# A Deep Genetic Method for Keyboard Layout Optimization For Bangla Language

1805094 - SHEIKH HASANUL BANNA and 1805107 - PARTHO KUNDA, Bangladesh University of Engineering and Technology, Bangladesh

## 1 INTRODUCTION

Keyboard is still the most popular method for text communication. Given its significant usage, having a keyboard that increases typing speed, while being ergonomic is an important issue.

QWERTY keyboard layout, named using the first 6 letters, is the first layout adopted worldwide and it is still the most used keyboard layout. But it was not developed keeping typing speed and user comfort in mind. Later on, Dvorak and Colemak layouts gained some popularity, as they reduced typing effort by a large amount.

There have been some attempts to find better keyboards using some automated techniques [7], [1]. Our report is inspired by the work done in [4]. This paper combined deep learning and genetic algorithm for generating efficient keyboard layouts. Effort value from carpalx was used to benchmark keyboard layouts.

There have been few works on Bangla keyboard layouts. Kamruzzaman et al. [3] presented an optimal Bangla Keyboard Layout utilizing data mining techniques to distribute the load equally on both hands, thereby maximizing typing speed and minimizing effort. Dasgupta et al. [2] developed a novel Bangla phonetic keyboard layout based on the Unicode 5.0 standard, using statistics for evaluation and heuristics for improvement. But, no work has been done using the genetic algorithm mentioned in [4] for Bangla keyboards. This report presents novel work in following topics:

- We show the effort value for the most popular touch typing layout for Bangla language, Bijoy.
- Generate a layout with significant improvement compared to Bijoy.
- Try CNN as prediction layer and record overall performance.

## 2 METHODOLOGY

In this section, first we will discuss the methodology in [4]. Then we include information about our dataset, preprocessing step, one of our modification with reasoning behind it and training details for our experiment.

### 2.1 Algorithm Flowchart

Flowchart of our algorithm is given in Fig. 1. Initially, we generate random keyboards to train our model. Then, we create population $P$ number of keyboards. Our model estimates their effort, and $S$ layouts from $P$ are selected with the lowest estimated effort value by the model. We run carpalx on these $S$ layouts, reducing the number of times we have to run carpalx from $P$ to $S$. Effort value from carpalx is considered to be true effort value and it is used to further fine-tune our model. These $S$ layouts are directly copied into the next generation. Also $R$ number of completely random layouts are added in each generation. Among the best $S$ from prediction layer, we select $T$ layouts from lowest true effort and they are used to generate $P - S - R$ new layouts using the cycle crossover routine, as discussed in 2.2. Every one of them undergoes 0 to $\mu$ number of random mutation. Each mutation consists of selecting two random keys and swapping them.

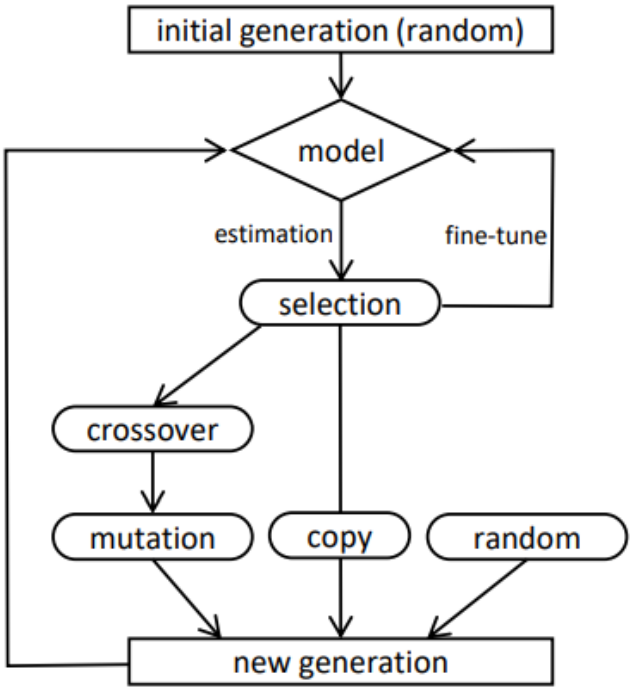


Fig. 1. Flow chart

### 2.2 Cycle Crossover Routine

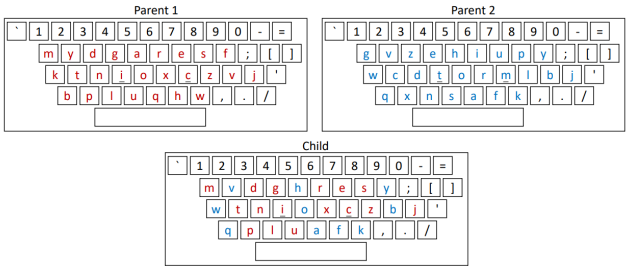


Fig. 2. Crossover

Authors' address: 1805094 - Sheikh Hasanul Banna; 1805107 - Partho Kunda, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh.

