Using a Recurrent Neural Network to Judge Quality of Human Produced Random Sequences

Julius Villamayor
CSC 497
University of Victoria
April 3, 2018

This paper explores the use of recurrent neural networks (RNNs) for sequence prediction on random data created by human participants. Participants were asked to move a tile within a 4x4 grid with the arrow keys of a keyboard. The goal was to move as randomly as possible. The resulting sequences were then trained on neural networks. Predictions on the moves of achieved a mean total accuracy of 0.441, 1.33 times higher than the expected accuracy on truly random data ($t(138) = 15.347$, $p < .001$). When splitting each participant's predicted results into 2 halves, the accuracy of the first half compared to the second half had a mean difference of -0.0005, and a moderately strong relationship, $r = .6068$ ($p < .001$). This suggests that RNNs can be used to compare random sequences from different sources for relative unpredictability.

Introduction

It is well documented in literature that humans have certain biases when perceiving randomness. For example, a meta-analysis from Bar-Hillel & Wagenaar (1991) found that when examining long, random, binary sequences with equal probability of 1 or 0, humans expect

more alternation (or shorter streaks of the same digit) than there actually are. This finding is consistent when judging and producing random sequences, as well as across feature dimensions, sensory modalities, presentation modes, and probing methods (Yu, Gunn, Osherson, & Zhao, 2017).

Related to the alternation bias is a bias for local representation. In random sequences, you can expect to find every option to be equally represented over a large sample. However, when looking at smaller samples, runs can occur that may result in subsamples with unequal distributions. While this is common in truly random sequences, human-produced randomness often overemphasizes local representation; i.e., representing all options equally within smaller subsamples (Yu et al., 2017). This manifests in what has been called the 1st order sequential effect: After a given selection, the same selection being chosen again occurs less than random chance. A truly random sequence would not have any effects based on previously made decisions.

Because randomness is a measure of lacking pattern or predictability, the current paper attempts to use recurrent neural networks to judge for randomness. Testing for randomness can be quite complicated, as there are many dimensions for testing for randomness. For example, the CAcert Research Lab has a Random Number Generator Analysis that tests for 7 dimensions of randomness: Entropy, Birthday Spacing, Matrix Ranks, 6x8 Matrix Ranks, Minimum Distance, Random Spheres, Squeeze (CAcert, 2018). While by no means a robust, absolute measure, a neural network could theoretically be used to compare two sequences for relative predictability just by simply making predictions on the sequences.

In this paper, the randomness of sequences were measured based on their prediction scores from an RNN. Unlike other neural networks, RNNs are distinct in that previous hidden state values are passed to the current hidden states to be considered along with the current input. This gives the networks the ability to remember sequences and exhibit dynamic temporal behavior. RNNs have been used for pattern and sequence recognition, and for making predictions in a variety of contexts. By learning from training sequences, RNNs can be used to generate sequences similar to the ones learned on. For example, Graves (2013) trained an RNN to produce handwritten text. Similarly, Chang, Martini, & Culurciello (2015) were able to use an

RNN to produce new pseudo-Shakespeare after being trained on a Shakespeare database. If an RNN is able to achieve a high accuracy on a sequence, then the sequence must have poor randomness. On the other hand, if the RNN cannot make predictions better than chance, then it suggests the sequence has good randomness.

Network Architecture

Using Keras and Python, a simple long short-term memory (LSTM) RNN was created with one hidden layer and Adam optimization. Experiments were run on a subset of the data (N = 40) for fine-tuning hyperparameters. These results are shown below in Figure 1.

