Keyboard Layout Optimization and Adaptation

Keren Nivasch
Computer Science Department, Ariel University
Ariel, Israel
kerenni@ariel.ac.il

Amos Azaria
Computer Science Department, Ariel University
Ariel, Israel
amos.azaria@ariel.ac.il

***Abstract*—Since the keyboard is the most common method for text input on computers today, the design of the keyboard layout is very significant. Despite the fact that the QWERTY keyboard layout was designed more than 100 years ago, it is still the predominant layout in use today. There have been several attempts to design better layouts, both manually and automatically. In this paper we improve on previous works on automatic keyboard layout optimization, by using a deep neural network to assist in a genetic search algorithm, which enables the use of a sophisticated keyboard evaluation function that would otherwise take a prohibitive amount of time. We also show that a better choice of crossover routine greatly improves the genetic search. Finally, in order to test how users with different levels of experience adapt to new keyboard layouts, we conduct some layout adaptation experiments with 300 participants to examine how users adapt to new keyboard layouts.**

***Index Terms*—Keyboard Layout, Genetic Algorithm, Neural Network**

I. Introduction

The modern QWERTY keyboard layout was introduced in the 1870's by Christopher Latham Sholes [1]. It has been suggested that the rationale behind the QWERTY design was to minimize type-bar jams by placing common letters far away from each other [2].

In the early 1930's, August Dvorak introduced the keyboard layout known today as *Dvorak* [3], which he hoped would be more ergonomic and lead to faster typing. Even though the QWERTY layout is still the most common layout in use, most major operating systems offer the option of switching to the Dvorak layout. Nevertheless, Dvorak has not gained much popularity, probably because QWERTY is already so entrenched.

Despite recent advances in automatic speech recognition, keyboard input still remains the most common method of text communication used today. While people do not pay much attention to the keyboard layout, it has a tremendous impact not only on the typing speed, but also on wrist pain and repetitive strain injury (RSI) [4]. Unfortunately, the QWERTY layout so popular today was designed for typewriters rather than keyboards; therefore, it is very likely that it is sub-optimal for modern use.

There have been several other attempts at creating better keyboard layouts. A popular alternative to QWERTY and Dvorak is the *Colemak* layout [5], introduced by Shai Coleman in 2006. It maintains the position of 17 keys of QWERTY, including many keys commonly used for keyboard shortcuts, with the hope of making it easier to learn for people accustomed to the QWERTY layout. While Colemak is not officially supported by Windows, creating and installing a custom layout in Windows can be easily done with the Microsoft Keyboard Layout Creator. Once a layout is installed, the characters appearing on the physical keys will not match the virtual characters, but this is a very minor issue; on the contrary, it encourages early adoption of touch typing.

Subsequently there were several attempts to find better keyboard layouts by automating the process [6], [7]. While a brute-force search over all possible arrangements is not feasible, due to the astronomically large number of different arrangements, there are many optimization algorithms that can be used instead. One example of a commonly used and efficient class of optimization algorithms is the genetic algorithm.

Genetic algorithms belong to the larger class of evolutionary algorithms. They are a technique inspired by the process of natural selection, which are commonly used to generate high-quality solutions to optimization and search problems by relying on the biologically inspired operations of mutation, crossover and selection. In a genetic algorithm there is a "population" of candidate solutions, each of which has a set of characteristics that can be altered. There is an objective function that assigns a "fitness" value to each solution. One typically starts with an initial random population, which will probably have very low fitness. The algorithm proceeds in "generations"; each generation is obtained from the previous one by selecting the most fit candidates and generating new candidates by a process of crossover. In addition, random mutations are performed on the selected candidates before being added to the next generation. While most of the crossovers and mutations are likely to reduce the fitness of the candidates, a small fraction of them will yield more-fit candidates, and the improved traits will gradually spread throughout the population. Hence, as the generations progress, the overall fitness of the population will increase.

In this paper we present a method for optimizing keyboard layouts using a hybrid approach of deep learning and genetic algorithms. Our method is fast and therefore allows the use of large corpora for training, as well as the use of complex fitness functions. One of the features of our method is the use of the *cycle crossover* routine [8], which greatly enhances the performance of the genetic part of the algorithm. We show that our method outperforms the state-of-the-art methods from the

literature even when using their own metrics[1].

As mentioned in [7], keyboard layout optimization techniques might be useful for a wider class of problems in which there are objects that must be placed in predefined locations, the objects will be accessed one after the other in some order, and the goal is to optimize the placement of the objects. Real-life examples of this scenario include books in a library and products in a vending machine.

Finally, we tackle the issue of keyboard adaptation. We examine whether experienced QWERTY typists adapt better to new layouts than inexperienced typists, whether, once experimenting with a new layout, users find it easier to adapt to another new layout, and whether it is easier or harder to adapt to common letter combinations compared to rare letter combinations. For that end, we run an experiment with 300 participants and three different keyboard layouts, namely, the standard QWERTY layout and two new layouts for three different keys.