Fig. 2 shows an example of the cycle crossover routine. Here, the routine first picked parent 1, picked from it the letter n, and copied the letter n to the child. The routine checked the letter n's location in parent 2; in that location, parent 1 has the letter l. The routine copied the letter l to the child, and checked its location in parent 2. In that location, parent 1 has the letter z. Continuing this way, the routine copied the letter z and then the letter d to the child, and then came back to the letter n, which was the initial letter copied from parent 1. This finished one cycle of the crossover routine. The routine then picked parent 2 and picked from it the letter b. Continuing as described before, the routine copied from parent 2 to the child the letters b, q, a, h, f, y, and v, and then came back to b and closed another cycle. This process continued until the child layout was complete.

### 2.3 Dataset

The dataset for the original paper was from [6]. This is a TED-Talk document, which is a collection of TED-Talk sessions.
The dataset used for bengali keyboard layout optimization is [5]

- The dataset is a csv with the following format : id,text,title,url.
- total number of lines were 181973
- total number of words were 18394452

We only included the text field in our model training.

### 2.4 Dataset Preprocessing

The text field contained non-bengali characters, which were removed. Bijoy layout does not have one to one mapping for all unicode characters ( e.g., আ). So, we replaced those bengali unicode characters with corresponding Bijoy Layout keystrokes ( e.g., আ = অ + া). Since, carpalx counts the triads for effort value generation, we saved all possible three consecutive keypress counts in a file. Instead of loading the input corpus everytime and creating triad counts, Carpalx has been modified to load these triad counts directly.

### 2.5 Prediction Layer Modification

Keys in the keyboard are arranged in a 2D space. Basic dense layer reads the data by flattening it. Since, the spatial arrangement of keys are meaningful, we tried using the CNN for the prediction layer.

Table 1. CNN Model Summary

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d (Conv2D) | (None, 24, 24, 32) | 320 |
| max_pooling2d (MaxPooling2D) | (None, 12, 12, 32) | 0 |
| conv2d_1 (Conv2D) | (None, 10, 10, 64) | 18,496 |
| max_pooling2d_1 (MaxPooling2D) | (None, 5, 5, 64) | 0 |
| conv2d_2 (Conv2D) | (None, 3, 3, 128) | 73,856 |
| flatten (Flatten) | (None, 1152) | 0 |
| dense (Dense) | (None, 64) | 73,792 |
| dense_1 (Dense) | (None, 1) | 65 |

## 3 RESULTS

For our experiments, the parameters were set according to given table.

| Parameters | Value |
|---|---|
| P | 1000 |
| S | 50 |
| T | 20 |
| R | 200 |
| $\mu$ | 5 |
| epoch | 200 |
| generation | 15 |

### 3.1 Bangla Keyboard

We see improvement in effort value in our proposed keyboard compared to the Bijoy layout.

| | Effort Value |
|---|---|
| Bijoy | 2.4616079548 |
| Deep Genetic | 1.7524804565 |

Table 2. Comparison between Bijoy and Deep genetic

The suggested layout is shown in Fig. 3

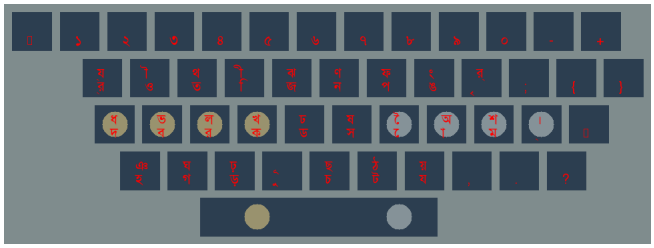

Fig. 3. Bangla Layout generated by Deep genetic algorithm

### Explanation

The middle row is the best for typing, with the first row being second when it comes to effort. The worst row for typing is the bottom-most row. The initial finger positions are shown in circles. Characters that are required the most are placed directly at initial finger positions(e.g., ক/খ, অ/ া ). The middle columns are also occupied by widely used characters(e.g., স/ষ, 'ু'/'ূ' ), as characters placed here require little effort to type. After that, the top rows are occupied by less frequent characters(e.g., 'ে'/'ৈ', 'ৃ'/'র্' ), and the bottom row is occupied by the least frequent characters(e.g., হ/ঞ, য/য় ).

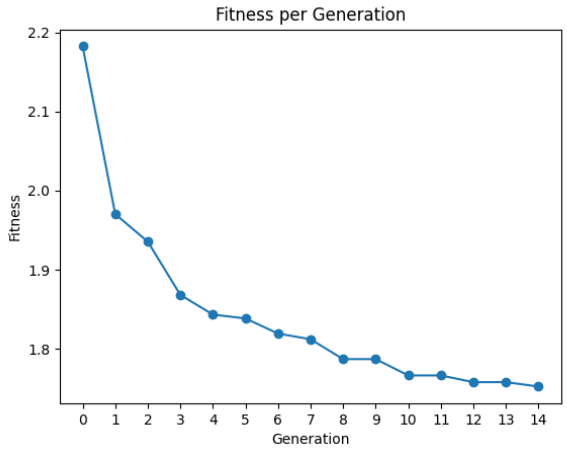


Fig. 4. Fitness plot for Bangla Layout

Effort value quickly reduces in the first 10 generations, as shown in Fig. 4. After that, the improvement per generation is minimal.

