International Journal of Internet, Broadcasting and Communication Vol.14 No.3 143-148 (2022)
http://dx.doi.org/10.7236/IJIBC.2022.14.3.143

IJIBC 22-3-17

Discernment of Android User Interaction Data Distribution Using Deep Learning

Jun-Won Ho

Associate Professor, Department of Information Security, Seoul Women's University, Korea
jwho@swu.ac.kr

Abstract

In this paper, we employ deep neural network (DNN) to discern Android user interaction data distribution from artificial data distribution. We utilize real Android user interaction trace dataset collected from [1] to evaluate our DNN design. In particular, we use sequential model with 4 dense hidden layers and 1 dense output layer in TensorFlow and Keras. We also deploy sigmoid activation function for a dense output layer with 1 neuron and ReLU activation function for each dense hidden layer with 32 neurons. Our evaluation shows that our DNN design fulfills high test accuracy of at least 0.9955 and low test loss of at most 0.0116 in all cases of artificial data distributions.

Keywords: *Deep Neural Networks, Android, User Interaction Data*

1. Introduction

Deep neural network (DNN) can be harnessed for diverse classification or prediction problems. We deploy DNN for classifying data distribution as Android user interaction data distribution or artificial data distribution. Android user interaction data distribution indicates the distribution of human user interaction trace data in Android collected from [1], whereas artificial data distribution represents the distribution of the number of random events that can be used in Monkey tool, in which users can setup the number of random events for Android applications. In our DNN design, we use TensorFlow and Keras, we adopt sequential model with 4 dense hidden layers and 1 dense output layer. We employ ReLU activation function for each dense hidden layer with 32 neurons and sigmoid activation function for a dense output layer with 1 neuron. Moreover, we utilize he_uniform initializer and set dropout probability to 0.01. We set 80% training cases and 20% test cases out of entire input data set. We also use binary cross-entropy function as loss function and employ Adam optimizer. We configure learning rate to 0.001. Our evaluation results exhibit that our DNN design attains high test accuracy of at least 0.9955 and low test loss of at most 0.0116 in all cases of artificial data distributions.

2. Related Work

Many researchers have devised diverse tools that create inputs automatically for applications running on

Manuscript Received: June. 17, 2022 / Revised: June. 20, 2022 / Accepted: June. 24, 2022
Corresponding Author: jwho@swu.ac.kr
Tel: +82-2-970-5607, Fax: +82-2-970-5981
Associate Professor, Department of Information Security, Seoul Women's University, South Korea

Android systems [3,5,6]. Android systems furnish Monkey tool that produces application inputs automatically [2]. In [4], deep learning with human dataset in [1] is utilized for automatically making inputs.

3. Discernment of Android User Interaction Data Distribution Using DNN

For our discernment of Android user interaction data distribution using DNN, we put to use Rico dataset, which is Android human dataset collected from [1]. More particularly, we utilize interaction trace dataset in Rico dataset as follows. We first extract a pair of N_u and N_c for each Android application in Rico interaction trace dataset, where N_u is total number of User Interfaces (UIs) and N_c is total number of X and Y coordinates, respectively. We then compute a fraction $F_i = \frac{N_c}{N_u}$ for each Android application A_i $(1 \leq i \leq N_a)$, where N_a is the total number of Android applications in Rico interaction trace dataset. Finally, we set pairs of F_i and D_i as Android user interaction data distribution for Android application A_i, where D_i indicates the frequency of F_i in N_a fractions over N_a.

We assume that Monkey tool [2] is used for providing artificial events to Android applications. Note that the number of artificial events for Android applications can be arbitrarily configured in Monkey tool. Hence, we set the distribution of artificial events as artificial data distribution. In particular, we utilize normal data distribution, uniform data distribution, and constant data distribution for artificial event distribution. Note that the total number of data in normal data distribution, uniform data distribution, and constant data distribution is equal to N_a, which is the total number of Android applications in Rico interaction trace dataset.

In artificial normal data distribution, we set mean and standard deviation as 8.0 and 2.0, respectively. In artificial uniform data distribution, we use the range of [1, 10]. In artificial constant data distribution, we make use of a fixed constant value of 5.

