Efficient Password Strength Checking on Low-Energy Devices using Neural Networks

SIL765: NSS Project Final Report, April 19, 2024

Supervised by Dr. Vireshwar Kumar

Ganraj Borade
2023MCS2478
Dept. of Computer Science and Engineering
Indian Institute of Technology, Delhi
Delhi, India

Vikas Saini
2023MCS2492
Dept. of Computer Science and Engineering
Indian Institute of Technology, Delhi
Delhi, India

***Abstract*—Traditional methods of authentication, such as human-chosen text passwords, are vulnerable to guessing attacks, posing a significant security concern. However, existing approaches for assessing password strength through adversarial guessing suffer from inaccuracies or computational demands that are unsuitable for real-time, client-side password validation on low-energy devices. In this work, we propose leveraging neural networks to model text passwords' resilience against guessing attacks, aiming to develop an implementation tailored for low-energy devices, specifically targeting the MSP430 device.**

The findings from [2] reveal that neural networks often outperform current state-of-the-art methods, such as probabilistic context-free grammars and Markov models, in guessing passwords. Additionally, it demonstrated that these neural networks can be highly compressed, reducing their size to mere hundreds of kilobytes while maintaining significant guessing effectiveness. Drawing on these outcomes, proposed implementation represents the first principled client-side model for password guessing, capable of analyzing a password's susceptibility to a guessing attack.

Overall, our contributions will enable more precise and practical password validation on low-energy devices than was previously feasible.

I. Introduction

In the realm of cybersecurity, human-chosen text passwords remain the cornerstone of authentication, with their prevalence expected to persist for the foreseeable future. However, this widespread reliance on passwords comes with inherent vulnerabilities, as users often resort to predictable patterns, making them susceptible to **password-guessing attacks**. To combat this threat, proactive password checking has emerged as a crucial tool for evaluating password strength. Traditional approaches typically involve simulating various **password-guessing techniques**, ranging from **probabilistic methods** to off-the-shelf password recovery tools. While effective, these methods are often computationally intensive, demanding significant disk space and execution time, rendering them unsuitable for real-time evaluation, particularly on low-energy devices.

In response to the shortcomings of existing password-guessing methods, the paper proposes a novel solution harnessing the power of **artificial neural networks (ANNs) to enhance the accuracy and practicality of password strength evaluation**. ANNs, designed to approximate complex functions, exhibit remarkable potential in generating password guesses, thus presenting an attractive alternative to traditional techniques [13]. The research endeavors to comprehensively explore the efficacy of ANNs in guessing passwords by varying model size, architecture, training data, and techniques. They compared the performance of ANNs against state-of-the-art models, including Markov models, probabilistic context-free grammars, and software tools utilizing mangled dictionary entries. Through extensive testing, they demonstrated the superior guessing capabilities of ANNs, particularly on non-traditional password policies and scenarios involving large numbers of guesses.

While improving password guessing accuracy is a significant advancement, **our work extends beyond mere efficacy to address the practical constraints of deployment, especially on low-energy devices such as the MSP430**. These devices, characterized by limited resources and stringent energy requirements, present unique challenges for implementing sophisticated password-guessing techniques. Leveraging insights from the research on ANNs, we aim to develop lightweight models optimized for efficiency on low-energy hardware. By fine-tuning model architectures, training methodologies, and compression techniques, we aspire to create a solution capable of accurately assessing password strength while minimizing energy consumption. This endeavor aligns with the broader goal of enabling real-time password evaluation on low-energy devices, thereby bolstering security in IoT ecosystems and other interconnected systems.

Evaluation of the proposed solution encompasses rigorous testing and benchmarking against existing methods to ascertain its effectiveness across various metrics, including accuracy, computational efficiency, memory footprint, and energy con-

sumption. Real-world simulations were conducted to assess performance under diverse conditions, simulating different password policies and attack scenarios. The findings from those evaluations provided valuable insights into the feasibility and scalability of our approach, guiding further refinement and optimization efforts.

In conclusion, our work endeavors to address the pressing need for accurate and practical password strength evaluation, particularly on low-energy devices. By leveraging artificial neural networks and optimizing for efficiency, we aim to overcome the limitations of existing methods, enabling real-time password checking in resource-constrained environments. The outcomes of this work will have far-reaching implications for enhancing cybersecurity in interconnected systems and advancing machine learning research in constrained environments. Through our contributions, we aspire to pave the way for a more secure and resilient authentication landscape, safeguarding critical digital assets against evolving threats.

II. Background and Related Work

Password-guessing attacks exploit the vulnerability of predictable passwords, with varying degrees of success depending on the attack method and system defenses. While phishing, keyloggers, and shoulder surfing render password strength irrelevant, rate-limiting policies can mitigate guessing attacks by locking accounts after a few incorrect attempts. To measure password strength, adversarial password guessing is commonly simulated, with various approaches explored in academia and utilized in password cracking tools. **Probabilistic context-free grammars (PCFGs) and Markov models** are widely studied methods for password guessing, each with its strengths and limitations.

