Multi-label Classification of Common Bengali Handwritten Graphemes: Dataset and Challenge

Samiul Alam, 1, 2 Tahsin Reasat, 1 Asif Shahriyar Sushmit, 1 Sadi Mohammad Siddiquee, 1
F uad Rahman, 3 Mahady Hasan, 4 Ahmed Imtiaz Humayun 1

1 Bengali.AI, 2 Bangladesh University of Engineering and Technology, 3 Apurba Technologies Inc., 4 Independent University, Bangladesh

Abstract

Latin has historically led the state-of-the-art in handwritten optical character recognition (OCR) research. Adapting existing systems from Latin to alpha-syllabary languages is particularly challenging due to a sharp contrast between their orthographies. The segmentation of graphical constituents corresponding to characters becomes significantly hard due to a curved writing system and frequent use of diacritics in the alpha-syllabary family of languages. We propose a labeling scheme based on graphemes (linguistic segments of word formation) that makes segmentation inside alpha-syllabary words linear and present the first dataset of Bengali handwritten graphemes that are commonly used in an everyday context. The dataset is open-sourced as a part of the Bengali.AI Handwritten Grapheme Classification Challenge on Kaggle to benchmark vision algorithms for multi-label grapheme classification. From competition proceedings, we see that deep learning methods can generalize to a large span of uncommon graphemes even when they are absent during training.

1 Introduction

Speakers of languages from the alpha-syllabary or Abugida family comprise of up to 1.3 billion people across India, Bangladesh, and Thailand alone. There is significant academic and commercial interest in developing systems that can optically recognize handwritten text for such languages with numerous applications in e-commerce, security, digitization, and e-learning. In the alpha-syllabary writing system, each word is comprised of segments made of character units that are in phonemic sequence. These segments act as the smallest written unit in alpha-syllabary languages and are termed as Graphemes (Fedorova 2013); the term alpha-syllabary itself originates from the alphabet and syllabary qualities of graphemes (Bright 1999). Each grapheme comprises of a graphene root, which can be one character or several characters combined as a conjunct. Root characters may be accompanied by vowel or consonant di-

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Figure 1: Orthographic components in a Bangla (Bengali) word compared to English and Devnagari (Hindi). The word 'Proton' in both Bengali and Hindi is equivalent to its transliteration. Characters are color-coded according to phonemic correspondence. Alpha-syllabary grapheme segments and corresponding characters from the three languages are segregated with the markers. While characters in English are arranged horizontally according to phonemic sequence, the order is not maintained in the other two languages.

acies- demarcations which correspond to phonemic extensions. To better understand the orthography, we can compare the English word Proton to its Bengali transliteration প্রোটোন (Fig. 1). While in English the characters are horizontally arranged according to phonemic sequence, the first grapheme for both Bengali and Devanagari scripts have a sequence of glyphs that do not correspond to the linear arrangement of Unicode characters or phonemes. As most OCR systems make a linear pass through a written line, we believe this non-linear positioning is important to consider when de-
signing such systems for Bengali as well as other alphasyllabary languages.

We propose a novel labeling scheme for handwritten Bengali text and based on it, we have curated the Bengali Handwritten Grapheme Dataset which contains 411883 images of 1295 unique commonly used handwritten graphemes and ~ 900 uncommon graphemes (exact numbers are excluded for the integrity of the test set). The proposed labeling scheme fundamentally addresses the issues in the past labeling schemes in alpha-syllabary languages and provides an alternate grapheme based approach for Bengali; which can also be extended to other languages. The utilization scope of this dataset is not limited to the domain of OCR; since the dataset has labels for different components of each grapheme, it also creates an opportunity to evaluate multi-task or multi-label classification algorithms. The training set and closed test set is available at https://www.kaggle.com/c/bengaliai-cv19.

2 Related Work

Although several datasets (Alam et al. 2018; Biswas et al. 2017; Rabby et al. 2019; Sarkar et al. 2012) have been made for Bengali handwritten characters, their effectiveness has been limited. This could be attributed to the fact that they were formulated following contemporary datasets of English characters. All these datasets label individual characters or words. These datasets work well for English and can even be adapted for other document recognition tasks, setting the standard. Most character recognition datasets for other languages like the Devanagari Character Dataset (Acharya, Pant, and Gyawali 2015; Obaidullah et al. 2016; Dutta et al. 2018) and the Arabic Printed Text Image Database (Farrah Moghaddam et al. 2010; AlKhateeb 2015) were created following their design. However, they do not boast the same effectiveness and adaptability of their English counterparts. Languages with different writing systems therefore require language specific design of the recognition pipeline and need more understanding of how it affects performance. To the best of our knowledge, this is the first work that proposes grapheme level recognition for Bengali.

