# 2021 International Symposium on Cyber Security Cryptography and Machine Learning (CSCML 2021)

# TECHNICAL REPORT

Editors

**Shlomi Dolev and Oded Margalit**

Technical Report #21-01

July 19, 2021

The BGU-Negev Hi-Tech Faculty Startup Accelerator,

Department of Computer Science,

Ben-Gurion University, Beer Sheva, Israel

CSCML 2021

# 2021 INTERNATIONAL SYMPOSIUM ON CYBER SECURITY CRYPTOGRAPHY AND MACHINE LEARNING

JULY • 8-9• 2020
VIRTUAL CONFERENCE

**PROF. SHLOMI DOLEV**
GENERAL CHAIR

# CSCML 2021

## SPONSORS

אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

BGU NHSA

FCCS

CBG Cyber@ Ben-Gurion University of the Negev

IEEE COMPUTER SOCIETY
Israel Chapter

IBM

Springer

Lecture Notes in Computer Science
LNCS LNAI LNBI

Information and Computation

**In cooperation with**

INTERNATIONAL ASSOCIATION FOR CRYPTOLOGIC RESEARCH
iacr

# Table of Contents

**PhD Student Research Track** ..... 1

**Introduction** ..... 2

**Veritable Computing Using Computation Fingerprint Within FHE (Preliminary Version)** ……3
*Shlomi Dolev and Arseni Kalma*

**Incorporating Improved Neural Network Motion Vector Predictions in the HEVC HM Reference CODEC 27**
*Raz Birman, Shevach Riabtsev, Hananel Assor, Dor Avni, and Ofer Hadar*

**Climb of Spam in digital security: A State-of-the-Art Review of machine learning-based Decision trees, Extra Trees in tree-based model and Gaussian, Bernoulli, and Multinomial from Naïve Bayes model**...33
*Priyanka Dhara, Shubham Bhattacharjee, Sohom Bhattacharya, Subir Gupta*

**MrC: a Medical Record Chain System based on blockchain** .....42
*Harel Jerbi, Shimon Shai Idan, and Hadassa Daltrophe*

**The Time for Reconstructing the Attack Graph in DDoS Attacks** .....51
*Dina Barak-Pelleg and Daniel Berend*

**Automatic Real Time Platoon Formation** ..... 1
*Shlomi Dolev, Ehud Gudes, Ayushi Srivastava and Hannah Yair*

**Encoded video motion vectors as an alternative to a GPU processor used by OP to extract human skeletal movements** .....73
*Yoram Segal and Ofer Hadar*

**Fraud and Adversarial Machine Learning** .....91
*Medina Williams*

**Location Functions for Self-Stabilizing Byzantine Tolerant Swarms** .....93
*Yotam Ashkenazi, Shlomi Dolev, Sayaka Kamei, Fukuhito Ooshita, Koichi Wada and Yoshiaki Katayama*

**Vision based-control** .....11
*Teodor Linnik*

**Entrepreneurship Pitch Track** ..... 168

**Introduction** ..... 169

**nSure.ai** ..... 170
*Alex Zeltcer*

**TREAT** ..... 187
*Itay Katzav*

**Analytics-Model.com** ..... 188
*Idan Moradov*

**Super Polymer** ..... 204
*Hagai Ortner*

**Clean Tech by Abir Holdings Inc** ..... 212
*Solomon Ibragimov*

**A multi-agent system for managing the operating system** ..... 213
*Roie Zivan*

**MrC: a Medical Record Chain System based on Block-chain** ..... 227
*Harel Jerbi, Shimoon Shai Idan and Hadassa Daltrophe*

**V@FHE** ..... 229
*Senia Kalma*

**AnyMaint** ..... 246
*Vladimir Krasniansky*

# PhD Student Research Track
# Chair: Oded Margalit

**PhD Student Research Track chaired by Prof. Oded Margalit**

This year's CSCML was held virtually (COVID-19), but we had dozen great talks from several academic institues and diverse research fields like the emerging are of FHE (Fully Homomorphic Encryption) and BlockChain technologies, some classic machine learning works, innovative cyber security research and more.

I hope we'll have the conference face-2-face next year, but in any case, I strongly recommend all researchers in Cyber-Security, Machine Learning and Cryptology to submit their work (if relevant) and participate in the PhD track next year – it is a great way to learn about the state of the art research and receive helpful feedback on your own work.

Looking forward to CSCML 2022.

Regards,

Prof. Oded Margalit,
Computer Science department, BGU
and Cyber Security Innovation Center, Citi

# Verifiable Computing Using Computation Fingerprint Within FHE

(Preliminary Version)

Shlomi Dolev and Arseni Kalma

Department of Computer Science, Ben-Gurion University of the Negev,
Beersheba, Israel
dolev@cs.bgu.ac.il
arseni.kalma@gmail.com

**Abstract.** We suggest using Fully Homomorphic Encryption (FHE) to be used, not only to keep the privacy of information but also, to verify computations with no additional significant overhead, using only part of the variables length for verification. This method supports the addition of encrypted values as well as multiplication of encrypted values by the addition of their logarithmic representations and is based on a separation between hardware functionalities. The computer/server performs blackbox additions and is based on the separation of server/device/hardware, such as the enclave, that may deal with additions of logarithmic values and exponentiation.
The main idea is to restrict the computer operations and to use part of the variable for computation verification (*computation fingerprints*) and the other for the actual calculation. The verification part holds the FHE value, of which the calculated result is known (either due to computing locally once or from previously verified computations) and will be checked against the returned FHE value. We prove that a server with bit computation granularity can return consistent encrypted wrong results even when the public key is not provided. For the case of computer word granularity the verification and the actual calculation parts are separated, the verification part (the consecutive bits from the LSB to the MSB of the variables) is fixed across all input vectors. We also consider the case of Single Instruction Multiple Data (SIMD) where the computation fingerprints index in the input vectors is fixed across all vectors.

**Keywords:** Fully Homomorphic Encryption · Cryptography · SIMD · Verification.

## 1 Introduction

The fundamental problem of *how can a delegator verify that the delegatees performed the computation correctly, without running the computation itself?* formed in [1] and arose as early as 1992. Important works, as [2] or [3], advanced the field of computation delegation by expanding current theoretical models, such as random oracle or standard assumption models, but did not offer a practical enough solution. Other general methods, such as the ones presented in [4] consider hardware design of arithmetic

circuits. This problem was also stated explicitly in [5], "Such a mechanism of general secure computation outsourcing was recently shown to be feasible in theory, but to design mechanisms that are practically efficient remains a very challenging problem". [6] Stated that zero-knowledge proofs do exist outside the domain of cryptography and number theory, using no assumptions. One of the only practical breakthroughs, that resulted in a real working application, is the SNARK [7]. Although it is designed to be practical, and having an extremely efficient verification procedure, the complexity of creating the verification itself is poly-logarithmic to the length of the original computation and may nullify the benefit in delegating the computation. In addition to that, it only works on known (cleartext) input while typically delegation of computing is based on encrypted inputs and FHE.

Several other approaches were suggested in [8–13] verify computations with schemes involving the usage of *authentication tags*, *signatures* or different *MACs*, however none suggest to couple the result with the indication in a monolithic fashion. A separate indication on a correct computation does not imply that another instance of the computation is also correct, as presented in those schemes. Other approaches are based on check-able traces of the computation such as MPC-in-the-head [14] or the probabilistic check-able proofs (PCP) [15], where several randomly (or blindly) chosen steps in the MPC/PCP can be checked to validate the computation. These approaches either requires space overhead and/or interactions with the servers. In some scenarios a user cannot change the (e.g., the server is in a cloud computing scenario) system when an indication on wrong computation is received. Moreover, the servers may decide to act maliciously in burst, so repetition may not assist in learning the number of repetitions needed, let alone the overhead needed in repeating the computation. Moreover, we suggest a more efficient approach by coupling the computation values with the verification section in the same unit of computation, while mantaining a "preprocessing" model [16–18] which can work in a transitive manner, and in proportion of $\log_2$ to the number of inputs.

Fully Homomorphic Encryption (FHE) [19, 20] enables computation of arbitrary functions on encrypted data without knowing the secret key. Many of the FHE schemes (e.g., [21–34]) followed Gentry's suggested blueprint [19]. FHE becomes more and more practical, and in many scenarios include the evaluation of various calculation or algorithms on encrypted and sensitive data [35–39]. FHE was implemented by several open-source libraries, where the libraries we focus on are Microsoft's SEAL [40] and IBM HELib [41], each implementing different schemes from the ones stated above, where we are interested in BGV [23] and CKKS [34] schemes. Unfortunately, while FHE copes with honest but curious servers, it is not designed to cope with malicious/Byzantine [42] servers. Note that in the cloud the identity of the server that executes the delegated computing task is not known, and therefore there can be no binding of a server to (wrong) results, and later re-execution may yield the same no binding results, thus, malicious servers can try their luck forever. We propose to add computation fingerprints, that are analogous to error detection codes and can detect deviation from the requested computation. Those computation fingerprints will be computed exactly once by the delegator, by executing the requested computation on some random base values, where those values may act as a witness for future calculations of

the same circuit/program/procedure. The result of this calculation will be used, as the same base values will be coupled to all future other computation requests, and the result will be accepted if and only if the fingerprint result was received from the server as previously calculated. Along with that, we give real implementation examples of usable functions and survey possible attack models on our proposed schemes.

We start with the impossibility result, showing that an adversary can return arbitrary predefined results encrypted with the unknown key used to encrypt the FHE inputs, blindly and consistently, using various techniques. Later on, we consider restricted computation capabilities, where blackbox procedures are used while keeping the computation capabilities to compute any arithmetic circuit over the inputs. In Section 3, we present a scheme for adding *computation fingerprints* inside a FHE number, represented as encrypted bits, allowing only additions. To verify the computations we use unary or integers, along with various restrictions applied to the blackbox procedures to make our scheme safe. Following that, in Section 4 we present an additional technique, using logarithmic representations, supporting also multiplication, thus, supporting the computation of complex computation as any polynomial and therefore any arithmetic circuit. Here we also consider that the input is represented as an encrypted value (word), and not as encrypted bits like in the previous section. In Section 5 we survey an implementation example using existing FHE libraries, different number representation implications, and managing the work of values under different fields. Lastly, in Section 6 we survey operations, supporting computation fingerprints that indicate whether the required program/computation is performed on the inputs. Where unlike in the previous section the inputs can be vectors used with SIMD [43] operations, the inputs are not restricted to integers (represented using a computer word), and can be general values including floating point numbers. In Section 7 we complete with our conclusions.

## 2 Consistent Wrong FHE Results in Bit Granularity

We present an impossibility result for detecting wrong computation by repeating the computing possibly using different (FHE) keys and comparing the decrypted results when the computer is allowed to perform bitwise operations. We assume the case where the server does not have the public key, therefore we also consider the case where the computation depth does not exceed the one implying a need to bootstrap [19], proving that even in such a case the server can return an arbitrary result. We start with a small scale introduction (*blind conditioning*) and later present a scheme for the server to produce consistent wrong results to the same function. One important ingredient in our demonstration is the execution of blind conditioning.

**Blind conditioning.** One may assume that encryption prevents some possible conditioned computation, we demonstrate the opposite. We present an important feature possible on homomorphically encrypted data, which is the ability to perform *blind if* on it. This does not allow us to read the input as plaintext, rather, it lets us run calculations and conditions blindly on (encrypted) data. Namely, we demonstrate blind execution of a basic condition: if a specific bit $b$ is true (say, represented as

an encrypted 1) output $f(x)$, otherwise, output $g(x)$. We plan to use the bit $b$ and calculate the following $b \cdot f(x) + (enc(1) - b) \cdot g(x)$. The implementation is done as described in Algorithm 1, by having a specific $bit$ as the conditioned value, a function $f(x)$ for the positive case, and another function $g(x)$ for the negative case (extracting an $enc(1)$ value will be demonstrated in the next section).

**Algorithm 1:** Implementation of blind conditioning by a specific bit

**Require:** $enc(bit), f(x), g(x)$
return $(enc(bit) \cdot f(x) + (enc(1) - enc(bit)) \cdot g(x)))$

The above manipulation is based on bit representation of the encrypted value, namely, a vector representation of the data. This allows access to single bits, which offer greater flexibility in doing blind conditioning. Having access to an encrypted bit lets us blindly "figure" the bit value and act accordingly, at the cost of several multiplications and an addition of the argument built earlier. It is important to note again that we are not able to "see" the actual clear text bit value, but able to work on it and output a result dependent on it. The *blind conditioning* will be later extended to an even more elaborate technique that is also capable of implementing blind switch/case programming primitives (see Algorithm 4), rather than a single blind if.

At first glance it seems possible to repeat a computation encrypted with different keys, and find out whether the server is computing correctly, however, as we show next, the server can be answering consistently wrong answers. The ability to send consistent output can be trivially demonstrated by a policy of the server in which the server uses the input as the output. Such a policy may obviously be suspected by the computation delegator. The server may employ more sophisticated consistent wrong computations, such as, constantly adding (or constantly subtracting) as long as the operation is different from the requested computation, the input variables (in case the input consists of more than one operation).

The computation delegating party may have means to check the result. For example, checking the value of the least significant bit, to reflect the bit anticipated value. One may also occasionally compute the result to compare with the result the server sends, this still gives non-neglected probability for using hunchbacked wrong results, and in the scope of cloud computing does not necessarily reveal the malicious server [44]. Other self-testing self-correcting techniques [4, 45] can be used to verify the result. The malicious server may design and tailor a function that nullifies the benefits of (such) easy (easier than the actual calculation) attempts to check techniques. We next prove that the server can implement *any* function on the encrypted inputs and be consistent with the answers, repeating the same outputs to the same corresponding (before encryption) inputs, and also be tailored to be consistent across outputs.

We now prove that the server can execute any (wrong) function, and still, possibly return consistent outputs across inputs, coping with self-testing/correcting checks, on any encrypted input, and output an encrypted answer for the chosen function. Given an encrypted input, the server produces an arbitrary encrypted (wrong) com-

putation result of its choice. This result is encrypted with the same key used to encrypt the input, while the server has no access to the encryption (public) key itself. The bootstrapping procedure requires the public key for refreshing the ciphertext to allow unlimited number of operations, still our result concerning the ability to return consistent wrong output is stronger as we do not supply the public key to the server, thus, also address more limited HE schemes; schemes that are constrained to a certain computation depth.

To ensure that the choice of the wrong result is consistent with future queries with the same input, possibly encrypted by different key(s), the server may (explicitly or implicitly) construct in its memory a (clear-text) lookup table, mapping inputs to (specific wrong) outputs.

Every time an encrypted input is received, the server employs a procedure to return an encrypted output with the unknown key used on the encrypted input, using its predefined lookup table, even without having the public key which was used for the encryption. This means that the server can save a particular and consistent result as cleartext in its lookup table, resulting in the same selected operation for all future inputs, even if the encryption key is repeatedly changed.

The implication is that the server can, after selecting a defined output for every possible input in its lookup table, return consistently manipulated results on all future inputs.

**Implementation details.** We are assuming that a malicious server has only a single encrypted input (which can be in fact a concatenation of several inputs), for which it wants to return a certain output, according to the input, described in its clear-text Look-Up-Table (LUT).

| Input | Output |
|---|---|
| 0...00 | $Output_0$ |
| 0...01 | $Output_1$ |
| 0...10 | $Output_2$ |
| .. | .. |

**Fig. 1.** Lookup table example

For the first step, we show how a malicious server obtains an encrypted value 1 when needed, for the use of setting the selected outputs from its plaintext Look Up Table (LUT) to an encrypted form, without having the public encryption key. Note that, obviously, the avoidance in supplying the public key is limiting the manipulation power of the server, but still allows any computation, as the delegator may supply upfront all needed encrypted values as part of the inputs. Our technique uses solely FHE supported functions, multiplication and addition, to create a bitwise OR operation between two arbitrary encrypted values (bits). The implementation is based on the bit-

wise OR properties, which can be represented as OR$(x, y) = (x+y)-(x \cdot y)$, meaning that an OR operator will be implemented as a procedure with a single addition, multiplication, and subtraction. This OR operator is used upon all the bits of the encrypted input iteratively, resulting in an encrypted value 1, unless all of the encrypted input bits are 0. Obtaining an encrypted 0 value is much easier, by simply subtracting any selected input bit by itself. The case in which the input value is exactly 0 is addressed by the fact that when no encrypted 1 can be constructed, the output must be 0.

After having encrypted 1 and 0 values, the malicious server can use regular FHE operations to construct its desired encrypted output values by the received input, even without the public key. We can also negate specific bits in the encrypted input, by subtracting them from our constructed encrypted 1 value, using $Negation(enc(bit)) = enc(1) - enc(bit)$.

The next step will be the construction of a calculation that allows us to blindly find the correct row, in terms of the received encrypted input, and return our desired encrypted output. This is made blindly by forming an argument returning a true (1) value that is dependent on the input, for each possible row in the LUT. For every row in the LUT, the argument will consist of multiplication of every bit, or negated (*neg*) bit of the input, in a way that the multiplications will equal 1 if and only if the input matches the row input value. This makes the argument equal to 1 if and only if the input has the corresponding bits set to true or false according to the row input value, and to 0 otherwise. By multiplying the expected output at the end of the argument, we set the row value accodingly, if it was the correlated row input value. Thus, we (blindly) get the output of the row if the inputs match the row and 0 otherwise.

For example, for the first 2 rows in the LUT, we construct the following arguments:

$$neg(bit_n) \cdot \ldots \cdot neg(bit_2) \cdot neg(bit_1) \cdot neg(bit_0) \cdot Output_0$$

$$neg(bit_n) \cdot \ldots \cdot neg(bit_2) \cdot neg(bit_1) \cdot enc(bit_0) \cdot Output_1$$

And the last row will be set as follows:

$$enc(bit_n) \cdot \ldots \cdot enc(bit_2) \cdot enc(bit_1) \cdot enc(bit_0) \cdot Output_n$$

After calculating the value of all the possible rows, we sum the results, where we have zeros for all the rows, except for at most one row, which represents the received encrypted input, and that row calculation will result in our expected output corresponding to the input. It is important to note that we do not break any of the FHE properties, and are not able to read the encrypted input as cleartext, but constructing the blind (and computation heavy) argument, allows us to return a defined output for every possible input, using solely the encrypted input from the delegator.
A basic example defined and implemented using HELib of an input range of three bits, where the output for each possible in-

put (0-7) is set alternately to 0 and 1 can be found in [46].

**Algorithm 2:** Implementation of bitwise OR operator

**Require:** $x, y$
return $((x + y) - (x \cdot y))$

**Algorithm 3:** Implementation of the Negation operator

**Require:** $enc(value), enc(1)$
return $(enc(1) - enc(value))$

**Algorithm 4:** Use $LUT\ L$ for an encrypted input $x$

**Require:** $enc(x_0), enc(x_1), \ldots, enc(x_n), LUT\ L,\ enc(1)$
**for** $i = 0\ to\ n$ **do**
$negated_i = Negation(enc(x_i), enc(1))$
**end for**
$out[0] = negated_n \cdot \ldots \cdot negated_2 \cdot negated_1 \cdot negated_0 \cdot output_0$
$out[1] = negated_n \cdot \ldots \cdot negated_2 \cdot negated_1 \cdot enc(x_0) \cdot output_1$
$\ldots$
$out[n] = enc(x_n) \cdot \ldots \cdot enc(x_2) \cdot enc(x_1) \cdot enc(x_0) \cdot output_n$
**for** $i = 0\ to\ n$ **do**
$result = result + out[i]$
**end for**
return $result$

The conclusion from all the above is that a malicious server can arbitrarily manipulate the outputs in a consistent way across FHE values encrypted with different keys. Thus, we better restrict the computation primitives the server can use.

Next, we examine several possible restrictions, in particular, restricting the computer to execute only the addition operation. We still investigate the bit computation granularity and examine the binary addition in the process of inputs additions.

## 3 Fingerprints in Addition in Word Computation Granularity

A computation following our scheme will be traceable for verification purposes by adding encrypted input control bits, called *computation fingerprints*, inside a single fully homomorphic encrypted number [19], where the use of subtraction is handled as the addition of negative values. Also, from now on, the scope of each encrypted value is different, where the $encrypted$ notation will refer to a single number completely encrypted, instead of having different and separated encrypted bits. After understanding the proposed solution we survey and test its limits, analyzing the probability for successful detection of a malicious server's wrong calculation, where the soundness of our scheme and the *computation fingerprint* verification will be presented and surveyed later on.

**Arithmetic circuit representation.** We first design our (word) schemes using arithmetic circuits, that represent polynomials (possibly over a certain finite field).

Arithmetic circuits are defined as the multiplication or addition of two other variables or constants and provide a formal way to represent the polynomial computing complexity, of which we concentrate in our scheme. In the case of the addition of two polynomials, the resulted polynomial will have the same number of monomials, with component-wise addition, where the multiplication of polynomials will result in a long number of single monomials, added to each other. This representation and specifically the multiplication composition will increase (even exponentially) the size of the polynomial description, yet will be set within our scheme limitation as it will result in a (large) number of separated monomials, calculated by addition of (also, logarithmic represented values for facilitating multiplications) values, and added to each other.

**Restricting to blackbox additions.** One of the basic operations in computation is an addition, recalling that arithmetic circuits are a combination of additions and multiplications. We examine the server capabilities to manipulate the addition operation when the computation is executed over bit representation. The ability to replace the carry bits computed in each bitwise operation by 0, will lead to the result of 00 when adding 01 to 01. In other words, the server can just ignore or set to zero any carry bit that arose from the addition operation, resulting in a wrong result.

Thus, we further restrict the primitives of the server to execute blackbox word granularity additions, where the server calls a function that returns the correct addition result rather than involving a binary addition. In addition to that, in the next section we present a scheme not relying on such requirements.

This means that from now on, we focus on building a computation blackbox supporting environment, similar to the interpreter level (e.g., Java byte-code [47, 48]) or restricted operating system as Internet cafe (e.g., [49]). The schemes we present can implement any arithmetic circuit computation, while providing trust for the correctness of the computed calculation.

**Adversarial capabilities of blackbox addition restricted server.** We restrict the operations of the (possibly malicious) server to execute blackbox word granularity additions, while the server is still able to deviate from performing exactly one addition of each of the inputs to be added. This means that the use of blackboxs (analogously to the use of hardware machine opcode primitives) are always assumed, but still an undesired sequence of their operation can lead to wrong output. Note that the blackbox restriction prevents the server from manipulating the encrypted inputs otherwise.

**Binary fingerprints.** We suggest computation (addition) fingerprints to ensure the server added the required variables, exactly once. We construct and work with numbers that are composed of parts that fulfill two purposes; the first is to compute the calculation request itself, and the other part(s) for verifying the computation that was made, which we call *computation fingerprint*. All numbers will be segmented respectively, to support the correct addition of parts. The expected result of the fingerprint is known and can be computed by the delegator, and can be reused/reconstructed over different inputs, even working with different servers, as it is encrypted and its

exact position is unknown to any server. The final result from the server is verified, and deemed valid or not, based on the computed value of the *computation fingerprint*.

For example, if we would like to calculate $001 + 110$, we add four computation fingerprint bits in the tail of $001$ and $110$ to have, say, $0010010$ and $1100100$, and check whether the result ends with $0110$ prior to accepting the $111$ of the three most significant bits as the correct result. Note, that the choice of having the binary fingerprints to be located as the least significant bits is motivated by the automatic overflow processing of the actual calculation part, avoiding a mix between the fingerprints and the actual calculation portion. This is our motivation to choose the portion of the *computation fingerprint* to reside in the least significant part, as the delegator selects the values comprising the input and accordingly knows the expected result of the fingerprint. Note, that the malicious server may use fingerprint overflow to ruin the result while keeping the fingerprint correctness, in particular, the server can add an addend $2^m + 1$ times, where $m$ is the number of fingerprint bits. Thus, our addition blackbox is designed to set the result of addition to zero if there is an overflow from the fingerprint part.

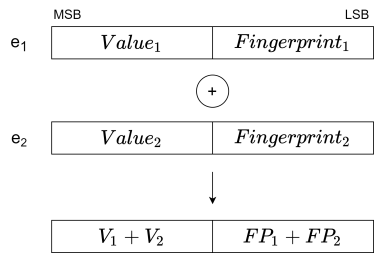



**Fig. 2.** Encrypted to encrypted addition

**Fingerprint manipulation.** Obviously, if all inputs are added up at most once and at least one input is added less than once there will be a wrong (missing) fingerprint detected in the result. Alternatively, the server may choose to add an input several times to create the *computation fingerprint* in a different manner. For now, we keep each input with a single fingerprint bit in a location different than any other addend for analysis and to prevent overflows from the fingerprint portion through to the calculation portion when the computation is correct. Next, we demonstrate that the adversary can try to guess the locations of the fingerprints and create fake fingerprints by adding an encrypted input more than once, without necessarily triggering an overflow from the fingerprint portion.

Assume that the server knows, or correctly guesses (as the input is encrypted), that there is an encrypted input $x = Enc(0000001)$, then the (malicious) server can produce any fingerprint by adding $x$ to itself, for example, $y = x+x = Enc(0000010)$, and $z = y + y = Enc(0000100)$. Moreover, the server can add $y + z$ to obtain the combination $Enc(0000110)$ of fingerprints.

The (malicious) server may add all addends but one, $x$, and try to create the missing fingerprint of the excluded addend by using the fingerprint of another addend $y$. The server may succeed in creating the missing fingerprint if the fingerprint of $y$ is smaller than the fingerprint of $x$, for this, there is $1/2$ probability of guessing correctly, and also guessing the right number of bits $d$ that separate the fingerprint of $x$ from the fingerprint of $y$, for this, the probability of a correct guess is $1/(m-1)$, where $m$ is the number of bits used for the fingerprints. If the guesses are right, which occurs with probability $1/(2m-2)$, then adding $y$ $2^d$ times to the sum that excluded $x$ will mask the absence of $x$.

We also note that, the computation fingerprints are probabilistic in nature, and the probability is related to the number of bits used for the computation versus the number of bits used for the fingerprints. For the sake of convenience we chose these number of bits to be equal, allowing the usage of a past verified computation to serve as fingerprints.

**Complete fingerprints.** While defining binary fingerprints and surveying the possibilities to manipulate them, we suggested keeping each input a single fingerprint bit, in a unique location. We strengthen this suggestion and require that the fingerprint bit length correlate to the number of addends used, maintaining the previous suggestion that a single, uniquely located fingerprint bit be used per input, so that the complete addition of inputs will result in a $1..11$ fingerprint. The influence of even a single redundant addition will be a definite overflow of the fingerprint.

Despite the fact that *complete fingerprints* allow the evident indication of excessive additions, as adding all inputs and any of the inputs more than once will create a fingerprint value different than $1..11$; still, a weakness exists as we describe next.

**Fingerprint overflowing.** In addition to the adversary possibilities surveyed earlier, we consider a different technique that can allow an adversary to create a calculation result with the expected fingerprint, while still producing a wrong calculation output.

Assuming the fingerprint value resides in the least significant part of the input, the adversary can add an input to itself $2^m$ times to clear out the fingerprint ($m$ is the number of fingerprint bits), while changing the computation value, where an extra addition of the same input will result in the inputs original fingerprint, with a different (corrupted) calculation section. The use of this corrupted input in the requested calculation by the server would create an output with the expected fingerprint value, and an arbitrary wrong calculation result.

Using *complete fingerprints*, an overflow will be immediately identified by the most significant bit in the fingerprint section. This bit can be used with the previously constructed bitwise NOT operator (algorithm 3), calculated on the overflow indicating bit, when at least one of the bits is (encrypted) 1 (when all are zeros, the fingerprint stays zero, which enables detection), multiplied by the given result. If no overflow occurred, this will result in the multiplication of the result by 1, which will not change it. This defines the final blackbox configuration, allowing only addition no overflow-carry operations. In the sequel (when discussing SIMD) we relax the use of

our predefined blackboxs.

**Fingerprint carry detection.** In the *complete fingerprint* scenario, as all inputs have a unique fingerprint bit location, no carry operation in the fingerprint section while adding is required, implying that any carry operation indicates an unintended (malicious) behavior. This behavior might represent an attempt to compensate for a missing input and will influence our blackbox definition, so it nullifies the result when a carry operation, in the relevant section, is detected.

We implement it using the OR and NOT bitwise operators (algorithms 2, 3), where the blackbox will OR all the carries that resulted in the fingerprint section, negate this result, and multiply it by the addition result to zero the final output when an undesired carry takes place. In the completing case, where no carries were made, the OR operation will result in a 0, then negated to 1, which will not change or impact the output. For this, the blackbox should be aware of the length $m$ of the fingerprint section, and should be specifically built for that fingerprint length. It is important to note that every input, whether it is the same value or not, is regarded independently and is associated with a unique fingerprint.

**Capabilities bound of a restricted adversarial.** The restrictions mentioned result in a tight bound on the adversarial capabilities following our scheme, using the addition no overflow-carry blackbox. We start with the deficient case, where a single missing addend will be obviously detected in the resultant fingerprint. The opposite behavior is the excessive addition of inputs, where a single extra addition will cause a definite overflow in the *complete fingerprints* scheme, which will nullify the whole result. The completing case for all of the above, addressed using the carry detection, will strictly nullify the addition result when any different than intended inputs are added. A scenario where the received input equals 0, is managed by using a *blind if*, and returning the same (0) output. An important note is that the encryption keys are changed in every different request, or use, of the server.

This sets a tight bound on the adversarial capabilities, so we can state the following lemma.

**Lemma 1.** *For every group of FHE input of bits, having $m$ least significant bits used as complete fingerprints, and the other $n$ bits used for computation, operated on by the addition no overflow-carry blackbox defined earlier, will result in the correct addition of the computation values and a complete fingerprint, or the value 0.*

*Proof.* The proof follows from the definition of the addition no overflow-carry blackbox. Let the FHE values $Input_1, Input_2, \ldots Input_i$, following our scheme, having complete fingerprint values, operate in the mentioned blackbox. Execution of the blackbox for any of the same inputs will result in a carry, which will be immediately nullified. Omitting a certain input from the calculation will result in a not *complete fingerprint*, and an attempt to compensate for it will involve the carry operation, which will nullify the result in such an event. Note that even if the fingerprint overflowing vulnerability surveyed earlier was not mitigated by the carry detection mechanism, it will be explicitly nullified at any fingerprint overflow.

**Integer fingerprints.** After considering binary fingerprints, we turn to consider integer fingerprints, where a random number $k$, bigger than one (as 0/1 values are the identity values for addition/multiplication) is chosen uniformly from the range $1$ to $2^m - 1$, such that if the addition operation consists of $i$ addends, then the number of overflow bits can be $\log i$ and thus, $k + \log i$ must be less than or equal to $2^m$ ($m$ is the number of fingerprint bits). This new scheme changes the fingerprint value after each addition by more than a single bit, in contrast to the binary fingerprints proposal. This prevents us from nullifying fingerprint carry operations (as it is now needed), yet adds more possibilities for a potential attacker, and lowers its probability to guess the right fingerprint to $1/2^k$. This still requires us to take into account potential overflows, as the adversary can cause an overflow by adding an input more than once. To overcome this, we require the same overflowing mitigation presented earlier. Obviously, the bootstrap of integer fingerprint requires computation, namely, addition of the first fingerprint (for the needed number of addends), later on, the result of the calculations (when no overflow is possible, as enough leading bit values are $0$) may serve as future fingerprint values, based on the correction of the first computed fingerprints yielding the correction of the following computations in a transitive manner.

**Different inputs subset probability.** Integer fingerprints expand substantially the range of possible fingerprint values after each addition, yet they might be maliciously used in a new manner. We can represent the verification process we proposed as a variant of the subset sum problem. The set $Z$ of $i$ positive integers are the fingerprints in each of the input values, and the target value $t$ is the expected fingerprint result $2^k$. We have a slight variation on the original problem, as the values (inputs) can be repeated. The arithmetic circuit (polynomial) which was requested to be calculated, represents the intended subset that achieves the target value, yet there might be other subsets that result in the same target. The subset sum problem was recently [50] proved to be solved in pseudo-polynomial time of $\widetilde{O}(t + i)$. Despite that, our scheme has a major difference, as the values used are encrypted, making our problem much more complex, also in the average case, namely, the *blind subset sum* problem, where the server does not know the actual values of its inputs, nor the target value. This leads us to the conclusion that for a calculation with $i$ addends, and a target value $2^k$, any value of the $k$ least significant bits of $m$ have a uniform probability to be chosen and the sum $\pmod{2^k}$ has also uniform probability in the range of $0$ to $2^k - 1$, thus the probability of using a (repeated) subset to gain the needed result is less than $1/2^k$.

**Multiplication by a constant.** Taking into account multiplication by a non encrypted constant, the multiplication can be considered as a shortcut of listing certain inputs several times as addends. More general multiplication can mix the fingerprint part with the actual computation variable, and therefore we suggest using discrete logarithm representation, as we describe next. The multiplication complexity occurs due to the influence of the bits in both parts in an unexpected manner in the same or completing parts, and their mutual dependency on each other in the

encrypted number. This mixture of bits prevents us from accounting for it in our operations, preventing any option to compensate for it in the fingerprint.

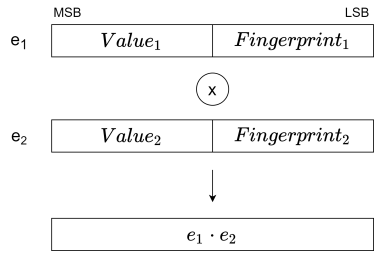



**Fig. 3.** Encrypted numbers multiplication

The loss of the multiplication operation on encrypted numbers limits and produces a weaker scheme and prevents it from fully implementing arithmetic circuits, which are composites of both additions and multiplications. We evaluate a solution to the deficiency of multiplication in the next sections.

## 4 Multiplication via Logarithms in Word Representation

We suggest using our addition techniques to verify multiplications, namely, use discrete logarithmic representation with fingerprints as inputs.

As our goal is to delegate the complete computation of an arithmetic circuit, we need to verify both additions and multiplications. One possibility is to regard an arithmetic circuit as a polynomial, multiplying to compute monomials, and adding the results of the monomials to complete the polynomial computation.

As a first possibility, the computing server may use LUT to convert the inputs to discrete logarithmic values and to exponentiate the result back before adding.

Then, we propose a possibility to delegate these tasks to a server/hardware/device (e.g., enclaved), or to further enforce that the server compute first multiplications, then use a function that exponentiates the result, while it preserves the encryption and the fingerprints of the inputs, adding an encrypted counter as we detail in the next section.

**Adversarial capabilities.** We have divided the multiplication process into two phases, the logarithmic addition of values, and the exponentiation of the result. The former was evaluated in the fingerprint manipulation section, in terms of adversarial capabilities, while the latter LUT phase involves the use of an encrypted result for each possible logarithmic addition outcome. The options open to an adversary with an encrypted LUT are limited, where an adversary can only change the order, or amount, of LUT operations on the input(s). This means that any atomic use of the LUT can not be interfered with, yet can be repeatedly abused. Any input other than the intended one, used by the LUT, will result in a wrong fingerprint output being sent to the delegator, or will be wrongly propagated during following operations.

An absence of exponentiation will result in a wrong (missing) fingerprint by the server. Any manipulation on the (encrypted) fingerprints values will have the same adversarial probabilities considered in the fingerprint manipulation section.

**Discrete logarithmic representation.** To imitate multiplication between encrypted numbers, we use logarithmic addition in base 2, which is based on the equation $\log(x \cdot y) = \log(x) + \log(y)$. The solution will be set by the delegator, which is completely transparent to the server. The delegator, instead of requesting multiplication of values, will send the (encrypted) logarithmic representation of the values to the server, which are obviously protected with fingerprints. This makes the server compute additions of log values, which are sent to the delegator as a result to be verified against the fingerprints and then to be exponentiated.

A basic case of computing $4 \cdot 8$, will be represented as $2 + 3$ in the $\log_2$ field by the delegator. Having the server compute this addition in the proposed scheme will result in the value 5, where the delegator will calculate the result of $2^5$ by itself, and achieve the value 32, which is the original result of the requested multiplication $4 \cdot 8$.

By using logarithmic addition we can mimic multiplication in our scheme, yet this will introduce other, smaller, limitations. First, this requires the delegator to calculate the logarithmic representation of the values, possibly by using LUT or caching the already computed logarithms for later reuse.

For example, in the function we show later as $F(x, y) = (2 \cdot x) + y + 3$ (all constants are encrypted), the calculation of the addition of constants will influence differently than intended, as those values should be reduced to represent their correct significance in the new field. This restricts the complexity of addition and multiplications calculations of individual factors, and form a calculation characterized by a long addition of many monomials. Moreover, not all log values will result in a round value, restricting to the use of a floating-point FHE scheme, while introducing rounding errors, where it is possible to operate on integers only in additions.

**Logarithmic lookup table.** A possible way to overcome the dependency on the delegator is to exponentiate logarithmic represented values by a blackbox procedure for exponentiation, while keeping the original fingerprint and adding fingerprints for counting the number of exponentiated results that were added.

This means that along with the ordinary fingerprints for additions ($FP_a$) which were explained earlier, there will be other (multiplication) fingerprint bits ($FP_m$) assigned to track and verify the number of LUT uses made by the server, in the same manner binary fingerprints were used to verify the execution of the addition operator on the input(s). The new fingerprint serves as an indication to avoid the possibility that the server adds the logarithmic representation with their fingerprints, rather than enforcing the exponentiation with fingerprints to ensure the avoidance of this scenario as well.

In continuation from the previous example of $4 \cdot 8$, the server will calculate $2 + 3$, resulting in 5, which is represented as 000101. The fingerprint is of equal size of six bits, let's say, 011000. For readability, we separate the computation and fingerprint parts with a space. The delegator will set a row in the LUT, for the input of

| Input as $Value\ \ FP_a\ \ FP_m$ | Output as $Value\ \ FP_a\ \ FP_m$ |
|---|---|
| 000 01 00 | 000 01 01 |
| 001 10 00 | 010 10 01 |
| 010 11 00 | 100 11 01 |
| .. | .. |

**Fig. 4.** Exponentiation lookup table example

000101 011000, the result of 100000 011001. This result is composed of the calculation result 100000 (32) and the fingerprint 011001. The fingerprint was changed from the original 011000, where the LSB was turned on. This bit is used to indicate in the final result received from the server that this specific exponentiation operation was indeed applied in the calculation process, as the fingerprint was changed accordingly.

This solution can make the server independent from the delegator and will allow it to continue calculating without interruptions, yet with a computational cost of using the LUT when switching from multiplication to additions.

For example, we calculate the polynomial (that represents an arithmetic circuit), say, $F(x, y) = 2 \cdot (x \cdot y + 32)$ for $x = 4, y = 8$. Following the previous example, we expand the fingerprint and calculation parts to eight bits. The requested $4 \cdot 8$, was changed by $log_2$ to $2 + 3$, resulting in 00000101 00011000.

Using the exponentiation LUT, the value was changed to 00100000 00011001, containing the requested value 32 (00100000), with the fingerprint value 00011001. The next operation is an addition of 32, having the fingerprint value 00100000. This sum is represented as 01000000 00111001.

Next, we request to multiply by 2, which is represented as the addition of the value 1. Adding the value 1 will obligate transforming 64 to 6 by the same $\log_2$, while retaining the fingerprint value. This will be achieved by using another LUT, changing the value 01000000 00111001 to 00000110 00111010, where the counting fingerprint value was increased by 1. Finally, we will add the value 1 with it's fingerprint 01000000 to result 00000111 01111010. By using the exponentiation LUT again, we will change the computation part, which equals 7, to 128, and increase the fingerprint by 1. This yields the final output of 10000000 01111011. The fingerprint value is known and consistent for the selected fingerprint values, and will be compared and verified when acquired by the delegator. The whole calculation process is shown in Figure 5.

**Lemma 2.** *For every group of FHE inputs, each having $m$ least significant bits with addition and multiplication (counting) fingerprints ($FP_a,\ FP_m$), and the other $n$ bits used for computation, operated with the addition no overflow-carry blackbox and the exponentiation LUT defined earlier, will result in the correct calculation of the computation and fingerprint parts, or the value 0.*

*Proof.* By Lemma 1 the logarithmic addition is correct or set to zero prior to the exponentiations using the LUT. In case the server adds the results of the logarithmic addition and outputs the result without using the LUT, detection of the missing counter in

| Operation | Values |
|---|---|
| $2+3$ | 00000010 00010000<br>00000011 00001000 |
| $LUT(5)=32$ | 00000101 00011000<br>00100000 00011001 |
| $32+32$ | 00100000 00011001<br>00100000 00100000 |
| $LUT^{-1}(64)=6$ | 01000000 00111001<br>00000110 00111010 |
| $6+1$ | 00000110 00111010<br>00000001 01000000 |
| $LUT(7)=128$ | 00000111 01111010<br>10000000 01111011 |

**Fig. 5.** Calculation using LUT example

$FP_m$ occurs. As the multiplication fingerprints $FP_m$ reside in the least significant bit part of the fingerprint section itself, even the bounded carry attack attempt using the addition fingerprints will not assist the adversary, as the carry bits could only overflow to the computation part. Any overflow from the computation part effecting the fingerprint section, will also nullify the output. The LUT operations, defined by the delegator, will include all possible monomial and fingerprint values, and will be set to change the $FP_m$ values in a distinct manner.

## 5 Integration and Implementation in Word Granularity

As we deal with multiplications (in logarithmic representation) exponentiation while preserving computation fingerprints, and operating additions, we may want to use blackbox operations that enforce the ordered operations we want to apply. One may use different isolated server/device/hardware restricted to execute only one portion in the above sequence of operation and send the results to the next entity to operate the next operation as required. One opportunity is to use the separated enclave as such computation hardware.

In the previous sections we discussed limitation, that could be enforced by executing different blackboxs as enclaves. The "Intel SGX" [51] is capable of creating the mentioned enclave, running encrypted code made for a specific program, where the execution of this code is inaccessible for other external processes or entities to interfere. Thus, it will enable us to enforce the use of additions exclusively, eliminating the possibility of any unexpected behavior. As mentioned earlier, in the next section discussing SIMD we present a scheme not using such requirements.

**Using existing software libraries.** For our example we used Microsoft SEAL library (C#), where we calculated the following function $F(x,y) = (2 \cdot x) + y + 3$, where all the constants are in their encrypted form. Allocating 6 bits as the size of each part, we have selected (3,2) as the predefined fingerprint values, where $F(3,2) = 11$. The fingerprint part was selected as the right (LSB) side, as this choice does not cause

overflows. The computation values, generally selected by the computation delegator for it's own use, in this example equal to $(4, 7)$, where $F(4, 7) = 18$. The addition of constant values was implemented as basic additions along with the requirement of receiving the added value in a doubled form, meaning that for the addition of the constant 3, the value is received with its shifted counterpart, so the addition will be of a number constructed as 3, added to the bitwise logically left shifted (indicated as *LSH*) by 6 bits counterpart of it, $LSH_6(3) = 192$, where the same was also set for subtraction. The constant multiplication was made as regular multiplication of the constant encrypted value. The input values were $LSH_6$ each, where each result was appended to the requested fingerprint values, in this example, $(3, 2)$.

The constructed input was set as $x = LSH_6(4) + 3 = 259$, $y = LSH_6(7) + 2 = 450$. After calculating $F$ in the form described above, the cleartext result was decrypted as $1163$, of which the least significant part, containing the fingerprint value is extracted using bitwise left and right logical shifting (*LSH*, *RSH*), by calculating $RSH_6(LSH_6(1163)) = 11$. Extracting the computation value from the most significant part was by $RSH_6(1163) = 18$. Verifying that the former equals our expected fingerprint value of $11$ confirm the calculating, stating that the calculation was verified to have been calculated correctly and that $F(4, 7) = 18$. Out of 3 required additions/subtractions in the original function $F$, the same 3 operations were made, keeping also the original single multiplication as is. An important note is that the extra addition operations made to the constant values should be computed just once per function definition, and could be reused later and on different input values or servers. This means that the calculation of $(LSH_6(3) + 3)$ is independent of the (other) input values, and could be calculated offline and used for all further calculations. The source code of this example can be found in [46].

**Supported libraries.** The main libraries tested were Microsoft SEAL [40] and IBM HELib [41], both use the BGV/BFV [23] and CKKS [34] schemes. Both of those libraries support C++ and work on Linux (Ubuntu), where SEAL also works on Windows OS with C# support. Note, that the CKKS scheme supports floating-point operations. The libraries were chosen for their popularity and accessibility, where each of them supports FHE with addition and multiplication. Along with that, HELib focuses on making SIMD operations much faster than previous libraries [41], allowing us to work with vectors of encrypted data, and apply a selected operation on all of the vector elements.

**Encrypted number representation.** As mentioned earlier we consider working on encrypted vector (bit) representation of the data, yet not all schemes have (encrypted) bit access, and on some of the FHE schemes we have an encrypted integer number representation. Working with regular FH encrypted numbers involves drastically more computations compared to the vector representation of the data, yet we are still able to operate on it, including the basic blind conditioning defined earlier.

On vector-based variables allowing specific bit access with $w$ bits, the computation complexity of blind conditioning can be represented as a calculation for each of the two possible values of a bit. This will be made for all the bits separately, thus, resulting

in a complexity of $O(w)$. On the other hand, with regular FHE numbers (without bit access) the blind conditioning will depend on all of the bit values of the input, combined. This means that we will have a condition for every possible value of the number as a whole, representing a $O(2^w)$ complexity, where $w$ is the total number of bits in the input. Due to that, we refer to both of the representations as equivalent in a qualitative (rather than quantitative) manner, as they have the same output capabilities.

**Fields.** While limiting the number of operations possible to eliminate the chance of overflowing each of the part limits, we can elaborate on a different way to *reinitialize* our verification layer, using modulo and finite fields. This can be implemented in a simple form by providing the modulo value as an encrypted number from the delegator, used by the server, to subtract any whole multiplication of the field. An overflow can be detected using a *blind if* (algorithm 1) on a bit indicating it, and the subtracted result will be used for further calculation. The provided encrypted value from the delegator could also possibly include a module value to be subtracted from the fingerprint part, yet the two parts are not dependent, and the fingerprint section can be set to not be affected by the subtraction.

**Lemma 3.** *Any calculation of an arithmetic circuit, representing a polynomial, having FHE inputs that are infused with computation fingerprints following our scheme, can be successfully verified by the delegator.*

*Proof.* By Lemma 1 and Lemma 2, the addition and multiplication of encrypted values, infused with fingerprints, will result in the correct calculation of the computation and fingerprint parts, or the value 0. Any polynomial is the addition of monomials, where each monomial is the multiplication of values. Thus, we can represent and calculate any polynomial by our scheme, therefore, we can calculate any arithmetic circuit. Receiving the expected fingerprint value will verify the correct execution of the arithmetic circuit, and any different fingerprint result will deem otherwise.

## 6 Restrictive SIMD as a vector with Fingerprints

We move on to the possibility of incorporating the *computational fingerprint* into SIMD manipulated data structures, such as vectors. Starting with vectors of integer elements, we proceed to vectors with only fingerprint integer values, enabling floating point arithmetic for the delegator.

**Background on floating point operation on data in SIMD.** The primary FHE floating point scheme is the CKKS [34], which works by encoding a vector of plaintext values to a polynomial. Each coefficient of the polynomial is originated from the corresponding vector element, which is projected to a different field, and then multiplied by a scaling factor to control the rounding error of the floating point number. This polynomial is defined by treating the vector values as the image for a selected set of roots, where the degree of the resulted polynomial will equal the vector length. In addition to that, there is a special technique called rescaling,

that consistently maintains the decryption structure small enough compared to the ciphertext modulus, as a kind of modulus switch operation [23].

The rescaling operation reduces the size of the ciphertext modulus, while maintaining a valid encryption and almost preserving the precision of the plaintext, using primes selected as part of the encryption parameters, so at each rescaling execution the ciphertext modulo is divided by one of the chosen primes. It is important to note that as in modulus switching, the number of primes limits the amount of possible rescaling operations, and thus limits the possible depth (level) of the computation. This limit will enable us to restrict overflow attempts caused by redundant SIMD operations.

FHE operations are obtained by manipulating each monomial of the polynomial accordingly. This succinct polynomial representation is well suited for SIMD calculations, where each operation will be acted upon in the same manner, on all polynomial coefficients. Still, we can regard each entry of the SIMD vector as a totally independent value.

**Adversarial capabilities in SIMD environment.** Although instructed otherwise, the adversary can compute a different arithmetic circuit (or general computation, which is not necessarily defined in a finite field) than the one the delegator instructs the server to use. Still, the server is bounded to perform the exact same sequence of operations on each entry of the data vector, and cannot execute different programs for each such entry. Moreover, there is isolation between vector entries, with no mutual influence.

We choose to restrict our vector components, both fingerprint and computation elements, to integers, as using floating point may result in rounding errors, possibly nullifying the effect of (very) small fingerprint numbers not in the scale of (much) bigger numbers when, say, they are added to each other, and this will result in approximating the small number to 0 (see e.g., [52]).

**Integers in SIMD.** Using the floating point CKKS scheme described earlier, we are able to manipulate vectors containing different elements in an efficient SIMD manner. We will adapt the *integer fingerprints* scheme to comply with this new state, where separate elements reside in a vector. As in the basic *integer fingerprints* scheme, the fingerprint element, manipulated by the requested arithmetic circuit from the delegator, will result in a value the delegator can better verify.

By limiting our scheme to work with only integer elements, we also restrict divisions, as this impairs any possibility of creating floating point values while executing a calculation. Regarding possible overflows, discussed in the previous sections, any overflow of the vector will affect all of its elements in the same manner, without mutual influence between different elements. This means that overflows might just create new (overflowed) values or return the vector to its original state, meaning that in this scheme, we do not need to require a blackbox that identify and act upon overflows.

Note that the usage of integers in SIMD allows us to skip the log representation conversion (using the LUT) described earlier for the word granularity case, making our scheme even more efficient. In addition, for the same arithmetic circuit, we can verify multiply isolated inputs in a vector, using a single fingerprint value in it.

**Fingerprints in floating point vectors.** SIMD supports floating point variables and operations, still, we prefer the fingerprints to be restricted to integer and to avoid divisions and subtractions altogether. Additionally, we avoid adding any constant floating point values to the input, as those also affect the fingerprint element. The reasons for restricting the fingerprints to be integer are related to the special rounding operations applied in the scope of float representations. Nevertheless, we allow floating point values to be used as the computation elements.

Avoiding divisions and subtractions includes limiting all fingerprint values to be a positive integer (as discussed above, greater than 1), to prohibit subtraction by an addition of negative values, or division as the multiplication of fractures.

As for avoiding divisions and subtractions, we can take, for example, a floating point computation element with a value of $0.0001$, added to a $10000000000000.0$ and then immediately subtracted by the same $10000000000000.0$. The addition and subtraction of the floating point element may be approximated to $0$, yet the corresponding integer fingerprint elements that maybe, say 7 and 4, will return to the original value of 7, which makes the floating point rounding invisible to the verifying delegator, thus, we might verify a wrong computation result as the right result.

Even if we use only integers as fingerprints, changing the ordering of operation, might be used to maliciously create the correct fingerprint result. The possibility of this manipulation will be thoroughly discussed in the following section.

An implementation example of calculating the polynomial $F(x, y) = (((2 \cdot x) + 1.5) \cdot (y \cdot 3)) + 0.1$, with SIMD manipulation of a vector containing integer fingerprint element and floating point computational values, using Microsoft SEAL, can be found in [46].

**Computation fingerprint circuit.** We would like to mention that one may also want to trace the order of operations made (e.g., first adding small numbers to be non negligible before adding them to a very large number, a serial addition of each one of them to the big number results in an approximation to 0). We suggest extending the tracing by having the blackbox produce (possibly during the computation or as a final result) a symbolic (with or without FHE known computed values) representation of the computation steps, in a form of an arithmetic circuit.

**Theorem 1.** *Under the surveyed restrictions, computation fingerprint circuit verifies the correctness of the computation made in SIMD.*

## 7 Conclusion Remarks

We presented a new approach to use computation fingerprints within FHE values to check a delegated computation result. Our word granularity and SIMD solutions are designed to check the final result of the computation, without any significant additional overhead.

Thus, having FHE inputs and a desired arithmetic circuit, any entity following our scheme can delegate the required computation to any other party, have a with high

probability guaranteed that the computation was made as requested, and accomplish this procedure by computing the calculation exactly once, with no redundancy.

The SIMD [53] optimization that many FHE libraries provide allows performing an individual computation, like addition or multiplication, on a vector of encrypted elements. This unique optimization will give us a substantial advantage over the word granularity FHE schemes, as the scope of the fingerprint method may be extended to the case of non-integer values, including floating point calculations (the fingerprint can still be restricted to integers, but the actual variables can be also beyond integers). Note that the restriction of using only SIMD server may yield a less efficient solution than the computer word solution, as it requires to compute the function more than once, either in parallel or sequential manner. Still, when a computation should be executed in parallel in the first place, or when the function is defined over non integer values (e.g., floating point) and is beyond the computation capabilities of an arithmetic circuit over a finite field, the SIMD setting can serve us better.

Verifiable computing may be used on almost any of the many FHE applications, as Homomorphic encryption in its essence is oriented toward delegating data to be operated in untrusted environments [54], thus, there is great benefit in verifying and providing assurance for those type of computations.

## References


1. Saikrishna Badrinarayanan, Yael Tauman Kalai, Dakshita Khurana, Amit Sahai, and Daniel Wichs. Non-interactive delegation for low-space non-deterministic computation. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 25, page 9, 2018.
2. Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 723–732, 1992.
3. Silvio Micali. Cs proofs. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 436–453. IEEE, 1994.
4. Shlomi Dolev, Sergey Lazarevich Frenkel, and Dan E Tamir. Probabilistic methods for self-correcting hardware design. *Informatics and Application, Russian Academy of Science,* 7(4):140–147, 2013.
5. Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. Delegating computation: interactive proofs for muggles. *Journal of the ACM (JACM)*, 62(4):1–64, 2015.
6. Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *Journal of the ACM (JACM)*, 38(3):690–728, 1991.
7. Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the snark. *Journal of Cryptology*, 30(4):989–1066, 2017.
8. Dario Fiore, Rosario Gennaro, Valerio Pastro, Efficiently Verifiable Computation on Encrypted Data. CCS 2014: 844-855
9. Rosario Gennaro, Craig Gentry, Bryan Parno: Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. CRYPTO 2010: 465-482
10. Rosario Gennaro, Daniel Wichs: Fully Homomorphic Message Authenticators. ASIACRYPT (2) 2013: 301-320
11. Kai-Min Chung, Yael Tauman Kalai, Salil P. Vadhan: Improved Delegation of Computation Using Fully Homomorphic Encryption. CRYPTO 2010: 483-501
12. Benny Applebaum, Yuval Ishai, Eyal Kushilevitz: From Secrecy to Soundness: Efficient Verification via Secure Computation. ICALP (1) 2010: 152-163

13. Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum: Fiat-Shamir From Simpler Assumptions, Cryptology ePrint Archive: Report 2018/1004.
14. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 21–30, 2007.
15. Sanjeev Arora and Shmuel Safra, "Probabilistic checking of proofs: A new characterization of NP", *Journal of the ACM*, 45 (1): 70–122, 1998.
16. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In *International Colloquium on Automata, Languages, and Programming*, pages 152–163. Springer, 2010.
17. Kai-Min Chung, Yael Kalai, and Salil Vadhan. Improved delegation of computation using fully homomorphic encryption. In *Annual Cryptology Conference*, pages 483–501. Springer, 2010.
18. Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Annual Cryptology Conference*, pages 465–482. Springer, 2010.
19. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.
20. Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.
21. Joppe W Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In *IMA International Conference on Cryptography and Coding*, pages 45–64. Springer, 2013.
22. Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual Cryptology Conference*, pages 868–886. Springer, 2012.
23. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.
24. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on Computing*, 43(2):831–871, 2014.
25. Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Annual cryptology conference*, pages 505–524. Springer, 2011.
26. Jung Hee Cheon and Damien Stehlé. Fully homomophic encryption over the integers revisited. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 513–536. Springer, 2015.
27. Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In *International Workshop on Public Key Cryptography*, pages 311–328. Springer, 2014.
28. Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 24–43. Springer, 2010.
29. Yarkın Doröz, Yin Hu, and Berk Sunar. Homomorphic aes evaluation using the modified ltv scheme. *Designs, Codes and Cryptography*, 80(2):333–358, 2016.
30. Léo Ducas and Daniele Micciancio. Fhew: bootstrapping homomorphic encryption in less than a second. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 617–640. Springer, 2015.
31. Craig Gentry, Shai Halevi, and Nigel P Smart. Homomorphic evaluation of the aes circuit. In *Annual Cryptology Conference*, pages 850–867. Springer, 2012.

32. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Annual Cryptology Conference*, pages 75–92. Springer, 2013.
33. Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1219–1234, 2012.
34. Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437. Springer, 2017.
35. Jung Hee Cheon, Miran Kim, and Kristin Lauter. Homomorphic computation of edit distance. In *International Conference on Financial Cryptography and Data Security*, pages 194–212. Springer, 2015.
36. Miran Kim, Yongsoo Song, and Jung Hee Cheon. Secure searching of biomarkers through hybrid homomorphic encryption scheme. *BMC medical genomics*, 10(2):42, 2017.
37. Kristin Lauter, Adriana López-Alt, and Michael Naehrig. Private computation on encrypted genomic data. In *International Conference on Cryptology and Information Security in Latin America*, pages 3–27. Springer, 2014.
38. Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 113–124, 2011.
39. Shuang Wang, Yuchen Zhang, Wenrui Dai, Kristin Lauter, Miran Kim, Yuzhe Tang, Hongkai Xiong, and Xiaoqian Jiang. Healer: homomorphic computation of exact logistic regression for secure rare disease variants analysis in gwas. *Bioinformatics*, 32(2):211–218, 2016.
40. Microsoft SEAL (release 3.5). https://github.com/Microsoft/SEAL, April 2020. Microsoft Research, Redmond, WA.
41. Shai Halevi and Victor Shoup. Algorithms in helib. In *Annual Cryptology Conference*, pages 554–571. Springer, 2014.
42. Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. In *Concurrency: the Works of Leslie Lamport*, pages 203–226. 2019.
43. David A Patterson and John L Hennessy. Computer organization and design: the hardware/software interface 5th ed., 2014.
44. Amitay Shaer, Shlomi Dolev, Silvia Bonomi, Michel Raynal, and Roberto Baldoni. B ee's s trategy a gainst b yzantines replacing byzantine participants. In *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*, pages 139–153. Springer, 2018.
45. Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of computer and system sciences*, 47(3):549–595, 1993.
46. Github code repository, https://github.com/seniakalma/VC-FHE, February 2021.
47. Bill Venners. The java virtual machine. *Java and the Java virtual machine: definition, verification, validation*, 1998.
48. James Gosling, Bill Joy, Guy Steele, and Gilad Bracha. *The Java language specification*. Addison-Wesley Professional, 2000.
49. Scott Alan Isaacson. Techniques for internet cafe service provider access, November 5 2009. US Patent App. 12/112,098.
50. Karl Bringmann. A near-linear pseudopolynomial time algorithm for subset sum. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1073–1084. SIAM, 2017.
51. Victor Costan and Srinivas Devadas. Intel sgx explained. *IACR Cryptol. ePrint Arch.*, 2016(86):1–118, 2016.
52. David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys (CSUR)*, 23(1):5–48, 1991.

53. Nigel P Smart and Frederik Vercauteren. Fully homomorphic simd operations. *Designs, codes and cryptography*, 71(1):57–81, 2014.
54. Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A Reuter, and Martin Strand. A guide to fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2015:1192, 2015.

