Karlsruhe Institute of Technology

Karlsruhe Reports in Informatics 2024,2
Edited by Karlsruhe Institute of Technology,
Faculty of Informatics
ISSN 2190-4782

Proseminar Mobile Computing WS 2023/24

Mobile und Verteilte Systeme
Ubiquitous Computing
Teil XXI

Herausgeber: Alexander Studt, Paul Tremper
Chaofan Li, Haibin Zhao, Michael Beigl

2024

KIT – The Research University in the Helmholtz Association

Karlsruhe Institute of Technology

Proseminar Mobile Computing
WS 2023/24

Mobile und Verteilte Systeme
Ubiquitous Computing

Teil XXI

Herausgeber

Alexander Studt, Paul Tremper
Chaofan Li, Haibin Zhao
Michael Beigl

Karlsruhe Institute of Technology (KIT)
Fakultät für Informatik
Lehrstuhl für Pervasive Computing Systems (PCS) und TECO

Interner Bericht 2024,2
ISSN 2190-4782

Vorwort

Die Seminarreihe Mobile Computing und Ubiquitäre Systeme existiert seit dem Wintersemester 2013/2014. Seit diesem Semester findet das Proseminar Mobile Computing am Lehrstuhl für Pervasive Computing System statt.

Das Proseminar Mobile Computing wird seit dem Wintersemester 2013/2014 in jedem Semester durchgeführt. Seit dem Wintersemester 2003/2004 werden die Seminararbeiten als KIT-Berichte veröffentlicht. Ziel der Seminarreihe ist die Aufarbeitung und Diskussion aktueller Forschungsfragen.

Dieser Seminarband fasst die Arbeiten der Seminare des Wintersemesters 2023-2024 zusammen. Die Themen der hier zusammengefassten Aufsätze umfasst die Themen "Solving the Long-Tail Problem", "Parallelization in Python", "Active Mobile Exoskeletons", und "Quantum Computing". Wir danken den Studierenden für ihren besonderen Einsatz, sowohl während des Seminars als auch bei der Fertigstellung dieses Bandes.

Karlsruhe, den 05. August 2024

Alexander Studt
Paul Tremper
Chaofan Li
Haibin Zhao
Michael Beigl

Inhaltsverzeichnis

Ton That Hoai An
Improving DRAM with Regards to Performance,
Energy-Efficiency and Reliability 1

Cornelius Niklas Breitschwerdt
Mobile Aktive Exoskelette in der Industrie 15

Quoc Anh Dang
NRAM as a Potential DRAM Replacement 37

Beyza Keskin
Quantum Annealing 48

Annemarie Schaub
Recent Development in AI in Video Games 63

Peter Bohner
Memory Considered Harmful:
The Rowhammer Vulnerability in DRAM 74

Nathan Ridinger
Solving the Long Tail Problem with Sampling Stratgies 86

Improving DRAM with regards to performance, energy-efficiency and reliablity

Ton That, Hoai An

Karlsruher Institut für Technologie, Fakultät für Informatik, 76128 Karlsruhe, Germany

Abstract. Many fields of computer science are dependent upon data. Much of this data needs to be stored in devices capable of retaining the stored data for a prolonged duration. These devices need to be reliable, safe, fast, cheap and simple to produce. Such devices are Random Access Memory types like Dynamic Random Access Memory, Static Random Access Memory or Resistive Random Access Memory. In this seminar paper we will look at a technology called Copy-ROW Dynamic Random Access Memory to improve upon the aforementioned properties while being flexible and cheap to produce. The evaluation results show a significant improvement in performance, energy-efficiency as well as reliability.

Keywords: memory · performance · energy-efficiency · reliability

1 Introduction

Random Access Memory (RAM) is an elementary component in each computation system on the market. The consumer-based evaluation nowadays is mostly based on the memory capacity the RAM module provides. This metric does not represent the entire capabilities of a RAM module. Other factors that need to be accounted for are performance (e.g., refresh latency, access-time), security, reliability and durability. Many new explored RAM types try to mitigate the weaknesses of previous RAM type generations. Nonetheless the common types of RAM on the market are still Dynamic Random Access Memory (DRAM) and Static Random Access Memory (SRAM). Both RAM types are well explored till date. The important difference between those is that, DRAM can be easily replaced by the consumer while on the other side SRAM is integrated into the hardware making it a less important concern for the consumer. The goal of this seminar paper is to delve into different techniques and structures to improve DRAM and enable new possibilities regarding performance, energy-efficiency and reliability.

2 Background

This section describes the DRAM organization and its operations that are necessary to understand the inner workings of DRAM.

2.1 DRAM - Organization

A typical DRAM based system is shown in Fig. 1. In those systems the bits flow through separate Input/Output (I/O) buses from DRAM to Central Processing Unit (CPU). This flow is controlled by several memory controllers of the CPU that are connected through a DRAM channel. A DRAM channel can be connected to several modules. The DRAM module itself is ordered hierarchical, starting from DRAM rank down to DRAM subarray. The DRAM module consists of billions of DRAM cells ordered in the aforementioned hierarchy. The DRAM module itself contains several DRAM ranks which are ordered again into DRAM chips. Those DRAM chips contain several DRAM banks where the actual DRAM subarray containing the DRAM cells is present. The subarray is structured like a matrix (e.g., 128x128) [11,2,5,6]. A row of cells in the matrix is connected by a wordline which is selected by a row decoder. The columns are connected with bitlines. At the end of the bitlines, a row buffer sits to buffer the data from the activated line. The row buffer also contains the bitline sense amplifiers (BLSA) to amplify the signal [2,5]. This structure enables the DRAM to continuously access and update the stored data.

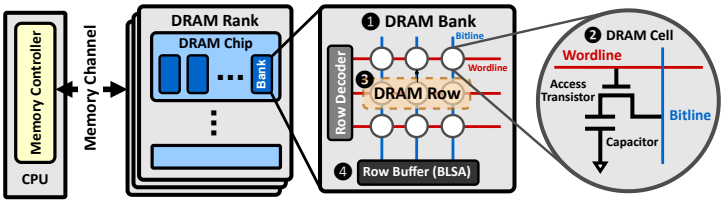

Fig. 1. DRAM module hierarchy [5]

2.2 DRAM - Internal cell functionality

A DRAM cell consists of a transistor and a capacitor as seen in Fig. 1. The DRAM cell contains a bit depending on the current charge level. Due to the nature of a capacitor and the size of the cell, the charge leaks with time and needs to be refreshed [11,2,5,4]. Periodically each row is activated, allowing the charges to travel through the bitlines towards the row buffer. For the periodic refresh, the bitlines need to be precharged to a certain level. Generally the required charge level is $V_{dd}/2$. The BLSA in the row buffer then amplifies the charge back into the cells. Each operation on a cell requires a specific amount of time until a stable state is reached, these times are called timing parameters [4].

2.3 DRAM - Operations

To communicate with the outside world, the memory controller provides the Precharge, Activate, Read, Write and Refresh operations. All these operations

are restricted by timing parameters to guarantee a correct execution. Within these operations the Refresh operation takes a special position as it is periodically called to prevent data loss due to charge leakage [2].

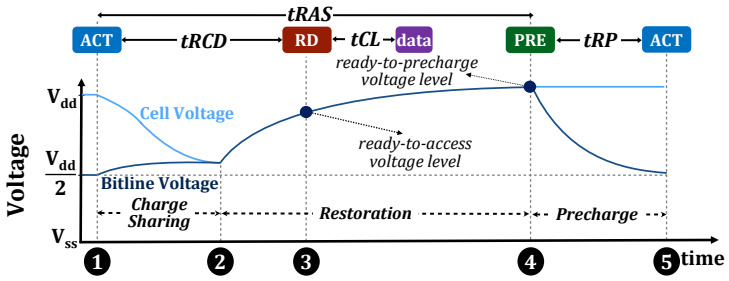

Fig. 2. DRAM operations, operation timings and current voltage during a Read [2]

2.3.1 Activate The Activate (ACT) operation applies an electrical current to the specified row address sent alongside the operation if the row was not already activated. The row decoder decodes the row address and determines the selected row. After the activation, the data contained in the cells are send to the row buffer. This operation is constrained by the timing parameter *Row Address to Column Address Delay* (tRCD) which is the required time until the data has stabilized in the row buffer. [2,5]. The operation requires the bitline to be precharged as opening the row allows the transistor charge to flow towards the bitlines. After stabilizing the charge, the BLSA detects the change in charge and restores the charge inside the transistor depending on the read change. The charge restoration is necessary as the capacitor does not contain the correct charge after the row opening [2].

2.3.2 Read The Read (RD) operation can be started after a row was activated. The column address is sent alongside this operation to specify the part inside the matrix to be retrieved. This data is then read and stored in the global row buffer. From the buffer the data then moves through a connected bus to the memory controller. The timing parameter for this operation is called *CAS latency* (tCL) and decides the required time until the data appears on the bus [2,4].

2.3.3 Write The Write (WR) operation operates similar to the ACT operation. The Write operation requires a charge restoration. Instead of just restoring the previous charge, the BLSA sends the to be written data into the capacitors. The timing parameter *Write Recovery Time* (tWR) defines the required passing time till the next operation [2,4].

2.3.4 Refresh As previously mentioned, a DRAM cell can not retain its state for a prolonged time due to the charge leakage. To ensure the correct retrieval of the data at any point in time, this refresh (REF) operation is periodically called in a specified interval. Each row is typically refreshed in a standardized time frame (e.g., 64ms) [2,5,4].

2.3.5 Precharge The Precharge (PRE) operation is necessary to close a row and prepare the bitlines for the next ACT. The timing parameters tRAS decides the time until the next PRE can be called. The time itself the PRE operations need is called *Row Precharge Time* (tRP). It allows the bitlines enough time to be charged to the required charge level [2,4].

3 Copy-Row DRAM

The Copy-Row DRAM (CROW) is a new technology proposed in the paper [2]. It tries to mitigate the bottleneck, which the regular DRAM causes, by using new mechanisms for faster, more energy-efficient and more reliable use of DRAM.

3.1 Copy-Row DRAM - Overview

CROW introduces two new components to achieve the desired properties.

The first component are *copy rows* that are, additionally to the regular rows, located in each subarray. Those can be used to copy or remap data stored in the regular rows [2].

The second component is the *CROW-table* located inside the memory controller. It tracks the occupied copy rows and its corresponding regular row [2].

As CROW uses two different types of rows, both need to be addressed separately thus resulting in another row decoder for the copy rows. This allows a new action called *multiple-row activation* (MRA) which activates copy rows and regular rows simultaneously. Two new DRAM primitives are possible due to this new action.

First, CROW can perform bulk data movement, it can copy content from a regular row to a copy row. This is achieved by activating the regular row first. After the ACT operation, the data is stored inside the buffers. The copy row is activated additionally to the regular row. The sense amplifers now redirect the charge towards the open regular row as well as the open copy row [2].

Second, CROW can perform reduced-latency DRAM access under certain conditions. If a regular row got copied beforehand into a copy row, the memory controller can activate both simultaneously [2]. Two charges are spreading inside the bitlines which results into faster charge time for the sense amplifiers. The sense amplifier can redirect the charge faster as a result thus decreasing the activation time.

3.1.1 Copy Row As mentioned previously, CROW contains two different types of rows as shown in Fig. 3. On one side the usual rows in every DRAM, denoted here as regular rows, on the other side the copy rows. These copy rows can be accessed independently as a second row decoder is used to address these rows. The occupied area of the second row decoder is much smaller than the one for the regular rows as the amount of copy rows will be kept small. Additionally to the regular row, now another address needs to be send from the memory controller. Due to the amount of copy rows, not many extra bits are necessary for the address.

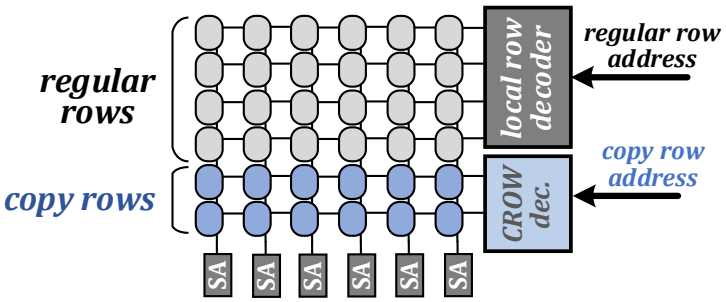

Fig. 3. CROW DRAM structure [2]

3.1.2 CROW-table The CROW-table is located in the memory controller. A rough organization of the data structure can be seen in Fig. 4. It stores data regarding the copied regular rows inside the copy rows. The CROW-table itself is n-way set associative, n is the amount of copy rows that are located in each subarray. The table is accessed by the memory controller by using the bank- and subarray-address to retrieve. The current design requires three fields for each entry. First, a valid field to check whether the copy row is valid. Second, a pointer towards the copied regular row. Third, additional information for CROW operations.

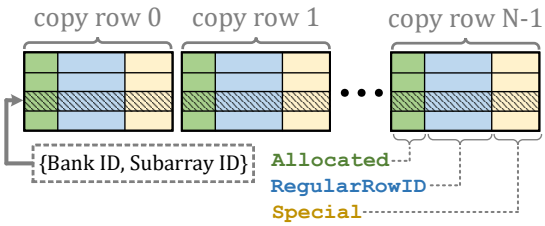

Fig. 4. CROW-table [2]

3.2 CROW - New operations

CROW introduces two new operations that exploit the new subarray structure. These operations aim to improve upon performance, energy-efficiency and reliability.

3.2.1 Activate-and-copy The Activate-and-copy operation (ACT-c) executes a copy operation from a regular row to a copy row. The operation first performs a regular row activation. After the sense amplifier is charged, the copy row is then activated as well. Due to there being two opens rows now, both will be charged with the corresponding data. For this operation the regular row address and copy row address are required. The addresses will then be supplied to the corresponding row decoder.

3.2.2 Activate-two The Activate-two operation (ACT-t) executes two activate primitives at once. Assuming the to be activated row is also existent in another copy row, the ACT-t operation activates both rows simultaneously. As this operation activates two rows at once, it requires the regular row and copy row address. A lookup into the CROW-table before ACT-c is required as the copy row might be invalid.

3.3 CROW - Improvement mechanisms

Three mechanisms are proposed in the paper [2] to improve upon the properties performance, energy-efficiency and reliability. The first strategy is called CROW-cache, it takes advantage of the MRA operation to achieve faster access time as well as less energy consumption. The second strategy is called CROW-ref and reduces refresh overhead by taking advantage of the copy rows for faster REF operations. The third strategy relies on remapping regular rows onto copy rows to counteract against rowhammer attacks.

3.3.1 CROW-cache The key idea behind the CROW-cache is to exploit the MRA operation to charge the bitlines from two locations. This results in an faster restoration of the sense amplifier, by using this fact the access towards specific rows can be increased.

Assuming N copy rows, N regular rows can be copied. The N copy rows will act as an cache and will store the most frequently accessed rows like a regular cache. The cache can be filled by using any operation systems cache-replacement-policy like First-In-First-Out. In case the cache-replacement-policy requires more bits, more bits need to be allocated for the CROW-table. The memory controller can now check on every access whether a row got copied. In case the CROW-table retrieves a miss, the regular row is getting copied to a copy row by using the newly introduced ACT-c operation. After the execution of the operation both rows contain the same data. Due to restoring the data to two different rows, we gain a slight overhead in restoration latency. The effect

on the entire is system is little as the cache-replacement-policy should provide a high hit-rate.

The second introduced operation ACT-t exploits the fact, that two rows containing the same data are existent in the DRAM. The key idea is to activate a copy row and regular row simultaneously to allow faster access. Assuming the to be access regular row already has an entry inside the CROW-table, we can execute and ACT-t. This results into faster charging of the bitlines and faster general activation time. By using this combination of operations the buffer retrieves the data faster and access-time can be speed up.

We gain another positive side-effect from activating two rows at once, an increased capacitance on the bitline. By exploiting this fact, we can relax the tRAS timing parameter by using a technique called partial restoration [10]. The key idea is to stop the row activation earlier than usual resulting into partially restored cells. The increased capacitance results into slower charge leak thus keeping the data until the next refresh. By combining the partial restoration technique with the previously mentioned techniques, we can achieve an even faster access time. Due to the partial charge state, those cells can only be accessed with ACT-t. The next section will touch upon this drawback. As previously mentioned, the access time can be decreased by using partially charged cells. There are two cases, where the the cells are not necessarily in that state.

First, the operations in between the PRE operation and the ACT operation surpass the time needed for full recharge.

Second, no further requests are made to the memory controller resulting in time to delay the PRE operation. In any other case, we need to store the information of the partial restoration state to prevent faulty accesses. This is accomplished by using the additional information bits inside the CROW-table. We use one bit to store the partial restoration state and call it isFullyRestored. This field is only false, if the time between ACT and PRE is below the the tRAS timing parameter. Due to the already existing timing information, the calculation of this field does not result into a significant overhead.

As already mentioned in previous sections, the copy rows act similar to a cache. The introduction of the partially charged states provides a new challenge regarding the eviction policy that needs to be addressed. Assume a copy row is evicted for a copy of another regular row, yet the copy row is not fully restored. The memory controller can access this information inside the CROW-table. To circumvent this issue, the memory controller issues an ACT-t operation and awaits the the regular tRAS timing parameter. After the passing of the regular timing parameter, the cell is fully restored. This trick results into a negligible overhead with the assumption of a high hit-rate.

3.3.2 CROW-ref CROW-ref is a mechanism to increase the refresh window by remapping weak regular rows to strong copy rows. This mechanism allows us to extend the worst-case timing parameter and reduce energy consumption. For the CROW-ref mechanism, a row retention profiler is necessary for identification of weak rows. The row retention profiler is run at start-up or during runtime

to analyze the weak and strong rows. The profiler tests all rows with specified procedures upon certain specifications to rate all rows. Works in the papers [1,7] find that most rows do not necessarily need worst-case refresh interval. This indicates, that using strong rows results into more efficient usage of refresh intervals. By using these results, we can calculate how many copy rows are neccessary to ensure a efficient usage of rows. The calculation for the probability of a weak row results from the counter probability for no weak cells.

$$P_{weak\_row} = 1 - (1 - P_{weak\_cell})^{N_{cells\_row}} \tag{1}$$

Using (1) and the binomial distribution, we determine the probability for more than n weak rows in a subarray.

$$P_{subarray\_n\_rows} = 1 - \sum_{k=0}^{n} \binom{N_{row}}{k} P_{weak\_row}^{k} * (1 - P_{weak\_row})^{N_{row}-k} \tag{2}$$

Under the assumption of $P_{weak\_cell} = 4 \cdot 10^{-9}$ and 512 rows per subarray with 8192 cells, we arrive at a probablity of $P_{subarray\_8\_rows} = 3.3 \cdot 10^{-11}$. This means, 8 copy rows will suffice for most objectives to cover most of the weak rows. By using the knowledge about weak and strong rows, we can remap all weak regular rows onto strong copy rows. The tracking of the copies happens with the CROW-table. After each remapping the CROW-table stores the regular row id in its corresponding field. At each memory access, the memory controller checks the CROW-table for a remapping. If the check is positive, the corresponding row is activated instead. By using the above-mentioned strategy, the worst-case timing parameter is extended due to strong rows having a longer lifetime. This results in less refresh periods as well as better energy usage. The profiling during runtime is necessary due to variable retention time (VRT) [9,8]. The VRT results into sporadic changes of retention time. To act against this behaviour, runtime profiling is necessary. The remapping strategy remains the same.

3.4 Row-hammer mitigation

The scaling nowadays result in many issues as bitlines and cells are positioned in close proximity. The electromagnetic fields surrounding each component may affect one another due to the distance leading to corrupted data. Such an attack is the rowhammer attack, the concept behind it is to consecutively activate a row to peturb nearby victim rows. Aside from reliability issues stemming from corrupted data, previous work also have shown that privileged access can be gained through this attack. CROW has the ability to mitigate this kind of attack by exploiting the copy mechanism. Several papers [3,11] already delved into the topic of row-hammer detection. By using CROW we can prevent a row from being corrupted. After detection, several ACT-c operations are executed to copy the adjacent victim rows to copy rows. By doing that, the content of the rows are saved, therefore negating the attack on the victim rows. For any following ACT operation, the corresponding copy row can be retrieved from the CROW-table inside the memory controller.

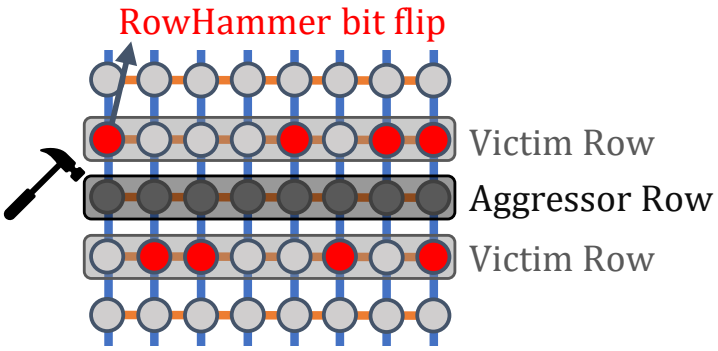

Fig. 5. Rowhammer attack [3]

4 Evaluation

Evaluation results on performance and energy-efficiency are presented in the paper [2] and will be discussed here.

4.1 CROW-cache

The evaluation for CROW-cache will be made on configurations with differing amount of copy rows. The CROW configuration will be denoted by CROW-N where N is the amount of copy rows. For the evaluation, CROW was run on several benchmark applications e.g., cactus. The quantity mentioned alongside the application refers to the misses per kilobyte instruction.

4.1.1 Single-core performance As discussed earlier, ACT-t provides an speedup due to faster access which can be seen in Fig. 6. CROW-1 already shows an average speedup compared to traditional DRAM of 5.5%, CROW-8 and CROW-256 respectively 7.1% and 7.8%. This improvement is also coupled with an more efficient energy-usage as more operations can be executed as result from the speedup. These numbers are based under the assumption of the Most Recently Used eviction policy. By just using this eviction policy, we can already see a high hit rate. This is proportional to the speedup as can be seen.

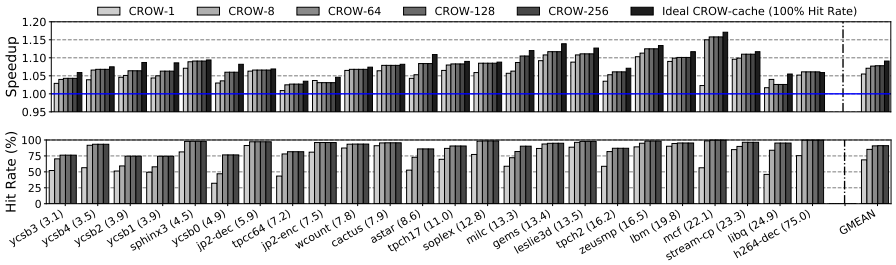

Fig. 6. Single-core performance [2]

4.1.2 Multi-core performance In Fig. 7 we can see the evaluations results for a four-core system. The red bars show the performance fluctuation between each core. The workload of each core is denoted by High (H), Medium (M) and Low (L). As can be seen, CROW-1 provides, on average, a 1.1% speedup, while CROW-8 and CROW-256 provide a 3.7% and 4.9% speedup, respectively. We can see, CROW-1 does not provide a great speedup boost in case of a multi-core performance system. Due to multiple cores accessing the same subarray, the copy row gets frequently replaced resulting into an inefficient usage of the CROW. In contrast to single-core performance, multi-core performance still achieves a significant speedup compared to the regular performance.

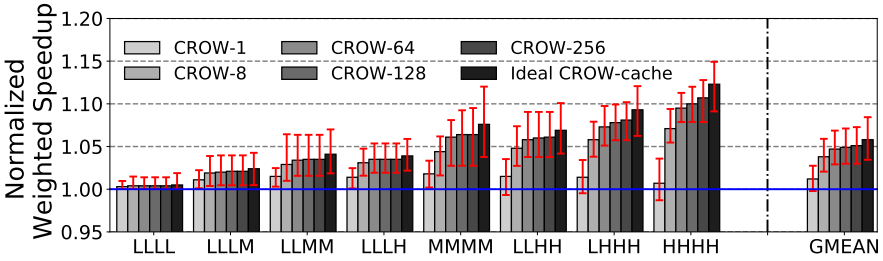

Fig. 7. Four-core performance [2]

4.2 Energy-efficiency

ACT-t and ACT-c are elementary operations for the CROW improvement strategies. Even though compared to ACT, both operations consume more energy on average, CROW-cache amortizes the energy-usage in general. This lies in more efficient usage of energy by executing operations faster. On average, the energy usage lies around 8.2% and 6.9% for single-core systems and multi-core systems respectively. It should be noted, the more operations are used, the more the energy-efficiency rises due to amortization over all operations

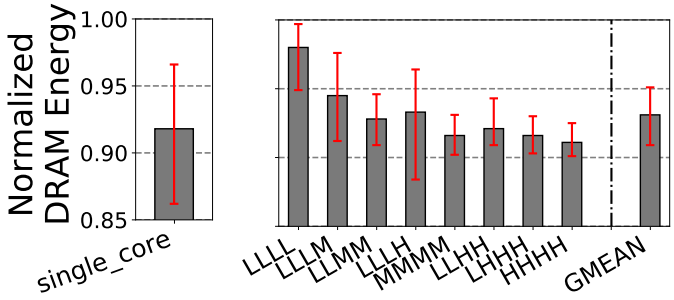

Fig. 8. CROW-cache energy consumption [2]

4.3 CROW-ref

For the CROW-ref evaluation, the refresh interval was extended to 128ms instead of the usual 64ms. CROW-ref provides benefits in the aspects performance and energy-efficiency. The interval extension results into less refreshes per cycle thus less energy used up for refreshes. As a result of less refreshes per cycle, other operations can be executed instead. Therefore a speedup in performance can be seen as well. It has to be noted, less rows are available for regular CROW-cache in case of using both strategies at once. The observation results for average energy consumption can be seen in Fig. 9.

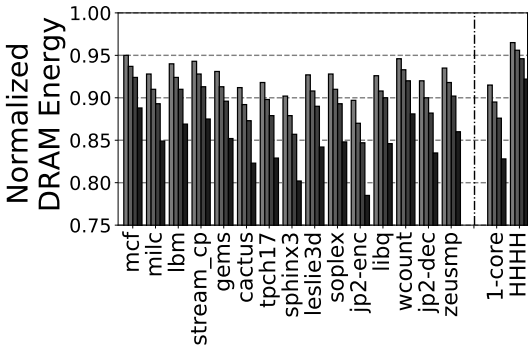

Fig. 9. CROW-ref energy consumption [2]

4.4 CROW-cache and CROW-ref

Furthermore, observations made in the paper [2] indicate a synergy in combining both mechanisms. CROW-ref does not necessarily use all copy rows and thus can be used for CROW-cache instead. This is due to the low probability of weak rows, resulting in free copy rows that can be used by CROW-cache instead. Only one extra bit is necessary to store which mechanisms is used for the copy row inside the CROW-table. Fig. 10 shows the comparisons of the two mechanisms separately as well as combined together with regard to the DRAM capacity for a single-core. We can identify, the benefits of the combination of both mechanisms is bigger than the sum of both mechanisms separately. Regarding the performance improvement, they are a result of the improved REF operations per cycle. Due to less REF operations, more operations can be executed in between, increasing the throughput of work-related operations. By decreasing the amount of REF operations, we can also see the increase that we would have gotten from CROW-ref independently. In total, we can mitigate the bottleneck by combining both mechanisms.

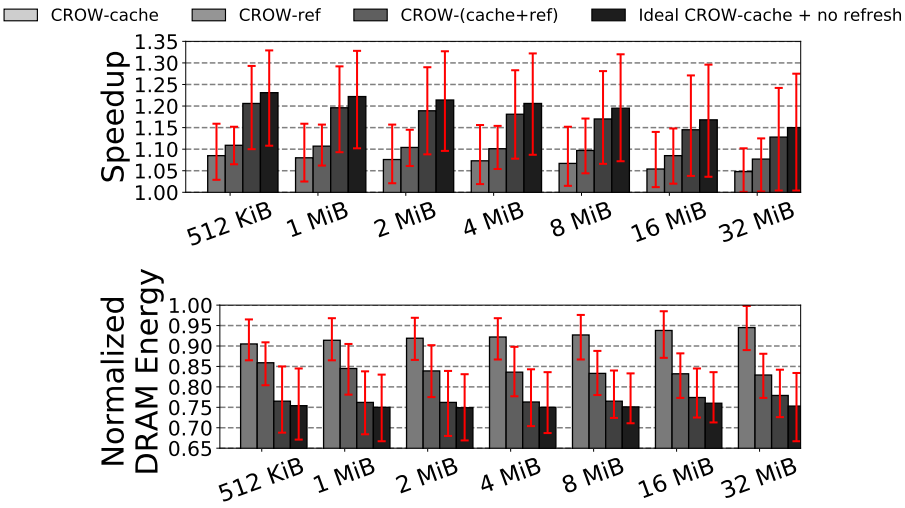

Fig. 10. CROW comparisons [2]

5 Conclusion

The paper [2] proposes a new structure called CROW that partitions the rows inside a DRAM into two types of rows which can be activated independently. By exploiting this fact, CROW introduced three strategies to improve upon performance, energy-efficiency as well as reliability. CROW-cache uses the copy rows as a kind of cache to achieve faster activation-time. CROW-ref profiles the rows of the DRAM to remap weak regular rows to strong copy rows. This achieved a reduction of refresh operation per cycle. CROW is also able to prevent rowhammer attacks by copying the victim rows into copy rows. By deploying all three mechanism improvement can be made on all respective fields.