3 Challenges of Bengali Orthography

As mentioned before in section 1, each Bengali word is comprised of segmental units called graphemes. Bengali has 48 characters in its alphabet- 11 vowels and 38 consonants (including special characters ‘ং’, ‘ঃ’, ‘ঁ’). Out of the 11 vowels, 10 vowels have diacritic forms. There are also four consonant diacritics, ‘ঃ’ (from consonant ‘ষ’), ‘ঁ’ (from consonant ‘ঈ’), ‘ঃ’ (from consonant ‘ঈ’) and ‘ঁ’; We follow the convention of considering ‘ং’, ‘ঃ’ as standalone consonants since they are always present at the end of a grapheme and can be considered a separate root character.

3.1 Grapheme Roots and Diacritics

Graphemes in Bengali consist of a root character which may be a vowel or a consonant or a consonant conjunct along with vowel and consonant diacritics whose occurrence is optional. These three symbols together make a grapheme in Bengali. The consonant and vowel diacritics can occur horizontally, vertically adjacent to the root or even surrounding the root (Fig. 2). These roots and diacritics cannot be identified in written text by parsing horizontally and detecting each glyph separately. Instead, one must look at the whole grapheme and identify them as separate labels. In light of this, our dataset labels individual graphemes with root characters, consonant and vowel diacritics as separate labels.

3.2 Consonant Conjuncts or Ligatures

Consonant conjuncts in Bengali are analogous to ligatures in Latin where multiple consonants combine together to form glyphs which may or may not contain characteristics from the standalone consonant glyphs. In Bengali, up to three consonants can combine to form consonant conjuncts. Consonant conjuncts may have two (second order conjuncts, eg. হি = হ + ি) or three (third order conjuncts, eg. হি = হ + ি + ন) consonants in the cluster. Changes in the order of consonants in a conjunct may result in complete or partial changes in the glyph.

3.3 Allographs

Figure 2: Different vowel diacritics (green) and consonant diacritics (red) used in Bengali orthography. The placement of the diacritics are not dependent on the grapheme root.

Figure 3: Examples of allograph pair for the same consonant conjunct ‘ঃ’ (3a and 3b) and the same vowel diacritic ‘ঁ’ (3c and 3d) marked in green. 3b and 3d follows an orthodox writing style.

It is also possible for the same grapheme to have multiple styles of writing, called allographs. Although they are indistinguishable both phonemically and in their Unicode forms, allographs may in fact appear to be significantly different in their handwritten guise (Fig. 3). Allograph pairs are sometimes formed due to simplification or modernization of handwritten typographies,
i.e. instead of using the orthodox form for the consonant conjunct \( \text{ষ} = \text{ঝ} + \text{়} \) as in Fig. 3b, a simplified more explicit form is written in Fig. 3a. The same can be seen for diacritics in Fig. 3c and Fig. 3d. It can be argued that allographs portray the linguistic plasticity of handwritten Bengali.

3.4 Unique Grapheme Combinations

One challenge posed by grapheme recognition is the huge number of unique graphemes possible. Taking into account the 38 consonants \( (n_c) \) including three special characters, 11 vowels \( (n_v) \) and \( (n_v^2 + n_c^2) \) possible consonant conjuncts (considering \( 2^{nd} \) and \( 3^{rd} \) order) there can be \( ((n_c-3)^3+(n_c-3)^2+(n_c-3)) + 3 \) different grapheme roots possible in Bengali. Grapheme roots can have any of the 10+1 vowel diacritics \( (n_{dv}) \) and 7+1 consonant diacritics \( (n_{dc}) \). So the approximate number of possible graphemes will be \( n_v + 3 + ((n_c-3)^3 + (n_c-3)^2 + (n_c-3)) - n_{dc} \) or 3883894 unique graphemes. While this is a big number, not all of these combinations are viable or are used in practice.

4 The Dataset

Of all the possible graphemes combinations, only a small amount is prevalent in modern Bengali. Therefore, we narrowed down the total number of graphemes to reduce the complexity of the dataset.

4.1 Grapheme Selection

To find the popular graphemes, we use the text transcriptions for the Google Bengali ASR dataset (Kjartansson et al. 2018) as our reference corpus. The ASR dataset contains a large volume of transcribed Bengali speech data. The transcription data by itself is very large and well standardized. It consists of 127565 utterances comprising 609510 words and 2111256 graphemes. Out of these graphemes, 1295 commonly used Bengali graphemes in everyday vocabulary is selected. Each candidate grapheme had to occur more than twice in the entire corpus or used in at least two unique words to be selected in our pool. Graphemes from highly frequent transliterations and misspelled words were also considered. The uncommon graphemes were synthesized by uniformly sampling from all the possible combinations and verifying their legibility.

