Performance Analysis of PyTesseract and EasyOCR for Bangla Optical Character Recognition on the Novel Bangla CrossHair Dataset

Abdulla Nasir Chowdhury
Department of Electrical and Electronic Engineering
Leading University
Sylhet, Bangladesh
abdullahnasirchowdhury1@gmail.com

Aftar Ahmad Sami
Department of Computer Science and Engineering
Leading University
Sylhet, Bangladesh
aftarahmadsami@gmail.com

Shakib Absar
Department of Computer Science and Engineering
Leading University
Sylhet, Bangladesh
sabsar42@gmail.com

Shah Masud Parvej Mamun
Department of Electrical and Electronic Engineering
Leading University
Sylhet, Bangladesh
smpmamun2000@gmail.com

Fuad Rahman
Department of Software Engineering
Taylor's University
Kualalampur, Malaysia
fuadrahman185@gmail.com

* Md. Saidur Rahman Kohinoor
Department of Information and Computer Science
King Fahd University of Petroleum and Minerals
Dhahran, Saudi Arabia
g202391630@kfupm.edu.sa

Abstract— **This paper presents a comparative study of key metrics for OCR engines in Bangla language processing. PyTesseract (Tesseract OCR) and EasyOCR were benchmarked on a novel dataset, "Bangla-CrossHair," created for testing OCR engines. This dataset combines samples from "Bangla Text Detection and Recognition," "Bangla Handwritten Characters," and "Bangla Handwritten Words" datasets and includes diverse image types, such as blurred, clear, torn, and tilted. Results show EasyOCR outperforms PyTesseract in several scenarios, while PyTesseract consistently demonstrated faster processing. Since many OCR engines provide pre-trained capabilities, traditional metrics like training and validation accuracy are challenging to measure for some models, including PyTesseract. Instead, metrics like Character Level Accuracy, Word Level Accuracy, Levenshtein Distance, Character Error Rate, Word Error Rate, Precision, Recall, and F1-Score were used for unbiased evaluation.**

Keywords—Bangla-CrossHair, PyTesseract, EasyOCR, Levenstein Distance, Comparison.

I. Introduction

OCR or Optical Character Recognition is the identification of textual data from a graphic. OCR Engines are algorithms developed to satisfy just that. Today OCR Engines have found usage in a multitude of machine learning applications. These include but are not limited to document digitization, data extraction (banking and finance, extraction of data from invoices, receipts, etc.), license plate recognition, content indexing (matching signatures, diagrams, etc.), translation and localization, text-to-speech applications, security and surveillance, medical records, document forgery detection, and many more [1].

A primitive approach in OCR Engines was to utilize template matching which is still applicable in modern day engines like the Tesseract OCR. However, this method when used raw has proven itself to be highly inefficient as can be observed in a study where detection is necessary from unconventional text images i.e., non-document graphics [4]. Today, the most popular OCR Engines are: Easy OCR,
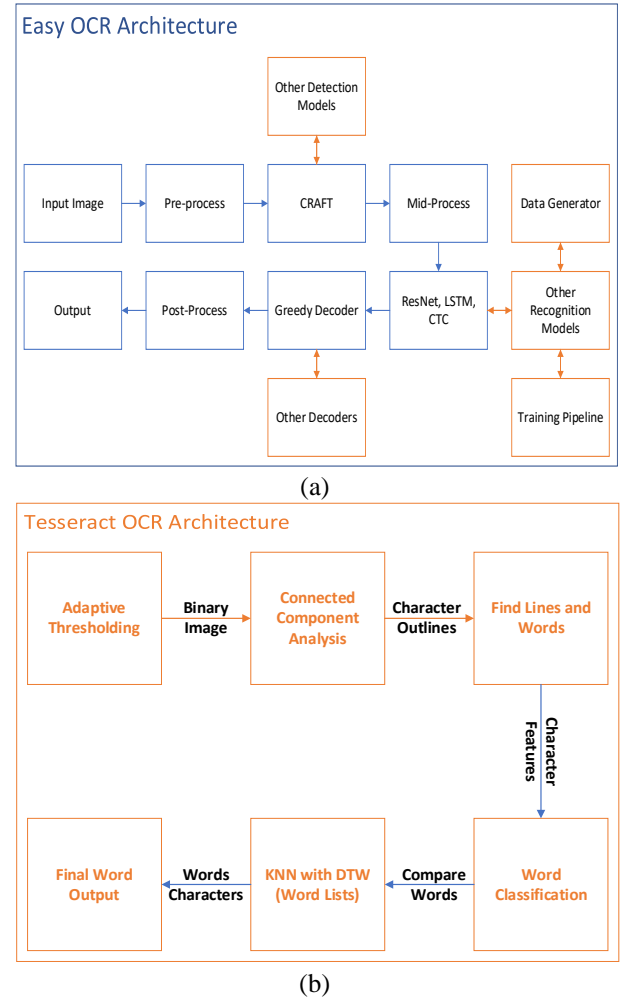