References

1. Choi, H., Hong, D., Lee, J., Yoo, S.: Reducing dram refresh power consumption by runtime profiling of retention time and dual-row activation. Microprocessors and Microsystems **72**, 102942 (2020). https://doi.org/https://doi.org/10.1016/j.micpro.2019.102942
2. Hassan, H.: Improving dram performance, reliability, and security by rigorously understanding intrinsic dram operation (2023)
3. Hassan, H., Tugrul, Y.C., Kim, J.S., van der Veen, V., Razavi, K., Mutlu, O.: Uncovering in-dram rowhammer protection mechanisms: A new methodology, custom rowhammer patterns, and implications (2022)
4. Kim, J.S.: Improving dram performance, security, and reliability by understanding and exploiting dram timing parameter margins (2021)
5. Luo, H., Olgun, A., Yağlıkçı, A.G., Tuğrul, Y.C., Rhyner, S., Cavlak, M.B., Lindegger, J., Sadrosadati, M., Mutlu, O.: Rowpress: Amplifying read disturbance in modern dram chips (2023)
6. Olgun, A., Bostanci, F.N., Oliveira, G.F., Tugrul, Y.C., Bera, R., Yaglikci, A.G., Hassan, H., Ergin, O., Mutlu, O.: Sectored dram: An energy-efficient high-throughput and practical fine-grained dram architecture (2022)
7. Patel, M., Kim, J.S., Mutlu, O.: The reach profiler (reaper): Enabling the mitigation of dram retention failures via profiling at aggressive conditions. In: 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA). pp. 255–268 (2017). https://doi.org/10.1145/3079856.3080242
8. Qureshi, M.K., Kim, D.H., Khan, S., Nair, P.J., Mutlu, O.: Avatar: A variable-retention-time (vrt) aware refresh for dram systems. In: 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. pp. 427–437 (2015). https://doi.org/10.1109/DSN.2015.58
9. Restle, Park, Lloyd: Dram variable retention time. In: 1992 International Technical Digest on Electron Devices Meeting. pp. 807–810 (1992). https://doi.org/10.1109/IEDM.1992.307481
10. Wang, Y., Tavakkol, A., Orosa, L., Ghose, S., Mansouri Ghiasi, N., Patel, M., Kim, J.S., Hassan, H., Sadrosadati, M., Mutlu, O.: Reducing dram latency via charge-level-aware look-ahead partial restoration. In: 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). pp. 298–311 (2018). https://doi.org/10.1109/MICRO.2018.00032
11. Zhou, R., Liu, J., Ahmed, S., Kochar, N., Rakin, A.S., Angizi, S.: Threshold breaker: Can counter-based rowhammer prevention mechanisms truly safeguard dram? (2023)

Mobile aktive Exoskelette in der Industrie

Cornelius Niklas Breitschwerdt

Karlsruher Institut für Technologie, 76131 Karlsruhe, Germany
cornelius.breitschwerdt@student.kit.edu

Zusammenfassung. Es werden Motivation hinter der Nutzung von Exoskeletten behandelt und ihre Anwendungsbereiche klassifiziert. Eine Reihe an Systemen wird vorgestellt und stellvertretend für ihre Klassen verglichen.

Schlüsselwörter: Mobil · Aktiv · Industrie · Exsokelett

1 Einführung

Exoskelette sollen Arbeitsabläufe beschleunigen und menschliches Versagen verhindern [24]. Sie ermöglichen es ihren Trägern, Lasten von bis zu 90 kg mit weniger Kraftaufwand und geringerer Anstrengung zu heben [33]. Ununterbrochene Produktion und weniger Ausfälle werden aufgrund von geringerer Erschöpfung und erhöhter Sicherheit erreichbar.

In dieser Arbeit wird vorgestellt, was ein Exoskelett bewirken kann, welche Arten von Exoskeletten es gibt und wie sie in der Industrie eingesetzt werden. Zudem werden mehrere Exoskelette vorgestellt und anschließend einige ihrer Eigenschaften, unter anderem Kraft, Gewicht und Aktivität, verglichen.

Es wird eine Einschränkung auf aktive, also durch z. B. Elektrizität oder Luftdruck betriebene, Exoskelette gemacht.

2 Motivation

Dieses Kapitel wird einen kurzen Überblick über die verschiedenen Gründe für die Nutzung von Exoskeletten in der Industrie geben.

2.1 Muskel- und Skeletterkrankungen – MSE

Muskel- und Skeletterkrankungen (MSE) umfassen Schädigungen und Störungen der Gelenke und des Gewebes. Am häufigsten betroffen sind der oberere Rücken und Nacken, andere Bereiche wie untere Gliedmaßen sind jedoch nicht ausgeschlossen [11]. MSE entstehen über einen längeren Zeitraum durch wiederholte Bewegungen und Vibrationen, schlechte Haltung und schwere körperliche Arbeit [12].

MSE können zu intensiven Schmerzen und Bewegungseinschränkungen führen. Dies kann tägliche Aufgaben erschweren oder unmöglich machen [23] und

somit zu einem Fernbleiben von der Arbeit führen. Für Firmen bedeutet dies einen finanziellen Rückfall. In Deutschland tragen Unternehmen Kosten von bis zu 28.7 Milliarden Euro pro Jahr für Krankheitstage, die mit MSE verknüpft sind [28].

Besonders betroffene Industrien sind die Land- und Forstwirtschaft, Bau- und Abbruchindustrie, das Gesundheitswesen und die Transportindustrie [6].

Die Nutzung von Exoskeletten kann die Verletzungsrate stark reduzieren und die Lasten auf den Körper minimieren. Dies führt zu besseren Arbeitsbedingungen und ermöglicht ein längeres und erfüllteres Berufsleben [6].

2.2 Muskelermüdung

Bei längeren Arbeiten ist Muskelermüdung ein wichtiger und gefährlicher Faktor. Sowohl Präzision als auch Kraft können hiervon reduziert werden und somit ein Unfallrisiko erzeugen.

Exoskelette können hier, vor allem bei Lastentransport, Werkzeugnutzung, Überkopfarbeit, gebückten und hockenden Arbeiten, dem Träger helfen über einen längeren Zeitraum seine Arbeit zu verrichten und zusätzlich die Verletzungsgefahr zu reduzieren [7]. Zudem kann sich der Komfort des Trägers durch die Nutzung eines Exoskeletts während dem Arbeiten steigern, indem es eine gewisse Menge an Gewicht trägt oder eine bequemere Arbeitsposition ermöglicht.

2.3 Leistungssteigerung

Exoskelette sind darauf ausgelegt, die körperlichen Fähigkeiten von Nutzern und damit auch ihre Arbeitskapazität zu erweitern. Dies hilft Firmen in der Balance zwischen Arbeitsaufträgen und Arbeiterkapazität [4].

Effizientere Bearbeitung von Aufgaben Diese Erweiterung der Fähigkeiten kann sich in Form von längerer Arbeitszeit ohne Pause, qualitativ besserer Arbeitsleistung [7] oder in weniger krankheitsbedingten Fehltagen zeigen. Letzteres ist Resultat von weniger biomechanischer Last auf den Körper [7]. Ein weiterer Punkt, der im Angesicht einer alternden Arbeiterschaft immer wichtiger wird, ist die technische Unterstützung, um die Fähigkeiten älterer Arbeiter zu erhalten [22].

Erweiterung der menschlichen Fähigkeiten Eine andere Sicht auf die Produktivitätssteigerung ist die Erweiterung von Fähigkeiten um Eigenschaften, die ohne Exoskelette für einen Menschen nicht möglich sind.

Eine dieser Möglichkeiten ist die Kraftsteigerung, welche besonders in der Logistik Anwendung findet. Hier wird das Exoskelett genutzt, um dem Träger mehr Kraft zu verleihen [10], wenn möglich ohne die Präzision zu beeinträchtigen.

Sicherheitssteigerung Obwohl Exoskelette nicht als traditionelle persönliche Schutzausrüstung bezeichnet werden, ist einer ihrer größten Motivationspunkte an industriellen Arbeitsplätzen die Verletzungsvorbeugung [7] und Unfallrisikominimierung, sowie die Minimierung der damit verbundenen Kosten [6].

3 Anwendungsbereiche in der Industrie

In diesem Kapitel geht es speziell um die Bereiche der Industrie, in welchen Arbeiter besonders von Exoskeletten profitieren können. Diese werden im Folgenden klassifiziert und mit Beispielen versehen.

Es wird in drei Klassen unterteilt [38]. Die Kriterien, anhand welchen unterschieden wird, sind Stärke und Art der physikalischen Unterstützung. Diese Klassen werden später zur besseren Einordnung von Exoskeletten genutzt.

K1 Ungewöhnliche Arbeitspositionen und Bewegungen Diese Klasse beinhaltet Arbeiten, bei denen ungewöhnliche und unbequeme Positionen oder Bewegungen gehalten bzw. wiederholt werden. Hierzu zählen Überkopfarbeiten, welche das Halten der Arme über den Schultern bzw. dem Kopf verlangen, aber auch gebückte oder hockende Arbeiten. Beispiele hierfür sind Montage, Schweißen oder Fließbandarbeit in der Automobilindustrie [38].

K2 Manipulation schwerer Lasten Diese Klasse beinhaltet Lastenarbeit. Speziell Manipulationen von Massen über 4 kg, welche nur gegen die Gravitation gehoben bzw. gehalten werden. Beispiele hierfür sind die Logistik, in welcher Arbeiter Pakete verschiedenster Massen über variierende Distanzen bewegen, oder die Automobilindustrie, mit der Montage von z. B. einem Kabelbaum [38].

K3 Unterstützung beim Montageaufwand Diese Klasse beinhaltet die Manipulation von Massen in alle Richtungen. Statt dem Kraftaufwand gegen die Gravitation spielen hier Kräfte in z. B. horizontale Richtung die Hauptrolle. Beispiele hierfür sind Fließbandarbeiten oder das Installieren von hermetischen Dichtungen in der Automobilindustrie.

Aufgrund von zunehmender Komplexität im Design und Einschränkungen in der Masse, gibt es nur eine kleine Menge an Exoskeletten, die multidirektional Kraft aufwenden können und somit für diesen Anwendungsfall gebaut sind [38].

Grenzen dieser Klassifizierung Kfz-Mechatroniker/innen [31] sind eindeutig **K1** zuzuordnen, Logistik, Bau und Montage gehören **K2** an. Die Automobilindustrie oder Metallindustrie, so wie auch viele andere Berufe, gehören jedoch mehreren Klassen an.

Die Grenzen in der Klassifizierung stehen also nicht endgültig fest und sind nur als Hilfestellung zur besseren Einordnung und zu einem besseren Vergleich von Exoskeletten gedacht.

4 Definitionen

4.1 Exoskelett

Ein Exoskelett ist ein Mensch-Roboter-System, welches Muskelkraft- und Ausdauersteigerungen in verschiedensten Umgebungen für den Nutzer bringt, während dieser die Aufgaben der Position, Steuerung und Wahrnehmung übernimmt. Die mechanischen Teile sind hierbei am Nutzer angebracht [21].

4.2 Mobilität

Das Gewicht eines Exoskeletts ist ein wichtiger Faktor der Mobilität. Ist es zu schwer, schränkt es den Nutzer in seinen Bewegungen, seiner Geschwindigkeit und seiner Reaktionszeit ein. Es ist zudem wichtig, wie das Gewicht gelagert ist, also ob es vom Exoskelett selbst getragen wird oder auf z. B. den Hüften des Nutzers lastet. Dies kann Auswirkungen auf die Ausdauer des Nutzers haben und somit die Reichweite und Tragedauer beeinträchtigen.

4.3 Aktivitätsgrad

Der Aktivitätsgrad eines Exoskeletts wird durch die Art von Energie, die es nutzt, klassifiziert [13]. Hierbei wird zwischen externer und selbst generierter Energie unterschieden.

Aktiv Aktive Exoskelette nutzen einen oder mehrere Antriebsmechanismen, bzw. Aktuatoren, welche externe Energiequellen benötigen. Beispiele hierfür sind hydraulische und pneumatische Zylinder oder elektrische Motoren [4].

Aktive Exoskelette sind z. B. dazu geeignet schwere Massen, manche bis zu 100 kg, zu manipulieren. Sie sind jedoch durch die zusätzliche Hardware, wie z. B. Motoren, Kabel und Batterie, meist schwerer und teurer als ihre passiven Gegenstücke [4].

Passiv Passive Exoskelette nutzen Mechanismen, welche Energie vom Nutzer speichern und diese zu einem späteren Zeitpunkt effektiv freigeben [4]. Beispiele hierfür sind elastische Elemente wie z. B. Federn [13], [4]. Gelenke ohne eine Antriebsmethode zählen auch hierzu. Diese haben meist eine stabilisierende Aufgabe und dienen zum Schutz des Nutzers.

Passive Exoskelette sind z. B. gut geeignet, um Werkzeuge über einen längeren Zeitraum auf einer Höhe zu halten [13] und sind aufgrund ihrer Unabhängigkeit von aktiven Energiequellen flexibel einsetzbar.

Quasi-Aktiv Quasi-aktive Exoskelette nutzen eine Mischung aus aktiven und passiven Antriebsmechanismen, hier sind vor allem einzelne Gelenke bzw. Aktuatoren gemeint. Diese quasi-aktiven Gelenke bestehen aus einem aktiven Teil, welcher für eine gewisse Anzahl von Freiheitsgraden zuständig ist. Zum anderen bestehen sie aus einem passiven Teil, welcher für die restlichen spezifizierten Freiheitsgrade zuständig ist.

Quasi-aktive Exoskelette sind weniger komplex und leichter als aktive Exoskelette, was sie weniger fehleranfällig und komfortabler macht. Dennoch haben sie die Vorteile von aktiver Krafterzeugung auf den relevanten Freiheitsgraden. Sie werden z. B. für Hebe- und Laufaufgaben mit Gewicht verwenden [19].

4.4 Energieversorgung

Aktive und quasi-aktive Exoskelette benötigen einen Energiespeicher. Es ist auch möglich, dass mehrere Energiearten und damit mehrere Energiespeicher zugleich verwendet werden. Dies ist jedoch seltener, da hierdurch die Komplexität steigt. Ein solcher Speicher kann ein Akkumulator oder Drucklufttank [13] sein. Je nach Anwendungsort und -fall kann auch eine kabel- oder schlauchgebundene Energiequelle, z. B. Stromnetz oder Hallendruckluftsystem, verwendet werden [13]. Dies schränkt jedoch die Mobilität ein.

4.5 Kontrolle

Aktive Kontrolle Der Nutzer des Exoskeletts kann das Level an Unterstützung oder den Funktionsmodus während der Ausführung von Aktivitäten anpassen. Er hat also direkten Einfluss auf die Abläufe des Exoskeletts.

Passive Kontrolle Die Bewegungen des Nutzers werden von Sensoren erfasst und durch Algorithmen in Bewegungen des Exoskeletts umgewandelt. Der Nutzer hat keine Möglichkeit, während des Betriebs in diesen Ablauf einzugreifen.

4.6 Körperabdeckungsbereich

Exoskelette können anhand der Bereiche des Körpers, welche sie unterstützen, kategorisiert werden. Am häufigsten wird zwischen Oberkörper, Unterkörper und Ganzkörper unterschieden [4], es kann jedoch auch ein speziellerer Körperteil angesprochen werden.

Oberkörperbezogene Exoskelette werden in diesen wissenschaftlichen Arbeiten behandelt: [15], [36], [26]. Einige in der Industrie verwendete Exoskelette sind hier zu finden: [5], [17], [35].

Unterkörperbezogene Exoskelette werden in diesen wissenschaftlichen Arbeiten behandelt: [20], [40], [3]. Einige in der Industrie verwendete Exoskelette sind hier zu finden: [30], [37].

Ganzkörperbezogene Exoskelette werden in diesen wissenschaftlichen Arbeiten behandelt: [8], [25]. Ein in der Industrie verwendetes Exoskelett ist hier zu finden: [33].

5 Konkrete Anwendungsbeispiele

Hier werden einige Beispiele als Stellvertreter für größere Gruppen und Charakteristiken vorgestellt. Diese werden im anschließenden Kapitel verglichen.

5.1 Lucy

Dieses Exoskelett ist der wissenschaftlichen Arbeit [29] entnommen. Es wurde für diese Arbeit ausgewählt, da es die Kategorien der oberkörperbezogenen, druckluftbetriebenen und aktiv kontrollierten Exoskelette vertritt.

Motivation dieses Exoskeletts Die Hauptmotivation für Lucy ist Forschung. Ziel ist hierbei verschiedenen Ansätzen zur Industriellen Assistenz zu testen und evaluieren. Hierbei haben Effizienz, Komfort und Nutzerakzeptanz Vorrang.

Aufbau Lucy ist ein Exoskelett, welches sowohl aktive als auch passive Aktuatoren nutzt. Der aktive Teil wird durch Druckluftzylinder betrieben([39], [16], [27]). Diese sind an einen Drucklufttank oder, über eine Schlauchverbindung, an ein Druckluftsystem angeschlossen. Zweiteres führt zu einer Einschränkung der Mobilität, sowohl räumlich als auch hinsichtlich der Rotation um die eigene Achse.

Die Kraftübertragung geschieht entlang der Oberarme, des Rückens und der Hüfte. Hierbei sind die mechanischen Teile so nah wie möglich am Körper des Nutzers befestigt (Abb. 1). Die Körperabdeckung ist somit der Oberkörper.

Zudem wird Lucy aktiv gesteuert. Das bedeutet, dass der Nutzer, während er arbeitet, die Stärke der Unterstützung anpassen kann. Diese wird über Knöpfe am Griff des genutzten Werkzeugs eingestellt.

Anwendungsbereich Lucy ist der Klasse K1 (siehe K1 Ungewöhnliche Arbeitspositionen und Bewegungen) zuzuordnen. Es ist darauf ausgelegt, das Arbeiten mit Werkzeug, insbesondere das Überkopfschrauben oder -schleifen, zu erleichtern.

Industrien, in denen es verwendet werden kann, sind Automobilindustrie und Baugewerbe.

Es ist anzumerken, dass Lucy nicht kommerziell verwendet wurde. Es wurde lediglich in Studien, die einer Anwendung in der Industrie gleichen, getestet.

Fähigkeiten und Limitierungen Lucy ist in der Lage, eine beträchtliche Menge an Unterstützung zu bieten.

Eine erhöhte Geschwindigkeit im Arm, und damit ein Potenzial zur Produktivitätssteigerung, wurde gemessen. Dies bezieht sich jedoch nur auf einige Testfälle und müsste daher weiter untersucht werden.

Die aktive Kontrolle wurde von einigen Nutzern als ihr Kontrollgefühl steigernd beschrieben. Dies kann sich positiv auf die Nutzerakzeptanz auswirken.

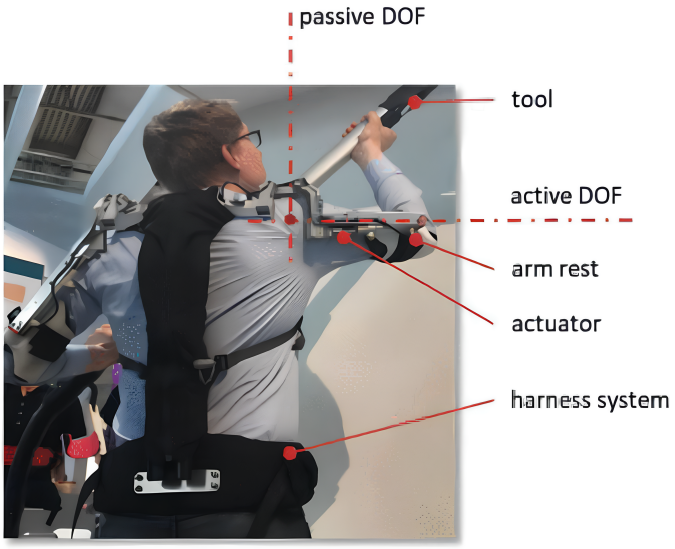

Abb. 1. Aufbau des Exoskeletts Lucy. Beschriftung der aktiven und passiven Freiheitsgrade (hochskaliert aus [29]).

Zudem wurde erfolgreich gezeigt, dass gerade für Aufgaben mit variablem Gewicht eine aktive Steuerung von Vorteil ist und dass hierfür Knöpfe am Werkzeuggriff ausreichen.

Weitere Forschung an diesem Exoskelett wird benötigt, um Auswirkungen auf Muskeln und Bewegungen des ganzen Körpers zu beurteilen.

5.2 UMExoLEA

Dieses Exoskelett ist der wissenschaftlichen Arbeit [32] entnommen. Es wurde für diese Arbeit ausgewählt, da es die Kategorien der unterkörperbezogenen und durch Elektrizität betriebenen Exoskelette vertritt.

Motivation dieses Exoskeletts Dieses Exoskelett ist dafür gedacht, manuell arbeitenden Arbeitern in ihrer Mobilität zu assistieren, indem es bei Hebe- und Trageaufgaben das Gewicht vom Unterkörper und Rücken des Nutzers übernimmt.

Zudem wird es aktuell in der Forschung genutzt, um Designauswirkungen und Steuerungselemente für Unterkörper-Exoskelette zu erforschen [32].

Aufbau UMExoLEA nutzt sowohl aktive als auch passive Aktuatoren. Es handelt sich um ein Unterkörper-abdeckendes Exoskelett. Jedes Bein besitzt 6 Freiheitsgrade: Einen aktiven und einen passiven Freiheitsgrad im Hüftgelenk, einen

aktiven und einen passiven Freiheitsgrad im Kniegelenk und 2 passive Freiheitsgrade im Knöchel. Die aktiven Aktuatoren sind durch elektrische Motoren angetrieben. Dies hat zur Folge, dass ein Rucksack, in dem sich eine Batterie befindet, getragen werden muss. Die Abb. 2 verdeutlicht den Aufbau von UMExoLEA.

Das Exoskelett wird durch jeweils 2 Gurte an den Beinen, einem Gürtel und dem Rucksack mit dem Körper des Nutzers verbunden.

Die Steuerung des Exoskeletts findet mithilfe von Winkelpotentiometern, Grenzfrequenzsensoren und einem Algorithmus, welcher die benötigte Kraft abschätzt, statt. Der Nutzer hat keinen direkten Einfluss. Somit wird UMExoLEA passiv gesteuert.

Das Gewicht liegt ohne Rucksack bei ungefähr 11 kg. Hierbei wurde auf die Wahl der Aktuatoren, des Materials, der Energieeffizienz und der Komplexität geachtet, z. B. wurde auf aktive Knöchelgelenke verzichtet.

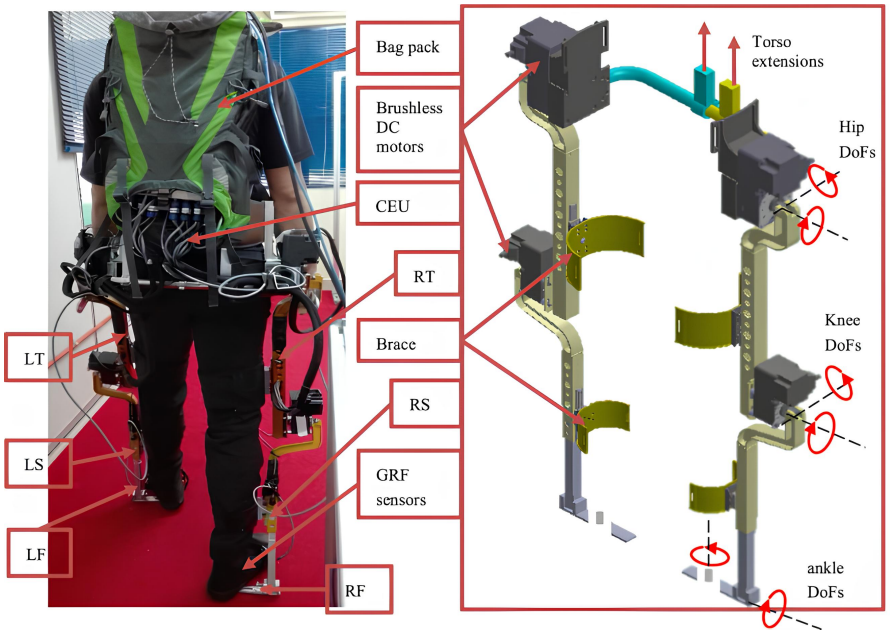

Abb. 2. Aufbau des Exoskeletts UMExoLEA. Beschriftung der 12 Freiheitsgrade (hochskaliert aus [32]).

Anwendungsbereich UMExoLEA ist der Klasse K2 (siehe K2 Manipulation schwerer Lasten) zuzuordnen. Seine Aufgaben sind Assistenz bei Hebe- und Trageaufgaben.

Die vorgesehenen Anwendungsbereiche sind Logistik und Be-/Entladen. Einige Beispielindustrien sind Produktion, Bau und Pflege.

Es ist anzumerken, dass es sich bei UMExoLEA bisher nur um einen Prototyp handelt, welcher in der Praxis nicht verwendet wurde.

Fähigkeiten und Limitierungen Das Exoskelett ist in der Lage bis zu 36% der Muskelaktivität sowohl beim Laufen als auch beim Knien zu reduzieren. Weitere Tests sind für eine Ganzkörperevaluation und genauere Messwerte nötig.

Es wird aktuell ein PC außerhalb des tragbaren Systems verwendet. Der Rucksack beinhaltet, abgesehen von der Batterie, ein Kommunikationsmodul. Somit besteht eine räumliche Abhängigkeit zu einem externen System.

Die Kraftkontrolle beruht auf einem Linearen-Quadratischen-Gaus-Drehmoment-Kontroller (linear-quadratic-gaussian-regulator), der möglichst genaue Gelenkwinkel benötigt. Diese werden allerdings nur vom Steuerungssystem abgeschätzt und sind somit nicht zuverlässig exakt. Zukünftige Arbeiten müssten die Drehmoment- und Gleichgewichtskontrolle durch einen neuen Ansatz, mit Rücksicht auf das gesamte dynamische System, ersetzen.

5.3 Body Extender

Dieses Exoskelett wurde der wissenschaftlichen Arbeit [14] entnommen. Es wurde für diese Arbeit ausgewählt, da es die Kategorien der ganzkörperbezogenen und vollständig-aktuierten Exoskelette vertritt.

Motivation dieses Exoskeletts Der Body Extender ist in erster Linie als Forschungsplattform gedacht. Es sollen der Transport und die Handhabung von Lasten bis zu 50 kg unter schlechten Bedingungen, sowie die Interaktion zwischen Mensch und Exoskelett untersucht werden.

Der praktische Hintergedanke ist das Assistieren beim Heben und Manipulieren, indem es das volle Gewicht vom Körper des Nutzers nimmt.

Aufbau Der Body Extender ist ein Ganzkörper-Exoskelett. Es besteht aus 2 Beinen mit jeweils 6 Freiheitsgraden, 2 Armen mit jeweils 4 Freiheitsgraden und 2 Parallelbackengreifern mit jeweils einem Freiheitsgrad. Alle 22 Freiheitsgrade sind aktiv und sind durch Elektromotoren angetrieben. Aus Wartungs-, Komplexitäts- und Kostengründen wurde hier für alle Gelenke der gleiche Motortyp verwendet.

Die Elektronik, Steuerung und Sensorik ist an Bord des Exoskeletts. Als Energiequelle wird ein Batteriesystem verwendet, welches sich aktuell noch getrennt auf dem Boden befindet (Abb. 3 b). Das System ermöglicht ca. 20 Stunden Betriebszeit bei einem durchschnittlichen Leistungsverbrauch von 900 Watt.

Der Nutzer ist an den Händen, Schultern, Füßen und der Hüfte mit dem Exoskelett verbunden (Abb. 3 a). Das Exoskelett trägt die 160 kg Eigengewicht selbst.

Die Steuerbefehle werden von 5 Sensoren an den Kontaktpunkten mit dem Körper des Nutzers und mithilfe von 2 Griffen für die Hände aufgenommen. Hierbei sind alle Eingaben, bis auf die Hände, passiv.

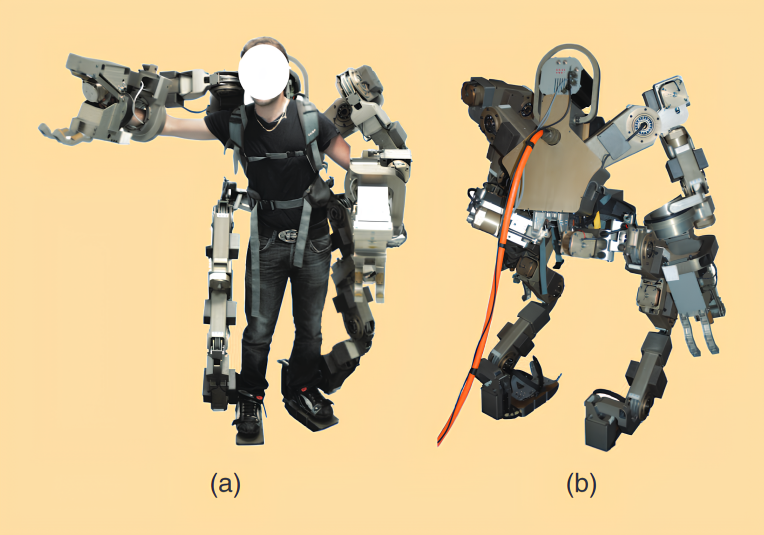

Abb. 3. Aufbau des Body Extender Exoskeletts (hochskaliert aus [14]).

Anwendungsbereich Der Body Extender ist der Klasse K2 (siehe K2 Manipulation schwerer Lasten) zuzuordnen. Er ist für die Manipulation und den Transport von schweren Massen in unbekannten Umgebungen gemacht.

Einige Beispielindustrien, in denen er verwendet werden kann, sind Logistik, Konstruktion und Flugzeug- bzw. Bootsbau. Zudem ist er für Notfall- und Bergungsarbeiten geeignet.

Fähigkeiten und Limitierungen Forschungsergebnisse haben gezeigt, dass der Body Extender bei Aufgaben wie Laufen oder Hocken bis zu 500 N Kraft aufwenden kann. Dies entspricht 50 kg pro Arm. Dabei ist eine Laufgeschwindigkeit von 0,5 m/s erreichbar. An dieser Geschwindigkeit ist zu erkennen, dass das System aktuell nicht auf Geschwindigkeit, sondern Kraft und das Heben von schweren Lasten ausgelegt ist.

Aktuell ist keine automatische Balancierung des Exoskeletts implementiert, der Nutzer muss diese selbst übernehmen. Das liegt an der vorläufigen Kontroller-Implementation. Diese setzt nur die Basisfunktionen wie Körperteilbewegungen,

Gewichtsmanipulation und Laufen. Sie verfügt jedoch über keine hohe Genauigkeit. Dies hat zu Folge, dass Nutzer eine gewisse Trainingszeit benötigen und dass alle Tests und Studien mit entsprechenden Sicherheitsmaßnahmen durchgeführt wurden. Durch zukünftige Entwicklung von angemessenen Kontrollmechanismen und Algorithmen kann dieser Mangel ausgebessert werden [14]. Für stabile Manipulation und Zusammenarbeit beider Hände ist weitere Entwicklung und Forschung an Kontrollmechanismen notwendig.

An den meisten dieser Limitierungen wird laut den Schlussfolgerungen aus [14] gearbeitet, jedoch sind noch keine Veröffentlichungen zu finden.

