Article

BBPE-AE: A Byte Pair Encoding-Based Auto Encoder for Password Guessing

Samane Ghafari *, Leila Safari , Mohsen Afsharchi

Posted Date: 11 September 2024

doi: 10.20944/preprints202409.0834.v1

Keywords: Password Guessing; Cybersecurity; Password Strength Assessment; Autoencoder Network; Long Short-Term Memory (LSTM) Networks; Byte-level Byte Pair Encoding

Article

BBPE-AE: A Byte Pair Encoding-Based Auto Encoder for Password Guessing

Samane Ghafari Sabil *, Leila Safari and Mohsen Afsharchi

Department of Electrical and Computer Engineering, University of Zanjan, Zanjan, Iran
* Correspondence: samane_ghafari@znu.ac.ir

Abstract: In today's rapidly evolving digital landscape, the significance of password guessing techniques in both offensive and defensive strategies is paramount. Passwords serve as a crucial line of defense for both individuals and corporations, safeguarding their sensitive systems and data. Therefore, assessing the effectiveness of these access credentials is a critical task. However, existing research in this field often encounters limitations, such as a lack of sufficient training data and extended model training times. Current methods often struggle with limited training data and lengthy training times. This paper introduces BBPE-AE, a novel Auto Encoder Network (AE) designed for password guessing. BBPE-AE utilizes Byte-level Byte Pair Encoding (BBPE) to extract frequent tokens from password datasets without length restrictions, employing a dynamic window technique to capture complex patterns. Experimental results on Hotmail and Myspace datasets demonstrate exceptional performance, achieving high similarity rates (BLEU-Unigram: 0.90, BLEU-Bigram: 0.82 for Hotmail; BLEU-Unigram: 0.90, BLEU-Bigram: 0.81 for Myspace). BBPE-AE generates realistic passwords that meet HIBP (Have I Been Pwned) standards, minimizing duplication. These findings highlight the effectiveness of BBPE-AE in enhancing security by generating realistic passwords, ultimately safeguarding sensitive systems and data.

Keywords: password guessing; cybersecurity; password strength assessment; autoencoder network; long short-term memory (LSTM) networks; byte-level byte pair encoding

1. Introduction

Password guessing techniques hold a pivotal role in both offensive and defensive strategies within the ever-expanding realm of digital security. These techniques equip individuals and organizations with the means to fortify their systems against unauthorized access attempts. Given the prevalent reliance on passwords for safeguarding sensitive data, it becomes imperative to assess the strength and resilience of these access credentials. Malicious actors employ various methods to gain unauthorized access, including brute force attacks, dictionary attacks, hybrid attacks, and rainbow table attacks. By gaining a comprehensive understanding of password guessing techniques, individuals and organizations can proactively mitigate the risk of unauthorized access and potential security breaches. This knowledge enables them to effectively detect and address potential vulnerabilities, thereby enhancing system security. By staying informed about the latest password guessing techniques and adopting appropriate countermeasures, individuals and organizations can significantly bolster the protection of their valuable digital assets.

Textual passwords are still one of the most common authentication methods [1] due to its affordability, easy accessibility, simple implementation, and reusability [2,3]. However, users often struggle to remember complex passwords, leading them to choose simplistic and commonly used strings. Additionally, users tend to reuse passwords across multiple services. Consequently, real-world passwords are easily predictable by attackers, rendering authentication systems vulnerable to guessing attacks [4,5]. This vulnerability makes passwords the weakest link in the entire security chain. As a result, significant efforts have been dedicated to strengthening passwords against guessing attacks.

 doi:10.20944/preprints202409.0834.v1

Initially, comprehensive policies were implemented to restrict users from selecting weak passwords. These policies enforced limitations on password length and character composition. For example, requiring a minimum of 8 characters with a combination of uppercase and lowercase letters, numerals, and special symbols. However, research has shown that these regulations have proven ineffective in bolstering password strength in practical scenarios [6]. Users can easily bypass these restrictions by employing various techniques, such as incorporating common patterns (e.g., capitalizing the first character), including personal information, or using identical or similar passwords across different services.

Conversely, implementing overly stringent restrictions results in passwords that are exceedingly difficult to remember, forcing users to store them in insecure locations, thus creating additional security concerns [7]. Therefore, it is crucial to accurately simulate password guessing to protect passwords from actual guessing attempts and ensure their security. Research in this domain aims to identify existing security flaws, eliminate vulnerabilities, and enhance the overall adaptability of the system. By simulating real-world password guessing scenarios with precision, researchers can uncover weaknesses in current security measures, develop robust countermeasures, and improve the overall resilience of password-based authentication systems.

The primary goal of our study is to leverage a deep learning-based methodology to improve the effectiveness of password guessing. To achieve this objective, we have implemented various techniques and approaches. Firstly, we have adopted a character-based approach for data pre-processing and preparation, specifically focusing on input data comprising leaked passwords. This pre-processing step ensures that the data is appropriately formatted and ready for further analysis. In addition to the character-based approach, we have incorporated the Byte-level Byte Pair Encoding (BBPE) technique. This technique enables us to segment the input data into meaningful units, enhancing the model's ability to capture intricate patterns and generate more accurate passwords. By utilizing BBPE, we can effectively represent the input data in a manner that optimizes the performance of our proposed model.

To facilitate the learning process and ensure accurate reconstruction of the original data, we have employed an Autoencoder (AE) architecture. The AE architecture enables the model to learn a meaningful representation of the input data, capturing its essential characteristics and features. By leveraging the power of AE, we can enhance the quality of generated passwords and improve the overall performance of our system. Furthermore, we have incorporated the attention mechanism into our model. The attention mechanism allows the model to focus on relevant parts of the input data, assigning varying degrees of importance to different elements. This attention mechanism enhances the model's performance and accuracy in the task of password guessing, as it can effectively prioritize the essential components of the input data and generate passwords with greater precision.

By combining these approaches and techniques, our study aims to develop a robust and effective password guessing system. The utilization of a character-based approach, BBPE segmentation, AE architecture, and attention mechanism collectively contribute to enhancing the outcomes of password guessing, resulting in more reliable and secure passwords.

2. Related Works

This research focuses on the critical area of password guessing, which involves deciphering users' passwords and simulating the actions of an attacker to assess password strength in real-world contexts. While password generation and password security are important aspects of this domain, this review delves deeper into the methods used for password guessing. To provide a comprehensive overview, we will examine three distinct approaches: rule-based, machine learning-based, and deep learning-based methods. In the following sections, we will conduct a thorough analysis of each approach. However, before delving into these analyses, we will first provide a brief review of two critical aspects of this domain: password generation and password security.

Beyond password guessing, research has also focused on developing robust password generators. For instance, [8] explores the use of deep learning techniques, particularly Long Short-Term Memory (LSTM) networks, to create a high-performing password validator. This research

 doi:10.20944/preprints202409.0834.v1

demonstrates the potential of deep learning to improve the security of password generation. Another recent study, [9], introduces two novel password generation models, PassGPT and PassVQT, leveraging the capabilities of large language models (LLMs). PassGPT which is based on the GPT-2 architecture, excels in guided password generation, while PassVQT employs a more complex transformer-based approach. These studies highlight the growing role of LLMs in password generation and their potential to create more secure and user-friendly password systems.

Research in password security focuses on developing methods to assess password strength and mitigate security risks. [10] proposes a novel machine learning approach to estimate the strength of Lithuanian user passwords, addressing the limitations of traditional English-only dictionary-based methods. This study highlights the potential of machine learning to improve password security assessments in diverse language contexts. Additionally, [11] investigates the use of the RoBERTa algorithm for password strength prediction, achieving impressive accuracy rates. This research demonstrates the effectiveness of advanced machine learning techniques in enhancing password security. Finally, [12] addresses the limitations of existing password datasets and proposes a novel method for generating multipurpose password datasets that can be used in various research areas. This approach aims to provide researchers with a more comprehensive and contemporary resource for password-related research, enabling advancements in password security and analysis.

In the subsequent sections, we will delve into password guessing techniques, exploring their underlying principles, strengths, limitations, and applications. This detailed analysis will provide a comprehensive understanding of password guessing techniques, aiding researchers and practitioners in selecting the most appropriate approach for their specific requirements.

2.1. Rule-Based Approaches

The rule-based methodology involves formulating specific rules or patterns that are commonly used in password creation. These rules are designed to emulate the behavior and preferences of users when constructing passwords. Examples of rule-based techniques include dictionary attacks, brute force attacks, and hybrid attacks. These methods rely on predefined rules and patterns to systematically guess passwords and assess their vulnerability.

The rule-based approach stands as the predominant technique for password guessing, wherein tools are utilized to expand pre-existing dictionaries of leaked passwords through the application of production rules, rather than exhaustively trying all possible combinations (as seen in brute force attacks, also referred to as dictionary-based attacks). However, the utilization of rules to augment the password lexicon presents certain limitations [13]. Firstly, the creation of rules necessitates expert knowledge in the field. Secondly, these methods only generate finite lists of passwords, thereby imposing constraints on the range and diversity of potential password guesses.