4.2 Labeling Scheme

Bengali graphemes can have multiple characters depending on the number of consonants, vowels or diacritics forming the grapheme. We split the characters of a Bengali grapheme into three labels:

1. Vowel Diacritics, i.e. ◌, ◌ো, ◌ৌ, ◌ী, ◌ু, ◌ূ, ◌ৃ, ে◌, ৈ◌, ে◌া, ে◌ৗ. If the grapheme consists of a vowel diacritic, it is generally the final character in the Unicode string. Graphemes cannot contain multiple vowel diacritics. The vowel diacritic label has 11 orthogonal classes including a null diacritic denoting absence.

2. Consonant Diacritics, i.e. ◌lesen, ◌ৃ, ◌ু, ◌ৃ, ◌, ◌ী, ◌ু, ◌ৃ, ◌ী. Graphemes can have a combination of consonant diacritic characters e.g. ◌ৃ = ◌ু + ◌ী. We consider each combination to be a unique diacritic in our scheme for ease of analysis. The consonant diacritic label has 8 orthogonal classes including combinations and a null diacritic.

3. Grapheme roots, which can be comprised of vowels, consonants or conjuncts. In Unicode these are placed as the first characters of a grapheme string. An alternative way of defining grapheme roots would be considering all the characters apart from diacritics as root characters in a grapheme. While possible orthogonal classes under this label can be a very big number (see Section 3.4), we limit the number of consonant conjuncts based on commonality in everyday context.

Grapheme recognition thus becomes a multi-label classification task where a vision algorithm would have to separately recognize grapheme roots, vowel diacritics and consonant diacritics.

4.3 Collection Statistics

Handwritten data is collected using printed template forms each containing upto 81 unique graphemes. Contributors from ages 0 – 50 with diverse educational backgrounds, provided 162 graphemes handwritten graphemes each. After curation, the final dataset contains a total of 411883 handwritten graphemes of size 137 by 236 pixels. Dataset collection tools and protocols are available at https://rb.gy/lawaoj.

4.4 Class Imbalance in Dataset

We divide the roots into three groups- vowels, consonants, and consonant conjuncts- and inspect class imbalance within each. The are linguistic rules which restrict the number of diacritics that may occur with each of these roots, e.g. vowel roots never have added diacritics. Although imbalance in vowel roots is not major, it must be noted because the relatively infrequent vowel roots ‘ঈ’, ‘উ’ and ‘ি’ share a close resemblance to the more frequent roots ‘ই’, ‘ঐ’ and ‘এ’ respectively.

The imbalance in consonant roots is however much more striking as we can see in Fig. 4. The imbalance here is twofold- in the number of total sample images of consonant roots and the imbalances in the distribution of the vowel and consonant diacritics that can occur with each consonant. The consonant conjuncts demonstrate imbalance similar to the consonant roots but with an added degree of complexity. We can visualize this imbalance much better via the chord diagram in Fig. 5. The consonant conjuncts are made up of multiple consonant characters and since the glyph of a consonant conjunct often shares some resemblance with its constituent consonants, highly frequent consonants may increase estimation bias for less frequent conjuncts containing them. This phenomenon is indeed visible and further discussed in Section 5.3.
Figure 4: Number sample images per consonant root. Each bar represents the number of samples which contain a particular consonant and the divisions in each bar represent the variations of diacritics in the sample images of that consonant.

5 The Challenge

The dataset is split by 50-25-25 between the train, public test and private test sets for the challenge, while making sure there is no overlap in contributors between sets. Throughout the length of the competition, the participants try to improve their standings based on the public test set results. The private test set result on the other hand, is kept hidden for each submission and is only published after the competition is over. Of the unseen graphemes, 88.4% were placed in the private test set to prevent over-fitting models based on public standings. This motivated the participants to build methods that have the capacity to classify grapheme components independently instead of fine-tuning models based on public test set standings.

5.1 Competition Metric

The metric for the challenge is a hierarchical macro-averaged recall. First, a standard macro-averaged recall is calculated for each component. Let the macro-averaged recall for grapheme root, vowel diacritic, and consonant diacritic be denoted by $R_r$, $R_v$, and $R_c$ respectively. The final score $R$ is the weighted average $R = (2R_r + R_v + R_c)/4$.