In contrast, the approach employs **artificial neural networks (ANNs) to model passwords, leveraging their ability to approximate complex functions and generate novel sequences**. ANNs are trained to predict the next character of a password based on preceding characters, with specialized techniques like **Monte Carlo simulations** [14] used to estimate guess numbers efficiently. This method offers advantages over traditional approaches, with ANNs capable of efficiently modeling natural language and transferring knowledge between related tasks. By utilizing ANNs, we aim to enhance the accuracy and efficiency of password guessing, particularly on low-energy devices where computational resources are limited.

In research paper, they employed recurrent neural networks (RNNs) due to their efficacy in generating text, particularly at the character level. RNNs possess the ability to process sequences and utilize internal memory to retain information about preceding elements. They experimented with two distinct recurrent architectures and focus on character-level models rather than word-level models, as no established dictionary exists for password generation. Additionally, they explored hybrid models incorporating sub-word units, such as syllables, to enhance prediction accuracy. **The size of the models varies, with one larger network containing over 15 million parameters and a smaller counterpart with approximately 683,000 parameters which are not suitable for deployment in resource-constrained environments like low-energy devices**. They investigated **transference learning techniques** to adapt models to non-traditional password policies with sparse training data, ensuring optimal performance under varying conditions.

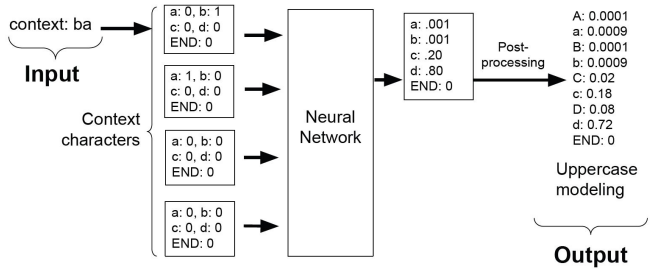


Fig. 1. **An example of using a neural network to predict the next character of a password fragment.** The network is being used to predict a 'd' given the context 'ba'. This network uses four characters of context. The probabilities of each next character are the output of the network. Post processing on the network can infer probabilities of uppercase characters.

For testing, they compared their neural network implementation against several password cracking methods, including **PCFGs** [10], **Markov models** [7], **JtR** [8] and **Hashcat** [9], utilizing multiple datasets to assess guessability. The testing datasets encompass passwords collected from Mechanical Turk and leaked plaintext passwords from 000webhost. We define five distinct testing datasets, each representing different password policies: **1class8** (passwords longer than eight characters), **1class16** (passwords longer than sixteen characters), **3class12** (passwords requiring at least three character classes and a minimum length of twelve characters),**4class8** (passwords requiring all four character classes and a minimum length of eight characters), and **webhost** (randomly sampled passwords from the 000webhost leak containing at least eight characters). These datasets facilitate comprehensive evaluation across various password policies and real-world scenarios, enabling robust assessment of the neural network implementation's performance [16,17].

The evaluation compares various password guessing approaches, including neural networks, Markov models, PCFG, JtR, and Hashcat. Despite **neural networks generally outperforming other models individually, the MinGuess method proves superior, suggesting the preference for using multiple guessing methods for accurate strength estimation**. Neural networks consistently outperform other models from around 10^{10} guesses onwards, matching or surpassing them before that threshold. **Comparing** the client-side neural network implementation to existing password-strength estimators reveals its superiority in accurately measuring passwords' resistance to guessing [6]. The approach exhibits up to half as many unsafe errors as heuristic-based client-side models like Dropbox's zxcvbn [11, 12] meter and the Yahoo! meter, which relies on less sophisticated heuristics. While zxcvbn is tailored to a specific password policy, our neural network approach

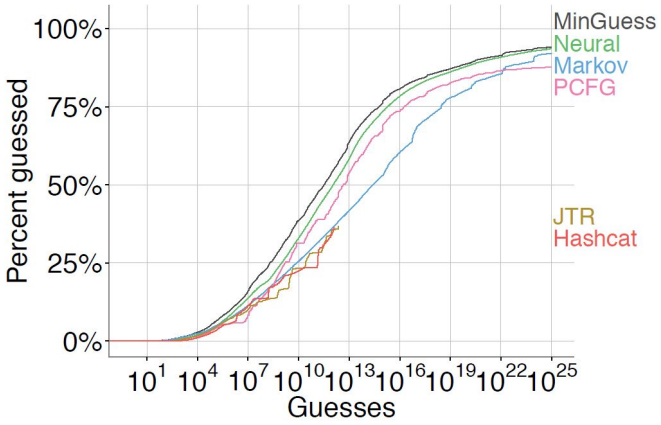


Fig. 2. Guessing of password sets for different guessing methods using the webhost passwords data set. *MinGuess* stands for the minimum number of guesses for any approach. Y-axes are differently scaled to best show comparative performance.

offers greater versatility, easily adaptable to different policies through retraining. This adaptability, coupled with its superior accuracy, positions neural networks as a promising solution for password strength estimation, outperforming existing methods and offering broader applicability across diverse password policies.