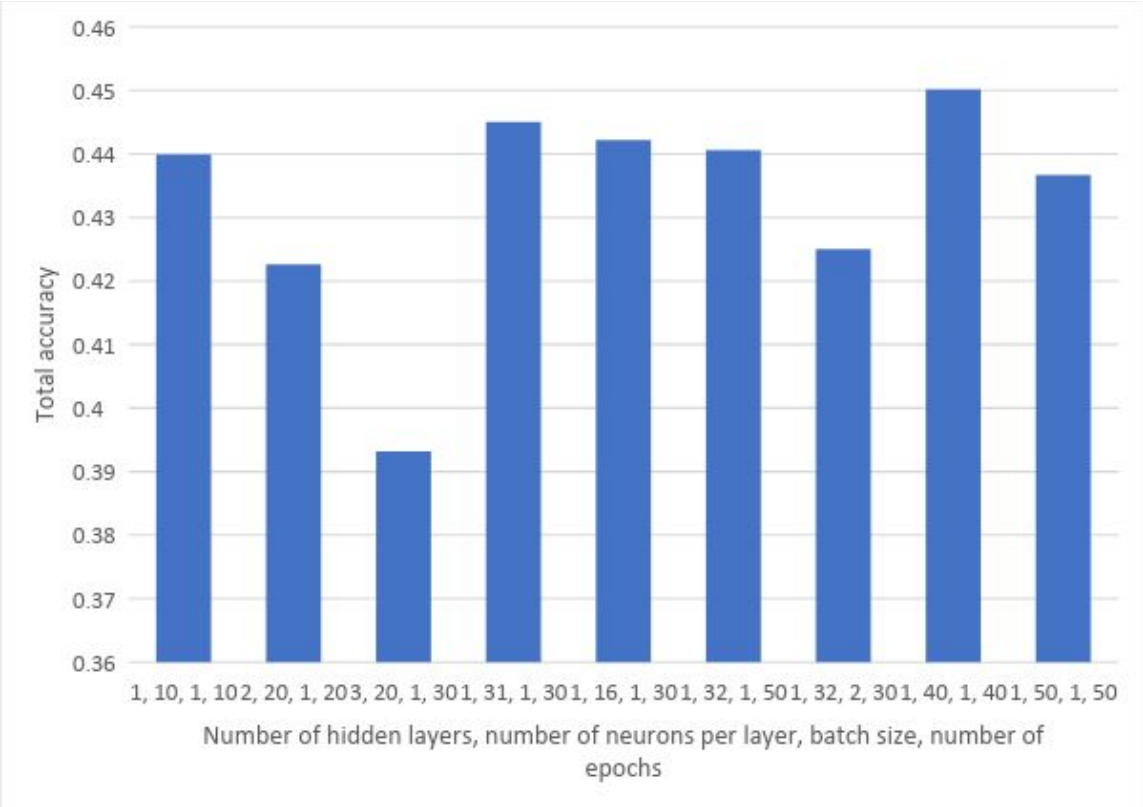

Figure 1. Accuracies of LSTMs with different hyperparameter values.

The results of these tests led to the selection of 1 hidden layer, 40 neurons per layer, a batch size of 1, and 40 epochs for configuration and training of all models. For each participant and their sequence, a new LSTM model is trained, independent of the models built for other

participants. Since the goal of the classifiers is to compare different sequences for quality of randomness, the hyperparameters are kept the same for every participant in order to ensure consistency.

Methods

The study was posted on the UVic Psychology Participation Pool. After participants viewed and volunteered for the study, they were presented with a consent form and task instructions. The instructions told participants to move randomly within a 4x4 grid using the arrow keys on their keyboard. They were then presented with the task, as visualized in Figure 2, and were able to complete the task on their own computers in a web browser.

