Journal of Multimedia Information System VOL. 10, NO. 1, March 2023 (pp. 45-52): ISSN 2383-7632 (Online)
https://doi.org/10.33851/JMIS.2023.10.1.45

Multi-Class Classification Prediction Model for Password Strength Based on Deep Learning

Seok Jun Kim[1], Byung Mun Lee[1*]

Abstract

Various indexes are being used today to evaluate the strength of passwords. In these indexes, the strength of a password is evaluated to be high if it takes longer for an attacker to predict it. Therefore, using such an evaluation, there is a problem that a leaked password may reduce the reliability of the index by increasing vulnerability if an attacker attempts to attack using a leaked password. Hence, estimating the leaked frequency when considering strength is important for reducing vulnerability. This paper proposes a password strength evaluation model using deep learning-based multi-class classification, which solves the existing problem of leaked frequency not being considered during evaluation. Data preprocessing modeling is critical to improve the performance of this model. Additionally, since selecting and extracting feature values of preprocessing data is also important, a model that accurately estimates the degree of leakage through an evaluation method of existing indexes is proposed. To evaluate the performance of the proposed model, an experiment that compares the password leaked frequency stored in a database using a password list was conducted. As a result, the proposed model correctly evaluated 99% of the 345 leaked passwords. Therefore, the effectiveness of the proposed model was verified.

Key Words: Artificial Intelligence, Deep Learning, Password Strength, Password Strength Prediction.

I. INTRODUCTION

When a user enters a password to subscribe to a service, the service typically displays the password's strength based on an evaluation index. Password strength can be evaluated by analyzing the characters that make up the password or by analyzing meaningful patterns. Common methods for analyzing the characters include "luds" or "NIST entropy" [1]. These methods evaluate password strength based on factors such as the number of characters in the password or the variety of character types used. For example, "a1b 2c3d4" is considered stronger than "12346" because it includes more characters and character types. On the other hand, "zxcvbn" analyzes the pattern of the password to determine its strength [2-3]. For instance, the strength of "bear123" is considered weak because it includes the word "bear," which is a commonly used word.

Current indexes evaluate password strength based on the time it takes for an attacker to predict the password. However, users often create new passwords using familiar words or combinations of passwords used for other services [4-5]. This means that an attacker may attempt an attack using passwords that have already been leaked instead of predicting passwords randomly. Since such attacks can be successful in a shorter amount of time than random predictions, leaked passwords are highly vulnerable [6-7]. Therefore, there is a need for improvement in the evaluation method that utilizes current indexes.

Research has also proposed new indexes that consider whether passwords have been leaked [8]. However, these indexes evaluate passwords solely based on whether they have been leaked or not, without taking into account the number of times they have been leaked. As a result, passwords are considered to have equal strength, regardless of the number of times they have been leaked. For example, even though "potato654321" has been leaked 50,000 times and "q1!g*3mL" has been leaked only once, both are treated as identical solely because they have been leaked. Therefore, there is a limitation in evaluating passwords that distinguishes them based on the difference in leaked frequency.

There are databases that store the frequency of leaked passwords [9]. These databases can provide a safer service by informing users how many times a password has been leaked if the leaked frequency of a password can be mapped and provided to the user. For example, if a query is made

Manuscript received March 07, 2023; Revised March 19, 2023; Accepted March 25, 2023. (ID No. JMIS-23M-03-009)
Corresponding Author (*): Byung Mun Lee, +82-31-750-4756, bmlee@gachon.ac.kr
[1]Department of Computer Engineering, Gachon University, Seongnam, Korea, bnb1324@naver.com, bmlee@gachon.ac.kr

for the password "sandwich," it can be determined that it has been leaked 16,084 times, thus making it possible to prevent the user from using a vulnerable password in advance.

However, it can be difficult for databases to always maintain the latest state of leaked data. As a result, a different solution method is needed, as there may be passwords that have been leaked but are not yet known. Therefore, a new solution is proposed using deep learning technology [10-12]. This solution utilizes the characteristic of deep learning technology that allows for the learning of the relationship between two values if the input and output data are determined. Once training is completed, the results can be predicted using only the input values.