To summarize, our contribution in the area of automatic layout optimization is three-fold.

1) We propose the use of deep learning to assist in a genetic algorithm process for finding an improved keyboard layout.
2) We show that the cycle crossover routine significantly outperforms the crossover routine that was previously used in the literature.
3) We conduct a user study with 300 participants to examine how users adapt to new keyboard layouts.

II. Related Work

Genetic algorithms have been used for keyboard design optimization. Yin and Su [6] considered several scenarios for the general keyboard arrangement problem, such as single-character and multi-character keyboards, single-finger and multi-finger typing, and optimization according to different criteria, such as typing ergonomics, word disambiguation, and prediction effectiveness. They offered an evolutionary approach using a cyber swarm method and showed that it produces keyboard layouts that are better than existing ones. Other works that use genetic algorithms for keyboard optimization are [7], [9]–[11].

In particular, in their recent work, Fadel et al. [7] developed a genetic-based algorithm that is used to find better layouts than QWERTY and Dvorak. Their algorithm works by iteratively performing the operations of Selection, Crossover and Mutation, on a population of candidate layouts. They measure the fitness of a layout using a simple objective function that sums the Euclidean distances between every single character in the training corpus and the nearest finger to it. Using their method they find layouts that are better than QWERTY and Dvorak according to their objective function. They call the best keyboard layout they found "}.?BZQ".

[1]A link for installing the keyboard layout generated by our method on Windows is available at: `https://github.com/kerenivasch/MKLOGA`

Krzywinski [12] introduced *carpalx*, which includes a more realistic and complex objective function for evaluating layouts. The carpalx typing effort model is based on *triads*, which are three-character substrings formed from the training text. The effort associated with typing a triad has two components: effort to hit a key (independently of preceding and successive strokes) and effort to hit the group of keys. Independent effort is based on finger distance and includes hand, finger, and row penalties associated with that key. The effort associated with the group of keys considers their *stroke path*, which is evaluated by taking into account hand-alternation, row-alternation, and finger-alternation.

The carpalx model is highly parameterized, as the formula for the effort involves many weights whose value can be adjusted. Hence, the user can decide what is important to her layout, so the model can be made highly subjective. For more details on the computation of the carpalx effort model see [12]. For this project we left all the carpalx parameters with their default values and did not change them.

Due to its high complexity, the carpalx objective function requires excessive computing power (approximately 0.6 seconds on a computer with Intel Core i7 CPU).

The carpalx project also includes an implementation of a simulated annealing based method for finding good keyboard layouts. Carpalx has been used to construct layouts optimized for the Filipino [13] and Latvian languages [14].

A problem related to the keyboard layout optimization problem is the Quadratic Assignment Problem (QAP). In this problem there are n facilities and n locations, and there is a distance between each pair of facilities, as well as a flow between each pair of locations. The objective is to assign the facilities to different locations in order to minimize the sum of the distances multiplied by the corresponding flows. This problem is somewhat similar to the keyboard layout optimization problem: If the function we wish to minimize is the total movement of the fingers, then the keys and the finger base positions correspond to the facilities. There are several works that tackle the QAP problem with genetic algorithms [15]–[17].

There are also several previous works that combine genetic algorithms with deep learning. Sehgal et al. [18] use a genetic algorithm to find the values of parameters used in a reinforcement learning task related to robotic manipulation. Potapov and Rodionov [19] implement a genetic algorithm with a crossover operator that uses a deep neural network. Hu et al. [20] combine a genetic algorithm and deep neural network models to construct property diagrams for grain boundaries.

Recently, Klein [21] developed a multi-step approach for generating keyboard layouts, with which he designed a new layout called *Engram*.

III. The MKLOGA Model

In this paper, we present our *Method for Keyboard Layout Optimization using a deep Genetic Algorithm (MKLOGA)*. The method improves the one described by Fadel et al. [7] in several aspects. First, MKLOGA uses a better crossover routine