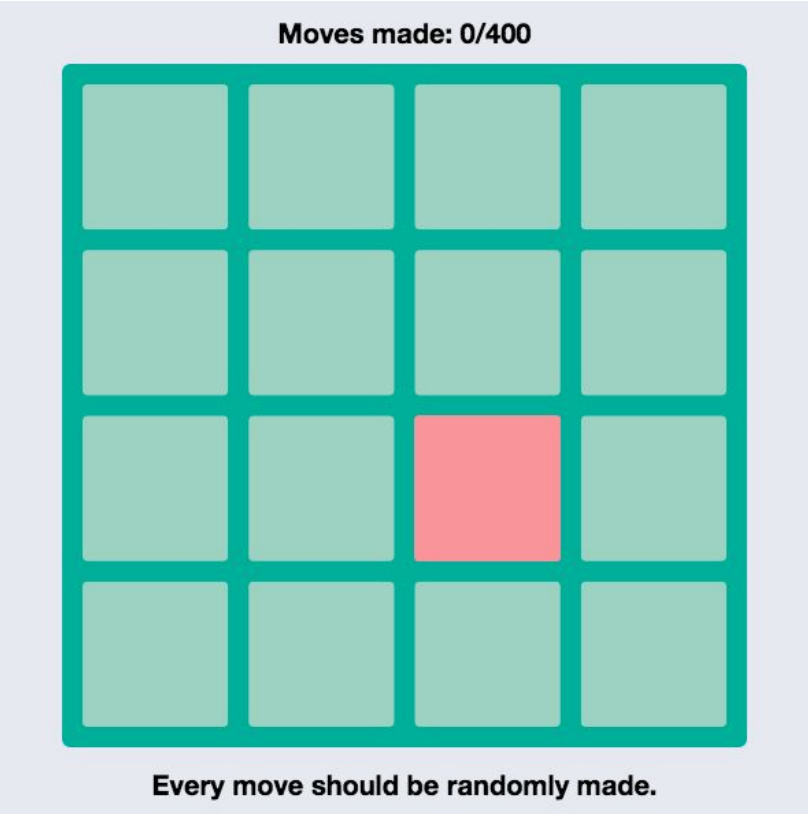

Figure 2. An screenshot of the game as presented to users.

Following the task, participants completed a post-experiment questionnaire collecting demographic information, and then finally were presented with a debriefing form.

139 participants completed the study. 29 participants were male, while 110 were female. The mean age was 23. Each participant made 1200 moves (3 blocks of 400-move trials). Participants were primarily undergraduate students from the University of Victoria.

Sanity Check: Expected Values and Results on Random Data

Expected Values

There are several consequences that result from the presentation of the 4x4 grid, even assuming perfectly random decision-making (i.e., every option being selected with equal probability). A straight-forward consequence is that after tallying all key presses, each key press should have been pressed with approximately equal distributions. That is, after 1200 moves, you can expect to see approximately 400 moves each for up, down, left, and right key presses.

Another consequence results from the spatial features of the 4x4 grid. Within the grid, there are 4 corner tiles, 8 side tiles, and 4 middle tiles. These 3 types of tiles have varying numbers of neighbouring tiles. As a result, these neighbouring tiles dictate both the probability of the tile being visited from another tile, as well as the number of options for moving to another tile. Corner tiles only have 2 options when selecting a next move. Side tiles have 3 options when selecting a next move. And middle tiles have 4 options when selecting the next move. So on completely random and unpredictable data, we would expect to be successful in a prediction on a corner, side, and middle tile ½, ⅓, and ¼ of the time, respectively. This is visualized in Figure 3.

.500	.333	.333	.500
.333	.250	.250	.333
.333	.250	.250	.333
.500	.333	.333	.500

Figure 3. Expected per tile accuracies on truly random sequences.

Assuming equal distributions of directions from each tile, a completely random sequence would converge on a steady state of visit distributions after a sufficiently long number of moves (in fact, these expected values reach the steady state to 4 decimals after 12 moves, as determined by Markov process calculation). Figure 4 provides a visualization of these expected visit distributions.

.0417	.0625	.0625	.0417
.0625	.0833	.0833	.0625
.0625	.0833	.0833	.0625
.0417	.0625	.0625	.0417

Figure 4. Expected per tile visit distributions on a sufficiently long random sequence.