# Incorporating Improved Neural Network Motion Vector Predictions in the HEVC HM Reference CODEC

**Raz Birman, Shevach Riabtsev, Hananel Assor, Dor Avni, and Ofer Hadar**
*School of Electrical and Computer Engineering. Department of Communication Systems Engineering. Ben-Gurion University, Israel*

**Abstract:** Motion Estimation is a fundamental component of Inter-Prediction, used in codecs to take advantage of inter-frame pixel value redundancies for compression purposes. Motion Vectors (MV) are used by P and B frames to point to similar blocks between sequenced frames. Further compression is achieved by performing Motion Vector Predictions (MVP) and transmitting only the Motion Vector Difference (MVD) between the MVs of neighboring blocks to that of the predicted one. While the H.264/AVC standard utilizes the median between the MVs of 3 neighboring blocks, HEVC, also known as H.265 had made a leap in accuracy by introducing the Merge and AMVP algorithms for Motion Estimation. The Merge candidates list provides the ultimate accuracy and cannot be further improved. However, there is potential for improvement of the MVP for blocks that use AMVP. We introduced Neural Network (NN) prediction of the MVP from the 2 AMVP candidates and incorporated the new predicted MVP value instead of the AMVP candidates. Multiple tests of different NN architectures and activation functions have led us to believe that the performance of HEVC can only be marginally improved. In this paper we present the results but more importantly, we describe the method which has been used to modify the HM standard reference codec, which allows other researchers to implement the same mechanism and further test MVP improvement algorithms.

## 1. Introduction

In recent years streaming video is becoming a predominant Internet content. Utilization of channels' bandwidth is extremely important for economically transmitting video of sufficient quality to satisfy ever increasing consumers' expectations. In order to accommodate these requirements, video compression algorithms have been greatly improved over the past two decades. The latest video compression standard that was released is H.266/VVC (Versatile Video Coding). The standard that was released before it is H.265, also known by High Efficiency Video Coding (HEVC) [1], [2]. Additional standard that is used by Google is AV1 from the Alliance of Open Media (AOMedia) [3]. The standards use multiple algorithms to reduce bandwidth for a given quality, however, the most dominant contribution stems from using spatial and temporal pixel values' redundancies. Inter-prediction leverages temporal similarities between pixel blocks of consecutive video frames in order to send residuals errors and Motion Vectors (MV). The MV is a pointer from the block of the target frame to the matching block of the source frame. P-Frames and B-Frames use forward and backward pointers respectively. In order to further reduce the required bitrate, the HEVC standard performs Motion Vector Prediction (MVP) of the block's MV and transmits only the residual error between the MVP and the Ground Truth MV (GTMV), which is the actual MV of the predicted block. The HEVC standard [5] utilizes two different methods for MVP – Merge and Advanced Motion Vector Prediction (AMVP). The Merge mode provides a list of up to 5 MV candidates from neighboring spatial and temporal blocks. In the 'Merge' case, one of the candidates is a perfect match of the block MV and therefore the index of this candidate is signaled between the encoder and the decoder. The AMVP method provides two neighboring MVs and indicates which of them is the best match. A residual - Motion Vector Difference (MVD), is necessary only in the AMVP case. Improving the MVP int he AMVP case will reduce coding rates. In addition, the MVP is also used to perform the search for the matching block and therefore, its accuracy will necessarily improve search time and computation efficiency of the encoder.

In this paper we are introducing an innovative method for improving MVP in the AMVP case by training a Fully Connected Neural Network (FCNN) to predict a more accurate MVP to the GTMV of the predicted block. We further incorporate the FCNN in the HM reference encoder/decoder model software and replace the MVP with our improved predicted values. We explain how to perform this coding change to the HM, which is a challenging task that, to the best of our knowledge, has not been published before, and will assist other researchers in their efforts to incorporate MVP improvements in the HM code. After performing multiple training runs of the FCNN with different network architectures and activation functions and for different movie types, we conclude, even though not scientifically prove, that our results are close to the best that can be achieved for AMVP improvement.

## 2. Related Work

Various efforts have been invested in improving MV prediction accuracy and search algorithms. In [6] the authors have tackled the same challenge and have indicated the drawback of having to add signaling. They offered one selection method which is based on the content statistics, thus allowing the decoder to perform the selection without adding signaling. We use neural networks in order to accomplish that. In [7] the authors present a new technique for motion vectors prediction based on spatial and temporal prediction. The motion vector of a moving object is tracked using spatial and temporal prediction and used as a starting point for the ME searching algorithm at the encoder end. The predicted motion vector is selected from several candidate motion vectors according to the block matching criterion. Experiments show that this spatial-temporal prediction reduces the number of computations performed by the motion search algorithm by 30% for MPEG2 encoding and by 40% for H.263 encoding. The Median method has typically yielded sufficiently accurate results of the GTMV for coding purposes; therefore, the majority of the research efforts have been invested in improving the efficiency of the motion estimation itself. In [8] a MV prediction method is presented. It is a Prediction Search Algorithm (PSA) for block motion estimation. The proposed method utilizes a linear combination of the motion vectors of the three adjacent blocks to obtain a predicted motion vector, namely, the initial search point. Simulation results show that the proposed PSA is better than the three-step search algorithm [9] and the four-step search algorithm [10] in terms of MSE with smaller computational workload. To improve the accuracy of the fast Block Matching Algorithms (BMAs), in [12], a new adaptive motion tracking search algorithm is proposed. Based on the spatial correlation of motion blocks, a predicted starting search point, which reflects the motion trend of the current block, is adaptively chosen. Experimental results show that the proposed algorithm enhances the accuracy of the fast center-biased Block-Matching-Algorithms (BMAs), such as the new three-step search [9], the four-step search [10], and the block-based gradient descent search [11], as well as reduces their computational cost. As in [9], L. Luo. et al., [13] propose a new prediction search algorithm for block motion estimation utilizing the linear weighing of the MVs of the 3 adjacent blocks. In [14] E. Kaminsky and O. Hadar propose a method for effectively analyzing and selecting the most suitable motion estimation algorithm. All these methods use conventional prediction approaches such as least square estimation of weights in a linear combination of neighboring vectors, Median prediction and so on. Due to substantial improvements in compression efficiency, the contribution of MVP error coding is becoming more attractive as means to obtain further bit rate reduction while retaining the same quality (since it is not quantized).

In recent years there have been efforts to harness the power of Neural Networks for improving predictions for video coding. More efforts have been invested in the improvement of Intra-Prediction [15][16][17][18]. However, there has also been some research for improving motion compensation. The primary focus of these research papers was on motion compensation at the single pixel level, which corresponds to optical flow. Thus, the authors of [19] use Convolutional Neural Networks (CNN) for predicting a heat map optical flow from consecutive video frames.

## 3. Using FCNN for predicting MVP

We have extracted from the HM code datasets of AMVP candidates along with the GTMV and used supervised learning to train the network to predict a best matching MVP. We trained 2 different networks for the $x$ and for the $y$ MVP component values. The network architecture that was used is depicted in Fig. 1 below. Each data sample of the input is of the structure depicted in Equation (1). The output of the network is either the $x$ or the $y$ of the MVP, as indicated in Equation (2). The data inputs were normalized for best utilization of the network dynamic range and for better convergence purposes.

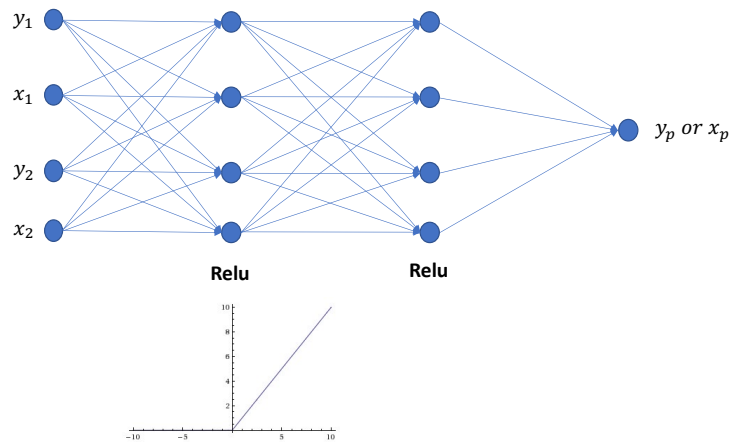



*Fig. 1: Fully Connected Neural Network used for predicting $x, y$ MVP values*

$$input_i = \{x_{MV1_i}, y_{MV1_i}, x_{MV2_i}, y_{MV2_i}\} \tag{1}$$

$$output_i = x_{GTMV_i}\ OR\ y_{GTMV_i} \tag{2}$$

The network was incorporated in the HM reference codec as a function with the trained weights and biases.

The networks were implemented in Python using Keras. The networks were trained with a full dataset per epoch. Keras criteria for halting training were defined with a patience parameter of 30 epochs of no improvement with a min-delta for loss function of 0.0001. The networks were optimized using the Adam Optimizer [20], which is an improvement of Stochastic Gradient Decent (SGD) algorithm [21], that is using Momentum, which is effectively a factored running average of the gradients in the different steps so far, and RMSprop, which introduces a factored square of the gradient in order to reduce variations in steeper directions and prefer more gradual and stable ones. We used the Adam optimizer with a constant learning rate of 0.001 and decay coefficients of 0.9 and 0.999 for β_1 and β_2 respectively, which are the default for Keras Adam optimizer library and were proven to provide sufficiently fast convergence rate.

## 4. Incorporating the modified MVP in the HM code

In order to replace the MVP in the HM reference encoder/decoder software, we have added 4 new integer variables (m_firstPredictorX, m_firstPredictorY, m_secondPredictorX and m_secondPredictorY), to the 'TComDataCU' struct. This struct represents the data of every CU. We used these variables to maintain the $x$ and $y$ components of the two AMVP candidates. We also implemented getters and setters for these variables. The FCNN was implemented as a function that gets the candidates variables as input, calculates the prediction and returns the result as output. The FCNN function was implemented in the file 'TEncEntropy.cpp'. The incorporation process is performed in the function 'encodePUWise' function, that is used for encoding the motion information of every PU block. In case the PU block is inter-block that uses the AMVP mode and not Merge, the candidates are inserted into the FCNN function. Using the GTMV and the FCNN output, we calculate the new MVD. If the new MVD is better than the old one, the old one is replaced by the new. If the old one is better nothing changes.

## 5. Results

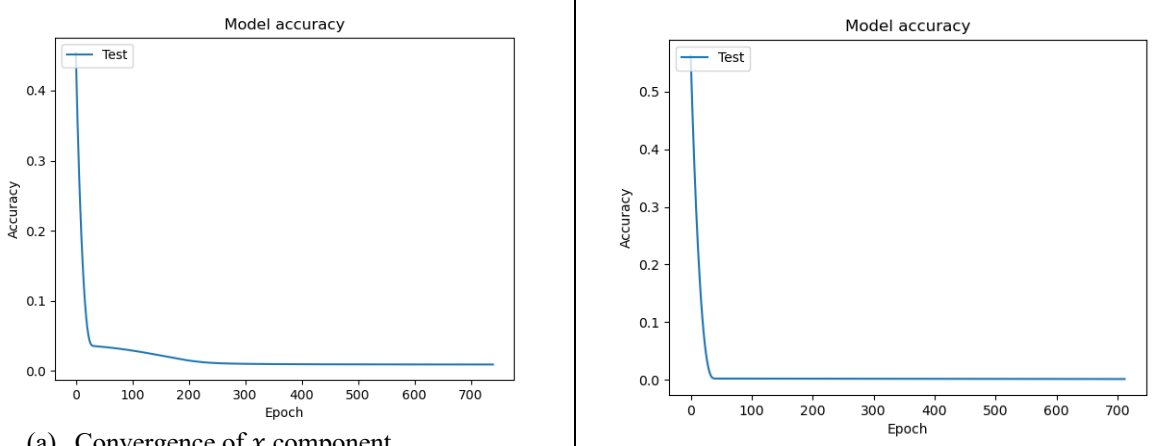
The HEVC/H.265 standard has accomplished substantial improvements of ME over H.264/AVC. It applies two methods for predicting MVs – Merge and AMVP. The Merge mode is based on a list of temporal and spatial neighbors, one of which is identical to the MV of the block. Therefore, our method is not applicable to Merge. The AMVP mode selects from two best matching neighboring blocks, therefore, we can improve the prediction using our regression neural network. In order to assess the improvement potential, we have extracted from the HM reference software codec the number of Inter-Prediction blocks that were predicted with AMVP vs. the ones that were predicted with Merge. We trained FCNN depicted in Fig. 1 with datasets that were extracted from four different movies. An illustration

of a single frame from each one of the movies that were used is provided in Fig. 2. The FCNN had indicated good convergence, yet the optimal results that were obtained were not too promising. The convergence graphs for the $x$ and $y$ components of the MVP are illustrated in Fig. 3. The results are provided in Table 1 for four different movies.

(a) Park Scene (b) Traffic

(c) Fast Forward Traffic (d) Star Wars

*Fig. 2: Four movies used for training and testing*




(a) Convergence of $x$ component (b) Convergence of $y$ component

*Fig. 3: FCNN convergence for the x and y components of the MVP*

As can be seen, the percentage of AMVP blocks from the total varies with the movie but ranges in the values of 13% - 37% for the movies that we selected. Our approach is limited to improving the MVPs of these blocks only. Furthermore, the best optimized FCNN had managed to improve only a subset of the MVPs in the range of 14% - 31%, which further reduces the potential improvement impact.

A regression neural network was applied, and the results were introduced into the HM reference software codec. The network prediction was chosen to replace the AMVP candidate when it provides a closer match to the GTMV. The

RD results do not show any improvement; however, marginal improvement was presented in the BD-Rate and BD-PSNR results. The results are provided in Fig. 4.

*Table 1: AMVP vs. Merge mode blocks extracted from HM for different movies ( $Q_p = 25$ and without I-Frames)*

| Movie | AMVP | Merge | % AMVP | % Improved |
|---|---|---|---|---|
| Park Scene | 11,096 | 155,830 | 20% | 19% |
| Traffic | 8,824 | 15,020 | 37% | 14% |
| Fast Forward | 11,096 | 63,304 | 15% | 22% |
| Star Wars | 16,248 | 112,696 | 13% | 31% |

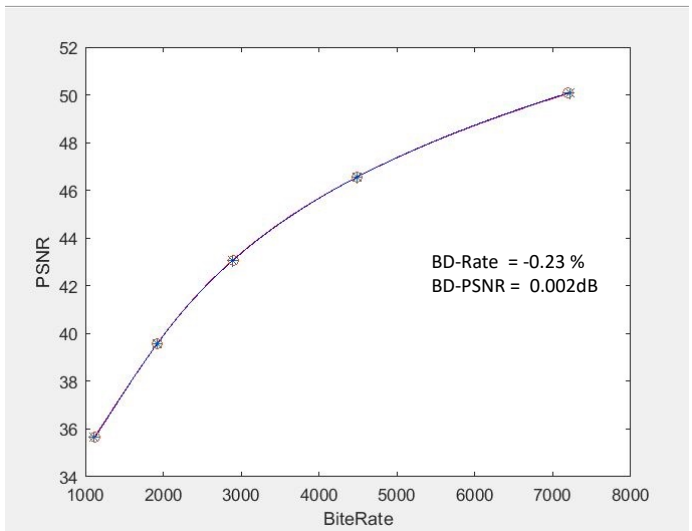



*Fig. 4: Rate Distortion along with improvements of BD-Rate and BD-PSNR for the Traffic movie*

The BD-Rate and BD-PSNR results for the 4 tested movies are depicted in *Table 2*.

*Table 2: BD-PSNR and BD-Rate results*

| Movie | BD-PSNR | BD-Rate |
|---|---|---|
| Park Scene | 0.0044db | -0.1196% |
| Traffic | 0.0022db | -0.0225% |
| Fast Forward | 0.0060db | -0.1129% |
| Star Wars | 0.0012db | -0.0286% |

## 6. Conclusions

In this research we have incorporate a Fully Connected Neural Network (FCNN) in the HM reference codec framework for predicting improved AMVP candidates of the HEVC standard. The HEVC/H.265 standard has made a big leap compared to H.264/AVC in providing a more accurate MVP. The improvement potential is restricted to the AMVP subset since the Merge candidates are already accurate and cannot be improved further. We have performed multiple simulations with a wide variety of NN architectures that span to multiple number of hidden layers, multiple neurons per layer and different activation functions. Since the optimization results always converge to the same values, we tend to conclude that the presently used, human crafted MVP algorithms, implemented by HEVC, are close to optimal. Yet, some marginal improvement has been accomplished and is described in the paper. In addition, we have dealt with the technical challenge of incorporating a new MVP in the HM reference codec. We expect that the technical method that is described in this paper may assist other researchers in modifying the HM code and allow them to further investigate MVP improvement directions.

The HEVC/H.265 standard utilizes fast Rate Distortion Optimization (RDO) that select several Intra and Inter prediction block candidates, determines the rate by mode bits and motion vectors and the distortion by the sum of

absolute difference (SAD) or the sum of absolute transformed difference (SATD). The method is used to rapidly prune the less probable cases. Our proposed FCNN predicted MVP can be considered for facilitating a faster Early Termination algorithm, based on the predicted MV.

## References


[1] Iain E. Richardson, The H.264 Advanced Video Compression Standard, 2nd Edition, April (2010)

[2] "ITU-T Recommendation H.265/ ISO/IEC 23008-2:2013 MPEG-H Part 2: High Efficiency Video Coding (HEVC)," (2013)

[3] D. Mukherjee, H. Su, J. Bankoski, A. Converse, J. Han,Z.Liu and Y. Xu, "An overview of new video coding tools under consideration for VP10: the successor to VP9" , SPIE., (2015).

[4] V. Sze, M. Budagavi, and G.J Sullivan, "High efficiency video coding (HEVC)". In Integrated circuit and systems, algorithms and architectures (Vol. 39, pp. 49-90). Springer., (2014)

[5] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard", IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1649–1668., (2012).

[6] G. Laroche, J. Jung, and B. Pesquet-Popescu, "RD optimized coding for motion vector predictor selection", in IEEE Transactions on Circuits and Systems for Video Technology, 18(9), pp.1247-1257., (2008).

[7] I. Ismaeil, A. Docef, F. Kossentini, and R. Ward, "Efficient Motion Estimation Using Spatial and Temporal Motion Vector Prediction", in Proceedings 1999 International Conference on Image Processing (ICIP) 24-28 Oct., (1999).

[8] L. Lijun, Z. Cairong, G. Xiqi, and H. Zhenya, "A new prediction search algorithm for block motion estimation in video coding", in IEEE Transactions on Consumer Electronics, Vol. 43 , No. 1 , pp. 56-61, Feb (1997).

[9] Li, Reoxiang, B. Zeng, and M.L. Liou, "A new three-step search algorithm for block motion estimation", in IEEE transactions on circuits and systems for video technology, 4(4), pp.438-442., (1994).

[10] L.M. Po, and W.C. Ma, "A novel four-step search algorithm for fast block motion estimation", in IEEE transactions on circuits and systems for video technology, 6(3), pp.313-317., (1996).

[11] L.K. Liu, and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding", in IEEE Transactions on circuits and systems for Video Technology, 6(4), pp.419-422., (1996).

[12] J. B. Xu, L. M. Po, and C. K. Cheung, "Adaptive motion tracking block matching algorithms for video coding," in IEEE Transactions on Circuits and Systems for Video Technology, Vol. 9, No. 7, pp. 1025-1029, Oct., (1999).

[13] L. Luo, C. Zou, X. Gao, and Z. He, "A new prediction search algorithm for block motion estimation in video coding", in IEEE transactions on consumer electronics, 43(1), pp.56-61., (1977).

[14] E. Kaminsky, and O. Hadar, "Multiparameter method for analysis and selection of motion estimation algorithms for video compression", in Multimedia Tools and Applications, 38(1), pp.119-146. (2008)

[15] T. Laude, and J. Ostermann. "Deep learning-based intra prediction mode decision for HEVC", in Picture Coding Symposium (PCS), IEEE., (2016).

[16] W. Cui, T. Zhang, S. Zhang, F. Jiang, W. Zuo, and D. Zhao, "Convolutional neural networks based intra prediction for HEVC". in arXiv preprint arXiv:1808.05734., (2018).

[17] J. Li, B. Li, J. Xu, and R. Xiong, "Intra prediction using fully connected network for video coding", in IEEE International Conference on Image Processing (ICIP) (pp. 1-5)., (2017).

[18] R. Birman, Y. Segal, A. D. Malka, and O. Hadar, "Intra prediction with deep learning", in SPIE Optics + Photonics conference, San Diego, California (USA)., (2018).

[19] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks", In Proceedings of the IEEE international conference on computer vision (pp. 2758-2766)., (2015).

[20] D. Kingma, and J. Ba, "A Method for Stochastic Optimization". arXiv:1412.6980 [cs.LG], December (2014).

[21] L. Bottou. "Large-Scale Machine Learning with Stochastic Gradient Descent". In Proc. of COMPSTAT (2010).

# Climb of Spam in digital security: A State-of-the-Art Review of machine learning-based Decision trees, Extra Trees in tree-based model and Gaussian, Bernoulli, and Multinomial from Naïve Bayes model

Priyanka Dhara[1], Shubham Bhattacharjee[1], Sohom Bhattacharya[1], Subir Gupta[1*]

[1]*Deparment of Master of Computer Application, B C Roy Engineering College, Durgapur, West Bengal 713206, India.*

**Corresponding authors*
*Email: subir2276@gmail.com Ph.:+91-9477462400,*
*ORCID 0000000209410749*

## Abstract

Security in the computerized world is the protection of computer frameworks and connections from information openness, theft of, or damage to their gear, programming, or electronic data. The field is getting effectively basic considering the comprehensive usage of computer frameworks, the Internet, and distant alliance standards like Bluetooth and Wi-Fi, and the improvement of devices that advance the Internet of things (IOT). Spam is maybe perhaps the best portrayal of this sort of assault. Of late, uncommon orchestrated trained professionals, researchers, and experts have uncovered the risk of spam validation, these assaults cause an uproar and underhandedness not exclusively to individual and ventures both. Spam can be perceived using various estimations at this point one of the consistent investigation spaces of spam revelation is the supervised machine learning (ML) technique. In the ML research locale, tree models and the Bayes model are famous. In this article, the fundamental goal is Heuristics assessment of five unique models of tree-based and Bayse-based in particular, Decision Tree, Extra Tree from tree model near, and Gaussian, Bernoulli, and Multinomial from Naïve Bayes approach with a contextual analysis of more than 40k in addition to spam remark datasets. This dataset is isolated into two classifications that are 75% are utilized for preparing and the rest 25 % utilized for testing. End of Heuristics assessment utilizing accuracy, precision, recall, f1 score, auc score, model training time and mean squared error estimation boundaries it can notice Decision Tree is appearing far hitter results than different models in numerous exhibition matrics.

***Keyword:-*** Bernouli Naïve Bayes; Decision Tree; Extra Tree; Gaussian Naïve Bayes; Machine Learning; Multinomial Naïve Bayes.

## 1. Introduction

Of late, the growing use of automated mode has actuated the issues of Spam. Specific customers and affiliations are impacted by Spam considering the numerous regards like social worth, internet business, and so forth Spam changes into a colossal test in electronic security network

flourishing[1][2]. To tending to the spam issues in the security area, the different technique is gotten by experts one of the steady standard viewpoints is Machine learning (ML)[3]. Arthur Samuel rushed to present ML in 1959. After a short period of time, ML is by and large applied in a collection of fields. Today, ML is used in some construction or another all things considered, every single other application and programming on the Internet, since ML has gotten so unavoidable, it has gotten the go-to respond in due order regarding adventures dealing with a combination of issues[4]. It might be used in a grouping of fields, including picture affirmation, material science, and showing data science, among others[5][6][7]. A couple of managed ML computations are open. In this paper, review the state of art of machine learning-based Decision trees, Extra Trees in tree-based model and Gaussian, Bernoulli, and Multinomial from Naïve Bayes model are examined with different measurement matrices like Accuracy Score, Precision Score, Recall Score, F1 Score, AUC Score, Model Training Time, Mean Square Error[8][9][10][11][12][13].

## 2. Methodology

For the Review of condition of-the-rt, one fundamental dataset and a practically identical climate are required. Figure 1 shows that the course for discovering the likelihood of spam. Therefore, 40k despite datasets are engineered utilizing open information source likewise as some really followed by sifting and evening through. Datasets are spilled into two classes, where 75% of the information is overseen for model preparing and sometime later another 25% of the dataset is managed for testing purposes. The course of action dataset goes through all classifiers and readies the ideal model. The entirety of the classifiers has an earth-shattering numerical structure. In district 2.1 all numerical clarifications promptly take a gander at. End of preparing rests 25% dataset which continues through two models self-rulingly with the last fair of uncertainty which will talk what mass is the believability of spam and ham. All models nearby outcomes, arranging time, F1 score, and so on are utilized to outline the condition of craftsmanship.

### *2.1 Mathematical Explanation*

(a) Decision tree –

This tree is modelled on the working mechanism of comparison sort widely used in data sorting applications. A linear decision tree generalizes a comparison tree for computation of functions which use real vectors y ∈ $R^n$ as input data. The linear function used in such cases is described in (1)

Output = $b_0 + \sum_{j=1}^{n} b_j y_j$ for real numbers from $(b_0 \ldots b_n)$ ----------(1)

Linear decision trees and comparison trees are similar because comparison among $y_a$ and $y_b$ is similar to the linear function of $y_a - y_b$. These trees can only take into account the functions that have union and intersection of half-space.

An algebraic decision tree further generalizes a comparison tree by generalizing a linear decision tree. It allows the input functions to be polynomials of degree k. The computation of f : $R^n \rightarrow \{0, 1\}$ where f(y) = 0 is possible if and only if m , n are distinct co-ordinates such that $y_m = y_n$. Also, it needs an algebraic decision tree with depth $\Omega(nlog(n))$. Unlike linear decision trees, these trees have a lower bound of $n^2$.

The task of Boolean decision tree is the computation of value of a w-bit Boolean function given by (2)

f : $\{0 , 1\}^n \rightarrow \{0 , 1\}$ for input values a $\in \{0 , 1\}^n$ ----------(2)

The task of computing the final result requires iteratively reading a bit of the input $a_j$ and the output f(a). Each decision may or may not be dependent on the previous decisions[14][15].

(b) Extra tree –

Boolean decision tree can be broadly classified into four main types – deterministic, randomized, non-deterministic and quantum. Extra tree classifier utilizes the underlying architecture of randomized decision tree. This tree is similar to Boolean decision tree but involves the introduction of additional nodes to an existing decision tree which can be controlled with the probability of $P_k$. $R_2(g)$, also known as the Monte-Carlo randomized decision tree complexity can be defined as the lowest depth randomized decision that has result f(z) which has a probability of at least $2/3$ for all z$\in$ $\{0 , 1\}^n$ i.e. it has bounded 2-sided error. $R_0(g)$, also known as the Las Vegas decision tree complexity is used to measure the expected depth of decision tree that might be correct i.e. it has zero-error. $R_1(g)$, denotes an error version which is one-sided[16].

(c) Bernoulli naïve bayes –

It is based on the assumption that the feature vectors are independent binary values which describe the inputs. Let, $y_j$ = a Boolean value which expresses the occurrence or absence of $j^{th}$ term from the pre-defined vocabulary being used while calculating. Hence, the probability of y occurring in a given class $D_k$ is expressed in (3)

P(y $|D_k$) = $\Pi_{j=1}^{n} P_{kj}^{y_j}(1 - P_{kj})^{(1-y_i)}$ ----------(3)

Where $P_{kj}$ = probability of generation of the term $y_j$ by class$D_k$.

The crux of this model's working procedure is to use binary term frequencies instead of term frequencies[17].

(d) Gaussian naïve bayes –

It is based on the typical assumption that the continuous data values of each class is a normal /or Gaussian distribution[18].

Let

a = the continuous attribute which is segmented into different classes

$\mu_k$ = the mean of a values linked with class $D_k$

$\sigma_k^2$ = Bessel corrected variance linked with the same class

u = some observation value

Thus, the probability density of u in the given class $D_k$ is computed using the function shown in (4)

$$P(a = u \mid D_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{\frac{-(u-\mu_k)^2}{2\sigma_k^2}} \qquad \text{----------(4)}$$

(e) Multinomial naïve bayes –

It is based on the assumption that the feature vectors of input data are the frequencies of certain events that get generated by a multinomial denoted by $(q_1 ... q_n)$ where $q_j$ = probability of event j occurring[19].

Let y = $(y_1 ... y_n)$ be a feature vector which is actually a histogram. The probability of observing y is expressed in (5)

$$P(y \mid D_k) = \frac{(\sum_{j=1}^{n} y_j)!}{(\Pi_{j=1}^{n} y_j)!} \Pi_{j=1}^{n} {q_{kj}}^{y_j} \qquad \text{----------(5)}$$

A multinomial naïve bayes model gets transformed into a linear classifier when the following conversion happens until (6)

$$\log P(D_k \mid y) \; \alpha \; \log ( P(D_k) \; \Pi_{j=1}^{n} {q_{kj}}^{y_j} )$$

$$\Rightarrow \log P(D_k \mid y) = \log P(D_k) + \sum_{j=1}^{n} y_j . \log q_{kj}$$

$$\Rightarrow \log P(D_k \mid y) = b + w_k^T y \qquad \text{----------(6)}$$

Where, b = $\log P(D_k)$ and $w_{kj} = \log q_{kj}$

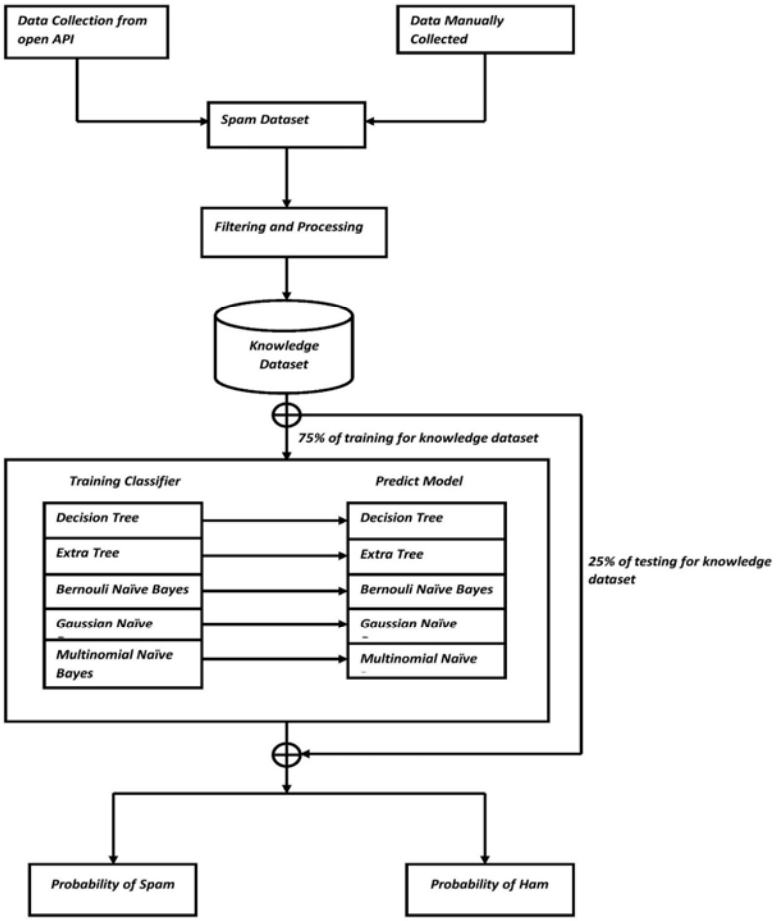



Figure 1:- Diagram for spam comment detection using Machine Learning Algorithm

## 3. Result Analysis

In this article, seven assessment impediments are applied specifically Accuracy Score, Precision Score, Recall Score, F1 Score, AUC Score, Model Training Time, Mean Square Error to evaluate the presence of five amazing models. AUC curve, precision-recall curve, and, heatmap show up in figure 2. Where the AUC curve is the degree or level of prominence. It conveys how well the model is fit for isolating between classes. The higher the AUC, the better the model is in gathering 0s and 1s. AUC goes from 0 to 1 to the extent of importance. An AUC of 0.0 exhibits that a model's notions are completely mixed up, while an AUC of 1.0 shows that its evaluations are correct. Consequently, the precision-recall curve, which is used for data recovery settings, is showing up in a lot of recovered reports and huge chronicles. On the other hand, heatmap proposes that a particularly strong visual guide for a watcher, allowing the speedy transmission of evident or data-driven information.

Table 2 appearance the Accuracy Score, Precision Score, Recall Score, F1 Score, AUC Score, Model Training Time, Mean Square Error thinks about the execution of every one of the five procedures using different measurement matrices. According to the consequences of table 1, it noticed the F1 and accuracy and the precision score of decision matrices are 92.8617, 94.4390, and 92.9909 separately which is higher than the other four classifiers, that decision matrices F1 score, accuracy score, and precision score are altogether better than the other. The recall and AUC scores of Bernoulli matrices are 98.4860 and 98.1751 individually which is higher than the others.

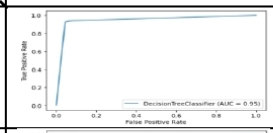
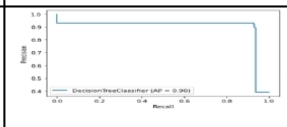
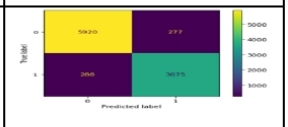
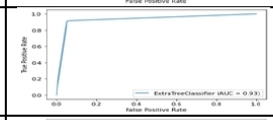
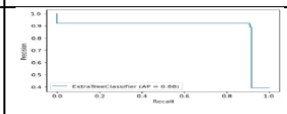
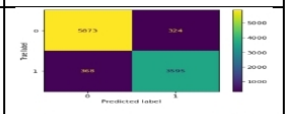
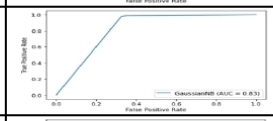
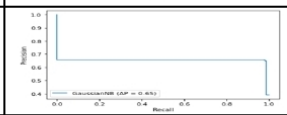
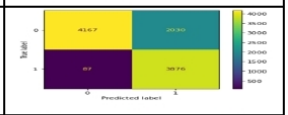
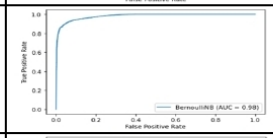
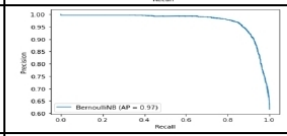
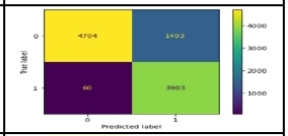
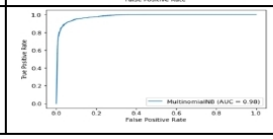
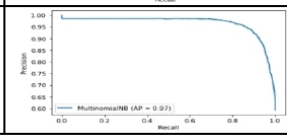
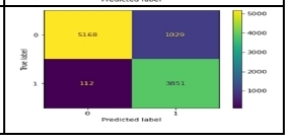
| Measurement / Algorithms | AUC Curve Graph | Precision-Recall Graph | Heatmap/Accuracy Recall Graph |
|---|---|---|---|
| (a) Decision Tree |  |  |  |
| (b) Extra Tree |  |  |  |
| (c) Gaussian Naïve Bayes |  |  |  |
| (d) Bernoulli Naïve Bayes |  |  |  |
| (e) Multinomial Naïve Bayes |  |  |  |

Figure 2:- Measurement of AUC Curve, Precision Curve, Heatmap of five different algorithm (a) Decision Tree (b) Extra Tree (c) Gaussian Naïve Bayes (d) Bernouli Naïve Bayes (e) Multinomial Naïve Bayes

So here Bernoulli matrices beat in recall and AUC score more than the others. In contrast with the other four procedures, Gaussian Matrices are adequate for Model preparing time since they save minor exertion for tanning, for example, 2.5135 contrasted with the other four methodologies. In mean square error, the most key recorded worth is decision matrices which have the most un-worth 0.0556 which is lower than contrasted with different matrices.

**Table 1:- Outcome Limit of the five Models**

| Parameter List | Decision Metrices | Extra Tree Metrices | Bernoulli Metrices | Gaussian Metrices | Multinomial Metrices |
|---|---|---|---|---|---|
| Accuracy score | 94.4390 | 93.1890 | 84.7146 | 79.1634 | 88.7697 |
| Precision score | 92.9909 | 91.7326 | 72.3314 | 65.6282 | 78.9139 |
| Recall score | 92.7328 | 93.5907 | 98.4860 | 97.8047 | 97.1739 |
| F1 score | 92.8617 | 91.2205 | 83.4063 | 78.5490 | 87.0971 |
| AUC score | 94.5827 | 93.1708 | 98.1751 | 82.9680 | 98.0452 |
| Model training time | 241.4770 | 6.9574 | 5.4996 | 2.5135 | 4.5559 |
| Mean Square Error | 0.0556 | 0.0681 | 0.1529 | 0.2084 | 0.1123 |

## 4. Conclusion

State of art for performace assessment are assembled in two huge parts that is an indispensable condition and sufficient condition. Even though a fundamental condition is a condition that ought to be accessible for an event to occur then again sufficient condition is a condition or set of conditions that will make the event. An imperative condition ought to be there, yet it alone doesn't give sufficient motivation to the occasion of the event. In this heuristic, locale increases F1 (which is figured using the precision and recall) score and cut-off MSE is the central condition for any programmer similarly as in industry and subject matter expert. Although AUC score lift and model planning time limit is a sufficient condition. Using this notion it can undeniably be ensured that considering all condition decision trees is wonderful every one of the five methodologies in the case of binary labeling dataset.

## Conflict of interest

The authors declare that there is no conflict of interest in the present work.

## References:

[1] S. Hedley, “A brief history of spam,” Inf. Commun. Technol. Law, vol. 15, no. 3, pp. 223–238, 2006, doi: 10.1080/13600830600960758.

[2] C. A. Shue, M. Gupta, C. H. Kong, J. T. Lubia, and A. S. Yuksel, “Spamology: A study of spam origins,” 6th Conf. Email Anti-Spam, CEAS 2009, no. January, 2009.

[3] M. Crawford, T. M. Khoshgoftaar, J. D. Prusa, A. N. Richter, and H. Al Najada, “Survey of review spam detection using machine learning techniques,” J. Big Data, vol. 2, no. 1, 2015, doi: 10.1186/s40537-015-0029-9.

[4] A. Fahfouh, J. Riffi, M. Adnane Mahraz, A. Yahyaouy, and H. Tairi, “PV-DAE: A hybrid model for deceptive opinion spam based on neural network architectures,” Expert Syst. Appl., vol. 157, p. 113517, 2020, doi: 10.1016/j.eswa.2020.113517.

[5] S. Gupta, J. Sarkar, M. Kundu, N. R. Bandyopadhyay, and S. Ganguly, “Automatic recognition of SEM microstructure and phases of steel using LBP and random decision forest operator,” Meas. J. Int. Meas. Confed., vol. 151, p. 107224, 2020, doi: 10.1016/j.measurement.2019.107224.

[6] S. Gupta et al., “Modelling the steel microstructure knowledge for in-silico recognition of phases using machine learning,” Mater. Chem. Phys., vol. 252, no. March, p. 123286, 2020, doi: 10.1016/j.matchemphys.2020.123286.

[7] S. Panda, A. K. Ghosh, A. Das, U. Dey, and S. Gupta, “Machine Learning-based Linear regression way to deal with making data science model for checking the sufficiency of night curfew in Maharashtra , India,” vol. 1, no. 2, pp. 168–173, 2021.

[8] D. Tests, “Basic Principles of ROC Analysis,” Semin. Nucl. Med., vol. VIII, no. 4, pp. 283–298, 1978.

[9] B. Ozenne, F. Subtil, and D. Maucort-Boulch, “The precision-recall curve overcame the optimism of the receiver operating characteristic curve in rare diseases,” J. Clin. Epidemiol., vol. 68, no. 8, pp. 855–859, 2015, doi: 10.1016/j.jclinepi.2015.02.010.

[10] A. Berger and S. Guda, “Threshold optimization for F measure of macro-averaged precision and recall,” Pattern Recognit., vol. 102, 2020, doi: 10.1016/j.patcog.2020.107250.

[11] Y. Sasaki, “The truth of the F-measure,” Teach Tutor mater, pp. 1–5, 2007, [Online]. Available: http://www.cs.odu.edu/~mukka/cs795sum09dm/Lecturenotes/Day3/F-measure-YS-26Oct07.pdf.

[12] X. Zhang, X. Li, Y. Feng, and Z. Liu, “The use of ROC and AUC in the validation of objective image fusion evaluation metrics,” Signal Processing, vol. 115, pp. 38–48, 2015, doi: 10.1016/j.sigpro.2015.03.007.

[13] Z. Wang, T. Zhang, Y. Shao, and B. Ding, “LSTM-convolutional-BLSTM encoder-decoder network for minimum mean-square error approach to speech enhancement,” Appl. Acoust., vol. 172, p. 107647, 2021, doi: 10.1016/j.apacoust.2020.107647.

[14] B. Kamiński, M. Jakubczyk, and P. Szufel, “A framework for sensitivity analysis of decision trees,” Cent. Eur. J. Oper. Res., vol. 26, no. 1, pp. 135–159, 2018, doi: 10.1007/s10100-017-0479-6.

[15] J. R. Quinlan, “Simplifying decision trees,” Int. J. Hum. Comput. Stud., vol. 51, no. 2, pp. 497–510, 1999, doi: 10.1006/ijhc.1987.0321.

[16] B. Naik, “Extra-tree learning based Socio-economic factor analysis and multi-class adaptive boosting meta-estimator for prediction of agricultural productivity,” Indian J. Sci. Technol., vol. 13, no. 29, pp. 2081–2101, 2020, doi: 10.17485/ijst/v13i29.839.

[17] L. Sayfullina et al., “Efficient detection of zero-day android malware using normalized bernoulli naive bayes,” Proc. - 14th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. Trust. 2015, vol. 1, pp. 198–205, 2015, doi: 10.1109/Trustcom.2015.375.

[18] M. Ontivero-Ortega, A. Lage-Castellanos, G. Valente, R. Goebel, and M. Valdes-Sosa, “Fast Gaussian Naïve Bayes for searchlight classification analysis,” Neuroimage, vol. 163, pp. 471–479, 2017, doi: 10.1016/j.neuroimage.2017.09.001.

[19] V. Balakrishnan and W. Kaur, “String-based multinomial naïve bayes for emotion detection among facebook diabetes community,” Procedia Comput. Sci., vol. 159, pp. 30–37, 2019, doi: 10.1016/j.procs.2019.09.157.

# MrC: a Medical Record Chain System based on blockchain

Harel Jerbi[0000−0003−3280−5746], Shimon Shai Idan[0000−0003−3807−2598], and Hadassa Daltrophe[0000−0003−2305−125X]

Shamoon College of Engineering (SCE), Israel

Who is the owner of your health documents? Although the answer to this question may seem straightforward and intuitive, unfortunately, we are far from the situation where you are the one who has the key to this vital information. Hospitals, health maintenance organizations (HMO), private doctors, and different medical institutions produce large quantities of medical information about each of us, and there is a need to maintain this information privately and synchronously.

In this paper, we propose the medical record chain system (MrC), with its blockchain architecture, as a tool to achieve this goal.

The MrC system is connected to different medical service providers and obtains updated medical information about each patient. In this way, a patient will always have access to her or his medical data, which is organized in one place no matter where the information was produced.

In addition, it is the patient who can temporarily permit the service providers to expose any private medical information. On the other side, an incentive can be given to medical service providers to employ the system by allowing access to extensive medical data-set which have undergone de-identification.

Development and implementation of the MrC system will return ownership of the data to where it belongs—the patient; will improve patient's health by allowing multiple health providers to have more accurate information about them, and will advance public health aims by making broad de-identified data-sets more accessible for medical research.

# 1 Introduction

A common frustrating situation that almost every person facing during his life is the moment when you are going to get medical treatment in the hospital or taking an X-ray photo, and in the end, you are sent with a summary report or huge photo file on DVD and have been told to go to your HMO to continue your treatment. Sometimes the continuing treatment is not immediately, and there is a possibility that those summary/photos will be lost or damaged at home or that you forgot to take them to your doctor at the next meeting. Why in this technological era we even need to handle this kind of stuff? Why is all the system in the medical organization are not connected and update each other?

This paper suggests the medical record chain system (MrC) connecting all the service provider organizations to an external blockchain[1] based system. Since we don't want every patient to hold any other medical information besides his own, we choose to use consortium blockchain [2], where we can choose how to spread the data in the network and who will take part in the consensus algorithm.

Also, by using smart contract[3] and private Ethereum [2], we will have the ability to give each service provider and patient permission by the actual part they are taking in the system. For example, a patient can permit a specific doctor to expose his medical information, and now the doctor can add the medical records to the patient private chain.

## 1.1 Related work

Blockchain, which is a growing list of data that is maintained securely by a distributed network, was first suggested by Nkumoto [4] in the context of virtual currency (Bitcoin). Since then, the blockchain idea has extended to varied applications in many fields, including economics [5], transportation [6], culture [7]. The review conducted by Agbo et al. [8], identified some applications of blockchain technology in healthcare and highlighted state-of-the-art features of the development and limitations of the technology. A paper by Shahnaz et al. [1] suggests separating the system into three different layers. The first Shahnaz layer is the user layer, where each user (patient, doctors,

pharmacy, and so on) will connect with the other layers in the system, each user by his permission. In the second Shahnaz layer, users will implement the blockchain code and use proof-of-work [13] as a foundation for consensus, in which miners compete to append blocks. Each miner will experience a success probability in proportion to their computational effort. In the last Shahnaz layer, the application layer, all the functionalities, such as adding, updating, and deleting transactions, occur under the aegis of a smart contract. In this solution, the information (i.e., the transaction) is stored in the cloud. A different solution, presented in [14], suggests creating a digital archive for each patient; the archive will contain the patient's medical information. Access to this archive will happen by blockchain technology, with access to the individual's archive controlled by that patient. They use a Merkle tree; its primary functionality is to release, share, or save the medical record. All the medical information will be encrypted with the digital signature of the service provider charged with creating the record and the patient key. Here also, the data saved is encrypted in the cloud. In our suggested MrC system, we adopt these approaches to solving the consensus algorithm. Instead of using the cloud, we prefer to save the records in the distributed network as explain hereafter.

# 3 Overall Description

MrC is a new medical system base blockchain that supports the medical organization's internal medical system. Every user in the system will have two keys, a public key and a private one, both used for encryption and a digital signature. When the medical record has been creating, the internal system sends it to the MrC system automatically, and the private key of the patient will encrypt the information (could be text or file); the medical record will include four parameters: <new medical record, public key of the service provider, public key of the patient, patient ID>.

The system creates a' transaction' from the data that it gets and sends it to the entire network (the miners) for the verification and block creation process. We set the miners according to the size and fraction of the medical organizations in the country. In this way, a particular organization will not have control of more than 51% of the network, and thus the addition of blocks will be reliable according to **PoA [3]**.

The mining role will spread randomly between all miners in the system; when a miner gets the time to mine, he will take a transaction from his transaction's pool, creates a new block, and inserts the transaction into the block. Now the miner will publish the block via broadcast message to the other miners. The miners will notify that a new block has been added, and they can verify and post the block. The block has been added to the private chain of the patient that the transaction was about him.

Usually, MrC did not ask for information from the internal system of the medical organization; it is only listening. The only time the system asks for information is when the patient login to the system for the first time, and the system needs to build his private chain with his medical history. The service providers can see the patient medical record history only if he permits them, and the patient can delete the service provider from the list of those who have permission to his MrC.

## 4 System description

In a regular public blockchain, the entire network holds all the information of the system; however, in the MrC system, due to privacy reasons, the main public chain will contain only the keys with an encrypted pointer to a private chain represent each patient.

Many private medical organizations and doctors are not connected to any known medical service providers; hence, we must approve them with an authorized governmental entity, e.g., the ministry of health. The ministry of health's role will be to confirm all the service providers: hospitals, HMOs, private doctors, and others. Using the information about the medical organizations, the ministry of health split the permission to each medical organization and the miners [3].

Since every service provider might insist on keeping his original system, we will create a bridge between our system and the medical organization system. Every time a new medical record has been created in the internal system, it will be sent to our system, compressed, and the output will be encrypted with the patient's private key and inserted into the patient's private chain. In that way, the service provider kept their system, and yet each patient will have

one place that holds all his medical information. To encourage the medical organization to use the system, we consider two options: 1) we will let the medical organization get some anonymous medical records of patients for research. The medical organization achieves those anonymous medical records in exchange for their participation in the consensus algorithm (optional: let the medical organization choose what kind of medical record to get, for example, cancer/COVID-19, etc.)

2) The medical organization will earn currency for taking part in our system and running miner users; with those currencies, they will be able to pay tax to the government as property tax.

### 4.1 Blockchain data structure and decentralized network

The blockchain network has two parts, the main public chain and many short private chains, as described herein.

**Public chain.** The public chain is open to all service providers. The chain's blocks contain identification and public key exposure to everyone and some encrypted personal details like name and age. Within the public chain blocks, service providers can access the private chain that holds the patient's medical record. Access to the private chain, however, is allowed only if the patient provides the private key to the requesting provider. A new block is inserted into the public chain when a new child is born or a new person joins the system. The Ministry of Interior, which is the only organization that can add blocks, will manage the public chain.

**Private chain.** This chain holds the medical history of the patients. The first block in the private chain contains the patient's private information; subsequent blocks contain specific medical information about the patient. The only way to get to the first block is via the public chain. The hash value of the first block is a value in a dictionary at the main public chain. Patients cannot personally add, write, or otherwise create blocks (i.e., information), but they have read-only access to their record at any time. The patient can also create predetermined time limits when other service providers can read or write information. Subject to the constraints referred to above, the ability to search in this chain is constant because the medical information maintained in the chain is relatively small. (One might consider holding Merkle-tree structures inside the chain, which further make the system secure, stable, and efficient to search [12]). *Figure 2* provides a high-level illustration of the data structure.

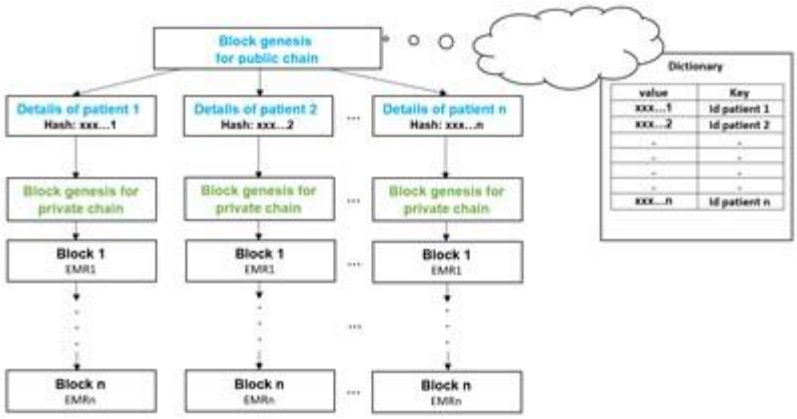



Fig. 1. *Illustration of the data structure. The first layer of a data structure is the main chain which is public. The patients are accessible using a dictionary. The second layer is the private chain of each patient which encrypted by public and private keys.*

Our network is based on different entities that include patients, institutional service providers, and private doctors. These entities will be designated as nodes and will be part of the complete peer-to-peer network. Every node will take a different role in our system based on its responsibility and the permissions granted. Following [10, 11], we state below different types of nodes that each entity might hold according to its nature:

- Patient node: A node that holds only the header block of the patient's private chain.
- Light node: A node that holds only the header of the block, without the transactions. This node is not updated about transactions and does not save them; thus, the light node requires a minimum amount of memory. Suitable for mobile and weak computers, light nodes will not be able to broadcast or verify transactions. The system will encompass a considerable amount of data, and not every node in the system needs to hold the entire amount of information.
- Miner node: A node that is responsible for mining. When a miner node broadcasts a new block to the network, a light node receives only the block's header, while a full node receives the complete block with the transaction.
- Full node: A node that holds and syncs with all relevant blocks in the net. A full node verifies the transactions and updates the blockchain; it can also send a broadcast message as needed.

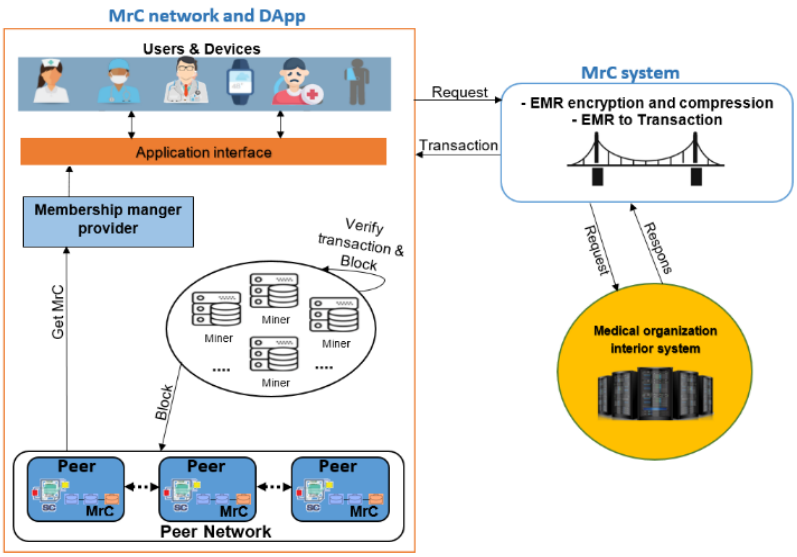



Fig. 2*: Context diagram for MrC system. At the baseline, we illustrate the decentralized medical network that holds the blockchain which is the main database of the MrC system. A patient can access his private medical information using his private key. A medical institution may read and write data from the chain by a dedicated protocol that sends to the overall network the information as a transaction for mining and verification.*

## 4.3 experimental study

As part of our final project for the B.sc. in software engineering, we implement a prototype of our system that simulates the main idea of the MrC. We used the virtual machine that Ethereum provides (EVM)[15] for building the blockchain network. We write the smart contract in "solidity"[16] to create permission for the service providers and create the MrC private chain of each patient. We ran and tested the system on our local machine using "truffle"[17] and "ganache"[17].

For the frontend part, we used "React", and for the connection between the "Dapp"[18] (distributed blockchain network application) and the Ethereum network, we used "web3"[19], an open-source software. Finally, using "metamask"[20] we created the digital account and entering the network.

## References


[1] A. Shahnaz, U. Qamar, and A. Khalid, "Using blockchain for electronic health records," IEEE Access, vol. 7, pp. 147 782–147 795, 2019.

[2] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in 2017 IEEE in- ternational congress on big data (BigData congress), IEEE, 2017, pp. 557–564.

[3] I. Bashir, Mastering Blockchain: Distributed ledger technology, decentraliza- tion, and smart contracts explained, 2nd Edition. Packt Publishing, 2018, isbn: 9781788838672. [Online]. Available: https://books.google.co.il/books?id=3ZlUDwAAQBAJ.

[4] Nakamoto, S. (2019). Bitcoin: A peer-to-peer electronic cash system. Manubot.

[5] Kelly, J. (2016). Banks adopting blockchain'dramatically faster'than expected: IBM. Retried on April, 10, 2018.

[6] Merkaš, Z., Perkov, D., & Bonin, V. (2020). The significance of blockchain technology in digital transformation of logistics and transportation. International Journal of E-Services and Mobile Applications (IJESMA), 12(1), 1-20.

[7] O'Dair, M., Beaven, Z., Neilson, D., Osborne, R., & Pacifico, P. (2016). Music on the Blockchain. Ekblaw, A., Azaria, A., Halamka, J. D., & Lippman, A. (2016, August). A Case Study for Blockchain in Healthcare: "MedRec" prototype for electronic health records and medical research data. In Proceedings of IEEE open & big data conference (Vol. 13, p. 13).

[8] Agbo, C. C., Mahmoud, Q. H., & Eklund, J. M. (2019, June). Blockchain technology in healthcare: a systematic review. In Healthcare (Vol. 7, No. 2, p. 56). Multidisciplinary Digital Publishing Institute.

[9] N. Rifi, E. Rachkidi, N. Agoulmine, and N. C. Taher, "Towards using blockchain technology for ehealth data access management," in 2017 fourth international conference on advances in biomedical engineering (ICABME), IEEE, 2017, pp. 1–4.

[10] P. Zhao, H. Cheng, Y. Fang, and X. Wang, "A secure storage strategy for blockchain based on mcmc algorithm," IEEE Access, vol. 8, pp. 160 815–160 824, 2020.

[11] C. Zhang, C. Xu, J. Xu, Y. Tang, and B. Choi, "Gem2-tree: A gas-efficient structure for authenticated range queries in blockchain," in 2019 IEEE 35th international conference on data engineering (ICDE), IEEE, 2019, pp. 842–853.

[12] Zhuang, Y., Sheets, L., Shae, Z., Tsai, J. J., & Shyu, C. R. (2018). Applying blockchain technology for health information exchange and persistent monitoring for clinical trials. In AMIA Annual Symposium Proceedings (Vol. 2018, p. 1167). American Medical Informatics Association.

[13] Zhang, P., White, J., Schmidt, D. C., & Lenz, G. (2017). Applying software patterns to address interoperability in blockchain-based healthcare apps. arXiv preprint arXiv:1706.03700.

[14] Chen, Y., Ding, S., Xu, Z., Zheng, H., & Yang, S. (2019). Blockchain-based medical records secure storage and medical service framework. Journal of medical systems, 43(1), 1-9.

[15] https://ethereum.org/en/developers/docs/evm/

[16] https://docs.soliditylang.org/en/v0.8.4/

[17] https://www.trufflesuite.com/

[18] https://en.wikipedia.org/wiki/Decentralized_application

[19] https://web3js.readthedocs.io/en/v1.3.4/

[20] https://metamask.io/

# The Time for Reconstructing the Attack Graph in DDoS Attacks

D. Barak-Pelleg[1,2] D. Berend[3,4]

**Abstract**

Despite their frequency, denial-of-service (DoS) and distributed-denial-of-service (DDoS) attacks are difficult to prevent and trace, thus posing a constant threat. One of the main defense techniques is to identify the source of attack by reconstructing the attack graph, and then filter the messages arriving from this source. One of the most common methods for reconstructing the attack graph is Probabilistic Packet Marking (PPM). We focus on edge-sampling, which is the most common method. Here, we study the time, in terms of the number of packets, the victim needs to reconstruct the attack graph when there is a single attacker. This random variable plays an important role in the reconstruction algorithm. Our main result is a determination of the asymptotic distribution of this time.

The process of reconstructing the attack graph is analogous to a version of the well-known coupon collector problem (with coupons having distinct probabilities). Thus, the results may be used in other applications of this problem.

*Keywords and phrases*: DoS attack, DDoS attack, probabilistic packet marking, edge-sampling, coupon collector problem.

2020 *Mathematics Subject Classification.* Primary 60C05, 60F99; Secondary 60G70.

## 1 Introduction

### 1.1 DDoS Attack and PPM

A *denial-of-service* (DoS) attack is a cyber attack in which the victim, a particular computer on the internet network, is assailed by a single attacker, seeking to make the victim unavailable for service. This goal is accomplished by flooding the victim with fake data packets until it is unable to fulfill legitimate requests, or even collapses. A distributed-denial-of-service (DDoS) attack is similar, but with multiple attackers. Both types of attack are common as they are quite easy to launch. Despite their frequency, these attacks are difficult to prevent and trace, thus posing a constant threat (see [14] for the latest DDoS attack news).