In secure systems, password hashes are stored in the database instead of plaintext. To guess hashed passwords, two commonly employed software tools are HashCat [14] and John the Ripper [15]. These tools offer the capability to efficiently verify billions of passwords against password hashes [13]. However, it is crucial to recognize that if the password being decrypted is complex and exceeds the limitations of the predefined rules within these tools, they may not be able to successfully decrypt the password.

2.2. Machine Learning-Based Approaches

The machine learning-based approach leverages the power of machine learning algorithms to learn patterns and characteristics from training data. These algorithms are trained on large datasets containing known passwords and their corresponding features. By analyzing the training data, the machine learning models can identify patterns and relationships that aid in password guessing. These models can then be used to predict the strength of new passwords based on their learned knowledge.

Recent studies have highlighted the efficacy of machine learning approaches in password prediction. In the work presented in [16], they employed the TarGuess $-$ I^{+} model, which incorporates commonly used passwords and unique character sequences found in users' passwords. Another noteworthy contribution in this field is the introduction of a novel approach for partitioning

 doi:10.20944/preprints202409.0834.v1

passwords into fragments, as discussed in [17]. This approach presented three distinct models for password prediction at the chunk-level, utilizing Markov, PCFG, and neural-network-based FLA (Fast, Lean and Accurate) techniques, all of which yielded promising results. However, the widely used Markov model in this domain suffers from the issue of multiple duplicate passwords, which hampers its effectiveness and coverage. To address this limitation, a solution proposed in [18] suggests integrating the Markov model with a dynamic distribution approach. By adopting this approach, the probability distribution of passwords can be flexibly modified, encouraging a tendency towards a more uniform distribution during the password creation process. Experimental results have demonstrated that the incorporation of this dynamic mechanism in the Markov model leads to a decrease in the duplicate rate and an increase in the cover rate.

Further advancements in machine learning-based password prediction techniques have explored innovative ways to leverage semantic information and diverse model combinations. [19]introduces a novel word extraction approach for passwords and presents an improved PCFG model, called WordPCFG. WordPCFG demonstrates significant performance gains, highlighting the effectiveness of word-based analysis in password cracking. [20] provides a comprehensive evaluation of semantic password grammars, focusing on the impact of sample size, probabilistic smoothing, and linguistic information on password cracking. The study demonstrates that PCFGs remain competitive models compared to more recent neural network approaches, offering valuable insights into the effectiveness of semantic password grammars. [21] presents WordMarkov, a novel password probability model that incorporates semantic information based on Markov Chains. WordMarkov achieves significant improvement in attacking "long" passwords, highlighting the crucial role of semantic parts in cracking longer passwords. [22] introduces RFGuess, a novel random-forest based framework for password guessing, marking the first application of classic machine learning techniques in this domain. RFGuess proposes three password guessing scenarios: trawling guessing, targeted guessing based on Personally Identifiable Information (PII), and targeted guessing based on users' password reuse behaviors. The study demonstrates RFGuess's effectiveness in various guessing scenarios, particularly in targeted guessing based on PII. [23] explores the innovative GuessFuse approach, inspired by multi-view learning, for enhancing password guessing effectiveness. By treating guess lists generated by different password guessing models as multiple views of the data, GuessFuse leverages the diversity of these views to capture a wider range of password features, leading to improved guessing accuracy.

2.3. Deep Learning-Based Approaches

The deep learning-based methodology takes advantage of deep neural networks, which are capable of automatically learning complex patterns and representations from large amounts of data. Deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), can be trained on extensive password datasets to capture intricate features and dependencies. These models excel at recognizing and generalizing patterns, enabling them to make accurate predictions and assessments in password guessing tasks. The field of password guessing strategies has witnessed significant advancements, particularly with the introduction of various deep learning approaches.

The initial approach, described in [3] and further improved in [24], utilized the framework of Generative Adversarial Networks (GAN), surpassing its predecessor [25]. The base model incorporated the ResNet architecture, and subsequently, the inclusion of DenseNet led to the development of two models, namely DenseGAN1 and DenseGAN2 [26]. An alternative method was proposed in [27], which employed the base model [3] along with a training dataset generated using back translation techniques. However, this model exhibited unsatisfactory performance when applied to the specific dataset. Subsequently, two additional enhancements to the original model [3] were introduced: GS-PassGAN and S-PassGAN [13]. These enhanced models outperformed the base model in terms of overall performance. Furthermore, an enhancement known as GNPassGAN was proposed in [28], building upon the base model [3]. GNPassGAN introduced changes to the loss function, incorporated gradient normalization in the discriminator, and streamlined the

 doi:10.20944/preprints202409.0834.v1

generator architecture by modifying the activation function of the last layer. These modifications effectively reduced the occurrence of duplicate passwords. These advancements in deep learning-based password guessing strategies have demonstrated considerable improvements and have paved the way for more effective and efficient password guessing models. The utilization of GANs, along with the integration of architectures like DenseNet, has led to the development of enhanced models such as DenseGAN1, DenseGAN2, GS-PassGAN, S-PassGAN, and GNPassGAN. These models have collectively contributed to mitigating the limitations of earlier approaches and have yielded notable advancements in the field.

The study presented in [29] introduced an innovative methodology that integrates a GAN framework with a controller network to gain insights into the differences between probability distributions. By incorporating a controller, this model enabled the generator to effectively understand the actual probability distribution of passwords and approximate it to a uniform distribution. As a result, the number of iterations required was significantly reduced, leading to improved efficiency. In a related advancement, GENPass [30] introduced a novel deep learning model known as multi-source deep learning. This model consists of multiple generators and a classifier. By leveraging the power of multiple generators, GENPass enhanced the overall performance and accuracy of password guessing. Furthermore, GENPass-pro [30] represents an enhanced version of GENPass that incorporates probability-based techniques to further enhance the model's capabilities. These developments in combining GAN frameworks with controller networks and utilizing multi-source deep learning techniques have demonstrated notable improvements in the field of password guessing. The integration of a controller network has facilitated a better understanding of probability distributions, leading to more efficient and accurate password generation. Meanwhile, the utilization of multiple generators and probability-based techniques has enhanced the performance and effectiveness of deep learning models like GENPass and its enhanced version, GENPass-pro.

In addition to the aforementioned advancements, the REDPACK approach, introduced in [31], aims to enhance password decryption by combining multiple candidate password generation models, thereby reducing the number of attempts required. Unlike conventional GANs that rely on a generator to decrypt passwords, REDPACK takes a different approach by utilizing a discriminator to select a password that appears more authentic and likely to be correct. Moreover, [32] leverages the power of CNNs to identify patterns and inconsistencies in password structures, allowing for more accurate identification of missing characters and improving the overall decryption process. Furthermore, [33] introduces the concept of knowledge distillation in the context of password guessing. It treats password guessing as a language modeling task and employs a transformer-based model as the teacher. This teacher model is trained using a bidirectional masked language model, guiding the learning process of the student model and enhancing its performance. The experimental results demonstrate the effectiveness of this method in improving password guessing accuracy and efficiency. These advancements, including the REDPACK approach, the use of CNN for missing character detection, and the application of knowledge distillation with transformer-based models, have contributed to significant progress in the field of password guessing. These techniques have demonstrated improved decryption capabilities and enhanced performance, showcasing their potential in practical applications and further advancement of password security research.

Moreover, [34] introduces, a novel Chinese syllables and Neural Network-based (CSNN) password generation method specifically designed for Chinese password sets. CSNN demonstrates superior performance compared to traditional methods like PCFG and Markov Chain models, highlighting the potential of deep learning for enhancing password security in specific language contexts. [35] addresses the limitations of existing GAN-based password guessing models by proposing RLPassGAN, a novel method based on SeqGAN with policy gradients. RLPassGAN achieves significant improvements in terms of sample quality and cracking rates, demonstrating its effectiveness in generating high-quality password guesses. Finally, [36] introduces Universal Neural-Cracking-Machines (UNCM), the first self-configurable password model that adapts to target password distributions at inference time using auxiliary data. UNCM leverages deep learning to

 doi:10.20944/preprints202409.0834.v1

identify correlations between auxiliary data and corresponding passwords, creating a tailored password model for specific systems. This research establishes a new state-of-the-art for offline password guessing attacks and password strength metering, demonstrating the adaptability and effectiveness of deep learning in password security.