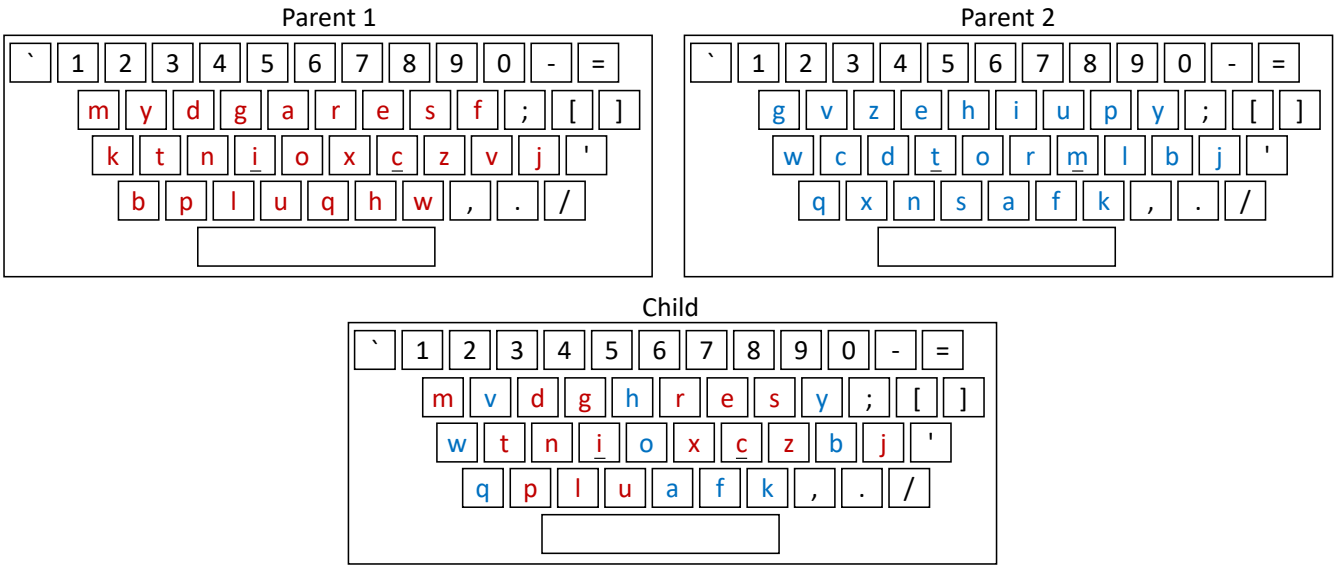


Fig. 1: Example of the cycle crossover routine.

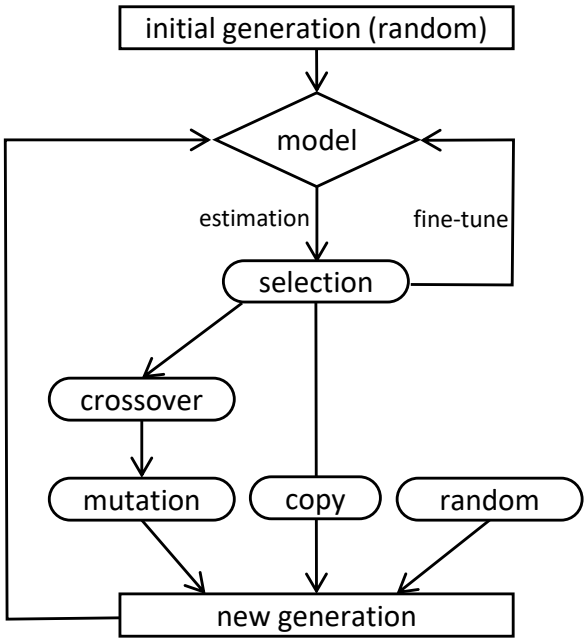


Fig. 2: MKLOGA flowchart.

for generating a new layout from its parents, as explained in section III-A below. In addition, MKLOGA uses the complex and more realistic objective function of carpalx [12] for evaluating layouts. Due to the excessive computing power required by the carpalx objective function, MKLOGA includes several improvements to the genetic algorithm process, one of which is the use of deep learning. All MKLOGA software is available at `https://github.com/kerenivasch/MKLOGA`.

A. The Cycle Crossover Routine

As mentioned above, MKLOGA uses the *cycle crossover* routine of [8] for generating a new keyboard layout K_3 out of two given keyboard layouts K_1, K_2. The crucial property of this routine (as opposed to the crossover routine of [7]) is that each key placement in K_3 is copied from either K_1 or K_2. As we show in Section V, the cycle crossover routine alone provides a significant improvement to the performance of the algorithm of [7].

We proceed to explain the cycle crossover routine for the

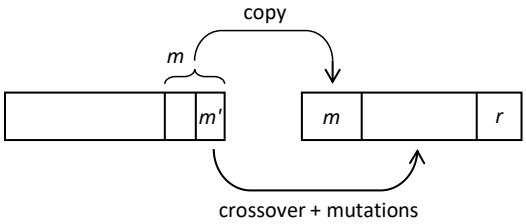


Fig. 3: The construction of the new generation from the previous one. First, r layouts are generated randomly. The best m layouts of the previous generation (according to the model estimate) are directly copied to the new generation, and the best m' of them (according to the true effort value) are used to generate new layouts through crossover and mutations.

sake of completeness. Let S be the set of symbols whose placement is allowed to change, and let P be the set of keys that can take symbols. We first pick a random parent K from among K_1, K_2, we call the other parent K'. Then we pick a random key $p_1 \in P$ and copy its symbol s_1 from K to K_3. Now we check in which key p_2, the symbol s_1 is located in K'. We copy the symbol s_2 of the key p_2 in K. Then we check in which key p_3, the symbol s_2 is located in K'. We continue this way until we return to p_1 and thus close a cycle. Hence, all the symbols in this cycle were copied from K.

If additional keys are left, we make another random choice for K, K' between K_1, K_2, and pick another random available key and repeat the process. This way every placement in K_3 has been copied from either K_1 or K_2.