TABLE I
1CLASS8

↓	Total	Unsafe
Neural Network	1311	164
zxcvbn	1331	270
Yahoo!	1900	984

TABLE II
4CLASS8

↓	Total	Unsafe
Neural Network	1826	115
zxcvbn	1853	231
Yahoo!	1328	647

Tables: The number of total and unsafe misclassifications for different client-side meters.

III. Problem Statement

In modern computing, GPUs are often costly and not energy-efficient, which presents a challenge for running resource-intensive tasks. This limitation is particularly significant in IoT (Internet of Things) applications where low-energy devices like MSP430FR5994 are preferred due to their cost-effectiveness and widespread use. However, the limited processing power of these devices can pose constraints on the complexity and scale of computations that can be performed efficiently.

Hence, our aim is to develop architecture for running neural networks on low-energy devices, specifically the MSP430FR5994 device.

This initiative aims to create a framework or architecture that enables neural network execution on the MSP430FR5994, addressing the challenges posed by costly and energy-intensive GPUs in IoT applications. The project focuses on optimizing algorithms and implementing strategies adapted for low-energy devices like the MSP430FR5994, ensuring efficient and effective neural network computations within the constraints of these devices. The goal is to bridge the gap between energy-efficient IoT devices and the computational demands of neural networks by developing specialized architectures and strategies tailored for running neural networks effectively on low-energy devices like the MSP430FR5994.

A. System Model

The system consists of the following components:

- **Neural Network Model**: The core of the system is a neural network designed to model text passwords and their resistance to guessing attacks. This model should be lightweight and optimized for low-energy devices, ensuring minimal computational overhead. The neural network should be trained on a dataset of human-chosen passwords, capturing patterns and structures that make passwords more resistant to guessing attacks.
- **Client-side Implementation:**
  The system should be implemented to run locally on low-energy devices, without the need for external resources such as GPU access. The client-side implementation should be optimized for efficiency, ensuring minimal computational and memory usage to accommodate the constraints of low-energy devices.

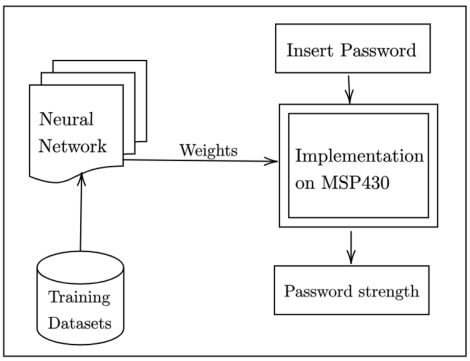


Fig. 3. System model

B. Threat Model

The primary threat to the system is password guessing attacks, where adversaries attempt to crack passwords by systematically guessing them. Adversaries may employ various techniques, including dictionary attacks, brute-force attacks, or social engineering, to guess passwords.

Specific threats to consider include:

- **Brute-force Attacks:** Adversaries attempt to guess passwords by trying all possible combinations systematically.
- **Dictionary Attacks:** Adversaries use a pre-defined list of commonly used passwords or words from dictionaries to guess passwords.

- **Phishing and Social Engineering:** Adversaries attempt to trick users into revealing their passwords through deceptive means.

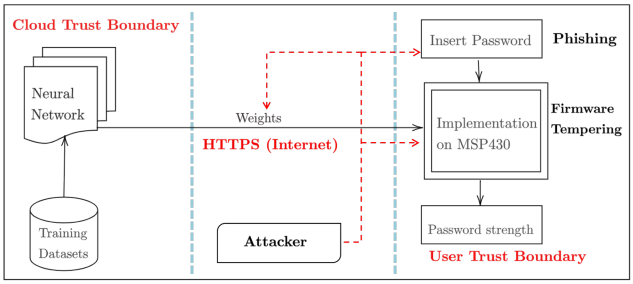


Fig. 4. Threat model

C. Key Challenges

- **Resource Constraints:** Low-energy devices have limited computational power, memory, and energy resources. Developing a system that can run efficiently within these constraints while still providing accurate password strength evaluation is a significant challenge.
- **Optimization for Low-energy Devices:** The neural network model and client-side implementation must be optimized for low-energy devices, considering factors such as algorithmic efficiency, memory usage, and energy consumption.
- **Real-time Performance:** The system should provide real-time feedback on password strength, with minimal latency. Achieving low latency while maintaining accuracy requires careful optimization and possibly trade-offs in model complexity.
- **Security:** Ensuring the security of the system against adversarial attacks is crucial. The system should be resistant to various password guessing techniques, including brute-force and dictionary attacks.
- **Training Data:** Acquiring and curating a dataset of human-chosen passwords for training the neural network model may pose challenges, particularly in ensuring diversity and representativeness of the data.
- **Model Compression**: Compressing the neural network model to reduce its size while maintaining performance is crucial for deployment on low-energy devices with limited storage capacity.