Beyond these advancements, recent research has focused on enhancing password guessing attacks using deep learning techniques. [37] introduces PassBERT, a novel bidirectional transformer-based framework for enhancing password guessing attacks. PassBERT leverages a pre-trained password model and employs three fine-tuning approaches to adapt it to real-world attack scenarios: conditional password guessing (CPG), targeted password guessing (TPG), and adaptive rule-based password guessing (ARPG). The study demonstrates significant improvements in all three attack scenarios, highlighting the effectiveness of bidirectional transformers in password guessing attacks. [38] introduces Arana, an analysis pipeline designed to detect and categorize remote password guessing attacks. Arana effectively identifies and labels suspicious groups of requests, potentially linked to the same attack campaign, demonstrating the effectiveness of clustering approaches in identifying and labeling suspicious groups of requests. [39] introduces PagPassGPT, a novel password guessing model designed to improve the quality of passwords generated in guided pattern guessing and minimize the creation of duplicate passwords. PagPassGPT leverages a pre-trained transformer (GPT-2) and incorporates pattern structure information as background knowledge, leading to a significant increase in hit rate and a reduction in duplicate generation. [40] proposes a targeted password guessing algorithm, PASS2EDIT, to model the increasingly damaging credential tweaking attack. PASS2EDIT utilizes a multi-step decision-making training mechanism and a classification neural network to learn the reaction of one-step edit operations to the existing password, demonstrating its effectiveness in this specific attack scenario.

3. Method

This section provides a detailed overview of our proposed method, as illustrated in Figure 1. Our methodology involves several key steps. Initially, we conduct a preliminary dataset pre-processing step, which involves removing duplicate passwords. Subsequently, the dataset is partitioned into two distinct sets: the training set and the test set. The primary focus of the pre-processing step is on the training set, where two sets of features are computed. The first set is based on BBPE, while the second set is derived from characters extracted from the leaked passwords. These feature sets are then used as inputs for the subsequent stages of our model. Following this, we employ the AE architecture to generate novel passwords based on the provided feature sets. These generated passwords are then evaluated by comparing them against the unseen test data.

The following subsections delve into each step in greater detail, illuminating the various processes involved and providing a more comprehensive understanding of our proposed method.

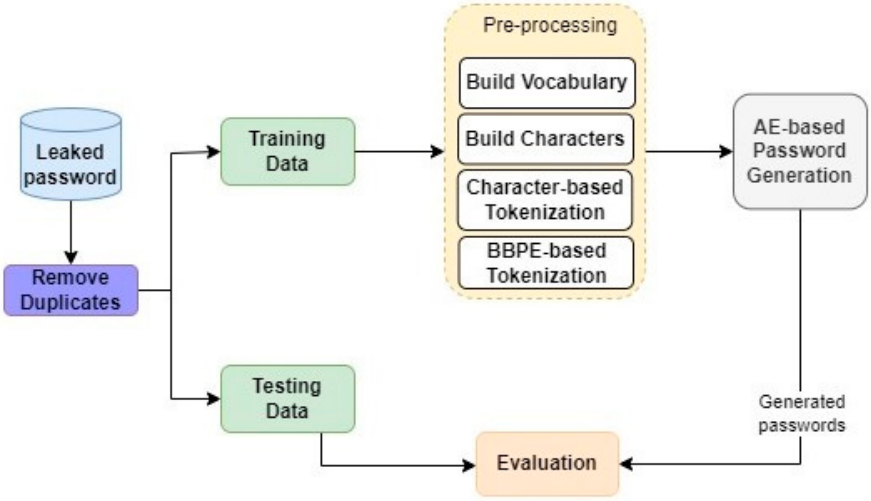

Figure 1. The overall structure of the proposed model.

 doi:10.20944/preprints202409.0834.v1

3.1. *Data Pre-Processing*

The initial step in our data cleaning process involves the removal of duplicate passwords from the dataset. Once this collection of unique passwords is obtained, it is further divided into two distinct sets: the training set, which comprises 80% of the cleaned data, and the testing set, which accounts for the remaining 20%. It is important to note that there are no specific constraints on the maximum length of passwords during this division process.

Moving on to the subsequent phase, we employ two distinct methods for processing the input data: character-based and token-based, also known as BBPE-based. These methods serve as the basis for training two separate models: one utilizing character-based processing, and the other employing BBPE-based processing. By utilizing these two different approaches to process the input data, we can explore and evaluate the effectiveness of each technique in training separate models. This approach allows for comprehensive analysis and comparison of the performance and outcomes of the character-based and BBPE-based models.

The BBPE algorithm, which is a variant of byte pair encoding (BPE), is utilized to handle character sequences that have unique representations at the byte level. This approach finds widespread application in subword encoding for various natural language processing tasks [30]. The BBPE algorithm begins by taking an initial word composed of individual bytes and subsequently combines pairs of frequently occurring bytes to form longer tokens. This process involves iteratively merging the most common pairs of bytes until a predetermined number of combinations is reached. By employing this technique, the BBPE algorithm effectively captures the recurring patterns and structures present in the character sequences, allowing for the creation of meaningful and compact representations.

In our approach, we employed a BBPE-based tokenizer to extract meaningful subword units from the text of passwords. By analyzing patterns across the entire dataset, which consisted of leaked passwords from the training set, the tokenizer identified subword units that represented either individual characters or combinations of characters commonly seen in passwords (byte-level sequences). To determine when to stop the repetition of the BBPE algorithm for each chunk, we introduced a parameter called min_frequency. This parameter served as a criterion, defining the minimum number of times a subword needed to appear in the training data to be included in the vocabulary set. In our experiments, we set this parameter to 10. Consequently, the tokenizer identified and indexed subwords that appeared at least 10 times in the training data. These unique subwords and their corresponding indexes were stored in a Vocabulary file. Based on the assembled vocabulary, our tokenizer generated tokens for each password in the dataset. These tokens were then recorded in the Tokens file, along with their respective indexes. Additionally, we extracted all distinct characters from the training data and stored them in the Characters file, alongside their respective identifiers or indexes. By employing this approach, we were able to effectively tokenize passwords, extracting meaningful subword units and capturing their relationships. The resulting Vocabulary, Tokens, and Characters files serve as valuable resources for subsequent stages of our password generation guessing models.

In order to further enhance the stored tokens, we introduced additional tokens, namely <START>, <END>, <OOV> (Out-of-Vocabulary), and <PADDING>. These tokens, along with their corresponding indexes, were incorporated into the Vocabulary and Characters files. In the Tokens file, the <START> token was appended at the beginning of each encrypted password, while the <END> token was appended at the end. This served to mark the start and end points of each password sequence.

Figure 2 provides a visual representation of the tokenization process, offering clarity and insight into the steps involved in transforming passwords into tokenized sequences. By integrating these additional tokens, we expanded the token vocabulary and enriched the representation of passwords in the tokenized form. This augmentation aids in subsequent stages of analysis, modeling, and generation, providing a more comprehensive and effective approach to password processing.

 doi:10.20944/preprints202409.0834.v1

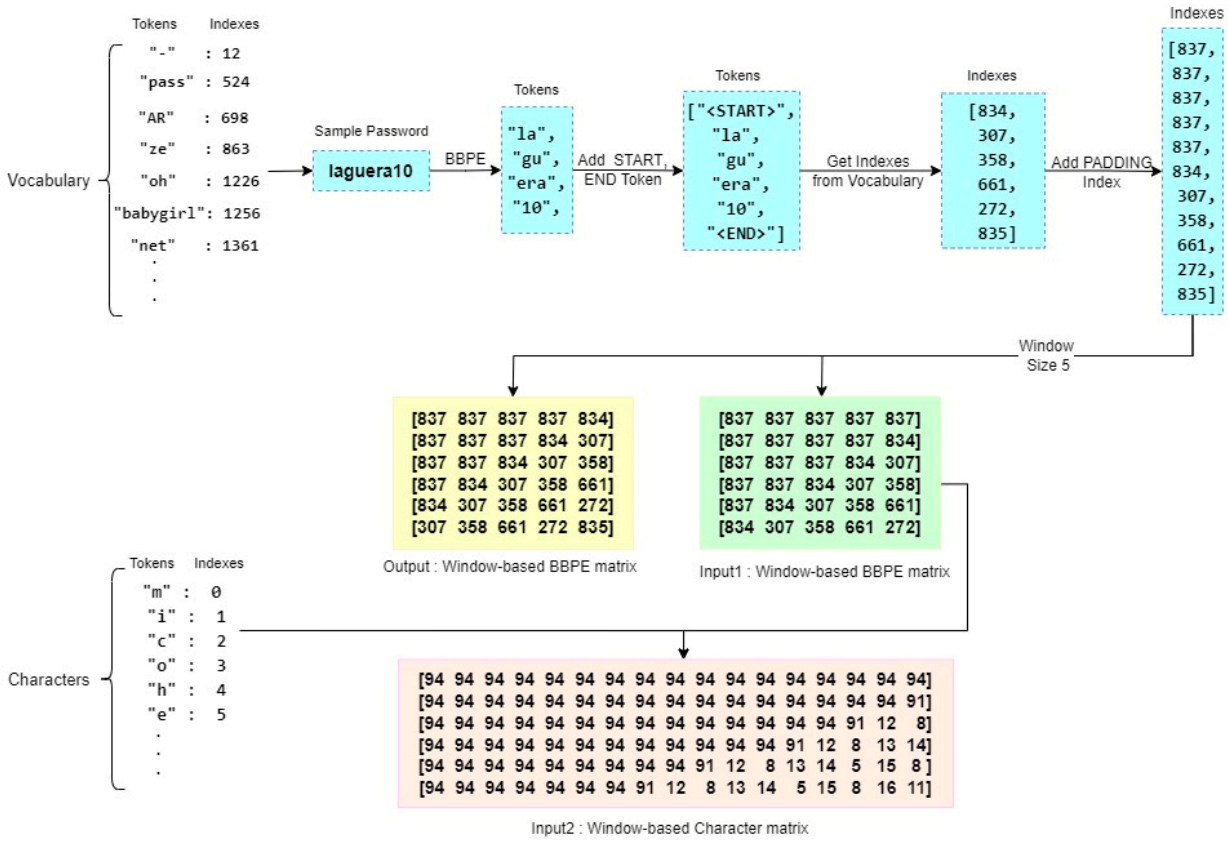

Figure 2. The process of BBPE-based and character-based data preparation.