(a)

(b)

Fig. 1.　　Architectures of OCR Engines: (a) Easy OCR [2], (b) Tesseract OCR, KNN (K-Nearest Neighbors), DTW (Dynamic Time Warping) [3]
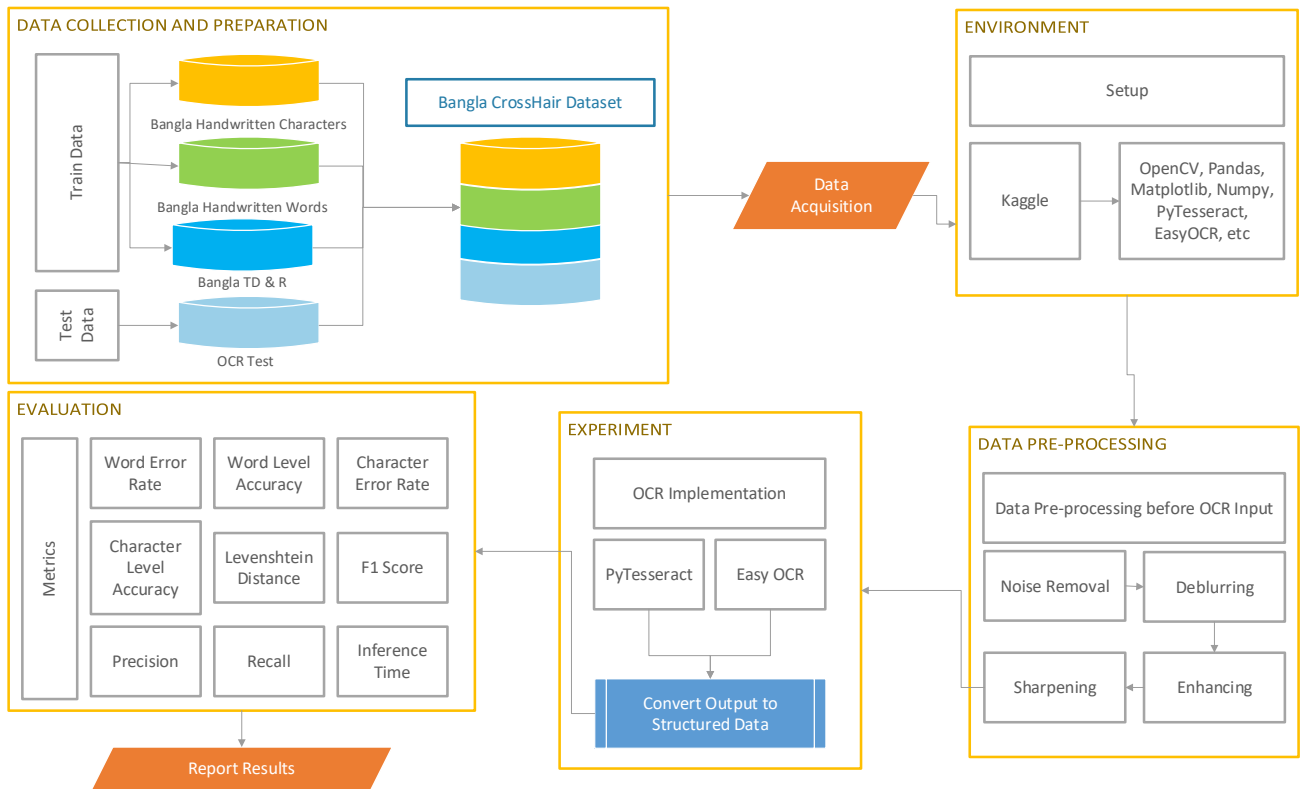
Fig. 2.   Methodology Flowchart

Tesseract OCR and Paddle OCR. Easy OCR and Paddle OCR have become highly popular due to ease of use and precision.