This paper proposes a Multi-class Classification Prediction Model for password strength based on deep learning that considers leaked frequency to evaluate password strength and solves the problem of degraded evaluation reliability of existing indexes when a password is leaked. By proposing a new model, we contribute to improving the evaluating method of the password. Therefore, proposed model can be used to strengthen the security of service which uses password to manage user account.

To improve the model's accuracy, it is crucial to extract effective feature values for classification through the selection and preprocessing of original data. Feature values are extracted by utilizing the evaluation method for existing indexes. Additionally, data from a database that stores leaked frequency is also extracted. The strength is evaluated using a solution that classifies the leaked frequency of the predicted password through a model trained by building a deep learning network based on the obtained feature values.

Section 2 of this paper outlines the process for creating the proposed Multi-class Classification Prediction Model for password strength based on deep learning. Section 3 details an experiment conducted to confirm the performance of the trained model obtained from the preceding process, and the results of the experiment are analyzed. Finally, Section 4 summarizes the conclusion.

II. RESEARCH METHOD

In this paper, the research will be conducted in four sequential processes to create the model. As illustrated in Fig. 1, the four processes are the Collecting process, Preprocessing process, Training process, and Evaluating process.

During the Collecting process, data is collected from external sources to be used as original data. During the Preprocessing process, feature values and label data are extracted from the collected original data and stored in a database. During the Training process, the stored training data is divided into training, validation, and test data sets. The model is then trained using the training and validation data sets. Finally, during the Evaluating process, the predicted results of the trained model are verified using the test data set and the model's performance is evaluated based on this verification.

2.1. Data Collecting

This study uses passwords and leakage frequency data as original data for training the artificial intelligence model. The process of collecting this original data from an external database is illustrated in Fig. 2.

The first step in collecting data involves collecting passwords using the "rockyou" text file provided by "Common Password Lists (rockyou.txt)" within Kaggle [13-15]. The text file contains a total of 14,341,564 unique passwords

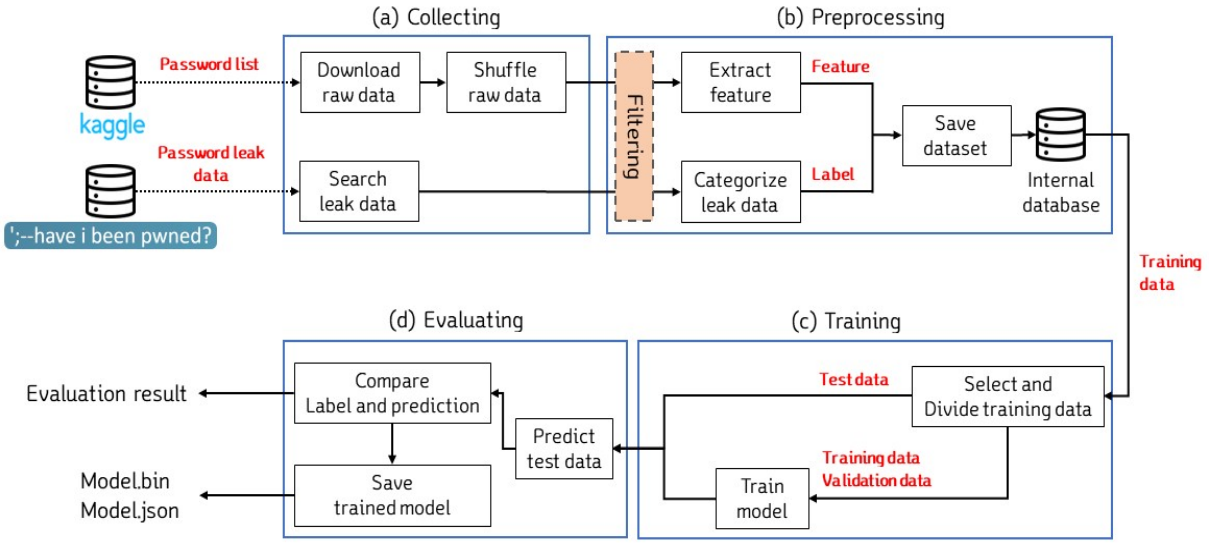


Fig. 1. The overall process of proposed model.

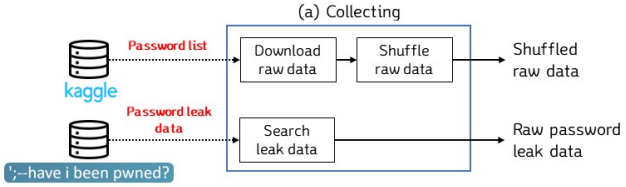


Fig. 2. The process of proposed model: collecting.