Following the previous steps, the index data then underwent a windowing process with a window size of 5 and a step size of 1. This windowing process involved traversing the input data and generating three distinct forms of data:

- X1: This represents the token indexes based on the window. It captures the indexes of tokens within each windowed segment of the input data.
- X2: This denotes the character indexes. It records the indexes of individual characters within the windowed segments.
- Y: This represents the target data, where the indexes are shifted one step forward from the X1 indexes. It captures the next token index in the sequence following the windowed segments.

By applying this windowing process, we create these three forms of data, X1, X2, and Y, which serve as inputs for subsequent stages of our model. This methodology facilitates the analysis and modeling of password sequences, enabling the exploration of relationships and patterns within the data.

The Y data, representing the target values, was encoded using the One-Hot encoding technique. One-Hot encoding is commonly utilized to convert categorical data, such as class labels, into binary vectors. It assigns a unique binary representation to each category, with one bit set to one to indicate the presence of the category and the remaining bits set to zero. In addition, we employed the crawl-300d-2M-subword model to encode the pre-processed tokens. This particular model provides word embeddings, which are dense vector representations of words used to capture semantic associations between them. The crawl-300d-2M-subword model is trained on a large corpus of text and can encode not only complete words but also subword units such as word roots or morphemes. This expanded coverage allows for a more precise representation of semantic information, including OOV terms. By following these procedures, we obtained the embedding matrix from the crawl-300d-2M-subword model. This embedding matrix serves as a mapping between the tokens and their corresponding

 doi:10.20944/preprints202409.0834.v1

embeddings. It enables the model to access and utilize the semantic information stored in the pre-trained word embeddings throughout the training process, enhancing its ability to capture and leverage the underlying meaning and context of the tokenized data.

3.2. AE-Based Password Generator

Figure 3 illustrates the architecture of the password generator module, which utilizes an AE framework with LSTM networks and a multi-head attention mechanism to generate passwords. The AE network consists of encoders and decoders that play distinct roles in the password generation process. The encoders are responsible for encoding and compressing the input passwords, effectively reducing their dimensionality and enabling the model to capture essential features and patterns from the input data. On the other hand, the decoders focus on restoring the data to its original representation space, reconstructing it based on the encoded information. During the training phase of the AE model, the objective is to minimize the discrepancy between the reconstructed output and the original input. By doing so, the model learns to extract significant features and patterns from the input data, enhancing its ability to generate passwords that align with the characteristics of the training set.

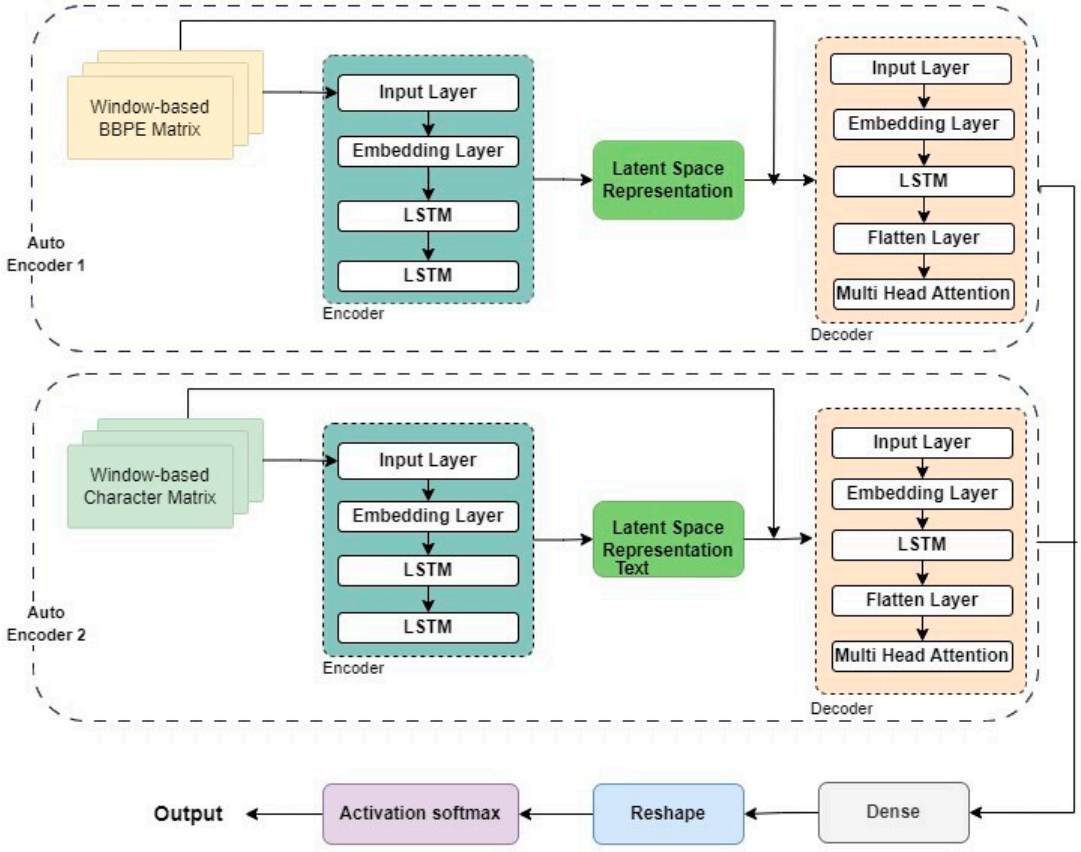

Figure 3. The proposed BBPE-AE network architecture for password generation (the training phase).

The AE network is structured using LSTM models, which are well-suited for representing and capturing sequential and long-term dependencies within. To further enhance the capabilities of the AE network, we incorporate the multi-head attention mechanism. This mechanism allows the model to selectively focus on different segments of the input data, assigning varying levels of importance to each segment. By leveraging this attention mechanism, the model gains the ability to prioritize and emphasize specific elements of the password input, giving greater significance to relevant features and patterns, and enhancing the efficiency and effectiveness of the model for password guessing. By

 doi:10.20944/preprints202409.0834.v1

selectively attending to crucial aspects of the input data, the model can more accurately capture the underlying patterns and generate passwords that align with the learned patterns. This approach enables the model to generate passwords that exhibit desirable characteristics and meet the requirements of password guessing tasks. Overall, the combination of LSTM models and the multi-head attention mechanism empowers the AE network to effectively capture sequential dependencies, assign importance to relevant elements, and leverage learned patterns for efficient password guessing. The details of the model settings is provided in Table 1 (Section 4.2).

3.2.1. Training AE

For training purposes, each token-based and character-based input undergoes training using separate AE networks. These networks share identical structures and parameter values, with the only difference being the input length they handle. As part of the pre-processing stage, we determine the length of the input data to appropriately configure the AE networks. Specifically, we assign a value of 5 to represent the input length for the first AE network, which operates on BBPE-based inputs. Similarly, for the second AE network, we assign the length of the max sequence of the character-based input. This ensures that each AE network is tailored to handle the specific input length requirements. Following the determination of input lengths, we transform the input sequences into tensors. This transformation enables efficient processing and computation within the network, as tensors provide a structured and optimized format for data representation. By converting the input sequences into tensors, we facilitate seamless and effective information flow through the AE networks during both training and inference stages. In summary, by employing separate AE networks for token-based and character-based inputs and configuring them based on the appropriate input lengths, we ensure that each network can effectively capture the distinctive properties of the input data.

In the model, both inputs are encoded using two LSTM layers. In the initial layer, we set the "return_sequences" option to True, allowing the network to generate sequences at each time step. This configuration ensures that the model can capture sequential dependencies and generate outputs accordingly. To obtain the final internal state, which comprises the hidden state and the cell state, for each input in the second layer, we enable the "return_state" parameter and set it to True. These modes are crucial for effectively decoding the model's outputs and maintaining the information flow through the network. Moving on to the decoder implementation, we utilize not only the initial states acquired from the encoders but also the inputs provided to each encoder. By incorporating this strategy and utilizing the information transmitted from the encoders, we enhance the precision and effectiveness of our model.