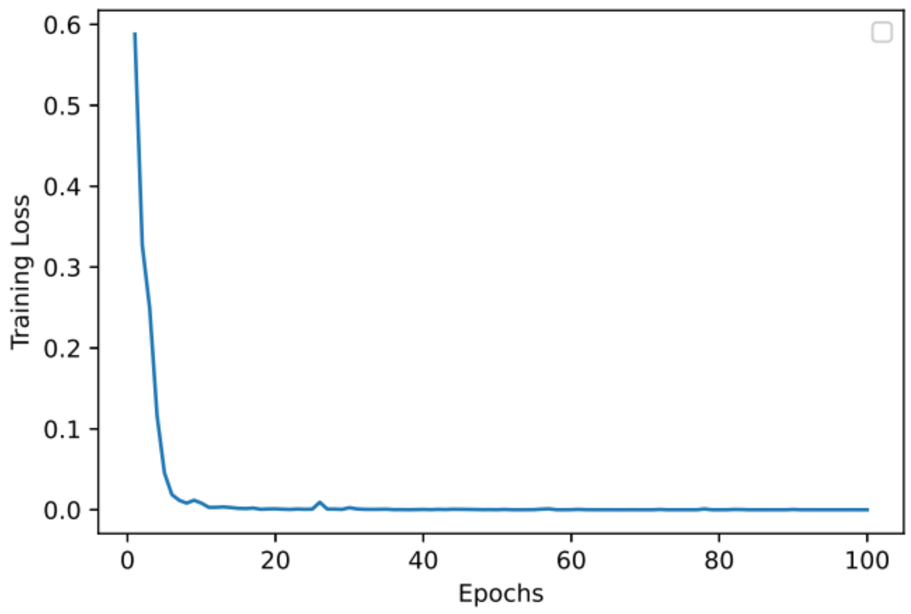

Figure 1. Variation of training loss over epochs when artificial data distribution is uniform distribution.

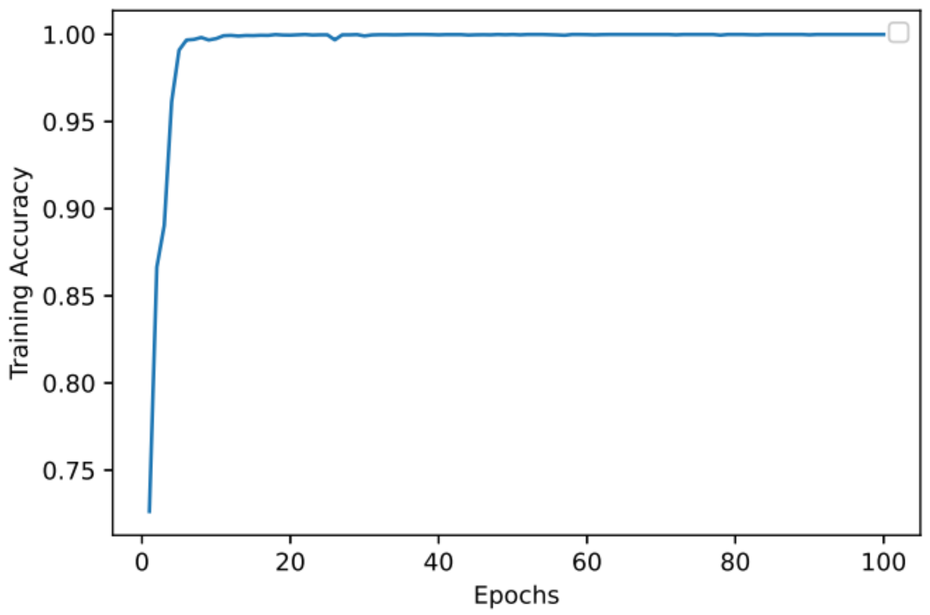

Figure 2. Variation of training accuracy over epochs when artificial data distribution is uniform distribution.

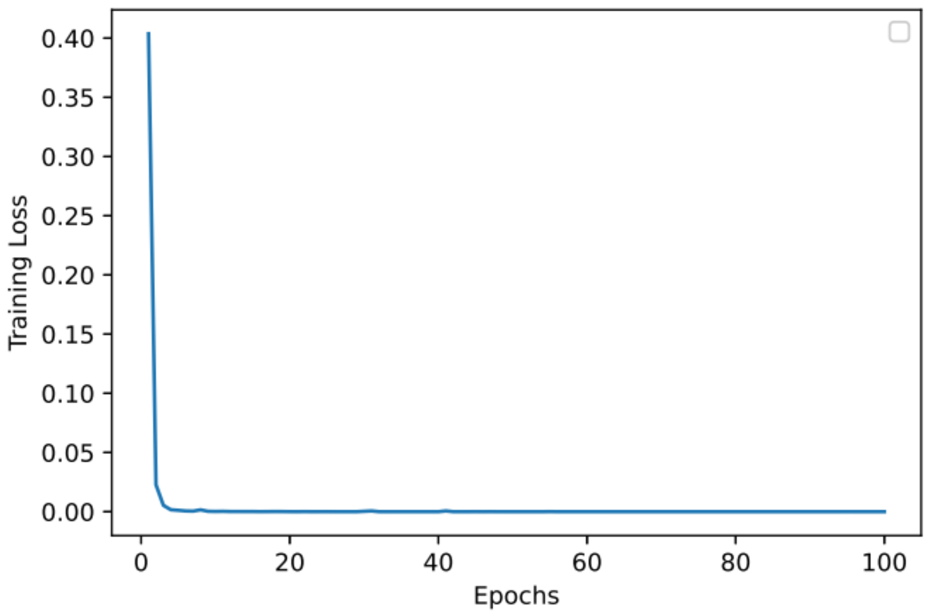

Figure 3. Variation of training loss over epochs when artificial data distribution is constant distribution.