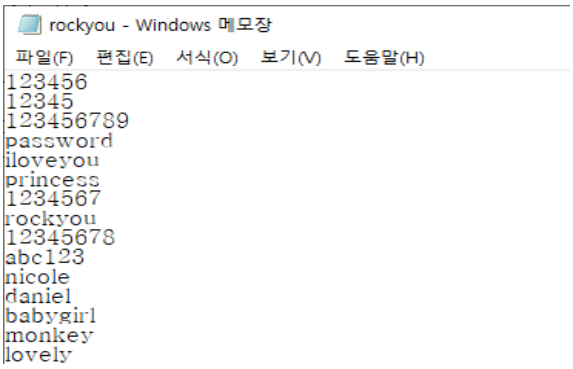


Fig. 3. Example of rockyou text file.

composed of combinations of uppercase and lowercase English alphabet characters, numbers, and special characters. These passwords have already been leaked and are open-source data distributed for research purposes, rather than being randomly generated. They are stored in cataloged character string form, as shown in Fig. 3.

Additionally, these passwords have widely spread scores of strength, as shown in Table 1. Therefore, they have to be filtered in the preprocessing process.

The text file collected in this manner is organized alphabetically. Therefore, the passwords in the file are randomly sorted using the Fisher-Yates shuffle algorithm before use [16].

Leaked frequency data stores the frequency at which passwords have been leaked. However, the leaked frequency of a password is unknown unless it has already been collected. Therefore, leaked frequency must be collected through an external database that stores this information. For this purpose, "Have I been Pwned" is used, which provides the leaked frequency of passwords through its own database [9]. If a user enters their password, the website provides the corresponding leaked frequency. However, the entire content of the original leaked frequency data on the website, from which the leaked frequency can be verified, is not publicly accessible to prevent abuse. Therefore, in this paper, the leaked frequency is collected by entering the password.

Table 1. The statistics of rockyou text file.

Name	Minimum	Maximum	Average
Luds	0	100	30.5
Zxcvbn	0	253.1	7.08
Length	1	285	8.75

2.2. Data Preprocessing

The original data collected during the data collection stage cannot be used as-is for training artificial intelligence. Therefore, it must be converted into data that can be used for training.

Accordingly, it is necessary to extract training data from the original data during the preprocessing stage, as illustrated in Fig. 4.

During the preprocessing process, passwords containing characters that are not used in Korea, such as Cyrillic characters, are first removed. After completing the filtering process, feature values are extracted from the remaining passwords. The feature values extracted include "luds," "zxcvbn," and "levenshtein distance.".

The "luds" feature analyzes the characters that make up passwords. However, there is no standard for "luds," and each service has a different evaluation criterion. Therefore, a specific criterion must be selected to extract a feature value [17]. In this study, "The Password Meter" is used as the criterion [18,19]. This method evaluates the strength of a password by dividing it into elements for which points are added or deducted, and then combining the scores. This strength value is used as the feature value. For example, the password "zm12l@q!" is evaluated to be 70 points, as shown in Fig. 5.

In the case of the password "zm12l@q!", it is assigned additional points for six out of the seven items for which additional points can be given. Among these items, the

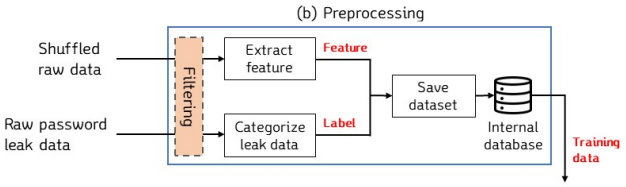


Fig. 4. The process of proposed model: preprocessing.

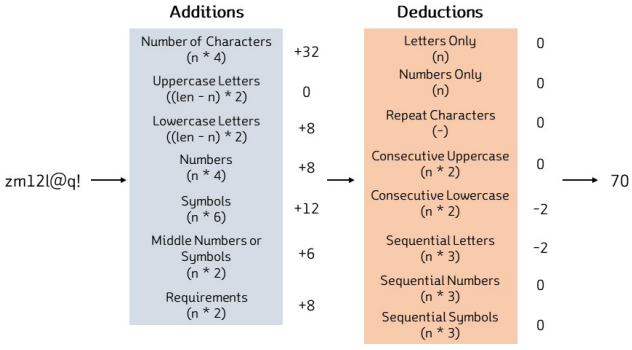


Fig. 5. The process of password evaluation using luds.