Several defense techniques and tools are available to deal with these attacks; usually, a combination of approaches is employed (see, for example, [15, 26] for surveys on defense techniques). One of the main approaches is to identify the source of attack, and then filter the messages arriving from this source. There are a few methods to implement this approach [1]. One of these methods is by reconstructing the attack graph. This graph is a tree type graph, in which the root represents the victim, the leaves represent the attackers, and the internal nodes represent the routers connecting

[1]Department of Mathematics, Ben-Gurion University, Beer Sheva 84105, Israel. E-mail: dinabar@post.bgu.ac.il

[2]Research supported in part by the Cyber Security Research Center at Ben-Gurion University.

[3]Departments of Mathematics and Computer Science, Ben-Gurion University, Beer Sheva 84105, Israel. E-mail: berend@math.bgu.ac.il

[4]Research supported in part by the Milken Families Foundation Chair in Mathematics and the Cyber Security Research Center at Ben-Gurion University.

the attackers to the victim. (Thus, in a DoS attack, the graph comprises a path.) There are various methods for reconstructing the attack graph (see [12]). In the current work we focus on Probabilistic Packet Marking (PPM), introduced in [3]. Specifically, we deal with edge-sampling, the most common method used in PPM.

In edge-sampling, there are two processes taking place simultaneously. The first is on the routers side: Each router in the network, upon receiving a packet, and before forwarding it, decides at random, with some probability $p$ (fixed for all routers), whether to mark it or not. If the packet has already been marked by a previous router, the new mark will override the old one. Thus, the probability of a packet received by the victim to have been last marked by the router at distance $i$ from him is $p(1-p)^{i-1}$. When a router marks a packet, it writes there its identity, and the next router (if it does not override the mark) adds to it its own identity and starts a counter. When any router farther along the path decides not to override the mark, it increases the counter by 1. Thus, when the victim receives a marked packet, the mark consists of the edge in the attack path corresponding to the (last) marking router and the router following it, and the distance of this edge from the victim. The second process is on the victim's side: The victim collects the marks in order to reconstruct the attack graph.

The victim starts collecting marks upon suspecting he is under attack; that is, when there is a sudden jump in the arrival rate of packets. When should this process be terminated? Namely, when should the victim decide it has obtained enough data in order to reconstruct the full attack graph? On the one hand, the longer the victim continues collecting marks, the greater the chance of being able to reconstruct the full attack graph. On the other hand, if the victim waits too long, it might collapse by the flood of incoming packets. The time (in terms of the number of packets) the victim needs in order to reconstruct the full attack graph, when there is a single attacker, is also referred to as the Completion Condition Number [22]. This random variable, which we will denote by $D$, plays an important role in the reconstruction algorithm. Savage, Wetherall, Karlin, and Anderson [21] considered the expected number of packets needed $E(D)$, and showed that $E(D) \leq \ln n/(p(1-p)^{n-1})$, where $n$ is the distance of the attacker from the victim. Thus, they suggested to wait until obtaining $\ln n/(p(1-p)^{n-1})$ packets. Sairam and Saurabh [22] showed that, in many cases, this number of packets may not be enough. They up-bounded the standard deviation of $D$ and suggested to add a third of this bound to the above bound on $E(D)$, thus increasing the reliability of the algorithm.

The process of obtaining the marks by the victim is analogous to a version of the coupon collector problem [21–23, 25]. We now recall this classical problem.

### 1.2 The Coupon Collector Problem

Suppose that a company distributes packages of some product and that each package contains a single coupon. There are $n$ types of coupons, and a customer wants to collect them all. Each time that he buys a package, he gets one of the types uniformly at random. We want to know how many packages need to be purchased on the average until getting all types of coupons. The problem goes back at least as far as de Moivre who mentioned it in a collection of problems regarding various games of chance [17]. The solution to this problem has been known for many years; the expected number of coupons we need to draw is $nH_n$, where $H_n = 1 + 1/2 + 1/3 + \cdots + 1/n$ is the $n$-th harmonic number. Asymptotically, this expectation is $n \ln n + \gamma n + O(1)$, where $\gamma = 0.577\ldots$ is the Euler-Mascheroni constant.

The problem, and various extensions thereof, have drawn much attention for many years (see, for example, [4, 8, 13, 18, 20]; see also the survey [5]). One of the extensions, considered by von Schelling [24], and Flajolet, Gardy, and Thimonier [7], dealing with the case where various coupons show up

with distinct probabilities, turns out to be very relevant to our problem. In the next subsection we will see that the reconstruction of the attack graph is naturally translated to this variant.

### 1.3 Edge-Sampling and Coupon Collecting

As mentioned above, in a DoS attack, the attack graph is just a path. Denote its vertices by $v_0, \ldots, v_n$, where the vertex $v_0$ represents the victim and $v_n$ represents the attacker, and its edges by $e_i = \{v_{i-1}, v_i\}$, $1 \leq i \leq n$. Each $e_i$ represents the link between the router at distance $i-1$ with that at distance $i$ from the victim.

To connect the reconstruction problem with the coupon collector problem, we match the victim of the DoS attack with the coupon collector, and each $e_i$ with the $i$-th type coupon. The event "the victim has obtained a packet marked by the link at distance of $i-1$ from him" is translated to "the coupon collector has received a coupon of type $i$". The victim has obtained the marks of all links of the attack path exactly when the collector has obtained all coupon types.

As indicated above, the version of the coupon collector problem we have here is where the coupons have distinct probabilities. Each coupon type $i$ shows with probability $p_i = p(1-p)^{i-1}$. Note that the sum of these probabilities is $\sum_{i=1}^{n} p_i = 1 - (1-p)^n < 1$, as at each step there is a probability of $(1-p)^n$ to obtain an unmarked packet. Thus, it will be convenient for us to add a "dummy" coupon of type 0, whose probability is $p_0 = (1-p)^n$, and a corresponding "dummy" edge $e_0$ to the attack path. This addition is inconsequential for the following reason. We take the marking probability to be $p = \lambda/n$, for some arbitrary fixed $\lambda > 0$, and assume that $n$ is large. Hence all "real" coupons have probabilities $\Theta(1/n)$, while the probability of the dummy coupon is $\Theta(1)$. The probability for the dummy coupon to be obtained last is therefore extremely small. Whether the goal is to collect only all real coupons, or it is to collect also the dummy one, is immaterial; the dummy coupon will anyway (most probably) arrive long before all real coupons have arrived.

### 1.4 Paper Organization

In Section 2 we define a continuous analogue of our problem, which is more convenient to deal with than our discrete model. Next we state the main results, first for the continuous version, and then for the discrete one. Section 3 presents results of simulations we preformed for both models.

## 2 Main Results

Let us first consider a continuous model, analogous to our problem. The idea of using a continuous model has been used several times in the classical case (see [10, 11]). In this model there are $n$ independent, incoming flows of coupons:

$$T_1 \sim \mathrm{Exp}\left(p_1\right), \ldots, T_n \sim \mathrm{Exp}\left(p_n\right),$$

where $T_i$ is the waiting time for the $i$-th type of coupon. Same as in the regular model, we are interested in the waiting time until all coupon types arrived. Differently from the regular model, the waiting times are exponential instead of geometric. Also, in the continuous model the variables are independent, whereas in the discrete model they are not. Thus, the probability that the $i$-th coupon type has not been seen until time $t$ is

$$e^{-p_i t} = e^{-\lambda/n(1-\lambda/n)^{i-1} t}.$$

Denote by $T$ the time until we get all coupons in the continuous model. Thus

$$T = \max_{1 \le i \le n} T_i.$$

Given a sequence $(X_n)_{n=1}^{\infty}$ of random variables and a probability law $\mathcal{L}$, write $X_n \xrightarrow[n\to\infty]{\mathcal{D}} \mathcal{L}$ if the sequence converges to $\mathcal{L}$ in distribution. Recall that a random variable $X$ is Gumbel distributed with parameters $\mu \in \mathbf{R}$ and $\beta > 0$, and we write $X \sim \text{Gumbel}(\mu, \beta)$, if its distribution function is given by [9]:

$$F(x) = e^{-e^{-(x-\mu)/\beta}}, \qquad x \in \mathbf{R}. \tag{1}$$

**Theorem 1.** *The asymptotic distribution of the waiting time for all coupons in the continuous model is given by:*

$$\frac{T - (e^{\lambda}/\lambda) \cdot n(\log n - \log\log n)}{n} \xrightarrow[n\to\infty]{\mathcal{D}} \text{Gumbel}\left(-\tfrac{e^{\lambda}}{\lambda}\log\lambda, \tfrac{e^{\lambda}}{\lambda}\right).$$

We will actually prove the following stronger version of the theorem, which provides information about the rate of convergence in Theorem 1. Denote:

$$T' = \frac{T - (e^{\lambda}/\lambda) \cdot n(\log n - \log\log n)}{n}. \tag{2}$$

**Theorem 1′.** *For $t' \in \mathbf{R}$ and $n \to \infty$,*

$$F_{T'}(t') = \exp\left(-e^{-\left(t' - \left(-e^{\lambda}\log\lambda/\lambda\right)\right)/\left(e^{\lambda}/\lambda\right)}\right) + O\left(\log\log n/\log n\right).$$

Note that the rate we obtain is rather slow, which goes hand in hand with the rate of convergence of other quantities in the coupon collector problem [2, 10].

**Theorem 2.** *The expected time until we get all coupons in the continuous model is*

$$E(T) = \tfrac{e^{\lambda}}{\lambda} \cdot n\left(\log n - \log\log n + \gamma - \log\lambda\right) + O\left(n\log\log n/\log n\right)$$

*as $n \to \infty$.*

Getting back to the discrete model, recall that $D$ is the number of coupons we need to collect in order to get all real types in the discrete case. Similarly to (2), denote:

$$D' = \frac{D - (e^{\lambda}/\lambda) \cdot n(\log n - \log\log n)}{n}.$$

**Theorem 3.** *The asymptotic distribution and the expectation of the time required for reconstructing the attack graph are given by:*

*a.*

$$D' \xrightarrow[n\to\infty]{\mathcal{D}} \text{Gumbel}\left(-\tfrac{e^{\lambda}}{\lambda}\log\lambda, \tfrac{e^{\lambda}}{\lambda}\right). \tag{3}$$

*Moreover, as $n \to \infty$,*

$$F_{D'}(d') = \exp\left(-e^{-\left(d' - \left(-e^{\lambda}\log\lambda/\lambda\right)\right)/\left(e^{\lambda}/\lambda\right)}\right) + O\left(\log\log n/\log n\right), \qquad d' \in \mathbf{R}. \tag{4}$$

*b. As $n \to \infty$,*

$$E(D) = \tfrac{e^{\lambda}}{\lambda} \cdot n\left(\log n - \log\log n + \gamma - \log\lambda\right) + O\left(n\log\log n/\log n\right).$$

**Remark 4.** We could have used [19, Th.2.1] to prove (3). However, this would not have saved us any of the work, as the main step is to prove that [19, Assumption 2.1] holds. Moreover, our proof is designed so as to obtain the quantitative estimate (4).

**Remark 5.** According to the theorem, the reconstruction time is roughly proportional to $e^\lambda/\lambda$. For $\lambda > 0$, the expression $e^\lambda/\lambda$ is minimal at $\lambda = 1$ (see Figure 1). Hence, as $n \to \infty$, the expectation $E(D)$ will be minimal very close to the point $\lambda = 1$. Thus, the optimal choice for the edge-sampling algorithm is $p = 1/n$ (as claimed by Savage et al. [21, p.300]).

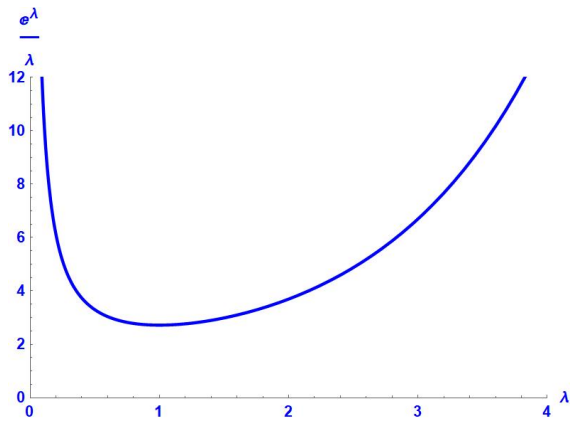



Figure 1: The effect of the coefficient $\lambda$ on the reconstruction time

## 3 Simulation Results

We have performed a simulation for the time needed to collect all types of (real) coupons. In our experiments, $\lambda = 1$, $n = 10^4$, and the number of iterations of each test is $M = 10^5$. Everything has been performed on Mathematica, and we point out several technical points that may be on interest to its users.

The simulation was preformed for both the discrete model and the continuous one. In the discrete case, in each of the $M$ runs we have drawn the coupons one by one, each drawing being independent of the others. In each drawing, the coupon of type $i$ was selected with probability $p_i = 1/n(1-1/n)^{i-1}, 1 \le i \le n$. We have continued the process until all $n$ types of real coupons have been drawn and saved the number of drawings. Thus, we have obtained a list of length $M$, of the times at which the various iterations completed their runs.

In the continuous case, in each of the $M$ iterations we selected $n$ random exponential variates with parameters $p_1, \ldots, p_n$, and took their maximum.

In Table 1, the first two columns present the sample means (rounded to the nearest integer) received in the two experiments. The third column shows the (identical) main term $en\,(\log n - \log\log n + \gamma)$ on the right-hand side in our expressions for $E(T)$ and $E(D)$ – Theorems 2 and 3.b, respectively. The last column presents the order of magnitude of the error term in both results, namely $n \log\log n/\log n$. Note that the two means are relatively very close, and both are in line with the theoretical main term, given the allowed error.

Not only the sample means are close, as may be seen in Table 1. In Figure 2 we present the (smoothed) PDFs of the simulation data for both models (using the default option "PDF" in

| $\overline{D}_M$ | $\overline{T}_M$ | $en\,(\log n - \log\log n + \gamma)$ | $n \log\log n / \log n$ |
|---|---|---|---|
| 207945 | 207885 | 205699 | 2410 |

Table 1: The sample means vs. the theoretical results on the expectation.

Mathematica's SmoothHistogram). The results for the discrete model presented by the smooth red line and those of the continuous by the dashed gray line.

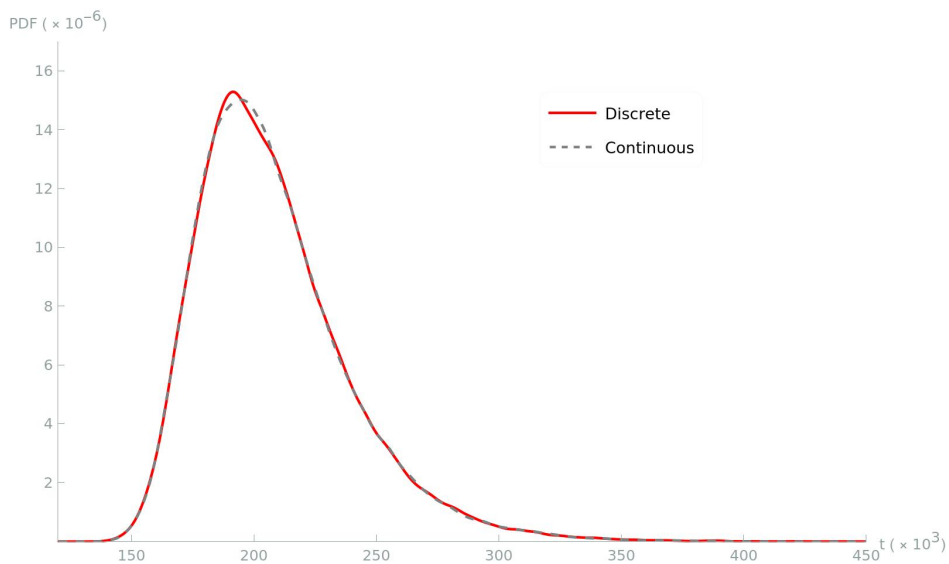



Figure 2: Smoothed histogram for the results of both models.

We have utilised the function FindDistribution to find Mathematica's guess for the most fitting distribution for the sample of $D$. We have repeated the simulation several times. In most cases, Mathematica guessed that the sample data is from a Gumbel distribution, but this was not always the case. In the simulation we have presented here, we received three guesses:

(i) First, we have given FindDistribution only the data of the simulation, without any "hints" as to the required distribution. In this case, Mathematica guessed that the sample from $D$ is from a mixture of two basic distributions:

$$0.73 \times \text{Gamma}(101, 1934) + 0.27 \times \text{LogNormal}(12.33, 0.19).$$

(ii) Second, we have added the option MaxItems $\to$ 1, which yields a single, best fitting distribution for the data. In this case, Mathematica's guess was Gubmel$(192983, 25762)$.

(iii) We have noticed that the mean and variance of the distribution suggested in the second guess do not fit those of our sample. Thus, we have specified for Mathematica to find the most fitting Gubmel distribution by utilising the option TargetFunctions $\to$ {ExtremeValueDistribution} (which is the name Mathematica uses for the distribution called Gumbel in our paper). In this case, Mathematica's guess was Gubmel$(193800, 24506)$.

In Figure 3 we present five graphs, generated by Mathematica. Four of them are based on the simulation data for the discrete model and the last depicts the prediction of the theoretical result. The red continuous line presents the (smoothed) probability density function of the simulation data,

same as in Figure 2. The three dashed lines present the probability density function of the three guesses (i)-(iii) above of Mathematica for the distribution most fitting the simulation data. The first is presented by a blue line of small dashes, the second – by a green line of medium dashes, and the third – by a black line of large dashes. The solid cyan line presents the density function of the $\text{Gumbel}(en\log n - en\log\log n, en) = \text{Gumbel}(190008, 27183)$ distribution. This distribution is the approximation of the distribution of $D$, corresponding to the approximation of the distribution of $D'$ by $\text{Gumbel}(0, e)$, as in (3).

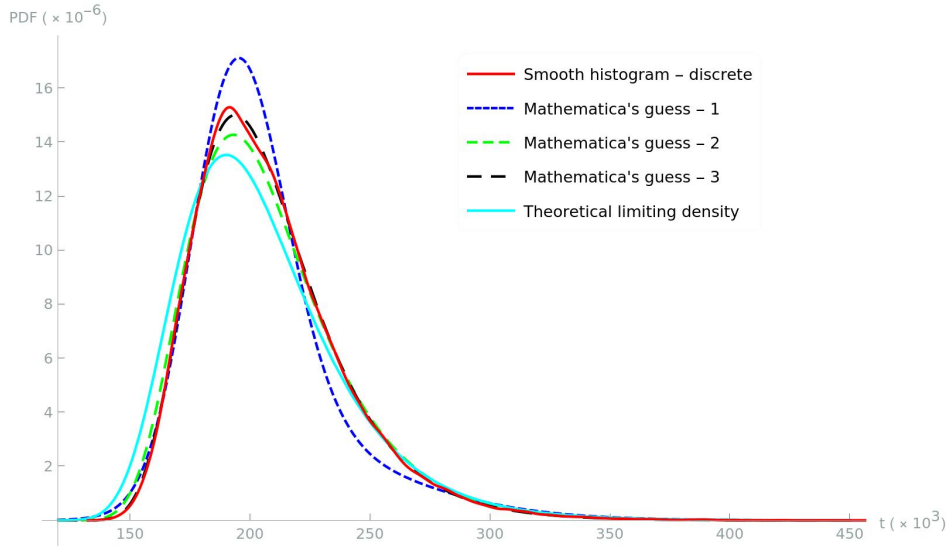



Figure 3: Smoothed histogram of simulation vs. Mathematica's guesses and the theoretical limiting distribution PDFs.

Note that here, when providing Mathematica with the hypothesized distribution type, Gumbel, we have an estimation problem of two unknown parameters $\mu$ and $\beta$. The simplest way to estimate these parameters is by the method of moments [16]. Employing Mathematica's EstimatedDistribution with the option ParameterEstimator → "MethodOfMoments", we get the same parameters as guess (iii) above. Recall that the method of moments estimator employs the sample moments to estimate the parameters. Thus, as expected, in this case we get a Gumbel distribution whose expectation and variance fit the sample mean and sample variance. For maximum likelihood estimation [6], the parameters are given implicitly, and thus more difficult to obtain. Employing Mathematica's EstimatedDistribution with the option ParameterEstimator → "MaximumLikelihood", we get $\text{Gubmel}(193878, 24218)$. In Figure 4 we depict three graphs, generated by Mathematica. As in Figure 3, the red continuous line represents the (smoothed) probability density function of the simulation data. The magenta dotted line is of the Gumbel distribution whose parameters were estimated by the method of moments, and the blue dashed line – for the maximum likelihood estimator.

In Figure 5 we have three CDF graphs. The two cyan solid lines represent the theoretical result in the second part of Theorem 3.a, in which we also add (and subtract) $0.25\log\log n/\log n$, namely a quarter of the expression in error term in (4). Thus, the top graph is of the function $F((t - en\log n + en\log\log n)/n) + 0.25\log\log n/\log n$, where $F(t) = \exp(-e^{-t/e})$. The bottom graph is of $F((t - en\log n + en\log\log n)/n) - 0.25\log\log n/\log n$. Finally, we illustrate the closeness of the simulation results to the theoretical result by adding the CDF of the smoothed histogram of the discrete simulation (using the option "CDF" in Mathematica's SmoothHistogram), This graph

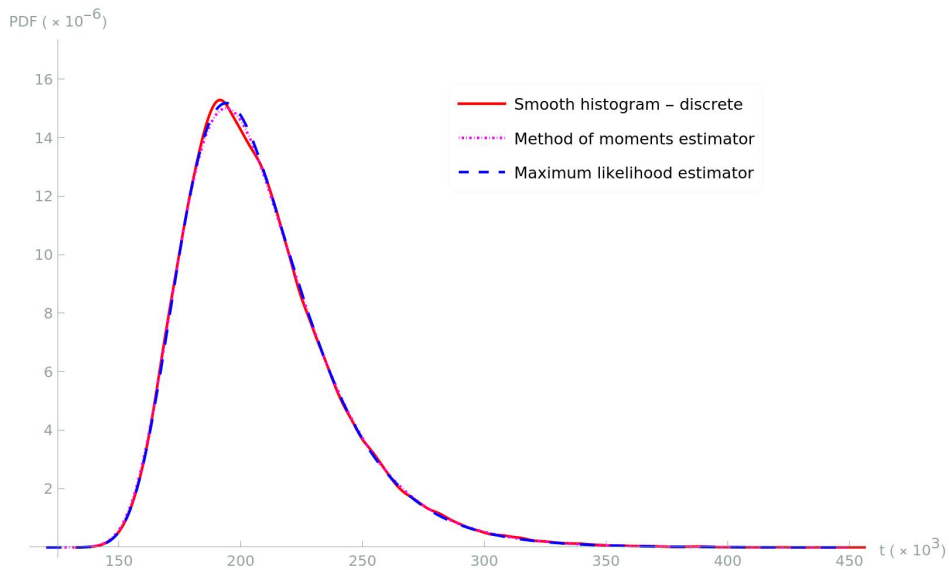



Figure 4: Smoothed histogram of simulation data vs. Gumbel PDFs with parameters estimated by the method of moments and maximum likelihood.

appears as a dashed red line, bounded between the two perturbations of the theoretical prediction.

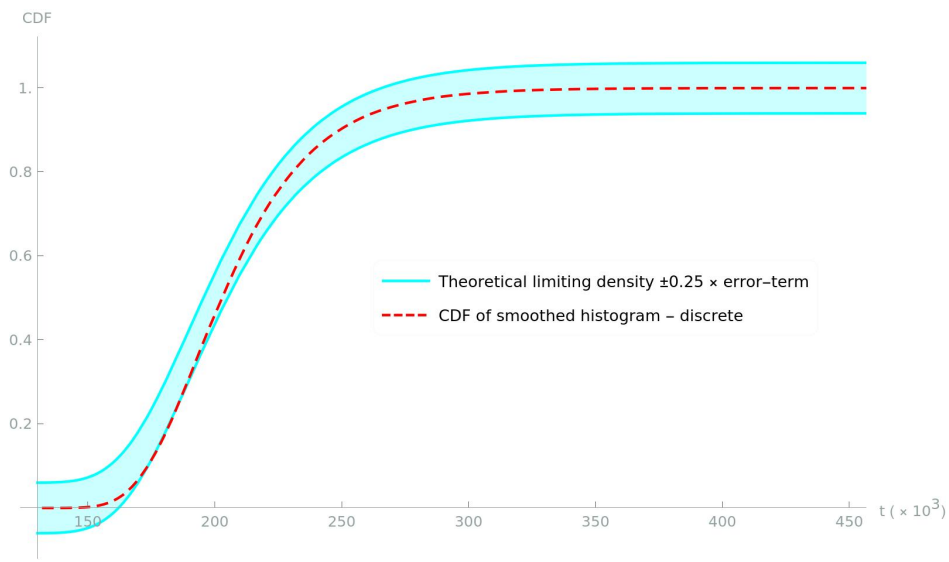



Figure 5: The CDF of the theoretical limiting distribution +/- twice the error term vs. the CDF of the smoothed histogram of the simulation.

## References

[1] Andrey Belenky and Nirwan Ansari, *On IP Traceback*, IEEE Communications Magazine 41(7) (2003), 142–153.

[2] Robert K. Brayton, *On the Asymptotic Behavior of the Number of Trials Necessary to Complete a Set With Random Selection*, Journal of Mathematical Analysis and Applications 7(1) (1963), 31–61.

[3] Hal Burch and Bill Cheswick, *Tracing Anonymous Packets to Their Approximate Source*, Computer Networks. In: Usenix LISA 2000 (1999), 319–327.

[4] Paul Erdős and Alfréd Rényi, *On a Classical Problem of Probability Theory*, Publ. Math. Inst. Hung. Acad. Sci., Ser. A 6 (1961), 215–220.

[5] Marco Ferrante and Monica Saltalamacchia, *The Coupon Collector's Problem*, Materials Matemàtics 2 (2014), 35.

[6] Mauro Fiorentino and Salvatore Gabriele, *A Correction for the Bias of Maximum-Likelihood Estimators of Gumbel Parameters*, Journal of Hydrology 73(1-2) (1984), 39–49.

[7] Philippe Flajolet, Danièle Gardy, and Loÿs Thimonier, *Birthday Paradox, Coupon Collectors, Caching Algorithms and Self-Organizing Search*, Discrete Applied Mathematics 39(3) (1992), 207–229.

[8] Leopold Flatto, *The Dixie Cup Problem and FKG Inequality*, High Frequency 2(3–4) (2019), 1–6.

[9] Emil J. Gumbel, *Statistical Theory of Extreme Values and Some Practical Applications. A Series of Lectures*, National Bureau of Standards Applied Mathematics Series No. 33, U. S. Government Printing Office, Washington, D. C. (1954), viii+51.

[10] Lars Holst, *On Birthday, Collectors', Occupancy and Other Classical Urn Problems*, International Statistical Review (1986), 15–27.

[11] Andrii Ilienko, *Limit Theorems in the Extended Coupon Collector's Problem* (2020) arXiv preprint arXiv:2002.00650.

[12] Ankunda R. Kiremire, Matthias R. Brust, and Vir V. Phoha, *Topology-Dependent Performance of Attack Graph Reconstruction in PPM-Based IP Traceback*, 2014 IEEE 11th Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA (2014), 363–370.

[13] Pierre-Simon Laplace. *Théorie Analytique des Probabilités*, Vol. II, 1812, Éditions Jacques Gabay, Paris, 1995 (Reprint of the 1820 third edition).

[14] Latest DDoS Attack News, The Daily Swig (2021, June 16). *https://portswigger.net/daily-swig/ddos*

[15] Georgios Loukas and Gülay Öke, *Protection Against Denial of Service Attacks: A Survey*, The Computer Journal 53(7) (2009), 1020–1037.

[16] Smail Mahdi and Myrtene Cenac, *Estimating Parameters of Gumbel Distribution using the Methods of Moments, probability weighted Moments and maximum likelihood*, Revista de Matemática: Teoría y Aplicaciones, (2005), 1409-2433.

[17] Abraham de Moivre, *The Doctrine of Chances*, 1756, republished 1967 by Chelsea: New York.

[18] Amy N. Myers and Herbert S. Wilf, *Some New Aspects of the Coupon Collector's Problem*, SIAM Review 48(3) (2006), 549–565.

[19] Peter Neal, *The Generalised Coupon Collector Problem*, Journal of Applied Probability 45(3) (2008), 621–629.

[20] Donald J. Newman and Lawrence Shepp, *The Double Dixie Cup Problem*, The American Mathematical Monthly 67(1) (1960), 58–61.

[21] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson, *Practical Network Support for IP Traceback*, Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (2000), 295–306.

[22] Ashok S. Sairam and Samant Saurabh, *A More Accurate Completion Condition for Attack-Graph Reconstruction in Probabilistic Packet Marking Algorithm*, 2013 National Conference on Communications (NCC), IEEE (2013), 1–5.

[23] Ashok S. Sairam and Samant Saurabh, *Increasing Accuracy and Reliability of IP Traceback for DDoS Attack Using Completion Condition*, International Journal of Network Security 18(2) (2016), 224–234.

[24] Hermann von Schelling, *Coupon Collecting for Unequal Probabilities*, The American Mathematical Monthly 61 (1954), 306–311.

[25] Shigeo Shioda, *Some Upper and Lower Bounds on the Coupon Collector Problem*, Journal of Computational and Applied Mathematics 200(1) (2007), 154–167.

[26] Saman T. Zargar, James Joshi, and David Tipper, *A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks*, IEEE Communications Surveys & Tutorials 15(4) (2013), 2046–2069.

# Automatic Real Time Platoon Formation *
**(Preliminary Version)**

Shlomi Dolev, Ehud Gudes, Ayushi Srivastava and Hannah Yair**.

Ben-Gurion University of the Negev, Be'er Sheva, Israel
{dolev@cs,ehud@cs,hannaya@post}.bgu.ac.il

**Abstract.** Identifying traffic platoons and managing vehicles on the road effectively is a challenging task that is currently investigated both in academia and industry. The challenges include the need for fast real-time gathering of relevant information, such as vehicles location, moving direction and speed, and instructing the vehicles in real-time to respect traffic policies according to the gathered information.
In this work we present new algorithms to define platoons that are updated dynamically on the fly, allowing much better control over the traffic to gain efficiency. A platoon representative vehicle is chosen and the set of vehicles in the platoon is identified based on inductive distance criteria, that are continuously checked, updating the platoon memberships. In this paper, we present the main algorithms to identify and control the platoon and demonstrate this detection using a vehicle simulator.

**Keywords:** Platoon · Platoon identification · Vehicle-to-vehicle (V2V)· Vehicle-to-infrastructure (V2I)· Connected-vehicle (CV)· Inter-vehicle (IV) · Cooperative-adaptive-cruise-control (CACC).

## 1 Introduction

The realization of computer (totally) controlled vehicles is approaching fast. One facet of the computer control is based on remote driving, where no driver seats in the car. Such capability becomes economic when a single remote driver can drive many vehicles instead of only one car. The assistance of a computer system can leverage the capabilities of a single driver to define procedures to be applied to a group of vehicles –procedures that when executed on the controlling computer initiate commands to many vehicles in parallel. In particular, such procedures are based on the concept of *Platoon*. A platoon is defined as a group of cars moving in the same direction as a selected car, called, *representative vehicle*, and at a distance of each other which is less than a given *threshold*. The advantages of creating a platoon are well-known [1] and include increased safety and fuel saving.

* This research was partially funded by the Andromeda MAGNET Consortium, by the Lynne and William Frankel Center for Computer Science and by Rita Altura chair in computer science.
** Corresponding auther

Whenever there is a command for platoon creation around a representative vehicle, our scheme represents the geographic roads map in the area as a directed graph whose nodes are junctions and the edges represent the lanes connecting certain junctions. For every edge we define the vehicles moving on the corresponding lane and vice versa. Assume the platoon is created around a representative vehicle $R$. We calculate the distances between a candidate vehicle $C$ and every platoon member $P_i$ (that moves in the same direction). If $C$ resides within the threshold distance $X$ from any platoon member, then $C$ is admitted into the platoon. The threshold $X$ depends on the velocities of the cars (i.e., the larger are the velocities, the larger is $X$). The platoon calculation propagates along the edges that are adjacent to the edge $R$ resides on. We take the candidates currently residing on these edges, calculate their distances from the platoon members and merge a candidate $C$ into the platoon if $C$ is sufficiently close to some platoon member.

The main contribution of our work is the new dynamic and history agnostic definition of platoons. In the sequel, we compare our on-the-fly dynamic definition of platoons with existing methods in the literature. We also demonstrate the flexible and robust platoon definition using, SUMO, a well-known vehicles simulator. [2]

The rest of this paper is structured as follows. Section 2 discusses related work, Section 3 defines the problems and presents the algorithms to solve them, Section 4 presents an example of a simulator run showing a platoon, and Section 5 concludes the paper.

## 2 Related Work

An overview on platooning systems appears in [3], where the following recent vehicle platoon projects are summarized:

- ***SARTRE*** that defines a platoon as a collection of vehicles when the leader of the platoon is a heavy vehicle that is driven remotely and all the rest vehicles follow the leader. In addition, the platoon is dynamic and communication is V2V that is based on ITS-G5.

- ***PATH*** that is based on the assumption that all the vehicles are automatic including the leader. In 1994 PATH first tested the control of a four-car platoon at four meters separation at highway speeds,in 1997 PATH developed an eight-car automated platoon for the National Automated Highway System Consortium (NAHSC) Demo '97. The communication is V2V that is based on sensor signal processing, coordination, and lateral and longitudinal control.

- ***GCDC*** In 2011 Grand Cooperative Driving Challenge developed a system for controlling platoon automatically and it is achieved by a combination of

V2V & V2I communication and advanced sensor fusion and control.

- ***EnergyITS*** A national ITS project by the Japanese Ministry of Economy, Trade, and Industry, started in 2008, interested in an automated truck platoon and an evaluation method of the effectiveness of ITS on energy saving.

- ***SCANIA − platooning*** Scania's main interest in platooning is hence focused on heavy vehicle platooning on highways with focus on minimizing fuel consumption. This interest is expressed in two national Swedish projects:
  - Distributed Control of a Heavy-Duty Vehicle Platoon:
    The main focus of the project is how a single-vehicle operating in a platoon should be efficiently controlled without jeopardizing safety, The control architecture has been developed based on distributed control, the communication is V2V that is based on sensors information like radar, cameras, etc,
  - iQFleet:
    The project covers several research topics where platooning is one. Platooning research is focused on how platoons should be controlled with respect to other road users, road topology, infrastructure, etc.

In addition to the above platoon projects described in [3] we reviewed several more recent platoon projects. These are described and compared briefly in the following table. Table 1 summarizes the main differences among recent platooning projects.

Table 1: Comparison of platooning projects,Projects(NA stands for Not-Applicable

| Reference | Communication type | Dynamic platoon | Traffic condition | Vehicle type | Purpose | Identification algorithm |
|---|---|---|---|---|---|---|
| Tiaprasert et al. [4] | V2V communication and GPS device | Yes | Critical time-headway. | NA | Mathematical model for real-time platoon recognition using CV technology | Based on geometric distribution with parameters |
| Rajamani et al. [5] | IV communication | No | Highways by achieving significantly higher traffic flow | Eight fully automated cars | Integrated longitudinal and lateral control system for the operation of automated vehicles in platoons | NA |
| Tsugawa et al. [6] | IV communications | No | Expressway before public use | 3-fully automated trucks | Saving and global warming prevention | NA |

| | | | | | | |
|---|---|---|---|---|---|---|
| Chen et al. [7] | V2I and V2V communication | Yes | A high-density segment of highway. | NA | Enhancing traffic efficiency and sustainability | Focused on the radar-based CACC |
| Praveen et al. [8] | NA | Yes | A four lane divided urban road | Real time traffic | To understand the platooning phenomenon under heterogeneous traffic | Based on camera-videos data |
| Browand et al. [9] | NA | No | An unused airfield runway | 2 trucks | Fuel saving | NA |
| Zhao et al. [1] | V2I and V2V communication | Yes | High variation in urban traffic flow | Mixed automated vehicles and human-driven vehicle | To minimise the fuel consumption for platoons and drive the platoons to pass the intersection on a green phase | Based on similar traffic state |
| He et al. [10] | V2I communications | Yes | Cumulative headways or critical headways | Transit buses and passenger and bicycles. | To optimize arterial (network) traffic signals for multiple travel modes | Based on geometric distribution with parameters |
| Our work | V2I communication using GPS | Yes | NA | NA | Identify platoons and control the traffic remotely | New algorithm that is based on graph methods |

The next section presents the new algorithm that we have developed

## 3 Automatic real-time platoon formation

As was mentioned above, a platoon is defined as a group of cars moving in the same direction defied around a selected representative vehicle. The platoon consists of every vehicle that is at a distance less than threshed distance from a vehicle that belongs to the platoon. Our scheme represents the map area as a directed graph whose nodes are junctions and the edges represent the lanes connecting neighboring junctions. For every edge we define the vehicles moving on the corresponding lane and vice versa. Assume the platoon is created around a representative vehicle $R$. We calculate the distances between a candidate vehicle $C$ and every platoon member $P_i$. If $C$ resides within the threshold distance $X$ from any platoon member, then $C$ is admitted into the platoon. $X$ may depend on the velocities of the cars (i.e., the higher are the velocities, the higher is $X$). The platoon calculation propagates along the edges that are adjacent to the

edge that $R$ resides on. We take the candidates currently residing on these edges, calculate their distances from the platoon members and merge a candidate $C$ into the platoon if $C$ is sufficiently close to some platoon member.

### 3.1 Platoon definition criteria

- Vehicle location, this criterion is a value obtained as a function of time and vehicle according to GPS location, and marked:

$$location(x,t) = location\ of\ x\ b\ GPS\ in\ time\ t \tag{1}$$

- Direction of travel, this criterion is a value obtained as a function of time and vehicle according to the road number obtained by the location of the GPS, and marked:

$$direction(x,t) = lane\ location\ of\ x\ in\ time\ t \tag{2}$$

- Vehicle speed, this criterion is a value obtained as a function of time and vehicle by location at two consecutive time points, and marked:

$$speed(x,t) = speed\ of\ x\ in\ time\ t \tag{3}$$

- Distance between two vehicles moving in the same direction, this criterion is a value obtained as a function of time and two vehicles moving in the same direction, according to their location and marked:

$$dist(x_1,x_2,t) = the\ distance\ between\ x_1\ to\ x_2 \tag{4}$$

  Where the distance is calculated by the shortest path along the same lane, according to the algorithm map road between $x_1$ to $x_2$ in time $t$.

- Minimum distance, this criterion is a value obtained as a function of time and two vehicles moving in the same direction, according to their speed and a fixed and marked value

$$minDist(x_1,x_2,t) = X + \left(\frac{speed(x_1,t)}{t_v}\right) \tag{5}$$

  Where $X$ is a given threshold and $t_v = 3.7[m/s]$ is a velocity threshold that was evaluated in [11]

- Minimum distance indicator, this criterion is a Boolean value obtained as a function of time and two vehicles moving in the same direction, according to the minimum distance and marked:

$$indMinDist(x_1,x_2,t) = \begin{cases} T\ ,if\ \ dist(x_1,x_2,t) \leq minDist(x_1,x_2,t) \\ F\ ,if\ \ dist(x_1,x_2,t) > minDist(x_1,x_2,t) \end{cases} \tag{6}$$

– Platoon affiliation indicator, this criterion is a Boolean value obtained as a function of time and two vehicles moving in the same direction, according to the minimum distance and marked:

$$indPlatoon(x_1, x_2, t) = \begin{cases} T, \ if \ indMinDist(x_1, x_2, t) = T \\ F, \ if \ indMinDist(x_1, x_2, t) = F \end{cases} \tag{7}$$

– Platoon, this criterion is a set of vehicles traveling in the same direction for which the minimum distance indicator receives a true value and is obtained as a function of time and a group of vehicles moving in the same direction and marked:

$$isPlatoon(X, t) = \{x | x \in X \ and \ \exists y \in X \ suchthat \ indPlatoon(x, y, t) = T\} \tag{8}$$

– isLeader, this criterion identifies the first vehicle in platoon obtained as a function of time, direction and group of vehicles moving in the same direction, according to the location of the vehicle and marked:

$$isLeader(X, direction, t) = first, \ according \ to \ the \ traveling \ direction. \tag{9}$$

### 3.2 Outline of the algorithm

We first build a scheme that represents the map area as a directed graph whose nodes are junctions and the edges represent the lanes connecting the junctions. When we get the representative vehicle $R$, we collect all the information of $R$ and the vehicles that are on the same edge of $R$ .

We build two sorted groups *pos*, *neg* where *pos* is the group of vehicles in front of $R$ and *neg* includes the cars that are behind $R$. We build these groups in iterative fashion, adding all vehicles that are within the threshold distance from the last added vehicles.

Once all the vehicles on the current edge are considered, the platoon calculation propagates along the edges that are adjacent to the edge $R$ resides on.

In Fig. 1, we present the transformation of part of the (SUMO) simulation map(that, in turn, can be based on google maps) to a directed graph to calculate the platoons in the code. The lanes are represented by straight lines, and the nodes represent the location just before and just after the junction for each lane.

In Fig. 2, we illustrate the algorithm operation by three arrows that describe the calculation of the platoon members, one strait arrow with the red vehicle direction describes the calculation operation of the members in front of the red vehicle, and the two other arrows describe the calculation operation of the members that are behind of the red vehicle.

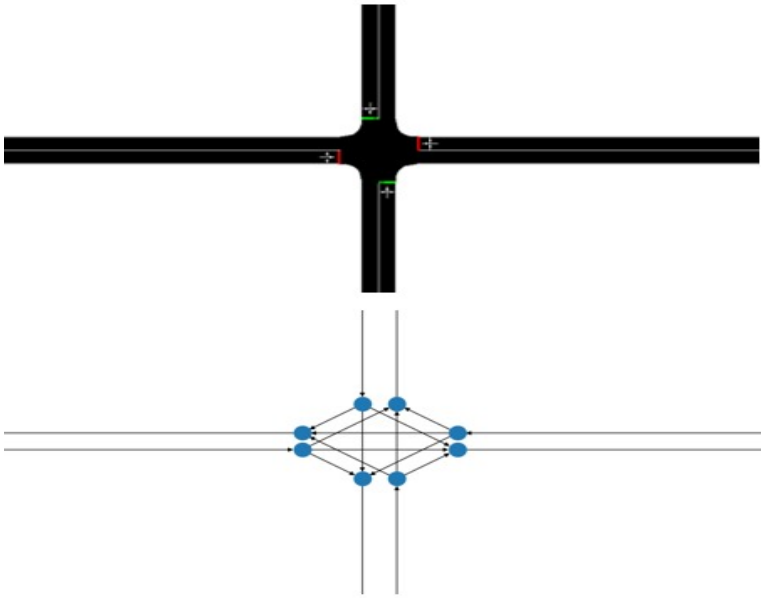


**Fig. 1.** Map transformation

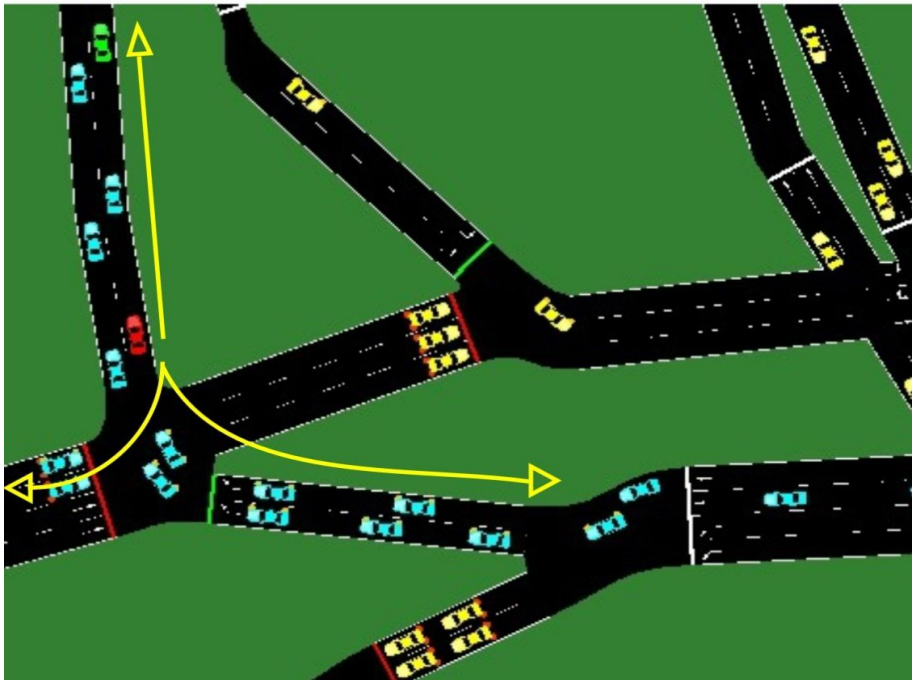


**Fig. 2.** Simulator snapshot

### 3.3 Implementation of the algorithm

Alg. 1 constructs the graph from the road map. In the first part of the algorithm(lines 3-9), we build edges and vertices objects according to the data obtained from the database of the simulator, in the second part of the algorithm(lines 10-14), we build the connection edge objects according to the data obtained from the details of each junction contained in the database of the simulator.

**Algorithm 1:** buildGraph(*lanes*, *junctionsLinks*):

```
   Result: Build map
1  nodes=[]
2  edges=[]
3  for lane in lanes do
4      nodes['from'+lane]=NodeF([],[])
5      nodes['to'+lane]=NodeT([],[])
6      edge[lane]=Edge(NodeF,NodeT,[],length)
7      nodes['to'+lane][0].add(Edge)
8      nodes['from'+lane][1].add(Edge)
9  end
10 for link in junctionsLinks do
11     edge[link[0]+link[1]]=Edge(link[0],link[1],[],length)
12     nodes['to'+link[0]][1].add(Edge)
13     nodes['from'+link[1][0].add(Edge)
14 end
```

In Alg. 2, we calculate the members of each platoon per time-stamp according to the road map and representative vehicle. We construct two lists Neg and Pos that include all the vehicles that are in front-of or behind-of the representative vehicle in the platoon respectively.

**Algorithm 2:** buildVehiclesList(*representative*,*bd*):

```
   Result: Create platoon
1  currentEdge = db.getEdge(representative)
2  pos= the vehicles front of the representative in the current edge in
     sorted list
3  neg = the vehicles behind the representative in the current edge in
     reverse sorted list
4  Neg = getNegVehiclesPlatoon(currentEdge, representative, neg, bd)
5  Pos = getPosVehiclesPlatoon(currentEdge, representative, pos, bd)
6  return Pos + Neg
```

In Alg. 3, we calculate the members of the platoon which are in front of the representative vehicle, per time-stamp according to the road map and vehicles located in the road map. The algorithm begins with the representative vehicle

as a current vehicle and with a sorted list of vehicles that are located in front-of-the current vehicle on the same edge. After that the algorithm scans the list and calculates the distance between current vehicle to the first vehicle in the list and check(lines 4-6) if the distance is close enough according to the thresholds $X$ and $t_v$ if no,(lines 7-9) the algorithm stops the scanning and returns an empty list, else,(lines 2-10) the algorithm update current vehicle to be the first vehicle in the list and check if the distance between current vehicle to the third vehicle in the list is close enough according to the thresholds $X$ and $t_v$ if not, the algorithm stops the scanning and returns the current list, else continue the list scanning as mentioned above. After successful scanning on the list of vehicles that located a front-of-the current vehicle in the same edge we pass to the out-edges according to the map and calculate the same in a recursive why.(lines 11-21)
We calculate the behind-to representative vehicle members of each platoon per time-stamp according to the road map and vehicles located in the road map in a similar way and therefore we do not present it here.

**Algorithm 3:** getPosVehiclesPlatoon(*currentEdge*, *currentVehicle*, *posList*, *bd*):

```
Result: all the vehicles that are front of the representative in the
        platoon
1  realList = []
2  for v in posList: do
3  |  d = bd.getDistanceBetweenTwoVehicles(currentVehicle,v)
4  |  if d <= ((X + currentVehicle.getSpeed()) / t_v): then
5  |  |  realList.append(v)
6  |  |  currentVehicle = v
7  |  else
8  |  |  return realList
9  |  end
10 end
11 v = the last vehicle in the pos list
12 neighborsEdges = bd.getNeighbors(currentEdge)
13 for n in neighborsEdges: do
14 |  currentEdge = bd.getEdge(n)
15 |  pos = currentEdge.getSortedVehiclesOn()
16 |  realList = realList+getPosVehiclesPlatoon(currentEdge,v,pos,bd)
17 end
18 return realist
```

We like to explain further the problem of computing distances. The distance is not simply the Euclidean distance between two vehicles, since that on a curved road will be wrong. On a curved road, we take instead the set of dense points which separate the two vehicles on the same lane and add their distances. The set of dense points is given by the simulator road map. The same distance calculation is made for constructing the two sorted lists in Algorithm 2.

## 4 Analysis of our algorithm

Objects definitions:
$n_t$ is the count of vehicles in the simulation in time $t$.
$n_{e_t}$ is the count of vehicles in the simulation on edge $e$ in time $t$.
$n_{p_t}$ is the count of vehicles in the simulation in platoon $p$ in time $t$.
Time complexity of the update of the simulation per unit-time:

- *all the vehicles updating location.* $\Theta(n_t)$
- *all the platoons updating.*
  - *get all the vehicles in the platoon.* $O(n_{p_t})$
  - *sort the vehicles group in the platoon.* $O(n_{p_t} \log(n_{p_t}))$
  - *delete all the vehicles that do not belong to the platoon.*$O(n_{p_{t-1}} - n_{p_t})$

Total time complexity:
$O(n_t + n_{p_t} + n_{p_t} \log(n_{p_t}) + (n_{p_{t-1}} - n_{p_t})) \leq O(n_t + n_t \log(n_t)) \leq O(n \log(n))$

We ran our experiments on a PC that is equipped with :
Memory: 8.0GB
Disk: SAMSUMG MZVLB512HBJQ-000L7
CPU: Intel® Core™ i5-8500 CPU @3.00GHz

Fig. 3 shows the execution unit-time result in the graph that describes the time required for execution per unit-time of updating the simulation as a function of the count of vehicles and count of platoons in the simulation and we can see that the execution performs gathering of information, platoon formation, and notification within one second.

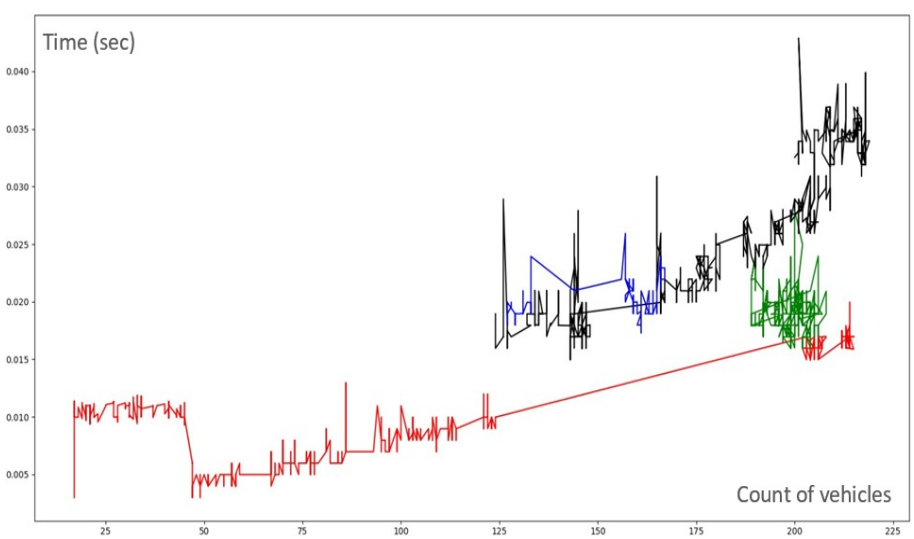



**Fig. 3.** Red= 0 platoons, Black= 1 platoons, Green= 2 platoons, Blue= 3 platoons

From the figure we see that the platoon calculation time increases with the number of vehicles and the number of platoons. However, the highest total time (40ms) is still well below the required threshold time of 1 second.

## 5 Simulator demonstration

In Fig. 2 and Fig. 4, we present different parts of the simulator road map and the platoon construction algorithm output which results from the simulation. In Fig. 2, we present one platoon identified by the algorithm, around the representative vehicle, and in Fig. 4, we present two platoons, where the red vehicles are the representatives, the green vehicles are the leaders of the platoon, the blue vehicles are the platoon members and the yellow vehicles are non-platoon vehicles.

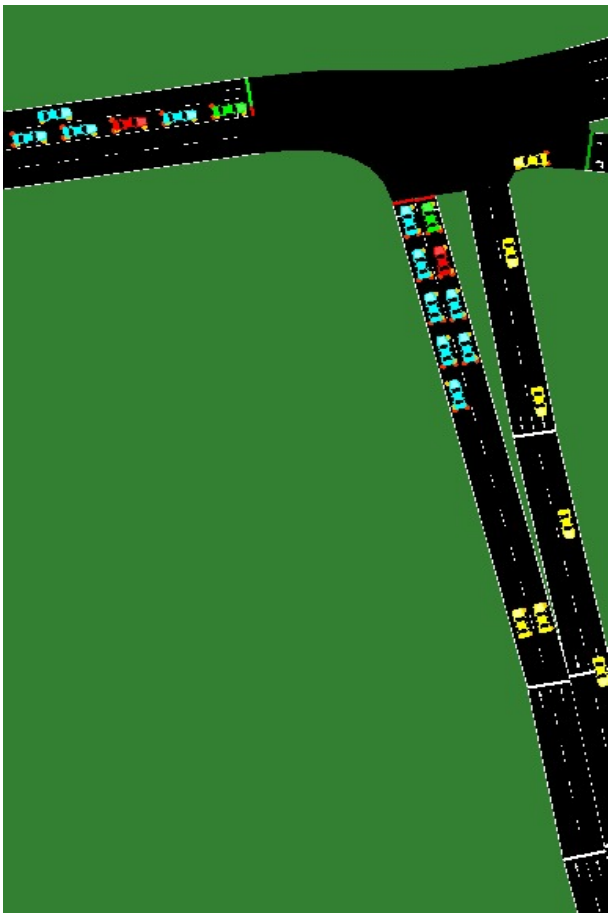


**Fig. 4.**

## 6 Conclusions

Few existing platoon recognition algorithms are based on GPS. However, none of them are based on the map of the roads and on representative vehicles. Our paper proposed a platoon recognition algorithm that is based on the map of the roads, representative vehicle, and GPS location. In addition, our algorithm can recognize platoons in any kind of roads straight or curved. This work can be extended to scheduling the road traffic according to the platoons identification.

## References

1. Weiming Zhao, Dong Ngoduy, Simon Shepherd, Ronghui Liu, and Markos Papageorgiou. A platoon based cooperative eco-driving model for mixed automated and human-driven vehicles at a signalised intersection. *Transportation Research Part C: Emerging Technologies*, 95:802–821, 2018.
2. Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.
3. Carl Bergenhem, Henrik Pettersson, Erik Coelingh, Cristofer Englund, Steven Shladover, Sadayuki Tsugawa, and Magnus Adolfsson. Overview of platooning systems. In *Proceedings of the 19th ITS World Congress :*, pages 1–7, 2012.
4. Kamonthep Tiaprasert, Yunlong Zhang, and Xin Ye. Platoon recognition using connected vehicle technology. *J. Intell. Transp. Syst.*, 23(1):12–27, 2019.
5. Rajesh Rajamani, Han-Shue Tan, Boon Kait Law, and Wei-Bin Zhang. Demonstration of integrated longitudinal and lateral control for the operation of automated vehicles in platoons. *IEEE Trans. Control. Syst. Technol.*, 8(4):695–708, 2000.
6. Sadayuki Tsugawa, Shin Kato, and Keiji Aoki. An automated truck platoon for energy saving. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2011, San Francisco, CA, USA, September 25-30, 2011*, pages 4109–4114. IEEE, 2011.
7. Zheng Chen and Byungkyu Brian Park. Preceding vehicle identification for cooperative adaptive cruise control platoon forming. *IEEE Trans. Intell. Transp. Syst.*, 21(1):308–320, 2020.
8. Praveen P.S. and Ashalatha R. Identification of platoon dispersion pattern under heterogeneous traffic conditions. *Case Studies on Transport Policy*, 8(1):101–111, 2020.
9. Fred Browand, John McArthur, and Charles Radovich. Fuel Saving Achieved in the Field Test of Two Tandem Trucks. Institute of Transportation Studies, Research Reports, Working Papers, Proceedings qt29v570mm, Institute of Transportation Studies, UC Berkeley, June 2004.
10. Qing He, K. Larry Head, and Jun Ding. Pamscod: Platoon-based arterial multimodal signal control with online data. *Transportation Research Part C: Emerging Technologies*, 20(1):164–184, 2012. Special issue on Optimization in Public Transport+ISTT2011.
11. Wikipedia. Two-second rule.

# Encoded video motion vectors as an alternative to a GPU processor used by OP to extract human skeletal movement*

Y. Segal and O. Hadar

*Communication Systems Engineering Department,*
*Ben Gurion University of the Negev (BGU),*
*Beer-Sheva, 84105,*
*Israel*

**Abstract.** Home exercises are significant in the rehabilitation process of physiotherapy patients, which lack immediate feedback as to the proper movement and therefore might humper patient treatment. To overcome this challenge, we offer a deep learning-based system that classifies the accuracy and quality of movements in videos. We capture body movement using OpenPose, which is a commonly used software package that extracts 25 body vertexes from a frame. However, in order to provide sufficient real-time tracking, OpenPose requires a Graphical Processing Unit (GPU), which is not available in a typical domestic setup. Our novel solution allows using OpenPose software at a meager frame rate with a CPU setup. We fill the missing gaps with a novel tracking algorithm based on Motion Vectors (MVs), which are extracted from the encoded body motion video. This innovative concept allows us to estimate and mimic a 30fps OpenPose point set with a regular CPU without any GPU support. We compare the accuracy results of our body vertex movement tracking algorithm with that of the OpenPose and show excellent matching results. We further propose a new accuracy assessment formula that takes into consideration the body vertex position confidence level, as it is extracted from the OpenPose software. In order to ensure consistent pose classification of our method, we further propose a Siamese Neural network classifier to compare our results to those of the OpenPose. A Siamese network is constructed of two identical CNN channels that fuse into one distance measurement layer. The distance measurement layer measures the Euclidean distance between the results of the two CNN channels and produces a scoring function that reflects the degree of similarity between the images entered into each of the Siamese network channels. The inputs to our Siamese network are two separate skeletal pose frames. The network indicates whether the pose of the two inputs is identical or different. This allows us to test the accuracy of classification of our outcome results. Overall, we show an improvement of x15 in the time it takes to process movements between our algorithm and the native OpenPose with excellent accuracy metrics.

* This work was supported by the Israel Innovation Authority (Formerly the Office of the Chief Scientist and MATIMOP).