5.4 AGADEXO Shoulder

Dieses Exoskelett wurde von Agade (Italien) entwickelt [2]. Es wurde für diese Arbeit ausgewählt, da es die Kategorien der schulterbezogenen und kommerziell erwerblichen Exoskelette vertritt.

Motivation dieses Exoskeletts Das Exoskelett ist auf die Assistenz bei Manipulation und Transport ausgelegt. Hierdurch sollen Unterbrechungen während Arbeitsschichten vermieden und kontinuierliche Produktion erreicht werden.

Aufbau Das AGADEXO Shoulder ist ein Oberkörper-Exoskelett, das speziell für die Entlastung von Schultern gemacht ist. Agadexo steht für „Anti-Gravity Active Device for Exoskelettons“ [9].

Das Exoskelett hat 2 aktive Freiheitsgrade, welche das Heben der Arme über den Kopf unterstützen. Es werden quasi-aktive Aktuatoren benutzt, die zum Teil aus elastischen Mechanismen und zum anderen aus elektrischen Motoren bestehen. Durch diese Aktuatoren-Wahl wird weniger Strom benötigt und damit die Größe der Batterie reduziert, was das Gesamtgewicht auf 4 kg bringt.

Das Exoskelett ist aus Carbonfasern gefertigt, was zusätzlich zu dem geringen Gewicht den Tragekomfort erhöht. Es ist an der Hüfte, den Schultern und den Oberarmen mittels Gurten befestigt (Abb. 4).

Das Exoskelett ist mit kabellosen Armbändern verbunden, welche während Hebeaufgaben zur Erkennung von Lasten genutzt werden. Dies geschieht über einen Algorithmus, welcher anhand der Sensordaten der Armbänder und der Aktuatoren bestimmt, in welchem Modus sich das Exoskelett befinden soll. Somit ist das Exoskelett passiv kontrolliert. Der Modus, wenn keine Last erkannt wurde, ist Transparenz. Hier minimiert das Exoskelett seine Aktivität und spart dadurch Energie. Der Modus wechselt, sobald Lasten erkannt werden. Hier unterstützt das Exoskelett mit durchschnittlich 36 Nm Drehmoment, was ungefähr 25 kg Hebe- bzw. Haltekraft entspricht.

Anwendungsbereich Es handelt sich bei AGADEXO Shoulder um ein Exoskelett der Klasse K2 (siehe K2 Manipulation schwerer Lasten). Es ist auf Hebe-

Abb. 4. Aufbau des AGADEXO Shoulder Exoskeletts (Ausschnitt aus animierter 360° Ansicht [2]).

und Transportaufgaben ausgelegt. Einige Beispielindustrien sind Automobil, Logistik und Einzelhandel.

Fähigkeiten und Limitierungen Das Lastenerkennungssystem ermöglicht es, die Batterielaufzeit zu erhöhen, was wiederum zu einer kleineren Batterie, geringeren Kosten und höherem Tragekomfort führt. Die Batterielaufzeit beträgt bis zu 20 Stunden.

Während Materialmanipulationsaufgaben ist das Exoskelett in der Lage, automatisch das Level an Unterstützung anzupassen und damit den Nutzer optimal zu entlasten. Es kann bis zu 25 kg gegen die Schwerkraft heben und dabei bis zu 40% der Schultermuskelbeanspruchung übernehmen.

5.5 Bes-hv

Dieses Exoskelett wurde von ULS Robotics Co. (China) entwickelt [37]. Es wurde für diese Arbeit ausgewählt, da es die Kategorien der unterkörperbezogenen, kommerziell erwerblichen und aktiv kontrollierten Exoskelette vertritt.

Motivation dieses Exoskeletts Bes-HV wurde entwickelt, um die Produktivität und Effizienz zu steigern, sowie die Arbeitsintensität zu verringern.

Aufbau Das Bes-HV ist ein Unterkörper-Exoskelett, welches speziell für die Unterstützung der Hüfte und des Rückens gebaut ist. Es ist in 2 Freiheitsgraden an der Hüfte aktiv angetrieben, einem für jedes Bein. Hier werden elektrische Motoren genutzt. Das Exoskelett ist an den Beinen, den Schultern (Abb. 5) und der Hüfte mittels Gurten befestigt, der Nutzer trägt hierbei das Gewicht hauptsächlich am Hüftgurt. Das Exoskelett wiegt 5,6 kg.

Es handelt sich um aktive Kontrolle, der Nutzer hat eine kleine Kontrolleinheit am rechten Schultergurt. Über diese lässt sich der Modus umstellen, es gibt „Gleichgewicht halten“, „Helfen/Heben“ und „Gehen“. Zudem kann die Stärke der Assistenz eingestellt werden.

Es gibt eine 4G/5G-Schnittstelle, über welche drahtlos Echtzeitinformationen ausgelesen werden können.

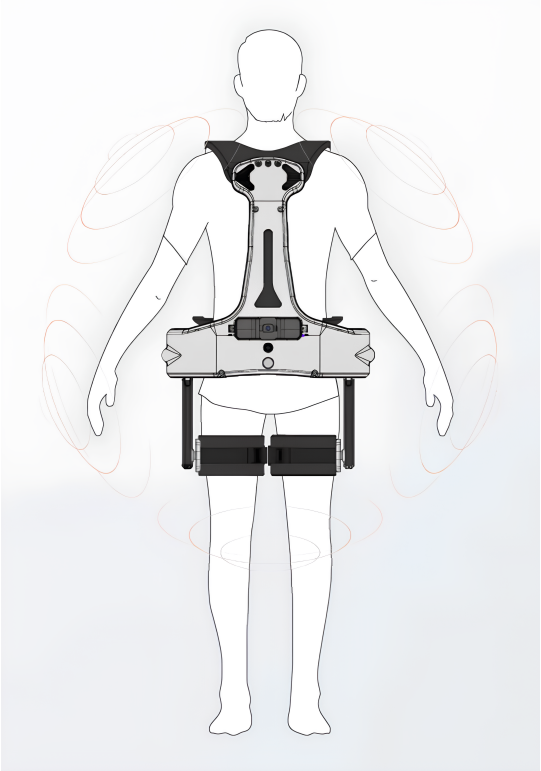

Abb. 5. Abbild des BES-HV Exoskeletts aus der Perspektive von Hinten (hochskalierter Ausschnitt aus [37]).

Anwendungsbereich Es ist der Klasse K2 (siehe K2 Manipulation schwerer Lasten) zuzuordnen. Es wird für das Heben und den Transport von Lasten verwendet. Einige Beispielindustrien sind Logistik, Montage und Handwerk.

Fähigkeiten und Limitierungen Bes-HV ist in der Lage, seinen Nutzer zu unterstützen, sodass dieser bis zu 30 kg mehr heben kann. Es reduziert dabei bis zu 60% der Last auf den unteren Rücken. Der Laufmodus ermöglicht eine Geschwindigkeit von 7,5 km/h. Die Akkulaufzeit beträgt 5-8 Stunden, je nach Betriebsmodus und Verwendung.

5.6 Guardian XO

Dieses Exoskelett wurde von der Sarcos Technology and Robotics Corporation (USA) entwickelt [33]. Es wurde für diese Arbeit ausgewählt, da es die Kategorien der ganzkörperbezogenen, kommerziell erwerblichen und vollständig-aktuierten Exoskelette vertritt.

Motivation dieses Exoskeletts Der Guardian verstärkt die Kraft des Nutzers, ohne seine Mobilität einzuschränken. Hierdurch sollen die Produktivität und die Sicherheit gesteigert werden.

Aufbau Der Guardian XO ist ein aktives Ganzkörper-Exoskelett. Es besitzt 24 Freiheitsgrade, alle sind durch elektrische Motoren angetrieben und somit aktiv. Hierzu wird ein schnell austauschbares Batteriesystem verwendet, was eine nahezu durchgängige Laufzeit ermöglicht. Die Steuerung wird durch 6 Sensoren an den Händen, Füßen, Rücken und Becken ermöglicht [18]. Zusätzlich kann das Level der Unterstützung durch einen kleinen Bildschirm beeinflusst werden [1]. Es handelt sich jedoch um passive Kontrolle, da das Kontrollsystem auf Bewegungen des Nutzers reagiert und das Exoskelett entsprechen bewegt.

Die Greifer bzw. Endeffektoren sind austauschbar und dadurch vom Nutzer für den gewünschten Anwendungsfall auswählbar. Zudem können die Arme in einer Position festgestellt werden, sodass der Nutzer seine Hände für Feinarbeiten verwenden kann. Dies ist in Abb. 6 zu sehen.

Das Exoskelett wiegt in etwa 70 kg, trägt sein Gewicht aber selbst [1]. Der Nutzer ist an Händen, Füßen, Hüfte und Oberkörper mit der Struktur verbunden.

Der Preis liegt bei ungefähr 100.000$ pro Jahr. Dies beinhaltet einen von Sarcos gestellten Techniker, der die Exoskelette betreut [1].

Anwendungsbereich Es handelt sich um den Anwendungsfall der Klasse K2 (siehe K2 Manipulation schwerer Lasten), wobei auch, mit dem richtigen Greifer, die Klasse K3 (siehe K3 Unterstützung beim Montageaufwand) argumentiert

Abb. 6. Abbildung des Guardian XO aus 4 Perspektiven (Ausschnitt aus Werbebroschüre [34]).

werden kann. Aufgaben können dabei von Manufaktur, über Außendienst bis zu Logistik reichen.

Einige Beispielindustrien sind Fertigung, Automobil, See- und Luftfahrt. Zudem kann dieses Exoskelett auch in militärischen Anwendungen genutzt werden.

Fähigkeiten und Limitierungen Der Guardian ermöglicht es einer einzelnen Person, bis zu 90 kg ohne die entsprechende Müdigkeit oder Anstrengung zu heben und damit die Sicherheit und Produktivität zu steigern. Dies entspricht einer Kraftsteigerung von bis zu 2000%, wobei 100% der Last auf die Exoskelettstruktur übertragen werden.

Der Freihand-Modus ermöglicht es dem Nutzer, die Exoskelettarme unabhängig von deren Belastung festzustellen und seine eigenen Arme frei zu nutzen.

Sollte der Guardian unerwartet einen Energieversorgungsausfall erfahren, so ist die Hardware auf passives Bremsen und einen sicheren Stopp ausgelegt.

Die Reaktionsgeschwindigkeit des Kontrollsystems ist hoch genug, dass der Nutzer sein Gleichgewicht selber halten kann und kein Gleichgewicht-Kontrollalgorithmus benötigt wird [1].

Guardian ermöglicht die Nutzung von zusätzlicher Schutzausrüstung.

In Tests wurde festgestellt, dass Nutzer bis zu 50 kg Gewicht tragen konnten, es sich jedoch nur wie ungefähr 6 kg anfühlt [18].

6 Vergleich/Gegenüberstellung

Im vorigen Kapitel wurde sechs Exoskelette, welche verschiedene Kriterien und Gruppen vertreten, vorgestellt. Hier werden diese Exoskelette stellvertretend für ihre Gruppen verglichen.

Die Tabelle 1 zeigt einige wichtige Kriterien der Exoskelette in direktem Vergleich.

Exo / Kriterium	Lucy	UMExoLEA	Body Extender	Agadexo Shoulder	BES-HV	Guardian XO
Preis	-	-	-	-	-	100.000$ pro Jahr
Energieart	Pneumatisch	Elektrisch	Elektrisch	Elektrisch	Elektrisch	Elektrisch
Körper-abdeckung	Oberkörper	Unterkörper	Ganzkörper	Schulter	Hüfte/ Rücken	Ganzkörper
Mobilitätslevel	Schlauch-gebunden	Batterie-Rucksack	Externe Batterie	integrierte Batterie	integrierte Batterie	schnell aus-tauschbare Batterie
Aktivitätslevel in Freiheitsgraden	2 aktiv, 2 Passiv	4 aktiv, 8 passiv	22 aktiv	2 quasi-aktiv, 2 passiv	2 aktiv	24 aktiv
Anwendungs-bereich	K1	K2	K2	K2	K2	K2 mit Ten-denz zu K3
Motivation	Forschung an indus-trieller Assistenz	Hebe- und Trageaufga-ben, Unterkörper und Rücken entlasten	Forschung an Mensch-Exoskelett Interaktion, Transport und Hand-habung von Lasten	Manipula-tion und Transport von Lasten, kontinuier-liche Pro-duktion	Produkti-vität, Effizienz, Arbeits-intensität verringern	Kraft-steigerung, Produkti-vität, Sicherheit
Kraftaufwand Reduktion (des Nutzers)	-	36% der Muskelakti-vität	-	40% Schul-termuskel-beanspru-chung	60% der Last auf unteren Rücken	100% der Lasst
Kraft des Exoskeletts	-	-	50 kg pro Arm	25 kg	30 kg mehr heben	90 kg
Gewicht	-	11 kg (ohne Batterie)	160 kg	4 kg	5,6 kg	70 kg
Reifegrad	Forschung, keine Praxis	Forschung mit Praxis-Bezug	erster Prototyp	Kommerziell	Kommerziell	Kommerziell

Tabelle 1. Vergleich der vorgestellten sechs Exoskelette.
„-“ bedeutet, dass keine Informationen vorliegen.

6.1 Vergleich der Energiearten

Die Energieart und die daraus folgende Energieversorgung sind wichtige Faktoren in der Planung und im Aufbau eines aktiven Exoskeletts.

Besonders ist hierbei die Effizienz relevant. Energie muss möglichst effizient transportiert und am Ziel in meist kinetische Energie umgewandelt werden. Darum haben sich bei aktiven Systemen vor allem Elektrizität (UMExoLEA, Body Extender, AGADEXO Shoulder, Bes-hv, Guardian XO) und Druckluft (Lucy) durchgesetzt.

Zudem ist Zugänglichkeit ein wichtiges Kriterium der Energiewahl, hiermit ist die Speicherung der Energie gemeint. Der Energiespeicher muss immer mit dem Exoskelett verbunden und somit in dessen Nähe sein. Eine Batterie eignet sich hierfür, da sie relativ kompakt eine große Menge an Energie speichern kann. Ein Drucklufttank hingegen kann Energie nicht so kompakt speichern, was zu einem größeren Tank oder einer geringeren Laufzeit führt.

Schlauch-gebundene Systeme (Lucy, Body Extender) sind hiervon ausgenommen. Hier ist relevant, welche Energieart in der näheren Umgebung vertreten ist. Elektrizität ist dabei häufiger zu finden, Druckluft Infrastruktur muss meist extra gebaut werden oder ist nur an bestimmten Orten wie Fabrikgebäuden zu finden. Zudem haben schlauchgebundene Systeme eine eingeschränkte Mobilität, sowohl was Distanz zum Anschluss, als auch Rotationsfreiheit um die eigene Achse angeht.

Wird ein quasi-aktiver Krafterzeuger genutzt (AGADEXO Shoulder), so ist die Energieart des aktiven Teils noch immer relevant, es wird durch den passiven Teil lediglich eine Teillast übernommen und umverteilt.

6.2 Kraft des Exoskeletts in Relation zur Körperabdeckung

Die Körperabdeckung ist für ein Exoskelett ausschlaggebend. Sie bestimmt den Betrag an Kraft, welche das Exoskelett aufbringen und damit seinem Nutzer abnehmen kann. Hierbei lassen sich Exoskelette in 2 Gruppen aufteilen:

Zum einen Exoskelette, die am Nutzer befestigt sind und von ihm getragen werden (Lucy, AGADEXO Shoulder, Bes-hv). Diese assistieren dem Nutzer, indem sie das Gewicht, welches zu tragen ist, umverteilen. Die Limitierung ist meist, wie viel der Nutzer komfortabel mit den Beinen heben kann. Ein Beispiel für ein solches System ist AGADEXO Shoulder, welches das Gewicht von den Armen auf den Rücken umverteilt und somit die Oberarme und Teile des oberen Rückens entlastet.

Dem gegenüber stehen Exoskelette, welche ihr Gewicht selbst tragen und in welche der Nutzer meist hinein geschnallt wird (UMExoLEA, Body Extender, Guardian XO). Diese ermöglichen es, größere Lasten zu heben, da der Nutzer nicht mehr das Limit festsetzt. Er steuert lediglich das Exoskelett, trägt aber selbst kaum Gewicht. Die neue Limitierung ist die Kraft, welche durch die Aktuatoren erzeugt werden kann. Diese kann theoretisch errechnet werden und dient somit als Sicherheitsrichtlinie und meist auch als softwarelimitierter Maximalwert.

6.3 Aktivitätslevel in Relation zum Gewicht

Eine interessante Gegenüberstellung ist das Aktivitätslevel, welches hier über die Anzahl von aktiven Freiheitsgraden gemessen wird, und das Gewicht eines Exoskeletts. Das Gewicht steigt, je mehr Freiheitsgrade das Exoskelett implementiert und vor allem je mehr Freiheitsgrade es aktiv unterstützt. Dies lässt sich wie folgt erklären: Jeder Freiheitsgrad benötigt Hardware und jeder aktive Freiheitsgrad zusätzlich einen Krafterzeuger. Dies sieht man im Vergleich von Bes-hv, mit 2 Freiheitsgraden und 5,6 kg, zu UMExoLEA, mit 4 aktiven Freiheitsgraden und 11 kg. Die passiven Freiheitsgrade sind hier vernachlässigbar, da sie nicht viel mehr als die umliegende Struktur wiegen.

Speziell zu betrachten sind quasi-aktive Krafterzeuger. Diese ermöglichen einen geringeren aktiven Erzeugeranteil und somit eine Gewichtsreduktion, da weniger Hardware benötigt wird. Dies ist im Vergleich zwischen Bes-hv, welches 2 Freiheitsgrade aktiv unterstützt und 5,6 kg wiegt, und AGADEXO Shoulder, welches 2 Freiheitsgrade quasi-aktiv unterstützt und 4 kg wiegt, gut zu erkennen.

Implementationsdetails, Energieart und Körperabdeckungsbereich beeinflussen diese Gegenüberstellung, somit sind diese Erkenntnisse nicht universell anwendbar, können aber unter Vorsicht als Richtlinie verwendet werden.

6.4 Anwendungsbereich

Ein Exoskelett kann nach seinem Anwendungsbereich kategorisiert werden. Hierzu wurden im Kapitel 3 speziell drei Klassen definiert. Anhand dieser kann man Exoskelette ebenfalls vergleichen. Der Vergleich ist jedoch nicht ausreichend genau, da die Menge an Anwendungsbereichen so groß ist, dass eine Einteilung in drei Klassen nicht ausreicht. Für einen genaueren Vergleich müsste man diese 3 Klassen weiter auftrennen, z. B. die Klasse K2 (siehe K2 Manipulation schwerer Lasten) nochmals anhand des Betrags der erzeugten Kraft oder der Körperabdeckung spalten. Dies führt jedoch zu einer höheren Komplexität, welche in dieser Arbeit durch die getrennte Betrachtung dieser zusätzlichen Aspekte umgangen wurde. Von der Klassenzuordnung zweier Exoskelette kann man nicht auf Ähnlichkeiten in deren Aufbauten oder Industriezweigen schließen. Ein Beispiel hierfür sind AGADEXO Shoulder und Guardian XO, diese unterscheiden sich in ihrem Aufbau sehr. Dennoch hilft diese Klassenzuordnung beim groben Einordnen und Unterscheiden.

6.5 Kosten vs. Nutzen

Für Unternehmen und Firmen besonders relevant ist die Kosten-Nutzen-Abschätzung.

Der Nutzen von Exoskeletten wurde bereits in Kapitel 2 behandelt.

Die Kosten haben eine große Spanne zwischen verschiedenen Exoskeletten. Faktoren in der Preisrechnung sind Körperabdeckung, Energieversorgung, Kraft des Exoskeletts, Aktivitätslevel und noch viele weitere. Ein Teil der hier behandelten Exoskelette werden in der Forschung entwickelt und genutzt (Lucy, UMExoLEA, Body Extender), diesen kann man keinen Preis zuordnen.

7 Fazit

Diese Arbeit zeigt eine Vielzahl an Anwendungsbereichen und Industrien auf, welche von Exoskeletten profitieren können.

Es wurden verschiedene Arten von aktiven Exoskeletten vorgestellt und anhand einiger Beispiele verglichen.

Exoskelette wurden überwiegend positiv dargestellt. Es gibt jedoch auch einige Herausforderungen, die noch zu überwinden sind.

Die Akzeptanz des Nutzers ist wichtig. Diese hängt vom Tragekomfort, der Einfachheit der Nutzung, physiologischen Auswirkungen auf den Körper und noch einigen weiteren Faktoren ab. Die Akzeptanz der Gesellschaft ist ebenfalls relevant. Diese hängt wiederum von der Akzeptanz vieler Nutzer, aber auch von der Akzeptanz der Firmenvorstände und dem Meinungsbild der Gesellschaft, ab. Ein Exoskelett kann noch so viele Vorteile haben, wird es nicht akzeptiert, so wird es nicht effektiv genutzt und führt zu einer schlechten Arbeitsmoral und geringerer Produktivität.

Ein weiterer Punkt ist die bereits angesprochene Kosten-Nutzen-Abschätzung. Fällt diese für Firmen nicht profitabel aus, werden Exoskelette und ihre Vorteile für Arbeiter nicht zur Verwendung kommen.

Die Energieversorgung hat in den letzten Jahren große Fortschritte gemacht, dennoch muss ein Exoskelett in regelmäßigen Intervallen an eine Energiequelle, z. B. ein Stromnetz, angeschlossen werden. Dies schränkt die Mobilität und Betriebsdauer ein.

Dies sind Punkte, die bei der Investition und Nutzung von Exoskeletten in Betracht gezogen werden müssen.

Literatur

1. Ackerman, E.: Sarcos demonstrates powered exosuit that gives workers super strength. https://spectrum.ieee.org/sarcos-guardian-xo-powered-exoskeleton (2024), accessed: 31.1.2024
2. Agade: Human-tailored robotics - agadexo shoulder. https://agade-exoskeletons.com/en/product/ (2024), accessed: 29.1.2024
3. Aguirre-Ollinger, G., Colgate, J.E., Peshkin, M.A., Goswami, A.: Active-impedance control of a lower-limb assistive exoskeleton. In: 2007 IEEE 10th International Conference on Rehabilitation Robotics. pp. 188–195 (2007). https://doi.org/10.1109/ICORR.2007.4428426
4. Alabdulkarim, S., Nussbaum, M.A.: Influences of different exoskeleton designs and tool mass on physical demands and performance in a simulated overhead drilling task. Applied Ergonomics **74**, 55–66 (2019). https://doi.org/https://doi.org/10.1016/j.apergo.2018.08.004, https://www.sciencedirect.com/science/article/pii/S0003687018302618
5. Auxivo AG: Deltasuit - das Überkopf-exoskelett für erweiterte schulterunterstützung. https://www.auxivo.com/de/deltasuit (2024), accessed: 3.2.2024
6. Bogue, R.: Exoskeletons – a review of industrial applications. Industrial Robot: An International Journal **45**(5), 585–590 (Jan 2019). https://doi.org/10.1108/IR-05-2018-0109, https://doi.org/10.1108/IR-05-2018-0109
7. Brian D. Lowe, W.G.B., Peterson, D.R.: Astm f48 formation and standards for industrial exoskeletons and exosuits. IISE Transactions on Occupational Ergonomics and Human Factors **7**(3-4), 230–236 (2019). https://doi.org/10.1080/24725838.2019.1579769, https://doi.org/10.1080/24725838.2019.1579769
8. Christensen, S., Bai, S., Rafique, S., Isaksson, M., O'Sullivan, L., Power, V., Virk, G.S.: Axo-suit-a modular full-body exoskeleton for physical assistance. In: Mechanism Design for Robotics: Proceedings of the 4th IFToMM Symposium on Mechanism Design for Robotics. pp. 443–450. Springer (2019)
9. Christensen, S.S.: User-centered modelling and design of assistive exoskeletons (2023)
10. Deng, M.J., Wang, Z., He, H.H., Xue, Y.: Design and weight lifting analysis of a strengthen upper limb exoskeleton robot. In: Industrial Design and Mechanics Power II. Applied Mechanics and Materials, vol. 437, pp. 695–699. Trans Tech Publications Ltd (12 2013). https://doi.org/10.4028/www.scientific.net/AMM.437.695
11. EU-OSHA: Muskel- und skeletterkrankungen. https://osha.europa.eu/de/themes/musculoskeletal-disorders (2023), accessed: 14.12.2023
12. European Agency for Safety and Health at Work and Podniece, Z and Pinder, A and Yeomans, L: Work-related musculoskeletal disorders – Back to work report. Publications Office (2007)
13. exoIQ GmbH: Passive vs. aktive exoskelette – funktionsweisen und charakteristika. https://www.exoiq.com/wissen/passive-vs-aktive-exoskelette-funktionsweisen-und-charakteristika (2024), accessed: 8.1.2024
14. Fontana, M., Vertechy, R., Marcheschi, S., Salsedo, F., Bergamasco, M.: The body extender: A full-body exoskeleton for the transport and handling of heavy loads. IEEE Robotics Automation Magazine **21**(4), 34–44 (2014). https://doi.org/10.1109/MRA.2014.2360287

15. Gopura, R.A.R.C., Kiguchi, K.: Mechanical designs of active upper-limb exoskeleton robots: State-of-the-art and design difficulties. In: 2009 IEEE International Conference on Rehabilitation Robotics. pp. 178–187 (2009). https://doi.org/10.1109/ICORR.2009.5209630
16. Heo, U., Kim, S.J., Kim, J.: Backdrivable and fully-portable pneumatic back support exoskeleton for lifting assistance. IEEE Robotics and Automation Letters **5**(2), 2047–2053 (2020). https://doi.org/10.1109/LRA.2020.2969169
17. Hyundai Motor Deutschland GmbH: Hyundai erleichtert Überkopfarbeit mit tragbarer roboterweste. https://www.hyundai.news/de/articles/press-releases/hyundai-erleichtert-ueberkopfarbeit-mit-tragbarer-roboterweste.html (2024), accessed: 3.2.2024
18. Kim, S., Srinivasan, D., Nussbaum, M.A., Leonessa, A.: Human gait during level walking with an occupational whole-body powered exoskeleton: Not yet a walk in the park. IEEE Access **9**, 47901–47911 (2021). https://doi.org/10.1109/ACCESS.2021.3068836
19. soo Kim, W., hoon Lee, S., don Lee, H., nam Yu, S., soo Han, J., soo Han, C.: Development of the heavy load transferring task oriented exoskeleton adapted by lower extremity using qausi - active joints. In: 2009 ICCAS-SICE. pp. 1353–1358 (2009)
20. Lee, H., Ferguson, P.W., Rosen, J.: Chapter 11 - lower limb exoskeleton systems—overview. In: Rosen, J., Ferguson, P.W. (eds.) Wearable Robotics, pp. 207–229. Academic Press (2020). https://doi.org/https://doi.org/10.1016/B978-0-12-814659-0.00011-4, https://www.sciencedirect.com/science/article/pii/B9780128146590000114
21. Lee, H., Kim, W., Han, J., Han, C.: The technical trend of the exoskeleton robot system for human power assistance. International Journal of Precision Engineering and Manufacturing **13**(8), 1491–1497 (Aug 2012). https://doi.org/10.1007/s12541-012-0197-x, https://doi.org/10.1007/s12541-012-0197-x
22. de Looze, M.P., Krause, F., O'Sullivan, L.W.: The potential and acceptance of exoskeletons in industry. In: González-Vargas, J., Ibáñez, J., Contreras-Vidal, J.L., van der Kooij, H., Pons, J.L. (eds.) Wearable Robotics: Challenges and Trends. pp. 195–199. Springer International Publishing, Cham (2017)
23. Martinez, F., Retolaza, I., Pujana-Arrese, A., Cenitagoya, A., Basurko, J., Landaluze, J.: Design of a five actuated dof upper limb exoskeleton oriented to workplace help. In: 2008 2nd IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics. pp. 169–174 (2008). https://doi.org/10.1109/BIOROB.2008.4762788
24. Martini, M., Botta, J.: Iron man am arbeitsplatz?–exoskelette zwischen effizienzstreben, daten-und gesundheitsschutz. Neue Zeitschrift für Arbeitsrecht pp. 625–637 (2018)
25. Matsubara, T., Uchikata, A., Morimoto, J.: Full-body exoskeleton robot control for walking assistance by style-phase adaptive pattern generation. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 3914–3920 (2012). https://doi.org/10.1109/IROS.2012.6385528
26. McFarland, T., Fischer, S.: Considerations for industrial use: A systematic review of the impact of active and passive upper limb exoskeletons on physical exposures. IISE Transactions on Occupational Ergonomics and Human Factors **7**(3-4), 322–347 (2019). https://doi.org/10.1080/24725838.2019.1684399, https://doi.org/10.1080/24725838.2019.1684399

27. Nassour, J., Zhao, G., Grimmer, M.: Soft pneumatic elbow exoskeleton reduces the muscle activity, metabolic cost and fatigue during holding and carrying of loads. Scientific Reports **11**(1), 12556 (Jun 2021). https://doi.org/10.1038/s41598-021-91702-5, https://doi.org/10.1038/s41598-021-91702-5
28. Orthexo.de - Ratgeber für Neuroorthopädie und Exoskelette UG: Industrielle exoskelette: Die 77+ besten exoskelette für die industrie, logistik, handwerk und pflege (2023). https://orthexo.de/exoskelette/industrielle-exoskelette/?phaina-interviewID=49lA7hgXIr423ekDyGXcz (2023), accessed: 14.12.2023
29. Otten, B.M., Weidner, R., Argubi-Wollesen, A.: Evaluation of a novel active exoskeleton for tasks at or above head level. IEEE Robotics and Automation Letters **3**(3), 2408–2415 (2018). https://doi.org/10.1109/LRA.2018.2812905
30. RB3D: exo back - by rb3d. https://www.rb3d.com/en/exosquelettes/exoback (2024), accessed: 3.2.2024
31. Redaktion Automotive-Aktuell.de: Wissenswertes über das exoskelett in der automobilindustrie. https://www.automotive-aktuell.de/wie-kommen-exoskelette-in-der-automobilindustrie-zum-einsatz-34483.html (2023), accessed: 14.12.2023
32. Sado, F., Yap, H.J., Ghazilla, R.A.R., Ahmad, N.: Design and control of a wearable lower-body exoskeleton for squatting and walking assistance in manual handling works. Mechatronics **63**, 102272 (2019). https://doi.org/https://doi.org/10.1016/j.mechatronics.2019.102272, https://www.sciencedirect.com/science/article/pii/S0957415819301059
33. Sarcos Technology and Robotics Corporation: Guardian® xo® - full-body powered exoskeleton. https://www.sarcos.com/products/guardian-xo-powered-exoskeleton/ (2024), accessed: 31.1.2024
34. Sarcos Technology and Robotics Corporation: Guardian® xo® brochure. https://www.sarcos.com/wp-content/uploads/Sarcos-Guardian-XO-Brochure.pdf (2024), accessed: 31.1.2024
35. Skelex B.V.: Skelex 360-xfr - the ultimate exoskeleton for overhead work. https://skelex.com/skelex-360-xfr/ (2024), accessed: 3.2.2024
36. Theurel, J., Desbrosses, K., Roux, T., Savescu, A.: Physiological consequences of using an upper limb exoskeleton during manual handling tasks. Applied Ergonomics **67**, 211–217 (2018). https://doi.org/https://doi.org/10.1016/j.apergo.2017.10.008, https://www.sciencedirect.com/science/article/pii/S0003687017302296
37. ULS Robotics Co.: Bes-hv - elecrically driven lumbar exoskeletton robot. https://www.ulsrobotics.com/en/h-col-130.html (2024), accessed: 31.1.2024
38. Voilqué, A., Masood, J., Fauroux, J., Sabourin, L., Guezet, O.: Industrial exoskeleton technology: Classification, structural analysis, and structural complexity indicator. In: 2019 Wearable Robotics Association Conference (WearRAcon). pp. 13–20 (2019). https://doi.org/10.1109/WEARRACON.2019.8719395
39. Yap, H.K., Lim, J.H., Nasrallah, F., Goh, J.C.H., Yeow, R.C.H.: A soft exoskeleton for hand assistive and rehabilitation application using pneumatic actuators with variable stiffness. In: 2015 IEEE International Conference on Robotics and Automation (ICRA). pp. 4967–4972 (2015). https://doi.org/10.1109/ICRA.2015.7139889
40. Yu, S., Huang, T.H., Wang, D., Lynn, B., Sayd, D., Silivanov, V., Park, Y.S., Tian, Y., Su, H.: Design and control of a high-torque and highly backdrivable hybrid soft exoskeleton for knee injury prevention during squatting. IEEE Robotics and Automation Letters **4**(4), 4579–4586 (2019). https://doi.org/10.1109/LRA.2019.2931427

NRAM as a potential DRAM replacement

Quoc Anh Dang

Karlsruhe Institut für Technologie, Fakultät für Informatik, 76128 Karlsruhe, Germany

Abstract. DRAM is globally used in for example over mobile devices, deskop pcs and data centers and it has been improved over 50 years. As a result, improving DRAM becomes more difficult. Meanwhile new materials have been discovered and people have started to test, if they could be a better alternative for DRAM. In this paper we're discussing the motivation, why replacing DRAM might be more suitable than trying to keep improving DRAM and why NRAM is a potential replacement for DRAM.