The data, codes and implementations are available in our public notebooks and repository on **Kaggle** and **Github** respectively [19], [20]. In this research study we strived to impartially benchmark the OCR Engines: EasyOCR and PyTesseract on the Bangla-CrossHair dataset available in Kaggle. The subsequent sections of this paper are organized as follows: Section II provides a comprehensive literature review, Section III the implemented methodology for our analysis, Section IV presents the results and evaluation of the outcome and finally, Section V covers the discussion and conclusion followed by the references.

II. LITERATURE REVIEW

Easy OCR makes use of popular deep learning algorithms for image classification, more appropriately convolutional neural networks like Residual Network and VGG paired with recurrent neural networks mainly LSTM (Long Short Term Memory) Networks and CTC (Connectionist Temporal Classification) as building blocks of it's architecture. It also makes use of the Greedy Decoder Algorithm before post processing and CRAFT (Character Region Awareness For Text Recognition) after pre-processing. The CRAFT Algorithm is highly efficient as it produces character level bounding boxes making it easier for the RNNs and CNNs to detect the words and characters of interest. In place of CRAFT an FPN (Feature Pyramid Network) or an FCN (Fully Convolutional Network) might find it's use to detect the text region. The logical conclusion is that the architecture of Easy OCR is fairly complex and heavily reliant on deep neural networks which contribute to slower performance in comparison to other OCR engines. However, Easy OCR provides highly accurate results when implemented for character recognition and performs well independent of

whether it is on a document or not [5]. It is also comparatively easier to use. The problem with Easy OCR is probably the limited contextual understanding which is common to every OCR and it's heavy dependence on image quality but these can be mitigated by training the engine on such challenging cases and pre-processing the input image respectively.

The architecture of Tesseract OCR is studied which, employs a much simpler architecture in contrast to Easy OCR [6]. In this OCR engine, only the LSTM RNN is utilized. This architecture relies heavily on traditional pre-processing and post-processing techniques such as Adaptive Thresholding, conversion of RGB to Binary, and Connected Component Analysis. Furthermore, KNN with DTW (Dynamic Time Warping) find their use case in post-processing. The simplicity of it's architecture makes it perform poorly when tasked with analyzing the characters from any uneven surface and those that aren't traditional documents. However, due to it's simplicity and earlier access, it found significant application in Bangla Optical Character Recognition and it's performance has been studied in several papers. It is especially popular in the detection of characters from license plates in Bangladesh [7] – [11]. The application of this algorithm is widespread and very satisfactory as can be seen from previous studies.

The comparison between their architectures is observed in Fig.1 (a) and (b). PyTesseract unlike Easy OCR allows users to fine-tune it easily so it can be used for fonts that are not available in the pre-trained version. This is a big plus for PyTesseract and one of the reasons behind its popularity in the computer vision community. Another pivotal reason behind it's popularity is it's simple architecture which consequently contributes to faster processing speeds. Due to the low graphics requirements of this OCR Engine, those getting started with this topic find PyTesseract more user-friendly. Another popular OCR engine is studied in [12], where the

architecture of PP-OCR (Paddle-Paddle Optical Character Recognition) but more commonly called Paddle OCR is studied. It utilizes CNN, RNN and transformer based models which makes it highly capable in separating text from uneven surfaces as seen in [13]. In [14] the architecture of OCRopus is studied along with it's performance on a characters dataset. OCRopus utilizes LSTM which is found similar to the architecture of Kraken OCR [15], which was developed for reading historical texts is also studied. The similarity stems from the fact that Kraken was developed as an improved fork of OCRopus. The improvement lies in the fact that, Kraken uses CNN in combination with LSTM. A study of Kraken's performance on historic Arabic texts was provided in [16]. Due to the absence of Bangla language support in these OCR engines, they did not find any application in our study. Furthermore, a novel OCR Engine was developed et al. Abdulla integrating the linear SVM classifier into the last layer of a ResNet50 network after training it on his custom dataset [17], however their engine could have benefited from a comparative analysis against PyTesseract and EasyOCR which could have provided insight into the performance of his proposed model.