**Keywords:** OpenPose; Video Encoder: Motion Vectors; Siamese Neural Network; Physical Rehabilitation; Telemedicine

## 1 Introduction

Telemedicine is a new and very popular field of medicine. It allows moving periodical examination from the medical clinic to the home environment. This approach is more comfortable for patients and also lowers treatment cost. Important enablers of telemedicine are suitable information technology and input sensors. Our goal is to allow home physiotherapeutic exercises using a common web camera (webcam). In order to facilitate such a setup, it should not require more than a standard domestic computing power and it should obtain sufficient accuracy using a home webcam. We have developed a novel tracking algorithm that leveraged the OpenPose [1] [2] [3] software to capture body vertex movements. However, the OpenPose requires a substantial Graphic Processor Unit (GPU), which is not available in a standard domestic setup and takes a significant time to run. Our novel solution uses OpenPose software at a meager frame rate while filling the gaps with a tracking algorithm, which is based on Motion Vector (MVs) [4] extracted from encoded video. This innovative concept allows us to estimate and mimic a 30fps OpenPose point set with a regular CPU without any GPU support. We further developed a camera system that allows precise recording of probands from lateral and frontal direction, allowing comparison with common web cameras. This approach can record motion and preprocess the signal for use with machine learning. the main building block of this article is the body's skeletal tracking system. There are precise systems with perfect tracking possibilities, but the price is high, and usually, assistance needed before or during operation makes it impossible to use them in the home environment. Our research focuses on a system with affordable prices and as easy use as possible. For body tracking, there is something like a gold standard. It is an expensive and complicated infrared marker-based system. One well-known system is the first commercial motion capture system in the early 1980s Vicon [5]. Similar systems with similar features are OptiTrack [6], Qualysis [7], or BTS [8]. Those systems based on multiple synchronous infrared cameras, many IR reactive markers, and body models. Advantages: Very precise, Long-term research and many existing bodies models. Disadvantages: Very expensive, long preparation for measurement, the need for qualified operators, and Impossible to use in the home environment. In this research, we are going to use OpenPose algorithm that offer good skeleton extraction quality. The alternative can be poseNe [9] or WrnchAI [10]. The research of OpenPose is presented in detail by authors of the system and publications [1], [2], [3]. The pose estimation is based on pre-trained body parts models. Models assembled using deep NNs. The system is frame-based. Pose estimation is done separately for every single image. Microsoft Kinect is one of the most famous 3D scanners [11]. The example of a full-body commercial capturing system can be Xsens [12], Rokoko [13], or [14] smart capture suit. The systems can be for full-body or specific applications. Dublin University presented a framework [15] for capturing the 3D trajectory of a golf swing or other sports equipment. The approach base on inertial sensors with accelerometers, gyroscopes, and magnetic sensors. They can obtain a 3D position in space, applying mathematics

transformation, Kalman filters from acceleration, angular velocity, and gravitational forces. In the following sections, we present the tracking algorithm. From there, we present the traceability quality by measuring the accuracy and the impact on classification networks. Next, we present the practical implementation of the traceability algorithm and, finally, the results of the experiments we performed while discussing the results and drawing conclusions.

## 2 Tracking algorithm

### 2.1 The challenge

The OpenPose software provides excellent pose estimation results. Therefore, we use it to serve as our motion capture and video processing algorithm. However, there are several limitations to use such a program. The most noticeable is that the resources required for OpenPose are significant since it requires a powerful GPU unit, which is not affordable for our target domestic end-users. Lower quality CPU will require a significant time (about 30-40 minutes) to process a 1-minute video clip. A sample image of an OpenPose skeletal vertexes extraction is illustrated in **Fig. 1**.

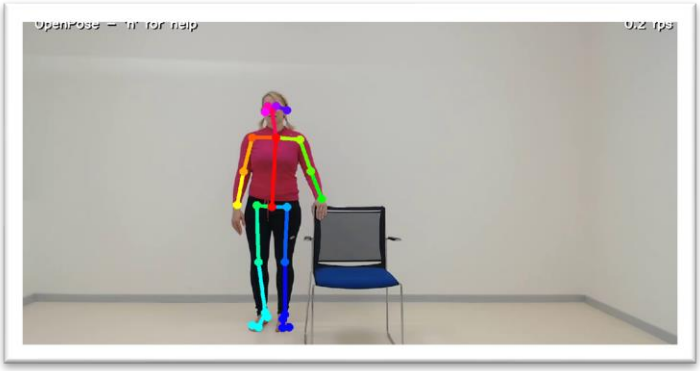


***Fig. 1.*** *OpenPose* human body vertexes

Our novel solution uses OpenPose software at a meager frame rate (about 1-3 frames per second). We developed a tracking algorithm to fill the gap of the missing frames. The idea is to extract Motion Vectors (MVs) from the H.265\H.264 encoded video [16] The MVs can be used to predict the movement of each skeleton vertex between consecutive frames. This innovative concept allows us to estimate and mimic a 30fps OpenPose point set with a regular CPU and without GPU support. We succeeded to converts a 1fps OpenPose skeletal vertexes location set into a 30fps OpenPose location set with a standard CPU that performs all processes top-down, including 1fps OpenPose processing time and the subsequent tracking algorithm.

### 2.2 Skeletal vertex position estimation

We feed a full-frame rate encoded camera video stream into the OpenPose. The video is fed twice to the OpenPose: At the first time, we feed it into OpenPose running on a PC with a powerful GPU. As a result, we can predict all skeleton vertexes at a full-frame rate - 30fps (see **Fig. 2** A). The second time, we run the OpenPose on a PC without GPU. Since we use PC without GPU, the Open Pose predicts skeletal body positions at a low frame rate (see **Fig. 2** B). Our tracking algorithm mission is to fill in the missing frames to generate full-frame rate OpenPose skeletal vertex position estimation (see **Fig. 2** C). OpenPose contracts the placement of a human skeleton in a frame with the help of 25 points in two-dimensional space. According to the video compression standard, the encoding algorithm broke down each video frame into macro-blocks. Each OpenPose vertex, locates in one of those macro-blocks. Assuming that there is movement between frames, then the video encoder builds a prediction of the next video frame by copying each macroblock from the previous frame into the new frame (Obviously, it is only a prediction, and according to the encoding algorithm, there is an error correction phase which is out of the scope for this article). Each moving macroblock in the new frame has a vector. The vector points to the macroblock location in the previous frame, which the encoding algorithm estimated as a similar macroblock. This vector is called the Motion Vector (MV). Because each vertex is in a macroblock, and each moving block between frames has a motion vector attached to it, the moving vertex's position in the new frame can be estimated (We name this process as Tracking algorithm). We start with a given Open pose skeletal frame, and for each vertex in a missing frame, we repeat the tracking algorithm. To reduce biased estimation, we restart the tracking algorithm per Group Of Poses(GOP). In other words, whenever we have an OpenPose estimation, we use it as a GOP starting point for our tracking algorithm. In **Fig. 2** D. The abstraction of the tracking algorithm's expected output is present in **Fig. 2** D. It presents the achievement of producing full-frame-rate skeletal movement estimation without a GPU. It is a mixture of OpenPose estimations and tracking algorithm estimations. A red dot represents a 2D position of one specific skeletal vertex in the frame. We demonstrate the vertex locations flow between frames where the blue arrow represents the MV. It presents the GOP divisions, as well, where each GOP is initiate by the

OpenPose skeleton, low fps, prediction as it presents in **Fig. 2** B.

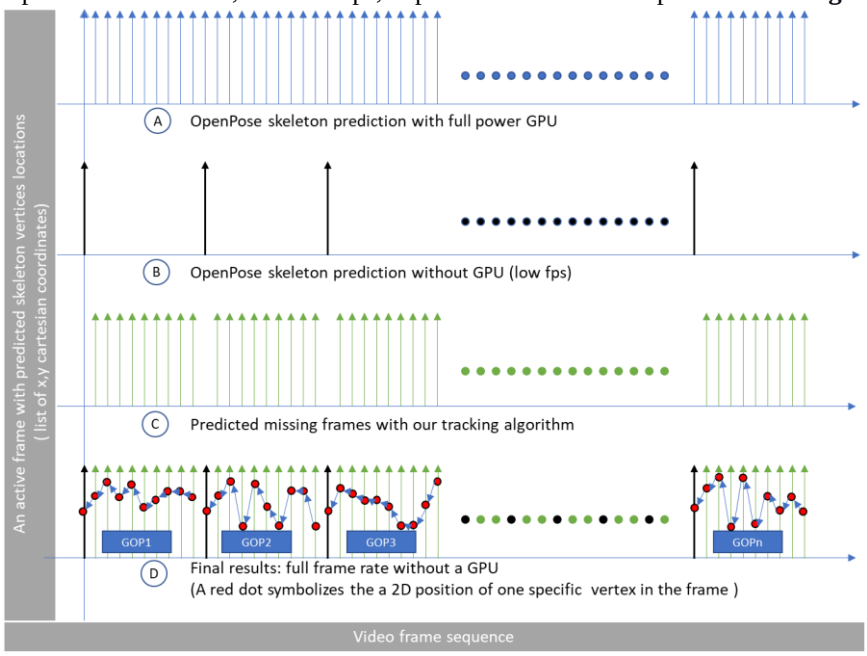



**Fig. 2.** Basic Tracing and Prediction concept

A-frame containing OpenPose body skeletal following by N frame contains our estimated tracking algorithm that predicts new skeletal locations called Group of Poses(GOP). We perform the prediction of each next frame in the GOP as follows: according to the encoded video algorithm protocol, i.e., H.264/H.265, a macroblock that moves from one location in the current frame to a new location on the next frame produces a Motion Vector (MV) - A vector that points from the new location to the previses location. We are interested only in the macroblocks that contain skeletal vertex positions. In our case, the MV direction estimates the new macroblock location in the next frame, i.e., our skeletal vertex new location. The encoding algorithm protocol provides for each frame a list of macroblocks and their corresponding MV. Given the body vertex coordinate in the current macroblock frame, we need to search in the next frame for an MV that points to our macroblock that contains our skeletal vertex. If we cannot find a matching MV in a predefine search radius, we assume that the vertex remains stationary and duplicate its location. To predict the skeletal body coordinates of the next frame, we repeat this process for all 25 body vertexes and all missing frames. Once we predict skeletal body location in the next frame, we use it as a reference and continue our tracking process to the next frame till the end of the GOP. Once we finish predicting a GOP, we start to predict the next GOP with the next OpenPose skeletal position as our initiate tracking algorithm location. Macroblocks can be of different shapes and sizes according to H.264 standard, and we considered it.

# 3 Measurement of trace accuracy and its effect on the classification process

We measure the prediction accuracy in two different ways:

(1) We measure the Waited Mean Square Error (**Eq.** (1)) between our predicted skeletal (**Fig. 2** D) and the OpenPose 30fps Skeletal (**Fig. 2** A).
(2) We measure and compare the influence of our prediction on the Neural Network Classifier (Autoencoder Classifier and Siamese network Classifier).

## 3.1 Waited Mean Square Error (WMSE) measurement

We propose a new and modified metric to assess accuracy, using the vertex position confidence level, which is available from the OpenPose software. We call the new metric Weighted Mean Square Error (WMSE) and calculate it according to Eq. (1). The OpenPose predict $N_0$ vertexes $v$ per frame where $v \in \mathbb{R}^3$. Let us define the OpenPose predicted vertex i, that is located in frame j as $v_{ij} = (x_{ij}, y_{ij}, c_{ij})$. Accordingly, we mark $v'_{ij} = (x'_{ij}, y'_{ij}, c_{ij})$ as our corresponding predicted vertex (On behalf of compatible vector representation, we replicate c to be the same in both $v_{ij}$ and $v'_{ij}$). Were $x, y$ is the vertex coordinate in the frame and $c$ is the OpenPose vertex location confidence level. The index i is the vertex id in the frame and j is the frame index in the movie where i ={1 … $N_0$} and j={1 … $N_F$}. We used a powerful PC with a GPU to produce full-frame rate OpenPose skeletal vertexes as our ground truth to measure our prediction accuracy. We use the WMSE to compare our prediction $v'_{ij}$ and the OpenPose prediction $v_{ij}$.

$$WMSE = \frac{1}{N_o N_F} \sum_{i=1}^{N_o} \sum_{j=1}^{N_F} C_{ij} \left(v_{ij} - v'_{ij}\right)^2 \qquad (1)$$

Where:

$N_o$ - are number of skeletal vertexes in a frame

i - is a vertex ID in a single frame i ={1 ... $N_0$}

$N_F$ - are number of frames in a movie session

j - is a frame ID in a movie session j ={1 ... $N_F$}

$C_{ij}$ - is the OpenPose vertex prediction confidence level

$v_{ij}$ - is a skeletal vertex that OpenPose predict its location

$v'_{ij}$ - is a skeletal vertex that our algorithm predict its location

$v, v'_{ij} \in \mathbb{R}^3$

### 3.2 Influence on Neural Network Classifiers

**Autoencoder Classifier**

We want to use our newly 30fps predicted JSON file to train a model that can classify and tag human skeletal movements. We use autoencoder CNN (Convolutional Neural Network) [3], as seen in **Fig. 3**. We reconstruct the JSON file coordinates into a Black/White skeletal image, and we fed it to the CNN. The autoencoder produces a latent vector in the middle of the network per input image.

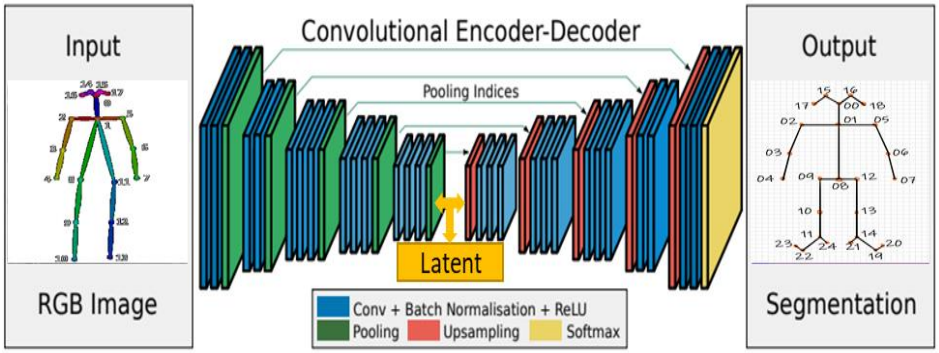



**Fig. 3.** CNN and Latent vector

The latent vectors serve as tagging mechanism for each pose in the movie. We measure the Euclidean distance between two latent vectors. If the distance is less than a predefined threshold, we claim that the two skeletal images are the same. We use the above mechanism to compare classification results between regular OpenPose 30fps Skeletal and our predicted skeletal. We demonstrate that our approach (without GPU) can achieve the same results as an OpenPose that runs on a powerful GPU. This method has a significant disadvantage. First, It takes a long time huge database to train the autoencoder. It must be done both for the encoding and decoding process. The decoding procedure is unnecessary for our purposes. The network only uses the training phase to find the optimal latent vector; the autoencoder encodes and decodes the image. Then, it compares the input image with the output image to achieve as minor an error as possible between those two. We declare that the network structure and waits that produce the lowest error contain the latent vector that best describes and refined the input image. I.e., it is the image label. Autoencoder classifier produces an optimal latent vector. Therefore, similar images will produce similar latent vectors. We switched from an autoencoder network to a Siamese twin's network to improve performance and reduce database size. The encoding phase in the two networks is similar, but we use a new neural network layer that measures distances between two latent vectors instead of the decoding part. The criteria are to produce a minimum distance between two similar images (see the following section for more information about the Siamese twin's network).

**Siamese network classifier.**

To validate the classification accuracy of the results, we have used a Neural Network classifier with our predicted and reconstructed images, compared to the OpenPose ground truth images. The classifier is based on Siamese network [17]. The Siamese network provides a slightly different approach from a regular classifier. This network does not classify an input to one of several classes. Instead, it takes two images as inputs and scores their similarity level. The network architecture is depicted in **Fig. 4**.

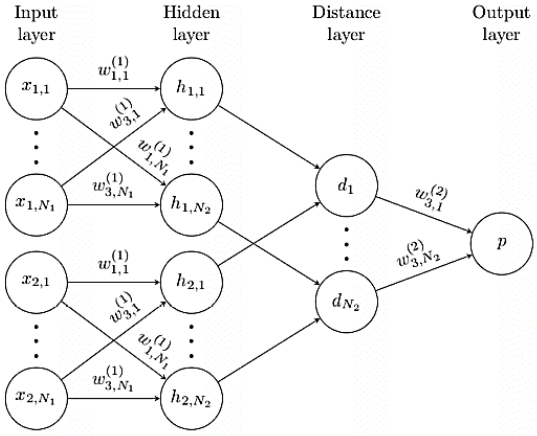



***Fig. 4.*** *Siamese* network classifier *[17]*

Siamese network is constructed of two identical Convolutional Neural Network (CNN) channels that fuse into one distance measurement layer. The distance measurement layer measures the Euclidean distance between the results of the two CNN channels and produces a scoring function that reflects the degree of similarity between the images entered into each of the Siamese network channels. The inputs to our Siamese network are two separate skeletal pose frames. The two frames either belong to the same pose class or contained different pose classes.

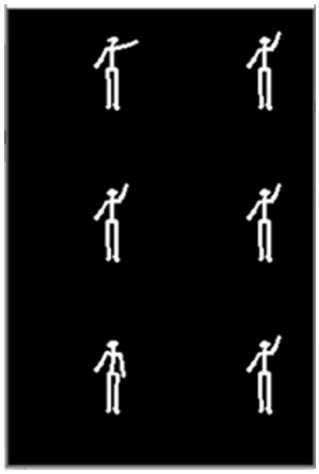
| Tuple of two images | Label | Remark |
|---|---|---|
|  | 0 | No pose similarity |
| | 1 | Maximum pose similarity |
| | 0 | No pose similarity |

***Fig. 5.*** *3-way* Inputs and classification of the Siamese network

The Siamese network output will be a floating-point number ranging between 0 and 1; one indicates that the two images have maximum pose similarity (the two images belong to the same pose class), and 0 indicates that they have no pose similarity (the two frames contain different poses). In our solution, we use a classical CNN network with some adaptations for our needs; with Siamese networks, each channel has a similar structure of regular convolutional and pooling layers. Instead of the regular SoftMax classification layer at the end of our optimized CNN module, we use a dense layer that produces a low dimension feature vector. Since the network has two images as inputs, we end up with two feature vectors (we name it the pose tagging mechanism). Then we calculate the Euclidean distance difference of these two layers and output the result to a single neuron with a sigmoid activation function (0 or 1). More precisely, we calculate the prediction vector as the formula of Eq. (2).

$$p = \sigma(\sum_{j} \alpha j \, |h^{j}_{1,l-1} - h^{j}_{2,l-1}|) \quad (2)$$

Where σ is the sigmoidal activation function. This final layer induces a metric on the learned feature space of the (L − 1)th hidden layer and scores the similarity between the two feature vectors. The training data for this Siamese network is structured such that each sample in the dataset contains a tuple of two images and their corresponding label: 0 for images with different poses, which means that they belong to different classes, and 1 for images that belong to the same class (see **Fig. 5**). The network architecture is further depicted in **Fig. 6**.

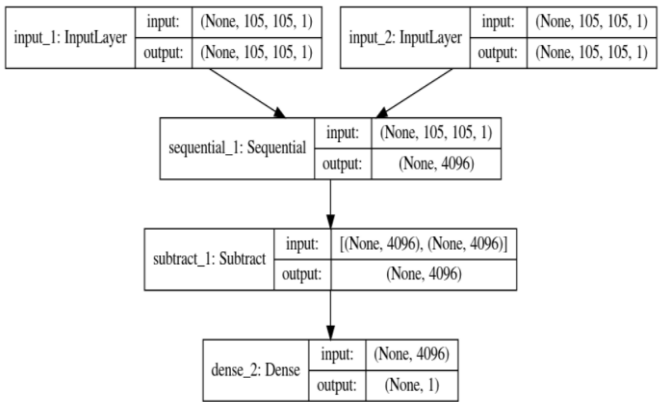



***Fig. 6.*** *Siamese* network architecture *[17]*

Note that, for every pair of input image, our model generates a similarity score between 0 and 1. However, just looking at the score, it is challenging to determine whether the model can understand similar images and distinguish dissimilar ones. An excellent way to assess the model is N-way one-shot learning. N-way is a method for testing the performance of the Siamese twin network. Take a limited number of pairs of pictures - N pairs. There is one pair of identical images in every N pairs of images, and the rest are different. Inject each pair into the Siamese twin's network and score the degree of variance between the pair of images. Whenever the lowest score is obtained on the pair that contains two identical images, we say that there was a correct classification. In any other case, we say there was an error. Repeat the experiment K times. Calculate the number of times out of the K attempts was correctly identified. As the accuracy percentages increase, we say that the Siamese twin network improves in performance.

## 4 Implementation

In the first stage, as seen in **Fig. 7**, we take a standard encoded 30 FPS video as input to the OpenPose. The result is a JSON file consisting of 2D joint values indicating the XY coordinates of the 25 joints in each video frame and a confidence level as a number between 0-1 for each joint. We also extract the Motion Vectors from the encoded video via FFmpeg software.

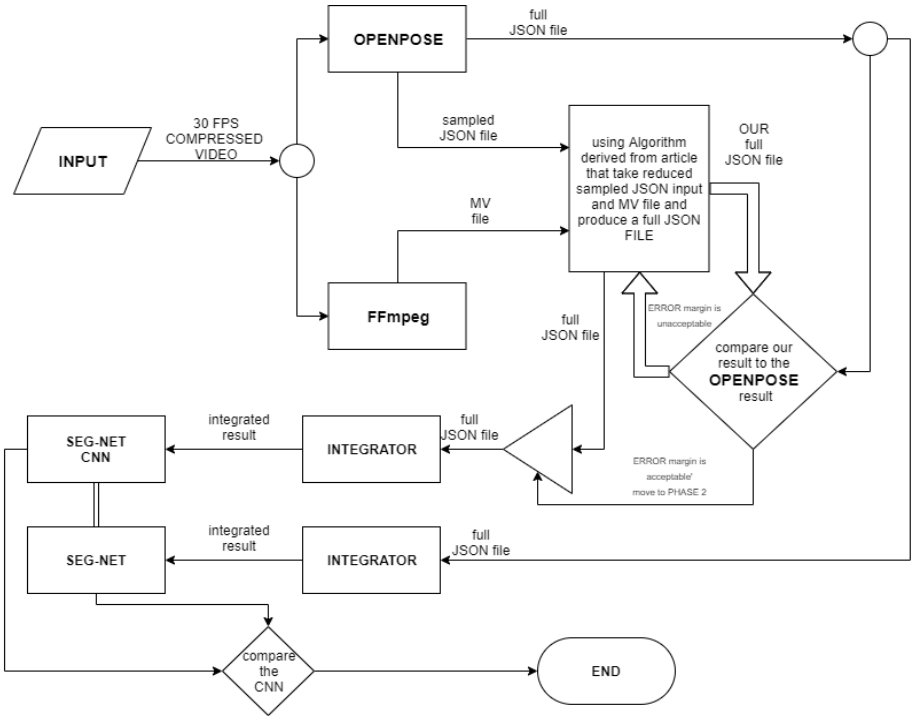


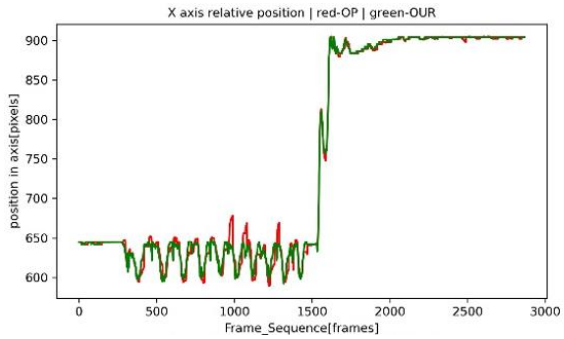

**Fig. 7.** Prediction *and measuring flow*

Afterward, using our tracking algorithm, we take the sampled JSON file (1fps) and the MV file and predict an entire JSON file for all the missing frames in the video - we produce 30fps skeletal from 1fps skeletal (We name it as OpenPose 30fps Skeletal). Finally, we compared the results to an OpenPose that uses GPU and can produce skeletal at 30fps.

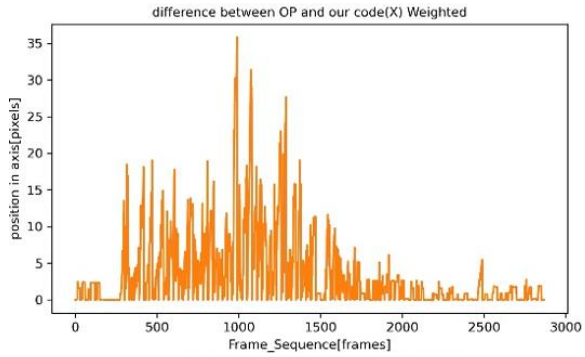
# 5 Results

## 5.1 Prediction vertexes via Motion Vector (MV) Tracking Algorithm

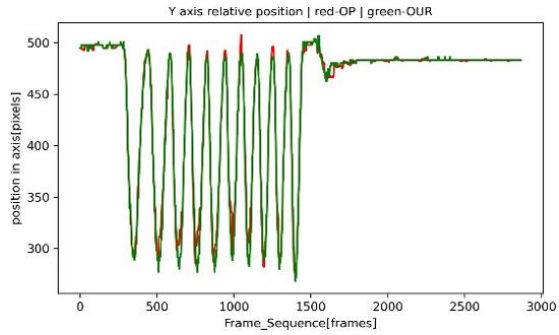
In our test we divided the input skeletal movement movie into Group of 30 Poses each (We name them GOP). The first frame of the GOP sequence is predicted with OpenPose, and all the other 29 frames are predicted with our MV tracking algorithm. The graphs in **Fig. 8** provides the comparison between coordinates of 2 different body vertexes as they are extracted with the OpenPose and with our own MV tracking method, respectively. (a) and (c) depict the coordinate value while (b) and (d) depict only the differences between the coordinates in consecutive frames.



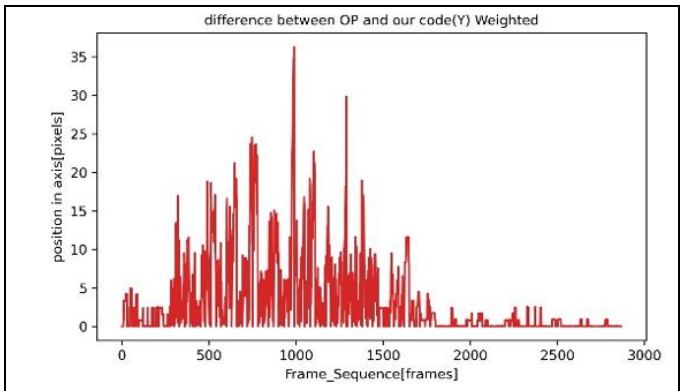

(a) Single vertex coordinate in consecutive frames




(b) Difference between single vertex coordinates




(c) Single vertex coordinate in consecutive frames




(d) Difference between single vertex coordinates

***Fig. 8.*** *Single* vertex tracking accuracy

### 5.2 Autoencoder Classification results

To achieve our goals, we attempted to develop a autoencoder system that determines whether a specific target movement is similar to a given reference movement. Our research contains three parts – studying the feasibility of latent vectors as a tagging mechanism, creating databases, and training a autoencoder neural network. To implement our system, we manipulated data extracted from OpenPose based on six different movements and used it to feed our network. Our findings are that latent vectors can contain enough data to fulfill our needs. Our measure of success is to achieve a decision accuracy of at least 85%.

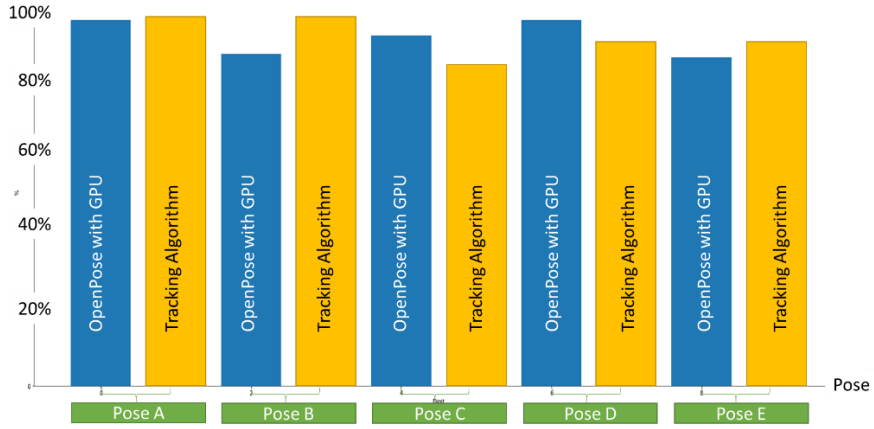



**Fig. 9.** Using an autoencoder network for comparison between OpenPose and a our tracking algorithm

### 5.3 Siamese network Classification results

We used the result of our tracking algorithm as input to the Siamese network. We tested the accuracy of the result using the N-Way method.

**Our experiment parameters:**

- We perform N-Way: validation of one-shot learning.
- We used N-Val: number of N-Way repetitions.
- N-Val = 100
- Perform N-Way validation every 100 epochs
- Number of epochs = 1000
- Success criterion: $if\ argmin(predictions[k]) = 0\ \rightarrow correct\ prediction$

$$Accuracy = 100 \cdot \frac{\#\ of\ correct\ predictions}{Nval}\ [\%] \qquad (3)$$

**Our experiment results**

Our experiment Siamese network Classification results is presented in **Fig. 10**

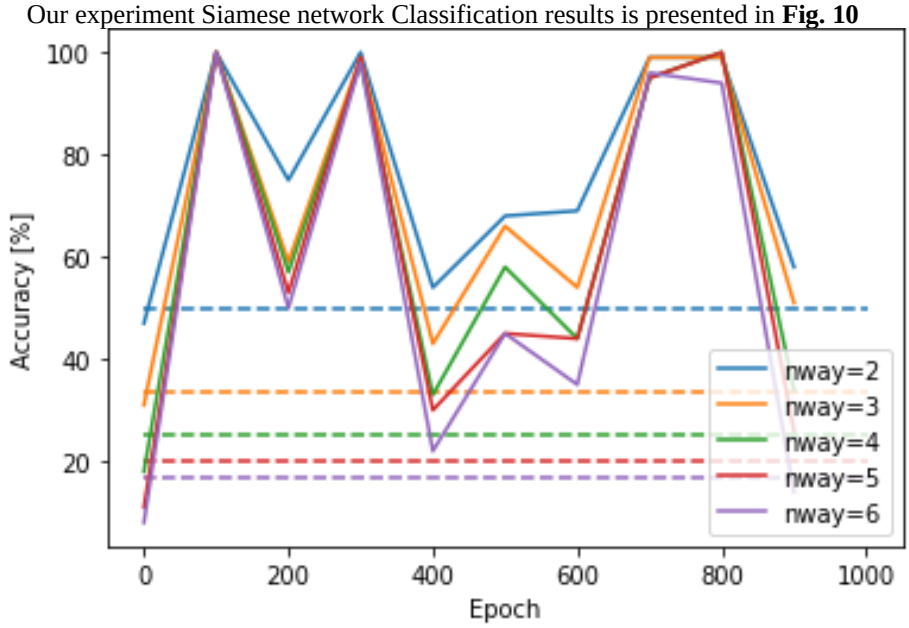



**Fig. 10.** N-Way accuracy vs Epoch

- 1 Siamese CNN trained for 1000 epochs
- 5 different n-ways evaluated (in parallel) every 100 epochs
- Dashed line is the accuracy of a random guessing system

Note that overall, n-way=6 is the lowest graph and n-way=2 is the highest graph.

### 5.4 Analysis of classification results

We can see that in both classification methods, we obtained high results. The experiment focused on comparing OpenPose input versus our tracking algorithm input to the autoencoder classifier network. It can be seen with certainty that the results are very similar, and we can assume that the minor differences in the accuracy are a result of inaccuracy in the network weight values. In the Siamese network, the experiment focused on using the tracking mechanism in favor of classification. We can see that the network achieved 100 percent accuracy in pose classifications. Moreover, it is clear that the larger the N in the N-WAY method then the greater the chance of error (the system tries to find one correct match from a wide range of available matches). Despite the difficulty, the system finds a combination of weights that produce 100 percent accuracy for each possible N. This result can be discerned after 100 epochs and after 300 epochs. The results of those experiments, which we performed over a wide range of pose datasets, prove the quality and stability of our tracking algorithm as a tool that supports OpenPose for PCs without GPU.

### 5.5 CPU & Memory performances utilization and optimization

We ran different scenarios of CPU / Memory utilization to optimize tracking algorithm performance. Since we track each body vertex independently from all other vertexes, it is possible to process them in parallel and thus accomplish further acceleration. OpenPose average frame rate when run on a very strong CPU (8-20 cores) can reach 0.1 – 0.5 fps. We use this as our baseline and all other results are compared to it. We managed to further increase its performance speed to 0.3 – 2 fps by reducing the accuracy of the net-resolution. The performance was improved by utilizing 3 different methods: (1) data loading schemes – dividing the MV database to multiple files and loading them per frame one at a time. This method is I/O bound and suffers from wasted time and CPU resources. Alternatively loading all MV files of all frames to memory and using the pickle library to quickly import python objects to memory; (2) MV search algorithm optimization – implementing an iterative tree search, whereas the search size is divided in each iteration according to the macro block size (16x16, 16x8, 8x8, 4x8, etc.); and (3) multi-processing – since the calculation of the vertexes can be done independently from one another, we use the multi-core processor for efficient parallel processing. A graph of the best preforming approach is provided in Fig. 1

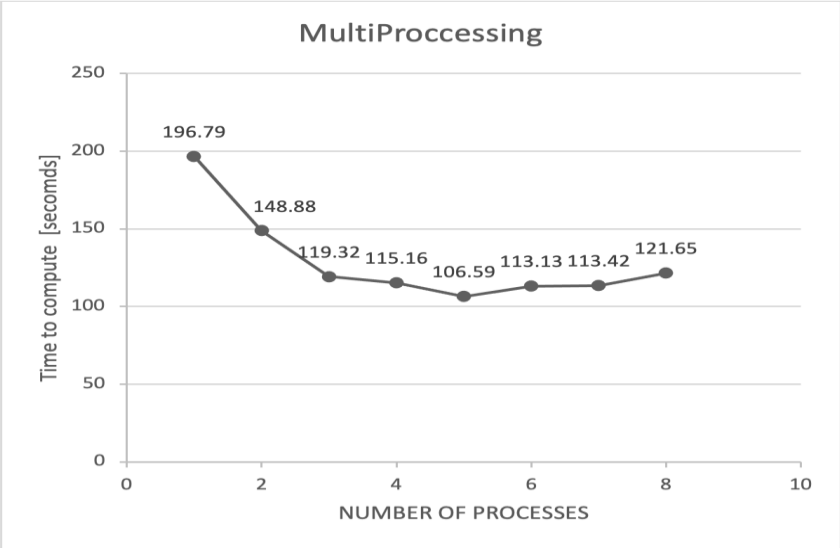



***Fig. 11.*** *Processing* time per no. of parallel process

The performances without GPU results are summarized in **Table 1**.

***Table 1.*** *Processing* performances without GPU

| *No* | *Activity* | *Processing 3-minute movie [Sec.]* | *Improvement factor* |
|---|---|---|---|
| 1 | Run OpenPose with regular CPU (without GPU) (reduced net-resolution) | 1,600 | 1 |
| 2 | Straightforward process each vertex separately | 1,200 | 1.33 |
| 3 | Load all frame MV to memory | 420 | 3.8 |
| 4 | Improve the search part in the tracking algorithm | 236 | 6.8 |
| 5 | Multithreading with shared memory | 378 | 4.23 |
| 6 | Multi-processes with shared memory | 286 | 5.6 |
| **7** | **Multi-processes without shared memory** | **106** | **15** |

## 6 Conclusion

This work succeeded in providing sufficient OpenPose real-time processing without the GPU dependency. Our novel solution allows using OpenPose software at a meager frame rate with only a CPU setup. This algorithm fills the missing gaps with human skeletal vertex locations via Motion Vectors (MVs) extracted from the encoded video. We compare the accuracy results of our OpenPose Plug-In algorithm with regular OpenPose based GPU and show excellent matching results. We proposed a new accuracy assessment formula (Waited Mean Square Error) that considers the OpenPose confidence level. We verified the accuracy and quality of our results using two orthogonal methods. One is measuring processing speed, and the other is measuring accuracy and error on the performance of neural networks. To check the consistent pose classification of our method, we used the Siamese Neural network classifier to measure the influence of our frames input compared to the regular OpenPose frame input. The results of those experiments, which we performed over a wide range of pose datasets, prove the quality and stability of our tracking algorithm as a tool that supports OpenPose for PCs to be used without a GPU. Overall, we show an improvement of x15 in the time it takes to process movements between our algorithm and the native OpenPose with excellent accuracy metrics.

## 7 References


[1] Z. Cao, G. Hidalgo, T. Simon, S. Wei and Y. Sheikh, "OpenPose: realtime multi-person 2D pose estimation using Part Affnity Fields," *arXiv preprint arXiv:1812.08008,* 2018.

[2] T. Simon, H. Joo, I. Matthews and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," *InProceedings of the IEEE conference on Computer Vision and Pattern Recognition,* pp. 1145-1153, 2017.

[3] S. Wei, V. Ramakrishna, T. Kanade and Y. Sheikh, "Convolutional pose machines," *InProceedings of the IEEE conference on Computer Vision and Pattern Recognition,* pp. 4724-4732, 2016.

[4] T. Ujiie, M. Hiromoto and T. Sato, "Interpolation-based Object Detection Using Motion Vectors for Embedded Real-time Tracking Systems," 18 june 2018.

[5] Y. Bai, H. Hu, Y. Li, C. Zhao, L. Luo and R. Wang, "Research methods for human activity space based on vicon motion capture system.," *2017 5th International Conference on Enterprise Systems (ES). IEEE,* Sep 2017.

[6] K. H. Cheung, "Optitrack - Estimation of Opti-track motion capture system data," Department of Electrical Engineering,, 2020.

[7] "Qualisys | motion capture systems," Mar 2020. [Online]. Available: https://bit.ly/YS-Qualisys.

[8] "Motion capture system | bts bioengineering," Smart-dx, Mar 2020. [Online]. Available: https://www.btsbioengineering.com/.

[9] Y. Chen, C. Shen, X. Wei, L. Liu and J. Yang, "Adversarial posenet: A structureaware convolutional network for human pose estimation.," *2017 IEEE International Conference on Computer Vision (ICCV),* pp. 1212-1221, Oct 2017.

[10] "Wrnch - teaching cameras to read human body language," Mar 2020. [Online]. Available: https://wrnch.ai/.

[11] Z. Zhang, "Microsoft kinect sensor and its effect.," *IEEE MultiMedia,* pp. 19(2):4-10, Feb 2012.

[12] "Home - xsens 3d motion tracking," Mar 2020. [Online]. Available: www.xsens.com.

[13] "Rokoko - motion capture system - smartsuit pro," Mar 2020. [Online]. Available: https://www.rokoko.com/en/.

[14] "Shadow motion capture system," Mar 2020. [Online]. Available: https://www.motionshadow.com/.

[15] A. Ahmadi, F. Destelle, D. Monaghan and N. O'Connor, "A framework for comprehensive analysis of a swing in sports using low-cost inertial sensors.," *SENSORS, 2014 IEEE,* pp. 2211-2214, Nov 2014.

[16] "wikipedia Inter_frame," [Online]. Available: https://en.wikipedia.org/wiki/Inter_frame.

[17] G. Koch, "Siamese Neural Networks for One-Shot Image Recognition," Graduate Department of Computer Science University of Toronto, Toronto, 2015.

# Fraud and Adversarial Machine Learning*

Medina Williams[0000−0002−3494−235X]

[1] Purdue University, West Lafayette, IN 47907, USA
[2] will2345@purdue.edu

**Abstract.** The ongoing COVID-19 pandemic is a natural experiment providing a way to assess the resiliency of trusted systems within the United States. This system shock caused a material shift towards newer mediums permitting for the adoption of new ways of working and living: artificial intelligence (machine learning), automation, and a heavy reliance upon social media. Hackers, fraudsters, and cybercriminals find and exploit vulnerabilities in trusted systems. It remains difficult to estimate all the effects of these new socio-technical capabilities (social media and artificial intelligence or machine learning) and their impact upon society's trusted systems (within the realms of cybersecurity). This study considers the problem of lingering effects from an attacked system of trust and proposes a new framework of prevention for practitioners and extends previous research.

**Keywords:** Machine Learning · Fraud · Adversarial Attack.

## 1 Introduction

Societies are constructed from systems of perceived trust within both tangible and intangible environments. When humans experience this trust through more than one sense, it only strengthens their resolve in the resiliency of these systems. Technological advancements through change improve performance of these systems and provide additional features or conveniences to both organizations and businesses through large-scale adoption. Over the last two decades, two technological phenomena – social media and cybersecurity – have become forcing functions for changing the way humans interact with the metaphysical and physiological worlds.

The ongoing COVID-19 pandemic is a natural experiment providing a way to assess the resiliency of our trusted systems. Individuals, businesses, and organizations were forced into new paradigms of existence and ways of work at an unprecedented pace due to this global health emergency. This caused material shifts in experiences and interactions leading to a heavy reliance on social media and cybersecurity mediums to ease into these new ways of working and living.

Digital transformation is now common business nomenclature as companies used the required shift in operational postures as an opportunity to innovate and accelerate change. Some of this change included a pivot towards newfound

* This study is being considered for grant funding by the ACFE.

technological capabilities including the integration of artificial intelligence, automation, and machine learning within everyday business operations. Economists typically stylize these adjustments as technological change. The resulting market effects include macro and micro shifts as businesses and workers become more productive. En masse, this shift includes trade-offs such as losses of lower skilled jobs and the length of time the economy takes to recover or repurpose those jobs elsewhere. There are many more trade-offs and adjustments as well.

Social media and cybersecurity capabilities accelerate the speed at which transactions and interactions can occur since events are no longer bounded by physical and spatial constraints. The prevalence of social media has created new sources of data for organizations and businesses. It has also allowed individuals to express themselves and their individualities in an open cyberspace forum with some semblance of trust baked into these systems. Yet, it is the reliance upon this facade and not the actual trust itself that determines individual's actions. Companies seek ways to monetize these information sources by combining them with new and existing technological advancements as previously mentioned above. While these advances have generally enabled progressive spillover effects, they in turn create many potential vulnerabilities as well.

For all the gains realized as these new technologies embed themselves further and become commonplace, there are additional trade-offs, vulnerabilities, and unintended consequences that find their way to the surface. Within these new paradigms, the terms hacker, fraudster, and cybercriminal all seem to converge into the same meaning – someone who finds and exploits a vulnerability in our trusted systems. It remains difficult to estimate all the effects of these new socio-technical capabilities (social media and artificial intelligence or machine learning) and their impact upon society's trusted systems (within the realms of cybersecurity).

When a vulnerability is found, typical operational protocols involve swift remediation and a savvy public relations campaign, the latter element being optional and dependent upon the magnitude of the organization involved. However, all with the goal of swiftly returning to business as usual with a few new controls and typically minor tweaks. But what if the dimensionality of this fix is misapplied to the problem at hand because of secondary and tertiary effects that linger just below the surface? More specifically, what if the frame of reference for remediation and proactive detection of attacks on the socio-technical interactions mentioned above – where cybersecurity, fraud, and machine learning intersect – is wrong?

This study brings all the facets previously mentioned and considers the problem of lingering effects from an attacked system of trust. A situation such as this leads into lax preparation and consideration of potential secondary vulnerabilities to those same systems. As such, the problem set from within this study uses a natural case study unfolding during the COVID-19 pandemic as a guide to amplify the problem and the urgency of much needed attention upon this matter.

# Location Functions for Self-Stabilizing Byzantine Tolerant Swarms

Yotam Ashkenazi[1], Shlomi Dolev[1], Sayaka Kamei[2], Fukuhito Ooshita[3], Koichi Wada[4], and Yoshiaki Katayama[5]

[1]Department of Computer Science, Ben-Gurion University of the Negev, Israel
[2]Department of Information Engineering, Graduate School of Engineering, Hiroshima University, Japan
[3]Graduate School of Science and Technology, Nara Institute of Science and Technology, Japan
[4]Department of Applied Informatics, Faculty of Science and Engineering, Hosei University, Japan
[5]Department of Computer Science, Nagoya Institute of Technology, Aichi, Japan

July 12, 2021

**Abstract**

Error Correcting schemes are employed to achieve self-stabilizing Byzantine tolerant swarms. The members of a swarm of $n$ robots out of which, eventually, at most $t < n/2$ are Byzantine, search for the location of $n-t$ robots that respect a predefined function (family, such a polynomial of $n-2t-1$ degree) over their locations. When the members identify such a respected function they use the function as the current state to move to respect the next function, preserving the (closure) $n-t$ function definition invariant. In case no single definite function can be defined by $n-t$ robots location, a default function is chosen (convergence), and the (eventually stabilizing at least $n-t$) correct robots move to the default function location, from which they change state as defined by the task.

# 1 Introduction

Coping with Byzantine and transient faults in a robot swarm is a significant important challenge of many-particle implementations. Traditionally, a swarm is designed to cope with a certain upper bound on the number of swarm participants that experience a fault or takeover (by an adversary that controls their actions). However, there are cases where a global fault, such as a momentary lighting, effects all members in the swarm. In such situations, automatic recovery in the form of self-stabilization in the presence of a minority of Byzantine robots is very important.

A novel location policy is presented here based on functions over the robots' location. The usage of the proposed policy enables the robots in the swarm to identify the Byzantine participants and to stabilize in spite of the Byzantine robots' actions. Namely, the policy presented allows the swarm to perform a task while ignoring the Byzantine robots' actions. The robots use a policy for their location choice, the policy that restricts their location to predefined functions, using error-correcting capabilities of polynomials (in case of unlimited location range) or shape structure function, such as a circle or ellipse (in case of a limited area bounded shape).

Note that polynomials over two or more dimensions can be defined by interpolation over the coordinates of the robots, and use efficient error correcting to verify that the degree of the obtained polynomial is not too large. Still, in some cases polynomial cannot describe shapes that share the same coordinates and are different only in the result. For example, consider a circle in two dimensions, where there are two points sharing the same $x$ coordinate. Another drawback in using polynomials for checking validity of robot positions and identify (and ignore) outliers/Byzantine participants is the missing restriction of proximity, as any coordinates are mapped to a resulting value. Even when the polynomial is defined over a finite field, coordinates that are far from each other may co-exists. Thus, we also introduce shape equations policies, where the location of the non-Byzantine robots obey the shape equation, whether the shape is a circle, an ellipse and alike. Each such shape can

be defined by a relatively a small set of points (in two or more dimensions). We use the majority of such (sub)sets of the robot locations to decide on outliers/.Byzantine participants that do not obey the majority agreement on the shape.

Often, a swarm of robots must perform a task with a specific structure defined in a two or three dimensions space. For example, a line of marching robots may clean a field, drones trying to reduce the air resistance by creating a three dimensional aerodynamic shape, possibly mimicking a swarm of fish moving within a higher-level dimension polynomial structure.

As we already stated, when dealing with robots in practice, we have to consider the presence of Byzantine, faulty, or malicious robots. These robots may not follow a given algorithm, possibly, due to experiencing a fault or due to an intentional malicious takeover that may intentionally interrupts/disturbs other robots. For example, a behavior caused by a bug in the software/control of honest robots or malicious malware injection. These possibly Byzantine robots may be temporarily or constantly controlled by an adversary. In particular, the number of such Byzantine robots may exceed (at least temporally) any given threshold up to all robots being faulty. Still, we would like the robots to execute their task once enough of them are correct(ed).

We describe novel location policies for the robots to perform a task and ignore the Byzantine robots. In these policies, robots move so that locations of non-Byzantine robots always satisfy a polynomial (for the case of non restricted range) or shape structure (for a limited area bounded shape). The number of Byzantine robots that does not prevent the robot swarm to achieve its goal is limited, obviously, when all robots are Byzantine no meaningful task can be achieved. Our policy is based on choosing the degree of the polynomials or the needed number of points for defining a function of a shape structure carefully to be small enough with relation to the number of robots, so that the (Byzantine) robot minority that does not obey the polynomial or shape can be identified and discarded. The robots move[1] according to predefined transition functions over their location. Keeping

[1] To simplify discussion we assume all robots move at once, or there is a synchronous clock that allows

an invariant defined by the function on the locations of the correct robots, namely the maximal degree of a polynomial being at most $n - 2t - 1$ ignoring at most $t$ robots (as we later detail an efficient error correcting technique exists for checking the existence of such a polynomial) is the heart of our polynomials based solution. In case of a shape, the invariant that the non-Byzantine robots preserve is a location choice that obey the shape, and the transition from one shape to the next keeps the invariant. In both cased, the invariant ensures that the Byzantine robots that do not respect the policy can be identified and ignored.

The chosen policy is useful in keeping the error correction-capability (and Byzantine identification) using a predefined degree for the polynomial or a (correctable, up to a threshold of non-respecting robots) function for the shape structure based on the number of robots and the maximum number of Byzantine robots. Thus, the policies allow the detection up to $t$ Byzantine participants.

In the case of a polynomial, our algorithm can be based on the efficient Berlekamp-Welch algorithm [8] (referred in the sequel as the BW algorithm) or the extension of the BW algorithm to three or more dimensions [2], to find at least $n-t$ robots that are located on points of a finite field polynomial of degree of at most $n-2t-1$, where $n$ is the number of robots. If the validation fails, a reset/regain-consistency procedure starts. One trivial possibility to execute a reset is to move to a default low degree polynomial (e.g., a line) and once the polynomial defined by at least $n - t$ participants, is of a small degree, then the identification of the Byzantine participants is enabled, allowing to continue to the next polynomial according to the task definition. The BW algorithm ensures that when the polynomial $P$ is of a degree that is at most $d = n - 2t - 1$, and at least $n - t$ honest robots stand on points of $P$ then the (non-Byzantine) robots find the polynomial $P$ with an efficient calculation. We note that the swarm task may restrict the polynomial choice beyond the maximal degree, and the additional restrictions should be applied and checked

enough smaller granularity steps to reach the positions in the next polynomial/function, prior to reexamining the next location.

too. For example, the distance of at least $n - t$ (non-Byzantine, obeying the polynomial) robots from each other is at most a constant $D$.

In the case of a shape defined by a given function $F$, our algorithm searches for $n - t$ robots that agree on a specific $F$, in the circle examples, specific center and specific radius. If the robots cannot find an agreed function over their location, i.e., do not find a valid shape, the robots move to a predefined shape and function, in the circle example, say, a circle with center $(0, 0)$ and predefined radius $k$. Namely, the robots try to validate whether the function of the shape structure is respected by at least $n - t$ robots.

The ability to define a single global function over the locations of the non-Byzantine robots can be exploit to serve as a (global memory of a) global state. Namely, the function maintained by the correct members of the swarm can be mapped to a number in a finite field, namely, the free coefficient of the polynomial. Analogously, for the case of a shape, The mapping can be the value of the shape defined by the non-Byzantine participants such as the circle/shape center.

We demonstrate that a scheme based on function abstraction over the robot locations together with fitting error correcting schemes (to identify and ignore Byzantine participants) is useful in implementing (series of) tasks. In particular, we demonstrate implementations of gathering, marching and exploration.

## 2 Problem Setting

Consider an $N \times N$ ($N \geq n$) board where $n$ robots can move on the board. The tiles are referred to by coordinates $(x, y)$ ($1 \leq x, y \leq N$). We define *up*, *down*, *left*, and *right* directions as directions from $(x, y)$ to $(x, y + 1)$, from $(x, y)$ to $(x, y - 1)$, from $(x, y)$ to $(x - 1, y)$, and from $(x, y)$ to $(x + 1, y)$, respectively.

Among the $n$ robots, at most $t$ robots are Byzantine, that is, they do not respect the rules and restrictions, and may move to an occupied location. We assume that robots that share the same (low granularity rough) coordinates can still be identified and map to

locations in the next *map*.

The robots can perform the tasks according to the following two different function structures: a polynomial structure and a shape structure. We assume the fully-synchronous (FSYNC) model [7, 5, 4]. That is, all robots repeat the following global Look-Compute-Move phases synchronously. In particular, there is a global pulse that triggers all robots to observe the location of all other robots, to compute where to move and to move to the desired location. We do not restrict the movement to be local, namely a robot can move to a far location prior to the next pulse. Note that multiple coordinates move can be supported by the use of global (Byzantine clock) synchronization. Such synchronization may allow robots to make several moves between any two consecutive synchronizing global clock pulses.

One can view the move between two consecutive states as a combination of sequence of low granularity moves *up*, *down*, *right* and *left*, that are not noticed, as steps are of higher granularity, a move to any location at once.

A *map* is defined as a static image of the board where each robot stand on 1 tile.

A *directed graph* is defined as a graph $G = (V, E)$, where $V$ is a set of maps and $E \subseteq V \times V$ is a set of directed links. Any two adjacent *maps*, $map_i$ $map_j$ connected by an arrow (transition) from $map_i$ to $map_j$ $\{map_i, map_j\} \in E$ iff at least $n - t$ robots obeying their transition instruction in $map_i$ and the rest located anywhere. There is an edge from $map_i$ to a next map in the graph for every such choice of at least $n - t$ correct moves and for every arbitrary possible locations chosen for the rest.

An *exhaustive task definition* is a directed graph (automaton) of distinct *maps*, such that, each *map*, is the set of coordinates of all robots (associated with a given time instance), and instruction for each robot where to move next.

As there is a (bounded) number of Byzantine robots, each *map* continuation can differ depending on the Byzantine robots ability to move anywhere. Note that Byzantine robots may stop being Byzantine and vise versa.

It is convenient to define a task in more abstract fashion, in particular in terms of

polynomials and functions.

A *polynomial based task* is a directed graph (automaton) of distinct *maps*, such that the locations of at least $n - t$ robots in each of the *maps* in the graph encode a polynomial of degree $n - 2t - 1$ or less, and any two adjacent *maps*, $map_i$ $map_j$ connected by an arrow (transition) from $map_i$ to $map_j$ identify which robot in $map_i$ moved to a location $map_j$. Just as in the exhaustive task definition, there is an edge from $map_i$ to a next map in the graph for every choice of at least $n - t$ correct moves and for every arbitrary possible locations chosen for the rest.

A *function based task* is a directed graph (automaton) of distinct *maps*, such that the locations of at least $n-t$ robots in each of the *maps* in the graph encode the same function[2], and any two adjacent *maps*, $map_i$ $map_j$ connected by an arrow (transition) from $map_i$ to $map_j$ identify which robot in $map_i$ moved to a location $map_j$. Just as in the exhaustive task definition, there is an edge from $map_i$ to a next map in the graph for every choice of at least $n - t$ correct moves and for every arbitrary possible locations chosen for the rest.

The robots performing the task infinitely often, perform a global transition from the current *map* that represents a swarm state to the next state in each round.

There is a need to map each robot location over the next polynomial or shape avoiding collisions among non-Byzantine robots and preservation of enough points for keeping the invariant, mitigating the risk of too many non-Byzantine robots choosing the same location on the next map.

The mapping of the robots over the next polynomial or shape can also be based on an abstract policy rather than exhaustive (and memory expensive) mapping. For example, a lexicographic order in the current map (order according to $x$ then according to $y$) defines the location of each robot, possibly also lexicographical from a first pre-agreed (say the point with $x = 1$) point on the polynomial. Thus, there exists an abstract fashion to efficiently define a task, one that is much (typically, exponentially) better than the exhaustive task

[2]Including common missing values/parameters, such as the coordinates of the center of a circle and the radius

definition.

In case the task includes shapes, the task can be defined by predefined shapes in many ways. We can define the task as stationary function forming a predefined shape. The task may further define the transition from an existing shape to the next shape and next location of the shape.

**Task requirements** are respected when the locations of the robots form a (distinct) *map* of the task, and in every transition at least $n-t$ robots obey the transition movements definition of the task from the current *map*.

Note that in the case of gathering the current *map* and the next *map* may only differ in the location of the (identified) Byzantine participants.

It is convenient to equip the robots by a procedure that identify the current locations of the robots as being a (distinct) *map* of the task, rather than searching for such a *map* when the task is exhaustively defined. In both the polynomial task definition and function task definitions, such a convenient procedure exits.

## 3 Function Based Tasks

The validation procedure for the case of function $F$ that defines a shape, examines whether the location of at least $n-t$ robots reside on (finite field integer) points that obey the function $F$. Whenever the validation is not successful (as the robots use global view and use the same validation procedure, the success or failure of the validation is identical in all correct robots), the robots create a default valid shape ignoring the Byzantine robots.

**Circle Example.** To make our discussion more concrete, we consider the case of a circle as an example, the validation of the circle can be based on searching for the locations of at least $n-t$ robots that agree on the center and radius of the circle. Such a search can be based on computing the center and radius of subsets of (non co-linear) three robots and checking the center and radius on the rest of the robots, or checking the most popular center and radius of subsets and in case no ties exists, using default center and radius.

The robots calculate the circle function, note that any three robots that are on distinct tiles/coordinates and are not on a line can define one circle. Assuming a valid circle for which the location of $n - t$ robots respect does exist, then the common center and radius of the circle they reside on is found during the examination of the polynomial number of such three robots subsets. Note that $t$ is chosen in a way that ensures the existence of at most one possible circle, in particular, as we prove in the sequel, when $n > 2t + 2$ and $n - t$ robots share a common circle center and radius then no other set of $n - t$ robots can share a circle with different center and/or radius.

The arguments for the uniqueness of the circle are based on examining the worst case scenario. Consider the case in which all Byzantine robots stand on a common circle different from the circle the non-Byzantine robots stand on, trying to manipulate the honest robots to choose the wrong circle. The Byzantine robots in this case can choose 2 honest robots and create 1 circle with them so the number of robots on the wrong circle is $t + 2$. As the number of robots on the correct circle is equal to $n - t$ by definition, we have to choose $n - t > t + 2$ to avoid the malicious strategy of the Byzantine robots, namely to choose $n > 2t + 2$. In general, assuming the number of points needed to define a function $F$ is $x$, the $t$ Byzantine robots can choose $x - 1$ honest robots to define a new shape. In this case we want that $n - t > t + x - 1$ so $n > 2t + x - 1$.

The number of all possible groups of 3 robots out of $n$ robots is $\binom{n}{3}$, which is equal to $n(n-1)(n-2)/6$ implying polynomial time complexity. We can then check that $n-t$ robots reside on the found circle parameters, hence, as long as $n > 2t + 2$, at most one circle can be defined in a *map*. If a valid circle is found the next *map* is used for movement. Note that the shape task may further restrict the possible combinations of radius and centers possibly implying that no *map* of the task is found. If the robots cannot find a valid circle, or the further restrictions do not hold then the robots may have a common convergence-function that locate a map closest to the circle they reside on if such a circle exists, or use a default circle, and move to this circle prior to continuing with the task. A default circle can be for example $x^2 + y^2 = r_0^2$, where $r_0$ is a predefined global value.

**Movements.** After creating the shape, robots can move/march according to the map transitions, for example in a smooth global movement that does not change the shape structure moving *up*, *down*, *right* and *left*. Such a movement can be defined globally by a transition abstract function, in particular, up can be defined by adding 1 to the $y$ coordinate of the center. In addition the robots can increase or decrease the size of the shape depending on the task. The Byzantine robots can join the (next) shape structure instantly, and stop being considered Byzantine, robots may also decide not to join the (next) shape and then are identified as Byzantine.

## 3.1 Shape Function Based Algorithm

An *execution* is defined as a move between two maps, $\text{map}_i$ to $\text{map}_j$ on a directed graph.

In case of general function based algorithm we define a function task execution as follows.
*Execution.* An execution is a finite or infinite sequence of *maps*.
*Function task valid execution.* Any execution that respect the *map* transition definition is a *function task valid execution*, or in short *valid execution*.

In particular cases, the maps and map transitions can be expressed abstractly by functions, where in the execution $(i)$ the shape structure holds, and $(ii)$ further possible restricted on the shape function parameters hold, such as, the radius is always greater than zero, and $(iii)$ further robot location restrictions over the shape, such as, the robot density on any given segment of length 5 is not more than $t + 1$, and $(iv)$ in case of shapes with edges, the minimal number of robot on each edge should be restricted.