5.2 Top scoring methods

The Kaggle competition resulted in 31,002 submissions from 2,059 teams consisting of 2,623 competitors. The participants have explored a diverse set of algorithms throughout the competition; the most popular being state of the art image augmentation methods such as cutout (DeVries and Taylor 2017), mixup (Zhang et al. 2017), cutmix (Yun et al. 2019), mixmatch (Berthelot et al. 2019) and fmix (Harris et al. 2020). Ensemble methods incorporating snapshots of the same or different network architectures were also common.

The winner took a grapheme classification approach rather than component recognition. The input images were classified into 14784 (168 x 11 x 8) classes, that is, all the possible graphemes that can be created using the available grapheme roots and diacritics. An EfficientNet (Tan and Le 2019) model is initially used to classify the graphemes. However, if the network is not confident about its prediction then the sample is considered as an unseen grapheme and it is passed on to the unseen grapheme classification pipeline. The pipeline consists of a CycleGan (Zhu et al. 2017) that is trained to convert handwritten graphemes into typeface rendered graphemes. An EfficientNet classifier trained on averaged recall for grapheme root, vowel diacritic, and consonant diacritic be denoted by $R_r$, $R_v$, and $R_c$ respectively. The final score $R$ is the weighted average $R = (2R_r + R_v + R_c)/4$.

5.2 Top scoring methods

The Kaggle competition resulted in 31,002 submissions from 2,059 teams consisting of 2,623 competitors. The participants have explored a diverse set of algorithms throughout the competition; the most popular being state of the art image augmentation methods such as cutout (DeVries and Taylor 2017), mixup (Zhang et al. 2017), cutmix (Yun et al. 2019), mixmatch (Berthelot et al. 2019) and fmix (Harris et al. 2020). Ensemble methods incorporating snapshots of the same or different network architectures were also common.

The winner took a grapheme classification approach rather than component recognition. The input images were classified into 14784 (168 x 11 x 8) classes, that is, all the possible graphemes that can be created using the available grapheme roots and diacritics. An EfficientNet (Tan and Le 2019) model is initially used to classify the graphemes. However, if the network is not confident about its prediction then the sample is considered as an unseen grapheme and it is passed on to the unseen grapheme classification pipeline. The pipeline consists of a CycleGan (Zhu et al. 2017) that is trained to convert handwritten graphemes into typeface rendered graphemes. An EfficientNet classifier trained on averaged recall for grapheme root, vowel diacritic, and consonant diacritic be denoted by $R_r$, $R_v$, and $R_c$ respectively. The final score $R$ is the weighted average $R = (2R_r + R_v + R_c)/4$.

5.2 Top scoring methods

The Kaggle competition resulted in 31,002 submissions from 2,059 teams consisting of 2,623 competitors. The participants have explored a diverse set of algorithms throughout the competition; the most popular being state of the art image augmentation methods such as cutout (DeVries and Taylor 2017), mixup (Zhang et al. 2017), cutmix (Yun et al. 2019), mixmatch (Berthelot et al. 2019) and fmix (Harris et al. 2020). Ensemble methods incorporating snapshots of the same or different network architectures were also common.

The winner took a grapheme classification approach rather than component recognition. The input images were classified into 14784 (168 x 11 x 8) classes, that is, all the possible graphemes that can be created using the available grapheme roots and diacritics. An EfficientNet (Tan and Le 2019) model is initially used to classify the graphemes. However, if the network is not confident about its prediction then the sample is considered as an unseen grapheme and it is passed on to the unseen grapheme classification pipeline. The pipeline consists of a CycleGan (Zhu et al. 2017) that is trained to convert handwritten graphemes into typeface rendered graphemes. An EfficientNet classifier trained on
a 14784 class synthetic grapheme dataset is used as an unseen grapheme classifier.

The second place team also built their solution upon a grapheme classifier, but the number of classes were limited to the 1295 unique graphemes in training. A post processing heuristic is employed to generalize for any unseen graphemes. For each class in a component (eg. consonant diacritic ‘.wr’), probabilities of all the unique graphemes in training containing the component class are averaged. This is repeated for every class for a component and the class with the highest average probability is selected. Architectures used are different variants of SE-ResNeXt (Hu, Shen, and Sun 2018).

The third placed team dealt with the unseen graphemes by using metric learning. They used Arcface (Deng et al. 2019) to determine if an input test sample is present or absent in the train dataset. If the grapheme is present, a single EfficientNet model is used that detects the grapheme components in a multi-task setting. Otherwise, a different EfficientNet models are used to recognize each grapheme component.