Figure 1 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 location of the letter `n` 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.

B. Using The Carpalx Objective Function

As mentioned, in order to obtain an improved keyboard layout, MKLOGA uses the complex and more realistic keyboard effort model of carpalx [12] for evaluating layouts. Since the keyboard effort model requires excessive computing power to evaluate, MKLOGA also includes a neural network for fast estimation of the effort. The neural network is initially trained on randomly generated layouts. After the training, the model is saved in order to be used as the initial model for the genetic part of the algorithm. During the genetic algorithm process, the neural network is fine-tuned by retraining it with some of the best layouts found in the current generation using their true effort value. The input layouts for the neural network are represented using a one-hot representation, as a square $0/1$-matrix whose size corresponds to the number of key positions that are allowed to change. The use of the neural network allows us to evaluate the expensive effort function only on a small number of layouts, leading to a significant speedup of the running time.

The genetic algorithm of MKLOGA proceeds as a sequence of generations. Each generation consists of a population of n layouts. The first generation is generated randomly. In each generation the layouts are evaluated and sorted according to the neural network's estimation. In order to construct the new generation, r layouts are first generated randomly. The best m layouts of the previous generation pass automatically to the new generation, and they are also evaluated according to the true effort function. The best m' of these are used to generate $n - r - m$ new layouts using the cycle crossover routine. Each new generated layouts also undergoes a random number between 0 and t of random mutations. Each mutation consists of selecting 2 random keys and swapping them. Figure 2 shows a flowchart of the MKLOGA algorithm, and Figure 3 shows how a generation is constructed from the previous one.

IV. Experiments

We first evaluated the effect of using the cycle crossover routine. For this, we took the code of [7], and replaced their crossover routine with the one described in Section III-A. We carried out the two types of experiments that were made by [7]: changing only the positions of the letters of the standard layout (called "Letters Only" in [7]), and changing also the positions of the punctuation symbols ("Letters and Punctuation").

We then proceeded to implement MKLOGA. The first step of the implementation was to train a neural network on a data-set of 4800 randomly generated layouts labeled with their effort values. The neural network had a hidden layer of size 64 with ReLU activation. For the genetic part, we used a population size of $n = 5000$, the number of random layouts added in each generation was $r = 1000$, the parameter m was 250, and m' was 100. The maximum number of mutations was $t = 5$. We ran the genetic algorithm for 30 generations.

As a first step, we allowed to change only the positions of the letters, except for one difference: Following the lead of some previous keyboard designs (carpalx [12], colemak [5]) we moved one letter from the top row of letters to the middle row, so that the top row contains 9 letters and the middle row contains 10 letters. As a second step, we also allowed to change the positions of the punctuation symbols as in [7].

The effort value was calculated using a corpus provided by [12] of size 267KB.

V. Results

A. The Cycle Crossover Routine

The cycle crossover routine led to a significant improvement in the performance of the genetic algorithm. As depicted

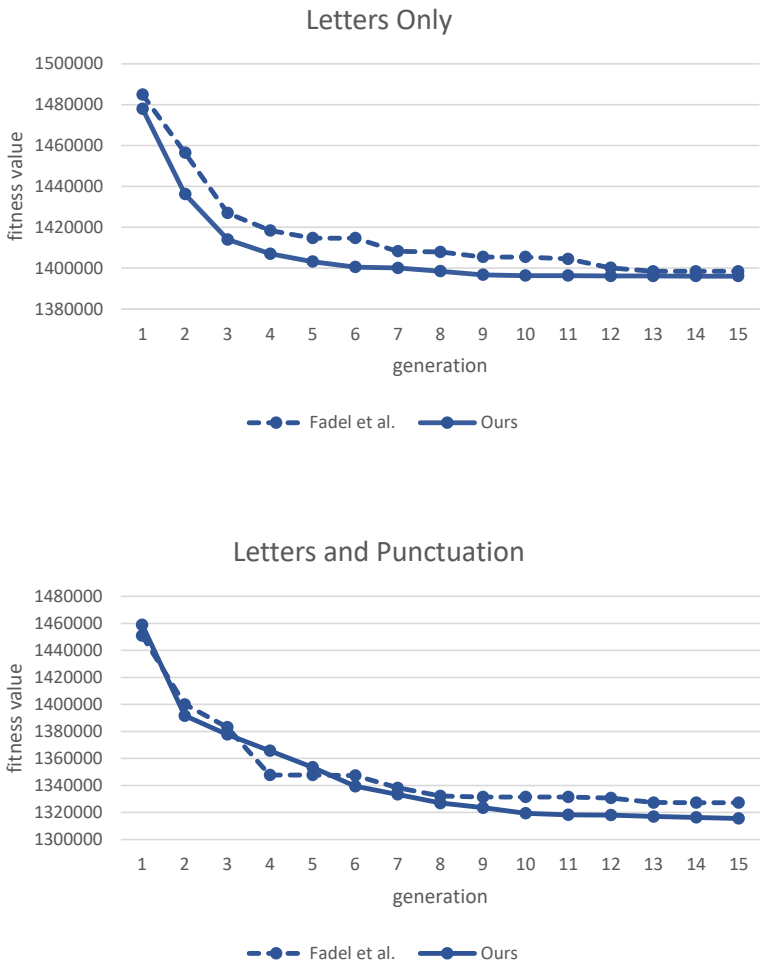


Fig. 4: A comparison between the performance of the genetic algorithm using the crossover method proposed by [7] and the cycle crossover method. The performance is measured according to the objective function defined in [7] (lower is better).