"number of characters in the password" item assigns 32 additional points since there are a total of eight characters in the password and four points are given for each character. For the deduction items, points are deducted for two items. For the "continuous lower-case character" item, two points are deducted since "m" follows "z" at the start of the password. By adding up all the points to add and deduct, the strength assigned to the password is 70 points.

"zxcvbn" analyzes passwords for meaningful patterns and uses different methods for each type of pattern [2]. The feature value extracted from this analysis is the necessary prediction frequency required to guess a password. These frequencies are calculated using common logarithms. For instance, a password with a feature value of 3 requires about 1,000 attempts because the required prediction frequency is 103, not just 3. The calculation process for the "zxcvbn" feature value involves three stages: Match, Estimate, and Search, as shown in Fig. 6.

First, during the Match stage, the password is divided into the smallest meaningful unit, and each unit is converted into its original form. For example, in the password "P@ssword", the special character "@" can be converted to the alphabet "a" or "A". This is because "P@ssword" does not have any meaning, whereas "Password" is an English word that means password. After all divisions are completed, a category is assigned to each unit.

During the Estimate stage, the necessary prediction frequency is calculated by taking into account the category for each unit that was divided in the Match stage. For example, while "P@ssword" itself may be a modified form of the word password, it can also be seen as a combination of the two words "pass" and "word." Therefore, "zxcvbn" calculates for all possible situations.

During the Search stage, the divided units are combined to form the original password and the prediction attempts with the lowest value are selected. "zxcvbn" evaluates that it is easier to predict "P@ssword" as a whole compared to separate words like "P@ss" and "word". Therefore, it is evaluated that less than about 100.95 attempts are needed to predict the password according to this evaluation. This value is then extracted as the feature value.

The “levenshtein distance" analyzes the similarity between a password and the word dictionary used in "zxcvbn", and this value is extracted as a feature value. The similarity value is zero if the password and the word match. For example, the password "tomato123" has a similarity of 3 compared to the word "tomato". This is because three more characters have to be added compared to "tomato". However, to obtain a similarity value, multiple words need to be compared with the password. Therefore, the highest similarity value among the values obtained through the comparisons is used as a feature value.

the feature values are extracted through different methods, the range of values for each feature also varies. The names and corresponding value ranges for each feature are shown in Table 2.

The "ludsScore" is a feature value that utilizes the strength score extracted from the luds category, which ranges from a minimum of 0 to a maximum of 100. The "zxcvbnScore" is another feature value belonging to the zxcvbn category, which utilizes the prediction frequency of the password as the score. The score starts at a minimum of zero and has no maximum limit. The "levenshteinScore" is a feature value belonging to the levenshtein distance category, which utilizes the highest similarity obtained by comparing the password with words. The score starts at zero

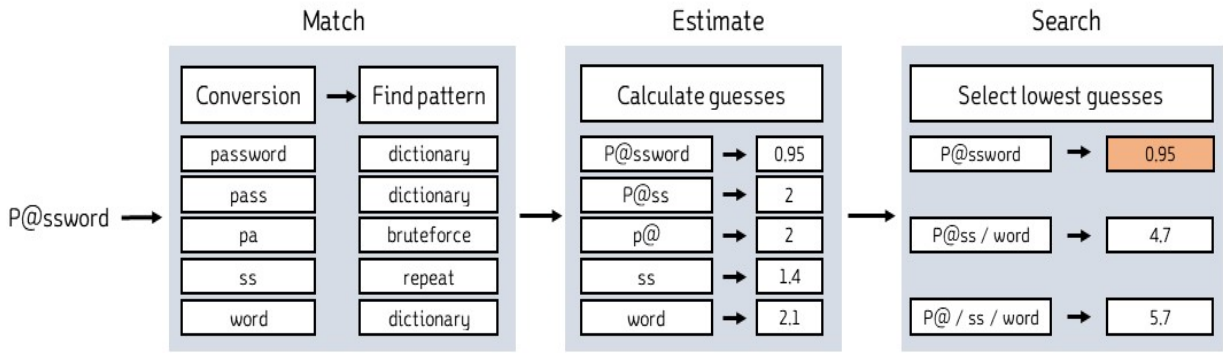


Fig. 6. The process of password evaluation using zxcvbn.