By combining the expected accuracies per tile and the expected visit distributions per tile, the following equation can be used to determine the expected total accuracy (i.e. proportion of correct predictions on directions given a random sequence):

$$\begin{aligned} acc_{expected} &= 4(\tfrac{1}{2})(0.0417) + 8(\tfrac{1}{3})(0.0625) + 4(\tfrac{1}{4})(0.0833) \\ &= 0.333 \end{aligned}$$

To summarize, there are three types of tiles within the grid with different expected direction and visit distributions. There are 4 corner tiles, with 2 options each for proceeding, and an expected visit frequency of 0.0417. There are 8 side tiles, with 3 options each for proceeding, and an expected visit frequency of 0.0625. And there are 4 middle tiles, with 4 options each for proceeding, and an expected visit frequency of 0.0833. A total accuracy of 0.333 is expected on truly random data.

Results on Random Data

The architecture was tested on randomly generated data to confirm the validity of the baselines established. Random data was produced by means of the default Python random library. 200 sequences were produced, each 1200 moves long, restricted to the same spatial limitations participants were restricted to. The same hyperparameters were used for the neural network.

Moves were evenly distributed between up, down, left, and right key presses. Mean presses were 300.440 (SD = 8.987, t(199) = 0.691, p = .491), 300.385 (SD = 8.947, t(199) = 0.607, p = .545), 299.530 (SD = 8.979, t(199) = -0.738, p = .461), and 299.645 (SD = 8.975, t(199) = -0.558, p = .577), respectively. None of the values were significantly different from the expected value of 300.

Tile visit distributions were as expected as well. The 4 corner tiles had an average visit rate of 0.04146 (SD = 0.003, t(199) = -1.157, p = .249) per move. The 8 side tiles had an average visit rate of 0.0624 (SD = 0.002, t(199) = -1.214, p = .227) per move. The 4 middle tiles had an average visit rate of 0.0838 (SD = 0.005, t(199) = 1.421, p = .157) per move.

Making predictions on the data achieved a total accuracy of .330, SD = 0.022. This was not different from the expected value of 0.333 (t(199) = -1.908, p = .058).

Quantitative Evaluation

The experiment involved 139 participants. For each participant, an LSTM was trained on their first 800 moves, and then tested on their last 400 moves.

Data

Simple examinations of the data showed biases in the way participants selected moves. Key presses were not equally distributed, overall biased towards left and right (although, some participants favoured up and down). Up was pressed an average of 290.763 times (SD = 30.839, t(138) = -3.531, p = .001) out of 1200 moves. Down was pressed an average of 291.014 times (SD = 30.854, t(138) = -3.433, p = .001) out of 1200 moves. Left was pressed an average of

308.957 times (SD = 30.854, t(138) = 3.423, p = .001) out of 1200 moves. Right was pressed an average of 309.266 times (SD = 30.844, t(138) = 3.542, p = .001) out of 1200 moves.

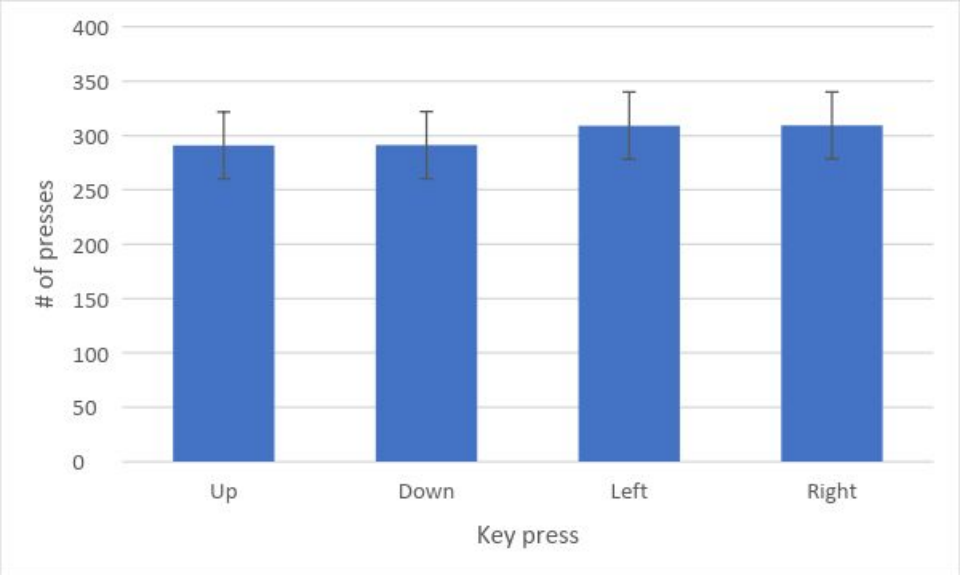

Figure 5. Mean key presses for 139 participants.