III. METHODOLOGY

In order to ensure that the benchmarking of the engines is actually useful to the research community on Bangla OCR, it is crucial to be as transparent as possible. We followed several steps in order to make sure that this was not only unbiased but also up to acceptable standards. The steps have been visualized in Fig. 2. and explained in the following sections.

A. Data Preparation and Exploratory Data Analysis
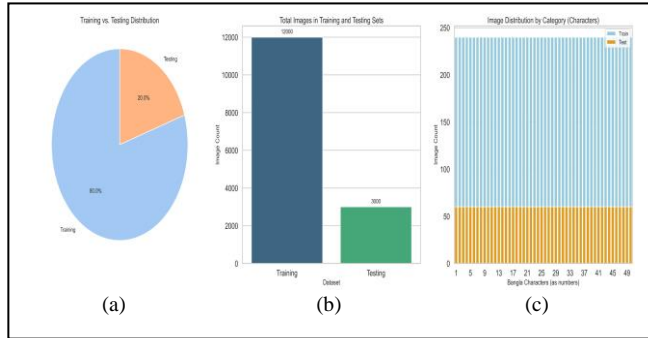

Fig. 3.   Bangla characters folder data analysis: (a) 80:20 Split, (b) Total Images per category (c) Data Balance
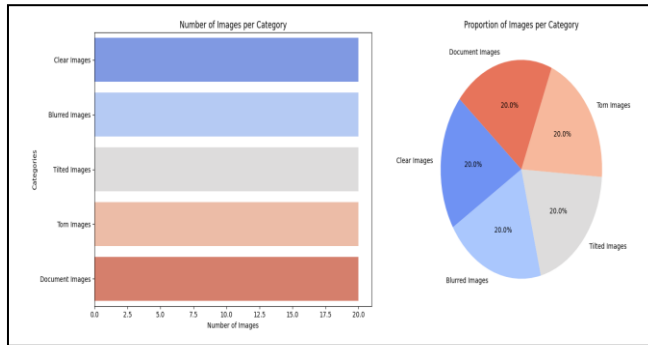

Fig. 4.   OCR Engine test folder data analysis (20 images per category depicting a balanced dataset for testing).

Preparation of Bangla-CrossHair Dataset: Upon acquiring the Bangla Handwritten Characters dataset from Kaggle we split the images into train-test-split with 12,000 training images and

3000 testing images. We perform EDA (Exploratory Data Analysis) on the dataset as can be seen in Fig. 3. Fig.4. delineates the status of the OCR Test folder while performing the tests with 20 images per category: blurred, clear, document, torn and tilted images. The bangla words folder contains a set of words in bangla to train the data on. The histogram of the distribution of image heights and widths is depicted in Fig. 5. It is derived from this information that the
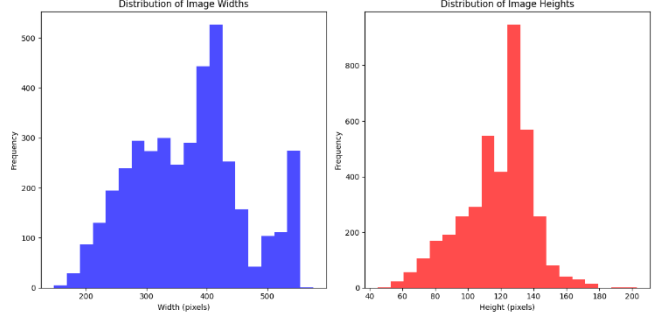
distribution is gaussian.

Fig. 5.   Bangla words folder data analysis

At the time of this study, this novel dataset stands at around 300 MB allowing it to be easily usable. It is available on Kaggle under the name "Bangla-CrossHair," so that it can serve as the common publicly available dataset for benchmarking OCR models. Further data will continue to be added to improve the robustness of the dataset. A sample of the dataset has been provided in Fig. 6.
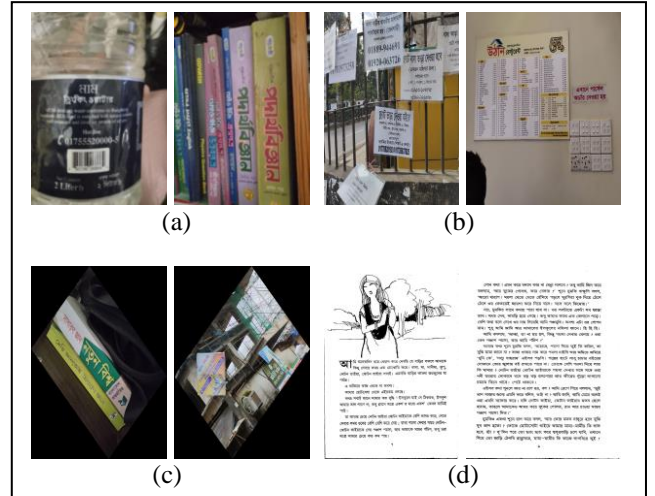