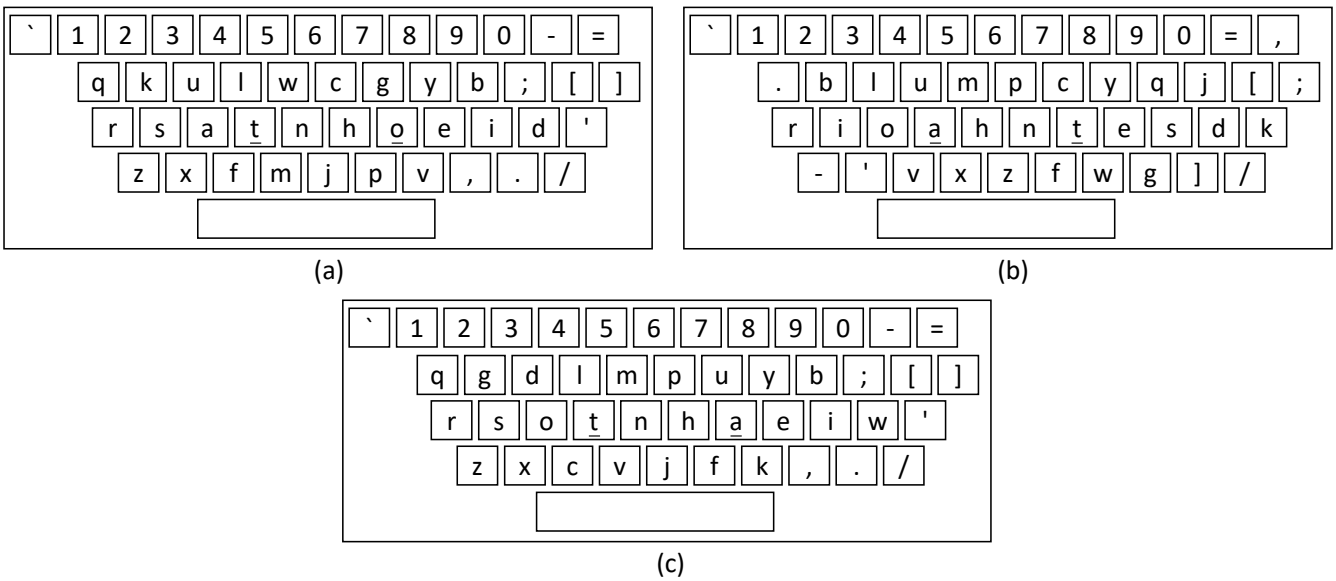
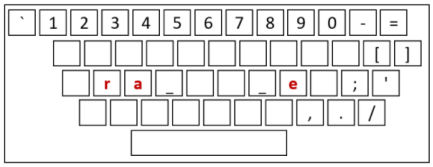


Fig. 5: (a) Best keyboard layout found by MKLOGA when moving only letter positions. (b) Best keyboard layout found by MKLOGA when moving also punctuation symbols. (c) Another keyboard layout found by MKLOGA, with ZXCV in place.

Let's start with a test on the QWERTY layout. Please type the following:

oym gano ca qip ljnvymjtsrbwl p nezrlawrztwwxk h rrqr uge



Now, use the layout shown above. Let's start with a warm up:

eaae arrrrr raareraaaraar aaeearae rrareerra ee aeee a ee are

Great! Now type the following using this layout:

are are era era are are ear are are are era are ear are a re are ear are era era

Now, use the new layout shown above. Let's start with a warm up:

qqzqjzqzq z qq zjqqqjzqq zj z qzjqqzqqjzz j z z qqqzzjqjj jq

Great! Now type the following using this layout:

jqz zqj jqz zjq zqj zjq zjq zqj zqj zjq zqj zjq jqz zqj j qz zjq zjq zjq jqz zqj

Fig. 6: The typing experiments

TABLE I: Results with the cycle crossover

	Letters Only		Letters & Punct.	
	effort	gens	effort	gens
Fadel et al.	1394663.75	88	1312948.02	97
cycle crossover	1394663.75	16	1311932.84	27

TABLE II: The carpalx effort value of different keyboards.

keyboard	effort
Qwerty	2.962
Dvorak	2.046
Colemak	1.796
Carpalx Sim. Ann.	2.038
}.?BZQ (Fadel et al.)	2.508
MKLOGA Letters Only	1.625
MKLOGA Letters & Punct.	**1.612**

in Figure 4, with the cycle crossover the objective function decreased much faster. Furthermore, for the case of Letters Only, with the old crossover routine, the genetic algorithm reached the final objective function value at generation 88, whereas with the cycle crossover routine this same value was achieved at generation 16. For the case of Letters and Punctuation, the cycle crossover yielded a lower final objective function value and it was also achieved in a much earlier generation. See Table I.