The decoder consists of an LSTM layer with both the "return_sequences" and "return_state" parameters set to True. This configuration allows the decoder to generate sequences and retain the final internal state for subsequent steps. In order to incorporate the multi-head attention mechanism and consolidate the decoder outputs, we initially flatten the output of the LSTM layer. This flattening process prepares the output for further processing. Subsequently, the flattened outputs are passed through the attention mechanism, enabling the model to selectively focus on different segments of the input based on their significance. This attention mechanism enhances the model's ability to capture important patterns and features during the decoding process. After the attention mechanism, we combine the results obtained for each AE and utilize a dense layer to reduce the dimensions. The dimensionality reduction is achieved by multiplying the window size with the number of signs acquired for the Y data. Finally, we apply the Softmax activation function to obtain the probability distribution over the different classes or categories, allowing the model to generate the final outputs with proper normalization. By following these steps, our model leverages LSTM layers, attention mechanisms, and dimensionality reduction techniques to enhance precision and effectiveness in generating accurate and meaningful outputs. The generated passwords will be evaluated based on provided metrics to reflect the strength of our proposed model in password guessing.

 doi:10.20944/preprints202409.0834.v1

3.2.2. Password Generation Using AE

After the model has been trained, we save the obtained weights to preserve its learned knowledge and enable future use. To generate passwords using the trained model, we rely on several input files from the training data: Vocabulary, Tokens, Indexes, and Characters files. We initiate the password generation process by making a sequence of calls in the following order:

1. Specify the desired number of passwords to be generated.
2. Provide the trained model.
3. Include the Vocabulary file.
4. Include the Characters file.
5. Specify the window length.

During this step, we utilize the inverse dictionary, denoted as "iVocabulary", which allows us to map token indexes back to their corresponding tokens in the Vocabulary file. This mapping is essential for generating coherent and meaningful passwords. Based on the specified number of passwords to generate, the following phases are repeated for each password. We begin by creating a list of size 5, representing the moving window of the event and serving as the initial input for the token-based model. This list is initialized with the "<PADDING>" tag, which signifies an empty or null value. This initialization sets the foundation for the password generation process, ensuring that the model starts with an appropriate context and can generate passwords that align with the learned patterns and structures. By following this sequence of calls and initializing the moving window with the "<PADDING>" tag, we establish the necessary framework for generating passwords using the trained model and associated input files.

In addition to the token-based input, we have a second input in the model that consists of characters. To process this input, we convert the list of tokens into a character-based representation, which is then stored in a new list of the same length as the second longest input. This conversion allows us to effectively handle character-level information within the model. After the conversion, the pre-determined values are transmitted to the model as primary inputs, enabling it to make predictions about the probability of the subsequent token. To ensure consistency and comparability, we standardize the obtained probabilities. The next token is selected from the standardized probabilities using a random selection technique. This randomness adds a degree of variation and unpredictability to the generated passwords, enhancing their security and uniqueness. The selected token is then appended to the list of length 5, which serves as the input for the model in the subsequent iteration. This procedure continues iteratively until the model's prediction reaches the "<END>" token, which serves as a stopping criterion for the password generation process and allows us to generate passwords of variable lengths. By following this iterative process, the model generates passwords by progressively predicting and appending tokens based on their probabilities until an "<END>" token is reached, indicating the completion of a password. This approach ensures that the generated passwords are coherent and consistent with the learned patterns and structures of the training data.

4. Results and Discussion

4.1. Dataset

Multiple password datasets have been publicly released, comprising compromised credentials obtained from specific websites over a period of time. In most cases, passwords are stored in databases as hashed values rather than their original plaintext form. Password guessing techniques typically require access to the plaintext passwords for training purposes, allowing the models to learn patterns and characteristics. With the availability of these datasets, researchers have the option to utilize either the plaintext or hashed forms of passwords. This flexibility enables the exploration of different approaches and techniques for password guessing.

 doi:10.20944/preprints202409.0834.v1

In this paper, we specifically employed the Myspace [42] and Hotmail [43] datasets for training and testing our model. These datasets contain passwords in plaintext format. The Myspace dataset comprises 37,126 passwords, while the Hotmail dataset consists of 8,930 passwords. These passwords exhibit varying lengths, reflecting the diversity and complexity of real-world password choices. By utilizing these datasets, we aim to train and evaluate our model's performance in generating passwords that align with the patterns and characteristics observed in compromised credentials and consequently, improving our proposed model's capability in password guessing.

As mentioned earlier, after performing the necessary pre-processing steps on the Myspace and Hotmail datasets, we divided them into training and testing sets using an 80:20 ratio. This division ensures that a significant portion of the data is used for training the model, while a separate portion remains reserved for evaluating its performance. Further refining our training dataset, we partitioned it into a 75:25 ratio for training and validation purposes. This additional split allows us to monitor the model's performance on unseen data during the training process and make adjustments if necessary. To enhance the representation of the input data and capture the symbols that arise from the pre-processing steps, we incorporated the crawl-300d-2M-subword resource. This resource provides a comprehensive collection of symbols and subwords that are commonly encountered in natural language data. By utilizing this resource, we enrich the vocabulary of our model, enabling it to effectively handle the diverse range of symbols and subwords present in the input data. This incorporation enhances the model's ability to capture and understand the nuances and intricacies of the passwords in the datasets.

4.2. Settings

The accuracy of password learning in the proposed network exhibits improvement as the number of input samples increases. To achieve optimal performance, we carefully selected the training hyperparameters based on extensive testing and evaluation. These hyperparameters are chosen to maximize the effectiveness of the model's training process. To enhance the overall efficiency of the network, we have incorporated the Adam optimizer with a learning rate of 8e-4. This optimizer not only provides computational efficiency but also requires less memory compared to other optimization algorithms. It is particularly well-suited for problems that involve a large amount of data or parameters, making it an ideal choice for our password learning task. Furthermore, we employed the Categorical cross-entropy loss function to compute the loss during training. This loss function is commonly used in classification tasks and aligns well with our objective of generating accurate and meaningful passwords. To facilitate the implementation of our proposed model, we utilized the Keras library. Keras offers a user-friendly and efficient framework for building and training neural networks.

The batch size, which represents the number of training samples used to estimate the error gradient, is an important hyperparameter in the learning process. In our studies, we have set the batch size to 128. This means that during each iteration, 128 samples from the training dataset are used to calculate the error gradient before adjusting the network weights. Epochs, on the other hand, are another crucial hyperparameter that determines the number of iterations the learning algorithm performs on the entire training dataset. In our case, a single epoch indicates that every instance in the training dataset has been utilized to update the network parameters. Specifically, we have conducted 12 epochs in our implementation. For clarity, the specific values of these hyperparameters used in our implementation are summarized in Table 1.

 doi:10.20944/preprints202409.0834.v1

Table 1. The final hyperparameters of the proposed model.

Parameter	Value
Batch Size	128
Epochs	12
Min Frequency	10
Vocabulary Size	30000
Window size	5
Step size	1
Embedding Size	300
Number of hidden units	32
Learning rate	8e-4
Optimizer	Adam

By carefully setting and documenting these hyperparameter values, we ensure a consistent and controlled training process for our neural network. These choices are informed by the established practices and considerations in the field of deep learning, allowing us to effectively train our model and generate accurate and reliable password predictions.

4.3. Evaluation Metrics

In the context of evaluating the strength of generated passwords in current research, the following evaluation criteria have been deployed:

4.3.1. Duplicate Rate

This metric focuses on the frequency of password repetition within a given collection of generated passwords. By analyzing the occurrence frequency, we can evaluate the diversity and distinctiveness of the generated passwords. A lower duplicate rate indicates a higher level of uniqueness among the generated passwords, indicating the model's ability to generate stronger and more robust passwords. To compute this metric, it is necessary to determine the proportion or ratio of repeated passwords in relation to the total number of generated passwords. This can be achieved by comparing each password with others in the collection, counting the number of passwords with more than one occurrence, and subsequently dividing it by the total number of generated passwords. This metric contributes to the overall assessment of the password guessing model, complementing other evaluation criteria and providing a comprehensive understanding of the model's performance in producing robust and unique passwords.