5.3 Submission Insights

For exploratory analysis on the competition submissions, we take the top 20 teams from both the private and public test set leaderboards and categorize their submissions according to quantile intervals of their private set scores as Tier 1 (> .976), Tier 2 (> .933, < .976), Tier 3 (> .925, < .933) and Tier 4 (> .88, < .925). It is seen that the Tier 4 submissions have low discrepancy between public and private test set metrics; suggesting these to be high bias - low variance estimators. The Tier 3 submissions were the total opposite and had high discrepancy on average, indicating fine-tuning of the model on the public test set. This discrepancy increases as we go to Tier 2 submissions but then decreases for Tier 1 submissions. In fact if we observe the progression of the Tier 2 teams, many of them were once near the top of the private leaderboard but later fine-tuned for the public test set. Contrary to intuition, error rate doesn’t significantly vary depending on the frequency of the unique grapheme in the training set, within each of these quantile groups. However, the error rate is higher by 1.31% on average for unseen graphemes, with Tier 1 error rate at 4.4(±0.07)%.

Considering the size of the challenge test set, possible reasons behind test set error could be label noise, class imbalance or general challenges surrounding the task. We start by inspecting the samples that were misclassified by all the Tier 1 submissions and find that only 34.8% had label noise. Significant error can be seen due to the misclassification of consonant conjunct classes which are binary combinations of ‘’ or ‘’\text{wr}’, with high false positives for the individual constituents. This can be attributed to the class imbalance in the dataset since combinations are less frequent that their primitives; separately recognizing the primitives in a multi-label manner could be a possible way to reduce such error. The vowel diacritic component has the highest macro-averaged recall, proving to be the easiest among the three tasks.

The false negatives of diferent grapheme roots give us significant insights on the challenges present in Bengali orthography. A pair of grapheme roots can have high similarity due to common characters or even similarity between glyphs of constituents. Probing the Tier 1 false negatives, we find that 56.5% of the error is between roots that share at least one character. Misclassification between consonant conjuncts with the same first and second characters accounts for 28.8% and 21.5% of the error. Confusion between roots by Tier 1 submissions highly correlate with similarity between glyphs and is visualized in Fig. 6. Edges correspond to the sum of false negative rates between nodes and is pruned if the sum is below .5%. Separate networks are formed by groups that are similar to each other but dissimilar with other. Class imbalance also plays an interesting role in such cases; misclassification of roots with high similarity between their handwritten glyphs can also be biased towards one class due to higher frequency, eg. roots with ‘’ or ‘’\text{w}’ being more frequent.

6 Discussions and Future work

In the competition, we have formulated the grapheme recognition challenge as a multi-class classification task but it is only one way of defining the grapheme recognition problem. It can also be posed as firstly, a sequence mapping task- where each character in the Unicode string is mapped to its corresponding glyph- and secondly a multi-label sequence mixture problem where
the grapheme root is a sequence of characters while di- 
critics are separate labels with orthogonal classes. Since 
there are many graphemes that are possible but never 
used in practice, the dataset can also be used to gen-
erate new graphemes (as demonstrated in Section 5.2) 
that are not in the dataset or less used in practice. One 
aspect not explored at all here is the possibility of gen-
erating consonant conjuncts not present in the dataset. 
Generating realistic consonant conjuncts could lead to 
better performances, especially on allographs.

Alongside benchmarking multi-task algorithms, this 
dataset can be used for evaluating the performance of 
generative models. The number of possible graphemes 
are extremely large but fixed to a finite number and 
governed by specific linguistic rules. So, the latent space 
of a well trained model should be able to capture a range 
of meaningful grapheme combinations. This is unlike 
what we expect from some more high dimensional data 
like CelebA (Liu et al. 2015) as there are constraints on 
the shape of human faces. Additionally, the grapheme 
dataset is relatively free from background clutter often 
found in CelebA dataset.

7 Conclusion

In this paper, we outlined the challenges of recognizing 
Bengali handwritten text and explained why a charac-
ter based labeling scheme- that has been widely suc-
cessful for English characters- does not transfer well 
to Bengali. To rectify this, we propose a novel labeling 
scheme based on graphemes and present a dataset based 
on this scheme. Crowd sourced benchmarking on Kag-
gle shows that algorithms trained on this dataset can 
generalize on unseen graphemes. This proves that it is 
possible to summarize the entire cohort of graphemes 
through some representative samples. This grapheme 
labeling scheme could be used as a stepping stone to 
solve OCR related tasks in not only Bengali but also 
other related languages in the alpha-syllabary family.