Keywords: DRAM · NRAM · Carbon nanotubes

1 Introduction

DRAM was invented in 1970 and has been excessively used since then for various electronic devices. For this reason researchers are constantly trying to improve the performance and efficiency of DRAM to keep up with the other electronic components like a central processing unit (CPU) of a personal computer (PC) in order to not becoming a bottle neck to them.

We will take a rough overview of DRAM itself and look into some of its characteristics and its challenges and why NRAM *could* be a replacement for DRAM in the future.

2 DRAM

A *Dynamic Random Access Memory (DRAM)* is mostly known for its excessive usage in PCs and mobile devices, due to its low cost and increasing improvement over the time. In order to understand if it's even worth it to try to replace DRAM with another technology, we will have to take a look into it's functionality what it implies.

2.1 Organization

One DRAM-Module holds in general several storage-matrices as shown in figure 2.1. If we want to access a specific cell, which holds the data 1 or 0, from the matrix, we'd set a current on the given *Word Line* from figure 2 which is selected by a multiplexer in the *Row Address Selection* from figure 2.1.

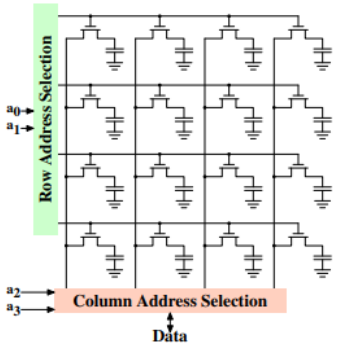

Fig. 1. A storage-matrix made out of DRAM cells as shown in figure 2.

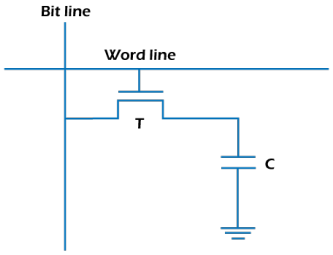

Fig. 2. A DRAM-cell where the capacitor (C) holds the state of the cell by holding the electrons, to represent the state *SET* and holding none electrons to represent the state *UNSET*.

Reading After setting a current on the given *Word Line*, the transistor *T* lets the electrons pass through it and moves the electrons inside the capacitor to the bit line, if there are any. Then the electric charge is read from the bit line and stored in a buffer where the column selection decides which bits should be moved into the data bus to send them to their destination.

Writing Writing into a DRAM cell works similar as reading from it. The difference is mainly, that we've already set the value on the bit line which is simply going through the transistor and then to the capacitor which will hold the value afterwards.

2.2 Disadvantages and challenges

Now after getting a rough overview how the two main actions (*reading* and *writing*) work, we can now go to its challenges and why they exist.

Refresh times One property of a capacitor is that it loses its charge, which would result into information loss. In order to avoid this, the DRAM module schedules a refresh after some milliseconds on the capacitors to avoid losing the information. No memory-operation can be executed during the refreshing time and also power has to be used for this action. Due to this property of losing the charge if the DRAM module doesn't receive power, DRAM belongs to the type of *volatile* memory [14].

To sum it up: DRAM allows only a specific time window where we can work with the actual data.

Variable Retention Time The maximum duration for a DRAM cell to hold it's value is also called its *retention time*. In general, good DRAM cells have a

high retention time and thus a low leakage from the capacitor while bad DRAM cells are the opposite, they have a low retention time due to a high leakage from the capacitor. There's the *Variable Retention Time Phaenomen (VRT)* which describes the behaviour of DRAM cells which randomly switch between good and bad DRAM cells. There are some solutions like an on-die ECC (Error Correcting Code) which fixes most of the data corruptions due to VTR but with the cost of more required space for an additional cell array and a higher power consumption [14].

To sum it up: DRAM suffers from non-deterministic behaviour of the capacitor which requires additional hardware to avoid them.

Thermal limits On the one hand DRAM modules are working very reliable from 0°C to about 85°C. Starting from 85°C to about 95°C it starts to get some smaller errors until it goes above 95°C where it starts losing data. But on the other hand automotive and other industries are using DRAM modules in wider temperature spans, up to 125°C. The higher the temperature gets, the higher the leakage of a capacitor becomes as well hence the capacitor needs to be refreshed more often but setting a current on the cell is also creating heat so it's a dead end if a DRAM module becomes too hot. That's where a shutdown is the only way out [14].

To sum it up: The industry requires more resistant memory modules to operate under difficult environments but DRAM is reaching its limits to fulfill them.

Limited minimum size It's desired to decrease the total size of a DRAM cell in order to put more cells for the same amount of space but this road can't be followed forever. There's an aspect ratio between the size of the DRAM cell and its effectiveness: The smaller a DRAM cell gets the smaller the capacitor has to become as well. Hence its charge-capacity will also decrease and more frequent refresh times are required to avoid unwanted data loss. To present the latest DDR5 has DRAM cells of about 15 nm and DRAM cells can't become smaller then 10nm otherwise they aren't effective enough to be used, so the limit is almost reached [14] [4].

To sum it up: DRAM cells are reaching their minimal size, hence the density can't be improved for a long time anymore.

Row-Hammer attack The required recharging times leads to an unsolvable challenge for DRAM-modules, called "row hammer" as explained in [12]. To sum it up, it's possible to flip some bits but unwanted, by frequently accessing it's nearby rows which will discharge the capacitors of our target row faster and if the recharge-timing can't keep up, the state of the capacitor will alter which could result into possible security exploits or data corruption [14].

To sum it up: DRAM suffers from an unfix able security issue where workarounds are only possible.

2.3 Summary

As we can see, DRAM includes various issues with only little space for improvement, if there's any. This is mostly due to the capacitor which is an essential component for a DRAM cell and improving the capacitor isn't possible. Therefore a replacement for the capacitor to hold the data is a possible way to fix or avoid the problems as told before. And now we'll introduce such a possible replacement for the capacitor and with that the DRAM cell: Nano-RAM.

3 NRAM

Nano-RAM or *Nanotube-RAM* (NRAM) is a different computer memory technology from the company *Nantero* who invented NRAM. Instead of relying on a capacitor as DRAM does it makes use of *carbon nanotubes* to save the state of a memory cell. We'll dive into some properties of carbon nanotubes first to get a rough understanding what a NRAM cell is made of.

3.1 Carbon Nanotubes

Carbon Nanotubes (CNT) are in general a very interesting material due to its various use cases for example as a building material, for solar cells but also for transistors and this is thanks to their structure which we will introduce now.

Structure CNTs consist only of carbon atoms whereby the carbon atoms have a honeycomb structure with hexagons and three bonding partners each. They can have different amount of layers as you can see in 3. CNTs with only a single wall are called *single-walled carbon nanotubes* (SWCNT) and CNTs with more than one layer are called *multi-walled carbon nanotubes* (MWCNT). The diameter of a SWCNT is commonly below about 2.5 nm while the diameter of a MWCNT in general starts from about 5 nm [10].

CNTs can be also either open or closed tubes, empty and filled tubes (for example with silver, liquid or noble gases). Depending on the structure, the electrical conductivity within the tube is metallic or semi-conductive. There are even CNTs which are superconducting at low temperatures. CNTs with an ideally hexagonal structure have a uniform thickness and are linear but kinked or narrowing tubes which even contain pentagonal carbon rings are also possible [2].

MWCNTs also share a lot of properties with graphite because the distance between two layers in a MWCNT (3.40Å) and graphite (3.35Å) are almost identical and both are only made out of carbon. As a result CNTs can be described by the same way as graphene by lattice vectors as shown in 4.

The greenish part should represent the CNT from a side view as if you'd hold the tube in front of you and each tube opening is at the top and at the bottom. As a result the vector $\vec{C_h}$ is the tangent of the tube which represents the line as if we would cut through the CNT. The vectors $\vec{a_1}$ and $\vec{a_2}$ are spanning a so-called

Fig. 3. On the left you can see a SWCNT and on the right a MWCNT .

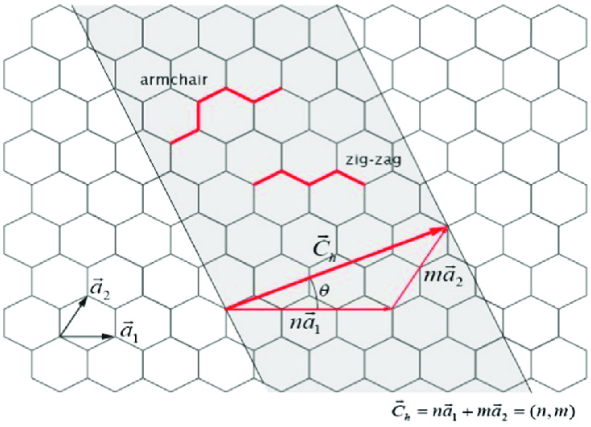

Fig. 4. Graphene honeycomb grid of a rolled up CNT .

unit cell. Both are basis vectors, hence linear independent, for the honeywomb. So $\vec{C_h}$ can be written (or calculated) as

$$\vec{C_h} = na_1 + ma_2$$

where $n, m \in \mathbb{Z}$. Also, due to the hexagonal symmetry it implies that $0 \leq |m| \leq n$. θ can be calculated by na_1 and ma_2 hence the degree of how the honeywomb got rolled up can be calculated as well. Therefore n and m are providing all necessary information to calculate other values from the CNT. They are often written as an index pair (n, m) to identify and distinguish all possible CNTs . Some types of index pairs have been named but since there are too many possible index pairs to give every single index pair a distinct name, three different classes of CNTs have been created: *armchair* if $n = m$ (see figure 5, *zigzag* if $m = 0$ (see figure 6), otherwise *chiral* [10].

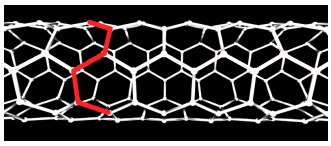

Fig. 5. An armchair CNT with a $(4, 4)$ configuration.

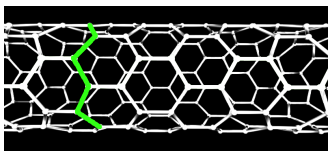

Fig. 6. A zigzag CNT with a $(8, 0)$ configuration.

Properties The index pair (n, m) also determines which kind of CNTs are metallic or semiconducting. If $\frac{(n-m)}{3}$ is an integer, then the CNT is metallic or else semiconducting. This means for example that all armchair CNTs are metallic and some zigzag or chiral CNTs are either metallic or semi conductive [2].

Regarding the electronics industry, CNTs have some other very attractive electrical properties: Their current carrying capacity is estimated to be 1000 times higher than that of copper wires and with a thermal conductivity of $6000\frac{W}{m*K}$ at room temperature, it's almost twice as high as that of diamond with $3320\frac{W}{m*K}$ so it's one of the best naturally heat conductors. As said before, they can also be used as semiconductors which makes them an attractive alternative for creating transistors which can withstand higher voltages and temperatures - and therefore higher clock frequencies - than silicon transistors [2].

Production One of the most common and promising ways to produce CNTs is by a *catalytic vapor phase deposition (CVD)* of carbon due to its simplicity regarding the required equipment, its operation and lower cost in comparison to other production methods [15]. Figure 7 shows a simple form of a CVD setup.

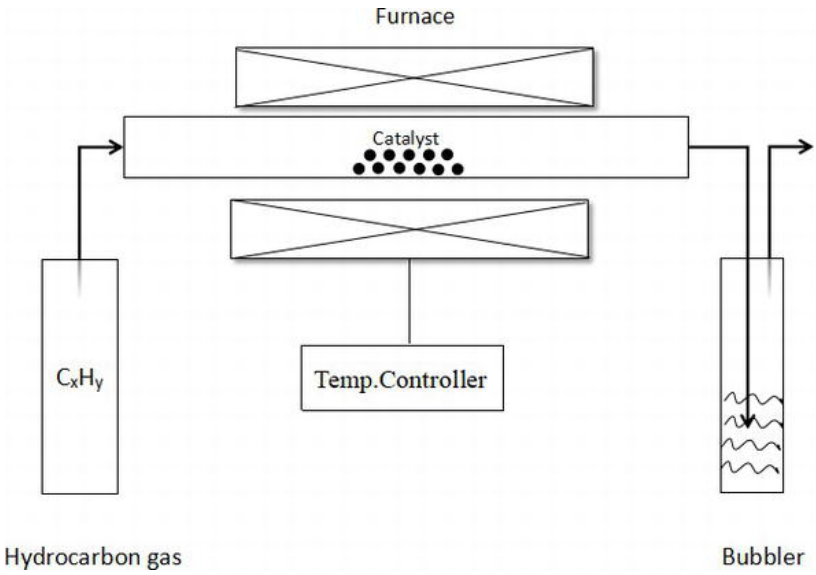

Fig. 7. Simple form of CVD setup to create CNTs.

In order to produce the CNTs with the CVD setup, a continuously flow of carbon atom-containing gas gets inserted into a tubular reactor which contains the catalyst and substrates at a sufficiently high temperature. The carbon atom-containing gas gets split up into carbon and hydrogen atoms at the substrate. From there, the carbon atoms are forming a CNT while the hydrogen keeps flowing through the tube and exits it [15].

3.2 The NRAM cell

As already told before CNTs have some very interesting properties for using them in transistors as they are behaving like a metal or a semiconductor, depending on their structure and in theory it has a lot of improvements in comparison to DRAM as shown in figure 8.

NRAM cells, as shown in figure 9, contain various amount of CNTs which influence the resistance of the cell: CNTs which are touching each other result into a lower cell resistance while not touching each other results into the opposite. This can be used to differentiate between 1s and 0s. In this case SET represents 0 and RESET 1. The data can be read by applying a little current between the upper and bottom electrode and test if a current flows through the cell. If the NRAM cell is in the SET state, no current is flowing due to the high resistance otherwise, if it's flowing, then the NRAM cell is in the RESET state due to the low resistance.

The state can be toggled by applying a small voltage greater than the read voltage between the top and bottom electrode. If the cell is in the SET state,

Timing	DDR4 SDRAM ns	DDR4 NRAM ns	DDR5 SDRAM ns
Row cycle	47.00	46.25	50.18
Access time	17.14	13.50	18.18
Row to column	15.00	23.00	18.18
Precharge	15.00	14.25	18.18
Write recovery	15.00	23.00	30.00
Activate to precharge	32.00	32.00	32.00
Refresh	350.00	0	350.00

Fig. 8. A comparison between DDR4 NRAM and DDR4 SDRAM and DDR5 SDRAM.

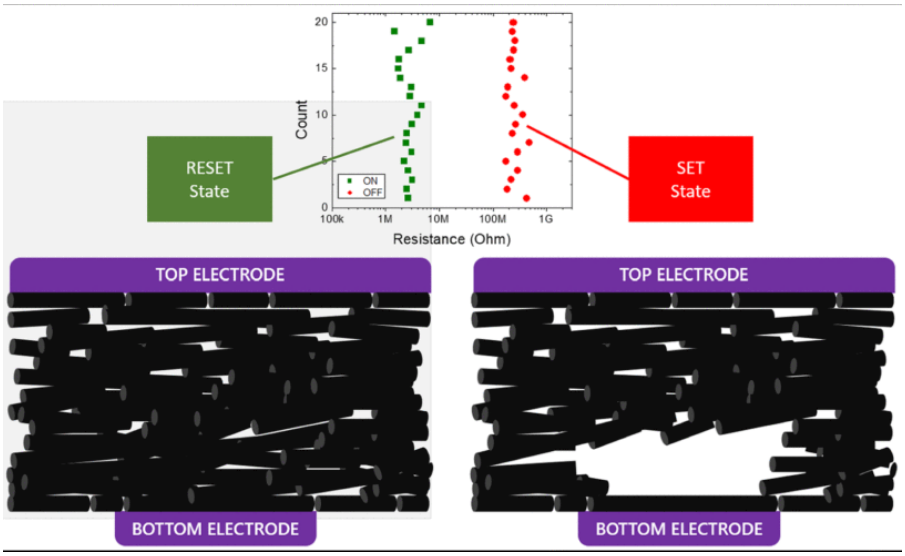

Fig. 9. A NRAM cell with its two different states. The black tubes should represent the nanotubes.

then the applied voltage will cause an electrostatic attraction between the CNTs. After the applied voltage is removed the CNTs remain in their state thanks to the Van-der-Waal's forces. Applying a small voltage (which is still greater than the read voltage) to the NRAM cell again will generate CNT phonon excitations with enough energy to separate the CNT from their conjunction points [7].

3.3 Advantages of NRAM in comparison to DRAM

Now after getting a rough overview of how a NRAM cell stores its data, we will introduce its improvements in comparison to DRAM and why it solves the issues of DRAM.

Non-Volatile Since the NRAM cells hold each state after setting it, NRAM belongs to the type of *non-volative memory* which means that the state of a cell doesn't get lost after removing the power supply. This gives us the opportunity for example to improve the boot up time of systems since their state after a full system shutdown still remains in the memory and it can continue where it stopped.

Better efficiency Compared to DRAM, NRAM doesn't require refresh times. Hence power is only required for reading and writing into the NRAM cell. In total NRAM can reduce the required memory power by between 21% and 50% compared to DRAM [9].

Also without the required refresh times there will be more time for computational relevant memory accesses which results into faster computation times.

Better Resistance Since it's required to have a higher activation energy than 5 eV to switch between the states of a NRAM cell, environmental noise like radiation, magnetism and vibration, which could cause problems to DRAM cells, can't affect NRAM cells. Those also have been tested on the space shuttle Atlantis without any problems [7] [9].

In addition to that, longer CNTs have a good thermal conductivity [11] [9] which reduces the risk of overheating the hardware.

Regarding endurance, it's assumed that the state of a NRAM cell can hold up to 300 years at over 300°C which makes NRAM attractive for the automotive for example [9].

Higher density and no Row-Hammer attack The minimal size for NRAM with less than 7nm [7] is smaller than the minimal size for DRAM cells. In addition to that no additional hardware is required to prevent row-hammer-attacks as this kind of attack doesn't exist for NRAM cells. That also counts variable retention times in since they also can't occur as NRAM cells aren't leaking anything. Thus there's more space which could be used for additional NRAM cells for a higher capacity and a better cost-per-area ratio, since no additional security-hardware is required.

3.4 State of the art

As we could see throughout this paper, NRAM seems to solve all problems from DRAM in *theory*, but that's not it: A NRAM module of 16 Mb was created and tested by [13] under a difficult environment where DRAM can't operate due to the high temperature. Not only that, the transistor size could even be reduced by 49%. Nevertheless at present there are no commercial products available yet and activity about NRAM became silent although Nantero told that there will be already chips available before 2010 [5] or soon after 2017 [1]. But companies are still believing on the technology and funded Nantero with $31.5 million dollar [6]. Fujitsu even acquired a licence of Nantero to start mass produce NRAM modules but that didn't happen either [3].

4 Summary

As we can see, in theory NRAM is a good candidate to replace DRAM with its improvements regarding efficiency, security, speed and future proof, but it lacks the full proof in practice. Time will tell us when NRAM modules are available and if they fulfill their expectations.

5 Image sources

- 2.1, `https://electronics.stackexchange.com/questions/533431/why-are-dram-cells-laid-out-in-a-square-with-regards-to-demux-size`, Accessed: 18.02.2024
- 2, `https://www.javatpoint.com/dram-in-computer-organization`, Accessed: 18.02.2024
- 3, `https://www.fuelcellstore.com/blog-section/carbon-nanotubes`, Accessed: 18.02.2024
- 4, `https://www.researchgate.net/figure/Schematic-diagram-of-the-chiral-vector-and-the-chiral-angle_fig1_260156240`, Accessed: 18.02.2024
- 5, `https://en.m.wikipedia.org/wiki/Carbon_nanotube#/media/File%3AArmchairCNT.png`, Accessed: 18.02.2024
- 6, `https://en.m.wikipedia.org/wiki/Carbon_nanotube#/media/File%3AZigzagCNT.png`, Accessed: 18.02.2024
- 7, `https://www.intechopen.com/chapters/67867`, Accessed: 18.02.2024
- 8, [8]
- 9, [9]

References

1. Carbon nanotube memory company's ship may finally come in. `https://spectrum.ieee.org/nrams-ship-may-have-finally-come-in`, Accessed: 18.02.2024.

2. Carbon nanotubes [kohlenstoffnanoröhre]. https://www.chemie.de/lexikon/Kohlenstoffnanorhre.html, Accessed: 18.02.2024.
3. Fujitsu targets 2019 for nram mass production. https://www.tomshardware.com/news/fujitsu-nram-nantero-carbon-nanotube,37437.html, Accessed: 18.02.2024.
4. Industry-leading ddr5 technology: Micron vs. samsung vs. sk hynix. https://www.techinsights.com/blog/industry-leading-ddr5-technology, Accessed: 18.02.2024.
5. Loser: Still waiting for nanotube memory chip. https://spectrum.ieee.org/loser-still-waiting-for-nanotube-memory-chip, Accessed: 18.02.2024.
6. With $31.5 million in new funds, nantero keeps up the good fight. https://spectrum.ieee.org/nantero-keeps-up-the-good-fight, Accessed: 18.02.2024.
7. S.Mohamed Jakkariya B Devikiruba. *AN OBSERVED NANORAM BASED ON CNT*, volume 3, Issue 5. IJAECS, 2016.
8. Bill Gervasi. Will carbon nanotube memory replace dram? *IEEE Micro*, 39(2):45–51, 2019.
9. Bill Gervasi and Rick Ridgley. Nram carbon nanotube. non-volatile memory... can dram be replaced? [white paper]. pages 1–10. NANTERO.
10. Stefanie Klumpp. *Separation von einwandigen Kohlenstoffnanoröhren*. PhD thesis, 2015.
11. Dong-Kwan Lee, Jongchan Yoo, Hyunwoo Kim, Byung-Ho Kang, and Sung-Hoon Park. Electrical and thermal properties of carbon nanotube polymer composites with various aspect ratios. *Materials (Basel)*, 15(4):1356, February 2022.
12. Kaveh Razavi, Ben Gras, Erik Bosman, Bart Preneel, Cristiano Giuffrida, and Herbert Bos. Flip feng shui: Hammering a needle in the software stack. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 1–18, Austin, TX, August 2016. USENIX Association.
13. Hitoshi. Saito, J. Watanabe, J. Seino, T. Tamura, N. Sashida, K. Hara, K. Kawabata, A. Fujii, J. Ohno, A. Nakakubo, M. Kojima, T. Shimoyama, H. Wada, L. Cleveland, H. Luan, R. Sen, N. Leong, T. Gallagher, and T. Rueckes. Development of 16 mb nram aiming for high reliability, small cell area, and high switching speed. In *2021 IEEE International Memory Workshop (IMW)*, pages 1–4, 2021.
14. Shigeru Shiratake. Scaling and performance challenges of future dram. In *2020 IEEE International Memory Workshop (IMW)*, pages 1–3, 2020.
15. Xiao-Di Wang, K. Vinodgopal, and Gui-Ping Dai. Synthesis of carbon nanotubes by catalytic chemical vapor deposition. In Hosam El-Din Saleh and Said Moawad Mohamed El-Sheikh, editors, *Perspective of Carbon Nanotubes*, chapter 2. IntechOpen, Rijeka, 2019.

Quantum Annealing

Fabian Jakob Brenneisen[1]

Karlsruhe Institute of Technology, Karlsruhe, Germany
fabian.brenneisen@student.kit.edu

Abstract. Quantum Annealing is a prominent Quantum Optimization Algorithm, that has seen great advances regarding research and development of hardware in recent years. It can be used to solve a variety of combinatorial optimization problems, such as boolean satisfiability or graph partitioning. This paper gives an overview of the theoretical foundations of Quantum Annealing, how to encode problems to be solved by a Quantum Annealer and the different potential applications of Quantum Annealing. It compares Quantum Annealing to universal, gate-based Quantum Computation and highlights some of the problems Quantum Annealing has. It also gives a brief overview of the current status of Quantum Annealing hardware.

Keywords: quantum annealing · combinatorial optimization · quantum optimization · adiabatic quantum computation

1 Introduction

The research field of Quantum computing has seen a great increase of activity in recent years. After theoretical developments of algorithms that have proven theoretical speedups on a Quantum Computer, like Shor's algorithm for prime factoring or Grover's search algorithm, research on hardware for Quantum Computing and further theoretical research on near term Quantum Algorithms started. Current Quantum Computing hardware, however, is highly influenced by noise from the environment and thus prone to error. This means there is a need for methods and algorithms that are not influenced by noise as much and could work more reliably on near term Quantum Computers. A prominent application, where such methods seem to work well, is Quantum Optimization. These methods include splitting the workload between a classical computer and a Quantum Computer with Variational Quantum Algorithms like the Quantum Approximate Optimization Algorithm or the Variational Quantum Eigensolver.

Another method used for Quantum Optimization that has attracted much attention due to comparatively fast progress in the development of hardware is Quantum Annealing, a heuristic optimization algorithm. D-Wave Systems, the most prominent company developing Quantum Annealers, launched its first commercially available machine in 2011 [26] and continued developing new Quantum Annealing hardware ever since. Since then, the hardware has matured to be competitive to gate-based Quantum Computers developed by companies like IBM or Google.

This paper will give first outline the history of Quantum Computing in general that led up to the current point and explain some important principles for talking about Quantum Optimization in general. The second part will then give an overview of the theoretical foundations of Quantum Annealing and explain different problem formulations, that are used to encode a problem for solving it on an Annealer. The last part will give some examples for applications of Quantum Annealing and give a brief overview over the current state of hardware developed by D-Wave Systems.

2 Background

This section will first give a brief overview over the history of Quantum Computing and then highlight different, currently active areas in the research field, as well as introduce some basic definitions for understanding the theory of Quantum Computing.

2.1 The History of Quantum Computing

The following is a brief summary of [11, Ch. 2] and will give an overview of early developments in the field of Quantum Computing:

Early developments Most commonly, Richard Feynman is credited to be the first person to conceptualize Quantum Computers. However, several researchers have proposed Quantum Computers before Feynman did. Paul Benioff describes how a Turing Machine could be constructed using a quantum mechanical system, thus describing the basic theory of Quantum Computing. Benioff then suggests, that there is a possibility of building such a system [2].

In 1981, Feynman held a talk at a conference about "Simulating physics with computers", that had significant influence on the early development of the research field of Quantum Computing. In his talk, Feynman argued that Quantum Computers might be better suited for describing a quantum mechanical system, as classical computers are unable to handle the number of variables involved [18].

First Quantum Algorithms After the basic theory of how a Quantum Computer could work was laid out, researchers began to investigate how algorithms for such systems could be constructed. David Deutsch was the first to develop an algorithm that showed an advantage in runtime of Quantum Computers over classical computers [11, Ch. 2]. He later generalized this algorithm in collaboration with Richard Jozsa. The problem Deutsch describes has a runtime of $\mathcal{O}(N)$ on a classical computer, but only a runtime of $\mathcal{O}(1)$ when using the Deutsch-Jozsa algorithm on a Quantum Computer [5]. The algorithm can determine if a black box boolean function f is balanced, i.e. the output is 0 or 1 for exactly half

of the input values respectively, by only querying the function once as opposed to at most n times using the classical method.

Although Quantum Algorithms up to that point had already shown an advantage over classical algorithms, the problems they were designed to solve, weren't important in real world applications. This changed when Ethan Bernstein and Umesh Vazirani described the Quantum Fourier transform (QFT) in 1993 [3], further showing Quantum Advantage and with their findings helping Peter Shor develop his famous algorithm for prime factoring [11, Ch. 2]. In his paper, Peter Shor describes Quantum Algorithms for factorization and calculating discrete logarithms in polynomial time, both having an exponential speedup over their classical counterparts [22]. Not only did his findings cause researchers to investigate new methods for cryptography as both problems are used in public key cryptography, but it also led to many others starting to enter the field of Quantum Computing.