Interestingly, tile visits were also not distributed as expected. The participants heavily favoured the middle tiles. Middle tiles were visited on average 0.100 times per key press, 1.2 times more often than expected from random decision making (SD = 0.019, t(138) = 10.151, p < .001). Consequently, side tiles were visited 0.058 times per key press (SD = 0.005, t(38) = -9.546, p < .001), and corner tiles were visited 0.033 times per key press (SD = 0.009, t(138) = -10.517, p < .001).

The sequences were also examined for 1st order sequential effects. In order to have comparable numbers, the first order sequential effect was only examined for middle tiles. That is, if the user was in a middle tile, the last direction used to get there was compared to the next direction chosen. This was done in order to always have 4 potential directions to choose from. Therefore, truly random data would have an expected value of 0.250. Unexpectedly, instead of having a value less than 0.250, participants showed a first order sequential effect of 0.398, indicating a preference for repeating the same key in these situations (SD = 0.132, t(138) =

13.205, $p < .001$). This could be related to participants preferring to stay in middle tiles. The effect of this preference for middle tiles on total accuracy is explored below.

Predictions

The average total accuracy for participants on the last 400 moves was 0.441, 1.32 times higher than the expected value of 0.333 (SD = 0.074, $t(138) = 17.236$, $p < .001$). The classifiers were able to successfully and consistently make predictions better than chance. Accuracies were higher than expected values for every tile type. The mean accuracy for corner tiles was 0.636, 1.27 times higher than the expected value of 0.5 (SD = 0.120, $t(138) = 13.323$, $p < .001$). The mean accuracy for side tiles was 0.449, 1.35 times higher than the expected value of 0.333 (SD = 0.071, $t(138) = 19.253$, $p < .001$). The mean accuracy for middle tiles was 0.368, 1.47 times higher than the expected value of 0.250 (SD = 0.101, $t(138) = 13.800$, $p <.001$).

There was a notable relationship between the average seconds per move and total accuracy. The relationship between seconds per move and total accuracy had a moderate positive relationship ($r = 0.340$, $p < .001$), suggesting that participants who took longer with each move ended up being more predictable, compared to participants who made moves more quickly. This relationship is visualized below in Figure 6.