Table 2. Description of feature data (n=feature score).

Feature	Name	Range of score	Description
luds	ludsScore	0≤n≤100	Estimated luds score
zxcvbn	zxcvbnScore	0.00≤n	Estimated guesses needed to crack password in zxcvbn
levenshtein distance	levenshteinScore	0≤n≤len	Highest similarity between word and password

Table 3. Label data (n=amount of leak).

Label	Range	Description
0	$n \geq 101$	The higher the range, the better password strength
1	$51 \leq n < 101$	
2	$26 \leq n < 50$	
3	$10 \leq n < 26$	
4	$n < 10$	

Table 4. The example of train data.

ludsScore	zxcvbnScore	levenshteinScore	Label
[[8,	3.05	2,	0],
[70,	9.2,	8,	3],
[42,	8.1,	8,	2],
[100,	14.9,	13,	4],
[22,	5.5,	3,	1],
	⋮		
[16,	5.39,	5,	1]]

when the password and the word match exactly, and the maximum value is the length of the password itself if it is not similar to any word.

Once the feature values are extracted, the label data is obtained using the leaked frequency data from the original data. The label data is divided into five sections based on the leaked frequency, as shown in Table 3, and each section is assigned a label value.

For example, if the leaked frequency of the password "a1mdlalsm" is 0, a label value of four is assigned. Conversely, for the password "password1234" with a leaked frequency of 36,522, a label value of 0 is assigned since the leaked frequency is 101 or greater. Therefore, a password with a label value closer to four can be considered to have excellent security since the leaked frequency is smaller. The data after the feature value and label data extraction process is completed is designated as the training data, which is stored in a database. An example of the stored training data is shown in Table 4, with the "ludsScore", "zxcvbnScore", "levenshteinScore" feature values, followed by the actual feature values, and "Label" assigned to each password based on its leaked frequency.

2.3. Model Training

During the Model Training process, the training data that was stored during the Preprocessing process is selected based on the feature values and label values of the data. The selected training data is then divided into three types: training data, validation data, and test data, as shown in Fig. 7.

During the selection of training data, outliers are removed to ensure smooth model training. However, the number of outliers may vary depending on the timing of leaked frequency data collection, which is obtained from an external database for label data. This is because accurate leaked frequency information may not be reflected in the data. Therefore, data for which the strength of the feature values and the label values are not proportional are regarded as outliers. If these values do not meet the selection criteria shown in Table 5, they are considered outliers and removed from the training data.

For example, if there is data with a "ludsScore" value of 1, a "zxcvbnScore" value of 30, a "levenshteinScore" value of 12, and a label of 3 assigned, it is removed because this data does not meet the selection criteria. Conversely, data with a "ludsScore" of 90, a "zxcvbnScore" of 10.1, a "levenshteinScore" of 11, and a label of 4 is kept since it corresponds to the selection criteria. The training data for which the selection has been completed is then divided into three sets: 70% for training data, 20% for validation data, and 10% for test data. The number of data points for each set is shown in Table 6.

The training data is used to train the model, while the validation data checks for overfitting or underfitting during the training process. These data are then input into the predesigned model, which consists of an input layer, an output layer, and one hidden layer. As only one label needs to be predicted among many, a multi-classification model is utilized, as shown in Fig. 8.

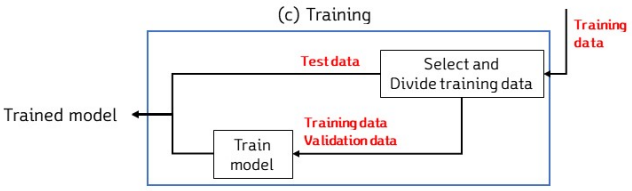


Fig. 7. The process of proposed model: training.

Table 5. Feature selection standard (n=feature score).