B. MKLOGA

As mentioned above, MKLOGA fine-tunes the neural network model during the course of the genetic algorithm. In our experiments, the loss of the model decreased from 0.048 to 0.0028 when moving only the letter positions, and from 0.076 to 0.00098 when moving also the punctuation symbols. Hence, the neural network model's prediction accuracy improved during the course of the execution.

When we ran MKLOGA moving only the letter positions, MKLOGA found a layout with an effort value of 1.625, and it did so already at generation 19. See Figure 5 (a). When we let MKLOGA move also the punctuation symbols, it found a layout with an effort value of 1.612, at generation 42. See Figure 5 (b). For comparison, the layout found by Fadel et al. [7] has an effort value of 2.508 (though, as mentioned above, they optimized for a different objective function). We also ran the simulated annealing code of [12] using its default parameters. We did so 10 times and took the average effort value of the produced layouts. See Table II, which also shows the effort value of a few other well known layouts, for comparison. Moreover, the neural network model of MKLOGA takes approximately only one millisecond to estimate the effort value of a layout, which is much faster than calculating the true effort.

VI. New Layout Adaptation Experiments

In this section we study the ability of humans to adapt to new layouts. For that end we carry out an experiment conducted with humans using multiple keyboard layouts.

A. Working Hypotheses

Our primary goal of the new layout adaptation experiments is to test the following hypotheses:

1) Do experienced QWERTY typists adapt better to new layouts than inexperienced typists?
2) Once experimenting with a new layout, would the users find it easier to adapt to another new layout?
3) Do users find it easier to adapt to common letter combinations than to rare letter combinations, when presented with a new layout? On the one hand, the user might recognize more quickly common letter combinations, and will also learn more quickly to type them. On the other hand, perhaps the old ingrained habits might cause the user to get confused.

B. Experimental Design

We conduct an experiment in which users are requested to type certain letter combinations using different keyboard layouts. In the first phase of the experiment, the users are presented a random text of length 60 that is composed of letters and spaces. The users are requested to type this text using the standard QWERTY layout. The experiment then continues with two additional phases (phases *two* and *three*), each focusing on a different partial keyboard layout.

One partial layout consists of the letters A, E, R located in the positions of D, K, S of the QWERTY layout. The second partial layout consists of the letters J, Z, Q located in these same positions. For each partial layout, the users are given two typing assignments. The first assignment is only used as a warm-up and contains a random collection of the three letters and spaces, of length 60. The second assignment, for which we record results, consists of 20 triplets, separated by spaces. For the *AER* partial layout, the triplets are *ear*, *are*, and *era*, which are actual English words. For the *JZQ* partial layout, the triplets are *zjq*, *jqz*, and *zqj*, which use the same keys in the same order. One of the partial layouts is used for phase two and the other for phase three, but the order between them is chosen at random for each user. Figure 6 depicts the different phases and assignments of the experiment. In all the phases, the user cannot proceed to type the next letter until she typed correctly the current letter.

Each user is asked a few questions about themselves (age, gender, education, and previous experience with touch typing).

We use the Mechanical Turk to run this experiment with 300 different people. A software bug caused four of the users' data records to become invalid. Therefore, 296 data records are used in our analysis.

C. Experimental Results

As expected, users made significantly fewer mistakes with the QWERTY layout (average 6.4) than with the two new layouts (average 54.5) ($p < 0.0001$).

Quite surprisingly, we found a very small correlation between the number of errors in QWERTY and the number of errors in the two new layouts (correlation coefficients of 0.08 and 0.11 for *AER* and *JZQ* respectively). In other words,

users that were good at QWERTY were not necessarily good in other new layouts. On the other hand, we found a very strong correlation between the number of errors in the two new layouts (correlation coefficient of 0.89). Hence, users that were good at one new layout were also good at the other one. We also note that learning the second new layout seems faster than learning the first new layout (regardless of which layout is being learned). Indeed, users completed the second phase significantly faster than the first (80.3s for the first and 63.2s for the second). This difference is statistically significant ($p < 0.05$ using one-tail t-test). Interestingly however, there was no statistically significant difference between the number of mistakes in the first and second phases (53.2 and 55.7 respectively).

We were expecting a more significant difference between the *AER* and *JZQ* experiments, since in the former the triplets consisted of common characters forming actual English words, but not in the latter. However, the difference in the number of mistakes was not statistically significant (average 56.2 for *AER*, 52.7 for *JZQ*), nor was the difference in the total typing times (average 74.7s for *AER*, 68.8s for *JZQ*). A more in-depth study is required here.