4.3.2. BLEU Score

This criterion is primarily utilized to evaluate the quality of machine-generated translations within the domain of machine translation tasks. It involves comparing the output of the machine translation with one or more reference translations to assess the degree of similarity or resemblance. Additionally, the BLEU score is employed in various other natural language-related tasks, including language generation, picture description generation, text summarization, and speech recognition. The BLEU score offers several advantages, such as rapid computation and affordable pricing, making it a practical choice for evaluating large-scale translation systems. Moreover, its results are interpretable and meaningful irrespective of the specific language involved. Importantly, the BLEU score establishes a strong correlation with human evaluation, providing valuable insights into the quality and effectiveness of the machine-generated text. This metric is employed to assess the precision of n-grams and incorporates the Brevity Penalty (BP) to evaluate the similarity of consecutive word sequences between the generated output and the reference text. The scoring system

 doi:10.20944/preprints202409.0834.v1

assigns a value between 0 and 1, where a score of 1 indicates a complete match between the tokens (n-grams) of the generated output and the reference translation, while a score of 0 represents a complete mismatch. By examining the alignment of n-grams and applying the BP, this metric provides an objective measure of how well the generated output aligns with the reference text. The formula for calculating the BLEU score is as follows:

$$BLEU = BP * \exp\left(\sum P_n\right) \quad (1)$$

BP, as showed in formula 2, deducts points when the generated output is excessively brief in comparison to the reference.

$$BP = Min\left(1, \frac{Generated\ output\ length}{Maximum\ reference\ output\ length}\right) \quad (2)$$

P_n represents the accuracy rating of an n-gram, which is determined using the subsequent equation:

$$P_n = \frac{\sum n-grams\ count\ in\ the\ generated\ text}{\sum n-grams\ count\ in\ reference\ text} \quad (3)$$

In this study, we utilized the BLEU criterion to evaluate the similarity between the generated passwords and the passwords within the test set. This criterion serves as a measure of the quality and effectiveness of the generated passwords by assessing their resemblance to the actual passwords in the test set. The criterion considers factors such as shared n-grams and sequence alignment to determine the degree of resemblance. To obtain an aggregate BLEU score, we compute the mean of the individual scores within the created password set. This allows us to summarize the overall performance of the password guessing model in terms of its ability to produce passwords that closely match the passwords in the test set. By employing the BLEU criterion and deriving the aggregate BLEU score, we establish a standardized and objective evaluation framework for the generated passwords.

4.3.3. HIBP

HIBP (Have I Been Pwned) is a widely recognized and reliable website[1] that serves as an extensive repository of compromised passwords and email addresses. Developed by Troy Hunt, its primary aim is to assist individuals in determining whether their personal information, including emails and passwords, has been compromised in any data breaches, thus potentially being vulnerable to further exploitation. HIBP operates by cross-referencing the user-provided email address with its comprehensive database of compromised data. A notable feature of HIBP is its ability to evaluate the safety and strength of passwords. Users can input their passwords, which HIBP then securely transforms into a hashed and anonymized form before searching within its database. If a match is found, it indicates that the password has been compromised in the past and is considered weak or susceptible. By leveraging the HIBP API provided, we can employ a novel and valuable metric for evaluating the strength and security of the generated passwords. This involves conducting a comparison between the generated passwords and the passwords stored in the HIBP repository. This approach enhances our ability to assess the robustness of the generated passwords and identify potential vulnerabilities. By utilizing HIBP and its API in our evaluation process, we integrate a reliable and innovative method for measuring the strength and safety of the generated passwords. This further enhances the security assessment and provides valuable insights into password quality and potential risks.

[1] https://haveibeenpwned.com

 doi:10.20944/preprints202409.0834.v1

4.4. Experimental Results

This section presents the results obtained from on-site experiments conducted on two distinct datasets, namely Myspace and Hotmail. The suggested model was fine-tuned and optimized through multiple iterations, specifically employing iteration rates of 10, 11, and 12. Furthermore, we employed the GNPassGAN model, as described in [28], as the baseline model to evaluate and compare the effectiveness of our approach. For training purposes, the model utilized a specific dataset, and the repetition period was configured to 3000, 4000, and 5000. To provide a comprehensive overview, Table 2 illustrates the training duration required for both the proposed model and GNPassGAN on the two aforementioned datasets. This information allows us to assess the time efficiency of each model and make informed comparisons between them.

Table 2. The training time of the proposed model and the base model.

	#Iteration	Time (minute)	
		Hotmail	Myspace
BBPE_AE	10	27	103
	11	28	121
	12	30	142
GNPassGAN	3000	61	140
	4000	150	162
	5000	186	140

As it is clear from Table 2, when training on the Hotmail dataset with 8,930 passwords, our proposed model exhibited a shorter time duration compared to the GNPassGAN model. In contrast, when training the Myspace dataset, which encompassed a larger volume of 37,126 passwords (four times larger), our proposed model required a longer duration to complete the training process compared to the Hotmail dataset. Despite this, the GNPassGAN model showed a more consistent state and necessitated fewer modifications throughout its implementation. It's worth noting that the GNPassGAN model, inherently a GAN network, necessitates a greater number of iterations to achieve convergence. These observations indicate that the training duration of the proposed model is influenced by the size and complexity of the dataset. While our model demonstrated efficiency in training the Hotmail dataset, it required additional time to process the larger Myspace dataset.

As depicted in Figure 4, through the utilization of the Hotmail dataset for model training, we were able to achieve a high-level accuracy of 0.86. Similarly, when employing the Myspace dataset for training, we achieved an accuracy of 0.87. Additionally, Figure 5 illustrates the variations in error rates throughout the model training process. The error rate was measured at 0.75 and 0.72 for Hotmail and Myspace datasets, respectively, suggesting a relatively low occurrence of errors. The increased accuracy and reduced error rate observed with the Myspace dataset, which had a larger and more diverse collection of passwords, highlight the beneficial effects of incorporating a broader range of data during the training process.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 September 2024 doi:10.20944/preprints202409.0834.v1

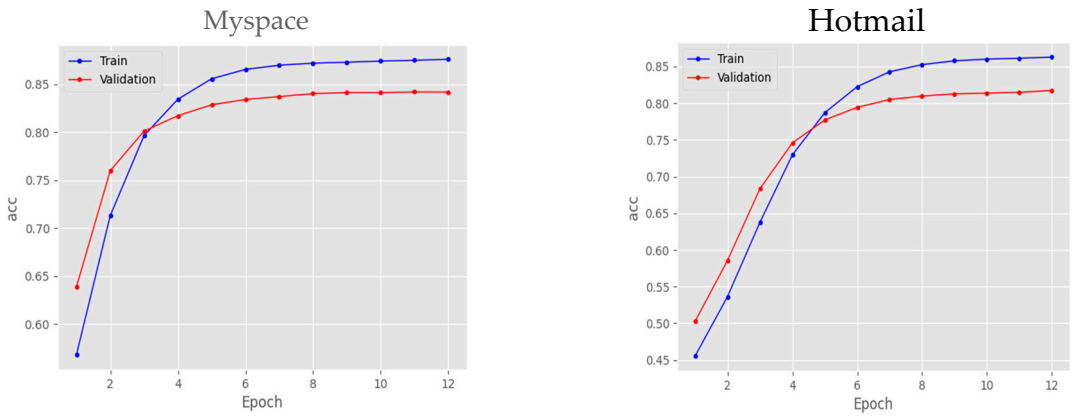

Figure 4. Variation of proposed model's accuracy over training epochs.

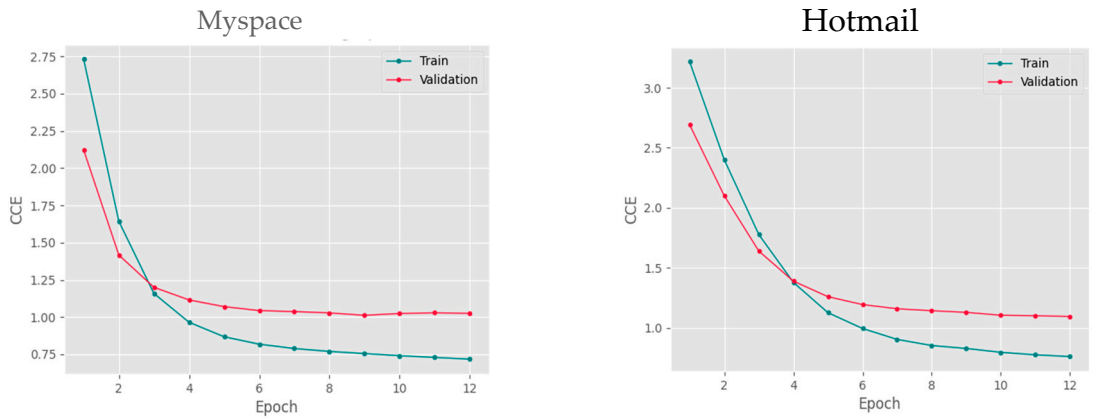

Figure 5. The fluctuations in loss throughout the training process.

Furthermore, we conducted an implementation of the GNPassGAN model on two selected datasets, employing different repetition parameters, and obtained the results, which are presented in Table 3. Upon increasing the number of repetition parameters in the GNPassGAN model for both the Hotmail and Myspace datasets, we observed improvements in all three metrics used to evaluate including duplicate rates, HIBP and BLEU scores. However, it is important to note that due to the specified number of passwords in each phase, the increase in repetition parameters resulted in a reduced number of unique passwords generated, leading to some duplication. This phenomenon can be attributed to the limited pool of passwords available for generation in the dataset. Moreover, analyzing the mean values of the evaluation metrics obtained from the application of the GNPassGAN model to the two datasets provides further insights. As the sample size increases, we observed enhancements in the BLEU criterion, indicating improved similarity between the generated passwords and those in the test set. Additionally, there was a significant reduction in the average duplicate rate, decreasing from 0.0043 to 0.0003. However, it is noteworthy that the HIBP rating exhibited different trends between the larger Myspace dataset and the Hotmail dataset. On average, the HIBP rating decreased by 258.5 for the Myspace dataset, while the decrease was 332.7 for the Hotmail dataset. This suggests that the approach employed in this study detected a reduced number of unique passwords in the larger Myspace dataset compared to the Hotmail dataset.

 doi:10.20944/preprints202409.0834.v1

Table 3. The results of the GNPassGAN model on two datasets, Hotmail and Myspace.