Fig. 6.   Sample Images of Bangla-CrossHair Dataset; (a) Blurred Images (b) Crisp Images (c) Tilted Images (d) Document Images

B. Setup

Installation of required libraries on Kaggle was done followed by preprocessing of the images (deblurring, sharpening, etc.) using OpenCV for the OCR Engine input. All experiments were performed on Kaggle. No GPU was used in order to measure time required impartially and because training the engines is beyond the scope of this paper.

C. Studied Metrics

The following metrics were studied:
1) Word Level Accuracy and Word Error Rate:

WLA (Word Level Accuracy) metric is calculated through a comparison of the words recognized by the OCR Engine with the ground truth words by calculating

the percentage of correctly recognized words among all the words in the ground truth text. The formula for WLA:

$$WLA = \frac{CRW}{N_w} \qquad (1)$$

Where;

CRW = number of correctly recognized words,

N_w = total number of words in ground truth text

Another method of calculating WLA is by subtracting the value of WER from 1. The equation is as follows:

$$WLA = 1 - WER \qquad (2)$$

Where;

WER = Word Error Rate[18]

The WER metric is used to measure the percentage of words that are incorrectly recognized or translated by the OCR Engine compared to a reference (ground truth) text. To calculate the word error rate for our case, word level-edit operations i.e., the summation of insertions, deletions and substitutions are mandatory. To calculate WER, the word level edit operations required to convert the engine detected text into ground truth text is normalized by the total number of words in the ground truth text.

$$WER = \frac{S + D + I}{N_w} * 100 \qquad (3)$$

Where:

S = number of word substitutions (words in the recognized text that are different from the ground truth text).

D = number of word deletions (words missing in the recognized text but present in the ground truth text).

I = number of word insertions (extra words present in the recognized text but not in the ground truth text).

The number (S+D+I) represents the total number of word level errors. WER can then be expressed as a percentage. Note that,

$$WER + WLA = 1 \qquad (4)$$

2) *Character Level Accuracy and Character Error Rate:*

CLA is measured by the percentage of characters that are correctly recognized by the engine compared to our reference (ground truth) text.

$$CLA = \frac{CRC}{N_c} \qquad (5)$$

Where:

CRC = number of correctly recognized characters

N_c = total number of characters in ground truth text

Similar to WLA, CLA provides an indication of the accuracy of our OCR Engine's character recognition capabilities, taking into account insertion, deletion and substitution errors. Obviously higher character level accuracy denotes fewer discrepancies between system output and reference text i.e., better performance.

In the same manner as WLA, CLA can also be calculated by the following formula:

$$CLA = 1 - CER \qquad (6)$$

Where;

CER = Character Error Rate [18]

The purpose of CER metric is to measure the proportion of characters that are incorrectly recognized by the OCR Engine compared to the ground truth text and is given by the formula:

$$CER = \frac{E}{N_c} * 100 \qquad (7)$$

Where;

E = (S + D + I), is total number of errors, which is the sum of insertions, deletions and substitutions.

As we know,

$$CER + CLA = 1$$
$$\Rightarrow \quad CER = 1 - CLA \qquad (8)$$

Where;

CLA = Character Level Accuracy [18]

3) *Levenstein Distance (Edit Distance):*

It is also known as the edit distance and is a measure of the similarity between two strings. it is given by the formula:

$$\text{lev}(a, b)$$
$$= \begin{cases} |a| & ; & \text{if } |b| = 0 \\ |b| & ; & \text{if } |a| = 0 \\ \text{lev}\big(\text{tail}(a), \text{tail}(b)\big) & ; \text{if head}(a) = \text{head}(b) \\ 1 + \min \begin{cases} \text{lev}(\text{tail}(a), b) \\ \text{lev}\big(a, \text{tail}(b)\big) \\ \text{lev}\big(\text{tail}(a), \text{tail}(b)\big) \end{cases} & \text{otherwise} \end{cases}$$
$$(9)$$

The edit distance between two strings s and t is calculated using dynamic programming and is defined by the above recursive formula. The higher the Levenstein distance the worse performance of the OCR Engine, as the minimum number of edits required to convert s into t is measured by the edit distance [18]. In our case, s will be the OCR Engine output and t will be the ground truth text.