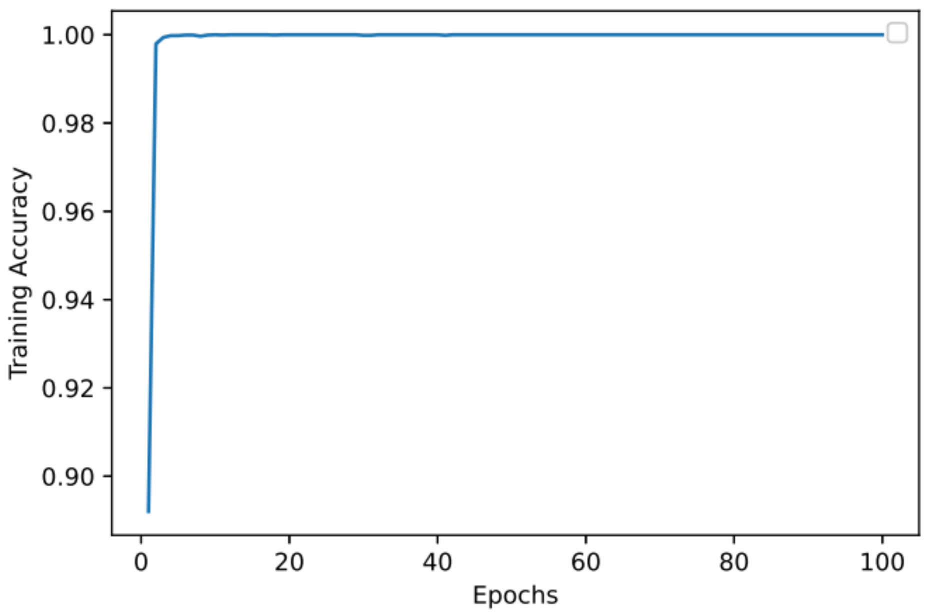

Figure 4. Variation of training accuracy over epochs when artificial data distribution is constant distribution.

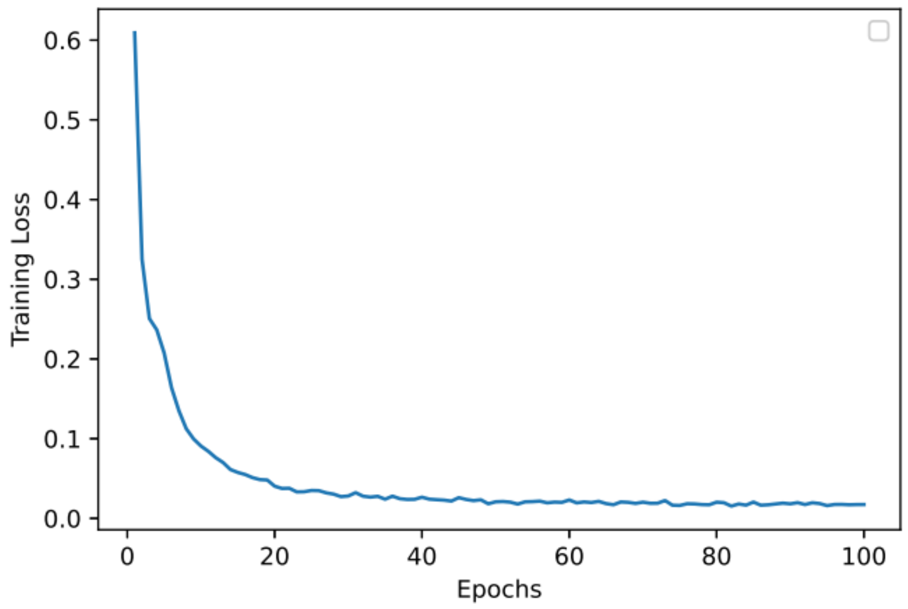

Figure 5. Variation of training loss over epochs when artificial data distribution is normal distribution.