IV. Proposed Solution

In this proposed solution, we address the challenge of running a neural network model trained on a GPU on a low-power MSP430FR5994 device. The limitations of the MSP430FR5994, with its 256KB total FRAM capacity and 8KB RAM, pose significant constraints on deploying complex models [3,5]. Our aim is to devise techniques to optimize the neural network for deployment on such resource-constrained devices.

The neural network used in the referenced paper is trained on GPU hardware. However, deploying such models on low-power devices like MSP430FR5994 is impractical due to cost and energy inefficiency concerns. Given the enormous parameters of the neural network, running it entirely on the MSP430FR5994 is unfeasible. Thus, our challenge is to adapt the neural network model to operate efficiently within the constraints of the MSP430FR5994 while maintaining acceptable levels of accuracy and effectiveness.

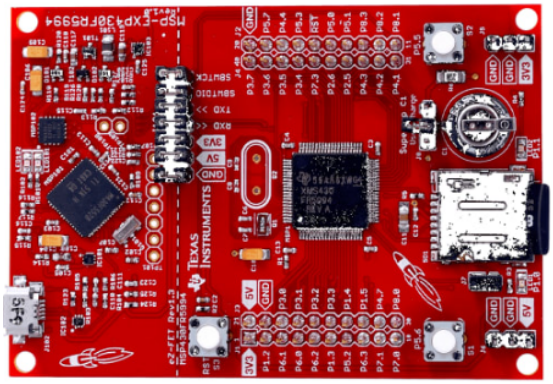


Fig. 5. MSP430FR5994 Device

Parameters	Package | Pins | Size	Features	Description
Frequency (MHz)	16		
Nonvolatile memory (kByte)	256		
RAM (kByte)	8		
ADC type	12-bit SAR		
Number of ADC channels	20		
Number of GPIOs	68		

Fig. 6. Some specifications of MSP430FR5994

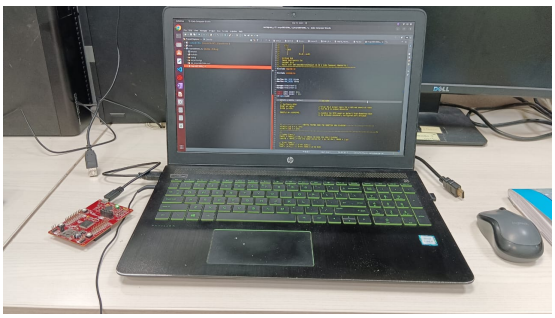


Fig. 7. Setup

- **Weight Pruning:** Weight pruning involves removing connections (weights) from the neural network that are deemed to be insignificant or less critical. By identifying and eliminating these connections, we can reduce the overall size of the neural network, thereby conserving memory and computational resources. We will employ

techniques such as magnitude-based pruning or iterative pruning to systematically remove weights while minimizing the impact on model performance.

- **Layer Reduction:** Another approach is to reduce the number of layers in the neural network architecture. This involves collapsing multiple layers into a single layer or removing entire layers altogether. By simplifying the network architecture, we can reduce the computational and memory requirements, making it more suitable for deployment on low-power devices like the MSP430FR5994. Techniques such as model distillation or using depth-wise separable convolutions can be employed.
- **Quantization:** Quantization involves reducing the precision of the weights and activations in the neural network. Instead of using floating-point representations, we can quantize them to lower precision formats such as fixed-point or integer representations. This reduces memory usage and computational complexity, making the model more efficient for deployment on resource-constrained devices. Techniques like uniform quantization or more advanced methods like K-means clustering can be utilized.
- **Knowledge Distillation:** Knowledge distillation involves training a smaller, more compact neural network (student) to mimic the behavior of a larger, more complex model (teacher). The student network learns not only from the original dataset but also from the soft targets or probability distributions produced by the teacher network. This allows for significant reduction in model size while maintaining performance, especially when the teacher network is a high-performing but bulky model.
- **Architecture Search:** Automated architecture search techniques, such as neural architecture search (NAS), aim to find the most efficient neural network architecture for a given task. These methods explore a vast search space of possible architectures and identify compact models that achieve comparable or even superior performance compared to handcrafted architectures. Techniques like reinforcement learning-based search or evolutionary algorithms can be employed for this purpose.
- **Knowledge Pruning:** Knowledge pruning involves removing redundant or unnecessary components from a neural network while preserving its functionality. Unlike weight pruning, which focuses on removing individual weights, knowledge pruning targets entire neurons, channels, or even entire layers based on their importance to the model's performance. This method can be more aggressive in reducing model size while maintaining accuracy.
- **Sparse Matrix Representations:** Instead of storing and computing operations on dense matrices, sparse matrix representations store only non-zero elements and their corresponding indices. Sparse matrices can significantly reduce memory footprint and computational overhead, especially for large neural networks with many connections. Techniques such as compressed sparse row (CSR) or compressed sparse column (CSC) formats can be employed for efficient storage and computation.
- **Factorization Methods:** Factorization methods decompose weight matrices into smaller, factorized matrices with fewer parameters. Techniques like matrix factorization, tensor decomposition, or low-rank approximation can be used to reduce the number of parameters in the model while retaining its representational capacity. These methods exploit redundancy and correlation in the weight matrices to achieve compression.