4) *Precision, Recall, F1-score:*

The ability to avoid false positives by a system is determined by it's precision. Generally, higher precision means better performance. The formula is given by:

$$\text{Precision} = \frac{TP}{TP + FP} \qquad (10)$$

Where;

TP = True Positives

FP = False Positives [18]

Recall, also known as sensitivity or true positive rate, is used to measure the proportion of correctly predicted positive instances among all actual positive instances in the dataset. The formula is given by:

$$\text{Recall} = \frac{TP}{TP + FN} \qquad (11)$$

Where:

FN = False Negatives [18]

The F1 score is the harmonic mean of precision and recall. To trade-off between precision and recall, F1 score is used. [24] The formula used to measure the F1 score is:

$$\text{F1 Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \qquad (12)$$

Here, output comparison, metric evaluation and inference time are studied on the same sample at different instances.

A. Output Comparison

TABLE I.   OUTPUT VISUALIZATION

Type	Image	PyTesseract	EasyOCR
Blurred			
Crisp			
Tilted			
Doc			

Table I illustrates that PyTesseract could neither detect nor classify text from challenging instances of images that are not regular documents while EasyOCR seamlessly detected the text locations with misclassifications. Improvement in EasyOCR's performance is possible if it is trained on such challenging instances which is beyond the scope of this paper.

B. Metrics Evaluation

TABLE II.   PYTESSERACT METRIC EVALUATION

PyTesseract					
Image Instances	WER	WLA	CER	CLA	LD
	100%	0%	100%	0%	182
	79.20%	20.80%	61.90%	38.10%	1035
	100%	0%	100%	0%	580
	55.60%	**44.40%**	**51.16%**	**48.84%**	**596**

For the metrics, WER, WLA, CER, CLA and LD of the OCR Engines are studied on a sample of the whole dataset to display on this paper in Tables II for PyTesseract and III for Easy OCR. Bold indicates better performance than it's counterpart. PyTesseract scores better only in LD for documents as can be seen from Table III and Table IV.

TABLE III.   EASYOCR METRIC EVALUATION

EasyOCR					
Image Instances	WER	WLA	CER	CLA	LD
	100%	**0%**	**98.35%**	**1.65%**	**179**
	60.58%	**39.42%**	**39.17%**	**60.83%**	**655**
	97.89%	**2.11%**	**80.86%**	**9.14%**	**469**
	61.44%	38.56%	53.39%	46.61%	622

All the metrics including precision, recall and F1 score have been measured on the entire dataset and the average performance of each has been shared here and can be seen in Table IV. Bold indicates better performance than it's counterpart in Table IV.

TABLE IV.   SUMMARY OF OCR TEST (AVERAGE)

EASY OCR								
Sl.no	WER	WLA	CER	CLA	LD	P	R	F1
1	29.5	70.5	13.1	86.8	187.7	91.4	93.8	92.6
2	**93.3**	**6.7**	**74.7**	**25.3**	**489.3**	**53.3**	**45.2**	**45.9**
3	**99.6**	**0.4**	**95.8**	**4.2**	**679.4**	**40.4**	**10.7**	**15.6**
4	**98.6**	**1.4**	**81.4**	**19.6**	**548.3**	**52.9**	**43.6**	**45.6**
5	**94.8**	**5.2**	**75.5**	**24.5**	**451.2**	**49.7**	**44.4**	**44.6**
PyTesseract								
1	**14.05**	**85.95**	**6.32**	**93.8**	**81.6**	**93.1**	**96.8**	**94.9**
2	94.37	5.63	88.6	11.4	**488.1**	36.8	25.2	26.4
3	100	0	99.4	0.6	688.2	13.0	3.12	4.68
4	100	0	100	0	692.8	0	0	693
5	100	0	88.95	11.0	464.1	34.7	29.7	29.4

Here, 1, 2, 3, 4, 5 stands for document images, clear images, blurred images, tilted images and torn images respectively.