Interestingly, we found no significant correlation between the users' performance (number of errors or typing time) and their age, gender, education or experience with touch typing.

Observations on specific keys: We took a look at the number of mistakes users made on typing each letter while using the different layouts. See Table III. In the QWERTY phase, users made most mistakes in keys V and J (134 and 131 respectively), and they made the fewer mistakes in keys M and I (20 for each). Interestingly, in the two other phases, users made the most mistakes in the letter located in the K position (letters E and Z of the new layouts), and the fewest mistakes in the letter located in the D position (letters A and J of the new layouts).

Since users made many mistakes when typing J in the QWERTY layout but they made few mistakes when typing this letter using the *JZQ* layout, this seems to indicate that the location of the letter is more significant than the letter itself.

Finally, we looked at the average time taken to type each letter during the QWERTY phase. See Table IV. For comparison, the table also shows the Euclidean distances used by [7] for evaluating layout fitness. There is a correlation of -0.147, which seems to indicate that the Euclidean distance is a poor measure of keyboard layout fitness.

VII. Discussion & Future Work

As mentioned above, the best keyboard layout found by MKLOGA achieves an effort value of 1.612. In comparison, the best layout offered by the carpalx project [12] (the one they refer to as "qgmlwb") achieves an effort value of 1.629 (in their website they give a value of 1.668, the difference being due to the use of a larger corpus). However, the layout that they recommend (which they refer to as "qgmlwy") leaves the keys Z, X, C, V, in the classical QWERTY positions due to their frequent use in keyboard shortcuts, and it achieves an effort value of 1.635 (1.670 according to their website). Interestingly, in one run, MKLOGA produced a keyboard with the keys Z, X, C, V, in place, with an effort value of 1.633. See Figure 5 (c). Finally, it is worth noting that the above-mentioned carpalx layouts were probably built with human assistance and not purely with a computer algorithm.

MKLOGA may be useful in developing good keyboard layouts for languages other than English. Also, as mentioned in [7], there is a need for left-hand only and right-hand only layouts for handicapped people. There is an increasing need for good layouts for smartphone keyboards, in which people type with only one finger.

In current form, MKLOGA calculates the effort value with a relatively small corpus; we plan to switch to a larger corpus in order to get more accurate results.

In addition, we plan to implement an even more realistic objective function that takes into account other relevant factors in typing. Also, we suspect that the carpalx effort model is not realistic enough, because it assigns a much better score to the Dvorak keyboard than to QWERTY, despite the fact that research shows that among experienced typewriters, it does not make much of a difference whether they use the Dvorak or the QWERTY layout. We would like a more realistic function that will explain this counter-intuitive fact so we can construct a keyboard layout that is truly better.

In the adaption experiments, we tested three different keyboard layouts. The results obtained seem to point that experienced QWERTY typists do not adapt better to new layouts, that once experimenting with a new layout users find it easier to adapt to another layout, and that the use of common letter combinations does not have a major impact on the difficulty to adapt to a new layout. However, the two new keyboard layouts were limited to three keys, since we did not want to require the participants to commit to a long-term study, and asking the participants to learn a full keyboard layout in a short time-frame seems infeasible. In future work, we intend to perform a longitudinal study and hope to extend the results obtained in this paper to an entire keyboard layout.

VIII. Conclusion

Despite the fact that the QWERTY keyboard layout was designed more than 100 years ago, it is still the predominant layout in use today. There have been several attempts to design better layouts, both manually and automatically. In this paper we proposed MKLOGA, which improves on previous works on automatic keyboard layout optimization, by using a deep neural network to assist in a genetic search algorithm. As we showed, MKLOGA enables the use of a sophisticated keyboard evaluation function that would otherwise take a prohibitive amount of time. We also showed that the cycle crossover routine greatly improves the genetic search. MKLOGA produced a better keyboard layout than previous algorithms, according to the realistic typing effort model of carpalx [12]. MKLOGA might be useful for developing good layouts for languages other than English, and for other situations in which objects must be placed in predefined locations.

TABLE III: Number of mistakes made by users in each letter. (a) QWERTY layout, (b) AER layout, (c) JZQ layout.

(a)

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	space
48	66	8	22	78	32	103	49	**20**	**131**	94	71	**20**	46	83	71	31	31	25	41	31	**134**	79	49	60	88	336

(b)

a	e	r	space
3297	3751	3695	5908

(c)

j	z	q	space
2966	3966	3291	5401

TABLE IV: Average typing time (seconds) of each letter in the QWERTY phase (first row) compared to the Euclidean distance used by [7] (second row).

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1.43	0.89	0.91	0.82	1.53	0.98	1.1	1.3	0.64	1.16	1.09	0.98	0.65	0.72	0.77	1.01	1.02	0.76	0.75	0.88	0.74	1.59	0.91	0.97	0.98	0.98
1.08	2.37	1.1	1.08	2.65	1.08	1.34	2.76	2.65	1.08	1.08	1.08	1.1	1.98	2.65	2.65	2.88	2.65	1.08	2.65	2.88	1.1	2.65	1.1	3.58	1.1

Finally, we conducted some layout adaptation experiments with 300 participants in order to examine how users adapt to new keyboard layouts. We found that experience in QWERTY typing does not seem to make a difference in adapting better to new layouts, that once experimenting with a new layout users find it easier to adapt to another layout, and that the use of common letter combinations does not have a major impact on the difficulty to adapt to a new layout.