After Shor developed his algorithm, Lov Grover made another significant contribution to the field. He developed a Quantum Algorithm for searching databases that has a quadratic speedup over classical algorithms.

These and several other theoretical developments sparked the interest of different companies and institutions to invest in, research and build hardware for Quantum Computing.

2.2 Applications and Limits of Quantum Computing

As we have already seen, Quantum Computing can offer significant speedups for specific problems. In this section, I will briefly present some of the currently most relevant areas of research in Quantum Computing.

Prominent applications As Feynman already suggested in 1981, Quantum computers can be used to simulate quantum mechanical systems [18]. Classically it is very hard to simulate such systems as the possible states of a quantum mechanical system scale exponentially. Quantum Computers, however, can process this large amount of information without the need for an exponential amount of physical resources. Consequently, Quantum Computing can be applied in various branches in physics and chemistry [7].

Another, currently very active, area of research is Quantum Machine Learning. One approach is to use Quantum Algorithms to run classically expensive algorithms or subroutines, that are commonly used. Other approaches include finding equivalent versions for different machine learning methods in quantum mechanics and encoding data in different ways in order to be processed on Quantum Computers [21].

One of the most promising applications for the near future are Quantum Optimization Algorithms. These include variational Quantum Algorithms like the Variational Quantum Eigensolver (VQE) or the Quantum Approximate Optimization Algorithm (QAOA) [16] and Quantum Annealing (QA) [19], which I will cover in more detail in this paper.

Limits of Quantum Computers While Quantum Computers may be able to utilize superposition to consider exponential amounts of data for computations, this data cannot be accessed. When measuring the Quantum Circuit, the state of the Qubits collapses to a single configuration and cannot be restored without knowing which state they were in [11, Ch. 1].

Today's Quantum Computers are in the Noisy Intermediate-Scale Quantum (NISQ) era, meaning that the amount of Qubits used in real Quantum Computers is relatively low (intermediate scale) and results of any computation will get increasingly noisy with deeper circuits, thus limiting the size of Quantum Circuits [17].

2.3 Qubits, Gates and Ising Spin Glasses

This section will introduce some important definitions and concepts for talking about Quantum Computing.

Qubits and Gates As opposed to classical computers, whose basic unit for computation is a bit, which can have the values 0 or 1, the basic unit of a Quantum Computer is a *Qubit*. Qubits can be in a superposition of two orthogonal basis states $|0\rangle$ and $|1\rangle$. This superposition state is represented by $|\Psi\rangle = a\,|0\rangle + b\,|1\rangle$ with $|a|^2 + |b|^2 = 1$ [11, Ch. 1]. When being measured, the Qubit will collapse to $|0\rangle$ with a probability of $|a|^2$ and to $|1\rangle$ with a probability of $|b|^2$ [11, Ch. 1]. The state of a system containing multiple uncorrelated Qubits can be represented by the tensor product of those Qubits $|v\rangle \otimes \cdots \otimes |u\rangle \equiv |v \dots u\rangle$ [11, Ch. 14].

Qubits can be manipulated using reversible unitary operators U, also referred to as *gates*, acting on one or multiple Qubits. Some important unary operators include the Hadamard gate H and rotation gates R_φ. H transforms a Qubit in state $|0\rangle$ into the *equal superposition* state $|+\rangle = \frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle$ and a Qubit in state $|1\rangle$ into the state $|-\rangle = \frac{1}{\sqrt{2}}\,|0\rangle - \frac{1}{\sqrt{2}}\,|1\rangle$. R_φ performs a rotation by the angle φ [11, Ch. 3]. Common multi-Qubit gates are the $SWAP$ gate, which swaps the state of two Qubits, and controlled gates like the $CNOT$ gate, which applies a NOT operator to the target Qubit, iff the control Qubit is in state $|1\rangle$ [11, Ch. 3].

Ising Spin Glasses Ising Spin Glasses, also referred to as Ising models, are $\mathcal{NP}$-hard optimization problems. They are used to encode a variety of different problems as a quantum mechanical representation, a Hamiltonian, to be solved via Quantum Computing [13]. Because Ising models are $\mathcal{NP}$-hard, there exists a mapping from every $\mathcal{NP}$-complete problem to an Ising model, meaning problems like boolean satisfiability (SAT) can be transformed to the corresponding Ising model and solved using a Quantum Computer. This formulation of problems is especially important for Quantum Optimization, which includes the main topic of this paper, Quantum Annealing. A list of different mappings for $\mathcal{NP}$-complete problems to Ising Spin Glasses can be found in [13].

3 Theoretical Foundations of Quantum Annealing

This section will first present the Quantum Adiabatic Theorem and then explain how its implications are used in Quantum Annealing (QA). After that, follows a comparison of QA to simulated annealing and to universal Quantum Computers.

3.1 The Quantum Adiabatic Theorem and Quantum Annealing

The adiabatic theorem states, that a system initially prepared in the ground state $\Psi(t)$ of a time-dependent Hamiltonian $H(t)$ will remain in that state $\Psi(t)$, if $H(t)$ changes "sufficiently slowly". The time evolution is given by the time-dependent Schrödinger equation:

$$i\hbar\frac{\partial}{\partial t}\left|\Psi(t)\right\rangle = H\left|\Psi(t)\right\rangle \tag{1}$$

The meaning of "sufficiently slowly" is determined by the smallest difference in energy between the ground state and the first excited state during the system's evolution [1, 26]. For quantifying the specific time needed for varying the state of a system, there exist a variety of different methods, some of which can be found in [1].

The statements of the adiabatic theorem are utilized for computation in QA [26]. The state of the system is initially prepared in the ground state of an easily solvable initial Hamiltonian H_i and then sufficiently slowly changed to a complex final or target Hamiltonian H_f. Due to the adiabatic theorem, the system will remain in the ground state, meaning after the annealing process, the system will be in the ground state, i.e., the solution of the final Hamiltonian H_f. This logic can be applied to Ising Hamiltonians which are, as discussed in section 2, $\mathcal{NP}$-hard.

In most cases, H_i is chosen to be a transverse field in x-direction,

$$H_i = \sum_{i=1}^{n}\sigma_i^x, \tag{2}$$

where σ_i^x is the Pauli-x matrix acting on the ith Qubit and n is the total number of Qubits in the system [26]. σ^x has the eigenstates $|x+\rangle = \frac{1}{\sqrt{2}}(|+\rangle + |-\rangle)$ and $|x-\rangle = \frac{1}{\sqrt{2}}(|+\rangle - |-\rangle$, which are both an equal superposition of the eigenstates of Pauli-z matrices σ^z. The Pauli matrices σ^x and σ^z are given by

$$\sigma^x = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} \quad \sigma^z = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}. \tag{3}$$

Thus, the system is initially prepared in the ground state $|\Psi(0)\rangle = |x-,\ldots,x-\rangle$. H_f can then be represented by the weighted sum of Pauli-z matrices applied to single Qubits and Pauli-z matrices applied to two interacting Qubits

$$H_f = \sum_{i\in V} h_i\sigma_i^z + \sum_{(i,j)\in E} J_{ij}\sigma_i^z\sigma_j^z, \tag{4}$$

where V is the set of vertices and E the set of edges of a graph $G(V, E)$ representing the connections between different Qubits. h_i is the energy of Qubit i and $J_{ij} = J_{ji}$ are the symmetric interaction strength of two connected Qubits i and j, that are connected by an edge in E.

For a given time $t \in [0, T_a]$, the transverse-field Hamiltonian $H(t)$ is given by

$$H(t) = A(t)H_i + B(t)H_f, \tag{5}$$

where $A(t)$ and $B(t)$ are monotonic functions with $A(0) = 1,\ B(0) = 0$ and $A(T_a) = 0,\ B(T_a) = 1$ [26]. According to the adiabatic theorem, the system will transition from the ground state $|\Psi(0)\rangle$ of H_i to the ground state $|\Psi(T_a)\rangle$ of H_f, iff the transition is sufficiently slow, meaning the annealing time T_a is sufficiently long. For $t = T_a$, where the magnitude of H_i reaches $A(T_a) = 0$ and the magnitude of H_f reaches $B(T_a) = 1$, the system will then be in a purely classical state and the Qubits can be measured to obtain the solution. The classical Ising model that is equivalent to (4) can be obtained by replacing the Pauli-z operators σ_i^z with classical spin variables $s_i \in \{-1, 1\}$.

Despite those theoretical guarantees, adiabaticity is hard to maintain in practice [26]. Current quantum systems are highly sensitive to noise from the environment, meaning the system could be kicked out of the ground state during the annealing process. The condition of "sufficiently slowly" is also difficult to fulfill, as the annealing time T_a depends on the gaps between low energy states of $H(t)$, which are unknown. Thus, T_a is only determined heuristically, making QA a heuristic optimization algorithm.

3.2 Comparison to Simulated Annealing

Although simulated annealing (SA) is not the state-of-the-art classical algorithm for solving Ising models, it is still useful to compare it to QA as there are many parallels, that help to understand how QA works and give an intuition for why QA might provide a Quantum Advantage.

SA can be used to find optimal or near optimal solutions for a cost function J defined on a finite set S [4]. For QA, the equivalent of J is the energy of the Hamiltonian H for a given (classical) state $|\Psi\rangle$ and S is equivalent to the set of possible states of the system. The SA algorithm is initialized in a state $x(0) \in S$ and in each iteration, a random neighbor j of the current state $x(t) = i$ is chosen. The next state is set to $x(t+1) = j$, if $J(j) \leq J(i)$. If $J(j) > J(i)$, the next state will still be chosen as $x(t+1) = j$ with a probability of $\exp(\frac{-J(j)-J(i)}{T(t)})$. Otherwise, $x(t+1) = i$ is chosen. The function $T(t)$, also referred to as "temperature" function or "cooling schedule" [4], is a monotonically decreasing function, whose value is decreased in each iteration of the algorithm. The initial value $T(0)$ is equivalent to the annealing time T_a in QA. By occasionally choosing less optimal states, SA is able to escape local minima and for sufficiently large initial values of $T(t)$ SA converges to a global minimum.

The key difference to QA is that large spikes in the cost function J need significantly more time to be overcome [26]. By utilizing quantum mechanical

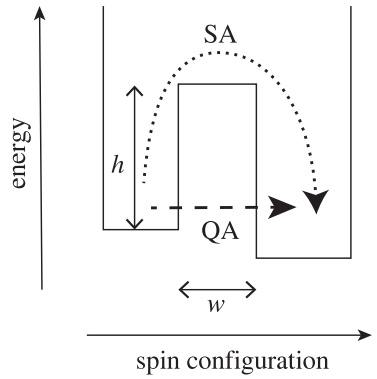

Fig. 1. Schematic diagram showing the difference between QA and SA in overcoming large spikes with height h and width w in the energy landscape [19, Fig. 1]

effects such as tunneling, QA can just tunnel through such large spikes directly to another local minima, eventually arriving at an optimal solution. This is schematically shown in Fig. 1.

3.3 Comparison to Universal Quantum Computers

Although the adiabatic theorem is a universal Quantum Computing paradigm, current Quantum Annealers cannot be used to implement everything needed for universal Quantum Computation [26]. This section discusses the differences between QA and the universal gate based approach of Quantum Computing and highlights some of the advantages and disadvantages of QA over universal Quantum Computing.

While universal gate based Quantum Computers are able to execute a wide variety of different algorithms, QA is a more specialized approach targeted at solving optimization problems [19]. Gate based Quantum Computers can implement arbitrary unitary operators U (or a subset of gates, that can be used to implement arbitrary unitary operators, see [11, Ch. 3.3]), thus being able to implement every possible Quantum Algorithm. Current Quantum Annealers on the other hand cannot be used to implement and efficiently execute every Quantum Algorithm, such as Shor's algorithm for prime factoring. Even within the realm of optimization problems which is the problem class Quantum Annealers are mainly used for, there exists a set of specifically constructed problem instances, for which it is hard to find a good solution using QA [23]. In these instances, gate based optimization algorithms such as QAOA or VQE can provide better solutions.

In general, however, current QA hardware is competitive to gate based Quantum Computers, as current Annealers have significantly more Qubits. IBM's newest gate based Quantum Computer *Condor* has a total of 1,121 Qubits [6], while D-Wave's newest Annealers based on the *Advantage architecture* have over

5,000 Qubits [15]. Because Annealers are leading in the industry when it comes to number of Qubits, they will initially also be the most prominent candidates for actually using Quantum Computing for solving optimization problems in practice, until larger scale universal Quantum Computers become available.

This means while Quantum Annealers may not be universal, they offer a solution for many practically interesting problems in the near future where universal fault-tolerant Quantum Computers are not yet available.

3.4 Problems with Quantum Annealing

This section discusses a few problems with QA in general and highlights the limitations of what is currently possible with QA hardware.

Just like any other Quantum Computer, Annealers are affected by noise from the environment. Specifically for QA, this means that the system can be kicked out of the ground state during the annealing process [19]. With longer annealing time, the impact of noise grows, thus decreasing the probability of the system remaining in the ground state. In practice, this means that QA can only be used to obtain approximate solutions and has to be run multiple times, resulting in a sample of the most likely solutions.

Noise also plays a big role for encoding continuous variables [26]. Continuous numbers are encoded using discrete binary variables, which poses problems for Annealers because they are analog computers. In a binary encoding of floating point numbers, the influence of each binary variable depends on exponentially spaced prefactors, meaning the noise of each binary variable has drastically different effects. Furthermore, as Annealers are analog computers, the input cannot be prepared with the same precision as with discrete binary variables in general.

The size of problems that can be solved using QA, is not only limited by the number of Qubits the machine has, but also by the factor at which problems scale when embedding them in specific hardware. As the hardware graphs are rather sparse (the degree of nodes in the hardware graph for the *Advantage* processor from D-Wave is 15 [15]), Ising models, which are in most cases rather dense, have to be transformed to sparse Ising models causing a space overhead. As Könz et al. [12] found, embedding problems into sparse hardware graphs causes not only a space overhead, but also a time overhead which is even more significant.

4 Problem Formulations

In general, problems can be transformed to Ising models, which I've already covered in section 2.3. However, there exist some other formulations which are often used either equivalently or as intermediate formulations before transforming to an Ising model. This section introduces such formulations and highlights some special cases where specific tradeoffs have to be made with current hardware.

4.1 Quadratic Unconstrained Binary Optimization Problems

The most prominent problem formulations besides Ising models are Quadratic Unconstrained Binary Optimization (QUBO) problems, as they can easily be transformed into Ising models by changing the computational basis

$$s_i \mapsto 2x_i - 1 \tag{6}$$

where $x_i \in \{0, 1\}$ is the binary variable of the QUBO problem and $s_i \in \{-1, 1\}$ is the spin variable of the Ising model [26, 10]. As this mapping is obviously polynomial, QUBO problems are, just like Ising models, $\mathcal{NP}$-hard. The solution of a QUBO problem is a configuration x of n binary variables $x_i \in \{0, 1\}, i = 1, \ldots, n$ that minimize an objective function

$$y = x^\top Q x = \sum_{i,j}^{n} Q_{ij} x_i x_j \tag{7}$$

where Q is either a symmetric or upper triangular matrix defining the wight of the interaction between different variables. Whether to choose a symmetric or upper triangular matrix Q for the problem depends on preference, as both forms can be transformed into one another.

4.2 Polynomial Unconstrained Binary Optimization Problems

In some cases it is easier to first transform the problem into a more unrestricted form of binary optimization problems, a Polynomial Unconstrained Binary Optimization (PUBO) problem, also called Higher Order Unconstrained Binary Optimization (HOBO/HUBO) problem in the literature. As the name suggests, PUBO problems are not restricted to quadratic terms only, but can instead contain polynomially large terms. A PUBO problem can be represented as

$$y = \sum_{p \in P} \left(c_p \prod_i x_{p_i} \right) \tag{8}$$

where P is the set of terms p, c_p is the coefficient and $x_{p_i} \in \{0, 1\}$ is the i-th variable in term p [9].

On gate-based Quantum Computers, such problems can be directly implemented with standard gates [8]. Current Quantum Annealers however, can only implement interactions between two Qubits [15]. This means, PUBO problems have to be transformed into QUBO problems to be solved with QA. This can be done through a process called quadratization. A simple example for a quadratization method would be "Reduction by Substitution" which was first formulated by Rosenberg in 1975 [20]. Here a variable pair (x_i, x_j) is picked and in every term $x_i x_j$ is substituted by a new auxiliary variable x_{ij}. To enforce the constraint $x_{ij} = x_i x_j$, a multiple of the penalty term $x_i x_j - 2x_i x_{ij} - 2x_j x_{ij} + 3x_{ij}$ is added. This process is repeated until all terms are at most quadratic, resulting in an equivalent QUBO problem. Except for some special cases, quadratization introduces a polynomial number of auxiliary variables, meaning more Qubits are necessary to implement the problem on hardware.

4.3 Constrained Optimization Problems

As the name suggests, both QUBO and PUBO problems are unconstrained. In real-world applications, however, optimization problems can generally have constraints, which cannot be directly enforced in the QUBO formulation. Instead, constraints are encoded in penalty terms that are larger, iff the constraint is not enforced. We've already seen such a penalty term which encodes the constraint $x_{ij} = x_i x_j$ in the section about the PUBO formulation above in the context of quadratization. By setting the binary variables in the term $x_i x_j - 2x_i x_{ij} - 2x_j x_{ij} + 3x_{ij}$ to different values, one can verify that this term is equal to 0 iff the constraint is satisfied, otherwise it is some value > 0. Constructing these specific penalty terms depends on the encoded problem. Some QUBO formulations can be found in a collection by Glover [10]. Often, penalty terms are scaled by a large factor λ to clearly separate invalid configurations from valid ones.

4.4 Example Transformation: Boolean Satisfiability

To get a better understanding for using the transformation methods and problem formulations explained above, I will demonstrate them on an example. In the following, I will transform the boolean formula

$$(\neg x_1 \vee \neg x_2) \wedge (x_3 \vee \neg x_1 \vee \neg x_4) \tag{9}$$

into an equivalent Ising model.

First, the formula has to be converted into a penalty term. For the first clause we can use quadratic penalties defined by Glover [10] leaving us with the penalty term $x_1 x_2$. As the second clause contains more than two literals, the quadratic penalties can not be applied directly. However, the quadratic penalty terms have the property that they are equal to 0 iff the clause is satisfied and equal to 1 iff the clause is not satisfied. This means a penalty term can be treated as a new boolean variable x_{ij} with $\neg x_{ij} = (x_i \vee x_j)$ (the same is true if one or both of the literals are negated). With this, we can rewrite the second clause as $((x_3 \vee \neg x_1) \vee \neg x_4)$ and use the quadratic penalties on the first two variables, leaving us with the penalty term $x_1 - x_1 x_3$. We then rewrite the clause as $(\neg x_{13} \vee \neg x_4)$ where x_{13} is equal to the previously determined penalty. We use the quadratic penalties again on this version of the clause, leaving us with the penalty term $x_{13} x_4$. In this term, we can substitute x_{13} with its penalty term resulting in the overall penalty $x_1 x_4 - x_1 x_3 x_4$ for the second clause. The overall penalty for the boolean formula is the sum of the penalties for the different clauses:

$$x_1 x_2 + x_1 x_4 - x_1 x_3 x_4 \tag{10}$$

This result is a PUBO problem whose solutions, i.e. the set of configurations where the term is equal to zero or in the ground state, are the same as for the original boolean formula. For any given configuration, the result of the penalty term is the number of unsatisfied clauses in the boolean formula. We can now

transform this penalty term into a QUBO problem by the means of quadratization using for example *Reduction by Substitution*, as described in section 4.2. As x_1x_4 is contained in two different sub-terms of the penalty term, we choose to substitute it for an auxiliary variable. Choosing to substitute x_1x_3 or x_3x_4 would also lead to a correct QUBO formulation. In order to increase the weight of the penalty for enforcing $x_{14} = x_1x_4$, we multiply it with the (arbitrarily chosen) factor 10. This results in the following QUBO problem:

$$x_1x_2 + x_{14} - x_{14}x_3 + 10x_1x_4 - 20x_1x_{14} - 20x_4x_{14} + 30x_{14} \tag{11}$$

Finally, we can transform this QUBO problem into the equivalent Ising model by changing from binary variables x_i to spin variables s_i with $x_i = \frac{1-s_i}{2}$. After simplifying, we get:

$$\begin{aligned} 8 &- \frac{9}{4}s_1 - \frac{1}{4}s_2 + \frac{1}{4}s_3 + \frac{10}{4}s_4 + \frac{39}{4}s_{14} \\ &+ \frac{1}{4}s_1s_2 + \frac{10}{4}s_1s_4 - \frac{20}{4}s_1s_{14} - \frac{1}{4}s_3s_{14} - \frac{20}{4}s_4s_{14} \end{aligned} \tag{12}$$

This formulation can now be embedded on QA hardware by setting the weights of individual Qubits and interactions between two Qubits to the corresponding factors.

5 Applications

This section will present the methods and results of some papers that use QA to solve different problems.

5.1 Boolean Satisfiability

This section covers the paper by Su et al. about solving SAT with QA [24]. Their contribution includes a new mapping from SAT to the Ising model, which differs from the approach presented in section 10, as well as methods to embed the resulting Ising models on a D-Wave Chimera hardware graph. The proposed mapping consists of three steps: First, the SAT formula is converted into an acyclic boolean network using only basic (i.e. unary or binary) logic operations. This boolean network is then converted into an Ising model by converting using a set of Ising primitives and chains. For example, the Ising primitive for the logic function $z = x \wedge y$ would be $-S_x - S_y + 2S_z - 2S_xS_z - 2S_yS_z + S_xS_y$. The constraints are then set so that the output from the Ising model is equal to a logical one. This mapping has two major benefits. Firstly, it works for an arbitrary SAT problem without the need to first convert it into a 3-SAT problem. Secondly, embedding the problem on D-Wave architecture can be approached with techniques from integrated circuit design, which scale well for real world applications. In order to solve the Ising model resulting from this mapping, the output Qubit is connected to an Ising primitive that is constantly equal to one.

To embed the problem onto a Chimera hardware graph, two physical Qubits of the graph are combined to one logical Qubit forming logic cells of complete graphs with four nodes where all nodes are connected to every other node. As the used Ising primitives need at most three Qubits, they can be directly embedded onto these logic cells. The individual logic cells are then placed onto the hardware graph and connected to one another using variations of already available algorithms.

The approach was evaluated using benchmark instances from the SAT competition and logic circuit benchmark. The results show that D-Wave Annealers can potentially be used to solve real world SAT problems as soon as machines with enough Qubits (about 100 times the sizes of a D-Wave 2X Annealer in case of the largest evaluated instances) become available.

5.2 Graph Partitioning

This section covers the paper by Ushijima-Mwesigwa et al. about using QA for graph partitioning (GP) [25]. GP describes the problem of dividing a graph $G = (V, E)$ into a partition $\Pi = (\Pi_1, \ldots, \Pi_k)$ of the set of vertices V where $\Pi_1 \cup \cdots \cup \Pi_k = V$ and $\Pi_i \cap \Pi_j = \emptyset$ for $i \neq j$ holds and the different parts are balanced, i.e. $|\Pi_i| \leq (1 - \epsilon)\lceil \frac{n}{k} \rceil$ for $1 \leq i \leq k$. The goal is to find such a partition where the number of cut edges is minimized. The main contribution of this paper is a QUBO formulation for dividing graphs into 2 or k parts and a comparison of the results using QA against state-of-the-art classical multi-level graph partitioners METIS and KaFFPaE as well as the best known solution from the Walshaw Archive in some cases. For QA on small graphs with up to around 70 vertices, the software tool *sapi* was used to run experiments directly on the D-Wave 2X Annealer. For QA on larger graphs with up to around 9000 vertices, the hybrid classical-quantum algorithm *qbsolv* was used.

The different solvers were evaluated on random graph models, graphs from the Walshaw Archive and quantum molecular dynamics structure graphs. Across all used graph sets, QA shows comparable result quality to the multi-level graph partitioners and in some cases even matches the quality of the best known solution for dividing the graphs into 2 equally sized parts. For dividing graphs into $k = 2, 4, 8$ and 16 parts, QA using *qbsolv* showed comparable or better results than METIS. These results indicate that QA could in practice be used to obtain high quality solutions for GP.

5.3 The Traveling-Salesman Problem

This section covers the paper by Martoňák et al. about using QA to solve the Traveling Salesman Problem (TSP) [14]. The main contribution of this paper is an Ising formulation of the TSP problem, including not only a final Hamiltonian H_f encoding the solution, but also an initial Hamiltonian H_i which is specifically tailored for the TSP problem. H_i however, was not used in the evaluation, as it would lead to high error when simulating QA. Instead, the standard transverse field Ising Hamiltonian was used. The goal for TSP is to find the shortest tour

between a set of cities i, where each city is visited exactly once. In the symmetric TSP problem considered in this paper, the distances d_{ij} between cities i and j are the same independent of the direction, meaning $d_{ij} = d_{ji}$. Each valid tour can be represented as a symmetric (or upper triangular) matrix U where $U_{i,j} = 1$, iff there is a connection between the cities i and j in the given tour. This leads to a natural QUBO formulation, encoding the length of a tour. The initial Hamiltonian H_i described in the paper encodes so-called *2-opt moves*, a technique that is commonly used in TSP solvers. In a 2-opt move, for example, one removes two edges e_{12} and e_{34} and then adds the edges e_{13} and e_{24} instead, resulting in a new tour. The evaluation of QA with this formulation shows that QA is superior to SA, but both QA and SA perform worse than current state-of-the-art TSP solvers. Furthermore, the algorithm was evaluated using only one specific TSP instance and is thus requires more extensive research.

6 QA Hardware

This section will briefly explain some of the details of QA hardware using the D-Wave Advantage architecture as an example, as it is currently the newest Quantum Annealer D-Wave has to offer. Although there are multiple companies developing Quantum Annealers, D-Wave Systems is by far the most successful and prominent one [26].

A Quantum Processing Unit (QPU) for QA consists of multiple Qubits and Couplers between those Qubits [15]. The system can be represented as a graph, where the nodes of the graph represent the Qubits and an edge between two nodes represents the Coupler connecting the two Qubits. The weights of both Qubits and Couplers can be set according to the factors of the QUBO to be solved on the machine. This graph is often referred to as the *hardware graph*. In the case of Advantages QPUs, D-Wave also calls this graph a *Pegasus graph*. The Pegasus graph consists of over 5000 nodes with an average degree of 15. This graph is rather sparse compared to the graph representation of many QUBO problems one might want to solve on the machine. To embed a QUBO onto the graph of an Annealer, it has to be converted into a sparse QUBO graph using a technique called *minor embedding*. This is done by breaking down individual logical Qubits into chains of multiple physical Qubits. In general, the higher the connectivity of the hardware graph is, i.e. the higher the average degree of the nodes, the smaller the chains for the embedding will be, resulting in better solution quality. Given this context, it is quite obvious that the Advantage architecture is a big step forward from the previous systems developed by D-Wave, which have an average node degree of 6.

7 Conclusion

Quantum Annealing (QA) can in theory be used to obtain (almost) optimal solutions for hard combinatorial optimization problems if the annealing time is sufficiently long. In practice however, as current and near future Quantum

Computers are prone to error and noise from the environment, and the annealing time thus has to be rather short to avoid completely ruined results due to noise, QA can only be used as a heuristic optimization algorithm to obtain approximate solutions.

There has been much progress in the research and development of QA hardware through companies like D-Wave Systems in recent years. Furthermore, Quantum Annealers are also easier to build and operate than universal, gate-based Quantum Computers [26]. However, especially the connectivity between Qubits on QA hardware is still really low compared to the connectivity needed to directly embed most relevant problems. This means problems have to be converted from dense graphs to sparse graphs, resulting in bigger problems in terms of the required number of Qubits, which are as a result also influenced more by noise.

In order to utilize the theoretical advantages of QA, there is still a lot of progress to be made on the hardware side. This means, unless there will be rapid developments, it is rather unlikely that QA can actually be used to solve large scale practically relevant problems in the near future. Apart from hardware, there could also be some progress made by researching direct transformations for specific problem classes to an embedding on a Quantum Annealer instead of the more general approach of transforming problems into a QUBO problem or Ising model.