C. Inference Time Comparison

Figures 7(a) and 7(b) depicts the inference time of the engines for 20 images over 5 categories. The average inference time over a 100 data points is presented in Table V.
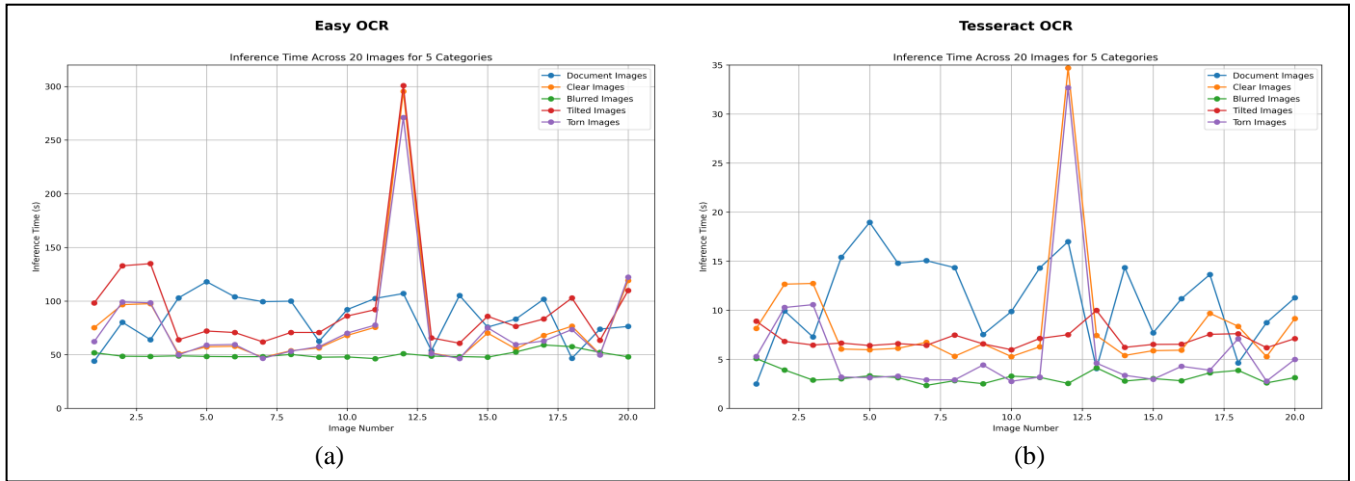
Fig. 7. Inference Time Comparison: (a) Easy OCR Inference Time (b) Tesseract OCR Inference Time

TABLE V. AVERAGE INFERENCE TIME COMPARISON

Instance	Easy OCR	PyTesseract
Document Images	88.53 (s)	**11.21 (s)**
Crisp Images	80.82 (s)	**8.68 (s)**
Blurred Images	52.31 (s)	**3.19 (s)**
Tilted Images	98.52 (s)	**7.03 (s)**
Torn Images	79.57 (s)	**5.93 (s)**

V. DISCUSSION AND CONCLUSION

We put an end to the long debate on which OCR Engine is better for Bangla OCR between PyTesseract and Easy OCR. The tests performed here can be reproduced easily as the dataset is available on Kaggle. It can be strongly concluded that while PyTesseract struggles to detect characters in challenging instances, EasyOCR can detects the text with the competitive accuracy. The success of EasyOCR can be contributed to it's deep learning architecture. PyTesseract is almost 10 times faster than EasyOCR as can be seen from the inference time analysis. This study also proves PyTesseract to be better than EasyOCR at extracting text from documents.

REFERENCES

[1] R. Raj and A. Kos, "A Comprehensive Study of Optical Character Recognition," 2022 29th International Conference on Mixed Design of Integrated Circuits and System (MIXDES), Wrocław, Poland, 2022, pp. 151-154, doi: 10.23919/MIXDES55591.2022.9837974.

[2] J. Doe, "EasyOCR," Public GitHub repository. [Online] Available: https://github.com/JaidedAI/EasyOCR, 2020.

[3] S. Schmidt, "Pytesseract," Public GitHub repository. [Online]. Available: https://github.com/madmaze/pytesseract, 2018.