In summary, optimizing neural network architectures for deployment on resource-constrained devices requires careful consideration of techniques such as weight pruning, layer reduction, quantization, knowledge distillation, knowledge pruning, sparse matrix representations, factorization methods, and automated architecture search. By employing these techniques judiciously, we can develop efficient models that meet the performance requirements of the application while operating within the constraints of the target device.

V. Evaluation

After initially familiarizing ourselves with the MSP430 device, we embarked on running basic programs to gain insights into its functionality and comprehended the intricacies of memory utilization. Our journey commenced with a rudimentary understanding of the device's capabilities, particularly focusing on memory utilization. To assess the extent of memory utilization, we adopted an iterative approach. We initiated by employing a single array and gradually incremented its size, meticulously observing the memory consumption. This empirical method enabled us to grasp the boundaries of memory allocation within the MSP430FR5994 microcontroller.

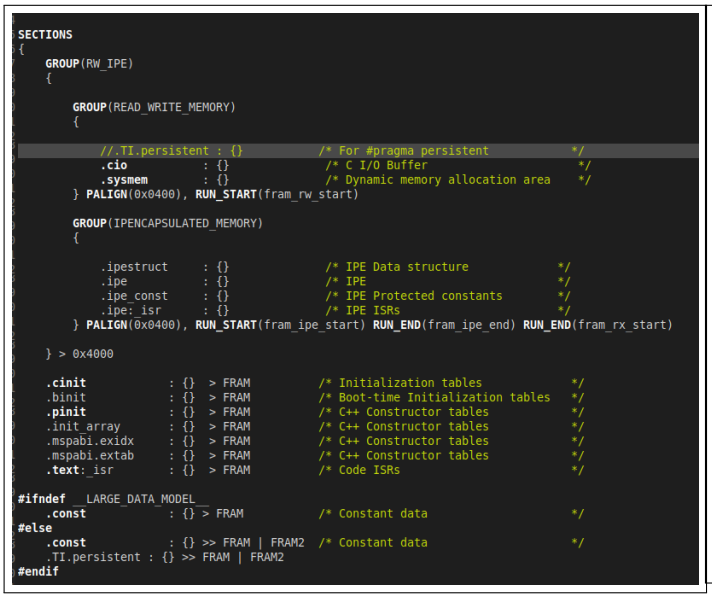
- We initially discovered that we could only utilize a modest portion of the available memory, approximately 48KB.
- Upon identifying this limitation, we delved into exploring potential solutions, which led us to examine the linker file.
- Modification of the linker command file emerged as the viable solution. We augmented the file by incorporating the directive as seen as Fig. 8:

```
.TI.persistent : {} >> FRAM | FRAM2
```

This directive facilitated the allocation of memory, leveraging both FRAM partitions sequentially, thereby significantly expanding the available memory space. Notably, the MSP430FR5994 features two distinct partitions of FRAM, one comprising 48KB and the other 208KB. By utilizing both partitions, we were able to effectively increase the memory capacity accessible to our application.

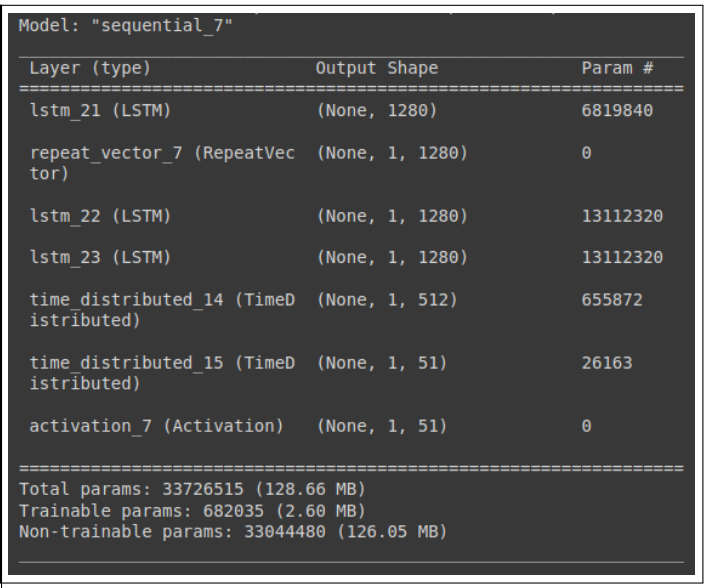
This strategic alteration propelled us to a pivotal juncture where we could harness the entirety of the MSP430FR5994 device memory, effectively leveraging the FRAM resource [4].