Acknowledgements

We would like to thank the Kaggle for their generous 
support through the Kaggle platform. We would like 
to thank the late Dr. Ali Shihab Sabbir for providing 
invaluable help and guidance to Bengali.AI since 
its inception. We are also grateful to our volunteers 
who participated and helped organize collection of data 
for our dataset. Full list of contributions available at 
https://rb.gov/qd3ur2.

References

Acharya, S.; Pant, A. K.; and Gyawali, P. K. 2015. Deep 
learning based large scale handwritten Devanagari character 
recognition. In Proc. 9th Int. Conf. SKIMA), 1–6.

Alam, S.; Reasat, T.; Doha, R. M.; and Humayun, A. I. 
2018. NumtaDB - Assembled Bengali Handwritten Digits. 
arXiv preprint arXiv:1806.02452.

AlKhateeb, J. H. 2015. A Database for Arabic Handwritten 
Character Recognition. Procedia Computer Science 65: 556 
– 561.

Berthelot, D.; Carlini, N.; Goodfellow, I.; Papernot, N.; 
Oliver, A.; and Raffel, C. 2019. MixMatch: A Holistic Ap-
proach to Semi-Supervised Learning. In Proc. NeurIPS, 
2019.

Biswa, M.; Islam, R.; Shom, G. K.; Shupon, M.; Mo-
hammed, N.; Momen, S.; and Abedin, M. A. 2017. 
BanglaLekha-Isolated: A Comprehensive Bangla Handwrit-
ten Character Dataset. arXiv preprint arXiv:1703.10661.

Bright, W. 1999. A Matter of Typology: Alphasyllabaries 
and Abugidas. Written Language & Literacy 2(1): 45–65.

Deng, J.; Guo, J.; Xue, N.; and Zafeiriou, S. 2019. ArcFace: 
Additive Angular Margin Loss for Deep Face Recognition. 
In Proc. 2019 IEEE/CVF Conf. CVPR.

DeVries, T.; and Taylor, G. W. 2017. Improved regulariza-
tion of convolutional neural networks with cutout. arXiv 
preprint arXiv:1708.04552.

Dutta, K.; Krishnan, P.; Mathew, M.; and Jawahar, C. V. 
2018. Offline Handwriting Recognition on Devanagari 
Using a New Benchmark Dataset. In Proc. 15th IAPR Int. 
Workshop DAS, 25–30.

Farrahi Moghaddam, R.; Cheriet, M.; Adankon, M. M.; Filo-
enko, K.; and Wisnovsky, R. 2010. IBN SINA: A Database 
for Research on Processing and Understanding of Arabic 
Manuscripts Images. In Proc. 9th IAPR Int. Workshop DAS, 
11–18.

Fedorova, L. 2013. The Development of Graphic Representa-
tion in Abugida Writing: The Akshara’s Grammar. Lingua 
Posnaniensis 55(2): 49–66.

Harris, E.; Marcu, A.; Painter, M.; Niranjan, M.; Prügel-
Bennett, A.; and Hare, J. 2020. FMix: Enhancing 
mixed Sample Data Augmentation. arXiv e-prints 
arXiv:2002.12047.

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-Excitation 
Networks. In Proc. IEEE/CVF Conf. CVPR, 2018: 7132-
7141. ISSN 2575-7075.

Kjartansson, O.; Sarin, S.; Pipatsrisawat, K.; Jansche, M.; 
and Ha, L. 2018. Crowd-Sourced Speech Corpora for Ja-
vanese, Sundanese, Sinhala, Nepali, and Bangledeshi Ben-
gali. In Proc. 6th Intl. Workshop SLTU, 52–55.

Liu, Z.; Luo, P.; Wang, X.; and Tang, X. 2015. Deep Learn-
ing Face Attributes in the Wild. In Proc. ICCV.

Obaidullah, S. M.; Halder, C.; Das, N.; and Roy, K. 2016. 
A New Dataset of Word-Level Offline Handwritten Numeral 
Images from Four Official Indic Scripts and Its Benchmark-
ing Using Image Transform Fusion. Int. J. Intell. Eng. In-
form. 4(1): 1–20.

Rabby, A. S. A.; Haque, S.; Islam, M. S.; Abujar, S.; and 
Hossain, S. A. 2019. Ekush: A Multipurpose and Multi-
type Comprehensive Database for Online Off-Line Bangla 
Handwritten Characters. In RTIP2R, 149–158.

Sarkar, R.; Das, N.; Basu, S.; Kundu, M.; Nasipuri, M.; 
and Basu, D. K. 2012. CMATERDB1: a database of un-
constrained handwritten Bangla and Bangla–English mixed 
script document image. IJDAR 15(1): 71–83.