[4] Swaroop, Paridhi & Sharma, Neelam. (2016). An Overview of Various Template Matching Methodologies in Image Processing. International Journal of Computer Applications. 153. 8-14. 10.5120/ijca2016912165

[5] D. R. Vedhaviyassh, R. Sudhan, G. Saranya, M. Safa and D. Arun, "Comparative Analysis of EasyOCR and TesseractOCR for Automatic License Plate Recognition using Deep Learning Algorithm," 2022 6th International Conference on Electronics, Communication and Aerospace Technology, Coimbatore, India, 2022, pp. 966-971, doi: 10.1109/ICECA55336.2022.10009215.

[6] Smith, Raymond W.. "An Overview of the Tesseract OCR Engine." Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) 2 (2007): 629-633.

[7] S. Chandra, M. A. -a. Nowshad, M. J. Islam and Marium-E-Jannat, "An automated system to detect and recognize vehicle license plates of Bangladesh," 2017 20th International Conference of Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 2017, pp. 1-6, doi: 10.1109/ICCITECHN.2017.8281781.

[8] M. N. I. Suvon, R. Khan and M. Ferdous, "Real Time Bangla Number Plate Recognition using Computer Vision and Convolutional Neural Network," 2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAIET), Kota Kinabalu, Malaysia, 2020, pp. 1-6, doi: 10.1109/IICAIET49801.2020.9257843.

[9] M. A. Hasnat, M. R. Chowdhury and M. Khan, "An Open Source Tesseract Based Optical Character Recognizer for Bangla Script," 2009 10th International Conference on Document Analysis and Recognition, Barcelona, Spain, 2009, pp. 671-675, doi: 10.1109/ICDAR.2009.62.

[10] M. T. Chowdhury, M. S. Islam, B. H. Bipul and M. K. Rhaman, "Implementation of an Optical Character Reader (OCR) for Bengali language," 2015 International Conference on Data and Software Engineering (ICoDSE), Yogyakarta, Indonesia, 2015, pp. 126-131, doi: 10.1109/ICODSE.2015.7436984.

[11] Saoji, Saurabh & Singh, Rajat & Eqbal, Ashiq & Vidyapeeth, Bharati. (2021). TEXT RECOGIRATION AND DETECTION FROM IMAGES USING PYTESSERACT. Journal of Interdisciplinary Cycle Research. XIII. 1674-1679.

[12] Chenxia Li, Weiwei Liu, Ruoyu Guo, Xiaoting Yin, Kaitao Jiang, Yongkun Du, Yuning Du, Lingfeng Zhu, Baohua Lai, Xiaoguang Hu, Dianhai Yu, Yanjun Ma, "PP-OCRv3: More Attempts for the Improvement of Ultra Lightweight OCR System," 2022, arXiv:2206.03001,

[13] Y. Ren, H. Yao, G. Liu and Z. Bai, "A Text Code Recognition and Positioning System for Engineering Drawings of Nuclear Power Equipment," 2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 2022, pp. 661-665, doi: 10.1109/ITOEC53115.2022.9734621.

[14] Breuel, Thomas. (2008). The OCRopus open source OCR system. 6815. 68150. 10.1117/12.783598.

[15] G Abarajithan, Chamira U. S. Edussooriya, "Kraken: An Efficient Engine with a Uniform Dataflow for Deep Neural Networks," 2021, arXiv:2112.02793

[16] d Benjamin Kiessling (PSL), Gennady Kurin, Matthew Thomas Miller, Kader Smail, "Advances and Limitations in Open Source Arabic-Script OCR: A Case Study," 2024, arXiv:2402.10943

[17] A. N. Chowdhury, S. Pal Summit, M. F. Ahmed Laskar, G. Mahfuz Chowdhury, I. A. Chowdhury and M. Hasan, "ALPR: ResNet50 powered Bangla License Plate Detectionand OCR by Root Mean Square Propagation Optimizer and Linear SVM Classifier," 2024 IEEE 9th International Conference for Convergence in Technology (I2CT), Pune, India, 2024, pp. 1-8, doi: 10.1109/I2CT61223.2024.10543675.

[18] Jurafsky, D., & Martin, J. H. (2021). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (3rd ed.). Pearson.

[19] A. N. Chowdhury, "Bangla-CrossHair", Kaggle Dataset. [Online]. Avalable: https://www.kaggle.com/datasets/abdullahnchy/ocr-bangla/data

[20] A. N. Chowdhury, "Bangla-CrossHair", Github repository. [Online]. Available: https://github.com/Abdullah-Nasir-Chowdhury/Bangla-CrossHair