## 3.2 Network Modification

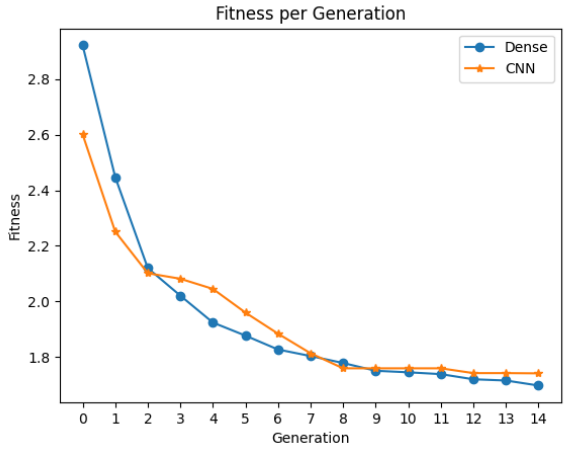


Fig. 5. Fitness for Dense vs CNN layer for English Keyboards

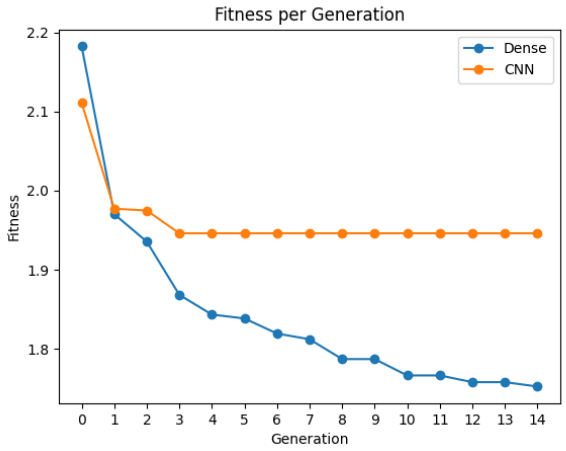


Fig. 6. Fitness for Dense vs CNN layer for Bangla Keyboards

Fig. 5 shows CNN and default Dense layer comparison for English layouts. CNN gave better results for the first few generations, but this improvement is not maintained in the later generations. Fig. 6 shows the same comparison, but for Bangla keyboards. In this case, CNN failed to improve after the 3rd generation. Whereas, with Dense layer for prediction, the result was as expected.

CNN model was much more complex, overfitting and predicting better scores for known keyboard layouts. As a result, new layouts were never included in $S$ and no significant improvement was seen.

## 4 CONCLUSION

In this report, we modified the work in [4] to quantify how good Bijoy Layout is and propose a new touch-typing layout. We have shown a significant reduction in effort value. We also tried incorporating CNN in our prediction layer, but found almost similar result.

In our experiment, we have restricted both upper and lower case bangla letter to the same key. For example, ক and খ, both are colocated on the same key and they cannot be moved to different keys. Future work can look into if enabling any letter to be placed into any key further improves the effort value.

We only worked with touch typed keyboard. Recently, most of the users write Bangla using phonetic keyboards. Converting our dataset to english phonetic key mappings and finding optimal keyboard layout considering both Bangla and English corpus can be included in future work.

## 5 ACKNOWLEDGEMENTS

This project has been supervised by Sheikh Azizul Hakim Sir.

## REFERENCES

[1] M. Al-Ayyoub A. Fadel, I. Tuffaha and Y. Jararwch. Qwerty keyboard? .?bzq is better! *2020 International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*, 2020.

[2] Tirthankar Dasgupta, Anupam Basu, Animesh Das, and Promathesh Mandal. Design and evaluation of Bangla keyboard layouts. *IEEE*, 4 2010.

[3] S. M. Kamruzzaman, Md. Hijbul Alam, Abdul Kadar Muhammad Masum, and Md. Mahadi Hassan. Optimal Bangla Keyboard Layout using Data Mining Technique. *arXiv (Cornell University)*, 9 2010.

[4] Keren Nivasch and Amos Azaria. A deep genetic method for keyboard layout optimization. *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, 11 2021.
[5] SADMAN ARAF. Bangla wikipedia dataset. https://www.kaggle.com/datasets/sadmanaraf/bangla-wikipedia-dataset.
[6] Stephen Palumbi. Following the murcury trail. http://www.ted.com/talks/stephen_palumbi_following_the_mercury_trail.html, 2013. Accessed: 2024-03-02.
[7] P.-Y. Yin and E.-P. Su. Cyber swarm optimization for general keyboard arrangement problem. *International Journal of Industrial Ergonomics*, 2011.