Tan, M.; and Le, Q. 2019. EfficientNet: Rethinking Model 
Scaling for Convolutional Neural Networks. In PMLR, vol-
ume 97, 6105–6114.
Appendix

A Grapheme Collection Form

Handwritten Graphemes were collected from 16 different template forms designed for efficient extraction by scanning. Forms also had additional information that include chirality, age group, gender, medium of instruction and location of primary school to allow for further study into handwriting. Names and IDs were collected to keep into account multiple forms submitted by the same person; but were later de-identified. A sample form is showed in Fig. 7.

Figure 7: Sample of OCR form for extracting handwritten graphemes

B Dataset collection & standardization

The data was obtained from volunteers in schools, colleges and universities across Bangladesh. A standardized form A with alignment markers were printed and distributed. A total of 2896 volunteers participated in the project. Each subject could be uniquely identified through their institutional identification number submitted through the forms. A random subject identifier was generated for each individual and the identifying information was removed. A summary of the dataset compilation protocol is detailed below:

Step 1: Collection The candidate data for the dataset was collected using forms where contributors filled up boxes with handwritten graphemes as prompted in boxes. There were 16 different form templates spanning the 1295 different graphemes to be collected. The templates were automatically generated using Adobe Photoshop scripts. Each template had a unique set of graphemes from the others. Since the 16 different templates were not dispersed uniformly every time data was collected, minor sampling bias was introduced during collection.

Step 2: Pruning and Scanning The forms were scanned and analysed carefully to remove invalid submissions and reduce sampling bias. In this step additional errors were introduced due to misalignment during scanning. Unfilled or improperly filled samples were still retained. All the forms were scanned using the same device at 300 dpi.

Step 3: Extraction An OCR algorithm was used to automatically detect the template ID. The template identifier asserted which ground truth graphemes where present in which boxes of the form. In this step, OCR extraction errors introduced label noise. The scanned forms were registered with digital templates to extract handwritten data, which sometimes introduced alignment errors or errors while extracting metadata. Unfilled boxes were removed automatically in this step.

Step 4: Preliminary Label Checking The extracted graphemes were compiled into batches and sent to 22 native Bengali volunteers who analysed each image and matched them to their corresponding ground truth annotation. In this step OCR errors and label noise was minimised. However additional error was introduced in the form of conformity bias, linguistic bias (i.e. allograph not recognized), grapheme bias (i.e. particular grapheme has significantly lesser number of samples) and annotator subjectivity. Samples selected as erroneous by each annotator was stored for further inspection instead of being discarded.

Step 5: Label Rechecking Each batch from the previous step, was sent to one of two curators who validated erroneous samples submitted by annotators and re-checked unique graphemes which had a higher frequency of mislabeled samples.

Step 6: Subjectivity Normalization A fixed guideline is decided upon by all curators that specifies how much and the nature of deformity a sample can contain. Based on this, subjectivity errors were minimized for unique graphemes with high frequency mislabeled samples.
Figure 8: Overview of dataset creation process. Green arrows refer to the bias/errors removed in each step and red refers to the ones inevitably introduced.

Table 1: Frequently discarded graphemes during label checking phases of dataset curation

| Root Category       | Top 5 Graphemes                                                                 |
|---------------------|---------------------------------------------------------------------------------|
| Consonants          | গ‌ুঁ গ‌ুর্ গ‌ু শ‌ু– ঘ ূ                                                        |
| 2nd order Conjuncts | ষ্ট–া ঙ্ক ৃ ষ্পা ঙ্ক ষ্কৰ্ী                                                   |
| 3rd order Conjuncts | িঙক্ত ক্ষ্ণ েক্ষ্ণৗ ঙ্ক্ষা িন্দব্                                                   |

### C Contribution Error and Subjectivity

The original data went through a rigorous curation process; approximately 12817 samples were discarded due to either label noise or corrupted submissions. Samples that would be illegible to human annotators without prior knowledge of the ground truth were discarded from the dataset. Although this was done to make sure the data is clean, it should be mentioned that a concrete definition of which samples should be considered legible does not exist. In fact some would consider a written sample perfectly legible while others would consider the same as absolutely unclear. If we look at the error data, among the top 100 unique graphemes that had the most erroneous contributions, 83 had consonant conjuncts as the grapheme root. Six out of the nine 3rd order consonant conjuncts in the dataset, were among the most erroneous graphemes. This matches the intuition that more complex grapheme glyphs are harder to discern as writers are more likely to make mistakes in typography when writing glyphs with more intricate patterns. The most common errors in writing graphemes categorized by the number of simple consonants in the root are given in Table 1.