References

1. Albash, T., Lidar, D.A.: Adiabatic quantum computation. Reviews of Modern Physics **90**(1), 015002 (2018)
2. Benioff, P.: The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. Journal of statistical physics **22**, 563–591 (1980)
3. Bernstein, E., Vazirani, U.: Quantum complexity theory. In: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing. pp. 11–20 (1993)
4. Bertsimas, D., Tsitsiklis, J.: Simulated annealing. Statistical science **8**(1), 10–15 (1993)
5. Deutsch, D., Jozsa, R.: Rapid solution of problems by quantum computation. Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences **439**(1907), 553–558 (1992)
6. Gambetta, J.: IBM Quantum Computing Blog — The hardware and software for the era of quantum utility is here. www.ibm.com/quantum/blog/quantum-roadmap-2033 (2023), accessed: 2024-02-16
7. Georgescu, I.M., Ashhab, S., Nori, F.: Quantum simulation. Reviews of Modern Physics **86**(1), 153 (2014)
8. Glos, A., Krawiec, A., Zimborás, Z.: Space-efficient binary optimization for variational quantum computing. npj Quantum Information **8**(1), 39 (2022)
9. Glover, F., Hao, J.K., Kochenberger, G.: Polynomial unconstrained binary optimisation–part 1. International Journal of Metaheuristics **1**(3), 232–256 (2011)
10. Glover, F., Kochenberger, G., Du, Y.: Quantum bridge analytics i: a tutorial on formulating and using qubo models. 4or **17**, 335–371 (2019)

11. Hidary, J.D., Hidary, J.D.: Quantum computing: an applied approach, vol. 1. Springer (2019)
12. Könz, M.S., Lechner, W., Katzgraber, H.G., Troyer, M.: Embedding overhead scaling of optimization problems in quantum annealing. PRX Quantum **2**(4), 040322 (2021)
13. Lucas, A.: Ising formulations of many np problems. Frontiers in Physics **2** (2014). https://doi.org/10.3389/fphy.2014.00005, http://dx.doi.org/10.3389/fphy.2014.00005
14. Martoňák, R., Santoro, G.E., Tosatti, E.: Quantum annealing of the traveling-salesman problem. Physical Review E **70**(5), 057701 (Nov 2004). https://doi.org/10.1103/PhysRevE.70.057701
15. McGeoch, C., Farré, P.: Advantage processor overview. D-Wave systems pp. 14–105 (2022)
16. Moll, N., Barkoutsos, P., Bishop, L.S., Chow, J.M., Cross, A., Egger, D.J., Filipp, S., Fuhrer, A., Gambetta, J.M., Ganzhorn, M., et al.: Quantum optimization using variational algorithms on near-term quantum devices. Quantum Science and Technology **3**(3), 030503 (2018)
17. Preskill, J.: Quantum computing in the nisq era and beyond. Quantum **2**, 79 (2018)
18. Preskill, J.: Quantum computing 40 years later (2023)
19. Rajak, A., Suzuki, S., Dutta, A., Chakrabarti, B.K.: Quantum annealing: An overview. Philosophical Transactions of the Royal Society A **381**(2241), 20210417 (2023)
20. Rosenberg, I.G.: Reduction of bivalent maximization to the quadratic case. (1975)
21. Schuld, M., Sinayskiy, I., Petruccione, F.: An introduction to quantum machine learning. Contemporary Physics **56**(2), 172–185 (2015)
22. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM review **41**(2), 303–332 (1999)
23. Streif, M., Leib, M.: Comparison of QAOA with Quantum and Simulated Annealing (Jan 2019)
24. Su, J., Tu, T., He, L.: A quantum annealing approach for boolean satisfiability problem. In: Proceedings of the 53rd Annual Design Automation Conference. p. 1–6. DAC '16, Association for Computing Machinery, New York, NY, USA (Jun 2016). https://doi.org/10.1145/2897937.2897973, https://dl.acm.org/doi/10.1145/2897937.2897973
25. Ushijima-Mwesigwa, H., Negre, C.F.A., Mniszewski, S.M.: Graph partitioning using quantum annealing on the d-wave system. In: Proceedings of the Second International Workshop on Post Moores Era Supercomputing. p. 22–29. ACM, Denver CO USA (Nov 2017). https://doi.org/10.1145/3149526.3149531, https://dl.acm.org/doi/10.1145/3149526.3149531
26. Yarkoni, S., Raponi, E., Bäck, T., Schmitt, S.: Quantum annealing for industry applications: Introduction and review. Reports on Progress in Physics (2022)

Recent developments in AI in video games

Joscha Mathias Reich[1][0009−0001−3269−3927]

[1]Karlsruher Institut für Technologie (KIT), Karlsruhe, Germany

Abstract. TBD.

Keywords: Artificial intelligence · video game · machine learning · neural network · reinforcement learning

1 Introduction

1.1 Categorisation of the topic

There are many definitions of the term "artificial intelligence" (AI) [24]. In 1955 John McCarthy et al. proposed to explore AI and speculated that "every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it" [28]. This led to the Dartmouth summer research project, which is seen by many as the birth of AI research [5].
The term itself refers to the intelligence of artificial systems (often computers or machines) as opposed to the intelligence of humans or other living beings [4].
A concise definition describes AI as "the study of how to make computers do things that people are better at" [31]. AI is a very general term that includes many sub-areas.
One of these applications has been AI since the development of the first video games in the 1950s. Traditionally, the goal of AI in video games was to simulate the intelligent behaviour of a non-player character (NPC), a character that isn't controlled by the player, in the game. This includes controlling everything from pathfinding and navigation to the character's behaviour, which is often simulated using finite-state machines (FSM) [33]. However, there are also other areas of application for AI in video games, such as the procedural generation of content in the game or player experience modelling, which can include dynamic difficulty adjustment (DDA) of the game, for example [36]. But AI in video games used to have many limitations: AI behaviour was quite simplistic and often had to break the rules of the game in order to keep up with players.
Recent breakthroughs have further opened the field of AI in video games. "AlphaStar" is an AI created by a research team at Google that published its findings in 2019 that can play the video game StarCraft II at grandmaster level [35]. "OpenAI Five" is another AI playing the video game "Dota 2" and was able to beat the then reigning world champions in 2019 [11].
These AIs have proven that the methods used in reinforcement learning not only work in games such as chess and Go [3], but also in video games that are played

in real time, i.e. the AI must be able to react to the player's actions in a fraction of a second. Another challenge is the enormous action space, for example 10^{26} possible actions at each step in the case of the AI "AlphaStar". In addition, there is unknown information in the games mentioned that the player has to explore (scout). These breakthroughs for AI in video games could also prove helpful in the future, because the demands (running in real time, large action space and incomplete information) on AI in these games are in many ways similar to many real world problems, such as weather forecasting, self-driving cars and more [2].

1.2 Introduction of important terms

Machine learning (ML) is a branch of AI that enables systems to improve their performance using data and algorithms without resorting to explicit programming. ML is divided into three main categories: supervised learning, unsupervised learning and reinforcement learning (RL). In supervised learning, the model learns from labelled data sets, while unsupervised uses unlabelled data sets, i.e. the data consists only of the input data. There is also semi-supervised learning, which is a mixture of both approaches. [16][9]
In RL, the model learns by an agent (an autonomously acting program) maximizing rewards through trial and error and with the help of a reward function. Successful attempts are encouraged, while failures are penalized. [16][17][14][9]
Two types of reinforcement learning are relevant for this topic: on-policy and off-policy reinforcement learning. While in on-policy reinforcement learning, there is only one policy which is being updated iteratively while training, in off-policy reinforcement learning there are two separate policies. A behaviour policy is used for data collection while also having a target policy for the training process [25].
Artificial neural networks (ANN) is one of the possible machine learning algorithms. ANNs were inspired by the biological structure of linked brain cells. There is a layer of input nodes, any number of additional layers and a layer of output nodes, each of which is linked to the others. All edges that connect two nodes have weights. By adjusting the edge weights, e.g. using the backpropagation algorithm during a training phase, the neural network can recognize patterns in the input data, for example. [18][16]
Finally, a trained AI model can be applied to new data in the inference phase, i.e. if, for example, an AI has been trained to recognize images of cats, the AI can recognize new images of cats in the inference phase [19].
Last but not least, AI in video games is dependent on fast reactions to changes in the input data. Parallelism therefore plays an important role in the inference phase. Parallelism refers to the parallel execution of processes in order to make better use of the available processing power. [22]

1.3 The considered games

Atari 2600 is a video game console developed by Atari, Inc., released in 1977 (for the US market, 1979 in Germany). The Atari console is known as one of the most relevant consoles in video game history.

To give input to the console, the user has to use a controller. The classic version consisted of a simple joystick with 8 directional outputs [6][1].
Beam Rider, Breakout, Enduro, Pong, Q*bert, Seaquest and Space Invaders are the seven games DeepMind trained their AI for in 2013 and was able to surpass the achievements of previous AIs in six out of those seven games. In three of those games, their AI even beat human experts [29].

StarCraft II is a real-time strategy (RTS) game published by Blizzard Entertainment in 2010 [13]. The genre is characterised by the fact that you have to build up an economy and control your units in order to defeat your opponent.
StarCraft II focuses on a multiplayer mode in which one player competes against another player in 1v1 and must destroy all of the opponent's buildings.
Each player plays one of three races: Terran, Zerg or Protoss. The races are distinguished by different units and buildings with different characteristics. These differences mean that the races have different strengths and weaknesses, each of which enables their own strategies.
In the game, each player has a field of vision around their units. Enemy units are recognised in this field of vision, so players have incomplete information about their opponent and must actively scout for information.
In 2019 DeepMind's AI agent, AlphaStar, was able to beat professional players and play consistently at a Grandmaster level (highest rank in Blizzard's matchmaking ranked queue) with and against all three available races, beating 99.8% of all human players. [15][8][20]

Dota 2 is a multiplayer online battle arena (MOBA) published by Valve in 2013 [10]. Five players compete in a team against another team of five players. Each player controls a single character, which has abilities that differ from character to character.
The aim of the game is to destroy the opposing team's "Ancient", which is the last building in a team's base. To achieve this, the players must accumulate gold and experience (for example by killing opposing players or neutral monsters that can be found on the map) so that the player characters become stronger and can then defeat the opposing team through good team coordination. Here too, a wide variety of strategies are possible thanks to the different abilities and respective strengths and weaknesses of the characters.
Similar to StarCraft II, the players in Dota 2 only have an incomplete view of the game and have to make do with incomplete information or scout information.
In 2019 the AI "OpenAI Five" was able to beat the then world champions (Team OG), making it the first AI achieving this at an esports game. [7][21]

2 In-depth description of the different AIs

To be able to discuss what methods are suitable for creating a state-of-the-art AI in video games, let's take a look at how these three AI agents were created.

2.1 ATARI 2600

The AI from DeepMind was trained to play seven Atari 2600 games: Beam Rider, Breakout, Enduro, Pong, Q*bert, Seaquest and Space Invaders.
DeepMind uses a reinforcement to learn a control policy. Its model is based on a convolutional neural network. The network training uses a version of the Q-learning algorithm (reinforcement learning algorithm that can learn the values of actions when in a specific state without constructing a model of the environment) and stochastic gradient descent (SGD, iteratively optimizes the objective function) to adjust the weights.
The algorithm is also off-policy – meaning that it only chooses the greedy strategy with a certain probability and otherwise chooses to go to new states, with the aim of exploring the entire state space.
The agent is only able to observe the current screen and has no additional internal information from the Atari emulator. But the current screen doesn't describe the current internal state of the emulator fully, because this state (including the game score, for example) could also depend on the prior states. Therefore, it is not possible to describe the current state of the game in a single state, and instead sequences of observations and actions must be used. Each of these sequences is represented as a state in a Markov decision process (MDP). MDPs can be used for reinforcement learning as it describes a set of states, the allowed actions, the probability that an action leads to another specific state and the expected reward for the given action. Using these individual sequences as states in an MDP allows predicting a reward for the respective sequence of actions.
This way, the agent is able to select the action which is expected to give the biggest reward after a certain amount of time [29].

Why not use Deep learning? Deep learning seems like a good choice for this kind of task, as it's been successful in areas like computer vision and speech recognition. But deep learning often requires a large, labelled dataset for training.
But in games the result is often very delayed and not immediately visible, meaning it's difficult to evaluate the corresponding rewards for an action if you want to use reinforcement learning.
Also, the data states in games aren't independent – one action leads to another specific state, depending on prior states. But in deep learning, the states are often assumed to be independent [29].

Why use a convolutional neural network? CNNs have so-called convolutional layers, which have matrices as their data input and use filters (called "kernels") to map the data to smaller matrices. CNNs can also use pooling layers (to combine the outputs of clusters for the next layer) and fully connected layers (where all inputs from previous layers are connected to each of the outputs for the next layer with weights and biases).
CNNs are often used in applications where the input data is visual data (pictures or video). CNNs can learn to recognize patterns in this visual data.
One other key feature used is experience replay, by storing experiences of the agent, with which the agent is able to (often randomly) sample those experiences and use them for future decision-making [32][27].
Another simplifying step taken is that the AI agent only selects which action to take every k^{th} frame, and simply repeating the action taken in-between k frames. This lets the agent train for k times the games in roughly the same time. The value of k was chosen to be 4 in all games, except in the game Space Invaders, k was chosen to be 3, because otherwise it led to important information being missed [29].

2.2 DeepMinds AlphaStar in StarCraft II

AlphaStar uses a policy to select an action. As input parameters, the policy has access to all previous observations until the current point in time. Observations include everything that describes the state of the game that a player can observe at any point in time. This includes information about all own units and building, as well as all visible enemy units.
These observations are processed with a network using a self-attention architecture. Self-attention describes a mechanism that identifies dependencies in the input data and focus on relevant information [34].
DeepMind introduced so-called scatter connections to combine spatial and non-spacial information and a deep long short-term memory (LSTM) system is used to be able to solve the problem of incomplete information in the observations [26]. All previous observations are processed by the LSTM to combat this problem.
For the first step, DeepMind relied on supervised learning, using a dataset of 971,000 replays of the top 22% of human players.
Before starting the supervised learning, for each of the replays, a strategy (or so-called "build order". This describes what the player built in which order but only in early stages of the game. A more well-known comparison would be chess openings.) was extracted, which was later used for rewarding the agent for sticking to the given strategy.
As the next step, reinforcement learning is used. Many different agents form a "league" in which the agents that are still training can play against other agents using a version of self-play. Fictitious self-play describes self-play that also plays against past versions of itself. This ensures that when one strategy beats the other strategy, no cycles form so that it also has to beat strategies of players in the past. An example for a strategy would be playing rock paper scissors. If

all present players prioritize playing rock, then it would be best to play paper. But after a few steps it would be best to play scissors and so on. If you have to play against all past versions of yourself, you would need to diversify your strategy [35].

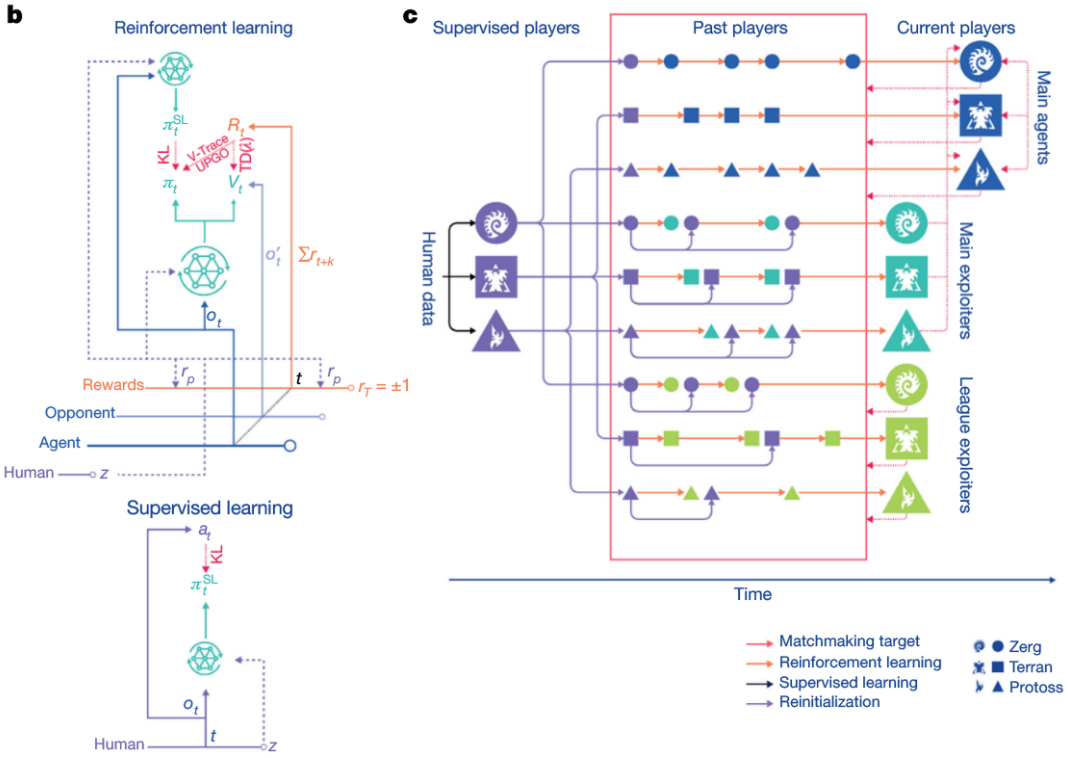

Fig. 1. AlphaStar model overview [35]

In **Fig. 1 b** the supervised and reinforcement learning is depicted. KL refers to the Kullback-Leibler divergence between the output of the model π_t and the human action a_t.
The reinforcement learning policy is updated using TD(λ), V-trace and UPGO).
Fig. 1 c shows a simplified model of the entire architecture. As a starting point, three groups of agents (one for each playable race) are used, which were trained via supervised learning and consecutively trained via reinforcement learning.
All agents together form a "league" of players. While training, after a specific time players add copies of themselves which are not updated any more to the league.

The main agents train against all players in the league. The goal of the league is to train the main agents to get as good as possible. To ensure this, a few special agents get introduced to the league.
The main exploiters are a group of agents that train against the main agents. They were created to expose weaknesses in the strategies of the main agents. Another group - the league exploiters - trains only against the past players [35].

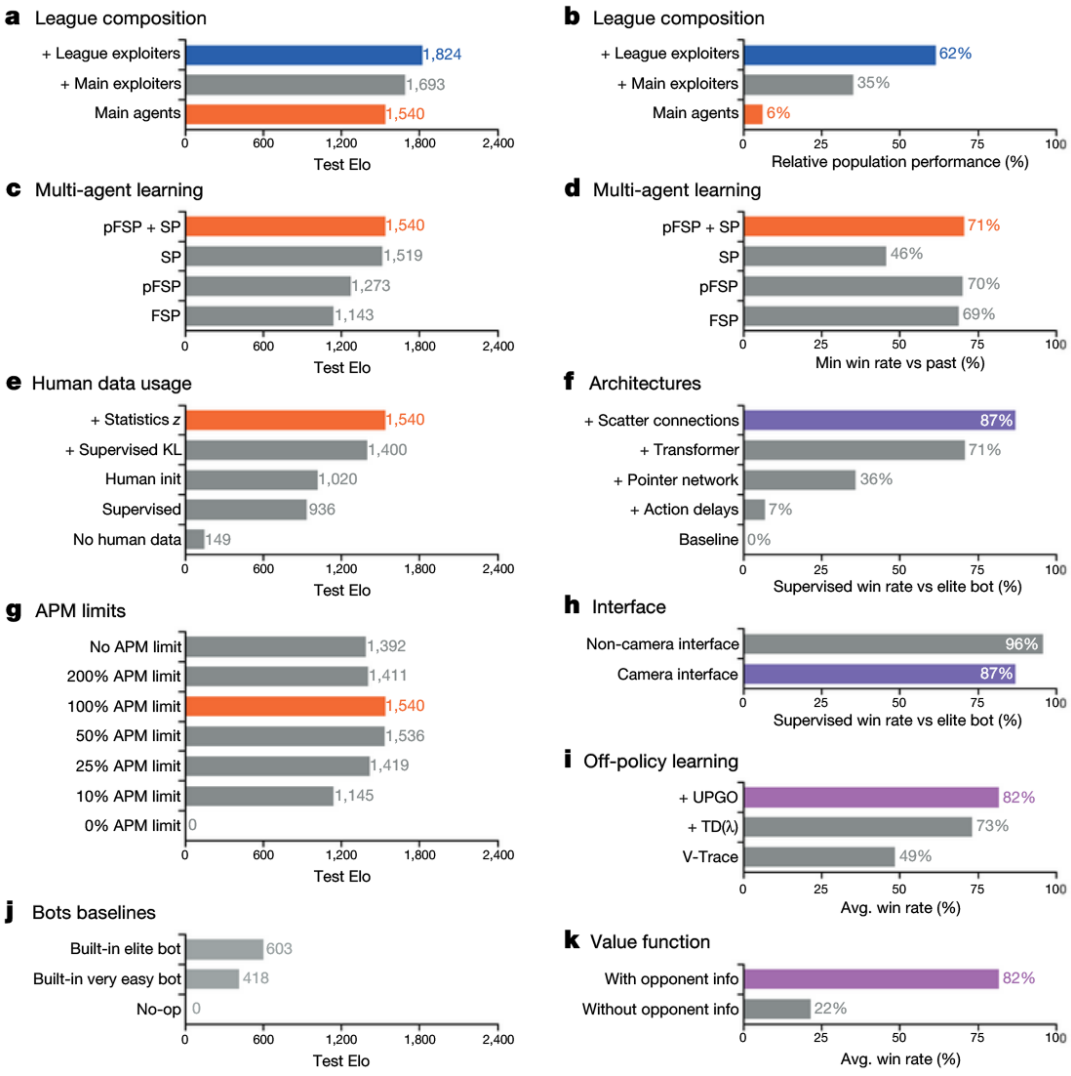

Fig. 2. Impact on performance of different components of AlphaStar [35]

In Fig. 2 key components of AlphaStar are listed and their impact on the performance of the AI agent is shown.
a and **b** show the impact of using the newly introduced main and league exploiters.
c and **d** show the impact of using (prioritized) (fictitious) self-play in different combinations.
f shows the impact of the architecture DeepMind used.
g refers to an APM limit. Interestingly, when limiting the allowed APM of the AI, the performance was better than without a limit.
h shows the impact of using the camera interface like a human player would, or using a non-camera interface, having full vision of the map.
i shows the different learning algorithms and their impact on performance [35].

2.3 OpenAI Five in Dota 2

OpenAI Five also uses a policy to determine actions given an observation, but there are a few exceptions to this, which are determined by a hand-scripted logic [30].

Each time step, observations get processed and passed through a LSTM, which

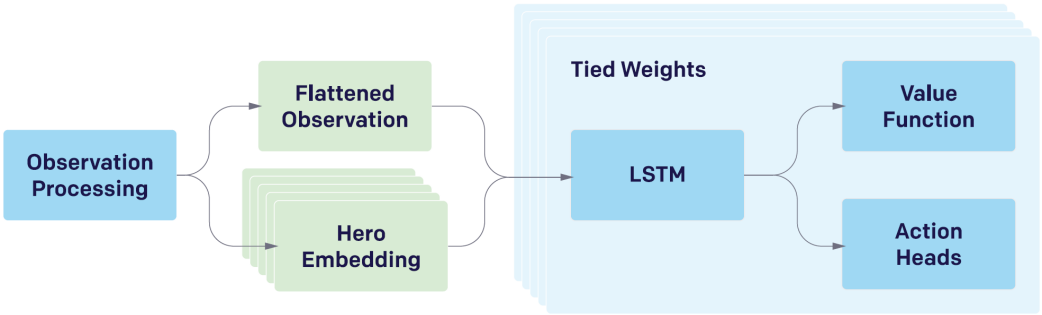

Fig. 3. Simplified OpenAI Five Model Architecture [30]

then produces an output action (and also and output from the value function). Each of the five heroes is controlled individually, meaning five similar agents are being used simultaneously.
The policy is trained using Proximal Policy Optimization (PPO) [12].

3 Similarities

All the considered AI agents were created using differing reinforcement learning methods. All AI agents were created using some sort of filtering of the data. All observations together contain a lot of redundant or irrelevant information. Filtering the data appropriately allows for faster training. Also, OpenAI Five and Atari AI have in common that they both only act on every 4th frame for simplification.
DeepMind's AlphaStar was created using a self-attention architecture, allowing to focus on the relevant information from the input data. OpenAI Five was created using
All AIs use a structure to retain temporal information. This is important as information from previous observation is necessary to describe the current game state, because game states are not independent.
DeepMind uses experience replay for remembering sequences of actions in Atari games. AlphaStar and OpenAI Five both use a long short-term memory system to deal with this problem.

4 Conclusion

We have looked at three different examples of complex AI agents for playing video games. Reinforcement learning in general seems to be the basis for training a modern video game AI.
An important property of training AI is training time, and thus not only optimizing the chosen AI architecture but also parallelizing the computations should be utilised. This can also allow for the use of AI under more demanding conditions in a different environment, or alternatively allowing the use of more computationally expensive and powerful models [23].

References

1. About Atari – Atari®, https://atari.com/pages/about
2. AlphaStar: Mastering the real-time strategy game StarCraft II - Google DeepMind, https://deepmind.google/discover/blog/alphastar-mastering-the-real-time-strategy-game-starcraft-ii/
3. AlphaZero: Shedding new light on chess, shogi, and Go - Google DeepMind, https://deepmind.google/discover/blog/alphazero-shedding-new-light-on-chess-shogi-and-go/
4. Artificial intelligence - Wikipedia, https://en.wikipedia.org/wiki/Artificial_intelligence
5. Artificial Intelligence (AI) Coined at Dartmouth | Dartmouth, https://home.dartmouth.edu/about/artificial-intelligence-ai-coined-dartmouth
6. Atari 2600 – Wikipedia, https://de.wikipedia.org/wiki/Atari_2600
7. Dota 2 – Wikipedia, https://de.wikipedia.org/wiki/Dota_2
8. Game - StarCraft II Official Game Site, https://starcraft2.blizzard.com/en-us/game
9. Machine learning - Wikipedia, https://en.wikipedia.org/wiki/Machine_learning
10. Multiplayer online battle arena - Wikipedia, https://en.wikipedia.org/wiki/Multiplayer_online_battle_arena
11. OpenAI Five defeats Dota 2 world champions, https://openai.com/research/openai-five-defeats-dota-2-world-champions
12. Proximal Policy Optimization, https://openai.com/research/openai-baselines-ppo
13. Real-time strategy - Wikipedia, https://en.wikipedia.org/wiki/Real-time_strategy
14. Reinforcement learning - Wikipedia, https://en.wikipedia.org/wiki/Reinforcement_learning
15. StarCraft II – Wikipedia, https://de.wikipedia.org/wiki/StarCraft_II
16. Was ist maschinelles Lernen? | IBM, https://www.ibm.com/de-de/topics/machine-learning
17. Was ist Reinforcement Learning? – Reinforcement Learning erklärt – AWS, https://aws.amazon.com/de/what-is/reinforcement-learning/
18. What are Neural Networks? | IBM, https://www.ibm.com/topics/neural-networks
19. What is AI inferencing? | IBM Research Blog, https://research.ibm.com/blog/AI-inference-explained
20. Starcraft II, Blizzard Entertainment (2010)
21. Dota 2, Valve (2013)
22. et al., S.A.: Parallel Computing Research at Illinois: The UPCRC Agenda; page 6 (Nov 2008), https://web.archive.org/web/20180111165735/https://graphics.cs.illinois.edu/sites/default/files/upcrc-wp.pdf
23. Clemente, A.V., Castejón, H.N., Chandra, A.: Efficient Parallel Methods for Deep Reinforcement Learning (May 2017), http://arxiv.org/abs/1705.04862, arXiv:1705.04862 [cs]
24. Ertel, W.: Grundkurs Künstliche Intelligenz. Springer Fachmedien Wiesbaden, Wiesbaden (2016). https://doi.org/10.1007/978-3-658-13549-2, http://link.springer.com/10.1007/978-3-658-13549-2
25. Fujimoto, S., Meger, D., Precup, D.: Off-Policy Deep Reinforcement Learning without Exploration
26. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Computation **9**(8), 1735–1780 (Nov 1997). https://doi.org/10.1162/neco.1997.9.8.1735, https://ieeexplore.ieee.org/abstract/document/6795963, conference Name: Neural Computation

27. Lin, L.J.: Reinforcement Learning for Robots Using Neural Networks
28. McCarthy, J., Minsky, M.L., Rochester, N., Shannon, C.E.: A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955. AI Magazine **27**(4), 12–12 (Dec 2006). https://doi.org/10.1609/aimag.v27i4.1904, https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1904, number: 4
29. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing Atari with Deep Reinforcement Learning (Dec 2013), http://arxiv.org/abs/1312.5602, arXiv:1312.5602 [cs]
30. OpenAI, Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., Pinto, H.P.d.O., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., Zhang, S.: Dota 2 with Large Scale Deep Reinforcement Learning (Dec 2019), http://arxiv.org/abs/1912.06680, arXiv:1912.06680 [cs, stat]
31. Rich, E.: Artificial intelligence and the humanities. Computers and the Humanities **19**(2), 117–122 (Apr 1985). https://doi.org/10.1007/BF02259633, https://doi.org/10.1007/BF02259633
32. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized Experience Replay (Feb 2016), http://arxiv.org/abs/1511.05952, arXiv:1511.05952 [cs]
33. Smolyakov, I.Y., Belyaev, S.A.: Design of the Software Architecture for Starcraft Video Game on the Basis of Finite State Machines. In: 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). pp. 356–359 (Jan 2019). https://doi.org/10.1109/EIConRus.2019.8656866, https://ieeexplore.ieee.org/abstract/document/8656866, iSSN: 2376-6565
34. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, , Polosukhin, I.: Attention is All you Need. In: Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
35. Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J.P., Jaderberg, M., Vezhnevets, A.S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T.L., Gulcehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Kavukcuoglu, K., Hassabis, D., Apps, C., Silver, D.: Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature **575**(7782), 350–354 (Nov 2019). https://doi.org/10.1038/s41586-019-1724-z, https://www.nature.com/articles/s41586-019-1724-z, number: 7782 Publisher: Nature Publishing Group
36. Yannakakis, G.N.: Game AI revisited. In: Proceedings of the 9th conference on Computing Frontiers. pp. 285–292. ACM, Cagliari Italy (May 2012). https://doi.org/10.1145/2212908.2212954, https://dl.acm.org/doi/10.1145/2212908.2212954

Memory considered harmful: The Rowhammer vulnerability in DRAM

Péter Bohner[1]

Karlsruher Institut für Technologie, 76131 Kalrsuhe, Germany

Abstract. The Rowhammer vulnerability is a fundamental issue of Dynamic Random Access Memory *(DRAM)*, whereby repeated access of a row of memory may corrupt adjacent parts of the memory. Despite being known for a decade, mitigations against it are still an active area of research and newly produced DRAM modules are still affected. In this paper, we will explain the vulnerability as well as and exploitation and mitigation strategies.

Keywords: DRAM · Hardware Security · RowHammer

1 Introduction

Dynamic Random Access Memory *(DRAM)*, has long served as a fundamental cornerstone of modern computing, providing fast and volatile data storage for a myriad of applications, most notably as the main memory of almost all computers. Preservation of memory integrity is paramount for reliable and secure operation, as both computer hardware and software assume that RAM is infallible. That is, the data read from memory is the same as the data written.

However, with ever increasing memory density, faults stemming from the fundamental construction of DRAM, called *disturbance errors*, have re-emerged[8]. These can be reliably triggered and thus exploited by the *Rowhammer* vulnerability, which was publicized in 2014[8][1]. Rowhammer leverages the repeated activation of rows in memory to induce disturbances manifesting in bit flips in adjacent rows. Because adjacent rows of memory may lie in different *security contexts* (different processes, sandboxes, virtual machines *(VM)*, etc.), Rowhammer allows an attacker to violate a system's security guarantees.

Rowhammer is far more than a theoretical vulnerability, with publicly available proof-of-concept exploits, including process privilege escalation and cross-VM side channels, having been demonstrated.[12][6][15]. It is especially dangerous, because exploiting the vulnerability only relies on accessing memory, which any code no matter how unprivileged must do. For example, reference [6] introduces a functional exploit from Javascript running inside a browser. Despite being known for almost a decade, research into this vulnerability is still ongoing[2] with new mitigations being proposed in 2023[11]. As the vulnerability has been

[1] The industry has been aware of it since at least 2012[8]

[2] primarily at ETH Zürich and TU Graz, see references [7],[10],[11][13]

present in memory produced since 2010[8] and and up to the present day[3][7], and such memory is used in countless devices, Rowhammer will likely remain relevant for years to come.