Unlike polynomials, shape based algorithm restricts the locality of the shape, while polynomials can be defined over an infinite two dimensions or three dimensions space. Furthermore, in some shapes, two points with the same $x$ coordinate are valid, while such a definition is not possible in the case of a polynomials.

In case the shape is defined by several edges, such as a square, the minimal number of honest robots on each edge should be $t + 1$. The $t + 1$ lower bound ensures that the Byzantine robots cannot create an edge without the honest robots, so at least 1 honest

robot should located on each edge.

Assuming the shape required minimal $x$ points to define the shape function. The robots consider all possible subsets of size $x$ of robots. For each such subset, the robots calculate the shape function. Assuming there are $n-t$ residing a desired shape. The complexity to find a shape if exist equal to $\binom{n}{x}$, which is equal to $n(n-1)(n-2)*...*(n-x+1)/x!$. As $x$ is constant, the complexity to find a shape is bounded by $O(n^x)$.

If the robots cannot find a valid shape, all the honest robots move to a predefined shape. Moving to the pre-defined shape can be too drastic, therefore the predefined correct shape can be a function of the current globally known locations in a deterministic way.

Once the shape is created, robots can move without changing the shape structure according to the task definition. The task may cause the shape to move to other coordinates and/or change radius/scale.

The solution is self-stabilizing as the (previously) Byzantine robots compute the same next *map* as the non-Byzantine robots, and therefore can immediately join the shape structure. Once the previously Byzantine robots joined the shape they are not regarded as Byzantine as Byzantine by the rest of the robots.

**Algorithm 1:** Finding a shape algorithm

```
Each robot looks on all other robots location on the board
Each robot split the other robots including itself into all the options of groups of r robots, ordered by a
  common criteria where r is defined as the minimal number of points to define the shape.
For each group, each robot calculates the shape equation or structure, considering the ordered groups of r
  robots (x_1, y_1), (x_2, y_2), ..., (x_r, y_r)
Each robot checks whether there are n − t participants that respect the found shape
if Shape is found then
  | Validates the restrictions defined and stops, and move according to the map in the shape task.
end
else
  | {* when robots failed to find a shape *}
  | Robots move to a default shape locating themselves according to their order, where first considering
  |   the value of x and then the value of y to break symmetry (recall that there is also an order in a set
  |   of robot that share the same coordinates) in the current map.
end
```

The circle case can be generalized to any 2D shape function.

## 4 Polynomial Based Tasks

In case the task is defined by polynomials, we choose to use polynomials to detect (and ignore) erroneous locations, to do so the non-Byzantine participants identify a polynomial of degree $n-2t-1$, where $n$ is the total number of robots, and $t$ is the upper bound on the number of Byzantine robots. For example, if $n$ is 3 and $t$ is 1 the degree of the polynomial is 0, thus, the polynomial is a fixed $y$ polynomial, that can definitely identify one Byzantine participant, once the two correct robots share a specific $y$ they can keep sharing $y$'s forever ignoring the Byzantine participant.

We are interested in self-stabilizing solution, where the non-Byzantine participants may be started in any position. For example, the swarm can be started in a *map* where the location of each robot resides in a point with distinct $y$, and therefore a polynomial of degree 0 cannot be defined by the location of two or the three robots to obey the previous example. Consecutively, robots can identify that there is no polynomial with at most $n-2t-1$ degree, for which the location of $n-t$ robots agree. In this case, there is a need to change locations ensuring the invariant that there are at least $n-t$ robots over a polynomial of degree that is at most $n-2t-1$.

There are several ways to enforce the creation of polynomial with low enough degree, for example, all robots compute the polynomial $P$ of all locations, which may be a polynomial of degree $n-1$, and move to a location obeying a polynomial $Q$ with only the last $n-2t-1$ monomials of $P$. At most $t$ may not move accordingly, but still $n-t$ will obey the polynomial $Q$, establishing the polynomial degree requirement and the needed invariant.

Another way is to define a default polynomial as a pre-defined polynomial. If there is no polynomial with $n-2t-1$ degree, the robots move to the pre-defined polynomial with degree $n-2t-1$, for example, if $n=8$ and $t=2$, the robots can move to the pre-defined polynomial $f(x) = x^3 + x^2 + x$.

## 4.1 Polynomial Based Algorithm

*Polynomial degree requirement.* There is a polynomial $Q$ of degree $n-2t-1$, such that the locations of at least $n-t$ robots are residing on the points of $Q$.

*Degree convergence.* Starting in an arbitrary configuration, every execution with at most $t$ (fixed or non-fixed) Byzantine participants, has a suffix in which in every configuration the polynomial degree requirement holds.

*Polynomial task valid execution.* Any execution that respect the map transition definition is a valid execution.

In particular cases, the maps and map transitions can be expressed abstractly by polynomials, where in the execution $(i)$ the degree convergence holds, and $(ii)$ further possible restricted polynomial parameters hold, such as, the free coefficient is never negative holds, and $(iii)$ further robot location restrictions over the polynomial, such as, all robots are within at most predefined bounded distance holds too.

We consider two stages *Stage 1* and *Stage 2*. Initially, robots execute *Stage 1* and if the robots cannot solve the linear system as the required polynomial does not exist, they continue to *Stage 2*.

In *Stage 1*, the robots use the BW algorithm to check whether a polynomial with degree less than or equal to $n-2t-1$ exists. Let $P(x)$ be the polynomial of degree less than or equal to $n-2t-1$ where $P(a_i) = b_i$ for each non-Byzantine robot. Let $E(x)$ be the error polynomial that returns 0 for the Byzantine robots, or in other words $E(a_i) = 0$ for each $i$ s.t., $P(a_i) \neq b_i$. The BW algorithm defines another polynomial $Q(x)$ s.t., $Q(x) = P(x) \times E(x)$. The BW algorithm solves the linear system $Q(a_i) = b_i \times E(a_i)$ for each $i$. BW algorithm proved that $P(x) = Q(x)/E(x)$ can be found in polynomial time. By solving the linear system the robots can find the polynomial $P(x)$ and verify that indeed

$n - t$ robots reside on $P(x)$.

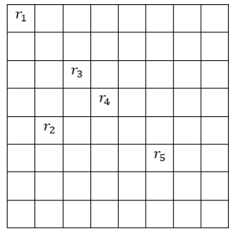



Figure 1: Using the BW algorithm with the $(0, 7), (1, 3), (2, 5), (3, 4), (5, 2)$ points, the robots calculate the $P(x) = 7 - x$ polynomial

As BW result is correct iff the number of erroneous points is no more than $t$, while in the self-stabilization scope no threshold $t$ can be assumed to hold in a given map, thus, the non-Byzantine robots can fail solving the linear system or solve it with (a common) wrong result, and need to verify that indeed $n - t$ robots reside on $P(x)$.

Then, when the polynomial task consists of all polynomial of degree less than or equal to $n - 2t - 1$, with no additional (coefficients and/or location over the polynomial) restrictions (and the map to the next location of each robot is defined by an abstract function, e.g., move up) then there is no need to search for the closest map in the polynomial task, and the robots continue with the tasks. Otherwise, the map in the polynomial task is searched, preferably by maintaining the index of the current map and using the arrows from the current map. If the polynomial with the required degree does not exist, the robots can identify this fact checking whether the BW linear system is solvable, and the result of the BW outputs can be verified by the locations of $n - t$ robots, otherwise, continue to *Stage 2*.

In *Stage 2* the robots move to a default (reset) polynomial and order themselves on the polynomial according to, say, a total order in the current map, where $y$ values break symmetry among points with identical $x$ values, and robots with the same coordinates are assumed to be ordered (in higher granularity around the point).

**Algorithm 2:** Finding a polynomial algorithm

Each robot looks on all other robots location on the board

Robots define two polynomials, $E(x)$ with degree $t$ and $Q(x)$ with degree $(n-2t-1)+t$ with a general monomials parameters $c_i$. $E(x) = c_1 + c_2x + c_3x^2 + ...c_{t+1}x^t$ and $Q(x) = c_{t+2} + c_{t+3}x + c_{t+4}x^2 + ...c_{n+1}x^{(n-2t-1)+t}$

Robots solve the linear system based on BW algorithm $b_i \cdot E(a_i) = Q(a_i)$ using all the points (robot locations) $(a_i, b_i)$ and, if succeeds to resolve $P(x)$, then they verify that $n-t$ robots reside on $P(x)$, search for $P(x)$ in the task maps, and if found move accordingly

**else**

{* when robots failed to find or verify $P(x)$ *}

Robots move to a default polynomial locating themselves according to their order, where first considering the value of $x$ and then the value of $y$ to break symmetry (recall that there is also an order in a set of robot that share the same coordinates) in the current map.

**end**

## 4.2 BW Algorithm Example

Consider the case where the total number of robots is 4, and 1 of which is Byzantine. The honest robots located on the polynomial $P(X) = 1 + 2x \bmod 7$ (the finite field is defined by mod 7), $P(x)$ is not (necessarily) known to the robots but is spelled out for the sake of the demonstrating example.

The honest robots are located on $(0,1), (1,3), (3,0)$ and the Byzantine is located on $(2,6)$, the Byzantine is unknown to the robots, thus, the locations of all 4 robots participate in the algorithm.

Define $E(x)$ to be the error polynomial that returns 0 for the Byzantine robots, the degree of $E(x)$ is $t$ and $Q(x) = P(x) \times E(x)$ with degree of $n-t-1$, in our case $t = 1$ and therefore the degree of $Q(x)$ is $n-2$. Let $a_i$ be the $x$ coordinate of the $i$'th point, and $b_i$ be the $y$ coordinate of the $i$'th point. Define and calculate $b_i \cdot E(a_i) = Q(a_i)$ using the location $a_i$ and $b_i$ for each robot $i = 1 \ldots 4$. Using the equation $b_i \cdot (c_1 + c_2x) = c_3 + c_4x + c_5x^2$.

Point 1 (0,1) yields $c_1 = c_3$

in turn yields (in mod 7) $c_1 + 6c_3 = 0$

Point 2 (1,3) yields $3c_1 + 3c_2 = c_3 + c_4 + c_5$

in turn yields (in mod 7) $3c_1 + 3c_2 + 6c_3 + 6c_4 + 6c_5 = 0$

Point 3 (2,6) yields $6c_1 + 5c_2 = c_3 + 2c_4 + 4c_5$

in turn yields (in mod 7) $6c_1 + 5c_2 + 6c_3 + 5c_4 + 3c_5 = 0$

Point 4 (3,0) yields $0 = c_3 + 3c_4 + 2c_5$

in turn yields (in mod 7) $6c_3 + 4c_4 + 5c_5 = 0$

$c_2 = 1$, a constraint that the most significant coefficient of $E(a_i) = 1$, yielding the last line in the next matrix.

Writing the five equations in a form of a matrix we get:

$$\begin{pmatrix} 1 & 0 & 6 & 0 & 0 & 0 \\ 3 & 3 & 6 & 6 & 6 & 0 \\ 6 & 5 & 6 & 5 & 3 & 0 \\ 0 & 0 & 6 & 4 & 5 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Solving the matrix and gets the results:

$$c_1 = 5, c_2 = 1, c_3 = 5, c_4 = 4, c_5 = 2$$
$$E(x) = 5 + x, Q(x) = 5 + 4x + 2x^2$$
$$P(x) = Q(x)/E(x) = 2x + 1$$

Now the robots can identify the Byzantine robots using the robot positions and the $P(x)$ equation. A mapping of the $P(x)$ to a state (index) can be done, just as the cryptography secret sharing scheme [6] does to define a common secret, using points to encode a polynomial $P$ with the secret being the value of $P(0)$ (or $P(0,0)$ in two dimensions polynomial) defined in the BW algorithm. Thus, the swarm may implement global error free memory, encoding a state and state transition in terms of the common global "secret" $P(0)$ (or $P(0,0)$). The swarm members can move to a state (defined by the locations of the

non-Byzantine robots) that encode the next desired ("secret", or) state.

## 5 Task Schemes

In this section we describe the way function representation can support the major basic tasks in the robot swarm missions. We demonstrate the way our function based policy can support executing the basic missions in self-stabilizing fashion, in the presence of Byzantine robots.

**A series of tasks.** In order to perform real world sequence of tasks, we defined several primitives:

Assuming robots synchronize their current (sub)task to be performed by a global clock value, that can be bounded by an integer $max\_clk$. The clock value is repeatedly changed from 0 to $max\_clk$ and back to 0 (just like modulo $max\_clk+1$ behaves). Let $max\_clk = i_1 + i_2 + \ldots + i_k$. In the first $i_1$ steps are dedicated to the first (sub)task, $i_2$ to the second (sub)task and so on and so forth.

Alternatively to the use of global clock, we can use the current locations of the robots to encode the currently performed (sub)task, as long as once the non-Byzantine robots respect requirements of a (sub)task they do not respect any other (sub)task requirement. If no (sub)task requirement holds, the robot move to a default position of, say, the first (sub)task in the infinitely repeated series of (sub)tasks.

Assuming the robots completed the current task, the next polynomial or shape is a function of the current polynomial, a predefined polynomial or a new shape. For example, if the locations of the non-Byzantine robots in the current map encode a line, $f_i(x, y, z) = z$ with we can define the next task to be, say, a cylinder shape $f_{i+1}(x + t, y + t, z) = (x` - 1)^2 + (y` - 1)^2 + z$, where $x' = x + t$ and $y' = y + t$, starting with time $t = 0$ then possibly moving the cylinder right by increasing $t$ gradually, until it is ready for the next change.

Recall that a state transition from a global view that respects the function invariant maps the robot in the current locations in a map to the next locations in a map respecting

the transition function in the (sub)task. The mapping is defined for the seemingly correct robots (the ones that are located on the current function) and the Byzantine robots (that may stop being Byzantine and join the next function in the predefined location mapping).

We can use more specific use cases for some of the tasks for example, we can define *moving right* to denote the movement of all the robots by increase the $x$ while keeping the $y$ values unchanged. We can define *moving up* as increasing the free coefficient $a_0$ by 1. In general, we keep the invariant of the solution, be it the maximal degree of the polynomial or the function of the shape.

Next consider the basic tasks as listed in [4].

**Gathering.** Under the global view and general movement assumptions gathering can be done very easy in one step. The transition is defined by for all robots to move to the same location. We can choose the location to be a function of the current location, say, the average location of all (non-Byzantine, if identifiable) robots. At this point all robots move to the same tile and the polynomial or shape is reduced to be a point.

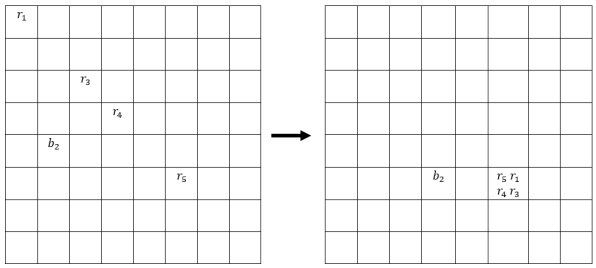



Figure 2: Honest robots are gathering to a specific location on the board. Byzantine robot move to another location and is identified as Byzantine

**Marching.** The transitions are defined by increasing or reducing the $x$ value and update the polynomial to keep the $y$ values, to move right or left, and up or down by increasing or reducing the free coefficient $a_0$. Assuming the robots need to march to a specific direction, the robots can move without changing the polynomial or shape structure (only $a_0$ is changed), or by changing the $x$ value and update the polynomial to preserve the $y$ values.

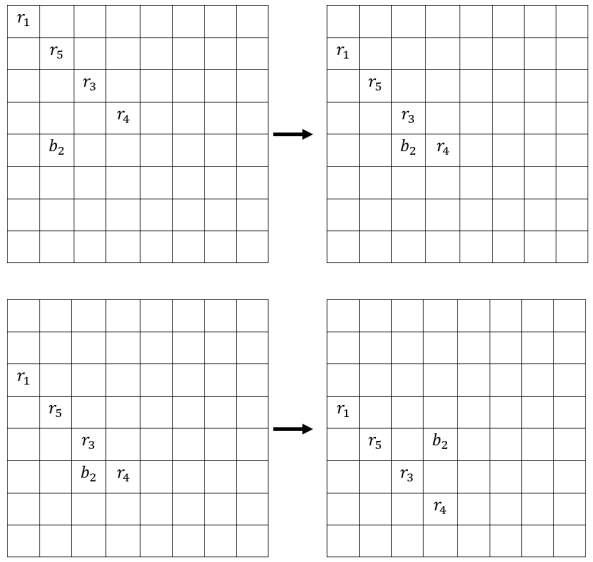



Figure 3: Honest robots march down ignoring the Byzantine robot

**Exploration.** The exploration task is the most complicated task, as at this task we need to pay extra attention to the Byzantine robots. Until now, the honest robots ignore the Byzantine robots and were able to complete the tasks. In the exploration task, the robots need to visit in predefined locations. In case a Byzantine robot decides not to visit a specific location, the honest robots need to visit the specific location instead of the Byzantine in order to complete the task.

The exploration task can be performed in several ways, one example to solve the exploration task is the following: The robots use a predefined state/map, each state represents a specific location the robots need to be located in. The robots can complete the task by moving through all possible combinations of possible locations infinitely often assuming no Byzantine robot is present. As robots do not have memory, the state is in fact represented by the polynomial and the actual-locations.

The task starts in a line segment shape (in fact a polynomial with only a free coefficient $a_i$). The *direction* is defined to be *right*, when $a_i$ is even and *left* when $a_i$ is odd. The

target is to create a row where all robots are located in consecutive tiles throughout the exploration.

All robots Look if the robots on the opposite side of the *direction* has an empty tiles in the row (tiles without a robot). In such a case, the robots stay in place. If not (there is a non occupied tile to the leading robot in the *direction* of the row), the robot moves such that the $x$ values is changed sequentially, and in each step, the robots move to the *direction* by increasing or reducing the $x$ values by 1, until the end of the row. When the robots reach the end of the row and *all* the ($n - t$ honest) robots are located on consecutive tiles the robots move up to the upper line (the mod operation brings all (honest) robots to the lower line as the next line).

In such a scenario, if a Byzantine robot does not follow the algorithm and does not move to the desired location, the other honest robots explore the point and ignore the Byzantine robot. On the other hand, the Byzantine robot can always start to follow the algorithm and join the start or end of the consecutive tiles.

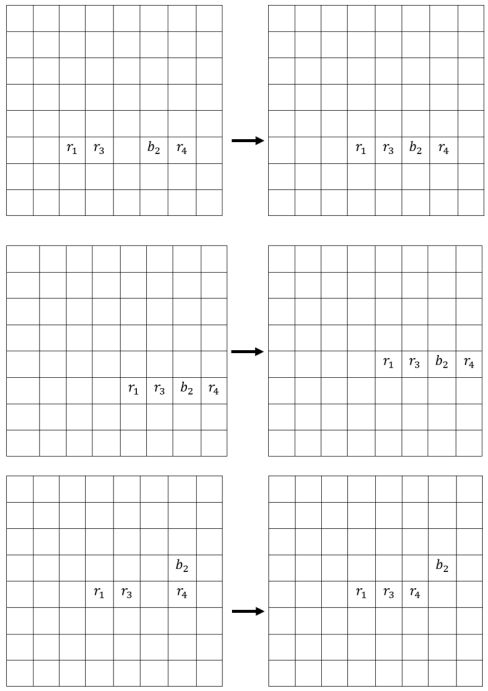



Figure 4: Robots start with $y = 2$ polynomial, create a consecutive tiles (the Byzantine $b_2$ is part of it). The robots move to the direction until the end of row, move up and to the other side. When the robots identify a missing tile (as a result of Byzantine robot action), the leading robots wait for the other honest robots to complete the consecutive tiles and then keeps moving

## 5.1 Locality

At the first stage we assume all robot can look on the entire board. Now, we would like to remove this assumption (For polynomial only, as the shape has limited area by definition). In a nutshell, all polynomial can be calculated using a number of neighbours robots assuming the number of Byzantine in the neighborhood is smaller than $t$‘.

For a line, assuming robots can see 3 more robots and only 1 of them is Byzantine, each robot can calculate the line equation. Without the loss of generality all the robot calculate

the same equation of the line.

[] The case of 3 robots are not accurate.

For a line, assuming robots can see 3 more robots in each side and only 1 of them is Byzantine, each robot can calculate the line equation. Without the loss of generality all the robot calculate the same equation of the line if exist. When a line is valid, the robots need to turn on a light for reset. If $t+1$ lights were turned on, that means there is no valid line.

[] why $2-3$ robots in each side? first, we must assume each robots can see both side to avoid clicks. Second, assuming robots can see 2 robots in each side, consider the case with no-Byzantine no valid line. $(1,1),(2,2),(3,3),(4,4),(5,3),(6,2)$ Robot on $(3,3)$ see the line $y=x$ assuming robot on tile (5,3) is Byzantine.

## 6 Conclusions

We demonstrated the usage of error correcting of functions and in particular polynomials in coping with Byzantine robots in swarms. In the Appendix we show that fundamental swarm tasks can be based on function abstraction including: gathering, marching and exploration. The implementations of these fundamental tasks are self-stabilizing in spite (of a bounded number) of Byzantine robots. Moreover, one can design a sequence of swarm tasks, moving from one task to the next. Typically Byzantine tolerant algorithms cope with a threshold on the number of Byzantine robots, as consistency of the swarm cannot be preserved when too many (e.g., all) robots are Byzantine. Attacker that can overtake several robots, will surely try to exceed the declared threshold on the number of compromised robots. Self-stabilization in spite of Byzantine robots tolerates periods in which even all participants are Byzantine. Once enough robots recover the self-stabilization property ensures that the entire swarm start convergence to act as desired again. We believe we establish a rich useful framework for the design and practical implementation of swarm of robots.

## References


[1] Yotam Ashkenazi et al. "Forgive & Forget: Self-Stabilizing Swarms in Spite of Byzantine Robots". In: 11th International Workshop on Parallel and Distributed Algorithms and Applications. IEEE, 2019, pp. 188–194.

[2] Hadassa Daltrophe, Shlomi Dolev, and Zvi Lotker. "Big data interpolation using functional representation". In: Acta Informatica 55. DOI, 2018, pp. 213–225.

[3] Xavier Défago, Maria Potop-Butucaru, and Philippe Raipin-Parvédy. "Self-stabilizing gathering of mobile robots under crash or Byzantine faults". In: *Distrib. Comput.* (2019).

[4] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. *Distributed Computing by Mobile Entities: Current Research in Moving and Computing*. Springer, 2019. ISBN: 978-3-030-11071-0.

[5] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. *Distributed Computing by Oblivious Mobile Robots*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2012.

[6] Adi Shamir. "How to Share a Secret". In: *Commun. ACM* 22.11 (1979), pp. 612–613. DOI: `10.1145/359168.359176`. URL: `http://doi.acm.org/10.1145/359168.359176`.

[7] Ichiro Suzuki and Masafumi Yamashita. "Distributed Anonymous Mobile Robots: Formation of Geometric Patterns". In: *SIAM Journal on Computing* 28.4 (1999), pp. 1347–1363.

[8] Loyd Welch and Elwyn Berleklamp. "Error correction for algebraic block codes". In: US Patent 4 633 470. 1986.

# Vision-Based Control

Teodor Linnik and Gera Weiss

Department of Computer Science, Ben-Gurion University of the Negev, Beersheba, Israel

**linnik@post.bgu.ac.il**
**geraw@cs.bgu.ac.il**

**Abstract.** Vision-based control is a technique for controlling the motion of a robot that relies on visual sensors. For example, a robot picking fruit guided by visual information and mobile robots which orient in space guided by information from sensor and camera for navigation and avoiding obstacles.

The focus of this thesis is on controlling drones using visual data obtained from the drone's camera. Large amount of visual data was collected and used for training neural networks using deep learning. Types of data that we collected were the history of commands and pictures that the drone took during different flights.

Our hypothesis was that commands given to a drone are a function of a short history of pictures before the command. We demonstrated that we can learn this function from examples by computational learning.

We managed to make the drone fly along a route, detect drone's approximate location along the flight and pass an obstacle based on data obtained from the camera.

We demonstrated that by collecting a large amount of data and using it for training neural networks it is possible to teach the network to fly a drone. While many challenges are still open, we proved that the concept works in practice.

# 1 Introduction

Micro Aerial Vehicles (MAV) are widely used for civilian and military purposes such as intelligence gathering, security, disaster rescue etc. This, of course, opens the question of how to control them.

The state of the art architectures in MAV control utilizes information collected by an on-board camera, GPS and sensors as input for a feedback loop, which generates, as output, the speed of the drone's rotors. However, none of the control systems provide perfect efficiency in drone control for the following reasons, among others. Some of the systems have miscalculations in feedback loops, others are expensive. Many environments, e.g., when surrounded by high rise buildings, are GPS-denied due to unavailability of the signal which means that some additional sensor has to be used for MAV control in those spots.

Our focus in this work is on indoor flights - without GPS and with very little margins of error. We need some other input that indicates the position of the drone relative to its environment. For indoor flights, most of the work that we are aware of focuses on using expensive sensors such as Light Detection and Ranging (LiDAR). We examine vision based control both because of practical issues such as cost, accuracy and weight and because of theoretical reasons, as vision is the main sensor used in nature. In a previous work, Efraim et al. [1] demonstrated that it is possible to use the lines of a corridors to stabilize an Unmanned Aerial Vehicle (UAV) in a flight along a corridor. Using Deep Learning (DL), we demonstrated that this can also be done without the complicated and mission specific calculations that Efraim developed.

In recent years the field of DL demonstrated impressive results in the areas of perception, planning and control. One of the examples is visual reasoning, where a Neural Network (NN) is asked to answer a question using a photo, movement in complex environments using Deep Reinforcement Learning (DRL). For example, autonomously human-like car driving [2].

We show in this work that it is possible to achieve better control performance and extend the applicability of vision based control of drones by applying Deep Neural Networks (DNN) to the task. DNN based controller takes a short video recorded by the drone's front-facing camera and gives the pitch, roll, yaw and throttle command that an operator usually sends by the remote control (see Figures 1.1 and 1.2). This is done by frequently imitating the operators of drones that translate what they see to actions with the sticks of the remote control (see Figure 1.3). In the end, we just learn a function that maps short movies of previous history to quadtuples of numbers. This method is better than the approach presented by Efraim [1], because it applies to more general situations, not just to flying in a corridor, and because it can use many visual cues, not only the lines of the corridor.

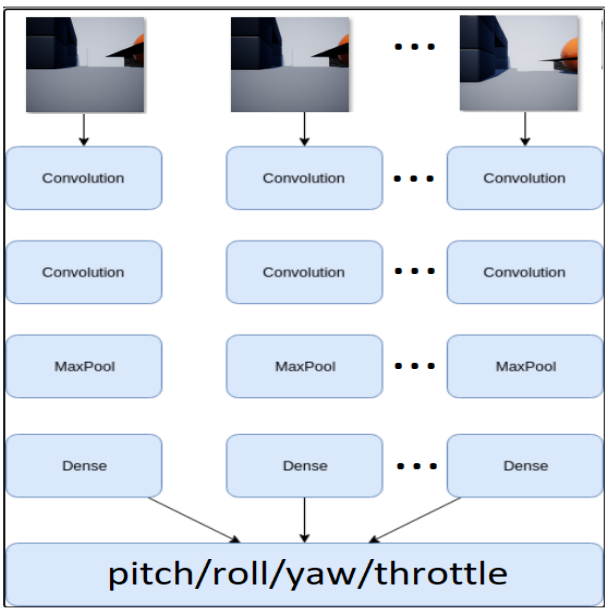



Figure 1.1: Structure of a DNN which gets short video record as an input (modified from [3]).

In this thesis we present our efforts to design methods that can fly drones by learning the function described above. We start by describing how we convinced ourselves that it is possible to identify the type of features needed to decide how to pull the sticks and combine them together to an autopilot that flies a drone using just the video feed from an on-board camera. We, then, describe how we have progressed towards a control system capable of identifying all the needed features and of deciding on the right

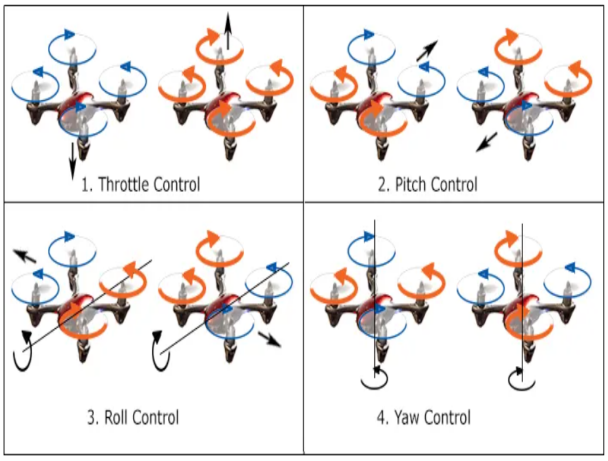



Figure 1.2: Change of drone's flight direction in response to different control command that we used in our work (taken from [4]).

Figure 1.3: Controlling a drone using the remote control sticks (taken from [5]).

command to give to the drone. We describe our network, the infrastructure that we have developed to support the training and evaluation of the network in experiments with different flight situations. The results are reported using statistics and flight examples.

# 2 Brief explanation of Deep Leaning

DL is a subset of Machine Learning (ML) in Artificial Intelligence (AI) that tries to mimic the functions of a human brain in making decisions based on a processed data. It is also known as a deep neural learning or deep neural network. From different types of input data such as videos, images, sounds, text etc the function model learns to perform correct classification or prediction. One example in which DL is used is teaching a function to predict if in a picture is a cat or a dog. In DL there is a term called 'loss function' (referred as 'loss') which is used to indicate how well adequate model performed predictions on data, how much this predicted value deviate from the correct value. The model (which is based on Neural Network (NN)) is learning to achieve high classification accuracy by training on large amount of examples (raw data) and correct classification data (label data) for each raw data. This type of learning is called deep learning because each model contains 'hidden' layers (in addition to layers for input and output data). Each layer contains different amount of nodes (neurons), while each neuron has weight. The user defines how many times (epochs) the NN will be trained on total amount of data. Each epoch consists of different amount of iterations depending on the batch size (BS). The BS is a number of training examples used per iteration. After each iteration the NN is trained through all layers and weights are updated. The way for updating the weight is done by optimizer (such as stochastic gradient decent, Adam) with main purpose to reduce loss function value. In order to control number of weights which should be updated in response to the loss function value (while moving toward a minimum value of a loss function) in each iteration we define the value of learning rate for each optimizer. The main purpose of DL is to train NN to assign weight to each neuron in a way to reach minimal value of a loss function of specific model. Data that we used were divided into trained data, that we used

for training our NN, and validation data, that we used to check the NN performance on data unknown to NN.

In this thesis we are going to work with following types of NN:

- Convolutional Neural Network (CNN), used when input is image data.
- Recurrent Neural Network (RNN), used when input is a sequence of data. For example, time series or words.
- Long Short Term Memory (LSTM), network from RNN family which prevents the vanishing gradient problem. The meaning of this problem is that the weight values are not being changed during the training, which can eventually lead to aborted training.
- Gated Recurrent Units (GRU), network also from RNN family. This network prevents the vanishing gradient problem in a similar way to LSTM but it's structured in a different way.

# 3 Previous Work

Several research groups proposed methods for overcoming existing problems in MAV control. One approach was introduced by Efraim et al. [1]. They used a controller based only on visual measurements from a front facing camera and Micro Electro Mechanical System (MEMS) angular velocity sensors as input, without the need for any external input. The novelty of this controller is that it uses the vision both for guidance and for stabilization of MAVs simultaneously. This method does not require any linear velocity estimation. In addition, MEMS angular velocity sensors are less sensitive to vibrations than accelerometers. This approach does not require explicit attitude feedback, which means that it does not use accelerometers and complicated estimation algorithms. Moreover, in cases like near hover situation, when the sensors provide poor information, this controller can be used as a backup strategy.

Another approach to control MAV was introduced by Goldman et al. [6]. This work proposed algorithms for controling a system that managed to efficiently stabilize the drone in front of a window. It was done by a dynamic allocation of Central Processing Unit's (CPU) resources, only when resources were really needed. Those algorithms were based on automata, state machines in which each state had assigned specific resource that they were supposed to use each time. Each state can be also called environmental condition. In this way, computational resources would be saved if environmental conditions really allow. By properly defining requirements, we can decide to use real time computation methods or not. Those advantages were shown by simulations and then by real experiments with better results than previous methods. They did three experiments to compare performance. The performance was calculated by the error between mean distance between $x$ (the real state) and $\hat{x}$ (the Kalman filter state), $|x\text{-}\hat{x}|$. First experiment used 10% amount of CPU consumption, second used 85% of CPU consumption and the third one related to their approach used the Automaton for basically allocating the CPU resources only when

they were needed. As expected, the performance of the second experiment was better than the first. The performance of the second experiment was a little bit better than the third one but using the Automaton for allocating resources only when they were needed made them to spend much less CPU resources.

Padhy et al. [7] proposed a method to control a drone by sending instructions to a drone to move right, left, forward or stop each time when a drone should move in specific direction while keeping constant altitude. Briefly, they used DL to train Convolutional Neural Network (CNN) DenseNet-161 pretrained model with four output classes-movements right, left, forward and stop. They used Euclidean Loss function:

$$\text{Euclidean Loss} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

where, $y$ and $\hat{y}$ represent the ground truth and predicted class labels. N is the size of mini batch. They collected a lot of image data in different corridors. They drastically increased a number of image data by doing various augmentation techniques, such as zooming and flipping. They basically ran two type of experiments:

1. No-Collision-Ratio (NCR): Number of times over the total number of trials the quadcopter successfully navigated the whole length of the corridor without any collision with the walls.

2. Full-Flight-Ratio (FFR): Number of times over the total number of trials the quadcopter successfully navigated the whole length of the corridor, albeit there may be slight side-wise collisions with the walls, which do not hamper the direction of the UAV.

The accuracy according to the first experiment (NCR) was 0.733, whereas the accuracy according to the second was 0.847. As we can see, there is still place for improvement. For training the network they used obstacle-free environment which is not corresponding to the real-life environments.

Because of collision of the drone with the corridor walls and the fact that deviation can happen also by rotation left or right, they suggested a new approach [8] in which a drone flies only along designed central bisector line (CBL), which is positioned in the middle of the floor spanning the whole length of the corridor and is parallel to the side walls of the obstacle-free environment and the rotation angle was zero degree from

the CBL. This is an imaginary line, which is used as a reference to measure translation and rotational deviation of the UAV from the center of the corridor. Translation or rotational deviation from the line could occur from factors such as wind, drone's rotor turbulence etc. Because of this they considered all drone's commands (pitch, roll, yaw). During the flight, each time when any deviation from the CBL happened, first priority was to retrieve drone's direction angle of 0 degree from the center by giving rotational command. Second priority was to re-translate drone's position to the CBL line by generating suitable command (left or right). Whenever the drone's position was aligned with the CBL, the drone moved forward. They trained neural network separately according to rotational deviation and translation deviation. For training their model they used different pre-trained networks (AlexNet, VGG-16, InceptionV3, ResNet-50, ResNet-101, ResNet-152, DenseNet-201, DenseNet-161). For each network they tried 3 different evaluation metrics, Mean Squared Error (MSE), Mean Absolute Error (MAE) and Mean Relative Error (MRE). According to the experiment, the network DenseNet-161 performed the best in terms of all the evaluation metrics either for translational or rotational deviation tests. As they obtained good results in most of the corridors, this implies that flight depends in a big portion of the specific environment for flight, which in their case was a corridor.

Garcia et al. [9] proposed a method for drone navigation in an obstacle-free corridor by obtaining a short video stream from the front facing camera as an input and the goal was to detect if there were dead-ends, doors and intersections in hallways (using CNN). Around each recognized object they used pixel-scale dimensions of the bounding boxes for using Support Vector Regression (SVR) model to estimate the distance around recognized object in order to perform specific action according to recognition: to continue to flight straight forward, turn left right (by 90 degree) or to make U-turn (by 180 degree). For this, they ran three experiments, the first two were considered as a preparation. First experiment were autonomous straight flights through hallways to test the drone autonomously flying in a straight line and test the action U-turn (rotate by 180 degree). Second experiment were autonomous flights with turns in order to tests drone's ability to autonomously detect intersections and make 90-degree left or right turns and make 180-degree U-turns when and where needed. The third experiment were multi-path autonomous flight missions in which drone had to fly along specific flight paths involving a number of different hallways/intersections to test the combination of actions mentioned before. During the experiments, few wall contacts occurred, but they were suc-

cessfully handled by the control node which permitted continuous flight, all demonstrating a success. They assumed that the only option to change the moving direction was to rotate by yaw with one of the following angles $(0, 90, -90, 180)$.

Amer et al. [10] suggested combination of CNN and Gated Recurrent Units (GRU) neural network or Fully Connected Neural Network (FCNN) in controlling the drone to fly only along specific path in Unreal Engine-based AirSim simulator, which contained different points along the predetermined road called waypoints. They used these waypoints during the experiment to calculate error distance between actual position of the drone and other waypoints according 2 different error metrics. Drone was kept at constant height, moved with constant velocity forward and flight direction was changing by rotating left or right for a yaw angle. They performed their experiments in two different simulated environments called Blocks and Landscape. The Blocks scenario represents abstract environment containing cubes with different arrangements and colors while the Landscape environment is more realistic which contains a complex scene containing frozen lakes, trees, and mountains. To be closer to real environmental condition they also generated noise. When they compared results obtained by two regressors, GRU and FCNN, they realized that working with FCNN brings less error which means it brings better performance. Novelty of this work is that they took into account the noise of a real environment such as wind in their simulations. But this work did not use 'roll' option command, which could have bring many benefits to the study.

Maciel-Pearson et al. [11]. suggested to identify clear flight areas and predict the flight behavior while exploring an unknown environment in order to eventually navigate a drone to specific point. The navigation route was divided into N waypoints. A drone was supposed to predict how to reach the next waypoint by predicting the next position and rotation of the drone in a given unknown environment in order to perform optimal exploration. Positional orientation was determined in NED (North, East, Down) coordinates in meters, while rotation of the drone was established in orientation quaternions because quaternions requires significantly smaller amount of memory than calculating rotational matrices, thereby making them more suitable for their work. For position and rotation orientation prediction they used Multi-Task Regression-based Learning (MTRL) approach, which means that for the same image input they used two identical NN which both included convolution, Fully Connected (FC), Pooling and dropout layers with only one difference between networks, which is the final layer for output. In the first NN the output

predicted the next position and in the second NN the output predicted rotational orientation in quaternions. They worked with the AirSim simulator. They collected data for training in Redwood Forest simulator's environment. Afterwards they performed experiments and compared the results with three other approaches for their specific task. The experiments showed that their method is capable of generalizing to unseen domains of an environment and has a larger coverage area than comparators. Their method demonstrates better navigation performance due to a wider Field of View (FoV) in comparison with other approaches.

# 4 Methods

For simulating flights sessions we used the open-source AirSim simulator [12]. This simulator is built on top of the graphical development tool Unreal Engine [13]. We used it for visually generating realistic and physically correct scenarios. Sample projects written in C++ programming language are available online [14]. When running the projects from Visual Studio, the window of Unreal Engine is opened and the user can turn on the drone's simulator. After that one can chose to connect a joystick to the computer to control the drone. Another way that we used is using a "client-server", where the server is the simulator and the client are commands sent by other running program written in Python (See Figure 4.1).

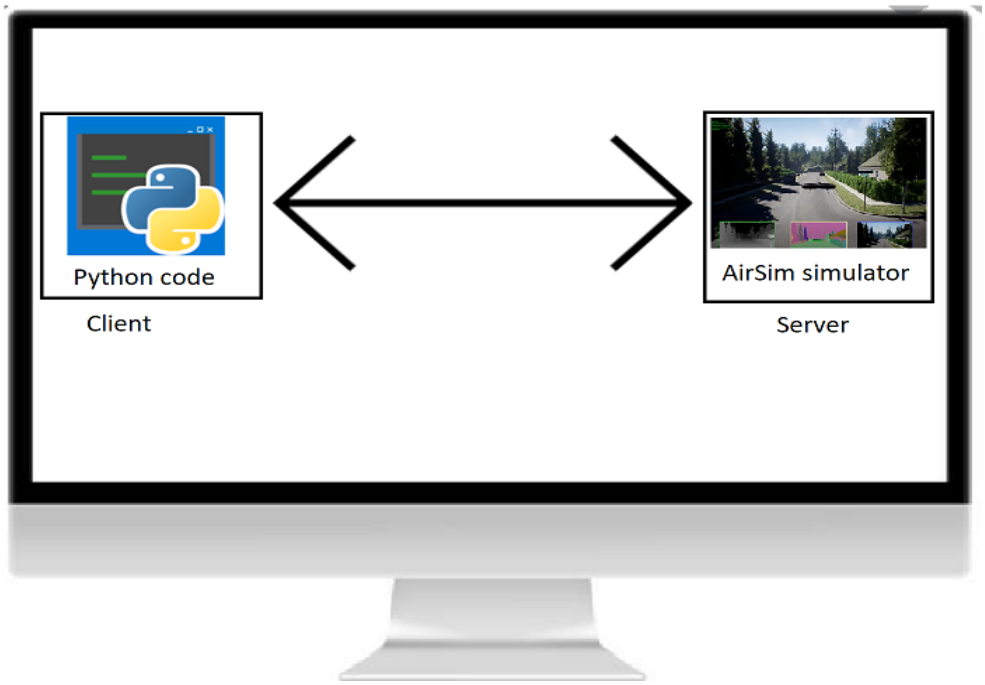



Figure 4.1: Communication between client and server in the same PC.

Using this method of control, we generated flight sessions. In each session we commanded the drone to fly in a specific direction or at a specific angle such as pitch, toll, yaw. In addition to giving commands to the drone, we generated noise that simulates ,e.g., wind by changing the aforementioned C++ codes. Each flight session was recorded in a specific folder, which contained the sub-folder of images taken during the flight session and a log file where each line contains the name of the image file taken in the previous sub-folder, and the average of preceding 100 pitch, roll, yaw, and throttle commands to the drone at the moment of taking the picture image by the camera (for getting those exact lines we also made changes in C++ code). A set of flights with specific combination of pitch, yaw, roll command was labaled to one specific class. We created a large amount of data in order to train neural network (in each experiment amount of data was different). We trained network with a Python code using the Keras [15] library.

We also needed a method that could predict the drone's action based on short moving history. We based our work on a paper by F. Patrice [16] in which the author performed classification according to a short video action. He used a large amount of short time video sessions, where each video session was related to one of the following action, that he used as class as: playing golf, making dribble, kicking ball. He used label data to train a neural network that he constructed. The construction was the combination of CNN and GRU/LSTM network.

We trained the network with two methods. The input for the network depends on the method we are using. In the first method called "Sliding frame sample", the sequence of $N$ images was recorded and K number of images, where $K < N$, is taken each time as an input for training the network as follows $\{0, \ldots, K\}, \{1, \ldots, K+1\}, \ldots, \{N-K, \ldots, N-1\}$. The input size is $N - K + 1$, see Figure 4.2.

In the other method called "Picking frames from the entire video", the entire length of the video is divided into $N$ number of frames, where $N > 1$ and the time interval between each frame is of the same length, see Figure 4.3.

For constructing the network we used two models. In the first one called "Custom ConvNet", the idea was to first create some CNN ('ConvNet') for several sequences of input images in chronological order, connect this to a time distributed layer in order to detect 'features' and produce one dimension output which is then injected to GRU or LSTM to process the "time series". This network is then connected to a dense net for classification.

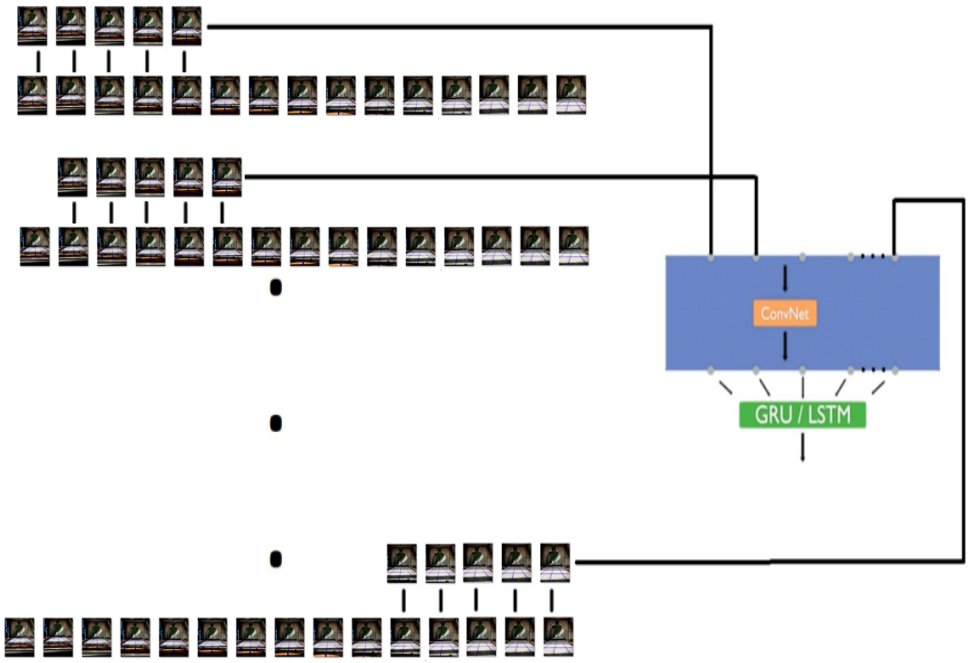



Figure 4.2: Sliding Frame Sample method, in this case $K = 5, N = 15$. Adapted from F. Patrice [16] .

The second model that we used is called "Standard model", which is 'pre-trained'. This means that this model had been trained before for solving similar problem to ours, it was not trained from scratch. In this model we used the special CNN called MobileNet and made it's $N > 0$ top layers "trainable" (weights influenced by our data), while the rest of the layers remained to be "untrainable" (weights are not influenced by our data).

We first trained the network using the "Custom ConvNet" model, then the "standard model" and compared the results. The aim was to use different models in our experiments and methods in order to improve previous works in the field and to obtain more optimal results.

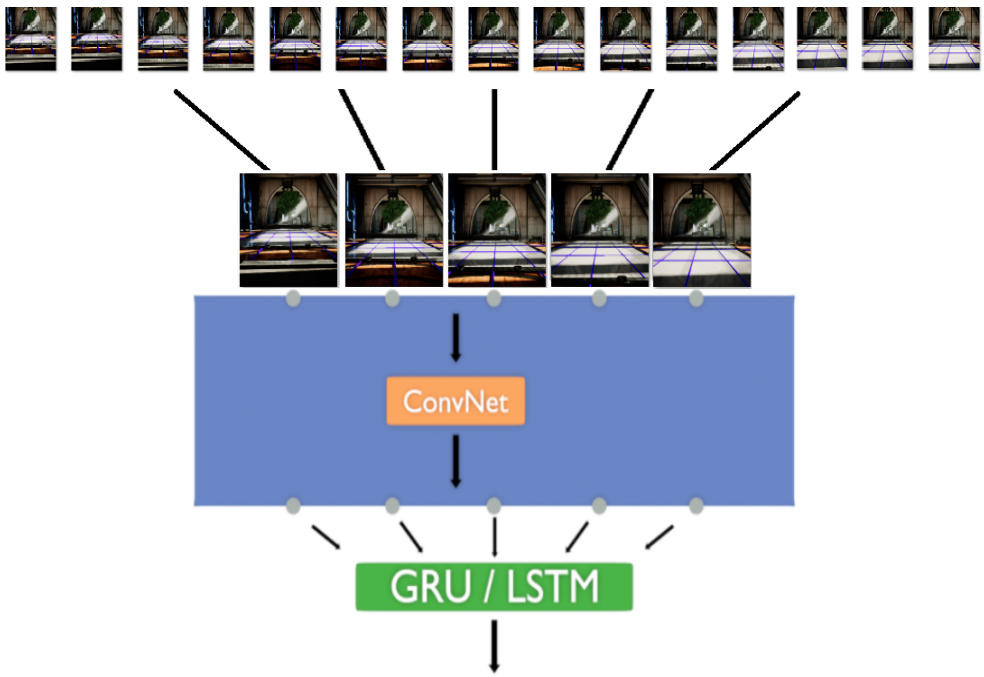



Figure 4.3: Network according to the method method called "Picking frames from the entire video". Adapted from F. Patrice [16].

# 5 Experimentation facilities and setup

For experiments we used Python (version 3.8) because this is high level language which is widely used for DL, Image Processing, flight simulator and in many of other fields.

For DL and for processing a large amount of data, we needed servers with high performance and memory and graphics processing units (GPUs). We used the server from Ben-Gurion University of Negev, "BGU ISE-CS-DT Cluster" which is a job scheduler and resource manager. It consists of a manger node and several compute nodes. A job (which in our case is an execution of a Python program) is allocation of compute resources such as RAM memory, CPU cores, GPU, etc. for a limited time. For constructing and training the network we used Keras [15] from TensorFlow [17] library. We used CPU with a clock speed of 2.2GHz, 1 GPU cores and for each running code we allocated memory of 24G byte.

# 6 Training network on simpler tasks

Our methodology towards designing the network for flying drones was to break the complex task by creating a set of data sets that focus on specific aspects of the composite task. Specifically, we observed that a neural network that can translate short videos that the drone captures into actions needs to be able to detect how shapes evolve in time, it needs to be able to detect movements of the camera, and it must be able to combine these together.

## 6.1 Reproduction of the previous work results

We started by repeating the results of F. Patrice [16] of using a combination between Recurrent Neural Network (RNN) such as LSTM or GRU with CNN for classifying video actions (such as golf, dribble, and kicking ball). We took sub-sequences of frames at equidistant times and used them to predict the action. We started with the data set from [18]. We chose the network proposed by F. Patrice (described in Chapter 4) as a starting point because we ,like him, want to classify videos (and use this classification to decide on a command that should be given to the drone). Of course, there is a difference between the classification that F. Patrice wanted, which is based mostly on content, and that we want, that should mostly be based on movements of the camera, but, as we will show below, these are not that different. We performed 14 different experiments. For each experiment we achieved different results because the train and validation data shuffled randomly for each epoch and for each experiments and the weights on the layers were initialized randomly. The range of validation accuracy in our experiments is from 72% to 92%. One example of the results we obtained by repeating F. Patrice work is presented in Table 6.1.

| | Our train | F. Patrice train | Our validation | F. Patrice validation |
|---|---|---|---|---|
| Accuracy | 0.88 | 0.82 | 0.80 | 0.74 |
| Loss | 0.36 | 0.48 | 0.58 | 0.48 |

Table 6.1: Comparison between our and F. Patrice results after reproduction of the network. The table presents one results from total of fourteen experiments that we have performed.

## 6.2 Shape Morphing

During the drone's flight we must predict next command based on changes in angle, location, and size of objects from drone's front facing camera.

Since the task that we wanted to perform involves detection of dynamic shape changes, we developed a simple data set with which we could test how different NN handle the task of identifying a transformation of one shape to another. Each shape had the same size, to avoid identification by size. Morphing process occurred in a random point inside the frame as seen, for example, in Figure 6.1. We also made sure that the only way that classification was done was by detecting the dynamics of the transformation, i.e., that it is not possible to classify only according to a single frame. Accordingly we created data and performed experiments.

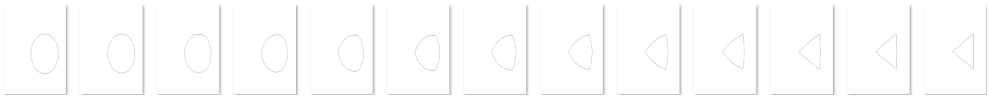


Figure 6.1: A circle that gradually morphs to a triangle.

In the beginning, we created 150 video clips for each class while total number of classes was four. The meaning of each of those classes is as follows:

1. First class - a circle that gradually morphs to a rectangle.
2. Second class - a circle that gradually morphs to a triangle.
3. Third class - a rectangle that gradually morphs to a circle.
4. Fourth class - a triangle that gradually morphs to a circle.

We first tried to apply our data exactly on the same network as F. Patrice (his first model). This gave us results that were not better than a random guessing as if we did not trained the network. We canceled the data

augmentation option (such as zoom, shift, rotate, shift image) which was useful for data described in the previous sub-section but not relevant here, we performed experiment by using GRU network from RNN family. This gave us around 89% average validation precision in this task, as detailed in Figure 6.4.

We also performed another experiment by using LSTM (instead of GRU). This network is also from RNN family. We did this in order to compare which one fits better our needs. This gave us around 77% average validation precision in this task, as detailed in Figure 6.5. In general, using GRU gave us better results than using LSTM, which convinced us that using GRU is better but in further experiments we tried LSTM too.

These numbers were still disappointing since we expected more than 95% precision. We decided to try F. Patrice second model: we added CNN layers by the well known MobileNet [19] network which is comparably smaller, faster, and more accurate. By doing this we achieved close to 100% precision (see Figure 6.2). We learned that using pre-trained neural network such as MobileNet gives us better results than manually trying to construct our own network.

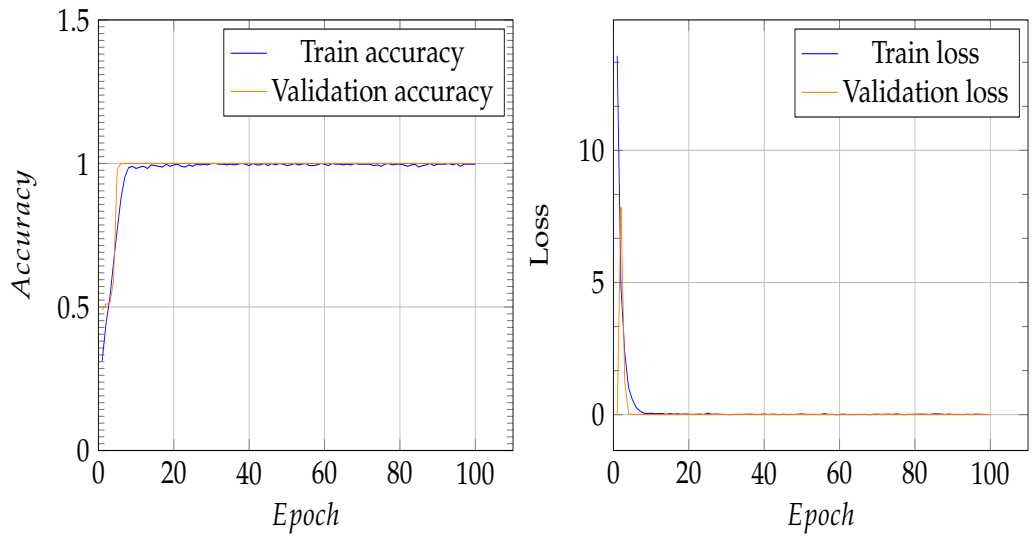



Figure 6.2: Accuracy (left) and loss (right) graphs associated to the classification of the conversion between circular, triangular and rectangular shapes. Blue lines represent training accuracy/loss and orange lines represent validation accuracy/loss.

Beyond shape morphing, we identified two other factors that we need our network to be able to recognize when flying drones: resizing and rotation.

To this end, we added to the process of shape morphing following factors: random size of the shapes, gradual translation during the shape morphing from one random point to another and rotations around the axes. For example, Figure 6.3 shows an example of a circle that gradually morphs to a rectangle, rotates, and moves.

Figure 6.3: A circle that gradually morphs to a rectangle including gradually translation and rotation.

We created video clips containing four types of action (same idea as before):

1. A circle that gradually morphs to a rectangle.
2. A circle that gradually morphs to a triangle.
3. A rectangle that gradually morphs to a circle.
4. A triangle that gradually morphs to a circle.

We stress, again, that predicting the class requires an analysis of the dynamics. It cannot be inferred just by looking at a single image.

According to these data we performed an experiment whose results are presented in the Figure 6.6.

We also tested in one of the experiments if usage of RNN is at all essential for our work. We replaced the time distributed layers in those networks with a "Flat" layer (Fully Connected Neural Network, FCNN) and compared the results obtained from training those different networks. As we expected, this replacement degraded the results (see for example Figure 6.7).

Up to now we used shape morphing data. In order to prove that shape morphing is really what the network detects, we replaced the shape morphing video data with video in which image of interest had constant shape throughout the whole video (see Figure 6.8). We chose the middle image from the sequence of the shape morphing data images to be the single image we will use in this experiment. Small time intervals were of the same duration like in the previous experiments. Obtained results are shown in

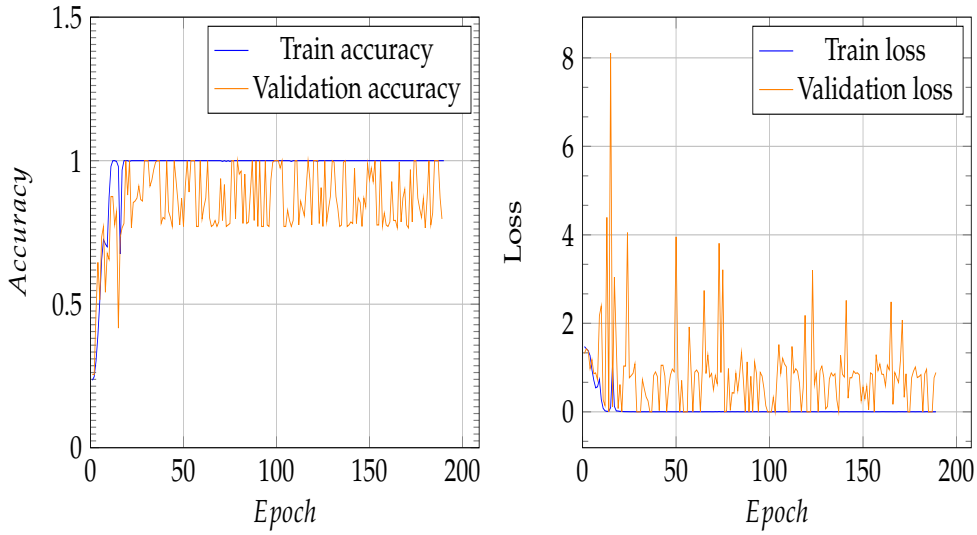



Figure 6.4: Accuracy (left) and loss (right) graphs associated to the classification of the conversion between circular, triangular and rectangular shapes. Blue lines represent training accuracy/loss and orange lines represent validation accuracy/loss.

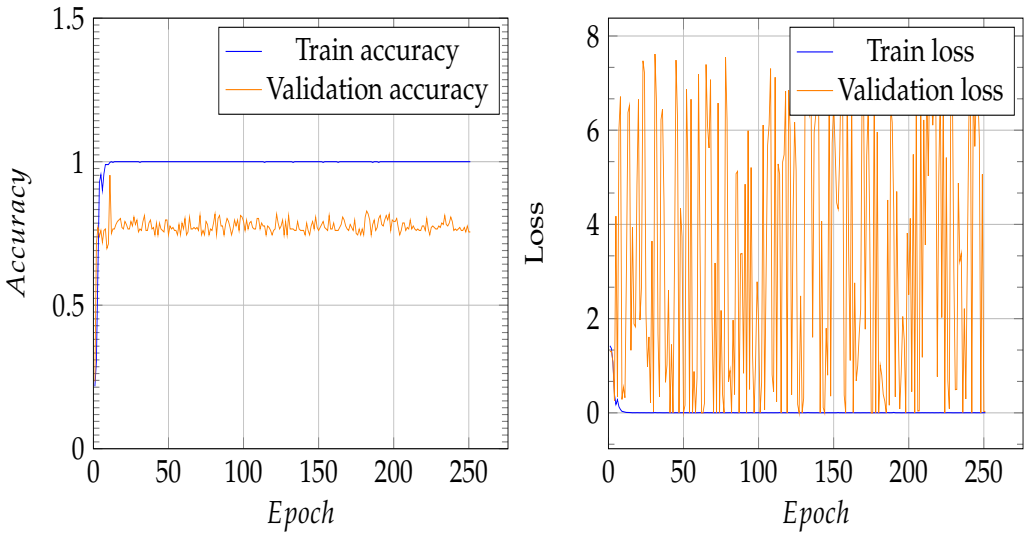



Figure 6.5: Accuracy (left) and loss (right) graphs associated to the classification of the conversion between circular, triangular and rectangular shapes. Blue lines represent training accuracy/loss and orange lines represent validation accuracy/loss.