Label	ludsScore	zxcvbnScore	levenshteinScore	Total
0	$0 \leq n \leq 10$	$0 \leq n \leq 4$	$0 \leq n \leq 3$	282
1	$12 < n \leq 30$	$4 < n \leq 6$	$4 \leq n \leq 6$	1,545
2	$40 < n \leq 60$	$6 < n \leq 8$	$6 < n \leq 8$	451
3	$60 < n \leq 80$	$8 < n \leq 10$	$8 < n \leq 10$	214
4	$n > 80$	$n > 10$	$n > 10$	950

Table 6. The amount of divided train data.

Label	Train data	Validation data	Test data	Total
0	208	46	28	282
1	1,093	297	155	1,545
2	330	76	45	451
3	158	34	22	214
4	688	167	95	950
Total	2,477	620	345	3,442

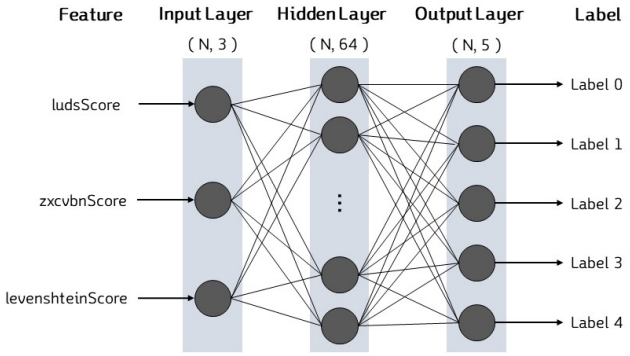


Fig. 8. The structure of proposed model.

The input layer receives three pieces of data and transmits them to the hidden layer. The data are the "ludsScore", "zxcvbnScore", and "levenshteinScore" feature values, respectively. In the hidden layer, the rules between the input values and the label values are trained. The activation function of the hidden layer is "ReLU". The output layer outputs the prediction result of the model based on the input values. Since the output layer must predict the five leaked categories, it consists of a total of five nodes. And the "Softmax" activation function, which outputs the probability for belonging to each category, is used. For the loss function of the model, "Categorical Crossentropy" is used. This function compares the predicted probability that is output from "Softmax" with the actual category. For the optimization function, "adam" is used. For the training, 16 and 40 are used as the batch size and epoch, respectively. The training is conducted through the model constructed like this. And when the training is completed, the Evaluating process of the trained model is started.

2.4. Model Evaluating

During the model evaluation process, the performance of the trained model is assessed and analyzed using the test data that was stored during the training process, as shown in Fig. 9. Once the evaluation is complete, the trained model is saved as a file on the server.

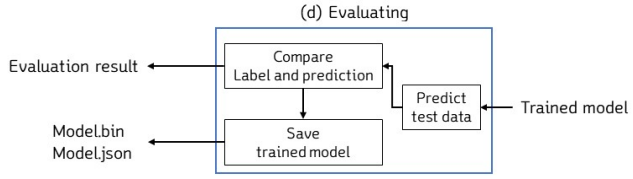


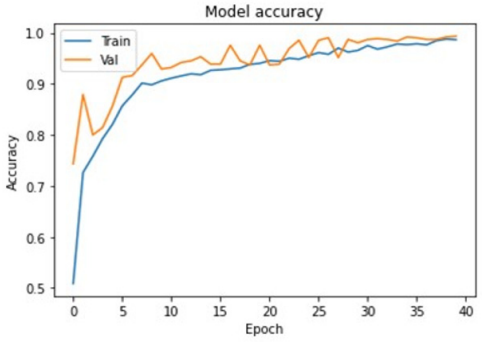
Fig. 9. The process of proposed model: evaluating.

To evaluate the performance of the trained model, the predicted values are first checked by inputting the test data. Since this data was not used to train the model, it verifies the performance of the trained model for new data. After the predicted values are outputted, the degree of match is checked by comparing them with the label values. The accuracy is calculated as the ratio of correct predictions based on the input values. Once the accuracy is calculated, the weight of the model is stored in "model.bin" and the structure of the model is stored in "model.json" inside the server.