Subsequently, we proceeded to extract the neural network architecture from a GitHub repository as seen in Fig. 9, leveraging the provided code. Our methodology involved extracting



```
SECTIONS
{
    GROUP(RW_IPE)
    {

        GROUP(READ_WRITE_MEMORY)
        {

            //.TI.persistent : {}          /* For #pragma persistent           */
            .cio          : {}             /* C I/O Buffer                      */
            .sysmem       : {}             /* Dynamic memory allocation area    */
        } PALIGN(0x0400), RUN_START(fram_rw_start)

        GROUP(IPENCAPSULATED_MEMORY)
        {

            .ipestruct     : {}            /* IPE Data structure            */
            .ipe           : {}            /* IPE                           */
            .ipe_const     : {}            /* IPE Protected constants       */
            .ipe:_isr      : {}            /* IPE ISRs                      */
        } PALIGN(0x0400), RUN_START(fram_ipe_start) RUN_END(fram_ipe_end) RUN_END(fram_rx_start)

    } > 0x4000

    .cinit           : {}  > FRAM         /* Initialization tables            */
    .binit           : {}  > FRAM         /* Boot-time Initialization tables  */
    .pinit           : {}  > FRAM         /* C++ Constructor tables           */
    .init_array      : {}  > FRAM         /* C++ Constructor tables           */
    .mspabi.exidx    : {}  > FRAM         /* C++ Constructor tables           */
    .mspabi.extab    : {}  > FRAM         /* C++ Constructor tables           */
    .text:_isr       : {}  > FRAM         /* Code ISRs                        */

#ifndef __LARGE_DATA_MODEL__
    .const           : {} > FRAM          /* Constant data                    */
#else
    .const           : {} >> FRAM | FRAM2  /* Constant data                    */
    .TI.persistent : {} >> FRAM | FRAM2
#endif
```

Fig. 8. Linker file



```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_21 (LSTM)              (None, 1280)              6819840

 repeat_vector_7 (RepeatVec  (None, 1, 1280)           0
 tor)

 lstm_22 (LSTM)              (None, 1, 1280)           13112320

 lstm_23 (LSTM)              (None, 1, 1280)           13112320

 time_distributed_14 (TimeD  (None, 1, 512)            655872
 istributed)

 time_distributed_15 (TimeD  (None, 1, 51)             26163
 istributed)

 activation_7 (Activation)   (None, 1, 51)             0

=================================================================
Total params: 33726515 (128.66 MB)
Trainable params: 682035 (2.60 MB)
Non-trainable params: 33044480 (126.05 MB)
_________________________________________________________________
```

Fig. 9. Neural Network Extracted

the neural network configuration in the form of a JSON file. Subsequently, we meticulously transformed the extracted model into a TensorFlow-compatible format. This conversion process facilitated a comprehensive examination of the neural network's parameters.

- Upon converting the model to TensorFlow format, we conducted an in-depth analysis of its parameters.
- We then checked the total number of trainable parameters essential for its operation, particularly in the context of password guessing.
- Our analysis revealed that the neural network encompasses a total of approximately 683,000 trainable parameters, underscoring the complexity and resource requirements inherent in its operation.
- **Development of Mechanism for MSP430 Compatibility:** The methods of weight pruning and layer reduction are primarily used keeping in mind the model's utility. The paper [1] has given the basic blueprint of implementation of the neural network on 16-bit microcontrollers. However, the layers needed for our model were not implemented in the paper's repository, such as LSTM layer and global max pooling 2D layer. We added their functionalities in the code.
  Upon testing, we observed that 80751 is the maximum number of parameters that are supported. This finding contradicts the paper's claim that it can maximally support 65,000 parameters.
- **Training and Accuracy Evaluation:** Secondly, the test data provided in the GitHub repository was utilized for training the neural network, enabling the evaluation of its accuracy. This step involved training the model with various architectures and hyperparameters to find the optimal configuration that balanced accuracy with computational efficiency. Techniques like cross-validation and robust evaluation metrics were employed to ensure the reliability of accuracy measurements, especially in scenarios with limited computational resources for extensive testing.
- **Model Optimization Techniques:** Thirdly, to make the neural network smaller yet effective, techniques such as weight pruning and layer reduction were employed. Weight pruning involved removing insignificant connections from the network, while layer reduction simplified the network architecture by reducing the number of layers without a significant loss in performance. Furthermore, quantization techniques were applied to represent weights and activations with fewer bits, reducing memory and computational requirements. Additionally, advanced optimization methods such as network architecture search or neural architecture synthesis were explored to automatically discover compact and efficient model architectures tailored to the constraints of the MSP430FR5994.
- **Compatibility with MSP430FR5994:** Lastly, creating an MSP430FR5994-compatible password guessability neural network required adapting the model architecture and training process to suit the constraints of the target device while ensuring that the model maintained its effectiveness in predicting passwords. This adaptation involved redesigning certain components of the neural network to leverage hardware features specific to the MSP430FR5994, such as its FRAM memory or low-power operation modes. Furthermore, rigorous testing and validation on the target device were essential to ensure that the adapted model met the performance requirements in real-world scenarios while operating within the resource limitations of the MSP430FR5994 platform. Result of security parameter can be seen in Fig. 10.