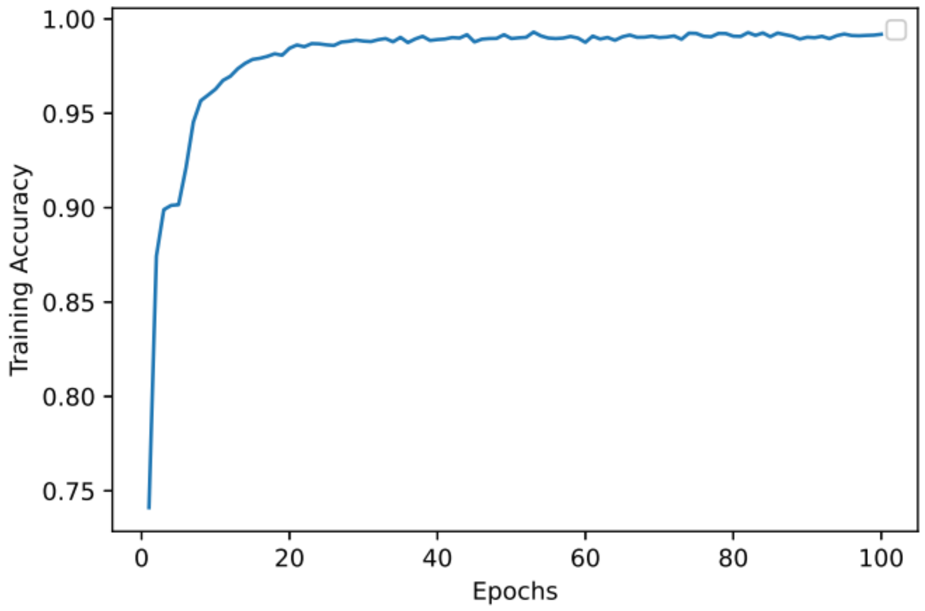

Figure 6. Variation of training accuracy over epochs when artificial data distribution is normal distribution.

As shown in Figures 1, 3, and 5, we observe that training loss tends to more quickly come close to 0 as epoch increases in cases of artificial uniform and constant data distribution than in case of artificial normal data distribution. As shown in Figures 2, 4, and 6, we notice that training accuracy tends to more slowly come close to 1 as epoch increases in case of artificial normal data distribution than in cases of artificial uniform and constant data distribution.

As shown in Table 1, we perceive that test accuracy achieves 1.0 when artificial data distributions are uniform and constant data distributions. We also see that test accuracy in artificial normal data distribution fulfills 0.9955. Finally, we notice that test loss is measured as at most 0.0116 in all cases of artificial data distributions. This means that our discernment of Android user interaction data distribution using DNN is robust.

Table 1. Test loss and test accuracy in artificial data distributions.

Artificial data distribution	Test Loss	Test Accuracy
Uniform data distribution	1.9729e-08	1.0
Constant data distribution	1.8902e-09	1.0
Normal data distribution	0.0116	0.9955

4. Conclusion

In this paper, we explore how Android user interaction data distribution is distinct to artificial data distribution by using DNN and real user interaction trace data in Android. Our evaluation results demonstrate that our DNN design discriminates between Android user interaction data distribution and artificial data distribution with test accuracy of at least 0.9955 and test loss of at most 0.0116 in all cases of artificial data distributions.

Acknowledgement

This work utilizes Android user interaction trace dataset collected from [1] and it is approved by IRB in Seoul Women’s University. This work was supported by a research grant from Seoul Women’s University (2022-0137). This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1F1A1057019).

References

[1] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols and Ranjitha Kumar. 2017. **Rico: A Mobile App Dataset for Building Data-Driven Design Applications**. In Proceedings of the 30th *Annual Symposium on User Interface Software and Technology (UIST '17).* (https://interactionmining.org/rico) DOI: https://doi.org/10.1145/3126594.3126651

[2] https://developer.android.com/studio/test/monkey.

[3] Y. Li, Z. Yang, Y. Guo, and X. Chen. DroidBot: A Lightweight UI-Guided Test Input Generator for Android. In IEEE/ACM 39th IEEE International Conference on Software Engineering Companion, 2017. DOI: https://doi.org/ 10.1109/ICSE-C.2017.8.

[4] Y. Li, Z. Yang, Y. Guo, and X. Chen. Humanoid: A Deep Learning-Based Approach to Automated Black-box Android App Testing. In 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2019, pp. 1070-1073, DOI: https://doi.org/ 10.1109/ASE.2019.00104.

[5] S. Hao, B. Liu, S. Nathy, W.G.J. Halfond, R. Govindan. PUMA: Programmable UI-Automation for Large-Scale Dynamic Analysis of Mobile Apps. In ACM MobiSys, 2014. DOI: https://doi.org/10.1145/2594368.2594390

[6] H. Zheng, D. Li, B. Liang, X. Zeng, W. Zheng, Y. Deng, W. Lam, W. Yang, T. Xie. Automated test input generation for android: towards getting there in an industrial case. In 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), pp. 253-262, 2017. DOI: https://doi.org/ 10.1109/ICSE-SEIP.2017.32