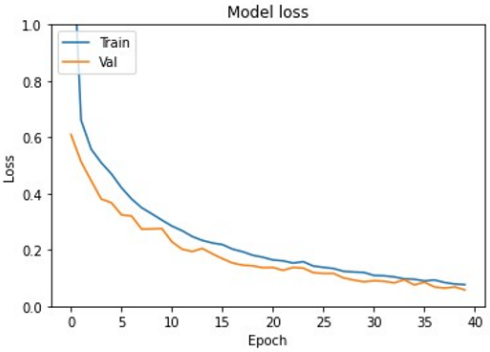
III. RESEARCH RESULT

An experiment was conducted to compare the predicted value and the label value of the trained model using the test data. The performance of the model was evaluated by checking the accuracy through the comparison result of the two values. Before the comparison, the accuracy and loss of the training data and validation data were verified during the training process of the model, as shown in Fig. 10.

For the proposed model, an accuracy of 0.98 and a loss of 0.07 were verified for the training data in the last epoch (epoch 40). For the validation data, an accuracy of 0.99 and



(a) Accuracy



(b) Loss

Fig. 10. Model training result.

Table 7. The prediction result of proposed model.

Label	Correct	Incorrect	Total
0	28	0	28
1	154	1	155
2	44	1	45
3	22	0	22
4	95	0	95
Total	343	2	345

a loss of 0.05 were verified. It was possible to confirm that the training was conducted stably for both types of data as the epoch repeated. The evaluation result of the trained model using the test data is shown in Table 7.

The first row of the table shows the actual label values of the evaluated passwords obtained from an external database. If the predicted result of the model matched the label value of the test data, it was classified as Correct. On the other hand, it was classified as Incorrect if the model failed to predict the label value of the test data accurately. For example, the label value of two was classified as Incorrect in Label 2 of Table 6 because the actual label value was two, while the model predicted the corresponding data's label to be three. During the evaluation with the test data, the proposed model correctly classified 343 out of 345 leaked passwords like this. Thus, the accuracy of the trained model for the test data is approximately 99.4%. Therefore, the effectiveness of the proposed model was confirmed.

Previously, we found research considering whether passwords have been leaked [8]. And the evaluated strength by the research has only one of two labels: strong or weak. Therefore, passwords which evaluated normal strength cannot be easily distinguished by their method. On the other hand, Proposed model has five labels. From label 0 to 4, the model has more label to distinguish the strength. Additionally, the proposed model has 99.4% of accuracy during test data classification. While research has 95.7% which is slightly less than the proposed model, proposed model is more effective than the research.

IV. CONCLUSION

Nowadays, password strength evaluation is utilized in many services. However, the existing evaluation model's accuracy can be questionable when the password has been leaked or is suspected of being leaked. To address this issue, this paper proposes a new method that extracts feature values from original password data and collects leaked frequency information from an external database. The leaked frequency is used to classify passwords into five labels. Additionally, a new Multi-class Classification Prediction Model is proposed that considers leaked frequency while predicting label values through the deep learning method using feature values. Finally, an experiment was conducted to verify the accuracy of the proposed model, and the results show that the trained model is highly effective. Thus, the proposed method offers an improved solution for password strength evaluation.

The existing evaluation models only consider either the composition or pattern of a password, making it difficult to meaningfully evaluate new types of passwords. However, the proposed model considers various feature values that account for both composition and pattern, enabling meaningful evaluations for new passwords.

In the future, the evaluation performance of the model can be improved by adjusting the ratio between the labels of the training data or conducting an overlapping filtering process of the original data Additionally, all password evaluation models, including the proposed one, do not account for attacks such as forgery or falsification of passwords stored in service servers [20]. These attacks fundamentally compromise the reliability of password evaluations, regardless of the accuracy of the evaluation model. Therefore, users should periodically check their passwords and refer to evaluation results to ensure their passwords are secure.

ACKNOWLEDGMENT

This work was supported by the Technology Innovation and Development Project (Grants No. RS-2022-001663321) funded by the Ministry of SMEs and Startups (MSS, Korea).

REFERENCES

[1] W. E. Burr, D. F. Dodson, R. A. Perlner, W. T. Polk, S. Gupta, and E. A. Nabbus, "NIST special publication 800-63-2 electronic authentication guideline," *National Institute of Standards and Technology*, Aug. 2013.

[2] D. L. Wheeler, "zxcvbn: Low-budget password strength estimation," in *USENIX Security Symposium*, Texas, Aug. 2016, pp. 157-173.

[3] K. H. Hong, U. G. Kang, and B. M. Lee, "Enhanced evaluation model of security strength for passwords using integrated Korean and English password dictionaries," *Security and Communication Networks*, vol. 2021, p. 13, Sep. 2021.