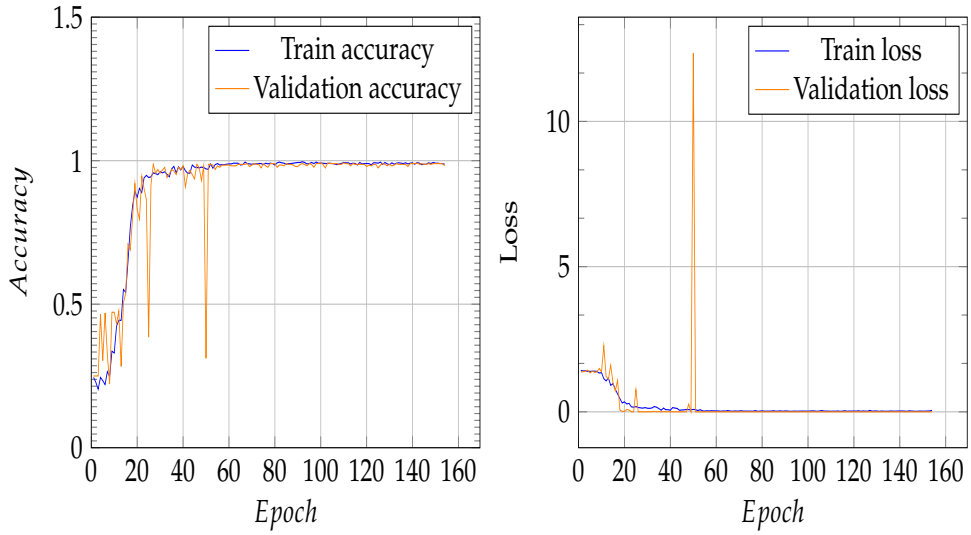



Figure 6.6: Accuracy (left) and loss (right) graphs associated to the classification of the conversion between circular, triangular and rectangular shapes. Blue lines represent training accuracy/loss and orange lines represent validation accuracy/loss.

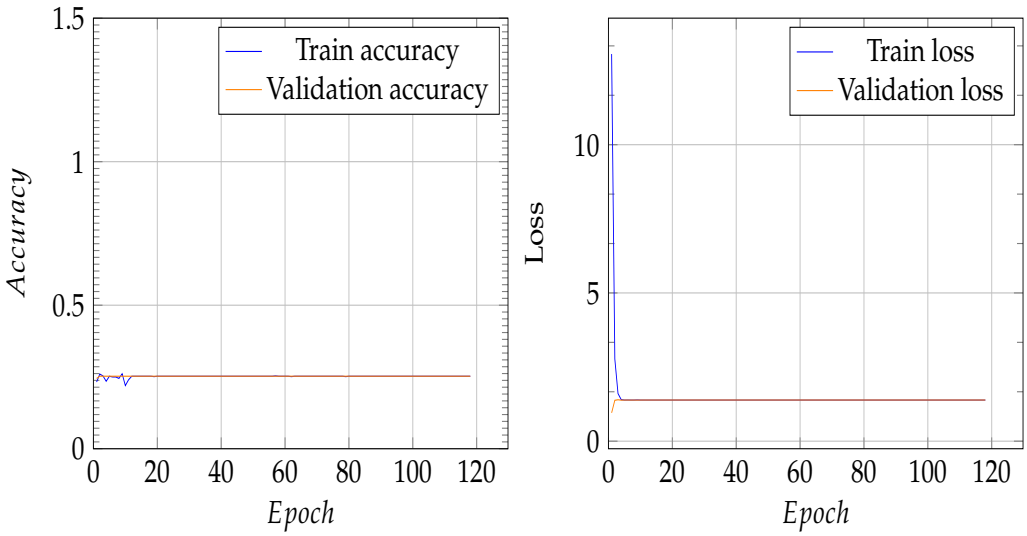



Figure 6.7: Accuracy (left) and loss (right) graphs associated to the classification of the conversion between circular, triangular and rectangular shapes. Blue lines represent training accuracy/loss and orange lines represent validation accuracy/loss.

Figure 6.9 (where we used GRU) and Figure 6.10 (where we used LSTM instead of GRU). We can see that the validation accuracy is around 25% which means that the network was not effective and that the results are similar to the results of random guessing. These are, of course, good news. Specifically, from these experiments we concluded that the network that achieves close to 100% precision does so based on a identification of shape morphing.

Figure 6.8: A video in which image of interest had constant shape throughout.

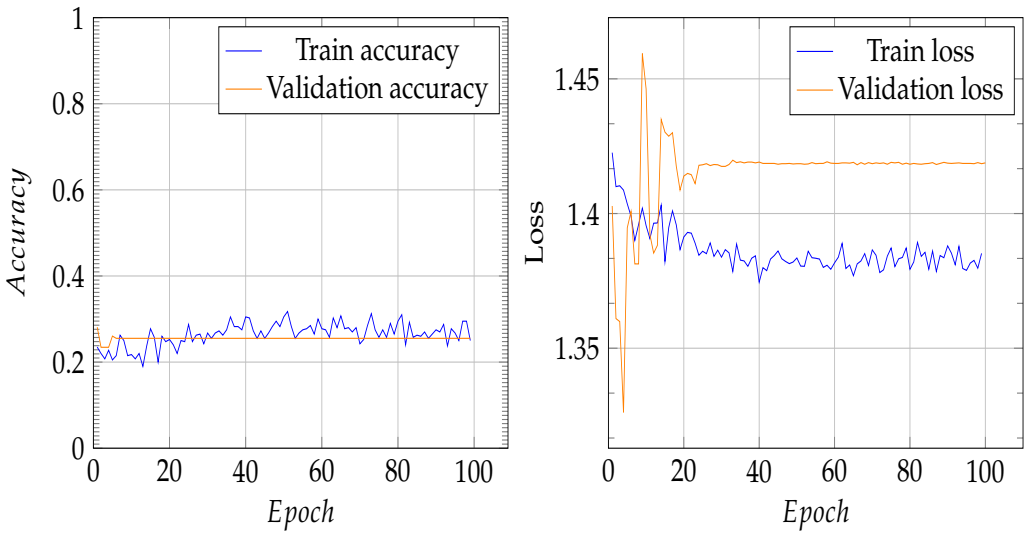



Figure 6.9: Accuracy (left) and loss (right) graphs associated to the classification of the conversion between circular, triangular and rectangular shapes. Blue lines represent training accuracy/loss and orange lines represent validation accuracy/loss.

We created sequences of images of the shape $(112, 112, 3)$ and converted them to video clips. We chose the shape $(112, 112, 3)$ because F. Patrice worked with those dimensions. This is also the shape of the images we read by the drone's camera. During the experiments we divided each clip into twenty equidistant intervals because each part of previous short history (interval) has the same value. Some clips were one second long and

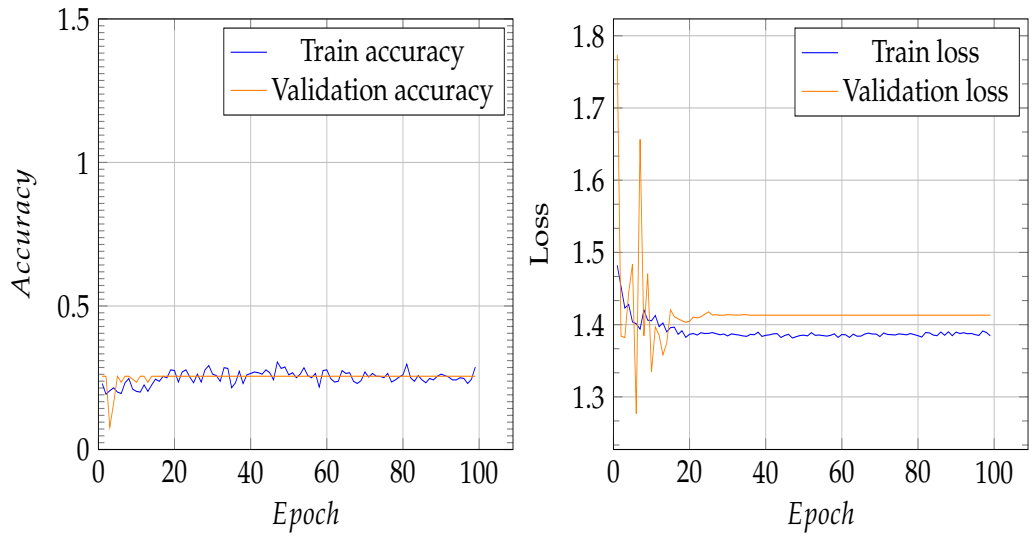



Figure 6.10: Accuracy (left) and loss (right) graphs associated to the classification of the conversion between circular, triangular and rectangular shapes. Blue lines represent training accuracy/loss and orange lines represent validation accuracy/loss.

some were three seconds long, which are typical history frame lengths that are needed for flying drones, according to our estimates.

We edited the networks in the following ways:

- We performed experiments using a method "Sliding frame sample" (described in Chapter 4), but running a program consumed enormous amount of memory and demanded huge number of iterations per epoch. As a result, completion of one epoch took substantial amount of time. For this reason we decided to continue with a method "Picking frames from the entire video".

- The batch size in our network was arbitrarily set to 8 in our initial experiment, as was in [16]. To test if this is the right number for us, we first tried to decrease it to 4, which means that we doubled the number of iteration per epoch. As expected this resulted with faster changes in training loss or accuracy between iterations (see, e.g., [20]) . But it also increased noise and decreased generalization performance (ability to predict data which was not used during the training, see, e.g., [21]). We then observed that our network gives us better results with increased batch size, thus we chose to use batch size 30, which is the maximal reasonable number, as explained next.

The size of the training data set is 268 (for validation we used data set size of 132 which means what total data set was 400), which gave us 9 iterations per epoch. In order not to reduce additional numbers of iteration per epoch we decided not to increase a batch size more with a goal to allow easier tracking of a train loss and accuracy changes during the epoch.

- During experiments we noticed that it took a long time until the loss functions converged. Looking at the numbers, we noticed that the problem was with the required rate of convergence - we only stopped the learning process after a very long period of almost zero error. We first tried to increase the learning rate (LR) to $10^{-3}$ and observed a degradation in performance. We then tried $5 \cdot 10^{-5}$ and obtained good results, but with slow convergence. We obtained the best results using LR value of $5 \cdot 10^{-4}$. From this we inferred that a value $10^{-3}$ makes a jump over minimum in loss function while the value of $5 \cdot 10^{-5}$ takes too long to converge or gets stuck in an undesirable local minima in the loss function.

- In some of the experiments we noticed a gap between training and validation accuracy (a phenomena known as over-fitting). While training accuracy converged almost to 100%, validation accuracy was significantly less (it went as low as 50% in some cases). For this reason, we decided to do the following. To each layer, after GRU/LSTM, we added a regularizer which during optimization puts a penalty when more neurons are active. These penalties are accumulated to the loss function that the network optimizes. This acts as a force for minimizing the number of active neurons. This force prevents overfitting in neural networks and consequently increases the accuracy of a Deep Learning model when training with a new data from the problem domain (see, e.g., [22]). During the experiment we tried to use $L_1$ and $L_2$ regularizers (see Figure 6.11 and Figure 6.12 ). $L_1$ regularization technique is called Lasso Regression and model which uses $L_2$ is called Ridge Regression (see [23]). We also tried regularizer values of 0.1, 0.01 and 0.001. We obtained the best result using $L_2$ regularizers and regularizer value 0.01. After doing this we managed to shrink a gap between training and validation accuracy under 15% in all the experiments.

Once we convinced ourselves that we can classify the shapes morphing videos using the RNN, we moved on to the more complex task of identifying 3D transformations.

$$Loss = error(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i| \tag{6.1}$$

Figure 6.11: Loss function according to the $L_1$ regularizer.

$$Loss = error(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2 \tag{6.2}$$

Figure 6.12: Loss function according to the $L_2$ regularizer.
where:

$y$ = True value
$\hat{y}$ = Predicted value
$\lambda$ = Regularization parameter
$w_i$ = weight in the index i

## 6.3 Rotation classification in real environment

After we got good results in the previous sub-section, we wanted to show that the network is able to classify between rotations of the camera along one of the axis's ($x$,$y$ or $z$) according to the Cartesian coordinates which correspond to yaw, roll, and pitch rotations (see Figure 6.13). We did this because a pilot that flies a drone needs to identify the rotations of the drone relative to its environment which translates to rotations of the camera. Specifically, in our opinion the main job of the pilots is to stabilize the drone along a desired flight path which essentially means that they need to react to tilts of the camera.

By modifying a software that generates a movie in a virtual 3D scene [25], we first created an environment with one object, ground and landscape in order to be as similar as possible to drone's real environment. We created three folders called 'yaw', 'roll' and 'pitch' for videos of corresponding camera rotations. Each folder contained 400 sub-folders each containing twenty-four images that represent a sequence of camera shots in chronological order. We chose to use twenty-four images because video technology operates at twenty-four images per second and, as mentioned above, because we estimated that 1s is a reasonable history of video for determining which command should be given to a drone. In each sequence we

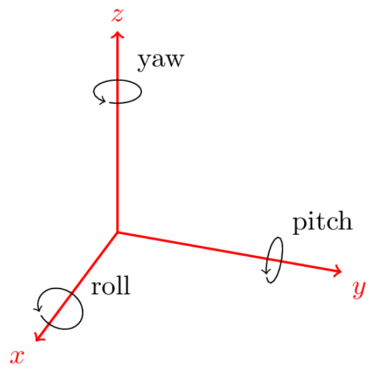



Figure 6.13: Rotations of the camera along one of the axis's ($x$,$y$ or $z$) according to the Cartesian coordinates which correspond to yaw, roll, and pitch rotations (taken from [24]).

placed and rotated the camera and the object at random and then started the movement. Each sequence was then converted to a video frame of one second by Python using "VideoWrite" from "OpenCV" library and fed as input for training the NN.

After we achieved sufficiently good results (validation accuracy 88%, see Figure 6.14 ) we learned that this trained network is able to detect which axis rotation is occurring. However, what still stayed unclear was how to detect the rotation direction (clockwise or counterclockwise along any axis). Note that, for training the network to navigate a drone we must predict rotation speed, not only amount. This is due to the Newton's second law of Physics that states that force is proportional to the square of rotation speed (see [26]).

To focus on these issues, we decided to work deeper around each specific axis. We did the same procedure of creating data as mentioned above. Briefly, by modifying [25], we created several folders related to specific class, each folder contained 400 sub-folders each containing twenty-four images that represent a sequence of camera shots in chronological order. But, this time we used different type of classes, which associated to different rotation camera intensities along **specific axis** (such as fast speed clockwise, medium speed clockwise, fast speed counterclockwise, and medium speed counterclockwise), **not** between rotations of the camera along one of the axis's ($x$,$y$ or $z$) as previous (see Figure 6.15).

We wanted to classify speed of pitch, roll and yaw (or along specific axis, $x$,$y$ or $z$) by dividing each of those speeds to ranges (we used ranges to take each time a **random intensity value from specific range**, Figure 6.16).

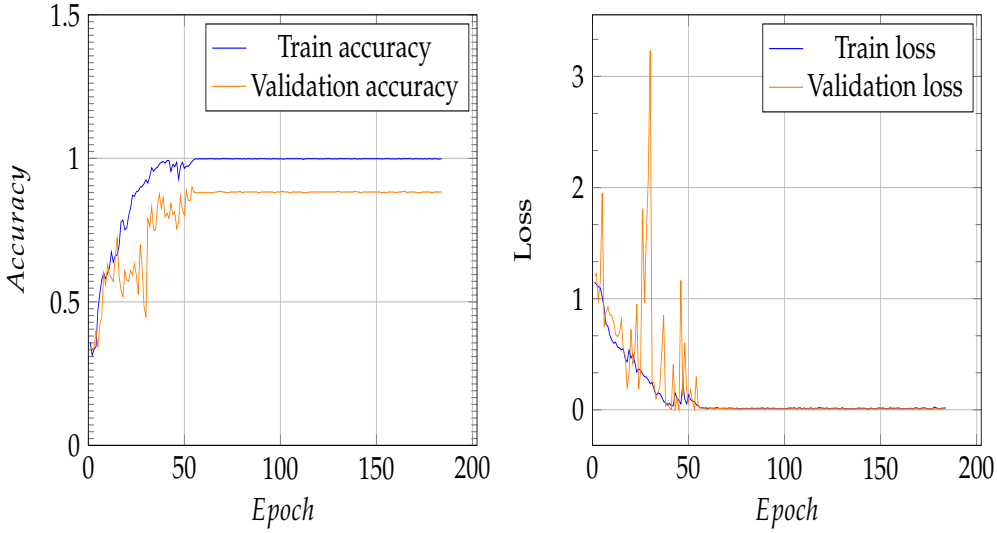



Figure 6.14: Accuracy (left) and loss (right) graphs associated to the classification of the rotating process 'yaw', 'roll' and 'pitch' for videos of corresponding camera rotations. Blue line represents training accuracy/loss and orange line represents validation accuracy/loss.

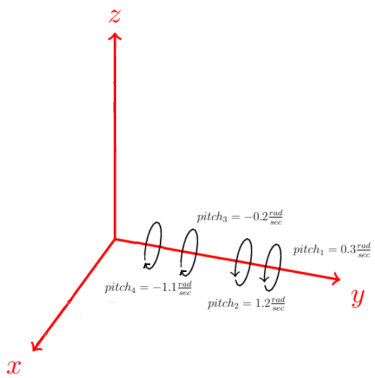



Figure 6.15: Examples of rotations along specific axis, each rotation belongs to different speed range which associate a different class (modified from [24]).

Values for each class were taking different ranges of values which were not overlapping. In addition, between each range there was an interval of values which was not used.

Here, the pitch power means angular speed in degrees per frame.

We achieved validation accuracy close to 100% (see Figure 6.17) on this data-set.

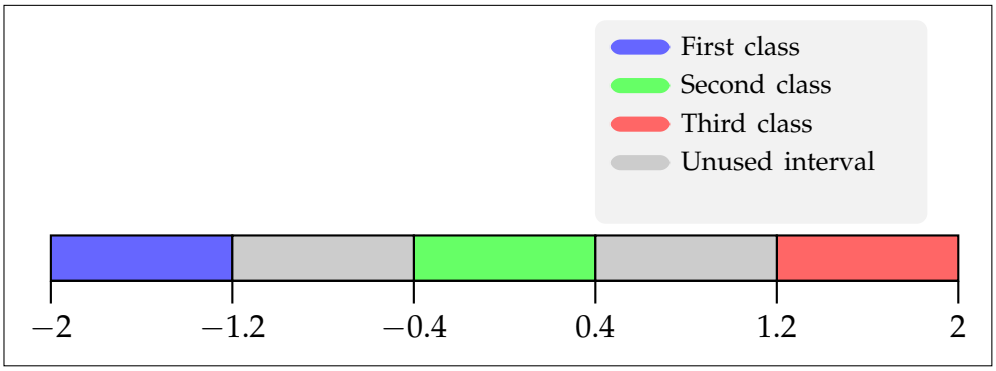



Figure 6.16: Ranges of pitch power values associated to classes. Ranges of blue (e.g., Figure 6.18 ), green and red color present three classes. Ranges of gray color present intervals between those three classes from where values were not taken as input data.

The last step before starting simulations of a drone's flight was to repeat the same with five (not three) power ranges (Figure 6.19) in order to verify that this network is able to classify between more rotation power ranges. Previously, between each range there was an interval value of 0.8 which was not used. This value is comparably big. In order to make classification more difficult, we decided to leave intervals value of only 0.1 between the ranges. We chose three ranges like previously and two additional ranges were chosen between each of three original ranges. Between each of those ranges was a small range of interval value 0.1 which was not used. We used five ranges instead of three, with unused interval 0.1 between them. We achieved validation accuracy of 92% (see Figure 6.20) on this data-set.

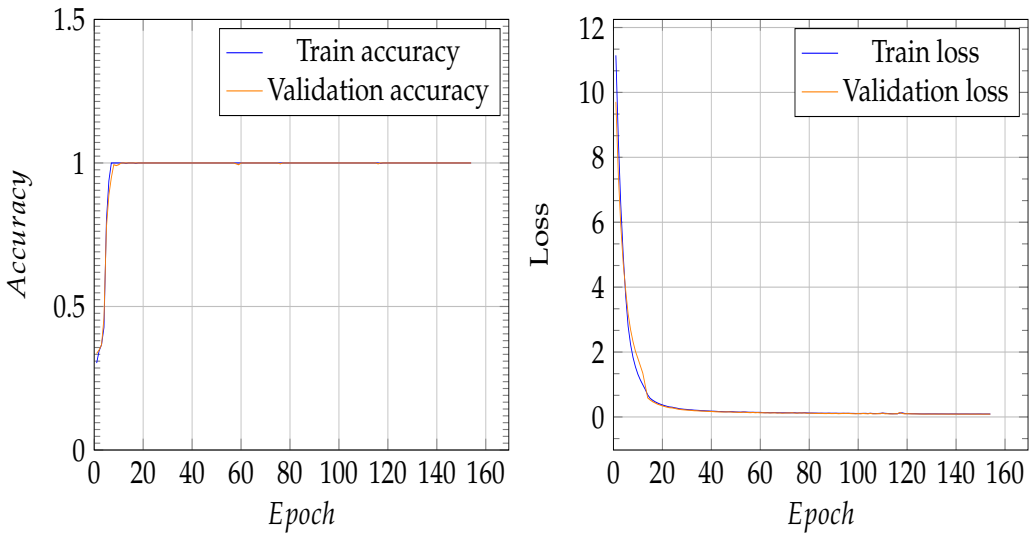



Figure 6.17: Accuracy (left) and loss (right) graphs associated to the classification of the ranges of pitch power values for videos of corresponding camera rotations. Blue line represents training accuracy/loss and orange line represents validation accuracy/loss.

Figure 6.18: Equidistant interval images sampled from the video camera's short rotation period along $x$ axis. For simplification this case shows thirteen images (not twenty as described before). Modified from [25].

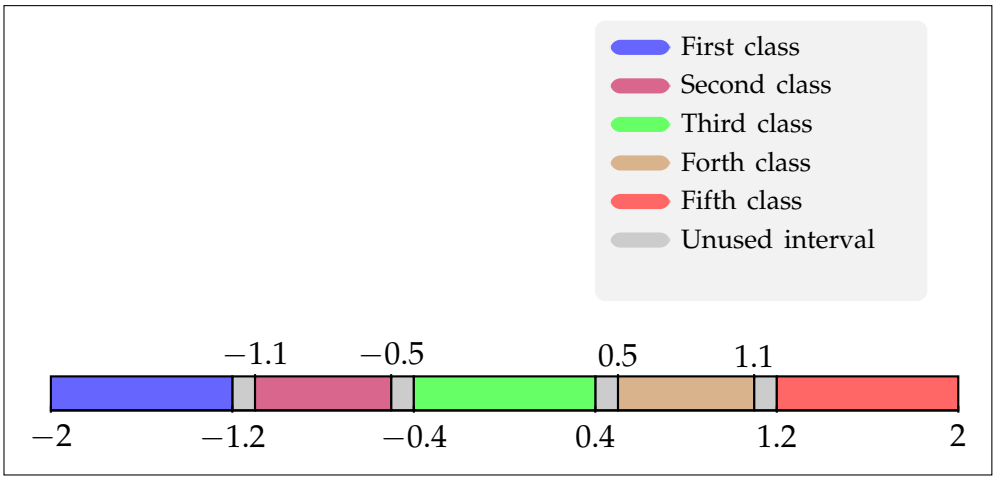



Figure 6.19: Ranges of pitch power values associated to classes. Ranges of blue, purple, green, brown and red color present five classes. Ranges of gray color present intervals between those five classes from where values were not taken as input data.

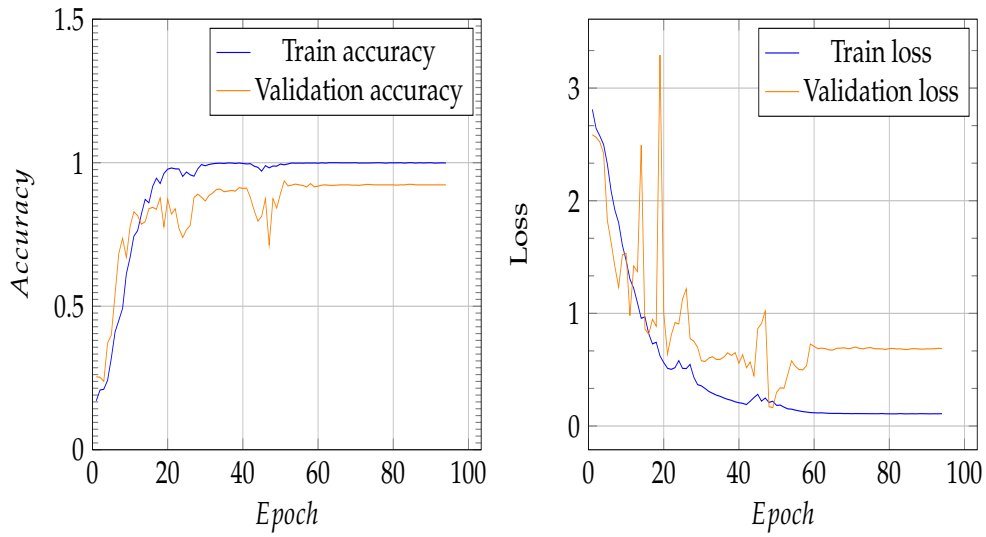



Figure 6.20: Accuracy (left) and loss (right) graphs associated to the classification of the ranges of pitch power values for videos of corresponding camera rotations. Blue line represents training accuracy/loss and orange line represents validation accuracy/loss.

# 7 Training network with the AirSim simulator

In this section we describe our work towards using the network described in Section 4 as an autopilot for a drone. The challenge was to implement a controller for a drone in a virtual environment and test its performance in different conditions. We created a data-set of short videos taken from the front facing camera of the virtual drone annotated them with the command that the pilot needs to give the drone right after the end of the video, trained a network to predict those commands and verified the performance of an autopilot that uses this network to steer the drone.

For simulations, we used Unreal Engine-based AirSim simulator driven by a Python code that connects to the simulator via socket and gives it high level commands that resembles commands given to a drone by a pilot (and, in some cases, also commands to an autopilot like "go to position"). Figure 7.1 shows a screen shot of the simulation environment. As seen in the figure, the simulator generates a realistic picture of a virtual world in addition to simulating the physics of the drone. This is, of course, essential to our work because we focus on vision based autopilot so we need to verify that our network is capable of handling real images with different kinds of obstructions and visual effects.

We collected our data using an autopilot that is built in the simulator. Using the high level command interface in Python, we instructed the drone to fly along a predesignated path. Internally, the simulator that has all the information of all the physical quantities drove the drone as a professional pilot would. We then instrumented the code of the simulator, as described in Chapter 4, to give us the commands that a pilot would need to give to the drone in order to emulate the autopilot behaviour. This gave us the information we needed for labeling the videos that we recorded with the commands that we want the network to generate.

We generated our flights by running a Python code that controlled the simulation and the autopilot. We instructed the drone to fly through a random path by telling it to move from point to point along the path. The autopilot corrected small deviations from the path caused by wind and other effects that the simulator injects. This resulted with flights that look similar to flights that human pilots would generate. We collected images from drone's camera in specific intervals. Each flight started in a different location or under a different yaw angle or in different a environment with a goal to have data from flights in various conditions (to avoid overfitting to a specific environment and/or flight conditions). Data were collected in the way described in the Chapter 4. Each second of each flight session produced a short video file obtained using the Python OpenCV library (by calling the function *VideoWriter*).

These videos where labeled by the command sent to the drone at the end of the second in which they were recorded.

The reason why we decided to obtain an image using the time interval of one second is because similar time interval was used in the study of F. Patrice [16]. In their study, in all video data set the minimal time record was 0.73 sec and it was used in videos associated to the class 'kick ball', while the average time of video length from this class was 1.558 sec. Since we needed to generate output based on short history, we decided to round a time up to 1 sec.

Specifically, since we want from autopilot to generate the pitch, yaw, roll and throttle commands that pilots send to the drone using the remote controller, we needed to train our network to be able to "guess" these commands based on the short video that preceded them. 67% of the collected data where used to train the network and 33% where used for validation.

An example of a video that we have collected can be seen in the Figure 7.2. This example shows how we trained the network to correct a pitch deviation - the drone visibly leans forward by a wind or by some other unintended noise and the command that the drone got after this video was to turn it up in order to fix this deviation.

We first flew the drone in a simple environment. To this end, we ran the simulation without virtual wind disturbances by disabling our modification to the low level code that generates the noise (see Chapter 4).

We first wanted to verify that our neural network is able to discern changes between received data from a drone's camera when commands such as pitch, roll, and yaw are given to a drone. We performed experiment in

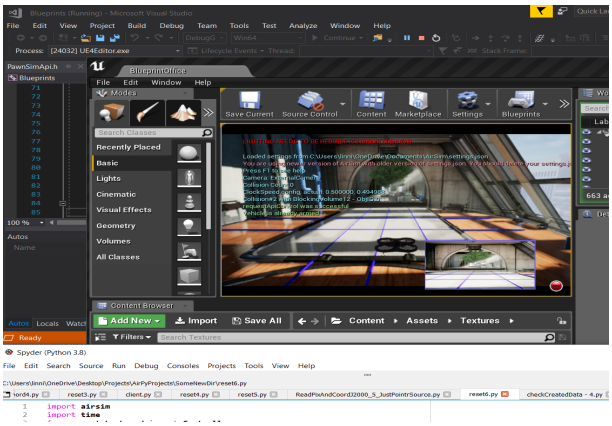



Figure 7.1: Screenshot of drone flying in the simulator AirSim from Unreal Engine program which was generated by running the code related to this in Visual Studio. On the bottom of the figure you can see the Spyder, working environment for Python.

Figure 7.2: Example of pitch action of drone inside random location in the room.

which we gave a command to the drone to move only in one direction. We collected data (for training and for validation) of the drone's flight sessions in the following way. When the drone was hovering over a fixed point on the floor at a specific altitude, we applied pitch command with specific value (while the values of the commands roll and yaw stayed zero). The value of the pitch was taken randomly from a specific range which was associated to it's class, while there was no overlapping between those ranges. We used the same logic as described in Section 6.3. In the same way as done in Section 6.3, between each range there was interval of values which was not used.

For training the network we used the ranges of the pitch power values shown in fig. 7.3.

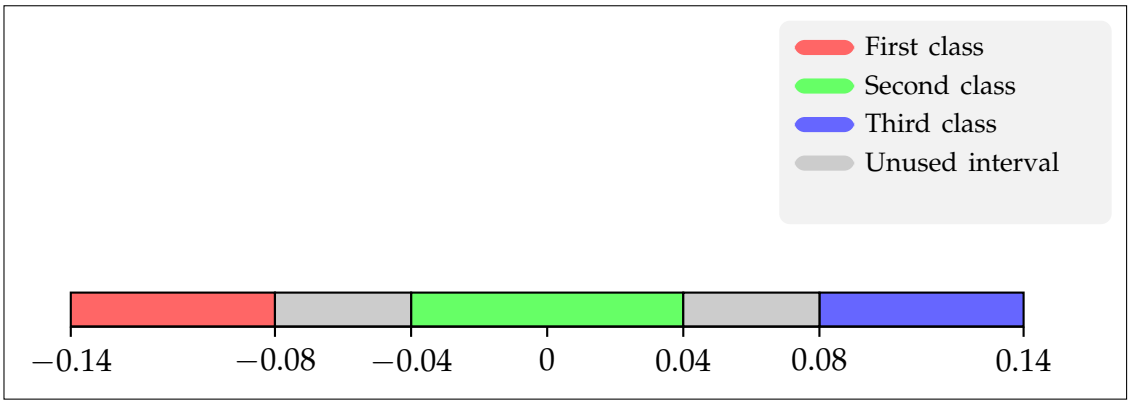



Figure 7.3: Ranges of pitch power values associated to classes. Ranges of blue, green and red color present three classes. Ranges of gray color present intervals between those three classes from where values were not taken as input data.

We collected 200 short videos for each of the classes. We used 67% for training and 33% of the data for validation, as before. In this experiment we managed to achieve validation accuracy of 89% (see Figure 7.4).

Following the same idea we performed individual experiments with roll commands in different ranges (while the values of pitch and yaw remained zero) and yaw command in different ranges (while keeping the values of pitch and roll zero).

In the experiments with roll commands we achieved validation accuracy close to 100% (see Figure 7.5), while in the experiments with yaw commands we achieved validation accuracy of 92% (see Figure 7.6).

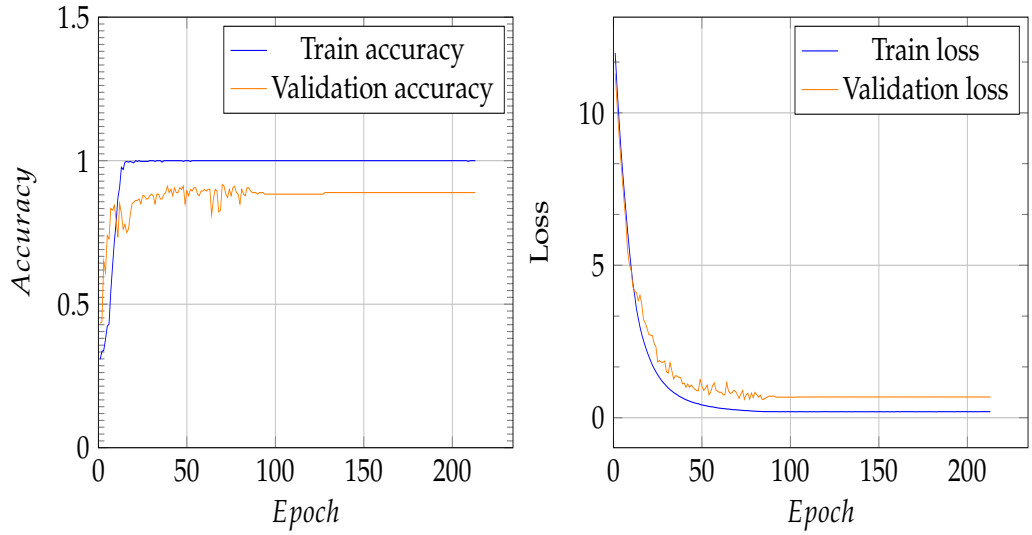



Figure 7.4: Accuracy (left) and loss (right) graphs associated to the classification of the ranges of pitch power values for videos of drone's flight. Blue line represents training accuracy/loss and orange line represents validation accuracy/loss.

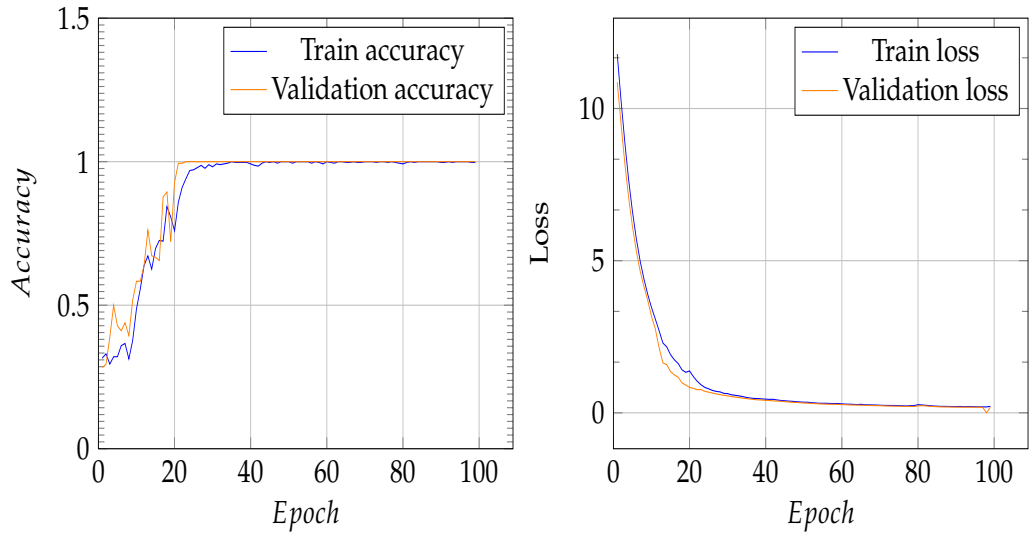



Figure 7.5: Accuracy (left) and loss (right) graphs regarding to the classification of the ranges of roll power values for videos of drone's flight. Blue line represents training accuracy/loss and orange line represents validation accuracy/loss.

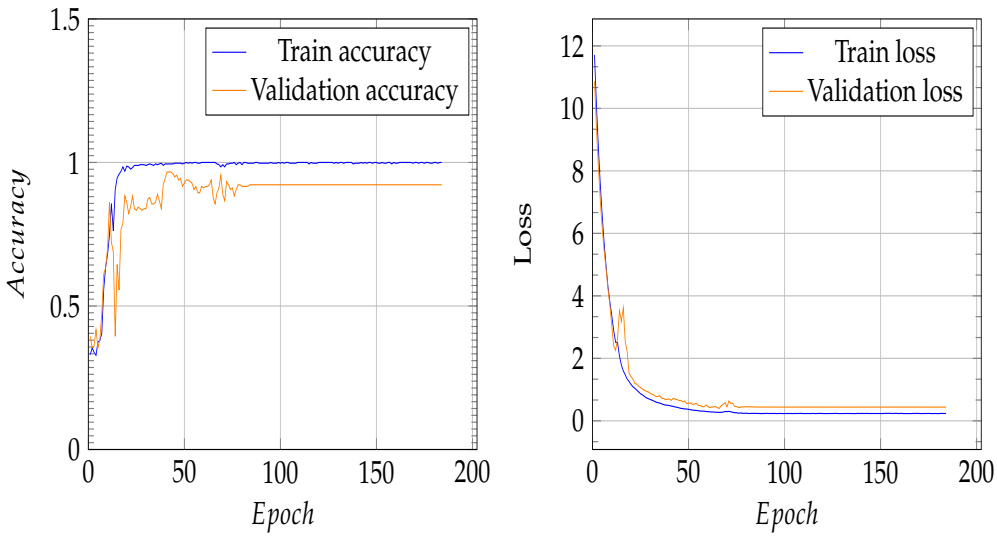



Figure 7.6: Accuracy (left) and loss (right) graphs associated to the classification of the ranges of yaw power values for videos of drone's flight. Blue line represents training accuracy/loss and orange line represents validation accuracy/loss.

From these experiments we learned that it is possible to achieve high quality prediction of drone's motion direction from the sequence of images taken by the drone's camera by combining CNN and RNN networks.

During the experiments, from high quality results we obtained, we started to suspect that something is wrong inside working space or the NN we worked with. We suspected that our results are not in the agreement with reality and that every data we give for training the network (such as data which include only black images for all classes) will give us high quality results which cannot be true. For this we needed to perform sanity check experiment. Up to now we used as data a sequence of images taken by drone's camera during drone's rotation and translational movements. Since existence of differences in subsequent images in a sequence of images obtained by the drone is essential for training GRU/LSTM network (same logic as was done in Section 6.2), we had to confirm that this dynamics exist. In order to do this we performed two experiments (the details will be described soon). Data that we used for both experiments were identical and were splitted in only two classes (not three as done before and described in Figure 7.3) with intention to have simple initial experiments. Pitch value of the first class was taken randomly from the range of negative numbers while the pitch value of the second class was absolute value of the pitch value of the first class.

In the first experiment we converted a sequence of images to a video which served us as an input data.

In the second experiment we repeated the same experiment with a difference that a video input consisted of only one image that was constant throughout the video length. For that image we chose the middle image from a sequence of collected images from the previous experiment.

In the first experiment we achieved validation accuracy of 90% (see Figure 7.7), while in the second validation accuracy was 54% (see Figure 7.8). As we hoped, the second experiment resulted in a value of validation accuracy approximate to the validation accuracy value of a random guessing.

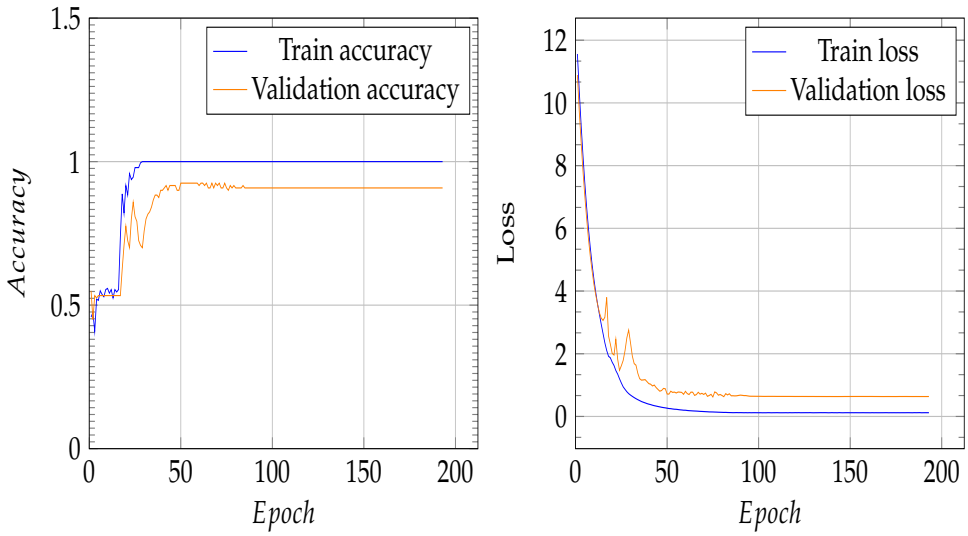



Figure 7.7: Accuracy (left) and loss (right) graphs assiciated to the classification of the ranges of pitch power values for videos of drone's flight. Blue line represents training accuracy/loss and orange line represents validation accuracy/loss.

Training accuracy of the second experiment was close to 100% because the network had learned to predict the class only based on a training dataset (phenomena called overfitting). The validation accuracy was 54% (not 50% as we would expect with random guessing) because of small deviation between average image angle associated to the first class and the second class.

Those experiments confirmed that input data for the NN has to be a sequence of equidistant interval images.

We performed 260 flights from a starting point to a destination point. For

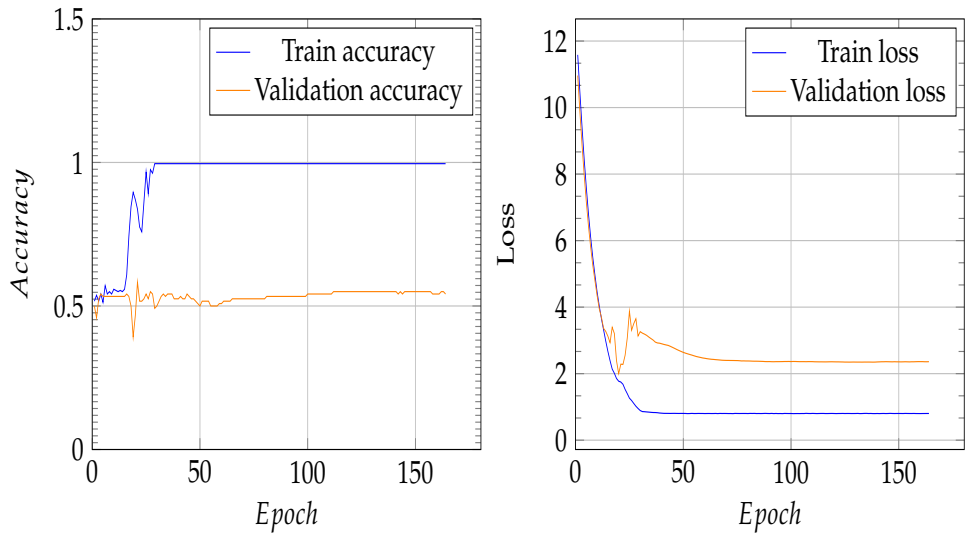



Figure 7.8: Accuracy (left) and loss (right) graphs associated to the classification of the ranges of pitch power values for videos of drone's flight. Blue line represents training accuracy/loss and orange line represents validation accuracy/loss.

each flight a starting point was chosen arbitrarily withing small range around a base point while the destination point remained constant. Each flight session was recorded. This generated image data and a text file which provided for each image its pitch, roll, yaw and throttle value which we used as a label (see Chapter 4). We created data and trained a network for predicting the appropriate pitch value at each point along the way. We worked with pitch values in the range of -0.03 to 0.03. We divided this range into 11 equidistant points, each point served as a specific class. Each value with the label pitch power was rounded to the closest point. The five images prior to the same label were used as an input data. The reason why we used five images and not two is to increase accuracy of our predictions. Using only two images could be efficient in a case of linear system. However, since our system is not linear, as it includes also rotation of a drone.

This data was used for training. Afterwards, we tested how our network performs as an autopilot for a drone. The drone flew with correct pitch angles but without stopping at the destination point. This lead us to the following hybrid control strategy: to use a separate network for flying and for landing and decide when to switch between the two using another network. We then started to train a network to find the point where the drone

will start to perform the stop process that we would call 'stopping radius' from the destination point. For each image during the flight we performed classification according to whether the drone is within the stopping radius or not.

We structured out a strategy for detecting stopping radius that we named Detect Radius Based CNN solution (DRBCNN solution) in the following way. Each drone's location was associated with a specific class, with the starting point being the first class and the destination point being the last class. All class numbers were organized in a way that number of a class was directly proportional to a distance from the starting point. When the drone was getting closer to the destination point, the number of the class gradually increased. For classifying drone's location based on input image we modified our previous NN which was combination of CNN and RNN by deleting the lines in code where "TimeDistributed" and "GRU" layers were added, which converted our previous network to pure CNN, which made classification from only single input image (network number 1). We used a very similar model because it provided us good results in previous experiments. For collecting data we performed 20 flights from starting point to destination point while each time starting point was chosen arbitrarily withing a small range. We used a total 50 of classes. We modified the function "angle_error" from [27] in order to use it as a metric (absolute error) for training the network (instead of "accuracy" metric). We did it in order to make the error value dependent on the interval between correct and predicted number of the class. Following the training, we achieved validation error of 2.0468 which is approximately 4%. According to this we used our network for drone's flight. We defined a class within a range of numbers and every number beyond this range was defined as a stopping radius.

As a sanity check, we tried to detect stopping radius in a naive way, Detect Radius Based Pixel solution (DRBP solution), in which the drone flew in a diagonal direction and in which in each iteration the drone performed pitch and roll commands and image was taken by drone's camera. According to the image we checked whether the color of a pixel located in the middle of the left edge of the image was red or not (see Figure 7.9). The red color in this pixel indicated the stopping radius. For this purpose we added a small red wall in the simulator's environment (see Figure 7.10, on the left side of the fourth image).

To further validate our solution, we also compared it to another naive method, Detect Radius Based Time solution (DRBT solution), in which the

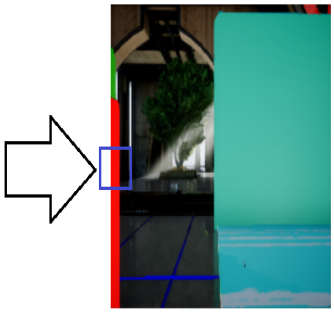


Figure 7.9: The pixel located in the middle of the left edge of the image which indicates a stopping radius.

drone stops withing expected radius by measuring elapsed time from the moment the drone start its trajectory. We performed five experiments and measured the average time of $t_1$=116.42 seconds, which took the drone to pass distance from the starting point A to the point which was at the closest distance from the destination point B (see Figure 7.11).

Our proposed solution was better than the two aforementioned naive solutions because of the following reasons:

- With the DRBP solution there is possibility that the drone can wrongly detect the stopping radius before it reaches there or not detect the stopping radius at all because of the environmental factors such as sun reflections. The DRBT solution rely only on elapsed time from the moment the drone started to move. If we move drone's starting point for a little from the base starting point, the drone can stop in a wrong location. The uniqueness of the DRBCNN solution was that we used NN which learned the environment appropriately, which enabled us to define where to stop the drone in adaptive way.

- For detecting stopping radius we performed three experiments. In each experiment drone flew in the same direction from starting point A to direction of destination point B with small deviation (see Figure 7.11), each experiment associated to specific method. In the first experiment we used the DRBCNN solution. In the second and third we used the DRBP and the DRBT solutions respectively. In the third experiment the drone started to move near the starting point A. The experiment results can be seen in the Table 7.1. We can see that by using the DRBCNN solution we were able to detect closest distance

| Solution | DRBCNN | DRBP | DRBT |
|---|---|---|---|
| Detected distance | 4.3542 m | 6.26332 m | 4.8675 m |

Table 7.1: Comparison between three different approaches DRBCNN, DRBP and DRBT for detection the distance from a destination point.

from the destination point (point B), because this solution gives us more freedom to decide when we want to command the drone to stop.

After we managed to solve how to detect where a drone needs to stop, our next goal was to make a drone bypass an obstacle during its flight. We did it in the following way. A drone was supposed to fly from staring point A to destination point C, but there was an obstacle between those points. In this case we decided to make the drone fly through a mid point B which was located on the right side from the obstacle (see Figures 7.10 and 7.11).

For NN based controller solution, in the beginning we collected data of flights from point A to point B. Since in this case the flight was not a straight line but a more complex path, we needed to take into account pitch and roll commands. We trained one neural network for pitch commands and another network for roll commands (networks number 2 and 3 respectively). After that we collected data of flights from point B to point C. As before, we trained another neural network for pitch commands and another network for roll commands for this part of the path (networks number 4 and 5 respectively).

After training those network (five neural networks in total), we performed experiments on a simulator (NN based controller solution). All the experiments started by hovering at point A. After that, the drone performed pitch and roll commands in each iteration using the two neural networks (networks number 2 and 3) until it reached the distance radius from the point B (which was recognized by the network number 1) and then networks number 4 and 5 were used for another pitch and roll commands. The resulting drone's route is presented in the Figure 7.12. There is deviation between drone's planned route and an actual route. One of the reasons why the drone finished a flight with comparably big deviation from destination point C (10 meters) is that the data that we used for training neural network are based on straight-line flights (without drone passing through another intermediate point, with only little deviation from the straight line which connects points A and B or line which connects points B and C). The second reason is that in our trained neural networks num-

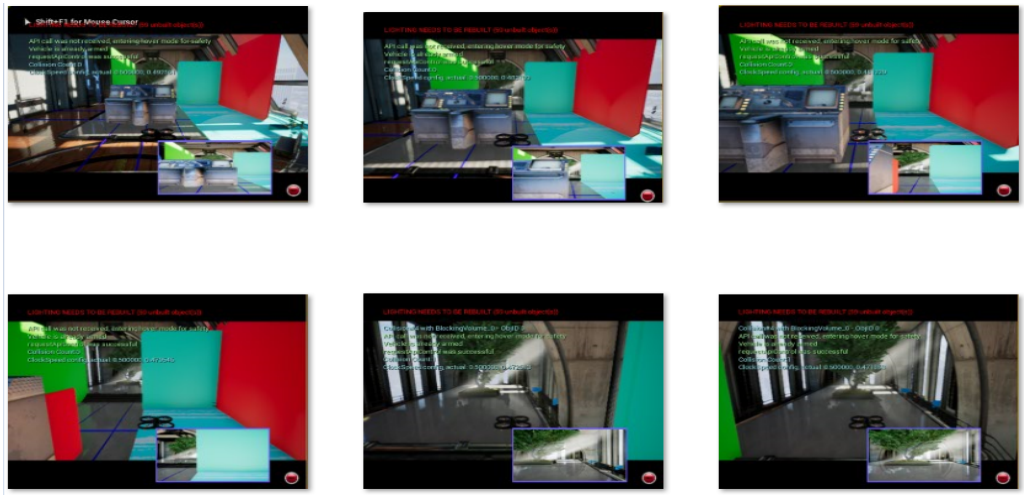


Figure 7.10: A sample of images showing how the drone is avoiding obstacles by taking a detour from the right side.

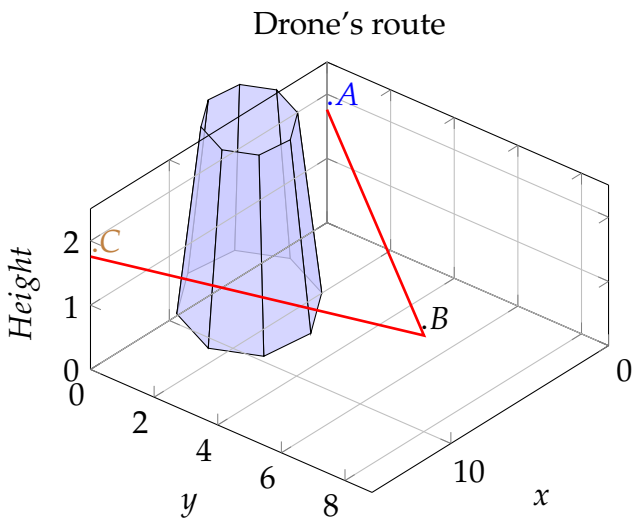



Figure 7.11: A 3D diagram showing the drone's route progresses according to original plan by taking a detour around an obstacle.

ber 2,3,4 and 5 validation accuracy was not sufficient high. Finally, there is the effect of the momentum while the drone was flying from point A to point B. That said, when the stopping radius was detected, the networks number 4 and 5 took control over drone's flight towards the point C (with effect of momentum from previous flight direction).

The good news is that the drone managed to overpass the obstacles using our trained networks.

As before, we performed a sanity check by trying to overpass the obstacle in a naive way, Random Sample Commands Based Time solution (RSCBT solution). We chose two specific folders of records of flights. The first folder contained information about the flight from point A to point B and the second information about the flight from point B to point C. Each folder contained sub-folder of images during the flight and label text file which contained the name of all image files from sub-folder. Each image file contained details about drone's average of 100 prior commands (pitch, roll) that were undertaken by the drone in that very moment (described in Chapter 4). We sampled specific pitch and roll commands ($pitch_1$ and $roll_1$) when the drone flew from point A to point B and specific pitch and roll commands ($pitch_2$ and $roll_2$) when the drone flew from point B to point C. According to data from those folders we checked time of flight from point A to point B ($t_2$=32 seconds) and time of flight from point B to point C ($t_3$=20 seconds). By knowing those times and using naive way of stopping radius detection which relies on elapsed time (in our case this is $t_2$=32 seconds) we performed the experiment. The drone started to fly from point A, each iteration the drone performed $pitch_1$ and $roll_1$ command until $t_2$=32 seconds had elapsed and then the drone performed commands $pitch_2$ and $roll_2$. The flight route is presented in the Figure 7.12. As we can see, the drone didn't hit the obstacle and it reached the distance of 2 meters from point C which is a best result up to now.

To further validate our solution we compared our solution to another naive method, which we named Commands in Proportion to Elapsed Time Solution (CPET solution). We took the same folders from previous solution. Next, we used label text files to save all pitch and roll commands when the drone flew from point A to point B (in $list_1$) and all pitch and roll commands when the drone flew from point B to point C (in $list_2$). The commands in $list_1$ and $list_2$ were saved in chronological order. We performed experiment by making the drone fly from point A to point B until $t_2$ seconds had elapsed. Each time drone performed pitch and roll commands from one of the indexes from the $list_1$, each time the index was chosen

in a proportion to elapsed time according to the array length. After $t_2$ had passed the drone flew from point B to point C until $t_3$ seconds had elapsed. Each time drone performed pitch and roll commands from one of the indexes from the $list_2$, each time the index was chosen in proportion to elapsed time according to the array length. During this experiment the drone finished in the close proximity from the starting point before even reaching the obstacle (see Figure 7.12).

In addition, we validated our solution by comparing our results to another naive method, Random Sample Commands Based Pixel Color solution (RSCBPC solution). This solution is the same as RSCBT solution (we chose this because it performed flight in a best way up to now), but instead of stopping after specific elapsed time, the drone stopped when image taken from the camera detected red pixel in the middle of the left edge of the image and then each time a drone performed $pitch_2$ and $roll_2$ commands.

According to the results (see Figure 7.12), controlling the drone using the RSCBT and NN based controller solutions, drone eventually reached the distance of 2 and 10 meters respectably from point C, which imply that the RSCBT is better than NN based controller solution. But what happens if we perform the same round of experiments but the drone will start from a little different initial location?

We performed the second round of experiments which were the same as before but this time we placed the drone at the distance of one meters left and three meters behind of the initial location. The route of flight in each of the experiments is presented in the Figure 7.13. When using the RSCBT solution the drone hit the obstacle. With the CPET solution we can see that the drone finished a flight before it even reached the obstacle. Using the RSCBPC solution the drone flew too far. Only in the experiment in which NN based controller solution was used, the drone managed to overpass the obstacle and reach the closest distance from the point C.

We also performed the third round of experiments in which all conditions were the same as in the first round with a difference that the starting point of a drone was chosen to be three meters in front of the initially chosen location. Each experiment flight route is presented in the Figure 7.14. By performing the RSCBT solution the drone reached the distance of five meters from point C (the best result from this round of the experiments). From CPET solution path we can see that drone switched flight direction before it even reached the obstacle. Using RSCBPC solution the drone reached the farthest distance from the point C.

As a conclusion, in all three rounds of experiments when the CPET solution was used, the drone finished flight before even reaching the obstacle. In addition, in all experiments using our NN based controller solution generated better results than using the RSCBPC solution (by approaching closer to the point C). Performance of the CPET solution depends on the initial drone's location because it takes into account elapsed time from the flight' start, as we saw in the second round of experiments when the drone hit the obstacle. Our solution was less dependent on the initial location because it is based on trained neural network which studied environment properly and performed action based on data from the camera.

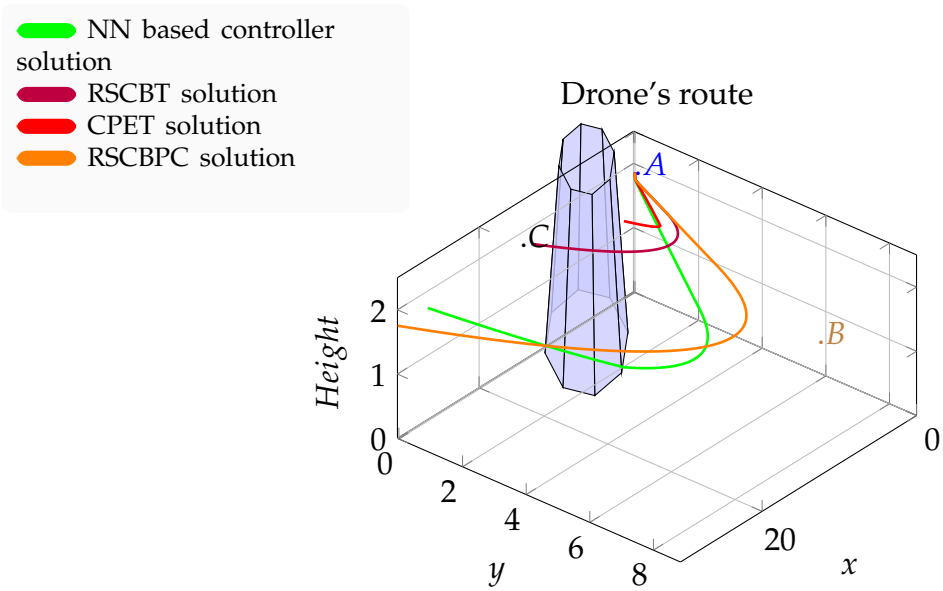



Figure 7.12: A 3D diagram showing the drone's progresses routes according to different solutions in first round of experiments. Initially the drone took off from a point $(0, 0)$.