	# Passwords to Generate	# Generated Passwords	Duplicate Rate	HIBP	BLEU	
					unigram	bigram
	5000 Iteration					
	3000	2944	**0.0023**	**574**	**0.88**	**0.79**
	2000	1984	**0.0020**	**383**	**0.88**	**0.78**
	1000	960	**0**	**186**	**0.87**	**0.76**
	4000 Iteration					
	3000	2944	0.0073	496	0.87	0.77
Hotmail	2000	1984	0.0045	332	0.87	0.77
	1000	960	0.0040	164	0.85	0.75
	3000 Iteration					
	3000	2944	0.0076	379	0.86	0.76
	2000	1984	0.0065	257	0.86	0.75
	1000	960	0.0050	134	0.85	0.73
	Average		**0.0043**	**332.7**	**0.86**	**0.76**
	5000 Iteration					
	3000	2944	**0**	**423**	**0.88**	**0.78**
	2000	1984	**0**	**283**	**0.87**	**0.77**
	1000	960	**0**	**153**	**0.86**	**0.76**
	4000 Iteration					
	3000	2944	0.0016	341	0.88	0.76
Myspace	2000	1984	0.0015	235	0.87	0.77
	1000	960	0	116	0.86	0.76
	3000 Iteration					
	3000	2944	0.0003	386	0.88	0.78
	2000	1984	0	263	0.87	0.77
	1000	960	0	127	0.86	0.75
	Average		**0.0003**	**258.5**	**0.87**	**0.76**

Likewise, we obtained the results of implementing our proposed model on the two datasets by varying the number of iterations, and these findings are summarized in Table 4.

 doi:10.20944/preprints202409.0834.v1

Table 4. The results of the proposed model on two datasets, Hotmail and Myspace.

	# Passwords to Generate	# Generated Passwords	Duplicate Rate	HIBP	BLEU	
					unigram	bigram
	12 Iteration					
	3000	3000	0.0006	719	0.90	0.82
	2000	2000	0.0005	472	0.90	0.81
	1000	1000	0.0010	249	0.85	0.79
	11 Iteration					
	3000	3000	**0.0010**	**790**	**0.90**	**0.82**
Hotmail	2000	2000	**0.0015**	**522**	**0.90**	**0.81**
	1000	1000	**0.0010**	**252**	**0.89**	**0.79**
	10 Iteration					
	3000	3000	0.0006	701	0.90	0.82
	2000	2000	0.0015	443	0.90	0.80
	1000	1000	0.0010	217	0.89	0.80
	Average		**0.0009**	**485**	**0.89**	**0.80**
	12 Iteration					
	3000	3000	0.0033	955	0.90	0.81
	2000	2000	0.0030	671	0.89	0.80
	1000	1000	0.0020	346	0.88	0.78
	11 Iteration					
	3000	3000	**0.0043**	**1006**	**0.90**	**0.81**
Myspace	2000	2000	**0.0045**	**674**	**0.89**	**0.80**
	1000	1000	**0.0050**	**316**	**0.88**	**0.78**
	10 Iteration					
	3000	3000	0.0060	940	0.90	0.81
	2000	2000	0.0050	632	0.89	0.80
	1000	1000	0.0040	311	0.88	0.78
	Average		**0.0040**	**650.1**	**0.89**	**0.79**

Upon careful analysis of the results presented in Table 4, the following observations can be made:

1. The outcomes obtained from the proposed model exhibit consistency across different repetition periods. However, a slight improvement in results, particularly in terms of the BLEU and HIBP criteria, can be observed when the repetition period is set to 11. Additionally, employing a repetition period of 12 leads to a reduction in the occurrence of repetitive passwords. It is worth noting that as the dataset size increases, with the utilization of the Myspace dataset instead of Hotmail, the duplicate rate also increases, rising from an average of 0.0009 in the smaller dataset to 0.004 in the larger dataset.

 doi:10.20944/preprints202409.0834.v1

2. The quantity of passwords allocated in the proposed model aligns with the number of passwords generated throughout all steps. In contrast, the GNPassGAN model generates a smaller number of passwords in all procedures.
3. When comparing the duplicate rate of the proposed model with the GNPassGAN model, it is evident that the proposed model performs better in the Hotmail dataset, while it exhibited a slightly higher duplicate rate with an average of 0.004, in contrast to the GNPassGAN model's average of 0.0003
4. The increase in dataset volume has a minimal impact on the quality outcomes measured by the BLEU metric. However, it has yielded positive effects on the HIBP findings, demonstrates that we were able to find more leaked passwords. This can be attributed to the proposed model utilizing additional patterns from the dataset, specifically through the BBPE deployment, to generate improved passwords.

These findings provide valuable insights into the performance of the proposed model across different repetition periods and datasets. The analysis highlights the effectiveness of the proposed model in generating unique and secure passwords, particularly when compared to the baseline model. Moreover, it emphasizes the influence of dataset size on the model's performance, with larger datasets contributing to enhanced outcomes in certain evaluation metrics.

Table 5 presents examples of generated passwords along with their corresponding HIBP evaluation results for the Hotmail and Myspace datasets. The examples extracted from the HIBP standard encompass peculiar codes identified within the set of disclosed passwords. In essence, the model adeptly produced deceptive codes that bear a striking resemblance to the passwords frequently selected by users in real-world situations, amidst the extensive array of compromised passwords.

Table 5. Sample generated passwords by the proposed model with HIBP evaluation result.

Generated Password (Hotmail)	HIBP result	Generated Password (Myspace)	HIBP result
ilovered	2633	moocow1	15320
1234567	4059875	mickey13	10968
superbeta	60	P32069	3
alexarun	25	ilovebill1	1882
moricita	5	Davy1972	1
marita15	187	coolmam1	13
babygirl18	13611	something1	39316
bettyteam	1	madman13	1401
carlos1991	1028	Neston12	3
CHARINI18	1	chrisy11	118

Figure 6 provides a comprehensive visual comparison between the outcomes achieved by the top-performing GNPassGAN model and the best-performing BBPE-AE model, employing a range of assessment criteria. An examination of the diagrams presented in Figure 6 unveils that the utilization of the AE network coupled with the BBPE component in the proposed model has resulted in exceptional outcomes when applied to datasets comprising passwords of varying lengths. This notable achievement serves as a contributing factor to the overall success of the recommended model. Upon careful analysis of the list of passwords generated by the model, a significant revelation came to light. A substantial number of these passwords were identified as suitable options for users, as they remained uncompromised in any known data breaches. This discovery underscores the model's ability to produce passwords that offer enhanced security and protect user accounts from potential breaches.

 doi:10.20944/preprints202409.0834.v1

Figure 6. The optimal outcomes of GNPassGAN and BBPE-AE on the Hotmail and Myspace dataset.

 doi:10.20944/preprints202409.0834.v1

5. Conclusions

This study introduces a novel deep learning approach, incorporating the BBPE tokenizer, to extract meaningful patterns and symbols from datasets contributing to the creation of robust and secure passwords. The model's effectiveness was evaluated by analyzing the frequency of the generated passwords and their similarity to a test password set using the BLEU criteria. The model's ability to generate strong passwords and reduce compromise risk is also demonstrated by evaluating the generated passwords against a test set and considering the prevalence of disclosed codes on the HIPE website. Moreover, in comparison to the baseline model, the findings suggest that this approach, is a promising method for assessing password strength. Consequently, it is advisable to consider employing the passwords generated by the proposed model and storing them securely using additional security measures, that ultimately ensure the proper safeguarding of user accounts, effectively mitigating the risk of breaches and unauthorized disclosure of user data. Further refinement of this approach could enhance password security and contribute to the development of robust authentication systems.

Funding: This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data Availability Statement: The data of this research is available on the provided GitHub link, https://github.com/S-Ghaffari/BBPE-AE.