This work aims to explain the rowhammer vulnerability in an accessible manner and delves into exploitation and mitigation techniques.

2 DRAM Basics

This section provides a basic overview of the operating principle and structure of DRAM, which is sufficient to understand the rowhammer vulnerability. It is mainly adapted and summarized from [8].

2.1 Operation and Structure of Dynamic RAM

DRAM stores individual bits of data in a memory *cell* comprising of a *capacitor*, which can be charged or discharged (this is the state of the cell), and an *access-transistor*.[8] These cells are arranged in a two-dimensional grid, called *subarrays*[11]. Each horizontal wire (*wordline*) can be used to *activate* a row of this grid (by pulling it high), allowing each cell's charge to be transferred through the vertical wires (*bitlines*) to the corresponding bit in the *row-buffer*. This buffer is used to fulfill all accesses to the row until another row needs to be accessed (and therefore the current row needs to be closed), at which point the data in the buffer is written back to the row (as activation destroys the data in the cell) through the bitlines, and the wordline is pulled low.[15] Closing a row is also referred to as *precharging*. A group of subarrays sharing a row-buffer is known as a *bank*. Multiple banks are grouped together into a *rank*. A DRAM *module* consists of one or more ranks. The memory controller of a system can issue commands to each rank of memory in parallel, thus increasing parallelism by allowing multiple accesses to take place at the same time.[8] In most desktop and server computer architectures, the memory controller has multiple *channels* to each of which multiple Dual Inline Memory Modules *(DIMMs)* can be connected. Each of these channels are completely independent of each other, further multiplying memory throughput. [15]

It is important to note that the memory controller addresses RAM by *channel, rank, bank, row, and column*[7] and is unaware of the physical layout of rows and columns inside the dram chips (the subarrays and thereby the physical proximity in 3D-space of DRAM rows). The memory controller uses the DDR protocol (see 1) to communicate with each rank of memory. [11][7]

2.2 Refreshing

Real electrical components, including those from which DRAM is made, are not perfect. This means that DRAM cells leak charge over time. Therefore there

[3] DDR4 is still in production

Command	Addressing	Description
ACT	Bank, Row	activate (open) a row
READ	Bank, Column	read a column
WRITE	Bank, Column	write a column
PRE	Bank	Precharge (close) a row
REF	-	Refresh multiple rows

Table 1. The DDR protocol[8][11][7]

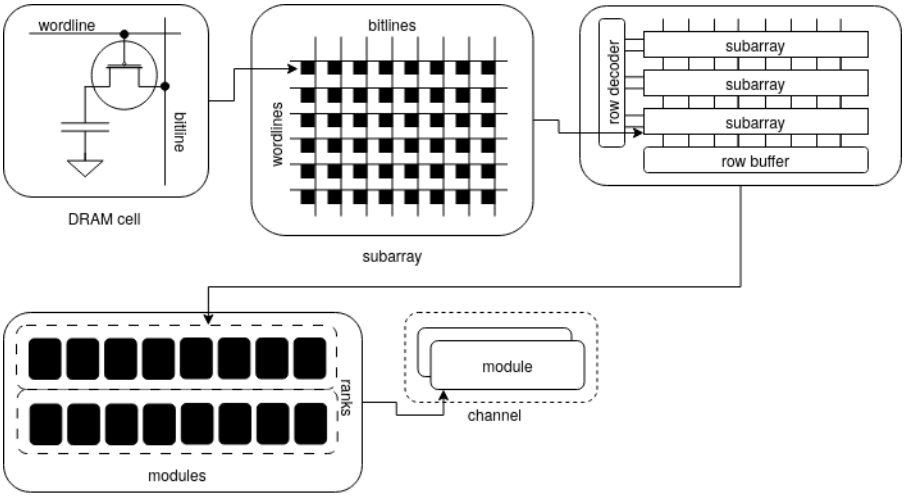

Fig. 1. Structure of DRAM, adapted from [8] and [7]

exists is a finite time, known as the *retention time*, before the capacitor voltage falls below a point where the states can be reliably differentiated (the so-called *noise-margin*), and data loss occurs. Each row needs to be periodically refreshed (activated and rewritten) in an interval shorter than the retention time. According to the JEDEC DDR3 specification, the retention time must be at least $64ms$.[8] There exists a separate REF command1, which refreshes several rows at once. It needs to be issued by the memory controller frequently enough that each row can be refreshed within the retention time. For DDR3 this requires 8192 REF commands, meaning one every $7.8\mu s$[8]

3 Disturbance Errors

An interference of a memory cell in another cell's operation is called a *disturbance*. If a disturbance is sufficiently large, the disturbed cell is pushed over it's noise margin, thereby experiencing a fault called a *disturbance error*.[8]. Disturbance errors have been known about since the first DRAM chips and have had hardware mitigations in place.[8][13]

As DRAM density has increased, three mechanisms have lead to memory cells being more susceptible to disturbance errors[8][15][5]: *Firstly,* each individual memory cell has to become smaller, which means each capacitor is smaller. The smaller a capacitor, the less charge it can hold, which reduces the cell's noise-margin, thereby increasing it's affinity for data loss. [8] *Secondly,* the distance between cells decreased, which increased the level of parasitic electrical interaction between cells, which can lead to data loss as well[15]. *Thirdly,* changes in process technology increased the number of cells susceptible to intercell crosstalk.[8]

4 The Rowhammer Bug

Yoongu, et al.[8] found that repeated toggling of a wordline (the *attacker*-row) may induce disturbance errors in memory cells located in nearby rows (the *victim*-row). [4] This can be achieved by alternating DRAM accesses to different rows of the same DRAM rank (*hammering*). To do this on a computer, a simple program can be written, which alternately reads from two memory addresses (X and Y) located on different rows, see figure 2 below. Because all modern processors have memory caches and out-of-order execution, this example needs to employ two tricks for this access pattern to occur. First, it uses the x86 *CLFLUSH* instruction to flush the caches after each write to both addresses. Second, it uses the *MFENCE* instruction to prevent the CPU from reordering memory accesses.[8]

```
loop:
mov (X), %eax
mov (Y), %ebx
clflush (X)
clflush (Y)
mfence
jmp loop
```

Fig. 2. x86 assembly Rowhammer example program from Yoongu, et al.[8]

Some cells are much more susceptible to hammering than others[13][8]. As tested by Yoongu et al [5], over 70% of cells that experienced a single error, had errors in 10 of 10 test iterations. This means, that there exist *victim cells* for rowhammer, which can be reliably attacked.[12][6]

[4] Crucially, access patterns not resulting in frequent toggling of wordlines did not cause disturbance errors. This is important for mitigation techniques.

[5] See section 7

[6] Interestingly, these are not correlated to *weak* dram cells, which are the first to lose data when not refreshed.[8]

5 Exploitation Challenges

Data loss caused by Rowhammer on its own may be undesirable, but is in itself not dangerous. It becomes a vulnerability because a program running in one security context (userspace/kernelspace, UNIX user/group, virtual machine, process sandbox, etc.) may affect memory used by another program in a different security context.[8][12], thereby violating that program's assumptions.

For example, the NaCl (native client) exploit demonstrated by [12], uses Rowhammer-induced bit-flips to alter previously validated (and deemed safe) program code. The Linux privilege escalation in the same paper uses Rowhammer to modify a processes' page table to gain write access to a SUID binary's code. These exploits are mostly built of well-known exploitation techniques (such as the latter exploit spraying kernel data structures (page tables)), using the new Rowhammer as the attack vector. Nevertheless, exploits face novel challenges described in the following sections.

5.1 Finding vulnerable addresses

Exploits need to determine memory addresses suitable for hammering, for which numerous layers of abstraction have to be reversed. The exploit needs to map memory it has access to, to the physical layout of the rows containing their data. Depending on the exploit type, this may be relatively easy, or exceptionally difficult. [14] While an exploit running on a Linux host need only use */proc/self/maps* to get page frame numbers, from which educated guesses about the physical memory layout can be made[12], a JavaScript exploit running in a browser can not even obtain the virtual addresses of it's own variables. Nevertheless, there exist multiple techniques to find vulnerable addresses from JavaScript, that have been used to develop full exploits.[14][6]

5.2 Bypassing Caches

To cause memory writes, the CPU caches must be bypassed. This can either be achieved like in 4 by directly flushing the cache, or by using an access pattern in memory that causes cache misses to the hammered addresses. The paper *Rowhammer.js*[6] goes into detail about such strategies.

6 Hammering techniques

Different ways of triggering rowhammer bit-flips have been developed over the years, including:

Random Hammering This is the basic basic variant of Rowhammer discussed in section 4, where a pair of random memory addresses located in different rows are alternatingly accessed.[8]

Double-Sided Hammering The first improvement to triggering Rowhammer comes from the first published practical exploit[12]: With double-sided hammering, instead of accessing two random rows, the row above and below the victim row are accessed alternatingly (these are the aggressor rows). This produces many more bit-flips than the random method, making the attack viable on more devices.[12] This technique does however require knowledge about the underlying memory geometry, to determine the suitable addresses.

Many-Sided Hammering The logic of double-sided hammering may be generalized to an arbitrary amount of adjacent addresses.[5]

HalfDouble HalfDouble is a hammering technique developed to bypass Rowhammer mitigation techniques which look for the aggressor-victim pairs described above. It uses four rows; the *far aggressor row*, located 2 rows away from the *victim row*, the *near aggressor row* located adjacent to the victim row and *the decoy row*, located away from the other rows. By alternatingly reading from the far aggressor row and the decoy row, with an infrequent (once every 1000 to 10000 iterations) read to the near aggressor row interspersed, bit-flips can be achieved in the victim row[9]. This technique massively complicates Rowhammer mitigations, as it disproves the assumption made by early research[8][12][6] that bit-flips may only be introduced by adjacent rows.

Blacksmith Blacksmith is an even more advanced hammering technique specifically designed to bypass TRR7.4 mechanisms/heuristics, by creating complex varying access patterns. It does this by mapping the signal properties of *phase*, *amplitude*, and *frequency* to memory accesses then fuzzing these values. The phase describes when the address is accessed (in relation to the others), the amplitude describes the amount of consecutive activations and the frequency describes the distribution of accesses over time. [7] With these access patterns, a significant number of bit-flips can be reliably introduced into DDR4 and LPDDR4X memory modules, which already have hardware-level mitigations against Rowhammer in place.[7][13]

7 Mitigations

Mitigations of Rowhammer can broadly be divided into three categories depending their place of implementation; purely software based, memory controller based, and in-DRAM approaches. The following describes and evaluates some of the most well-known mitigations in these categories.

Disabling cache flushes Looking at the code in 2, one may assume that making cache flushes a privileged instruction would mitigate the rowhammer vulnerability. Unfortunately, this is not sufficient, as multiple CLFLUSH-free attacks

have been demonstrated[13], including ones made from interpreted languages (e.g., JavaScript: *Rowhammer.js*)[6]. Rowhammer has also been demonstrated on ARM devices[16], where cache flushes are privileged. Restricting cache flush instructions has however made exploitation harder by reducing the number of bit-flips that can be achieved.[6] Therefore, it *may* be a useful mitigation, being comparable to restricting the *RDTSC*(read time stamp counter) instruction following *Meltdown* and *Spectre*.[7]

Shortening refresh intervals The simplest change to reduce the frequency of Rowhammer errors is to increase the refresh rate, as this is usually configurable by firmware. This was also the first mitigation to be implemented on devices, via BIOS updates[12].

The issue with this mitigation arises from the fact that sufficient shortening of the refresh rate incurs a very large performance overhead. To eliminate rowhammer bit-flips in DDR3, a refresh rate of as low as $8.2ms$ was necessary, a 7.8 fold increase in refreshes. This would result in the memory spending 11.0–35.0% of its time refreshing (up from the current 1.4–4.5%)[8]. A $> 30\%$ performance penalty is of course unacceptable. This was with basic single-sided hammering. With double-sided hammering[12] and modern DRAM being much more vulnerable to disturbance errors[13][11], it is likely that far shorter refresh rates would be necessary to fully protect modern memory from rowhammer, if this were the only mitigation implemented. Therefore, modern systems generally use the standard $64ms$ refresh rate or have halved the refresh rate to $32ms$, and mostly rely on other mitigations.

7.1 Error Correction Code (ECC) Memory

Rowhammer is far from the first source of memory corruption. Bit-flips occur (although rarely) during normal DRAM operation. [8]. To guard against this, mission-critical workstation and server applications have long used Error Correction Code (*ECC*) memory. ECC memory modules contain additional bank(s) of memory to store redundancy (error correction) information.[4] ECC memory should prevent rowhammer attacks from working, as bit flips can be corrected or at least detected. While ECC memory is effective in preventing almost all published exploits, it can be defeated, see 8. ECC memory also has other weaknesses, which lead to it being seldom used in consumer devices. Most importantly, storing redundancy information incurs a sizeable storage overhead, resulting in higher costs as more physical DRAM chips are needed for the same capacity. [10][4][8] Most ECC implementations reside in the memory controller, which needs to explicitly support it. This support is lacking from Intel's consumer CPUs. The parity calculations of ECC also result in a slight increase in latency and power consumption[4], which is undesirable in power-constrained mobile devices.

[7] RDTSC is helpful for developing Rowhammer attacks as well[15][14]

[8] They can be caused by EMI, high temperatures and even cosmic rays[4]

7.2 ANVIL

Rowhammer attacks have peculiar memory access characteristics, notably a high cache miss rate and high spatial locality of DRAM row accesses.[2] ANVIL is a software based mitigation in form of a Linux kernel module that uses hardware performance counters found in modern CPUs to identify rowhammer attacks and preemptively issue memory reads to potential victim rows. It achieves this by monitoring the last-level cache miss count and sampling memory accesses with these counters. [2][16] In the paper that proposes it, ANVIL is effective in mitigating Rowhammer, but there is little other data on it, aside of concerns with implementing ANVIL to guard against DMA-based[9] Rowhammer attacks on ARM[16]. Judging from the working principle of ANVIl, it seems highly likely that ANVIL would fail to protect against HalfDouble and Blacksmith.

7.3 Probabilistic Adjacent Row Activation (PARA)

Probabilistic Adjacent Row Activation (*PARA*) is the first low-overhead mitigation of Rowhammer. It was proposed by *Yoongu et al* in the same paper that discovered Rowhammer[8]. PARA is a mitigation performed by the memory controller, wherein upon closing a row (PRE command), the controller sometimes (with a low probability) opens one of the adjacent rows of the closed row. With this, it is unlikely that a victim row may be subject to sufficient hammering to cause bit-flips, without being refreshed. The paper showed that a probability of 0.1% for opening a row is sufficient, which resulted in a worst-case performance overhead across multiple benchmarks of $< 0.8\%$.[8]

PARA's other advantage is that it is stateless, requiring no additional metadata to be stored. And unlike TRR implementations discussed later, it can not be defeated by adversarial attack patterns, because there is no attack detection algorithm to trick. Unfortunately, PARA has a lot of downsides and is not a sufficient mitigation against Rowhammer. First, it requires the memory controller to know the physical layout of the memory to determine what rows are physically *adjacent* to each other. With most memory this is however not possible, as the geometry of the DRAM subarrays is hidden from the memory controller.[13] Second, PARA only protects from hammering coming from direct neighbours. New attacks such as *HalfDouble*[9] (see 6) have shown that Rowhammer attacks with a *blast-radius* of > 1 [11] are possible.

7.4 Targeted Row Refresh (TRR)

Targeted Row Refresh (*TRR*) is an umbrella term for all techniques that selectively refresh rows that are considered to be by some metric to likely be under a rowhammer attack. This includes memory controller based implementations such as Intel's *pTRR*, of which very little is known[5][13], but which suffers from the same issues lacking necessary information about the physical DRAM layout

[9] Direct Memory Access

as PARA implementations, and is therefore not fully able to protect against Rowhammer[5].

In-DRAM TRR was introduced into the DDR4 specification and is the "official" solution to Rowhammer[2][5] It consits of two parts: 1. The maximum activation count (MAC) is a value reported by the module to the memory controller, which sets the maximum number of times a row may be activated in a refresh interval. 2. The sampler, a part inside the module, which determines and keeps track of potential aggressor rows. If the MAC is exceeded for a row, the memory controller must issue activations to adjacent rows.[13][5] The implementation of TRR is DRAM manufacturer specific and little was known about the inner workings of this mechanism until the publication of TRRespass[5]. That paper reverse-engineered the secret algorithms used by DRAM multiple vendors and constructed multi-sided hammering patterns that reliably defeated these TRR implementations.

Therefore, even though TRR is the primary solution adopted by the industry to fix Rowhammer, current implementations have proven to insufficient to protect against Rowhammer attacks.

8 Defeating ECC

ECCploit is an exploit capable of inducing bit-flips in ECC protected systems. In modern systems ECC is generally handled at the memory controller level, which uses of the DRAM to store additional error correcting information, to be able to detect and correct bit-flips. One can think of storing k bits of data and r redundancy bits in a code-word of size $n = k + r$. The error correcting algorithm can then reconstruct the valid data word, even if the code-word has some errors.[4] The redundancy depends on the algorithm used, but all have a maximum amount of errors they can reliably detect and correct. For example, *Extended Hamming Codes* (commonly taught in computer science) are *Single Error Correcting, Double Error Detecting* (SECDEC), meaning any single bit-flip can be corrected and two bit-flips may be detected. Three or more bit-flips in a code-word may not be noticed.[4]

The key for exploiting Rowhammer in an ECC system is to cause bit-flips in the victim row, such that it remains a valid ECC codeword. This is not feasible to exploit by random-chance. To achieve this, the authors reverse-engineered the proprietary ECC algorithms of several CPUs and found a novel side-channel to determine if a bit had been flipped and corrected: When an error correction occurs, the access latency is higher. Using this side-channel, Rowhammer-vulnerable bits can be identified through exhaustive hammering of the available address space. Combined with the now known ECC algorithm, ECCploit can find a set of bit-flips that result in a valid code word when triggered.[4]

ECCploit is not a fast attack, requiring half an hour when the bit-flips can be directly observed or a week otherwise. Nevertheless, it has proven that ECC alone is not sufficient to fully protect a system from Rowhammer.[4]

9 Impact

After almost a decade since the discovery of Rowhammer, research into it is still ongoing, with new attacks and mitigations having been published recently[13]. Therefore, Rowhammer is still a relevant class of vulnerability. There are only four CVEs (see table 2) directly associated with Rowhammer attacks, only one of which is a full software exploit (*CVE-2022-42961*), even though many more attacks have been published in academia[6][14][12][13][16].[10] Despite proof-of-concept exploits being available for a lot of different platforms and targets [6][16][12], there are no reports of any malware exploiting rowhammer in the wild.[3][11] This is in stark contrast to other high-profile hardware vulnerabilities, like Spectre and Meltdown, where malware development was quick and widespread.[1]

All in all, Rowhammer is a fundamental flaw in a ubiquitous technology, which has proven to be exceptionally difficult to comprehensively mitigate, which is why it is still a great threat to information security.

CVE Designation	CVSS-3.1 Score	Description
CVE-2015-0565	10.0 critical	NaCl allows clflush instruction [12]
CVE-2020-10255	9.0 critical	TRRespass, see **??**
CVE-2021-42114	8.3 high	BLACKSMITH, see 6
CVE-2022-42961	5.3 medium	ECDSA key disclosure from wolfSSL

Table 2. CVEs related to Rowhammer[3]

References

1. Armasu, L.: Hundreds of meltdown, spectre malware samples found in the wild. https://www.tomshardware.com/news/meltdown-spectre-malware-found-fortinet,36439.html (2018), [Online; Accessed on 29.12.2023]
2. Aweke, Z.B., Yitbarek, S.F., Qiao, R., Das, R., Hicks, M., Oren, Y., Austin, T.: Anvil: Software-based protection against next-generation rowhammer attacks. In: Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems. p. 743–755. ASPLOS '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2872362.2872390, https://doi.org/10.1145/2872362.2872390
3. Byers, R., Turner, C., Brewer, T.: National vulnerability database. Tech. rep., National Institute of Standards and Technology (2024). https://doi.org/10.18434/M3436

[10] This is one of the reasons why the CVE system is frequently criticised as a metric for problem severity

[11] The author's search did not turn up any results for such malware

4. Cojocar, L., Razavi, K., Giuffrida, C., Bos, H.: Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks. In: S&P (May 2019), Paper=https://download.vusec.net/papers/eccploit_sp19.pdfSlides=https://www.ieee-security.org/TC/SP2019/SP19-Slides-pdfs/Lucian_Cojocar_Exploiting_Correcting_Codes_slides-ecc-new.pdfWeb=https://www.vusec.net/projects/eccploitPress=https://bit.ly/2UcucNv, best Practical Paper Award, Pwnie Award Nomination for Most Innovative Research
5. Frigo, P., Vannacci, E., Hassan, H., van der Veen, V., Mutlu, O., Giuffrida, C., Bos, H., Razavi, K.: TRRespass: Exploiting the Many Sides of Target Row Refresh. In: S&P (May 2020), Paper=https://download.vusec.net/papers/trrespass_sp20.pdfSlides=https://download.vusec.net/slides/trrespass_sp20.pdfWeb=https://www.vusec.net/projects/trrespassCode=https://github.com/vusec/trrespassPress=https://bit.ly/2UXWKJ4, best Paper Award, Pwnie Award for Most Innovative Research, IEEE Micro Top Picks Honorable Mention, DCSR Paper Award
6. Gruss, D., Maurice, C., Mangard, S.: Rowhammer.js: A remote software-induced fault attack in javascript. In: Caballero, J., Zurutuza, U., Rodríguez, R.J. (eds.) Detection of Intrusions and Malware, and Vulnerability Assessment. pp. 300–321. Springer International Publishing, Cham (2016)
7. Jattke, P., Van Der Veen, V., Frigo, P., Gunter, S., Razavi, K.: Blacksmith: Scalable rowhammering in the frequency domain. In: 2022 IEEE Symposium on Security and Privacy (SP). pp. 716–734 (2022). https://doi.org/10.1109/SP46214.2022.9833772
8. Kim, Y., Daly, R., Kim, J., Fallin, C., Lee, J.H., Lee, D., Wilkerson, C., Lai, K., Mutlu, O.: Flipping bits in memory without accessing them: An experimental study of dram disturbance errors. In: 2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA). pp. 361–372 (2014). https://doi.org/10.1109/ISCA.2014.6853210
9. Kogler, A., Juffinger, J., Qazi, S., Kim, Y., Lipp, M., Boichat, N., Shiu, E., Nissler, M., Gruss, D.: Half-double: Hammering from the next row over. In: Butler, K.R.B., Thomas, K. (eds.) 31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022. pp. 3807–3824. USENIX Association (2022), https://www.usenix.org/conference/usenixsecurity22/presentation/kogler-half-double
10. Marazzi, M., Jattke, P., Solt, F., Razavi, K.: Protrr: Principled yet optimal in-dram target row refresh. In: 2022 IEEE Symposium on Security and Privacy (SP). pp. 735–753 (2022). https://doi.org/10.1109/SP46214.2022.9833664
11. Marazzi, M., Solt, F., Jattke, P., Takashi, K., Razavi, K.: Rega: Scalable rowhammer mitigation with refresh-generating activations. In: 2023 IEEE Symposium on Security and Privacy (SP). pp. 1684–1701 (2023). https://doi.org/10.1109/SP46215.2023.10179327
12. Mark Seaborn, T.D.: Exploiting the dram rowhammer bug to gain kernel privileges. https://googleprojectzero.blogspot.com/2015/03/exploiting-dram-rowhammer-bug-to-gain.html (2015), [Online; Accessed on 01.12.2023]
13. Mutlu, O., Olgun, A., Yağlıkcı, A.G.: Fundamentally understanding and solving rowhammer. p. 461–468. ASPDAC '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3566097.3568350, https://doi.org/10.1145/3566097.3568350
14. Oren, Y., Kemerlis, V.P., Sethumadhavan, S., Keromytis, A.D.: The spy in the sandbox: Practical cache attacks in javascript and their implications. In: Pro-

ceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. p. 1406–1418. CCS '15, Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2810103.2813708, https://doi.org/10.1145/2810103.2813708
15. Pessl, P., Gruss, D., Maurice, C., Schwarz, M., Mangard, S.: Drama: Exploiting dram addressing for cross-cpu attacks. p. 565–581. SEC'16, USENIX Association, USA (2016)
16. van der Veen, V., Lindorfer, M., Fratantonio, Y., Padmanabha Pillai, H., Vigna, G., Kruegel, C., Bos, H., Razavi, K.: Guardion: Practical mitigation of dma-based rowhammer attacks on arm. In: Giuffrida, C., Bardin, S., Blanc, G. (eds.) Detection of Intrusions and Malware, and Vulnerability Assessment. pp. 92–113. Springer International Publishing, Cham (2018)

Solving The Long Tail Problem with Sampling Strategies

Nathan Ridinger

Karlsruher Institut für Technologie, TECO-Pervasive Computing Systems
Supervisor: Chaofan Li

Abstract. This paper explores the Long Tail Problem in machine learning, with a focus on visual recognition Neural Networks. It examines metrics like the Gini coefficient to quantify the long-tailedness of datasets. It also investigates the efficacy of sampling strategies such as oversampling, undersampling, and data augmentation to address the imbalance and enhance model generalization across diverse classes. Despite their benefits in improving minority class representation and dataset diversity, these strategies do not completely solve the problem. They may lead to overfitting, information loss, or introduce noise, without directly addressing the data's underlying distribution or class relationship complexity.

1 Introduction

Visual recognition Neural Networks have exhibited remarkable progress in identifying and categorizing objects within images, achieving unprecedented accuracy in numerous applications. However, the effectiveness of these models is significantly dependent on the training datasets having sufficient information for each class, allowing them to learn and generalize effectively. Some Datasets like CIFAR achieve this by enforcing a perfectly balanced distribution of instances between the classes, but this is sometimes not possible[12].
For example if we look at Visual Recognition of Animals, there are a lot of species that are facing extinction which means we will lose our ability to procure new images[7] and there are Animals that keep aloof such as the giant squid. As a result of that, there are a very limited number of photographs of those animals in their natural habitat [14]. Therefore, simply including more images of those animals in the training data is often not feasible. Fig 1 illustrates the class distribution within the iNaturalist dataset, created to foster advancements in automatic image classification. This data set encompasses more than 8,000 species, with a total of 450,000 images in its training and validation sets. In particular, this dataset exhibits a significant imbalance, where a few classes comprise the majority of the images, leaving a vast number of species underrepresented. Such an imbalance poses challenges for neural networks, which may struggle to accurately recognize and classify objects from these less represented classes due to insufficient exposure during training. This is a common problem in datasets and Training sets like iNaturalist which have a large discrepancy in the number

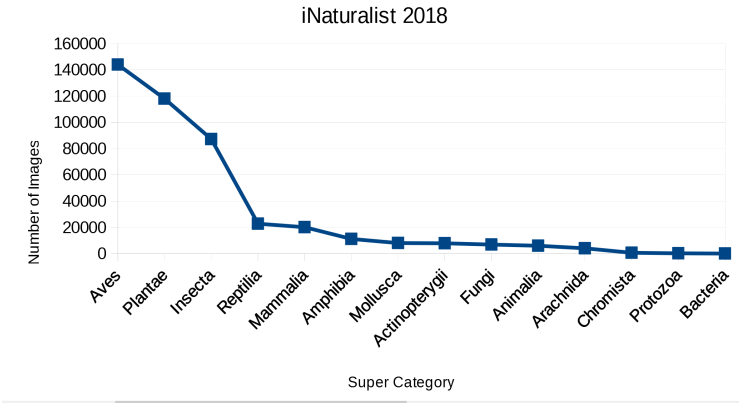

Fig. 1. The distribution of classes from the iNaturalist data Set[1]

of images between their classes are called Long Tail data sets and the attempt to solve the ramifications of those skewed data sets is called the Long Tail Problem. The problem is based on a frequency distribution which has been studied as part of statistics since 1946 [3] [19]. In Long Tailed data sets a large part of the data set corresponds to the same few classes. This part is called the Head and the corresponding class is called the Head class or frequent class and the rest of the set which has a relatively high amount of classes for a small part of the total images in the data set is called the tail and the class corresponding to this part is called tail class or rare class[16]. Although the Long Tail Problem is found in many fields from economics to healthcare in this Paper we will focus on the Long Tail Problem in Machine Learning and Visual Recognition[10].
These Long Tailed Datasets can still be useful if there aren't any alternatives but the precision of the Neural Network which is learning from the Data may suffer. To understand the ramifications of long-tailed data sets on neural networks, we first have to understand how Neural Networks learn from data. The learning process of visual recognition Neural Networks begins with initial layers detecting simple patterns such as edges, textures, and colors. As the data progresses through deeper layers, these networks assemble these rudimentary features into more abstract and sophisticated representations. These hierarchies of learned features eventually culminate in the network's ability to identify specific objects or classes.
After that, pooling layers reduce the spatial dimensions of the representation, decreasing the amount of computation needed, and aiding in creating features that are more robust to variations.
These networks owe their success to their capacity to learn intricate patterns and features directly from raw pixel data. However, their performance tends to falter when faced with the Long Tail problem.
The results of those flawed data inputs on the Neural Network can be varied. For

example, if we take Neural Networks that specialize in visual recognition and there is a significant disparity in the number of instances among different classes in a data set, one or more classes have substantially more or fewer examples compared to others. This poses a challenge during model training, where the model can bias its predictions toward the majority class due to its prevalence, leading to poor performance in the minority classes[18].
Alternatively, there are certain classes that have a wide variety of examples, while others have only a few variations or diverse instances; this is called sample imbalance. The sample imbalance refers to the unequal distribution of samples within each class. It can impact the ability of the model to generalize, especially in classes with limited representation. This can lead to overfitting on abundant examples and under performance on the diverse or rare instances within each class [8].
Although modern Visual Recognition Neural Networks excel in recognizing popular categories, their performance significantly diminishes when encountering these tail classes. Consequently, this limitation undermines the models' real-world applicability and reliability, when it comes to correctly recognizing images that are part of the tail classes. Balancing the input by providing a data set where all image classes have the same amount of data is difficult, and sometimes even impossible. The reason for that is there may not be enough images available for any given class to fully balance the data sets.
Even though the training set exhibits a Long Tail distribution, the test set is

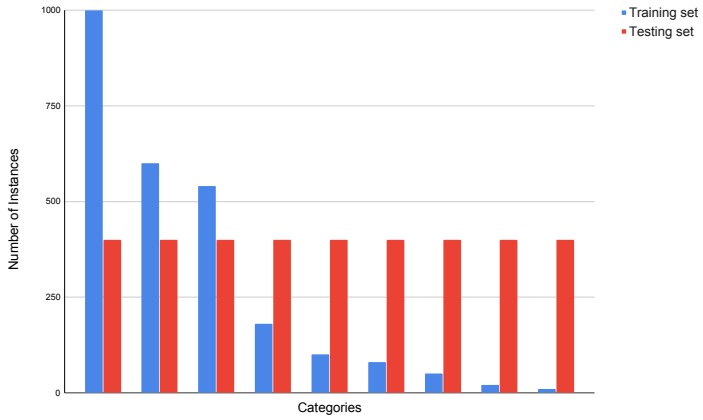

Fig. 2. Comparison between the number of images of the Training set and Test set of a typical Long Tailed data set[16]

often a balanced distribution. This can be achieved by simply repeating images in the Test class. Having the Test class be Long Tailed as well would lead to

the Neural Network ignoring the Tail classes in the Training set Since it would still achieve a high accuracy without having to learn anything about the Tail classes[16].
This paper will examine methods to enhance the representation and therefore impact of tail classes in the training process by rebalancing the training data through various resampling methods. This strategy aims to address the imbalance by adjusting the distribution of data across different classes, thereby improving the model's ability to learn from underrepresented classes. Techniques such as oversampling, which increases the presence of tail classes by duplicating existing samples, undersampling, which reduces the dominance of Head classes by removing some of their samples, and data augmentation, which artificially enlarges the dataset by creating modified versions of existing data, are pivotal in achieving a more balanced dataset. These methods not only contribute to a fairer representation of all classes but also enhance the model's generalization capabilities across a wider array of scenarios, ultimately leading to more robust and accurate visual recognition neural networks[6].