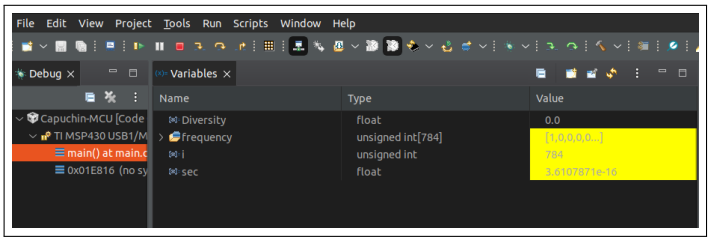


Fig. 10. Security parameter giving the probability of guessing the password.

A. Performance Analysis

As mentioned earlier, we can run any neural network with parameters $\leq$ 80,751. Our optimization strategies primarily involve weight pruning and layer reduction methods, which help in managing model complexity and computational resources efficiently.

B. Security Analysis

In our security analysis, we consider the scenario where the input to the neural network is a password represented in binary format. The output of the network is the probability of guessing the password correctly, which serves as a measure of security strength. To calculate the security parameter value, we determine the probability of successfully guessing the password by respective neural network. This probability is influenced by factors such as password complexity, network architecture, and security measures implemented within the neural network. Each bit of the password input represents a specific character or component of the password. This binary representation facilitates processing within the neural network and ensures compatibility with its input requirements. In addition to input encoding, security measures within the neural network, such as encryption techniques, authentication layers, and anomaly detection mechanisms, enhance the overall security posture. These measures mitigate risks associated with unauthorized access and data breaches.

VI. Discussion

In the course of addressing the challenges and proposing innovative solutions outlined in this work, several considerations regarding limitations and opportunities for future research have emerged. This section aims to explore these aspects in detail, highlighting areas for improvement and potential avenues for extending the scope of our findings.

A. Limitations

While our project has made significant strides in evaluating password strength using neural networks on low-energy devices, it is important to acknowledge certain limitations inherent in our approach. These limitations highlight areas where further research and refinement are necessary to optimize performance and address potential challenges.

- **Scope of Password Strength Evaluation:** This project focuses exclusively on evaluating the strength of passwords against guessing attacks using neural networks. It does not address the broader aspects of implementing password management systems or authentication protocols, which are essential components of comprehensive cybersecurity strategies.
- **Potential Margin of Error:** While demonstrating the feasibility of password guessing on low-energy devices, there may be inherent limitations in accuracy. The use of multiple model reduction methods could introduce errors in prediction accuracy, estimated to be within a range of 5-10 %. Future efforts should aim to mitigate these inaccuracies through improved model optimization techniques.
- **Constraints on Neural Network Size:** The project is limited by the capacity of low-energy devices like the MSP430FR5994, which may not accommodate larger neural network models due to constraints in FRAM capacity. This limitation underscores the need for further research into efficient model compression and deployment strategies tailored for resource-constrained environments.

B. Future Work

Looking ahead, several exciting avenues for future research and development emerge from our current work. These future directions aim to expand the applicability and effectiveness of password evaluation techniques on low-energy devices, paving the way for enhanced cybersecurity practices and innovative solutions.

- **Extending to Other Low-Energy Devices:** This implementation can serve as a blueprint for deploying cloud-trained neural networks on other low-energy devices, particularly 16-bit microcontrollers. Exploring the adaptation of this approach to a wider range of hardware platforms will broaden the applicability of cloud-trained models in IoT and edge computing scenarios.
- **Exploration of 32-bit Microcontrollers:** Future investigations can explore the utilization of 32-bit microcontrollers to evaluate the scalability and performance improvements achievable with higher-capacity hardware. This exploration will expand the scope of potential applications for cloud-trained neural networks in resource-constrained environments.
- **Further Optimization Techniques:** Ongoing research efforts should focus on refining optimization techniques for running neural networks on low-energy devices. This includes exploring novel compression methods, quantization techniques, and hardware accelerators to enhance efficiency and minimize computational overhead, enabling more extensive deployment of neural network models on constrained hardware platforms.

VII. Conclusion

In the realm of cybersecurity, traditional password authentication methods face vulnerabilities due to predictable patterns, making them susceptible to guessing attacks. To address this, the paper proposes leveraging artificial neural networks (ANNs) for password strength evaluation. ANNs,

particularly recurrent neural networks (RNNs)[14, 15], are explored for their effectiveness in generating password guesses. The research investigates various architectural configurations and training methodologies to enhance guessing accuracy.

The proposed solution aims at developing a JavaScript implementation tailored for low-energy devices, specifically targeting the MSP430. Challenges such as resource constraints and real-time performance are addressed through techniques like weight pruning, layer reduction, and quantization. The evaluation involves rigorous testing against existing methods, demonstrating the superior performance of ANNs in guessing passwords.

To ensure compatibility with the MSP430, the planned evaluation includes developing mechanisms for running the neural network on the device, training and evaluating its accuracy, and optimizing the model for efficiency. Techniques like model quantization and knowledge distillation will be employed to reduce computational and memory requirements while maintaining effectiveness.