ACKNOWLEDGMENT

This research was supported in part by the Ministry of Science, Technology & Space, Israel.

REFERENCES

[1] J. Stamp, "Fact of fiction? the legend of the QWERTY keyboard," 2013, https://www.smithsonianmag.com/arts-culture/fact-of-fiction-the-legend-of-the-qwerty-keyboard-49863249/.

[2] J. Noyes, "The qwerty keyboard: A review," *International Journal of Man-Machine Studies*, vol. 18, no. 3, pp. 265–281, 1983.

[3] N. Baker, "Why do we all use qwerty keyboards?" 2010, https://www.bbc.com/news/technology-10925456.

[4] The Age, "Wrist pain? try the Dvorak keyboard," 2004, https://www.theage.com.au/technology/wrist-pain-try-the-dvorak-keyboard-20041210-gdka9g.html.

[5] S. Coleman, "Colemak," 2006, http://colemak.com.

[6] P.-Y. Yin and E.-P. Su, "Cyber swarm optimization for general keyboard arrangement problem," *International Journal of Industrial Ergonomics*, vol. 41, no. 1, pp. 43–52, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169814110001058

[7] A. Fadel, I. Tuffaha, M. Al-Ayyoub, and Y. Jararwch, "Qwerty keyboard? .?BZQ is better!" in *2020 International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*. IEEE, 2020, pp. 81–86.

[8] I. Oliver, D. Smith, and J. R. Holland, "Study of permutation crossover operators on the traveling salesman problem," in *Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA*. Hillsdale, NJ: L. Erlhaum Associates, 1987., 1987.

[9] C. Liao and P. Choe, "Chinese keyboard layout design based on polyphone disambiguation and a genetic algorithm," *International Journal of Human–Computer Interaction*, vol. 29, no. 6, pp. 391–403, 2013. [Online]. Available: https://doi.org/10.1080/10447318.2013.777827

[10] M. Govind and V. V. Panicker, "Optimization of a single finger keyboard layout using genetic algorithm and topsis," *International Journal of Scientific & Engineering Research*, vol. 7, no. 2, pp. 102–105, 2016.

[11] A. H. H. Onsorodi and O. Korhan, "Application of a genetic algorithm to the keyboard layout problem," *PLOS ONE*, vol. 15, no. 1, pp. 1–11, 01 2020. [Online]. Available: https://doi.org/10.1371/journal.pone.0226611

[12] M. Krzywinski, "Carpalx keyboard layout optimizer," 2005, http://mkweb.bcgsc.ca/carpalx/.

[13] J. M. R. Salvo, C. J. B. Raagas, M. T. C. M. Medina, and A. J. A. Portus, "Ergonomic keyboard layout designed for the filipino language," in *Advances in Physical Ergonomics and Human Factors*. Springer, 2016, pp. 407–416.

[14] V. Vitolins, "Modernized latvian ergonomic keyboard," *arXiv preprint arXiv:1707.03753*, 2017.

[15] H. Azarbonyad and R. Babazadeh, "A genetic algorithm for solving quadratic assignment problem (qap)," in *Proceeding of 5th International Conference of Iranian Operations Research Society (ICIORS), Tabriz, Iran*, 2012. [Online]. Available: https://arxiv.org/abs/1405.5050

[16] R. K. Ahuja, J. B. Orlin, and A. Tiwari, "A greedy genetic algorithm for the quadratic assignment problem," *Computers & Operations Research*, vol. 27, no. 10, pp. 917–934, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054899000672

[17] A. Hameed, B. Aboobaider, M. Mutar, and N. Choon, "A new hybrid approach based on discrete differential evolution algorithm to enhancement solutions of quadratic assignment problem," *International Journal of Industrial Engineering Computations*, vol. 11, no. 1, pp. 51–72, 2020.

[18] A. Sehgal, H. La, S. Louis, and H. Nguyen, "Deep reinforcement learning using genetic algorithm for parameter optimization," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, 2019, pp. 596–601.

[19] A. Potapov and S. Rodionov, "Genetic algorithms with dnn-based trainable crossover as an example of partial specialization of general search," in *International Conference on Artificial General Intelligence*. Springer, 2017, pp. 101–111.

[20] C. Hu, Y. Zuo, C. Chen, S. P. Ong, and J. Luo, "Genetic algorithm-guided deep learning of grain boundary diagrams: addressing the challenge of five degrees of freedom," *Materials Today*, vol. 38, pp. 49–57, 2020.

[21] A. Klein, "Engram: a systematic approach to optimize keyboard layouts for touch typing, with example for the English language," 2021.