- In Section 2 we will examine the Gini coefficient as a metric to evaluate the Long Tailedness of datasets and AUC as a measure of Accuracy for Neural Networks.
- In Section 3 we will discuss different ways to alter the input data to mitigate or at least reduce the negative effects of a Long Tailed data set on our Neural Network.
- In Section 4 we will discuss the effects of our resampling techniques by evaluating the accuracy of datasets that have been rebalanced.
- and in Section 5 we will conclude the paper

2 Metrics

In order to effectively quantify the long-tailedness of datasets and to quantify the improvement in performance we can achieve by resampling the input data we first have to discuss some metrics by which to quantify those datasets. We will discuss Standard deviation, the Imbalance factor, and the Gini coefficient which is the Metric we will use to rank datasets in Section 4 on their long tailedness in this paper and we will discuss AUC as a Metric to evaluate the performance of a Neural Network.

2.1 Standard deviation

Standard deviation is a measure in statistics that quantifies the amount of dispersion of a set of data values. It is used to understand how the numbers in a data set are spread out from the mean of that set. A low standard deviation indicates that the data points are close to mean, whereas a high standard deviation indicates that the data values spread out over a wider range.

The formula for the standard deviation σ is expressed as follows

$$\sigma = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(n_i - \mu)^2} \tag{1}$$

where k represents the total number of data values in the population, n_i is each individual data values, and μ is the mean of all data values [16]. In the context of measuring long-tailedness, the number of data values represents the number of instances in any given class.

2.2 Imbalance factor

The imbalance factor calculates the ratio of the number of instances in the most frequent class to the number of instances in the least frequent class. A higher ratio suggests a more significant long tail.

$$\beta = \frac{max\{n_1, n_2, n_3, ..., n_k\}}{min\{n_1, n_2, n_3, ..., n_k\}} \tag{2}$$

A dataset is considered imbalanced if the number of instances in one class significantly outnumbers the instances in one or more of the other classes. Although the imbalance factor is used as a qauntifier for the long-tailedness of datasets it is affected by extreme classes and could be unrepresentative of the dataset as a whole.

2.3 Gini coefficient

The Gini coefficient was developed in 1912 by Corrado Gini to measure inequality of income, wealth, or consumption in a specific country or group. The inequality found in long tailed datasets is similar to the inequality between each category This means the Gini Coefficient can also quantify inequality in class distribution. The Gini coefficient is calculated based on the Lorenz curve, which plots the cumulative proportion of the total number of instances against the cumulative proportion of classes, sorted from the most populous to the least populous. The Gini coefficient is a number between 0 and 1 where a 0 represents perfect equality, meaning each class has the same number of instances and a Gini coefficient of 1 represents perfect inequality, meaning all instances belong to a single class, which is the extreme case of a long-tailed distribution.
The Gini coefficient is calculated in four steps. First, we sort our classes by the number of instances in ascending order, which means if the number of classes is k and n_i is the number of instances in the i-th class $n_i, (i = 1, 2, ..., k)|n_j \leq n_j + 1 \forall j = (1, 2, .., k - 1)$ After that we calculate the normalized cumulative distribution C_i by calculating the cumulative sum of n and normalizing it. To normalize the cumulative distribution, we divide each cumulative sum by the total number of classes k:

$$C_i = \frac{1}{k} \sum_{j=1}^{i} n_j \tag{3}$$

Calculating the normalized cumulative distribution is about turning raw cumulative sums into a proportion or percentage of the total, providing a basis for evaluating and comparing distributions in terms of equality or inequality

C_i represents the probability of drawing an instance from one of the is smallest classes if we draw an instance at random. Our smallest class starts at C_1. After we have the normalized cumulative distribution for our data set we can calculate the Lorenz curve which is a representation of C_i as follows:

$$L(x) = \begin{cases} C_i, & x = \frac{i}{k} \\ C_i + (C_{i+1} - C_i)(kx - i), & \frac{i}{k} < x < \frac{i+1}{k} \end{cases} \tag{4}$$

B represents the area to the lower right of our previously calculated Lorenz curve L(x)

$$B = \int_0^1 L(x)dx = \sum_{i=1}^{k} \frac{C_i + C_{i-1}}{2} * \frac{1}{k} \tag{5}$$

For a perfectly balanced dataset the Lorenz Curve is an identiy Line and B would be 0.5. After that we can calculate $A = 0.5 - B$ and then the Gini coefficient δ by calculating

$$\delta = \frac{A}{A + B} \tag{6}$$

Referring to Fig. 3, a perfectly balanced distribution like CIFAR [12] would result in the Lorenz curve aligning with the Line of Equality, leading to B being equal to 0.5 and the Gini coefficient δ being 0. In contrast, in a scenario where a single class covers all the data, B would reduce to 0, and consequently, δ would increase to 1. The iNaturalist data set has a Gini coefficient of 0.620
The Gini coefficient provides a concise, quantifiable measure of the distribution's long-tailedness, offering insights into the challenges and strategies needed for effective data analysis and machine learning model training on such datasets. This can be particularly useful for comparing the degree of imbalance across different datasets or evaluating the effect of balancing techniques. It is also sensitive to changes at different points in the distribution, not just extreme values. This makes it a useful tool for identifying shifts in the distribution's shape, including changes in the tail [16] .

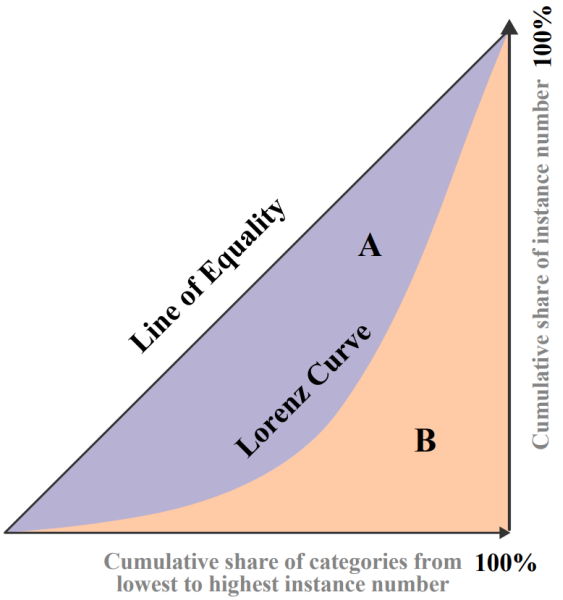

Fig. 3. A representation of a possible Lorenz Curve for a long tailed dataset[16].

2.4 Area Under the ROC Curve

AUC, or Area Under the Receiver Operating Characteristic (ROC) Curve, offers a comprehensive measure of a neural network's ability to distinguish between classes in classification tasks, particularly in binary classification. It provides an aggregate metric of performance across all possible classification thresholds, essentially quantifying the likelihood that the model correctly ranks a randomly chosen positive instance higher than a randomly chosen negative instance. To use this metric to rank a visual recognition Neural Network we have to transform the task. To transform a visual recognition problem into a binary classification problem, one typically reformulates the task so that the neural network must decide between two distinct outcomes for each input. This could involve distinguishing between images containing a specific object versus those that do not, or identifying whether an image meets a certain criterion for example "Is there a cat in this image?" Yes or No. [17] AUC assesses how well the model can distinguish between positive and negative instances. A "positive instance" refers to an example that belongs to the class of interest for example, images with cats, while a "negative instance" refers to an example that does not belong to that class so, images without cats. When the AUC is used as a metric, it essentially quantifies the likelihood or probability that, given one positive instance and one negative instance chosen at random, the classification model will assign a higher

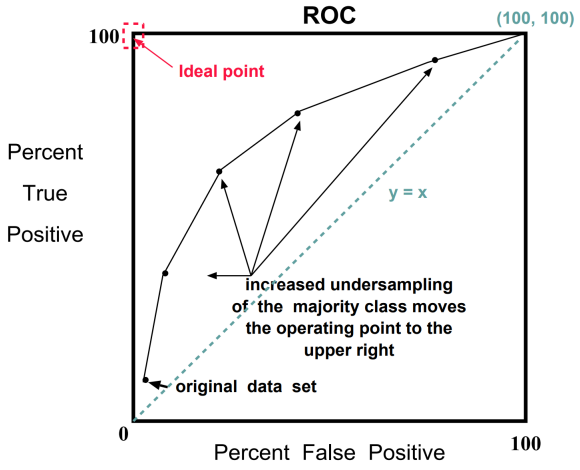

Fig. 4. Illustrating ROC curve development via increased under-sampling of the majority (negative) class shifts performance from the lower left to the upper right[5].

score or probability to the positive instance compared to the negative instance. This is a direct measure of the model's ability to differentiate between the two classes. An AUC of 0.5 implies that the model does no better than random chance at ranking positive instances higher than negative ones, indicating no discriminative power. An AUC close to 1.0 indicates a high probability that the model will rank positive instances higher than negative ones, showcasing excellent discriminative ability. Conversely, an AUC close to 0 would suggest that the model consistently ranks negative instances higher than positive ones, which could indicate a model that is performing inversely to expectations.
To calculate AUC, one first generates the ROC curve by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. The TPR (or sensitivity) measures the proportion of actual positives correctly identified, while FPR measures the proportion of actual negatives incorrectly identified as positive. The AUC is then calculated as the area under this ROC curve, which can be done using numerical integration methods.
In Chapter 4 we will measure the improvements that resampling had on the Neural Network by determining The AUC of the original Dataset and the AUC of the resampled dataset.

3 Sampling Strategies

If a dataset is imbalanced an intuitive idea is to rebalance the dataset to reduce or even eliminate the negative effect on the training of the Model. We will discuss

three ways to deal with those problems namely oversampling, undersampling, and data augmentation

3.1 Oversampling

oversampling is a common method in the field of deep learning. The simple variant of oversampling is called random minority oversampling. This technique works by augmenting the dataset through randomly replicating instances from minority classes until the dataset is balanced or reaches a balancing threshold for the class distribution, set by the user.
This method is relatively straightforward easy to implement and has been shown to improve the performance on the model on minority classes [4]. It does however have its drawbacks. Since this method only replicates existing instances it doesn't add any new information to the minority classes. This could lead to overfitting. Overfitting happens when the model learns not only the underlying patterns in the training data but also the noise and random fluctuations. The main issue with an overfitted model is that while it performs exceptionally well on the training data, it performs poorly on new, unseen data meaning it has low generalizability [5].
A better method that tries to mitigate overfitting is called Synthetic Minority oversampling Technique or SMOTE for short. More on that in the data augmentation section.

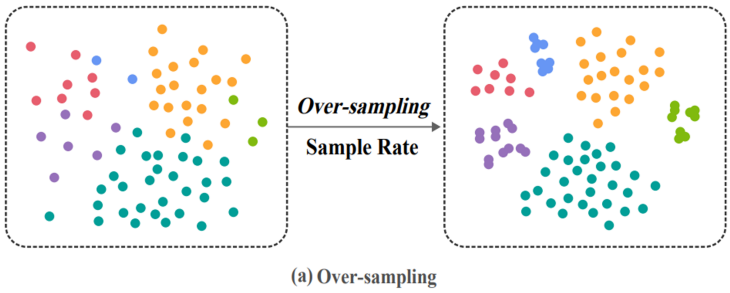

Fig. 5. Visual representation of oversampling as a method for rebalancing [16].

3.2 Under-sampling

In contrast to oversampling, under-sampling works by deleting data from the head classes. Similarly to oversampling, the basic version of Under-sampling works by randomly selecting instances of the head classes and deleting them until the number of instances in the head classes matches the number of instances in the tail classes or it reaches a treshhold set by the user [16].

While it may seem unintuitive that deleting data would improve the viability of the model there is evidence that there are situations where under-sampling is preferable to oversampling[9]. The main factor is if the original dataset is big enough. In large datasets, reducing the size of the majority class might still leave enough data to train the model effectively. Another positive point is that under-sampling is computationally less expensive and faster because it reduces the dataset size, which can be beneficial when dealing with very large datasets and limited computational resources. A disadvantage is that by randomly discard-

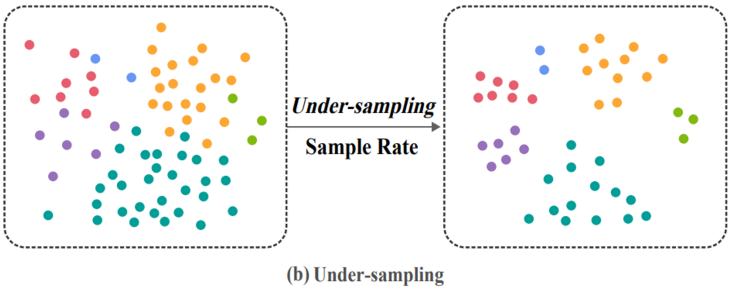

Fig. 6. Visual Representation of Undersampling as a Method for rebalancing [16].

ing instances of the head classes it runs the risk of deleting important instances and therefore reducing the viability of the model. An alternative that tries to

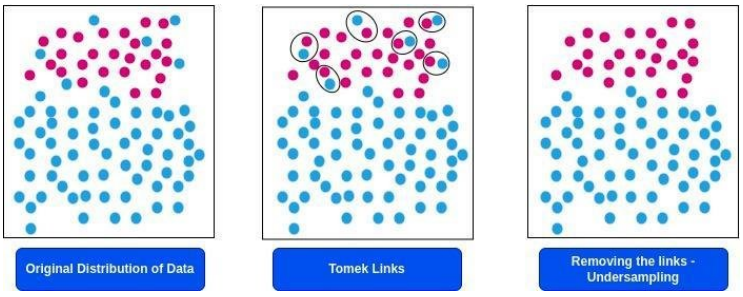

Fig. 7. Tomek Link Removal as a Method to minimize overlapp between classes [11].

address this shortcoming is called one-sided selection or OSS. The aim of OSS is to reduce the imbalance in the dataset, but in a way that also aims to preserve the most informative instances in the majority class. OSS identifies Tomek links in the dataset. A Tomek link is a pair of instances of opposite classes such that

they are each other's nearest neighbor. Removing one or both instances of each Tomek link can help in cleaning overlapping class regions and improving the separability of the classes. If the Tomek Links Removal did not delete enough instances to balance the dataset OSS further removes instances from the majority class by focusing on those that are likely to be noise or border-line examples [13].
The advantage of OSS compared to the random under-sampling is that by removing redundant and noisy instances from the majority class, OSS can help in increasing the chance of preserving informative instances, maintaining the integrity of the majority class data.

3.3 Data-Augmentation

Both Under- and native oversampling have problems. Under-sampling can lead to the Model doing worse on the head classes than it normally would have since it had less data to learn them in the training phase and critical data could be deleted and oversampling can lead to overfitting. Data Augmentation offers a promising alternative to traditional oversampling techniques by artificially increasing the size of the dataset without necessarily replicating exact copies of the minority class instances. This method enhances the diversity of the training data, which can help improve the model's generalization capabilities. By applying transformations or introducing variations to the existing data, data augmentation can generate new, synthetic examples that maintain the underlying distribution and characteristics of each class. For instance, in image processing, augmentation techniques might include rotations, scaling, cropping, or adjusting the color balance of images. In text data, it could involve synonym replacement, sentence shuffling, or translation round-trip which translates a sentence to another language and back to the original language.
The key advantage of data augmentation lies in its ability to create a more robust and diverse dataset that helps models learn a wider range of patterns within the data, reducing the risk of overfitting associated with oversampling. However, it is crucial to apply data augmentation techniques thoughtfully to ensure that the generated samples are realistic and relevant to the problem at hand. Overuse or inappropriate application of data augmentation can lead to models learning from noise or irrelevant variations, potentially harming their performance. SMOTE is a way to achieve this form of oversampling. SMOTE creates new instances which are similar to the existing ones but have small variations that are created by interpolating neighboring data points[4].

SMOTE introduces variability into the dataset by providing new information to the model. It creates new, synthetic samples, this can mitigate the overfitting problem seen with random oversampling.
However SMOTE is more complex to implement and not suitable for all data since SMOTE assumes that the feature space is meaningful, which might not be true for all kinds of data, like categorical data. It can also occur that by altering the data, SMOTE can create instances that are not realistic. This happens

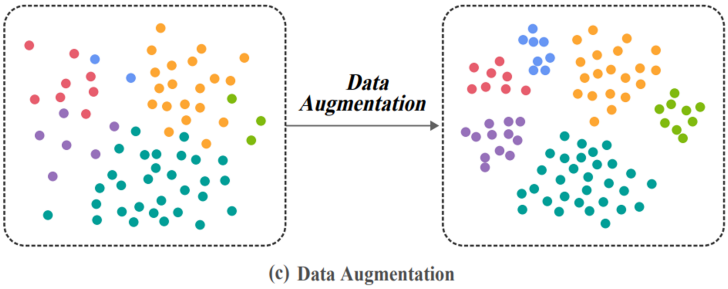

Fig. 8. Visual Representation of Data Augmentation as a Method for rebalancing [16].

especially if there is a large disparity in the number of instances between the head and tail classes[5]. A possible Algorithm to create new synthetic samples with SMOTE looks like Algorithm 1 from the Journal of Artificial Intelligence Research[5]. The algorithm begins by checking if N is less than 100%, in which case it randomizes the minority class samples since only a portion of them will be used for generating synthetic samples. It then adjusts T and N to ensure that the number of synthetic samples generated matches the specified percentage increase. The main loop of the algorithm iterates through each minority class sample, calculates its k nearest neighbors, and then calls the Populate function to generate N synthetic samples per minority class sample.
The Populate function creates synthetic samples by interpolating between the minority class sample and one of its k nearest neighbors chosen at random. For each attribute, it calculates the difference between the neighbor's attribute value and the original sample's attribute value, multiplies this difference by a random number between 0 and 1 to ensure variability, and adds the result to the original sample's attribute value. This process is repeated N times or until the desired number of synthetic samples is generated, each time incrementally adding to the index of the synthetic samples array [5].

4 Effects of Resampling on Datasets

To evaluate the effects of the resampled datasets on the efficacy of the Neural Network we have to look at the difference in performance. Table 2 shows the results of a study that evaluated the effects of different resampling techniques on the performance of 13 unbalanced datasets. The Letter and Splice datasets have been split into two datasets each: Letter-a and Letter-vowel, Splice-ie and Splice-ei[2]. The performance is meassured as an AUC score scaled up to a score of 0 to 100 while the numbers within the brackets represent the associated standard deviations. The highest performing datasets are marked in bold, with a gray background. Table 1 shows the level of imbalance prior to resampling. All

Algorithm 1 SMOTE Algorithm

Input: Number of minority class samples T; Amount of SMOTE $N\%$; Number of nearest neighbors k
Output: $(N/100) \times T$ synthetic minority class samples
if $N < 100$ **then**
    Randomize the T minority class samples
    $T \leftarrow (N/100) \times T$
    $N \leftarrow 100$
end if
$N \leftarrow \text{int}(N/100)$
$k \leftarrow$ Number of nearest neighbors
$numattrs \leftarrow$ Number of attributes
$Sample[][]$: array for original minority class samples
$newindex \leftarrow 0$: keeps count of number of synthetic samples generated
$Synthetic[][]$: array for synthetic samples
for $i \leftarrow 1$ to T **do**
    Compute k nearest neighbors for i, and save the indices in $nnarray$
    Populate(N, i, $nnarray$)
end for
procedure POPULATE(N, i, $nnarray$)
    while $N \neq 0$ **do**
        Choose a random number between 1 and k, call it nn
        for $attr \leftarrow 1$ to $numattrs$ **do**
            $dif \leftarrow Sample[nnarray[nn]][attr] - Sample[i][attr]$
            $gap \leftarrow$ random number between 0 and 1
            $Synthetic[newindex][attr] \leftarrow Sample[i][attr] + gap \times dif$
        end for
        $newindex \leftarrow newindex + 1$
        $N \leftarrow N - 1$
    end while
end procedure

Table 1. Total Number of Instances and long tailedness measure of the original datasets

Data set	Total Number of Instances	Gini Coefficient δ	Imba Factor β
Pima	768	0.15234	1.8760
German	1000	0.2	2.3333
Post-operative	90	0.2333	2.74953
Haberman	306	0.23529	2.77786
Splice-ie	3176	0.25913	3,1511
Splice-ei	3176	0.26008	3,1684
Vehicle	846	0.26478	3,2517
Letter-vowel	20000	0.30610	4,15729
New-thyroid	215	0.33721	5,1425
E.Coli	336	0.39881	8,59692
Satimage	6435	0.40272	9,27749
Flag	194	0.41237	10,41552
Glass	214	0.42056	11,59445
Letter-a	20000	0.46050	24,31645
Nursery	12960	0.47446	38,21568

Datasets get resampled until their are perfectly balanced. and have a GINI coefficient of 0. By examining the outcomes detailed in Table 2, we can gain valuable insights into which resampling techniques are most effective.

Table 2. AUC results for the original, over-sampled, and under-sampled data sets

Data set	Original	Rand Over	Smote	Smote+Tomek	Rand Under	Tomek
Pima	82.33(5.70)	86.03(4.14)	85.97(5.82)	85.56(6.02)	81.49(4.29)	83.11(4.65)
German	85.94(4.14)	85.56(4.31)	84.51(4.55)	84.02(3.94)	84.54(3.32)	85.90(3.99)
Post-operative	78.23(15.03)	71.33(23.43)	68.19(26.62)	47.99(16.61)	55.52(24.47)	66.45(23.29)
Haberman	67.91(13.76)	73.58(14.22)	75.45(11.02)	78.41(7.11)	68.40(10.17)	69.59(13.30)
Splice-ie	99.30(0.30)	99.09(0.27)	99.19(0.28)	99.13(0.31)	98.80(0.40)	99.18(0.43)
Splice-ei	99.47(0.61)	99.52(0.60)	99.52(0.26)	99.51(0.32)	99.25(0.48)	99.44(0.60)
Vehicle	98.45(0.90)	99.13(0.75)	99.04(0.85)	99.04(0.85)	97.80(0.94)	98.41(0.90)
Letter-vowel	98.81(0.33)	98.84(0.27)	99.15(0.17)	99.14(0.17)	98.26(0.28)	98.90(0.18)
New-thyroid	94.98(9.38)	98.89(2.68)	98.91(1.84)	98.91(1.84)	94.87(5.00)	94.98(9.38)
E.Coli	92.50(7.71)	93.55(6.89)	95.49(4.30)	95.98(4.21)	88.64(12.46)	94.03(5.56)
Satimage	94.82(1.18)	95.52(1.12)	95.69(1.28)	95.69(1.28)	92.86(1.29)	95.11(1.29)
Flag	76.65(27.34)	79.78(28.98)	73.87(30.34)	82.06(29.52)	78.35(29.98)	78.59(28.75)
Glass	88.16(12.28)	92.07(12.09)	91.27(8.38)	91.27(8.38)	80.47(13.25)	87.00(16.75)
Letter-a	99.67(0.37)	99.78(0.29)	99.92(0.12)	99.92(0.12)	99.46(0.42)	99.67(0.37)
Nursery	99.96(0.05)	99.99(0.01)	99.75(0.34)	99.53(0.31)	98.76(0.22)	99.89(0.08)

The findings indicate that over-sampling techniques, particularly Smote + Tomek, yield very positive outcomes, especially in scenarios where minority classes have few instances. Surprisingly, Random over-sampling, often deemed less effective, demonstrated competitive performance compared to the more sophisticated methods. It's recommended to utilize Smote + Tomek for datasets with scarcely represented minority classes, which tend to present challenges in classification accuracy for imbalanced datasets. For imbalanced datasets where the minority classes still have a large amount of instances, Random over-sampling, being less computationally demanding, remains a viable option that can still deliver significant outcomes.
It should also be noted that balancing the dataset through resampling does not always improve the results of the Neural Networks and although undersampling techniques do not reach the highest accuracy scores, they can still outperform the other resampling methods in certain cases and also enhance the computational efficiency of the Neural Network. So they are still a viable option if computational power is limited [2][15].

5 Conclusion

The Long Tail Problem poses a significant challenge in machine learning, especially in classification problems, where rare classes are underrepresented in the dataset. Sampling strategies, including oversampling, undersampling, and data augmentation, have been widely adopted as solutions to mitigate the imbalance and enhance the model's ability to generalize across both common and rare classes.

Oversampling increases the presence of minority classes by replicating their examples, undersampling reduces the dominance of majority classes by removing some of their instances, and data augmentation artificially enlarges the dataset by generating new examples through modifications of existing ones, thereby offering a more balanced view.
While these strategies undoubtedly contribute to mitigating the negative effects of the long tail problem, they do not completely eradicate them. Oversampling can lead to overfitting, where models might memorize the minority class examples rather than learning generalizable patterns. Undersampling, on the other hand, risks losing valuable information by discarding data from the majority classes, potentially undermining the model's performance on more common scenarios. Data augmentation, although beneficial in creating a more diverse training set, can introduce noise or irrelevant variations if not carefully tailored to the specific characteristics of the data and the task at hand.
Moreover, these strategies do not directly tackle the underlying distribution of the data or the complexity of the relationships between classes. They primarily focus on making the training data appear more balanced, without necessarily enhancing the model's intrinsic ability to discern subtle distinctions among classes, especially when those distinctions are not well-represented in the available data.
While sampling strategies like oversampling, undersampling, and data augmentation are valuable tools against the Long-Tail Problem, they are not always enough to create perfect datasets. Their effectiveness is contingent upon careful application and combination with other techniques, such as sophisticated model architectures, advanced feature engineering, and perhaps most importantly, the acquisition of more representative data. Thus, it is my assumption that addressing the Long Tail Problem will remain a multifaceted challenge where employing additional strategies such as Metric learning, Transfer learning, and Model design, alongside resampling the dataset, may potentially offer the most effective outcomes for the Model.

References

[1] URL: `https://github.com/visipedia/inat_comp/tree/master/2018`.

[2] Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. "A study of the behavior of several methods for balancing machine learning training data". In: *ACM SIGKDD explorations newsletter* 6.1 (2004), pp. 20–29.

[3] George W Brown and John W Tukey. "Some distributions of sample means". In: *The Annals of Mathematical Statistics* 17.1 (1946), pp. 1–12.

[4] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. "A systematic study of the class imbalance problem in convolutional neural networks". In: *Neural networks* 106 (2018), pp. 249–259.

[5] Nitesh V Chawla et al. “SMOTE: synthetic minority over-sampling technique”. In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[6] Xiaohua Chen et al. “Imagine by reasoning: A reasoning-based implicit semantic data augmentation for long-tailed classification”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 1. 2022, pp. 356–364.

[7] Mark J Costello, Robert M May, and Nigel E Stork. “Can we name Earth’s species before they go extinct?” In: *science* 339.6118 (2013), pp. 413–416.

[8] Yin Cui et al. “Class-balanced loss based on effective number of samples”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 9268–9277.

[9] Chris Drummond, Robert C Holte, et al. “C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling”. In: *Workshop on learning from imbalanced datasets II*. Vol. 11. 2003, pp. 1–8.

[10] Guo Haixiang et al. “Learning from class-imbalanced data: Review of methods and applications”. In: *Expert systems with applications* 73 (2017), pp. 220–239.

[11] Javad Hassannataj Joloudari et al. “Effective Class-Imbalance Learning Based on SMOTE and Convolutional Neural Networks”. In: *Applied Sciences* 13.6 (2023), p. 4006.

[12] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).

[13] Miroslav Kubat, Stan Matwin, et al. “Addressing the curse of imbalanced training sets: one-sided selection”. In: *Icml*. Vol. 97. 1. Citeseer. 1997, p. 179.

[14] Tsunemi Kubodera and Kyoichi Mori. “First-ever observations of a live giant squid in the wild”. In: *Proceedings of the Royal Society B: Biological Sciences* 272.1581 (2005), pp. 2583–2586.

[15] Gary M Weiss and Foster Provost. “Learning when training data are costly: The effect of class distribution on tree induction”. In: *Journal of artificial intelligence research* 19 (2003), pp. 315–354.

[16] Lu Yang et al. “A survey on long-tailed visual recognition”. In: *International Journal of Computer Vision* 130.7 (2022), pp. 1837–1872.

[17] Tianbao Yang and Yiming Ying. “AUC maximization in the era of big data and AI: A survey”. In: *ACM Computing Surveys* 55.8 (2022), pp. 1–37.

[18] Han-Jia Ye et al. “Identifying and compensating for feature deviation in imbalanced deep learning”. In: *arXiv preprint arXiv:2001.01385* (2020).

[19] Yongshun Zhang et al. “Bag of tricks for long-tailed visual recognition with deep convolutional neural networks”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 4. 2021, pp. 3447–3455.