Acknowledgements: We would like to express our sincere gratitude to all those who have contributed to the successful completion of this research paper. Special thanks to the University of Zanjan for providing the necessary resources and support for our work.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

AE	Auto Encoder
BBPE	Byte-level Byte Per Encoding
CNNs	Convolutional Neural Networks
RNNs	Recurrent Neural Networks

References

1. M.-D. Cano, A. Villafranca, and I. Tasic, Performance Evaluation of Cuckoo Filters as an Enhancement Tool for Password Cracking, Cybersecurity **6**, 57 (2023).
2. T. Zhang, Z. Cheng, Y. Qin, Q. Li, and L. Shi, Deep Learning for Password Guessing and Password Strength Evaluation, A Survey, in 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) (IEEE, Guangzhou, China, 2020), pp. 1162–1166.
3. B. Hitaj, P. Gasti, G. Ateniese, and F. Perez-Cruz, PassGAN: A Deep Learning Approach for Password Guessing, in Applied Cryptography and Network Security: 17th International Conference, ACNS 2019, Bogota, Colombia, June 5–7, 2019, Proceedings (Springer-Verlag, Berlin, Heidelberg, 2019), pp. 217–237.
4. E. Ribeiro de Mello, M. Silva Wangham, S. Bristot Loli, C. E. da Silva, G. Cavalcanti da Silva, S. A. de Chaves, and B. Bristot Loli, *Multi-Factor Authentication for Shibboleth Identity Providers*, Journal of Internet Services and Applications **11**, 8 (2020).
5. C. Gehrmann, M. Rodan, and N. Jönsson, *Metadata Filtering for User-Friendly Centralized Biometric Authentication*, EURASIP Journal on Information Security **2019**, 7 (2019).
6. K. Lim, J. H. Kang, M. Dixson, H. Koo, and D. Kim, Evaluating Password Composition Policy and Password Meters of Popular Websites, in 2023 IEEE Security and Privacy Workshops (SPW) (2023), pp. 12–20.
7. Password Guessing: Learn the Nature of Passwords by Studying the Human Behavior. https://dspace.unive.it/handle/10579/19986.

8. V. Belikov and I. Prokuronov, Password Strength Verification Based on Machine Learning Algorithms and LSTM Recurrent Neural Networks, Russian Technological Journal **11**, 7 (2023).
9. J. Rando, F. Perez-Cruz, and B. Hitaj, PassGPT: Password Modeling and (Guided) Generation with Large Language Models, in (2024).
10. E. Darbutaitė, P. Stefanovič, and S. Ramanauskaitė, Machine-Learning-Based Password-Strength-Estimation Approach for Passwords of Lithuanian Context, Applied Sciences **13**, 13 (2023).
11. Y. Mo, S. Li, Y. Dong, Z. Zhu, and Z. Li, *Password Complexity Prediction Based on RoBERTa Algorithm*, Applied Science and Engineering Journal for Advanced Research **3**, 3 (2024).
12. M. Vainer, Multi-Purpose Password Dataset Generation and Its Application in Decision Making for Password Cracking through Machine Learning, New Trends in Computer Sciences **1**, 1 (2023).
13. B. N. Vi, N. Ngoc Tran, and T. G. Vu The, A GAN-Based Approach for Password Guessing, in 2021 RIVF International Conference on Computing and Communication Technologies (RIVF) (IEEE, Hanoi, Vietnam, 2021), pp. 1–5.
14. Hashcat - Advanced Password Recovery. https://hashcat.net/hashcat/.
15. *John the Ripper Password Cracker*. https://www.openwall.com/john/.
16. Z. Xie, M. Zhang, A. Yin, and Z. Li, A New Targeted Password Guessing Model, in Information Security and Privacy: 25th Australasian Conference, ACISP 2020, Perth, WA, Australia, November 30 – December 2, 2020, Proceedings (Springer-Verlag, Berlin, Heidelberg, 2020), pp. 350–368.
17. M. Xu, C. Wang, J. Yu, J. Zhang, K. Zhang, and W. Han, Chunk-Level Password Guessing: Towards Modeling Refined Password Composition Representations, in Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (Association for Computing Machinery, New York, NY, USA, 2021), pp. 5–20.
18. X. Guo, Y. Liu, K. Tan, W. Mao, M. Jin, and H. Lu, *Dynamic Markov Model: Password Guessing Using Probability Adjustment Method*, Applied Sciences **11**, 4607 (2021).
19. H. Cheng, W. Li, P. Wang, and K. Liang, Improved Probabilistic Context-Free Grammars for Passwords Using Word Extraction, in ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2021), pp. 2690–2694.
20. A Large-Scale Analysis of the Semantic Password Model and Linguistic Patterns in Passwords | ACM Transactions on Privacy and Security. https://dl.acm.org/doi/abs/10.1145/3448608.
21. J. Xie, H. Cheng, R. Zhu, P. Wang, and K. Liang, WordMarkov: A New Password Probability Model of Semantics, in ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2022), pp. 3034–3038.
22. D. Wang, Y. Zou, Z. Zhang, and K. Xiu, *Password Guessing Using Random Forest | USENIX*, in (n.d.).
23. Z. Xie, F. Shi, M. Zhang, H. Ma, H. Wang, Z. Li, and Y. Zhang, *GuessFuse: Hybrid Password Guessing With Multi-View*, IEEE Transactions on Information Forensics and Security **19**, 4215 (2024).
24. I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, *Improved Training of Wasserstein GANs*, in *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Curran Associates Inc., Red Hook, NY, USA, 2017), pp. 5769–5779.
25. B. Hitaj, P. Gasti, G. Ateniese, and F. Perez-Cruz, *PassGAN: A Deep Learning Approach for Password Guessing*, arXiv:1709.00440.
26. C. Fu, M. Duan, X. Dai, Q. Wei, Q. Wu, and R. Zhou, *DenseGAN: A Password Guessing Model Based on DenseNet and PassGAN*, in (2021), pp. 296–305.
27. V. Garg and L. Ahuja, Password Guessing Using Deep Learning, in 2019 2nd International Conference on Power Energy, Environment and Intelligent Control (PEEIC) (IEEE, Greater Noida, India, 2019), pp. 38–40.
28. F. Yu and M. V. Martin, GNPassGAN: Improved Generative Adversarial Networks For Trawling Offline Password Guessing, in 2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW) (IEEE, Genoa, Italy, 2022), pp. 10–18.
29. X. Guo, Y. Liu, K. Tan, M. Jin, and H. Lu, *PGGAN: Improve Password Cover Rate Using the Controller*, J. Phys.: Conf. Ser. **1856**, 012012 (2021).
30. Z. Xia, P. Yi, Y. Liu, B. Jiang, W. Wang, and T. Zhu, *GENPass: A Multi-Source Deep Learning Model for Password Guessing*, IEEE Trans. Multimedia **22**, 1323 (2020).
31. S. Nam, S. Jeon, and J. Moon, Generating Optimized Guessing Candidates toward Better Password Cracking from Multi-Dictionaries Using Relativistic GAN, Applied Sciences **10**, 7306 (2020).

 doi:10.20944/preprints202409.0834.v1

32. M. Kaleel and N.-A. Le-Khac, Towards a New Deep Learning Based Approach for the Password Prediction, in 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) (IEEE, Guangzhou, China, 2020), pp. 1146–1150.
33. H. Li, M. Chen, S. Yan, C. Jia, and Z. Li, Password Guessing via Neural Language Modeling, in Machine Learning for Cyber Security: Second International Conference, ML4CS 2019, Xi'an, China, September 19-21, 2019, Proceedings (Springer-Verlag, Berlin, Heidelberg, 2019), pp. 78–93.
34. Y. Zhang, H. Xian, and A. Yu, *CSNN: Password Guessing Method Based on Chinese Syllables and Neural Network*, Peer-to-Peer Networking and Applications **13**, 6 (2020).
35. D. Huang, Y. Wang, and W. Chen, RLPassGAN: Password Guessing Model Based on GAN with Policy Gradient, in (2022).
36. D. Pasquini, G. Ateniese, and C. Troncoso, Universal Neural-Cracking-Machines: Self-Configurable Password Models from Auxiliary Data, arXiv:2301.07628.
37. M. Xu, J. Yu, X. Zhang, C. Wang, S. Zhang, H. Wu, and W. Han, Improving Real-World Password Guessing Attacks via Bi-Directional Transformers | USENIX, in (n.d.).
38. M. Islam, M. S. Bohuk, P. Chung, T. Ristenpart, and R. Chatterjee, Araña: Discovering and Characterizing Password Guessing Attacks in Practice | USENIX, in (n.d.).
39. X. Su, X. Zhu, Y. Li, Y. Li, C. Chen, and P. Esteves-Veríssimo, *PagPassGPT: Pattern Guided Password Guessing* via *Generative Pretrained Transformer*, arXiv:2404.04886.
40. D. Wang, Y. Zou, Y.-A. Xiao, S. Ma, and X. Chen, Pass2Edit: A Multi-Step Generative Model for Guessing Edited Passwords | USENIX, in (n.d.).
41. C. Wang, K. Cho, and J. Gu, *Neural Machine Translation with Byte-Level Subwords*, Proceedings of the AAAI Conference on Artificial Intelligence **34**, 9154 (2020).
42. *SecLists/Passwords/Leaked-Databases/Myspace.Txt at Master · Danielmiessler/SecLists*. https://github.com/danielmiessler/SecLists/blob/master/Passwords/Leaked-Databases/myspace.txt.
43. *SecLists/Passwords/Leaked-Databases/Hotmail.Txt at Master · Danielmiessler/SecLists*. https://github.com/danielmiessler/SecLists/blob/master/Passwords/Leaked-Databases/hotmail.txt.