Overall, the proposed solution offers a comprehensive approach to enhancing password strength evaluation, particularly on low-energy devices, contributing to a more secure authentication landscape in interconnected systems.

We were successfully able to exceed the parameter support of the paper [1] by 12,751 parameters.

Our code can be found out here : https://github.com/ganrajborade/ML-on-ultra-low-power-devices/tree/main

VIII. References

[1] T. Kannan and H. Hoffmann, "Budget RNNs: Multi-Capacity Neural Networks to Improve In-Sensor Inference Under Energy Budgets," IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), Virtual Conference, 2021, pp. 143-156. Available : https://doi.org/10.1109/RTAS52030.2021.00020

[2] W. Melicher, B. Ur, S. M. Segreti, S. Komanduri, L. Bauer, N. Christin, and L. F. Cranor, *"Fast, Lean, and Accurate: Modeling Password Guessability Using Neural Networks,"* in Proc. USENIX Security Symposium, Austin, TX, 2016, pp.175-191. Available: https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_melicher.pdf

[3] S. Heller, P. Woias, *"Microwatt power hardware implementation of machine learning algorithms on MSP430 microcontrollers,"* in Proc. IEEE International Conference on Electronics, Circuits and Systems (ICECS), Genoa, Italy, 2019, pp.25-28. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8964726

[4] *"MSP430FR5994,"* Texas Instruments, Rev. D, 2021. [Online] Available: https://www.ti.com/product/MSP430FR5994 [Accessed: April 16,2024].

[5] IDEA League PhD School on Transiently-Powered Systems, *"msp430-xor,"* GitHub, 2017. [Online] Available: https://github.com/phdschooltpc/msp430-xor [Accessed: April 16, 2024].

[6] KELLEY P. G., KOMANDURI S., MAZUREK M. L., SHAY R., VIDAS T., BAUER L., CHRISTIN N., CRANOR L. F., AND LOPEZ J, *"Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms,"* in Proc. IEEE Symp. Security & Privacy, San Francisco, CA, USA, 2012, pp.523-537. Available: https://users.ece.cmu.edu/~lbauer/papers/2012/oakland2012guessing.pdf

[7] MA J., YANG W., LUO M., AND LI N, *"A study of probabilistic password models,"* in Proc. IEEE Symp. Security & Privacy, Berkeley, CA, USA, 2014, pp.689-704. Available: https://dl.acm.org/doi/abs/10.1155/2020/8837210

[8] PESLYAK A., "John the Ripper password cracker",Openwall. [Online] Available: http://www.openwall.com/john/ [Accessed: April 16,2024].

[9] STEUBE, J. Hashcat, *"Hashcat - advanced password recovery,"* hashcat.net. [Online] Available: https://hashcat.net/oclhashcat/ [Accessed: April 16,2024].

[10] WEIR M., AGGARWAL S., MEDEIROS B. D., AND GLODEK B, *"Password cracking using probabilistic context-free grammars,"* in Proc. IEEE Symp. Security & Privacy, Oakland, CA, USA, 2009. Available: https://ieeexplore.ieee.org/abstract/document/5207658/authors

[11] WHEELER D., "zxcvbn, Realistic password strength estimation", Dropbox.tech, 2012. [Online] Available: https://blogs.dropbox.com/tech/2012/04/zxcvbnrealistic-password-strength-estimation/ [Accessed: April 16,2024].

[12] WHEELER, D. L, *"zxcvbn: Low-budget password strength estimation,"* in Proc. USENIX Security, Austin, TX, 2016, pp.157-173. Available: https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_wheeler.pdf

[13] CIARAMELLA A., D'ARCO P., DE SANTIS A., GALDI C., AND TAGLIAFERRI R, *"Neural network techniques for proactive password checking,"*, in Journal IEEE Transactions on Dependable and Secure Computing, volume 3, issue 4, 2006, pp.327–339. Available: https://ieeexplore.ieee.org/document/4012645

[14] DELL'AMICO M. AND FILIPPONE M, *"Monte Carlo strength evaluation: Fast and reliable password checking,"* In Proc. ACM CCS, Denver, Colorado, 2015. Available: https://www.researchgate.net/publication/281117950_Monte_Carlo_Strength_Evaluation_Fast_and_Reliable_Password_Checking [Accessed: April 16,2024]

[15] GRAVES, A, *"Supervised Sequence Labelling with Recurrent Neural Networks"*. Springer, 2012.

[16] HERLEY C., AND VAN OORSCHOT P, *"A research agenda acknowledging the persistence of passwords,"* IEEE Security & Privacy Magazine, volume 10, issue 1 , 2012, pp.28–36. Available: https://dl.acm.org/doi/10.5555/2360743.2360824

[17] UR B., SEGRETI S. M., BAUER L., CHRISTIN N., CRANOR L. F., KOMANDURI S., KURILOVA

D., MAZUREK M. L., MELICHER W., AND SHAY R, *"Measuring real-world accuracies and biases in modeling password guessability,"* in Proc. USENIX Security Symposium, Washington, D.C., 2015, pp.463-481. Available: https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-ur.pdf