotos-
```

Figure 6. `EASTER` for handwritten text. Samples showcase scenarios where overlapping and highly cursive handwriting lead to incorrect transcriptions.

as "ow" which is again a good enough guess given that the model is not making use of any language model or other post-processing steps.

8. Practical Considerations

As mentioned earlier, our objective was to have a high performing model along with consideration of training data volume. Since, this model is required for actual production use, a secondary objective was to reduce memory footprint within acceptable performance degradation. The results section showcased how our model is able to produce near-state-of-the-art performance with only a fraction of training data(see table 2). We experimented with different quantization techniques as well to reduce the memory footprint. Our 20-layer deepest variant (with residual connections) reduces to 28MB after applying Dynamic Range Quantization[34, 35] technique. This results in a decrease of 2% in word accuracy on the benchmarking dataset. This degradation was within acceptable range for our practical use.

9. Conclusion and Future Work

We presented a fast, scalable and recurrence free architecture called `EASTER` for handwritten and machine printed text recognition tasks. Inspired from fully convolutional architectures for automatic speech recognition, we discussed the building blocks and training process for EASTER. We described the dataset preparation process for both handwritten and machine printed text. We also discussed about the impact of weighted CTC towards faster convergence. We finally presented results on benchmarking datasets for both HTR and OCR tasks. EASTER is able to achieve significant improvements in WER and CER. EASTER's performance on internal datasets achieved near state of the art performance. We also showcased that the model performs equally well on line level data. Due to lesser number of trainable parameters and eventually smaller memory footprint, EASTER is easier to train and faster to use in production applications without much tooling. As part of future work we plan to utilise attention mechanisms to improve the decoding stage.

Acknowledgements

We would like to thank Kishore V Ayyadevara and Yeshwanth Reddy for their work and contributions to the OCR project and making sure it gets widespread adoption, Vineet Shukla for helpful discussions and inputs to improve the solution, and the whole OCR team for their contributions.

References

[1] S. Espana-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez. "Improving offline handwritten text recognition with hybrid HMM/ANN models". In: *IEEE transactions on pattern analysis and machine intelligence* 33.4 (2010), pp. 767–779.

[2] S. Mori, H. Nishida, and H. Yamada. *Optical character recognition*. John Wiley & Sons, Inc., 1999.

[3] D. Castro, B. L. D. Bezerra, and M. Valenca. "Boosting the Deep Multidimensional Long-Short-Term Memory Network for Handwritten Recognition Systems". In: *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. Los Alamitos, CA, USA: IEEE Computer Society, Aug. 2018, pp. 127–132. DOI: 10.1109/ICFHR-2018.2018.00031. URL: https://doi.ieeecomputersociety.org/10.1109/ICFHR-2018.2018.00031.

[4] P. Doetsch, M. Kozielski, and H. Ney. "Fast and Robust Training of Recurrent Neural Networks for Offline Handwriting Recognition". In: *Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR* 2014 (Dec. 2014), pp. 279–284. DOI: 10.1109/ICFHR.2014.54.

[5] A. Graves and J. Schmidhuber. "Offline handwriting recognition with multidimensional recurrent neural networks". In: *Advances in neural information processing systems*. 2009, pp. 545–552.

[6] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. "A novel connectionist system for unconstrained handwriting recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 31.5 (2008), pp. 855–868.

[7] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks". In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 369–376.

[8] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. "Dropout improves recurrent neural networks for handwriting recognition". In: *2014 14th International Conference on Frontiers in Handwriting Recognition*. IEEE. 2014, pp. 285–290.

[9] A. Poznanski and L. Wolf. "CNN-N-Gram for HandwritingWord Recognition". In: *CNN-N-Gram for HandwritingWord Recognition*. June 2016, pp. 2305–2314. DOI: 10.1109/CVPR.2016.253.

[10] F. Borisyuk, A. Gordo, and V. Sivakumar. "Rosetta". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018). DOI: 10.1145/3219819.3219861. URL: http://dx.doi.org/10.1145/3219819.3219861.

[11] P. Voigtlaender, P. Doetsch, and H. Ney. "Handwriting recognition with large multidimensional long short-term memory recurrent neural networks". In: *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE. 2016, pp. 228–233.