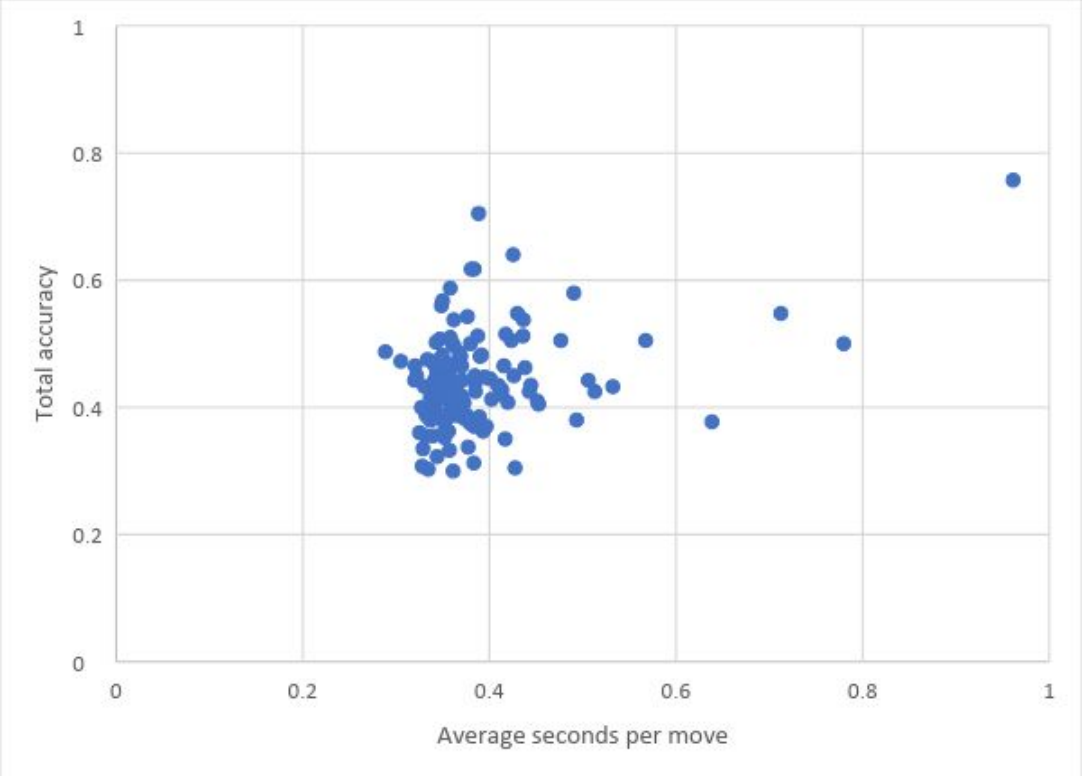

Figure 6. Total accuracies for 139 participants.

Figure 7 below plots accuracy relative to the middle tile visits per move, with the expected frequency as a vertical line for reference. There was a moderate negative correlation ($r = -0.289$, $p < .001$), suggesting that as participants favoured middle tiles, their total accuracy score was lower.