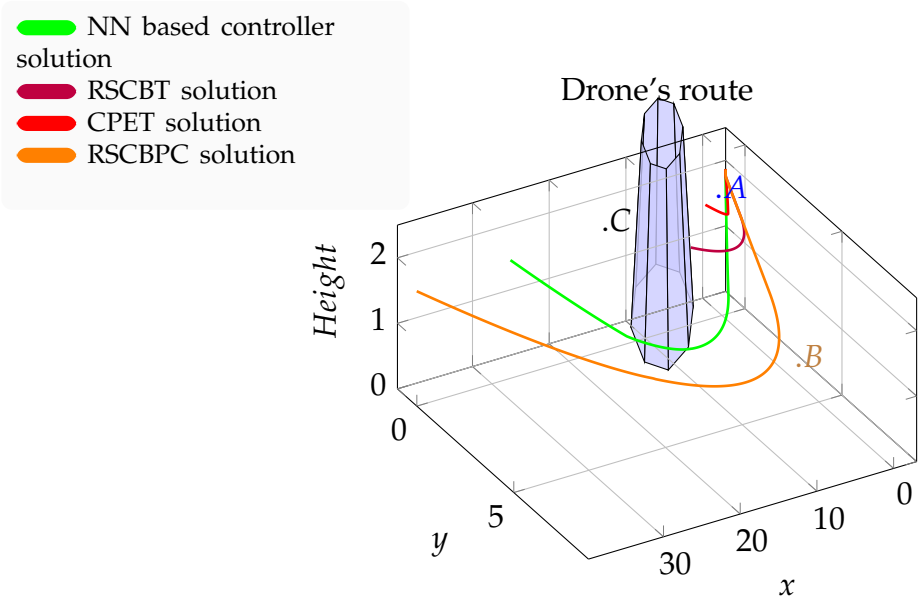



Figure 7.13: A 3D diagram showing the drone's progresses routes according to different solutions in second round of experiments. Initially the drone took off from a point $(-3, -1)$.

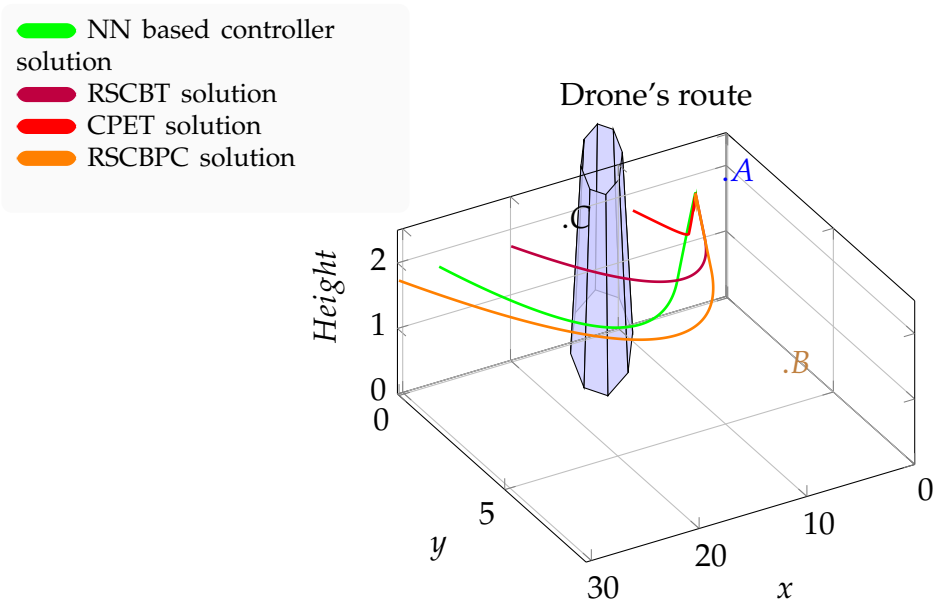



Figure 7.14: A 3D diagram showing the drone's progresses routes according to different solutions in third round of experiments. Initially the drone took off from a point $(3, 0)$.

# 8 Conclusion and future work

In this thesis, we developed a new way for controlling drones along specific route using DL with combination of RNN and CNN. We trained those networks on large amount of drone flight's data that we collected.

During the drone's flight along the route each time drone performs command based on short previous history of image data from drone's camera. In order to define which specific commands needed to be assigned we trained NN.

As a future work we plan to use also sensors such as gyroscope and accelerometer in order to improve drone's performance accuracy during flight. In addition, we plan to add environmental factors such as wind to our simulations in order to simulate flights in conditions that are more realistic. In this thesis during the actual drone's flight we used only pitch and roll commands. In the future work we plan to introduce also yaw and throttle commands. Here we combined five neural networks for over-passing the obstacle. In the next work we plan to use only single neural network for this purpose. In addition, final location reached by the drone in this thesis deviated in 10 meters from the destination. In the next work we will reduce deviation from the destination point. Finally, our ultimate goal in the future is to apply trained neural network for controlling real drone in real environment.

# Bibliography


[1] H. Efraim, G. Weiss, A. Roth, and A. Shapiro, "Vision based output feedback control of micro aerial vehicles in indoor environments", J Intell Robot Syst, vol. 87, pp. 169–186, 2017.

[2] M. Zhua, X. Wanga, and Y. Wangc, "Human-like autonomous car-following model with deep reinforcement learning", Transportation Research, vol. 97, pp. 348–368, 2018.

[3] https://medium.com/smileinnovation/how-to-work-with-time-distributed-data-in-a-neural-network-b8b39aa4ce00.

[4] https://dronequadcopterx.blogspot.com/2018/11/quadcopter-frame-yaw.html.

[5] https://www.123rf.com/photo_85980645_stock-vector-remote-controlled-aerial-drone-quadcopter-drone-with-camera-for-photography-or-video-contemporary-un.html.

[6] H. Goldman and G. Weiss, "Reactive scheduling of computational resources in control systems", Ben-Gurion University of the Negev, Beersheba, Israel, 2017.

[7] P. Ram Prasad, V. Sachin, A. Shahzad, C. Suman Kumar, and K. S. Pankaj, "Deep neural network for autonomous uav navigation in indoor corridor environments", Procedia Computer Science, vol. 133, pp. 643—650, 2018.

[8] P. Ram Prasad, V. Sachin, A. Shahzad, B. Sambit, and K. S. Pankaj, "Localization of unmanned aerial vehicles in corridor environments using deep learning", arXiv:1903.09021, 2019.

[9] A. Garcia, M. Sandeep S., E. Kiewra, and K. Ghose, "A convolutional neural network feature detection approach to autonomous quadrotor indoor navigation", in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, 11/2019.

[10] K. Amer, M. Samy, M. Shaker, and M. Elhelw, "Deep convolutional neural network-based autonomous drone navigation", Giza, Egypt, Nile University, 2019.

[11] B. G. Maciel-Pearson, S. Akçay, A. Atapour-Abarghouei, C. Holder, and T. Breckon, "Multi-task regression-based learning for autonomous unmanned aerial vehicle flight control within unstructured outdoor environments", arXiv:1907.0832, 2019.

[12] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles", in *Field and Service Robotics*, arXiv:1705.05065, 2017. `https://arxiv.org/abs/1705.05065`.

[13] T. Sweeney, *Unreal Engine*, 1998. `https://www.unrealengine.com/`.

[14] Microsoft, *AirSim repository*, 2018. `https://github.com/microsoft/AirSim`.

[15] F. Chollet, *Keras*, 2015. `https://keras.io/`.

[16] F. Patrice, *Training a neural network with an image sequence - example with a video as input*, 2019. `https://medium.com/smileinnovation/training-neural-network-with-image-sequence-an-example-with-video-as-input-c3407f7a0b0f`.

[17] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems", 2015. Software available from `https://www.tensorflow.org`.

[18] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serret, *HMDB: a large human motion database*, 2011. `https://serre-`

lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/#Downloads.

[19] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications", J Intell Robot Syst, arXiv:1704.04861, 2017.

[20] K. Shen, *Effect of batch size on training dynamics*, 2018. https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e.

[21] E. Hoffer, I. Hubara, and D. Soudry, "Train longer, generalize better: closing the generalization gap in large batch training of neural networks", Technion - Israel Institute of Technology, Haifa, Israel, 2017.

[22] A. Oppermann, *Regularization in Deep Learning — L1, L2, and Dropout*, 2020. https://towardsdatascience.com/regularization-in-deep-learning-l1-l2-and-dropout-377e75acc036.

[23] A. Nagpal, *L1 and L2 Regularization Methods*, 2017. https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c.

[24] https://math.stackexchange.com/questions/3171374/3d-axis-rotation-new-value.

[25] T. Skillman, P. Gaunt, and T. Ritchford, *Introduction to pi3d*, 2012-2019. https://pi3d.github.io/html/ReadMe.html.

[26] P. Jaiswal, *Demystifying Drone Dynamics!*, 2018. https://towardsdatascience.com/demystifying-drone-dynamics-ee98b1ba882f.

[27] M. Yousry, *Correcting Image Orientation Using Convolutional Neural Networks: Adding final operationalisation touch*, 2019. https://medium.com/@ylashin/correcting-image-orientation-using-convolutional-neural-networks-bf0f7be3a762.

# Entrepreneurship Pitch Track chaired by Prof. Ehud Gudes and Yonah Alexandre Bronstein

**Entrepreneurship Pitch Track chaired by Prof. Ehud Gudes and Dr. Yonah Alexandre Bronstein**

The Hi-Tech industry and the state-of-the-art research are becoming closer partners, as the implementation of research results becomes less sophisticated and requires less manpower. The goal of the CSCML pitch track is to expose the researchers to the world of entrepreneurs and vice versa for the sake of creating mutual value and advancing the economy and the society. Nine startups pitched this year in varied areas like cryptanalysis, IoT, Blockchain, visual analytics, and even garbage-to-energy conversion, polymer crystallization and virtual-to-real pet dog caring! It was heartening, this year as well as in past years, to note that, even in a business focused track, there were entries that could justifiably be considered "for the greater good of the people" – that is, even if they had business motives and priorities, they would still end up benefiting all of us. These entrepreneurs deserve all the encouragement that we in the community can give them, in whatever form is suitable. As was the case last years, the Entrepreneurship Pitch Track at CSCML 2021 did an excellent job of fulfilling this objective and consequently was a great success. It received endorsement from leading VCs (Ford research center, Incubit, Disruptive AI, lool Ventures…) and corporations (IBM, Microsoft, Checkpoint …).

Out of the nine start-ups pitched in the event, three were selected: Itay Katzav's "TREAT" was selected by the Entrepreneurship Pitch Track Committee and the audience as the leading entry and won the $500 prize. TREAT is for people who want a pet (or more pets), but due to their partner, landlord, time or cost reasons they can't. Based on Deepfake, TREAT gives those people the option to own a real pet and to make a real impact on its life. nSure.ai was presented by Alex Zeltcer, nSure.ai is a fraud prevention platform offering chargeback guarantee to merchants selling digital goods (digital giftcards, games, tickets, etc.) online, received second place. And "Analytics-Model.com" presented by Idan Moradov, which is an AI model that runs continuously and autonomously across every possible metric and data source, giving you real-time control over what's happening and saying goodbye to business' blind spots. Their model studies the data's normal behavior and finds anomalies instantly, understanding every signal type and seasonal patterns, placed third.

Looking forward to an even better CSCML 2022.

Regards,

Entrepreneurship Track Chairs

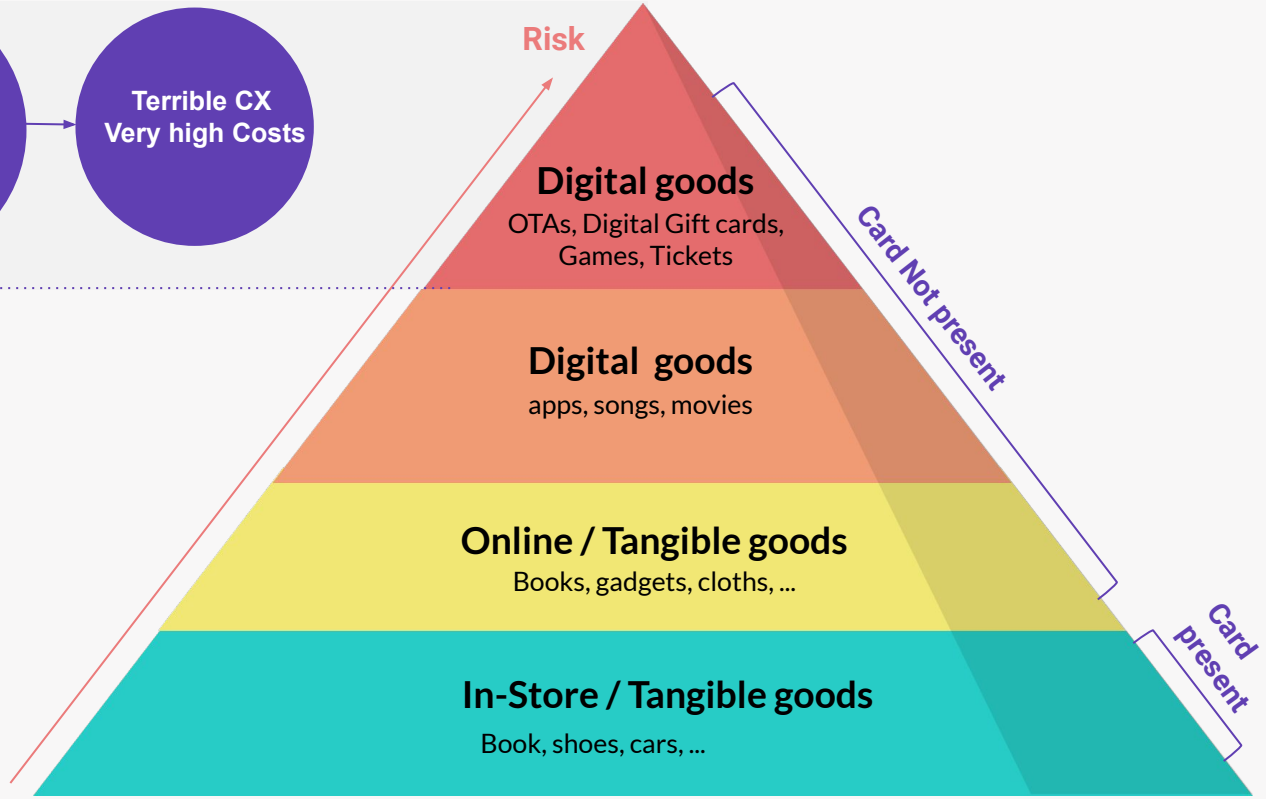
nSure.ai

# Sell **Digital Goods** with **Confidence**

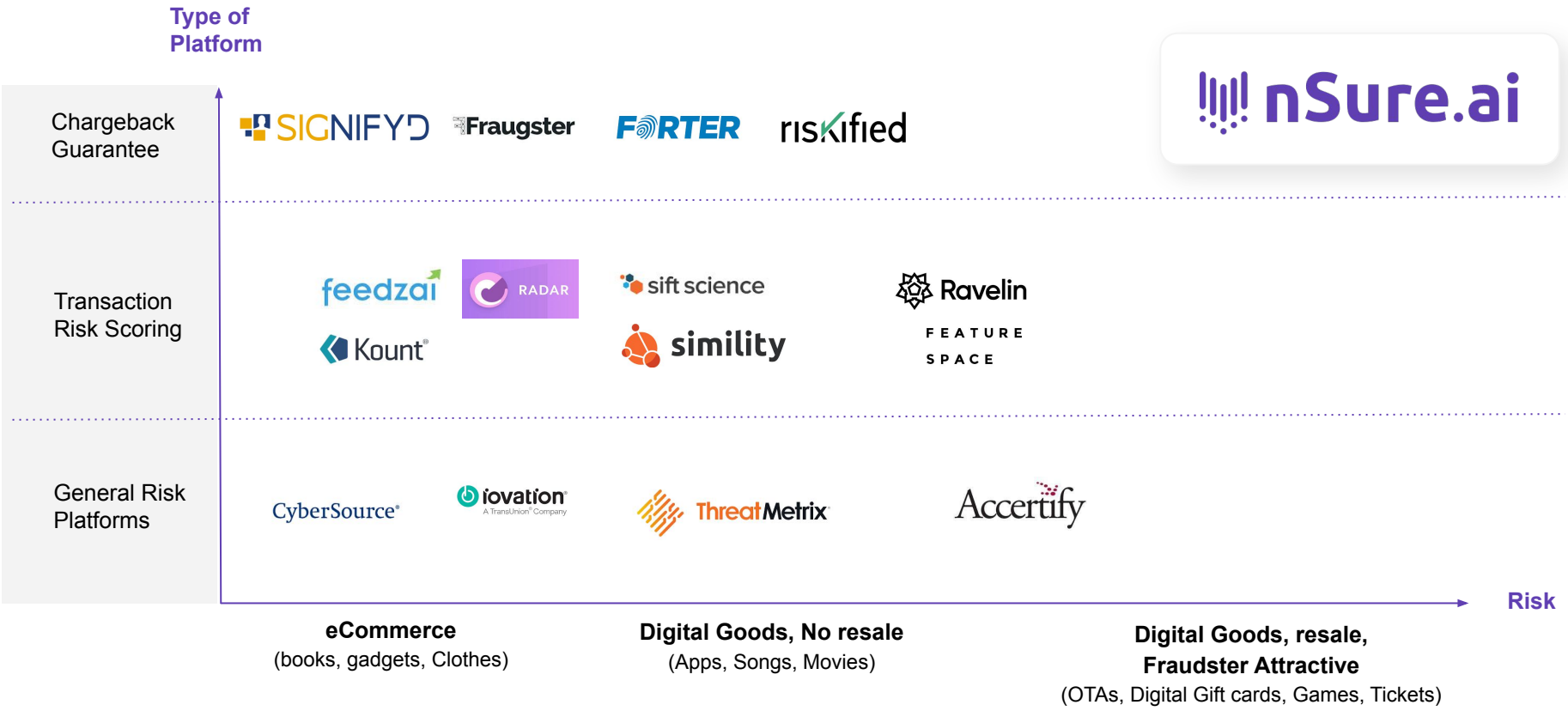
www.nsureai.com

# Digital Goods - Hardest to sell, impossible to grow

High Fraud Pressure → Excessive Declines Manual Reviews → Terrible CX Very high Costs



# World's first Chargeback Guarantee for Scalable Fraud



nSure.ai **envisions** that:

**Digital Goods will be sold like any other product**
by disrupting
the multidimensional inefficiency

# Huge Opportunity, Disruptive Technology

**$10B++**

Total
Addressable Market

nSure.ai

**80%**

Lower costs

**90%**

CX Improvement
(Less False decline)

# The Opportunity

# Digital Goods - Surprisingly Large and Fast Growing

TOTAL ONLINE TRANSACTION VOLUME $4.4T

TOTAL DIGITAL GOODS VOLUME ~$1T

CAGR 17%

TOTAL ONLINE TRANSACTION VOLUME $4.4T

TOTAL DIGITAL GOODS VOLUME ~$1T

# Digital Goods

**~ $600B**
Travel (OTAs) + Airlines

**~ $150B**
Digital Gift Cards

**~ $100B**
Games

**~ $60B**
Event Tickets

Consumer Fintech, Wallets, Neo Banks, BNPL

*Physical goods delivered in under an hour

# The Tech & Edge

# Key Differentiators

Reinsurance / Risk management framework

Data

Transaction Data

Behavioural Data

Data Enrichment

Data Processing & Decision

nSure.ai
Risk Model

**Feedback to Model** - We learn from **Every decision**

| | | | |
|---|---|---|---|
| Early 2017 | 84% Approved | 15% SoftApproval ™ | 1% StingBack ™ |
| Mid 2018 | 98% | 1.5% | 0.5% |

## Experts of our domain, we know what data to select and how to make the most of it

Our dedication virtual goods and our Knowledge of this domain starting @ zeek with a total of **6 years experience of fighting fraud** in virtual goods

## SoftApproval™ for suspected transactions

SoftApproval™ provides for Customer interaction and intent detection, post transaction thus allowing for a higher approval rate and an accurate feedback loop

## StingBack™ for Reducing Fraud Pressure

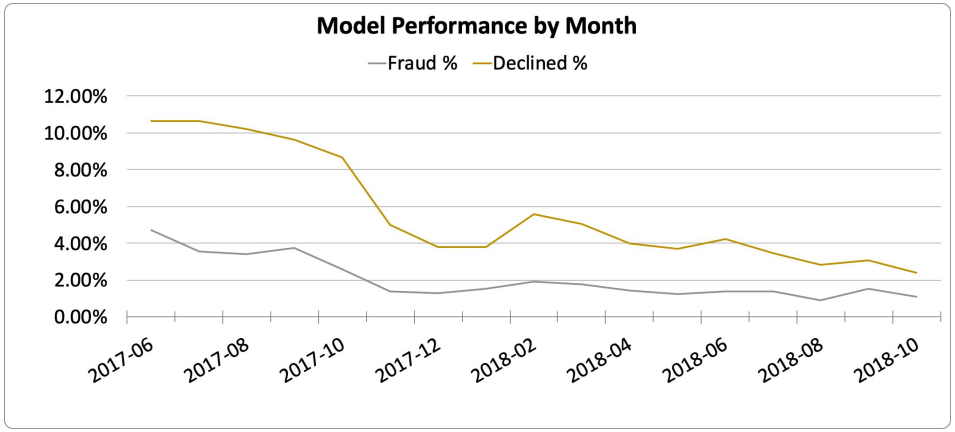
Without fighting fraud pressure actively, gray area remains large and “noisy”. StingBack™ de-centivises fraudsters from ever returning

# nSure.ai - Performance Superiority

- Benchmarking Performance (nSure.ai vs. leading Competitor):
  - Performance measured by balancing Chargeback vs. Excessive Declines
  - When Competitor tuned to Chargeback rate of 0.5% (nSure/s performance) - Excessive declines go up to 9%-10%
  - nSure's performance reduces this decline rate by 90%! To 0.88%

| | Competitor | nSure.ai |
|---|---|---|
| Approved (correctly) - True Negative | 95.72% | 97.91% |
| Excessive Declines - False Positive | 2.71% | 0.88% |
| Chargeback - False Negative | 0.81% | 0.51% |
| Declined (correctly) - True Positive | 0.79% | 0.71% |

- Reducing Fraud Pressure
  - Took 1.5 years fighting fraud pressure to come down from 5% fraud to 1% allowing declines to go down from 11% to 2%



# The Company & Team

# Company Overview

## Technology

- In production for the last 6 years demonstrating improved results month over month, securing over $100M per year with hundreds of retailers
- Patent Pending SoftApproval™ processes

## Investors

- Seed round by Kamet / AXA Insurance
- Extended to $6.8M in total by DisruptiveAI, Phoenix, Moneta and private investors

## nSure.ai

- Established March 2019 with the goal to disrupt the online fraud prevention market
- Starting with the most fraudster attractive products, nSure aims to reduce fraud costs by 90%

# Team

## Alex Zeltcer - CEO

- Former GM @ Zeek
- Former CEO @ Artizone, an online retailer and farmer's market selling to communities in Dallas, Chicago and Denver
- Former CEO @ SmarTeam (a Dassault Systemes company) – driving sales from $30M to $100M across a range of industries

## Ziv Isaiah - CTO

- Former Founder & CTO @ Zeek – largest European Gift Card Mall; Growing Digital gift cards to over $100M/year within 3 years
- Former CTO @ Ginger – a Pioneer of AI Based Natural Language Understanding acquired by Intel

## Directors

Gadi Tirosh (DisruptiveAI), formerly at JVP, chairman @ CyberArk
Noam Inbar (Phoenix), formerly VP of Strategy @ Forter
Michael Niddam (Kamet) - formerly at BCG

## Jonathan Weiner - Advisor

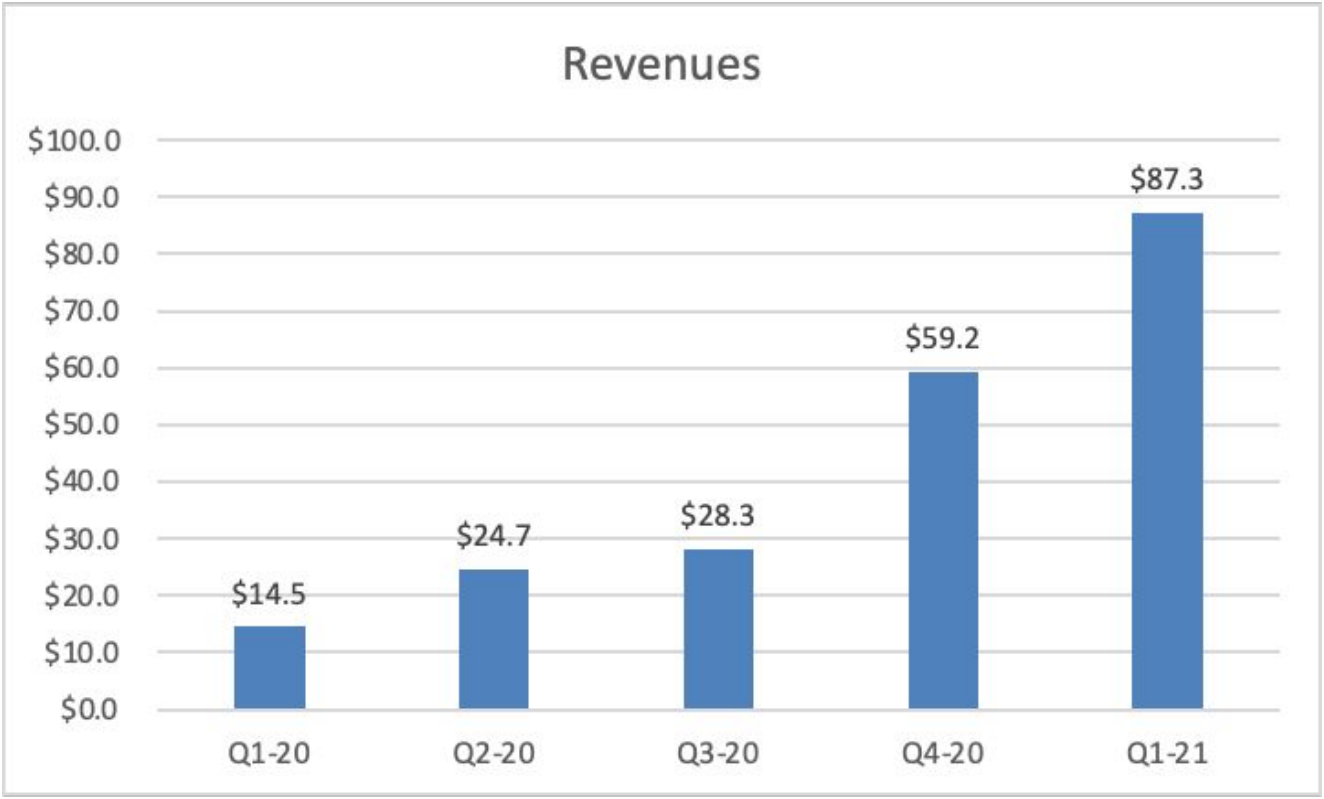
Jonthan is founder of Money 20/20 and a veteran of the prepaid world. Founded and sold TXVia to Google, becoming Gogle's wallet. Currently a partner at Oak HC/FT

## AI team

An A Team of engineers that lives and breathes Fraud protection models, most together for over 5 years

# Growth to date

# nSure.ai - Actual Performance to date



nSure.ai

# Thank you!

**Contact** Alex Zeltcer
**Phone** +972-54-6638665
**Email** Alex@nsure.ai

www.nsureai.com

Google for Startups
Campus Founders Community

# TREAT

## Own & impact a REAL pet based on AI

NEED

### WANT A PET, BUT CAN'T

Due to: Landlord, partner, already have a pet, can't commit, costly, etc.

SOLUTION

### HAVE A PET THE WAY IT FITS YOU

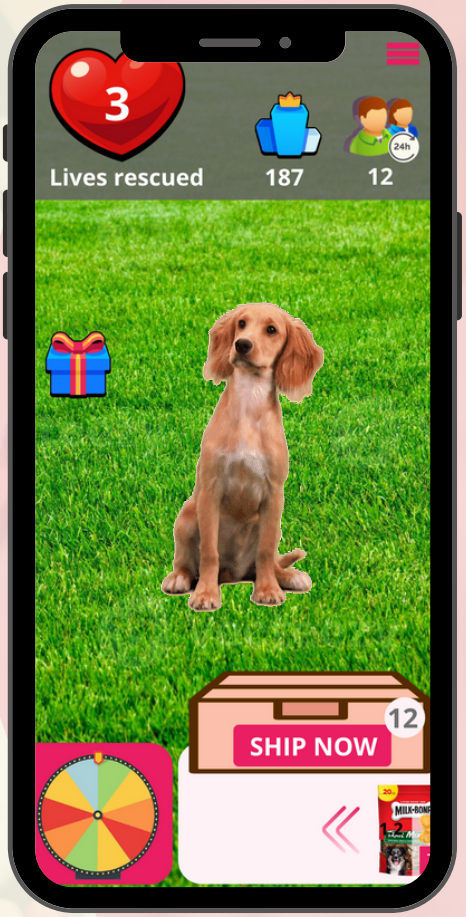
Based on Deepfake technology, have an impact on the real pet

HOW

1 **CHOOSE A PET**
TREAT creates its Deepfake

2 **TREAT THE WAY YOU WANT**
**Free:** Earn food and unlock prizes & info
**Basic:** Send the food
**Advance:** Take for a walk/foster/ adopt OR buy walks/kennel cleaning & get pictures

3 **RESCUE LIVES**
Get notified once adopted, and get a new pet




BUSINESS MODEL

### SUBSCRIPTION

Delivery, supplies and walk subscriptions, starting from $2/week

MARKET SIZE

### DEMAND

100M Americans who like pets and gaming

### SUPPLY

6.5M available dogs & cats each year in the U.S. (API) - $2.2B pet supplies market

PILOT NUMBERS - HOUSTON, TX

$174 annual revenue per paying user
10,000 annual recurring items
88% retention of paying subscribers

BENCHMARK: ANT FOREST

### 500M USERS, 150M+ TREES

Ant Forest, a Chinese app, lets you take care of a virtual tree, and once the tree is growing to a certain size, they plant in China

## Team

**Itay Katzav, CEO**
MBA, TAU & Duke University

**Max Lerner, CTO**
BSc in Mathematics & Web, BIU

## Advisory Board

**Rian Long**
Former VP User Aquisition, Wag!

**Mika, Kuusisto**
Former CRO, My Talking Tom (virtual pet, $1B exit)

# Analytics-Model.com

# PROBLEM

A lot of data is generated within your organization's data centers.

The data originates in different sources such as web analytics, inventory, customer behavior information and etc'.

The human brain cannot make sense of all this data by charts and tables.

Organizations fail to understand everything that happens in their data.

# OUR SOLUTION

Our AI model runs continuously and autonomously across every possible metric and data source, giving you real-time control over what's happening and saying goodbye to business' blind spots. Our model studies your data's normal behavior and finds anomalies instantly, understanding every signal type and seasonal patterns.

# OUR VISION

Be connected to all data sources and explain to companies simply what's happening in their business.

# OUR ADVANTAGE

While legacy business rules methodologies do not offer a meaningful benchmark for near-real time actionable insights, emerging AI-based products are heavily dependant on structured data and complex integration with every organization, rendering a non feasible solution for most commercial organizations. Even state-of-the-art solutions fail to check singular data points, because of technology limitations and are thus blind to cross-validation and root cause analysis of numerous asynchronous different streams of data.

# COMPETITIVE ANALYSIS

MULTI-PURPOSE ALGORITHM

FULL AUTOMATION

anodot

Analytics-model

outlier

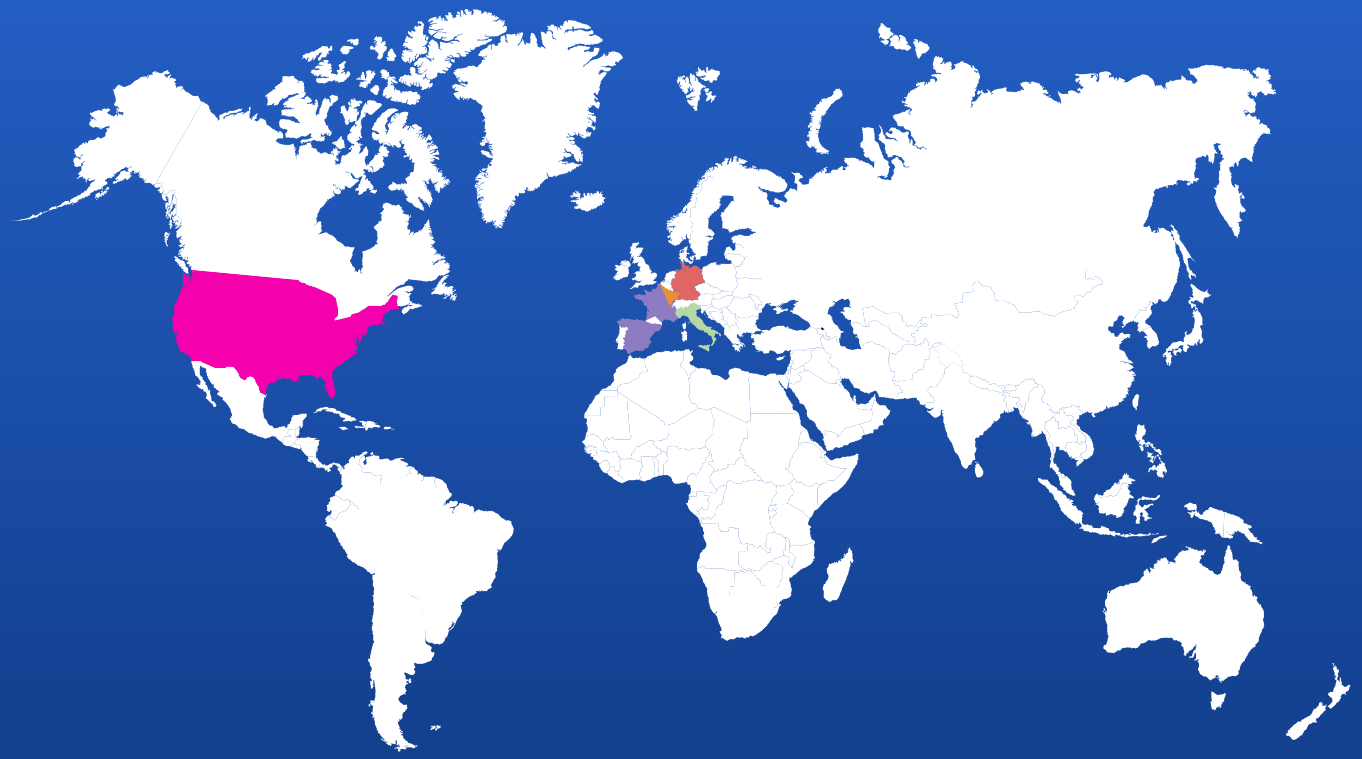
DATADOG

# MARKET SIZE

Anomaly Detection Market was valued at USD 2.97 billion in 2019 and is projected to reach USD 9.36 billion by 2027, growing at a CAGR of 16.65% from 2020 to 2027.

# GO TO MARKET

Autopilot for CEOs, CFOs, product managers and analysts in data-rich companies from the online industry.



**WEBINARS**

**PR & SEO**

**PERSONAL CONTACT**

# BUSINESS MODEL

*SUBSCRIPTIONS*
Renewable model for monthly billing

*INTEGRATION TYPES*
Integration with leading companies in the field

*EXTERNAL DATA SOURCES*
External data that gives a relative advantage to the company

*CUSTOM SERVICE*
For large companies with specific requirements

# OUR TEAM

Idan Moradov
**CEO**

MSc in Financial Mathematics from Bar-Ilan University. 12 years of experience in online companies as Head of analytics and BI. The founder of YourVoice - Music application that reached more than 6,000,000 downloads.

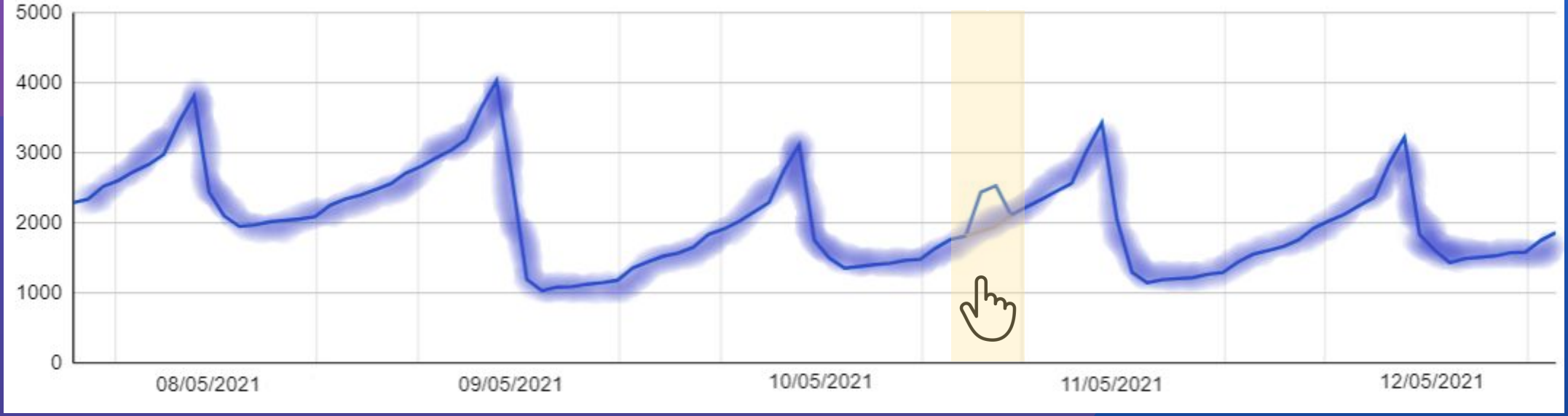
Avraham Morgenstern
**CTO**

Ph.D in Study Mathematics. Experienced Data Scientist. Built anomaly detection algorithms for years. Worked as Data Science at Anodot.

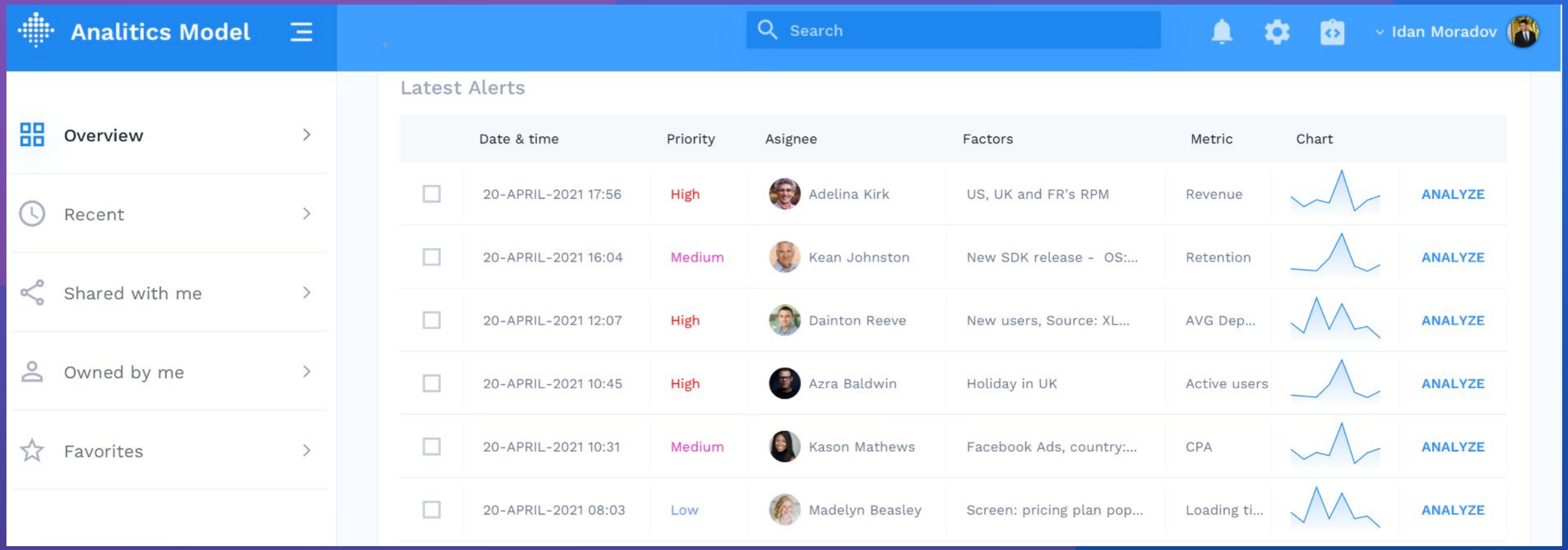
# ANOMALY DETECTION EXAMPLE

SPIKE ALERT 30%

A bug in loading the app caused many users clicks

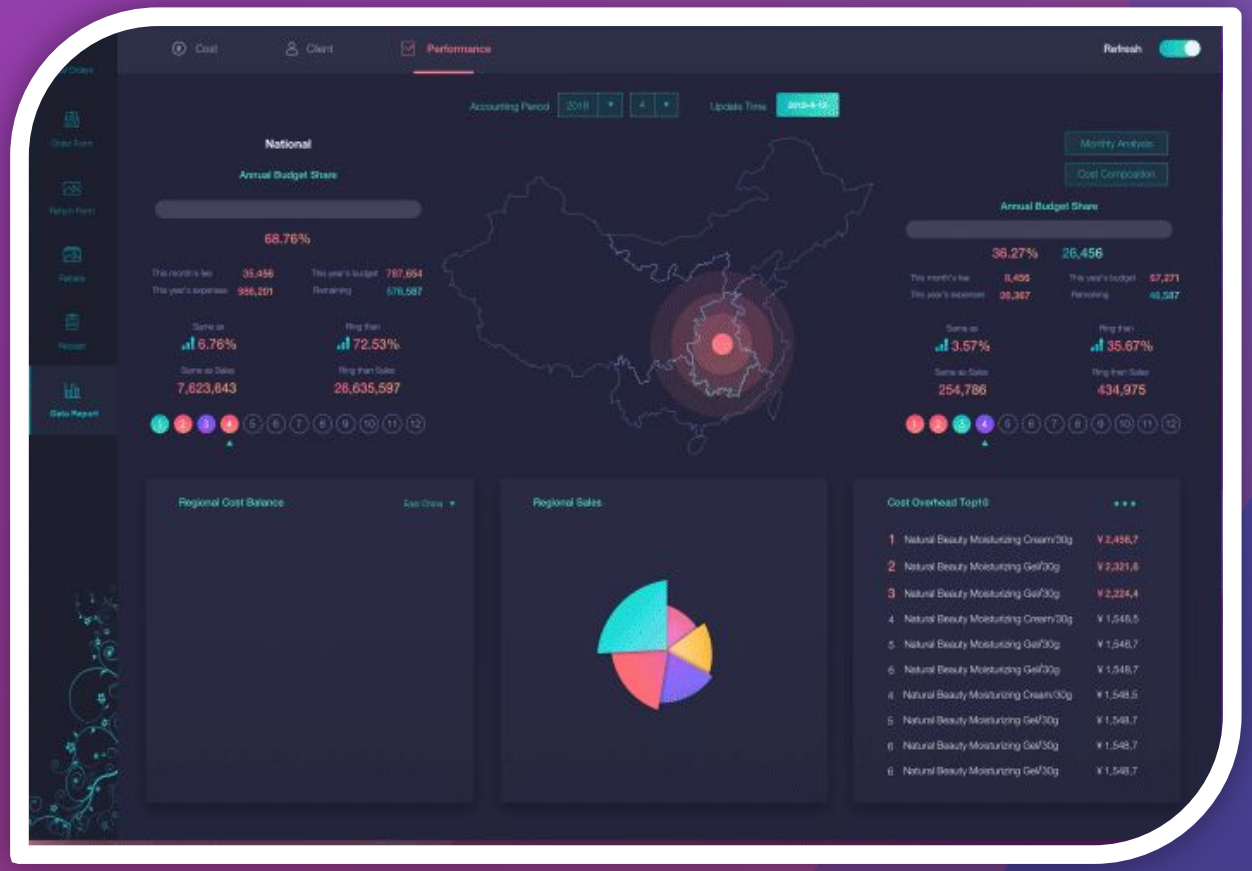


# ALERT LIST


Analitics Model
Search
Idan Moradov
Overview
Recent
Shared with me
Owned by me
Favorites
Latest Alerts
Date & time
Priority
Asignee
Factors
Metric
Chart
20-APRIL-2021 17:56
High
Adelina Kirk
US, UK and FR's RPM
Revenue
ANALYZE
20-APRIL-2021 16:04
Medium
Kean Johnston
New SDK release - OS:...
Retention
ANALYZE
20-APRIL-2021 12:07
High
Dainton Reeve
New users, Source: XL...
AVG Dep...
ANALYZE
20-APRIL-2021 10:45
High
Azra Baldwin
Holiday in UK
Active users
ANALYZE
20-APRIL-2021 10:31
Medium
Kason Mathews
Facebook Ads, country:...
CPA
ANALYZE
20-APRIL-2021 08:03
Low
Madelyn Beasley
Screen: pricing plan pop...
Loading ti...
ANALYZE

# ALERT EXAMPLE

**TOMORROW IS A HOLIDAY IN CHINA,
A DECREASE OF 5% IS EXPECTED.**

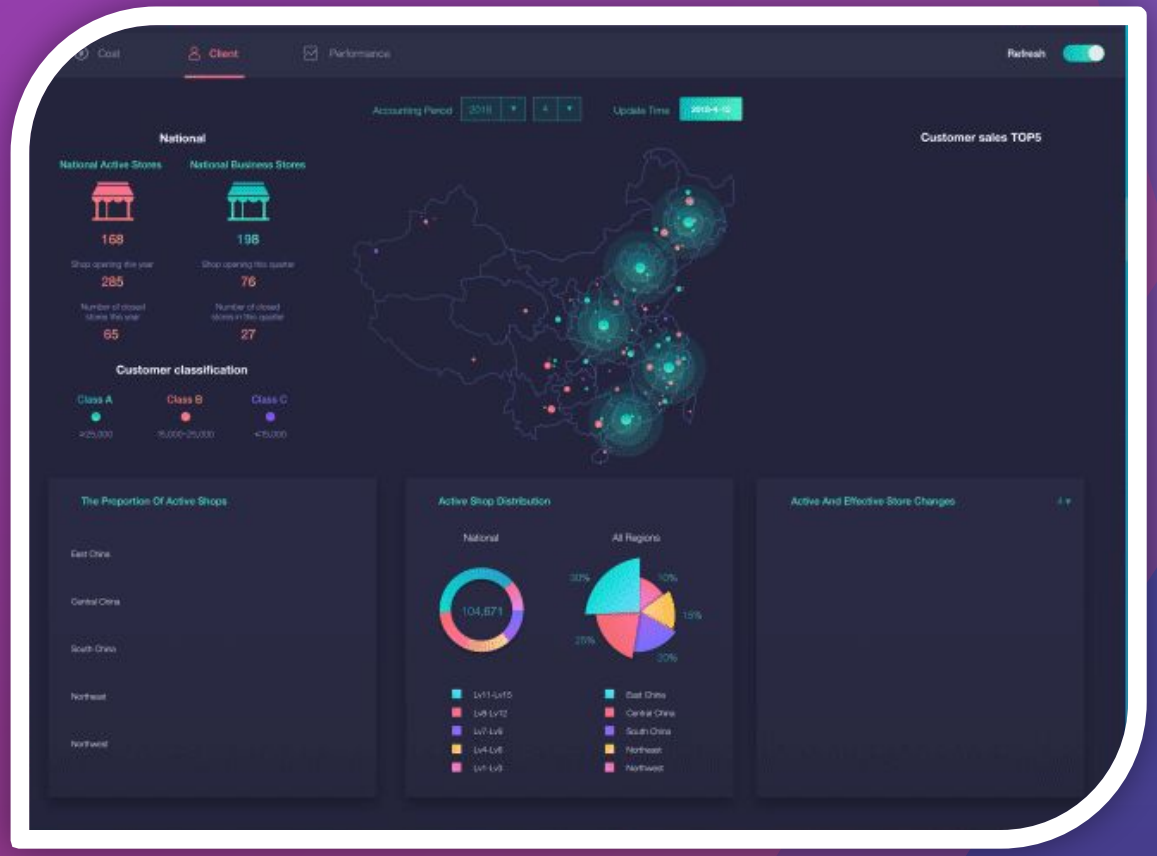


# ALERT EXAMPLE

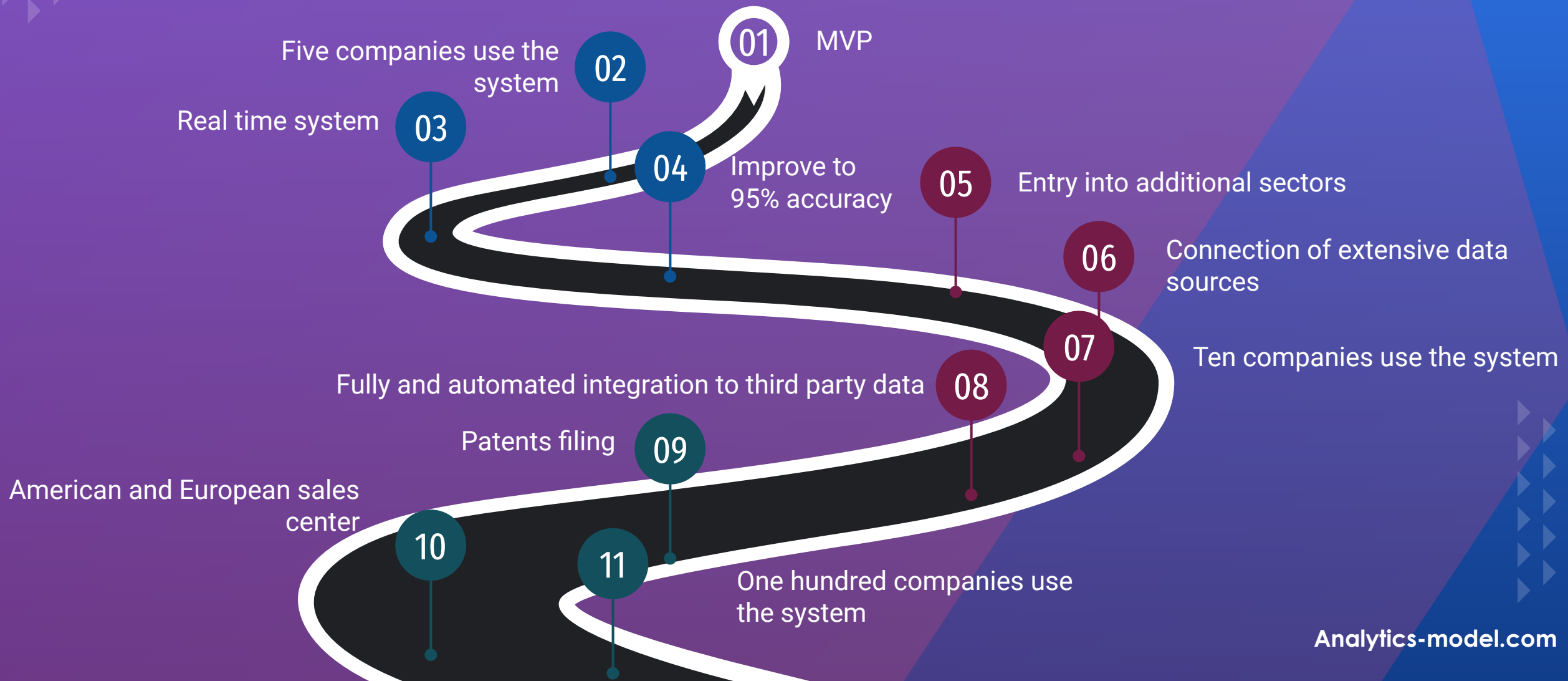
**AN UNUSUAL SPIKE IN ACTIVE USERS HAS BEEN DETECTED IN THE LAST 5 MINUTES.**

# ALERT EXAMPLE

**YOUR CONVERSION RATE HAS DROPPED SIGNIFICANTLY,**
**WE RECOMMEND STOPPING YOUR FACEBOOK RETARGETING CAMPAIGN.**



# ROADMAP




Analytics-model.com

SUPER POLYMER

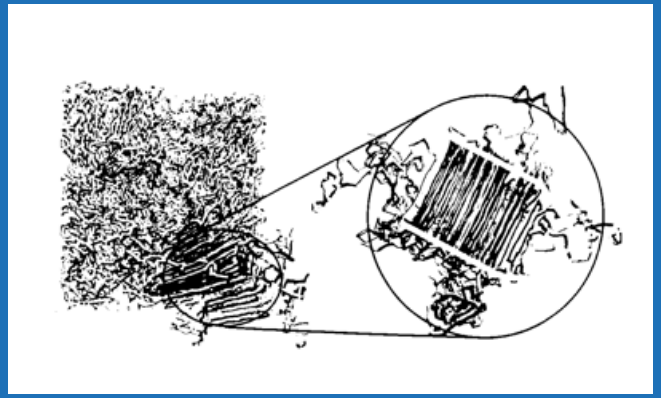
Fully Crystalline Polymers

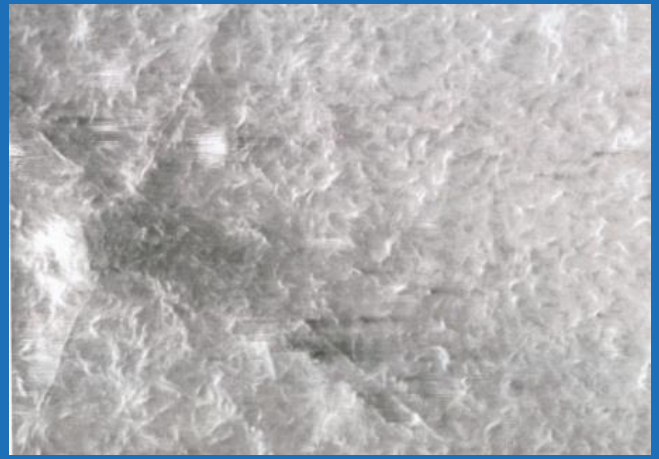
# Magic Crystals

2021

# Problem

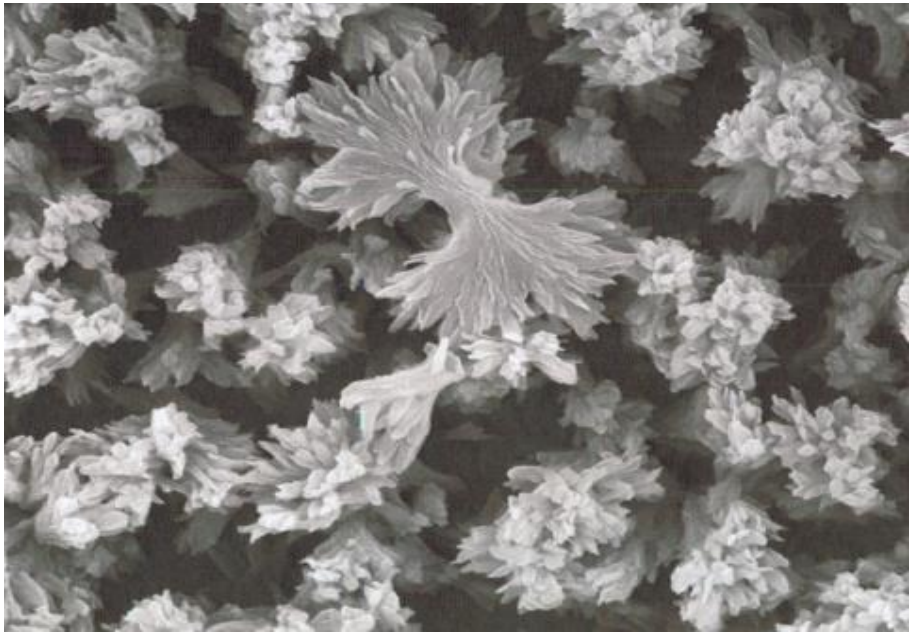
limited crystallinity Limit's polymer properties





HDPE processed in a regular method

# Our solution
## A new Super Polymer – 100% crystalline



Same HDPE processed in the new technology

# Applications

Conformal coating for PCB

Medical implant coating

Phase shift materials for Optics

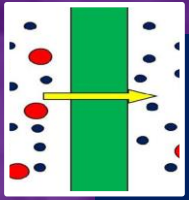
Wave retardant

Films by $m^2$

Food packaging

Coated glass

Cell culture substrate



Selective Membranes

# **Market –**
# Conformal coatings for PCB

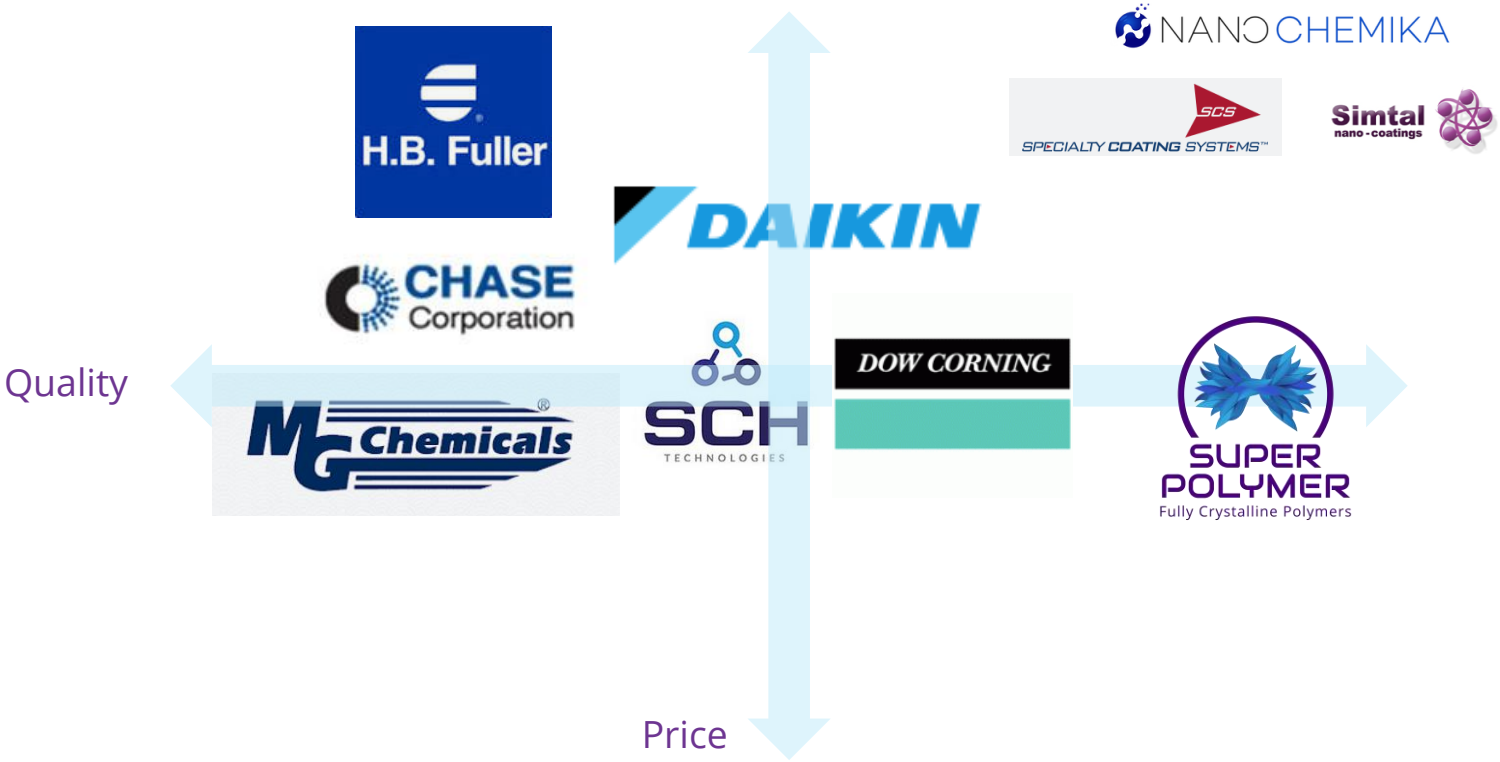
**Global market share by product, 2015 [2]**

**$1.1B**

[2025 [1]]

Source:
1. Markets & Markets, PCB coatings market, Dec 2020
2. Grand View Research, Mar 2018

# Competitors landscape



SUPER POLYMER

Fully Crystalline Polymers

# Thank you!

hagai@super-polymer.com

+972-58-7263197

77 Haenergia st. Beer Sheba 8470912 Israel

# Team

Evgeniy Mervinetsky
Head of R&D
11 Years

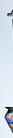
Kellie Katz
R&D Engineer
4 Years

Hagai Ortner
CEO
22 Years
Management and startups development

Atzmon Amitai
CTO
25 Years
Product growth expert in the Chemicals industry

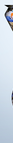
# E

Every month, we throw out enough rice, potatoes and pasta to fill up a giant skyscraper. All of these jars are recyclable!