[12] R. Smith. "An overview of the Tesseract OCR engine". In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. Vol. 2. IEEE. 2007, pp. 629–633.

[13] R. R. Ingle, Y. Fujii, T. Deselaers, J. Baccash, and A. C. Popat. "A scalable handwritten text recognition system". In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE. 2019, pp. 17–24.

[14] D. Coquenet, Y. Soullard, C. Chatelain, and T. Paquet. "Have Convolutions Already Made Recurrence Obsolete for Unconstrained Handwritten Text Recognition?" In: *Have Convolutions Already Made Recurrence Obsolete for Unconstrained Handwritten Text Recognition?* Sept. 2019, pp. 65–70. DOI: 10.1109/ICDARW.2019.40083.

[15] B. Shi, X. Bai, and C. Yao. "An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition". In: *CoRR* abs/1507.05717 (2015).

[16] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. "Deep structured output learning for unconstrained text recognition". In: *arXiv preprint arXiv:1412.5903* (2014).

[17] C. Lee and S. Osindero. "Recursive Recurrent Nets with Attention Modeling for OCR in the Wild". In: *CoRR* abs/1603.03101 (2016).

[18] J. Li, V. Lavrukhin, B. Ginsburg, R. Leary, O. Kuchaiev, J. M. Cohen, H. Nguyen, and R. T. Gadde. *Jasper: An End-to-End Convolutional Neural Acoustic Model*. 2019. arXiv: 1904.03288 [eess.AS].

[19] R. Collobert, C. Puhrsch, and G. Synnaeve. "Wav2Letter: an End-to-End ConvNet-based Speech Recognition System". In: *CoRR* abs/1609.03193 (2016). arXiv: 1609.03193. URL: http://arxiv.org/abs/1609.03193.

[20] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. "Natural Language Processing (Almost) from Scratch". In: *J. Mach. Learn. Res.* 12.null (Nov. 2011), pp. 24932537. ISSN: 1532-4435.

[21] U.-V. Marti and H. Bunke. "The IAM-database: an English sentence database for offline handwriting recognition". In: *International Journal on Document Analysis and Recognition* 5 (2002), pp. 39–46.

[22] A. Graves. *Generating Sequences With Recurrent Neural Networks*. 2013. arXiv: 1308.0850 [cs.NE].

[23] V. Pratap, A. Hannun, Q. Xu, J. Cai, J. Kahn, G. Synnaeve, V. Liptchinsky, and R. Collobert. "wav2letter++: The Fastest Open-source Speech Recognition System". In: *CoRR* abs/1812.07625 (2018). arXiv: 1812.07625. URL: http://arxiv.org/abs/1812.07625.

[24] H. Li and W. Wang. "A Novel Re-weighting Method for Connectionist Temporal Classification". In: *arXiv preprint arXiv:1904.10619* (2019).

[25] F. Chollet et al. *Keras*. https://keras.io. 2015.

[26] M. Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[27] A. Sharma and D. B. Jayagopi. "Towards efficient unconstrained handwriting recognition using Dilated Temporal Convolution Network". In: *Expert Systems with Applications* 164 (2021), p. 114004.

[28] A. Chowdhury and L. Vig. "An efficient end-to-end neural model for handwritten text recognition". In: *arXiv preprint arXiv:1807.07965* (2018).

[29] P. Krishnan, K. Dutta, and C. Jawahar. "Word spotting and recognition using deep embedding". In: *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. IEEE. 2018, pp. 1–6.

[30] A. Mishra, K. Alahari, and C. V. Jawahar. "Scene Text Recognition using Higher Order Language Priors". In: *BMVC*. 2012.

[31] K. Wang, B. Babenko, and S. Belongie. "End-to-end scene text recognition". In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 1457–1464.

[32] J. Wang and X. Hu. "Gated Recurrent Convolution Neural Network for OCR". In: *Advances in Neural Information Processing Systems*. 2017.

[33] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. *Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition*. 2014. arXiv: 1406.2227 [cs.CV].

[34] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. *Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference*. 2017. arXiv: 1712.05877 [cs.LG].

[35] R. Krishnamoorthi. *Quantizing deep convolutional networks for efficient inference: A whitepaper*. 2018. arXiv: 1806.08342 [cs.LG].