### D Label-Class Overview in Dataset

All classes for each of the three labels have been listed in their utf-8 form in Table 2.

Table 2: Table of All Label Classes

| Label | Class |
|-------|-------|
| Grapheme roots (168) | VOWEL ROOTS |
| অ, আ, ই, ঈ, উ, ঊ, ঋ, এ, ঐ, ও, ঔ |
| CONSONANT ROOTS |
| ক, খ, গ, ঘ, ঙ, চ, ছ, জ, ঝ, ঞ, ট, ঠ, ড, ঢ, ণ, ত, থ, দ, ধ, ন, প, ফ, ব, ভ, ম, য, র, ল, শ, ষ, স, হ, ড়, ঢ়, ◌ং, ◌ঃ, য়, ৎ |
| CONJUNCT ROOTS |
| গ্ম, ক্ষ্ণ, শ্ল, ণ্ড, স্প, ন্ট, ষ্ণ, প্স, ণ্ঠ, সব্, ণ্ট, ঙক্ত, ন্ড, সস, ক্ক, চ্ছ, ধব্, হ্ন, ঘ্ন, ষ্ম, ঙ্খ, স্ফ, ন্ত, ন্ঠ, প্ট, ক্ষ্ম, হব্, ম্ম, চ্ছব্, স্ত, হ্ল, ক্ট, ক্ত, মব্, শ্চ, গব্, ত্ন, ক্স, জ্ঞ, ন্দব্, ঙ্গ, হ্ম, ল্গ, ঙ্ক্ষ, ষ্প, ষ্ক, জ্জব্, ŀ, ড্ড, ঙ্ঘ, ন্ম, ন্তব্, ন্স, ফ্ল, জ্জ, ল্ক, ষ্ট, নব্, ফফ, ত্ত, ট্ট, ঞ্জ, দ্ভ, ল্ড, ণ্ণ, ļ, ষ্ঠ, ম্ন, লব্, স্ক, ন্থ, গ্ন, দ্ম, জব্, শব্, দ্ঘ, স্ম, ল্ট, স্ট, ত্তব্, দ্ধ, প্ত, ন্ধ, বব্, নজ, ষ্ফ, ক্ল, ম্ভ, তব্, ক্ষ, ন্দ, প্প, ঞ্চ, প্ন, দব্, শ্ম, শ্ন, ঙ্ক, দ্দ, ভ্ল, ত্ম, ম্ল, গ্ল, ł, স্থ, ব্ল, ত্থ, ফ্ট, ল্ল, ঙ্ছ, ন্ন, গ্ধ, স্ল, ম্প, ল্প, চ্চ, স্ন, প্ল, ল্ম |

| Vowel Diacritics (11) | Null, ু, ি, ী, ু, ি, ূ, ৃ, ৄ, ৆, ে, ৈ, ৉, ৌ |
| Consonant Diacritics (8) | Null, ৃ (র-ফলা), ু (র-ফলা), ী (রেফ-ফলা), ু (র-ফলা) |
E Consonant Conjuncts vs Diacritics

One question that arises while splitting the constituents of a grapheme into the three bins of our labeling scheme (see Section 4.2) is the ambiguity between consonant conjuncts and consonant diacritics. While conjuncts are formed by adding multiple consonants together, consonant diacritics also add consonants with other consonants but as demarcations that are completely different from the original glyph of the consonant. For example, the consonant diacritic '◌–' is completely different from its original form 'য'. Whenever it is added to a consonant root, the root retains its original glyph. This is not always the case for consonant conjuncts, where the consonants being added to might change its form significantly. In Bengali grammar, consonant diacritics are called Fola and defined separately from consonant conjuncts as Jukto Borno. The consonants that have diacritic forms do not construct conjuncts, eg. 'র' and 'ঈ' are not present as a second order conjunct constituent in Fig. 5.

F Training Set Metadata

The metadata collected through forms are compiled together for further studies on dependency of handwriting with each of the meta domains. Only the training set metadata is made public; the test set metadata will be made available upon request for benchmarking handwriting dependency with the metadata. The training set contains handwriting from 1448 individuals, each individual contributing 138.8 graphemes on average; 1037 of the contributors identified as male, 383 as female, 4 as non-binary and 24 declined to identify. The medium of instruction during primary education for 1196 contributors was Bengali, for 214 English and for 12 Madrasha (Bengali and Arabic); 33 are left-handed while 1192 are right handed. Of all the contributors, 93 fall in the age group between 0 – 12, 245 in 13 – 17, 1057 in 18 – 24, 22 in 25 – 35 and 2 in ages between 36 – 50.