[4] R. Shay, S. Komanduri, P. G. Kelley, P. G. Leon, M. L. Mazurek, and L. Bauer, et al., "Encountering stronger password requirements: User attitudes and behaviors," in *Proceedings of the Sixth Symposium on Usable Privacy and Security*, NewYork, NY, Jul. 2010, pp. 1-20.

[5] S. M. T. Haque, M. Wright, and S. Scielzo, "A study of user password strategy for multiple accounts," in *Proceedings of the Third ACM Conference on Data and Ap-*

plication Security and Privacy, NewYork, NY, Feb. 2013, pp. 173-176.
[6] L. Bošnjak, J. Sreš, and B. Brumen, "Brute-force and dictionary attack on hashed real-world passwords," in *Proceedings of International Convention on Information and Communication Technology*, Opatija, Jul. 2018, pp. 1161-1166.
[7] J. H. Jeong, Y. W. Cha, and C. H. Kim, "A study on the variable and dynamic salt according to access log and password," *Journal of Korea Multimedia Society*, vol. 24, no. 1, pp. 58-66, Jan. 2021.
[8] K. H. Hong and B. M. Lee, "A deep learning-based password security evaluation model," *Applied Sciences*, vol. 12, no. 5, Feb. 2022.
[9] Have I Been Pwned, 2021, https://haveibeenpwned.com/Passwords.
[10] S. K. Kim, "Affective computing among individuals in deep learning," *Journal of Multimedia Information System*, vol. 7, no. 2, pp. 115-124, Jun. 2020.
[11] J. Y. Kim, Y. L. Shin, and E. J. Choi, "An intrusion detection model based on a convolutional neural network," *Journal of Multimedia Information System*, vol. 6, no. 4, pp. 165-172, Dec. 2019.
[12] P. Liu, T. Lei, Q. Xiang, Z. Wang, and J. Wang, "Animal fur recognition algorithm based on feature fusion network," *Journal of Multimedia Information System*, vol. 9, no. 1, pp. 1-10, Mar. 2022.
[13] Common Password List (rockyou.txt), 2019, https://www.kaggle.com/datasets/wjburns/common-password-list-r ockyoutxt.
[14] M. Kaleel and N. A. LeKhac, "Towards a new deep learning based approach for the password prediction," in *Proceedings of International Conference on Trust, Security and Privacy in Computing and Communications*, Guangzhou, Feb. 2021. pp. 1146-1150.
[15] E. İ. Tatlı, "Cracking more password hashes with patterns," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 8, pp. 1656-1665, Aug. 2015.
[16] P. E. Black, "Fisher-Yates shuffle, " May. 2019, https://xlinux.nist.gov/dads/HTML/fisherYatesShuffle.html
[17] K. H. Hong and B. M. Lee, "Electrooculography filtering model based on machine learning," *Journal of Korea Multimedia Society*, vol. 24, no. 2, pp. 274-284, Feb. 2021.
[18] The Password Meter, 2010, http://www.password-meter.com/.
[19] Y. Yang, K. C. Yeo, S. Azam, A. Karim, R. Ahammad, and R. Mahmud, "Empirical study of password strength meter design," in *Proceedings of International Conference on Communication and Electronics Systems (ICCES)*, Coimbatore, Jul. 2020, pp. 436-442.
[20] K. H. Hong and B. M. Lee, "An access code key for verification service model on the blockchain in a door security," *Journal of Korea Multimedia Society*, vol. 25, no. 10, pp. 1416-1432, Oct. 2022.

AUTHORS

Seok Jun Kim is an undergraduate student majoring in Computer Engineering. Currently, he is working on the project about Password Evaluation Model using AI. His research interests include blockchain, AIoT, etc.

Byung Mun Lee received a B.S. degree in 1988 from Dongguk University, Seoul, Korea and a M.S. degree from Sogang University and a Ph.D. degree from University of Incheon Korea, in 1990 and 2007. He had worked for LG Electronics for 7 years. He is currently a professor in the department of Computer Engineering, Gachon University, South Korea. He had been at California State University Sacramento, USA from 2013 to 2014 as a visiting scholar. His research interests are IoT for healthcare, AIoT Smart Service, network protocols, blockchain, smart services, etc.