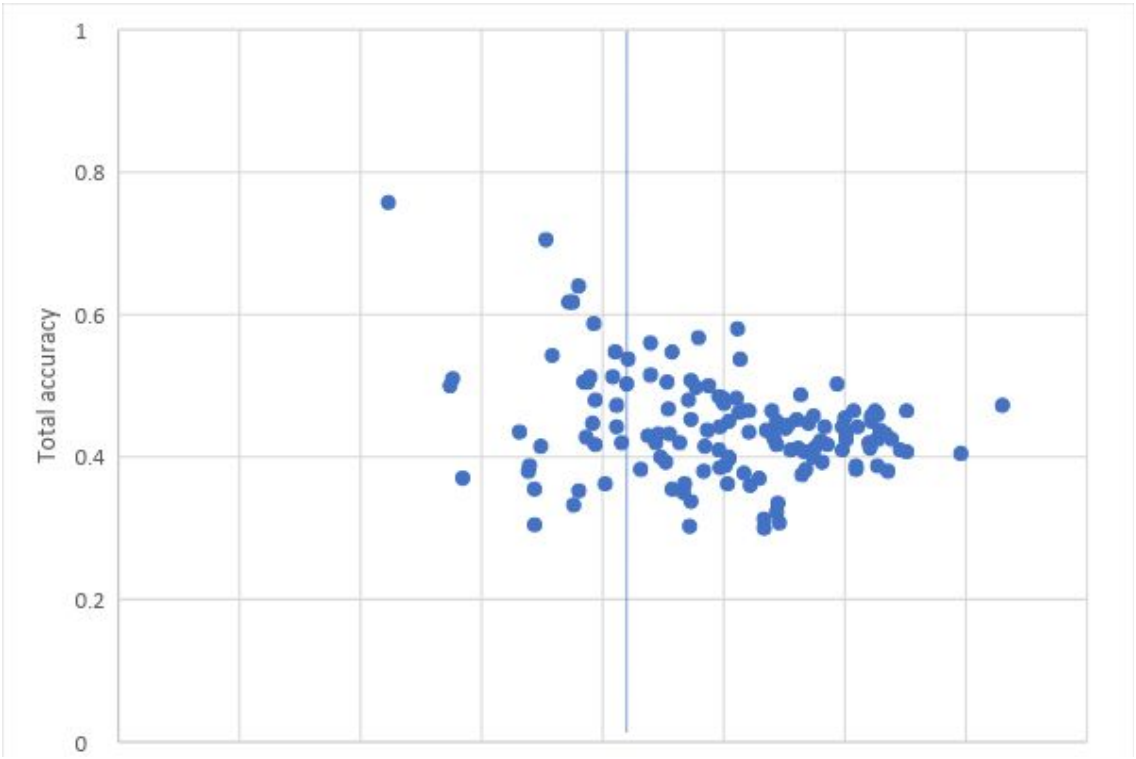

Figure 7. Total accuracies plotted against mean visit frequency of middle tiles per move.

Each person's accuracy is based on 400 predictions. Since we wish to determine the viability of using neural networks to compare random sequences for relative quality of randomness, reliability is of concern. In order to test the classifier for reliability, the 400 predictions are split into two halves the score of the first 200 predictions are compared to the second 200 predictions. The mean difference between these first half scores and the second half scores was -0.0005 (SD = 0.0740). This was not a significant difference ($t(138) = -0.0797$, $p = .937$). Correlating the first score with the second score shows a moderate-strong relationship ($r = 0.607$, $p < 0.001$). It seems that the classifiers are able to achieve consistent accuracies based on the person providing the sequences (and, therefore, the quality of unpredictability).

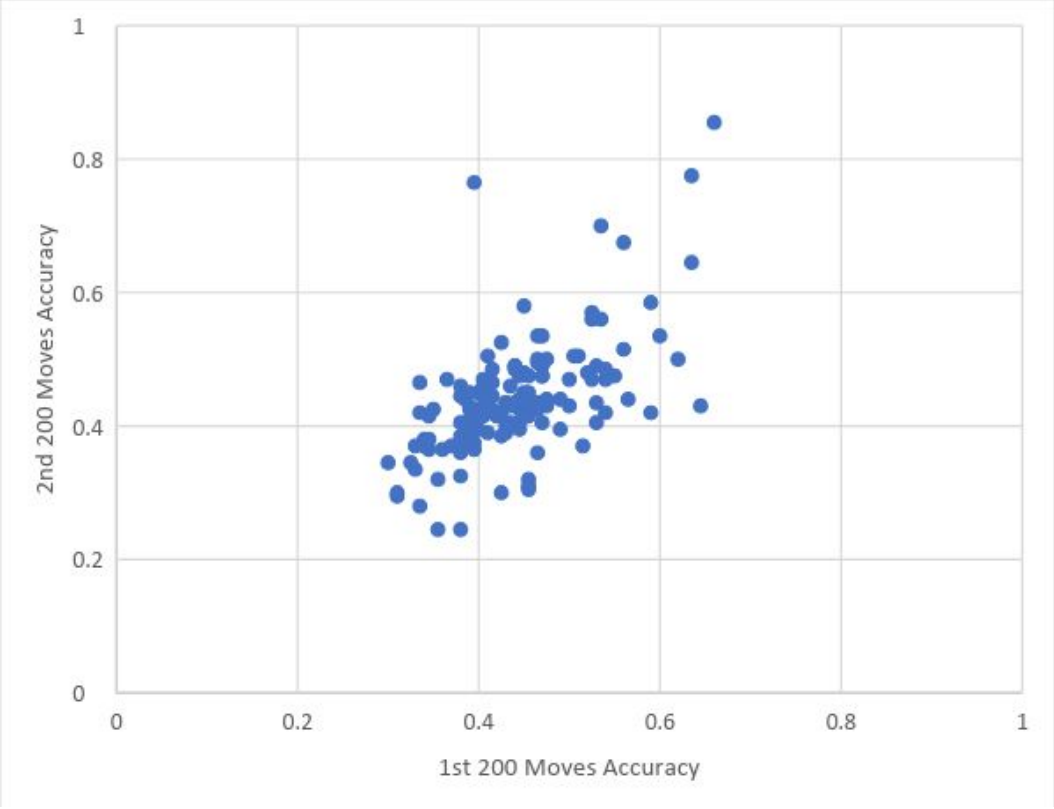

Figure 8. Comparing 1st 200 moves accuracy to 2nd 200 moves accuracy.

Discussion

When testing on random sequences (produced by Python pseudo-random modules), the RNNs' performances are no better than random guessing. On the other hand, biases existed in the sequences produced from humans that the RNNs were able to capitalize on. The networks are able to identify each participants' particular (and individual) tendencies, and guess correctly of 44.1% of the time, 1.32 times higher than the expected 33.3%.

While the 1st order sequential effect was investigated as found from previous papers with simpler tests for randomness production (Yu et al., 2017), this was not found here as participants favoured repeating key strokes for middle tiles. This could be due to fatigue, or due to the previously found tendency to prefer middle tiles (and to therefore move in such a way as to stay in middle tiles). Intuitively, it would make sense to prefer middle tiles if the goal was to stay as unpredictable as possible. A user who spent more time in middle tiles would have more situations where they would have 4 options, thus allowing for more options to choose a direction and allowing for a lower accuracy of prediction. Indeed, participants who spent more time in middle tiles had lower total accuracies.

Generally, participants who spent less time on their moves saw lower accuracies. This suggests that participants who took longer (and potentially spent more time thinking about each move) were less random in their decision-making. Yu et al. (2017) saw a similar finding when asking participants to choose a random number between 1 and 10, as participants disproportionately chose 7 when taking time to think about a random number, as it "felt" more random.

Tests in reliability were also favourable, as splitting the tests of each participant in two did not show systematically different scores. Furthermore, these scores correlated with an $r = 0.6068$, suggesting that using training two identically parameterized RNNs on two different sequences would provide a reliable comparison of the sequences in terms of predictability.

The results of this study suggest that neural networks can be used to judge sequences for predictability, and as a result, many future studies could build on this assumption. For example, participants could be split into control and test conditions, where the test condition could be configured such that the classifier would make predictions live and provide participants with feedback as to whether or not their movements were being guessed correctly.

Would participants who were being provided with feedback do better or worse on the task than participants who were not provided with feedback? These questions could be answered using techniques studied in the current paper.

Another particular interest would be correlating peoples' ability to produce random sequence with IQ test scores. Participants could be provided with a Post-Experiment IQ test. Would participants with better IQ scores be better or worse at the randomness task? How would a strong working memory interact with peoples' ability to act randomly?

It is worth noting a major limitation of this method. A participant with malicious intent who knows where the cut-off is between train and test data may choose to act with one strategy for the training portion and use another strategy for the testing portion in order to see a lower accuracy. This could be alleviated by splitting a user's data into multiple train and test sets of varying lengths in order to reduce such a participant's success with manipulating the network.

While many questions have been uncovered, the goal of this paper was to answer one question: Can we use neural networks to compare sequences for relative predictability? Sequences produced by humans were consistently more predictable for neural networks compared to sequences produced at random. In fact, sequences produced by random Python libraries could not be predicted on better than chance. While humans have many biased preconceptions of randomness, neural networks appear to be immune to these biases.

References

Bar-Hillel, M., & Wagenaar, W. A. (1991). The perception of randomness. Advances in applied mathematics, 12(4), 428-454.

CAcert (2018, Mar 29). Random Number Results. Retrieved from http://www.cacert.at/cgi-bin/rngresults

Chang, A. X. M., Martini, B., & Culurciello, E. (2015). Recurrent neural networks hardware implementation on FPGA. arXiv preprint arXiv:1511.05552.

Graves, A. (2013). Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850.

Yu, R. Q., Gunn, J., Osherson, D., & Zhao, J. (2017). The consistency of the subjective concept of randomness. The Quarterly Journal of Experimental Psychology, (just-accepted), 1-31.