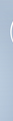
# A

About 2.01 billion tonnes of municipal solid waste is created in the world annually



# R

Reduce of Waste - guacamole of avocado can save our eco system & Planet.



# T

The energy produced from 100,000 tons of MSW can power up 25,000 homes.

# H

How can we solve this problem?

**AB'R HAS ANSWERS TO BOTH OF THESE BY-PRODUCTS!**

## WASTE TO ENERGY TECHNOLOGY

AB'R Holdings Inc is responsible for the development of technologies owned by its subsidiary companies. Our technology is based on the gasification of municipal and biomass waste. We set out the challenge to make a commercially viable, efficient and technologically advanced by deploying Beyond Incineration.

This system has been developed and tested over a period of years. It is a multi-stage process that can handle all municipal solid waste (MSW), plastic and commercial and industrial waste. Our PTA and Gasification Systems is environmentally friendly and produces no toxic emissions. The system is a closed loop system with no emissions to the atmosphere. The system is a cost effective and efficient process as it will use the energy of waste to power itself. Free any kind of raw material including hazardous waste.

## HOW IT WORKS?

WASTE

GARBAGE TRUCK

Our logistics are 4 times better than a conventional waste recycling plant.

PREPARATION BUNKER

01 Organic fraction separated from it's demand and stored under the preparation bunker. Metals are made to be separated by magnetic separation and sent to recycling. The shredded waste is then fed into a transporter that feeds the waste into the TCH furnace. The metals are delivered and suitable based on local recycled needs.

TCH FURNACE

**We do not burn or incinerate anything, hence avoiding pollution of the atmosphere.**

02 The waste of any raw material is fed into the TCH furnace. The waste is then processed and converted into a TCH-Char of chemical composition similar to charcoal. It is enough power production is large volume of syngas. The syngas is then cleaned and cooled before it is used to power the system.

Simultaneously, all organic pollutants in the raw material are destroyed and converted into syngas. The syngas is then used to power the system. The system is a closed loop system with no emissions to the atmosphere. The TCH-Char is then transformed into commercially viable products.

FOUNDRY

Thanks to the continuous cycle, we can produce commercial-grade TCH-Char for different raw materials.

CYCLONE

SCRUBER

anything that is separated from the syngas is then recycled back into the process. This will increase the efficiency in producing non-hydrocarbons.

03 The collected gas extracts from the TCH furnace passes through a cyclone, which removes the heavy elements and through a scrubber to detoxifies and clean the gas. The syngas is then compressed and cooled before it is used in Chemical Separator. The syngas is then used to power the system. The system is a closed loop system including the exhaust gas. The system is a closed loop system with no emissions to the atmosphere.

CHEMICAL SEPARATOR

Gas to carbon and salts, Carbon is used in the chemical industry.

MARKET

POWER STATION

04 The last part of the produced syngas burn is burned through an internal power. The generated power is fed back to the grid or local power provider. The system is a closed loop system including the exhaust gas. The system is a closed loop system with no emissions to the atmosphere. The system is a closed loop system. The non-cooled exhaust from the turbine generator is used for cogeneration.

GAS TO LIQUID

05 The syngas of our plant can have the option to convert the syngas into liquid fuel. The syngas is then cooled and the syngas has been in methanol synthesis and Fischer-Tropsch (FT) synthesis, which produces a mixture of liquid hydrocarbons, including naphtha, kerosene, diesel, and wax. Unreacted syngas from the FT synthesis is then used to power the system. The system is a closed loop system. The non-reacted syngas is fed into a gas turbine and nitrogen contaminants prevent a closed loop.

GASHOLDER

Carbon Sequestration can be also included in the chemical industry.

06 Our standard product contains gas is stored in the gasholder of our plant. The syngas can be sold as is, or further processed into various liquid fuel and chemical products. The system is a closed loop system including the exhaust gas. The system is a closed loop system with no emissions to the atmosphere. The system is a closed loop system. Components of hydrogen, HCl, H2 sulfur and carbon monoxide, CO, are used in the production of methanol and other petrochemicals for the chemical industry.

GREENHOUSE

Our products that are not used in the plant are sold as chemical products.

07 Our Greenhouse consists of an aquaponic system that produces vegetables and fish. Our greenhouse is a closed system. The system is a closed loop system including the exhaust gas. The system is a closed loop system with no emissions to the atmosphere. It can be used as a tourist attraction. The Greenhouse is a closed loop system. The system is a closed loop system. The system is a closed loop system. The system

# מערכת מרובת סוכנים לניהול מערך הניתוחים

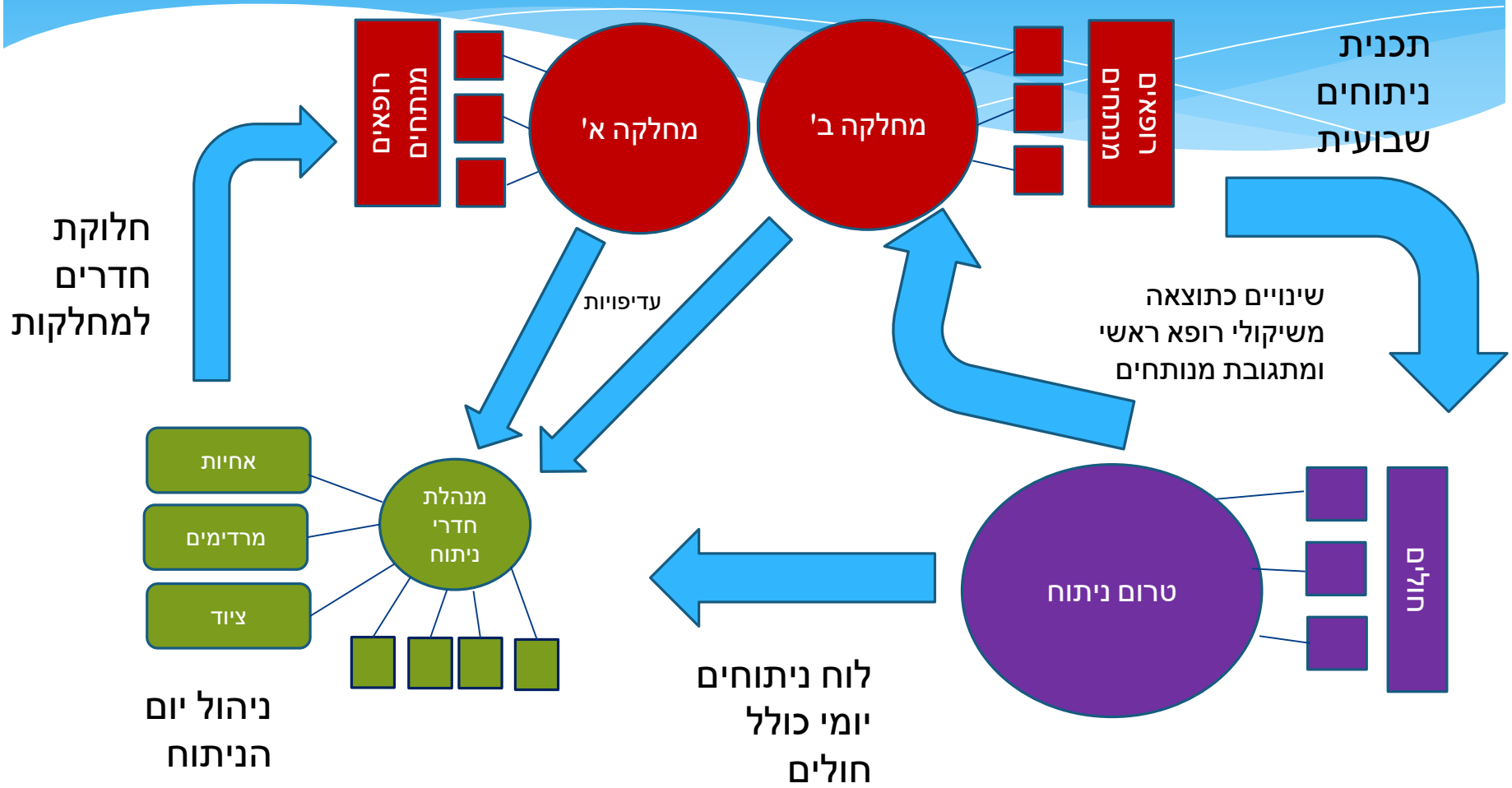
**אמנון מייזלס, ג'ילברט סבאג, רועי זיוון**

*נעם גאון, יובל גבאי, רז לפיד, עמית סולומון, נועה באייר, יאיר ואקנין*

# תאור כללי

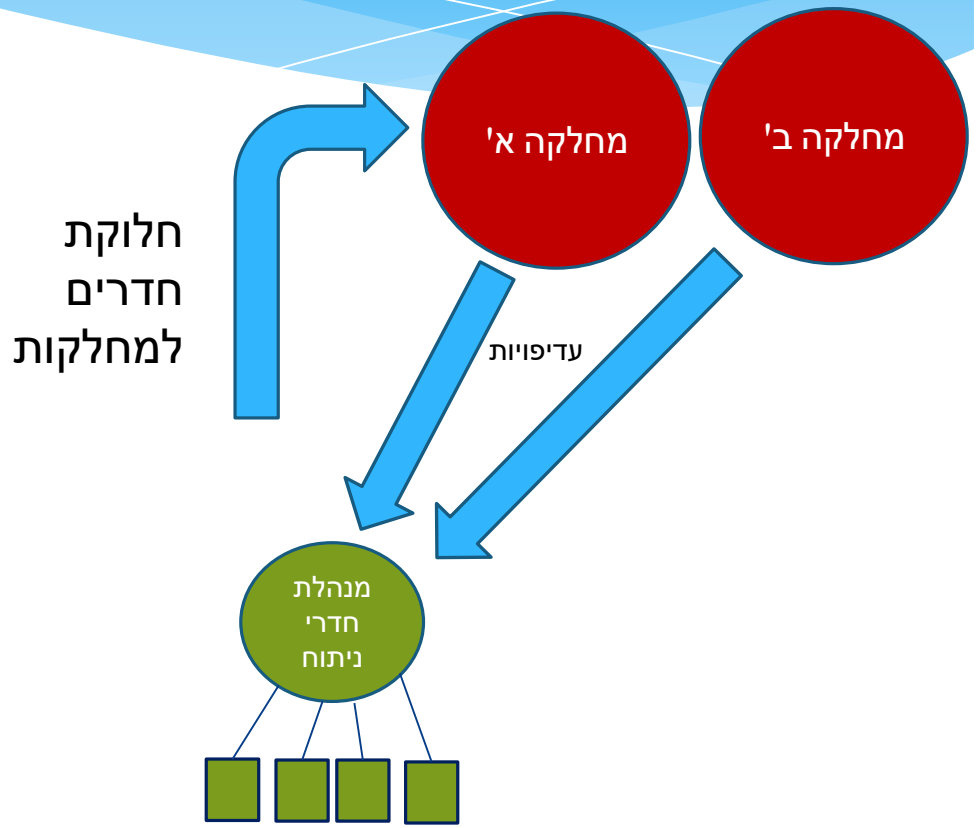
* בבית חולים סורוקה יש מעל ל 30-חדרי ניתוח
* הניתוחים מבוצעים על ידי מנתחים רבים , השייכים למחלקות שונות.
* בניתוחים משתתפים גורמים נוספים כגון , מרדימים ,אחיות ,וכו.'
* *מערך הניתוחים מהווה כ 40% מהכנסות בית החולים.*

# השיטה הקיימת -מבט מלמעלה



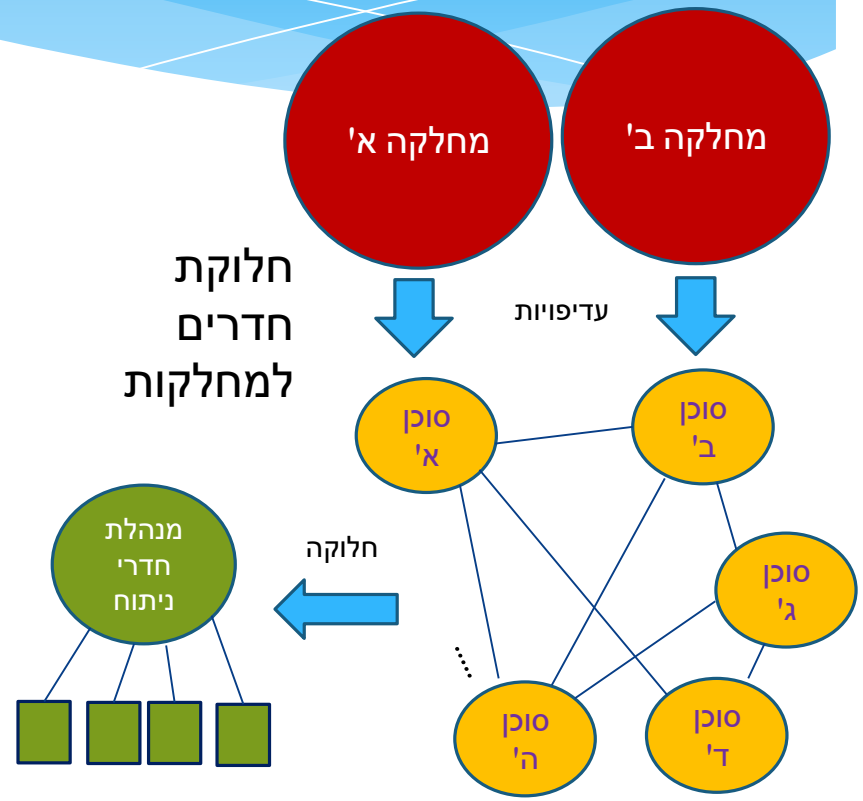
# שלב ראשון: הקצאת חדרי ניתוח למחלקות

- מצב כיום :
  - מחלקות מעבירות עדיפויות למנהלת.
  - המנהלת משבצת חדרים למחלקות בהתבסס על חלוקות קודמות .



# שלב ראשון: הקצאת חדרי ניתוח למחלקות

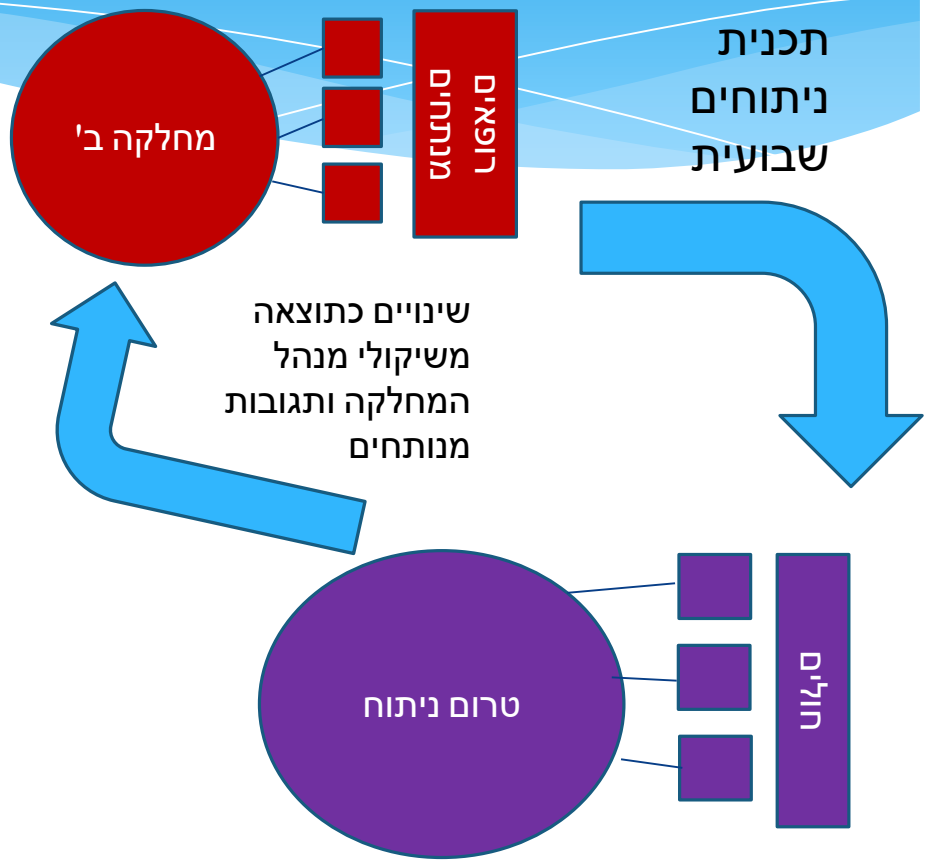
- מצב כיום :
  - מחלקות מעבירות עדיפויות למנהלת.
  - המנהלת משבצת חדרים למחלקות בהתבסס על חלוקות קודמות .
- המערכת המוצעת:
  - ראשי מחלקות מעדכנים את הסוכנים שלהם בעדיפויות.
  - מגבלות ההקצאה נקבעות על ידי המנהלת.
  - החלוקה מתבצעת על ידי אלגוריתם מבוזר בין סוכני המחלקות.
  - המנהלת מעודכנת בתוצאה.



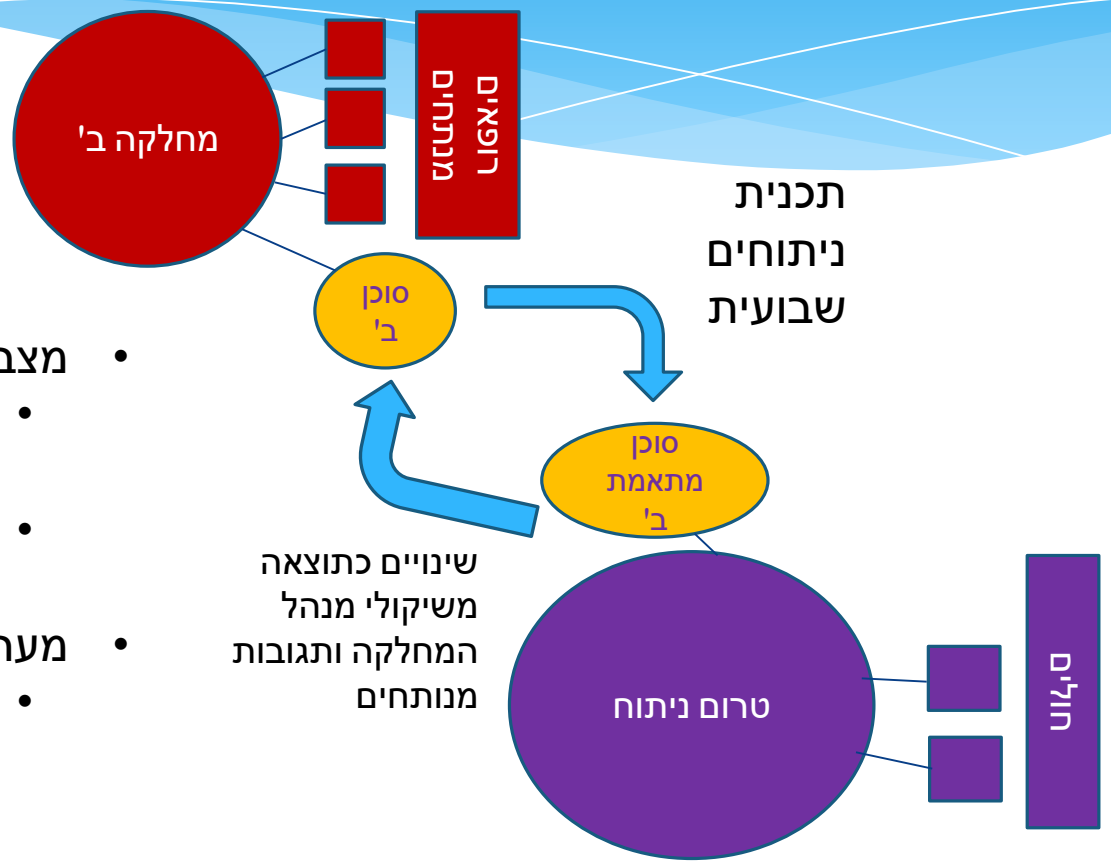
# שלב ראשון: הקצאת חדרי ניתוח למחלקות

# שלב שני: שיבוץ ניתוחים לימי ניתוח

- מצב כיום :
  - ראשי מחלקות ומתאמות טרום ניתוח.
  - אינטראקציה היראכרכית .

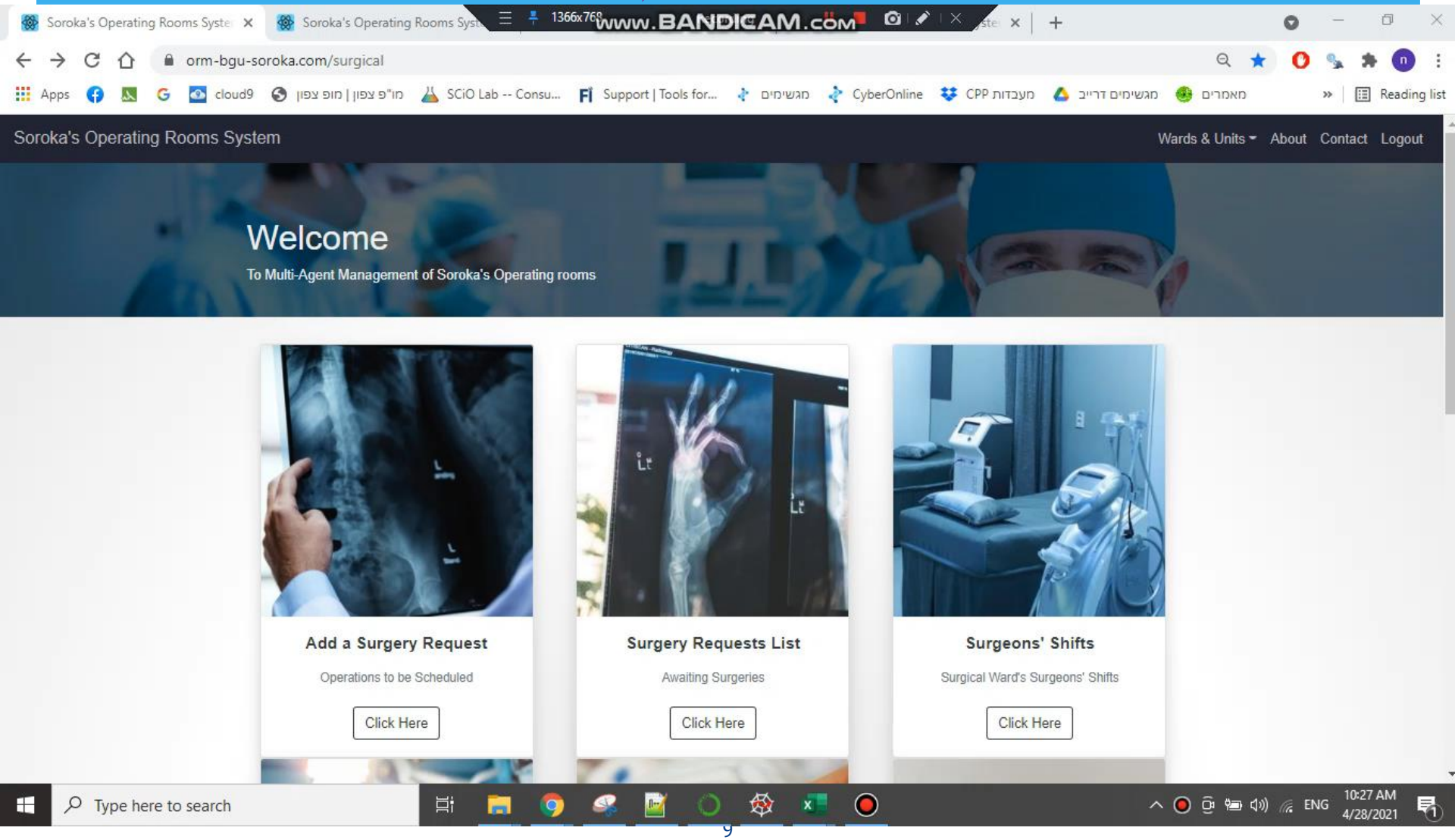


# שלב שני: שיבוץ ניתוחים לימי ניתוח




- מצב כיום :
  - ראשי מחלקות ומתאמות טרום ניתוח.
  - אינטראקציה היראכרכית .
- מערכת מוצעת:
  - ראשי מחלקות ומתאמות מיוצגים על ידי סוכנים המקיימים ביניהם משא ומתן
  - אפשרות לתיקונים אנושיים להחלטת הסוכנים.

# שלב שני: שיבוץ ניתוחים



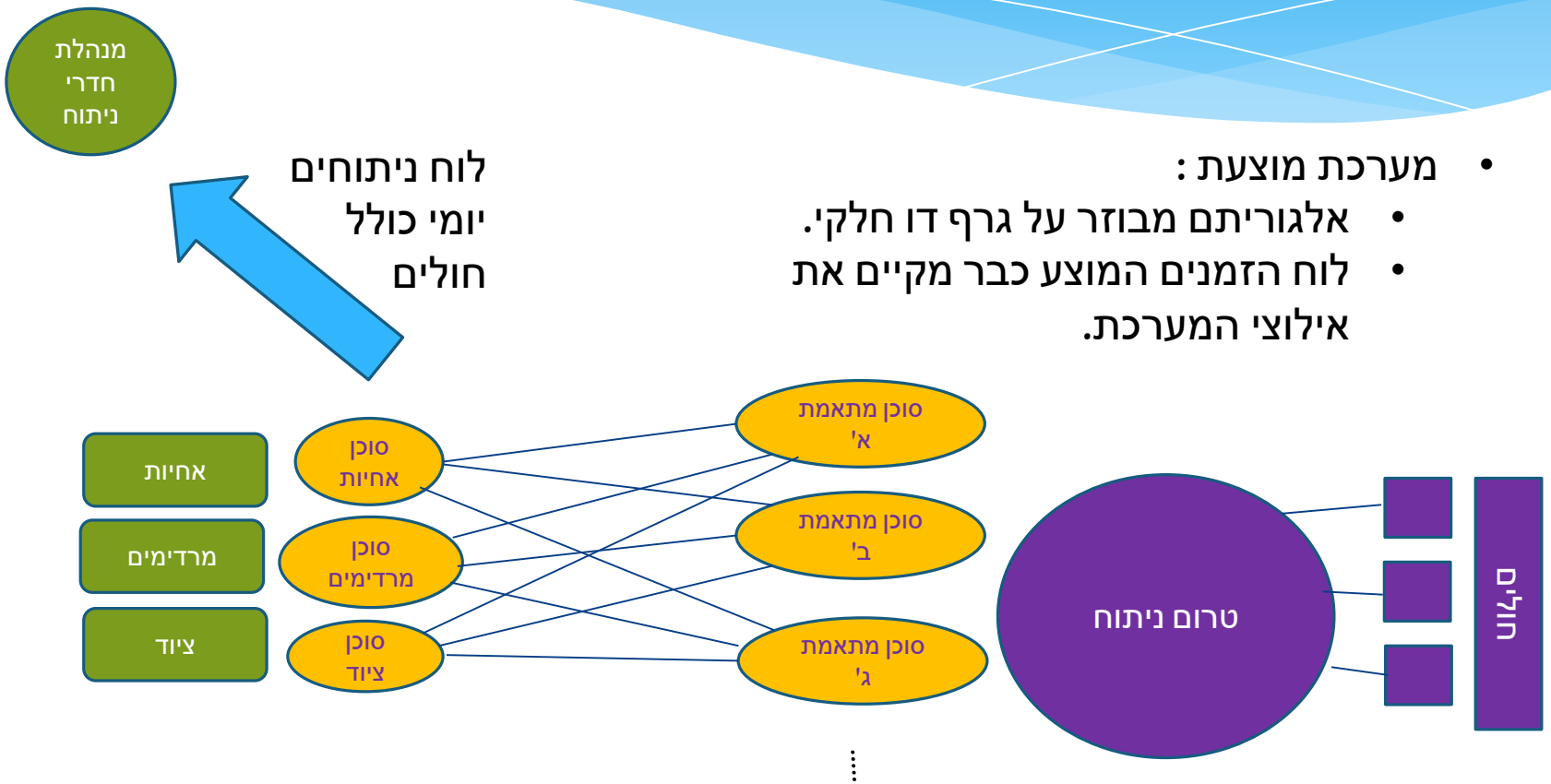
# שלב שלישי: קביעת לוח ניתוחים יומי

- מצב כיום :
  - "לפגישה של "12מגיעות המחלקות עם לו"ז.
  - אילוצים בין מחלקתיים נפתרים על ידי המנהלת בפגישה

# שלב שלישי: קביעת לוח ניתוחים יומי

- מערכת מוצעת :
  - אלגוריתם מבוזר על גרף דו חלקי.
  - לוח הזמנים המוצע כבר מקיים את אילוצי המערכת.

לוח ניתוחים יומי כולל חולים



# שלב שלישי: קביעת לוח ניתוחים יומי

Soroka's Operating Rooms Syste

www.BANDICAM.com

orm-bgu-soroka.com/surgical/partial_scheduling

today

April 2021

month week day list

**Unscheduled Requests:**

- Request 45
- Request 46
- Request 47
- Request 48
- Request 49
- Request 50
- Request 51
- Request 52

| Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|---|---|---|---|---|---|---|
| 28 | 29 | 30 | 31 | 1 | 2 | 3 |
| 4<br>3a 211676449.0 bariatri<br>3a 247179238.0 bariatri<br>4a 363905611.0 bariatri<br>5a 397897324.0 bariatri<br>5a 356445416.0 hernia<br>7a 395387718.0 hernia<br>9a 282603675.0 hernia | 5 | 6<br>3a 293931348.0 colon C<br>3a 337093277.0 colon C<br>5a 219050079.0 colon C<br>7a 322327678.0 Hospita<br>9a 389551841.0 Hospita<br>11a 316674722.0 biopsy<br>1p 342562214.0 biopsy | 7<br>3a 369797651.0 hernia<br>5a 213582042.0 hernia<br>7a 324422166.0 hernia | 8<br>3a 344204696.0 gall bla<br>3a 217103157.0 hernia<br>4a 217941682.0 hernia<br>5a 369220200.0 Proctol<br>5a 261014943.0 piles C<br>6a 290672802.0 Proctol<br>7a 220379068.0 Proctol<br>9a 340021058.0 Proctol | 9 | 10 |
| 11<br>3a 228051220.0 bariatri<br>3a 344103454.0 bariatri<br>4a 359518223.0 gall bla<br>5a 279580682.0 bariatri<br>5a 273679242.0 gall bla<br>7a 242764671.0 bariatri<br>9a 272519002.0 hernia | 12 | 13<br>3a 257552432.0 Hospita<br>3a 358428155.0 hernia<br>5a 309660164.0 Hospita<br>7a 399832177.0 hernia | 14<br>3a 218906687.0 gall bla<br>3a 330668825.0 hernia<br>4a 269254068.0 hernia | 15<br>3a 345543506.0 Proctol<br>5a 322828562.0 piles C<br>7a 360483179.0 piles C<br>9a 383525692.0 piles C<br>11a 237954730.0 piles C | 16 | 17 |
| 18<br>3a 331770449.0 bariatri<br>3a 303909359.0 gall bla | 19 | 20<br>3a 222784641.0 colon C<br>3a 366314023.0 cancer | 21<br>3a 288898769.0 hernia<br>5a 385472324.0 hernia | 22<br>3a 380051744.0 piles C<br>5a 243562262.0 Proctol | 23 | 24 |

Type here to search

18:03
27/04/2021

# אתגרים להמשך

* מידול עדיפויות משתמש +אלגוריתמי למידה
* פתרון בעיות דינמיות לניתוחים לא מתוכננים אחרי הפגישה של 12
* גיוס משאבים להפיכת המערכת למוצר

# Thank You

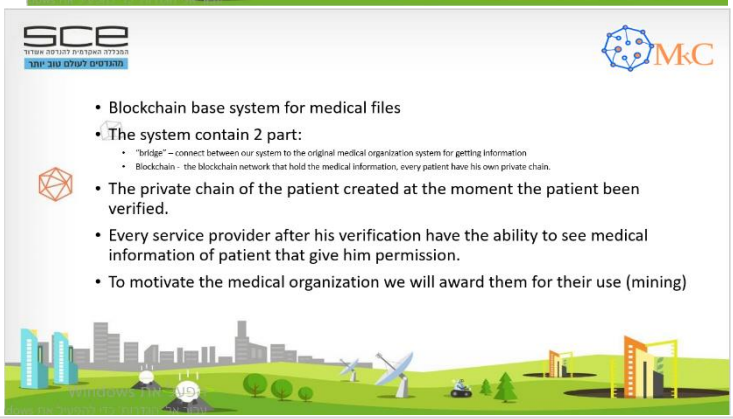
**Entrepreneurship Pitch**

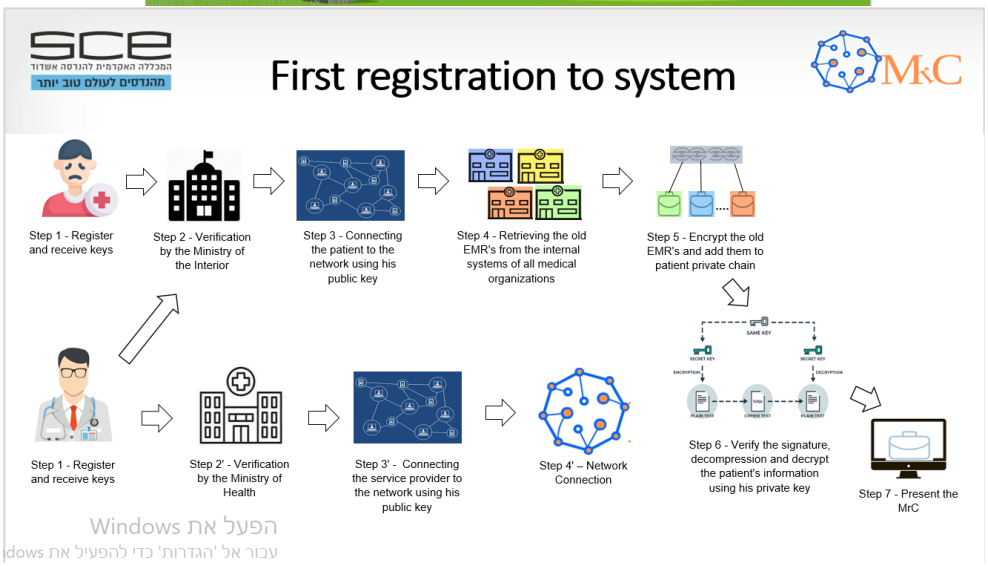
# MrC: a Medical Record Chain System based on blockchain

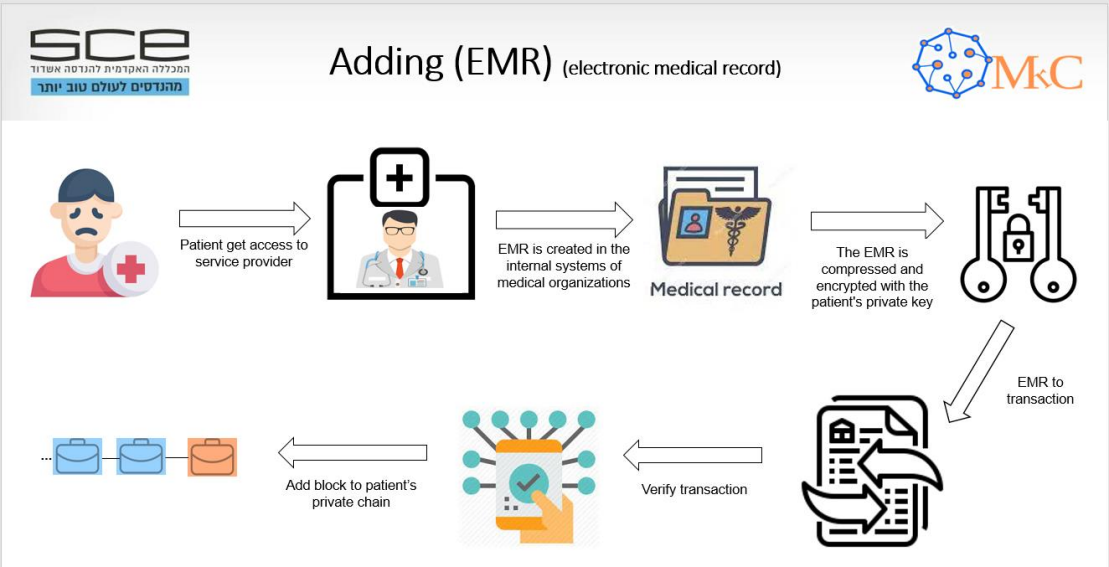

Harel Jerbi[0000−0003−3280−5746], Shimon Shai Idan[0000−0003−3807−2598]

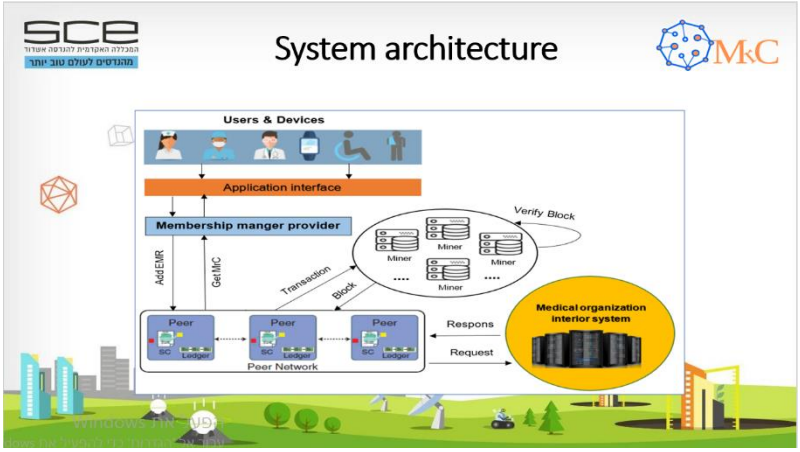
Mentor: Hadassa Daltrophe[0000−0003−2305−125X]

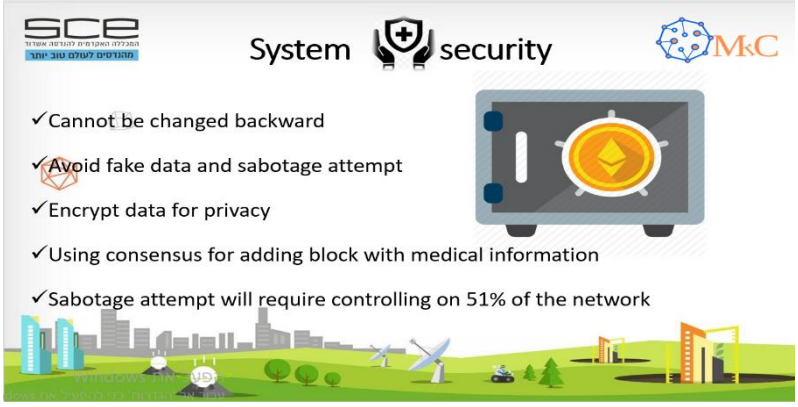
Sami Shamoon College of Engineering (SCE), Israel




SCE
המכללה האקדמית להנדסה אשדוד
מהנדסים לעולם טוב יותר
Adding (EMR) (electronic medical record)
MkC
Patient get access to service provider
EMR is created in the internal systems of medical organizations
Medical record
The EMR is compressed and encrypted with the patient's private key
EMR to transaction
Verify transaction
Add block to patient's private chain


SCE
המכללה האקדמית להנדסה אשדוד
מהנדסים לעולם טוב יותר
System architecture
MkC
Users & Devices
Application interface
Membership manger provider
Verify Block
Add EMR
Get MrC
Transaction
Block
Miner
Peer
SC
Ledger
Peer Network
Respons
Request
Medical organization interior system


SCE
המכללה האקדמית להנדסה אשדוד
מהנדסים לעולם טוב יותר
System security
MkC
✓Cannot be changed backward
✓Avoid fake data and sabotage attempt
✓Encrypt data for privacy
✓Using consensus for adding block with medical information
✓Sabotage attempt will require controlling on 51% of the network

# V@FHE

Arseni Kalma *Founder & CEO*
Prof. Shlomi Dolev *Consultant*

# The team

*Cryptography and Development Expertise*

## Arseni Kalma

Former Team Leader (“Captain”)
in the IAF (Ofek 324)

## Prof. Shlomi Dolev

Rita Altura Trust Chair
Professor, IEEE & EAI Fellow

# The Cloud Is Not Checked

Remote “public” clouds are unsupervised

Most of our sensitive data is operated without verifications of the data, operations, or results

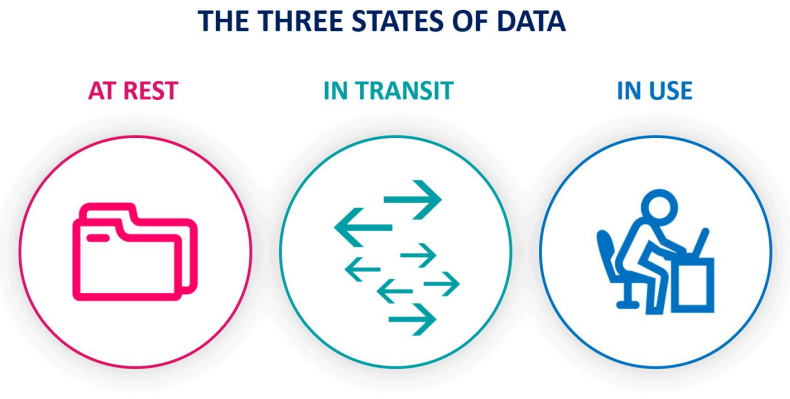
Financial, health and math calculations operated on remote machines, with **blind trust**

# And Not Always Encrypted..

Decryption is necessary to operate data

Different encryptions and transitions

Many encryption keys and points of failure



# Verifiable Computing in FHE

Verify computation made on encrypted data

All data encrypted at rest, every computation is verified, No additional overhead

# Milestones

*Show where you are in the process and what's left to tackle*

## January 20XX

Lorem ipsum dolor sit amet, consectetur

## June 20XX

Lorem ipsum dolor sit amet, consectetur

## October 20XX

Lorem ipsum dolor sit amet, consectetur

Jan Feb Mar Apr May Jun Jul Aug Sept Oct Nov Dec

## March 20XX

Lorem ipsum dolor sit amet, consectetur

## July 20XX

Lorem ipsum dolor sit amet, consectetur

# Appendix

***Show the audience you anticipated their questions.***

*Leave room for Q&A, but use the Appendix as a way to show that you both thought about those questions and have solid answers with supporting information. Let the audience test their understanding of the problem and the solution you've outlined - questions give them a chance to talk themselves into your approach, and give you a chance to show mastery of the subject.*

# FHE Means Working on Encrypted Data

Allow operations on encrypted values

In high demand, “only” for hiding data

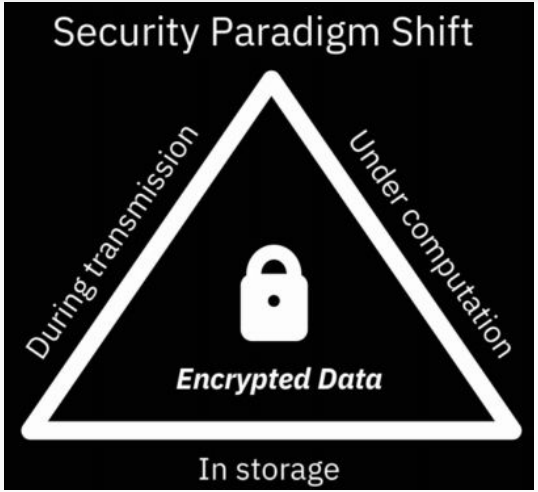
Intrado GlobeNewswire

Homomorphic Encryption Market Valuation of USD 268.92 Million By 2027 With a Healthy CAGR of 8.58% | North America to Remain Forerunner in Global Homomorphic Encryption Market

The INSIGHT Partners

**Publication Month:** Dec 2020 | **Report Code:** TIPRE00014748
**No. of Pages:** 149 | **Category:** Technology, Media and Telecommunications | **Status:** Published

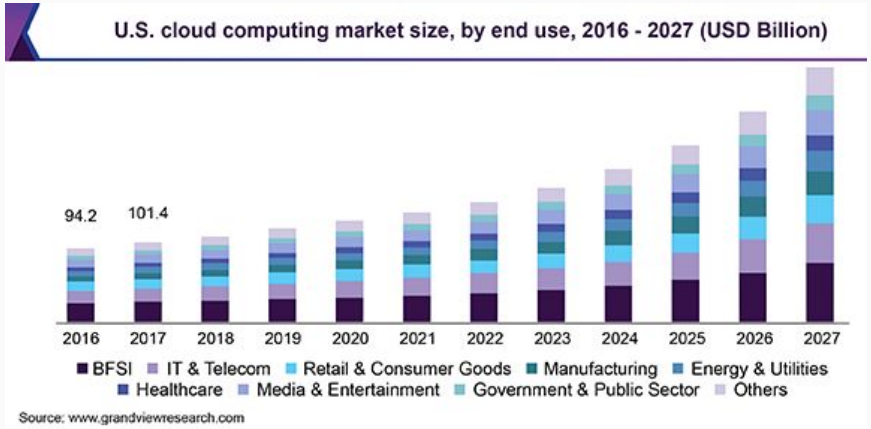
The homomorphic encryption market was valued at US$ 120.12 million in 2019 and is expected to reach US$ 246.29million by 2027; it is estimated to grow at a CAGR of 9.7% during 2020–2027.

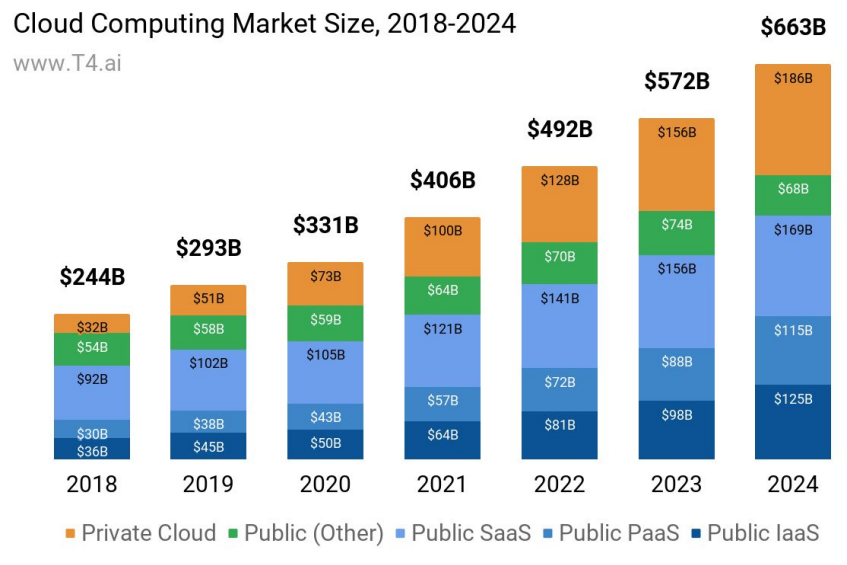


# Verified work on Encrypted Data - New Market

Access to a $500B market

Many different high-end sectors



# Competition

Present solutions require creating and checking the proof

Other solutions are interactive, verifying manually and partially

## Amazon and Capital One face legal backlash after massive hack affects 106M customers

BY NAT LEVY on August 9, 2019 at 12:16 pm

A group of angry customers filed a lawsuit against Capital One this week following the hack that affected more than 106 million people. And they aren't stopping there; the group also named Amazon Web Services, Capital One's cloud provider, alleging the tech giant is also culpable for the breach.

# The Technology: FHE + SIMD

# Verify FHE Operations

## Concise the calculation

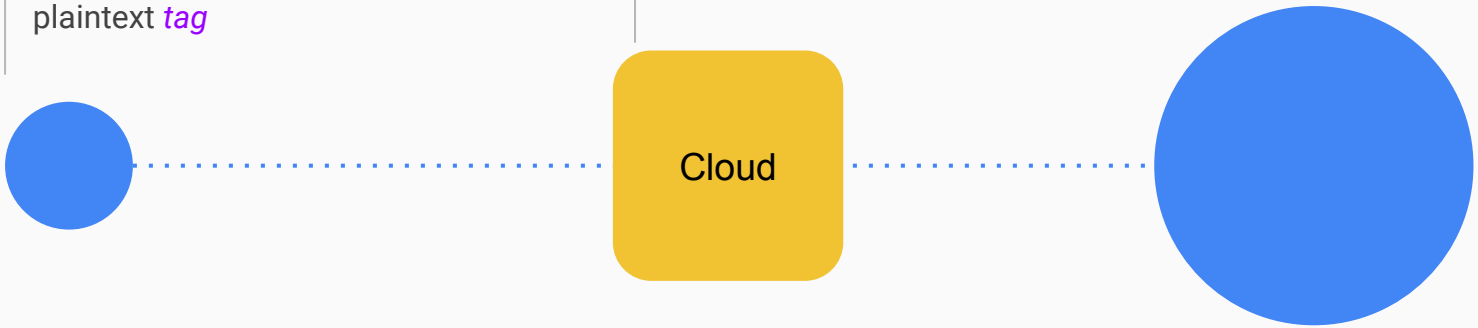
Operate the required calculation on *'witness'* input, resulting in a plaintext *tag*

## Operate

Append the *'witness'* to inputs, operating simultaneously, yet independently, while encrypted

## Verify

Decrypt and verify the resulted *tag*, use the result

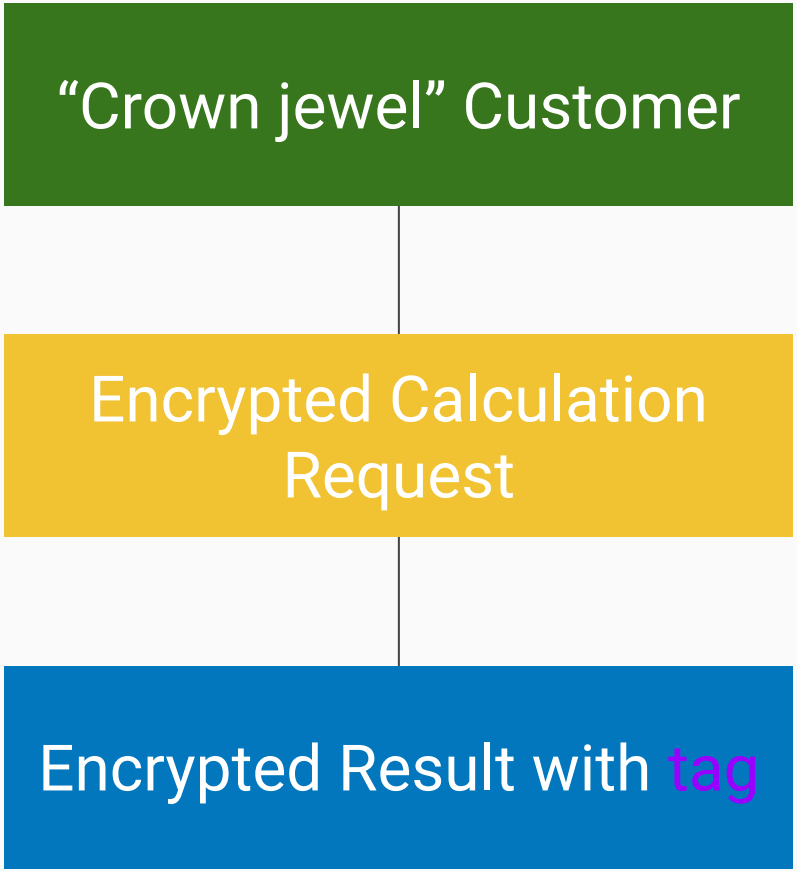



***Patent pending***

# Revenue Model

Operations on sensitive data
- Encrypted once only
- Encrypted tag in each result

Possible blockchain implementation

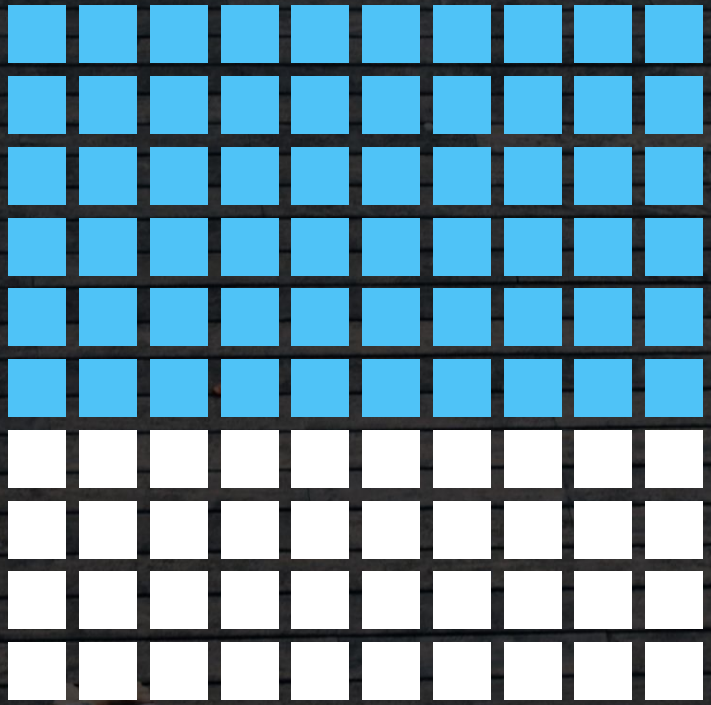


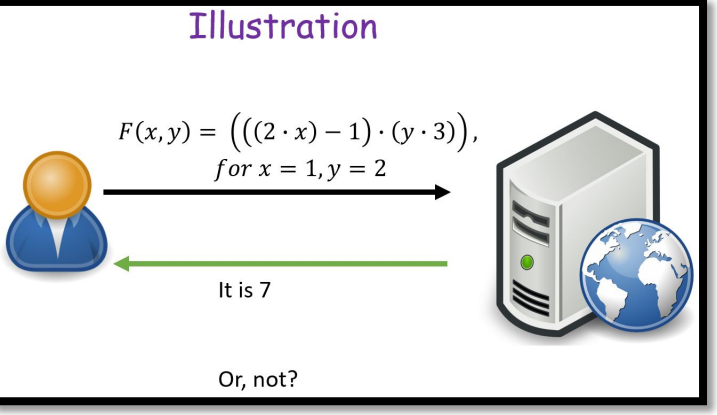
# Prestigious Revenue Potential

## Clouds are Point-Of-Failure

Most of our important data and operations are executed on some cloud

It is a good time to verify the cloud outputs





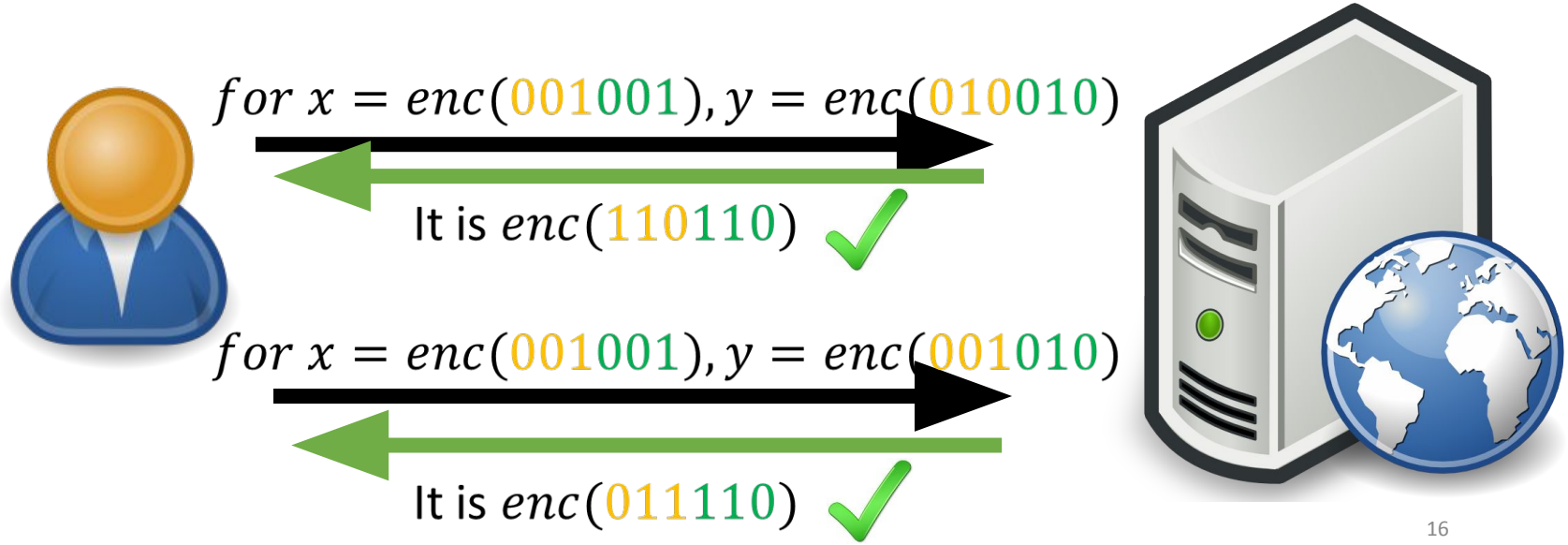

# Solution

$$F(x,y) = \left(((2\cdot x)-1)\cdot(y\cdot 3)\right),$$

<u>We know that for</u>

$$x = enc(001), y = enc(010)$$

$$F(x,y) = enc(011)$$



# SIMD scheme

- Vector elements are separated
- Tag will reside in a vector cell
- A single fingerprint value can verify multiple inputs

| | | | | |
|---|---|---|---|---|
| $Computation_{11}$ | $Computation_{12}$ | $Computation_{13}$ | $Fingerprint_1$ | $Computation_{14}$ |

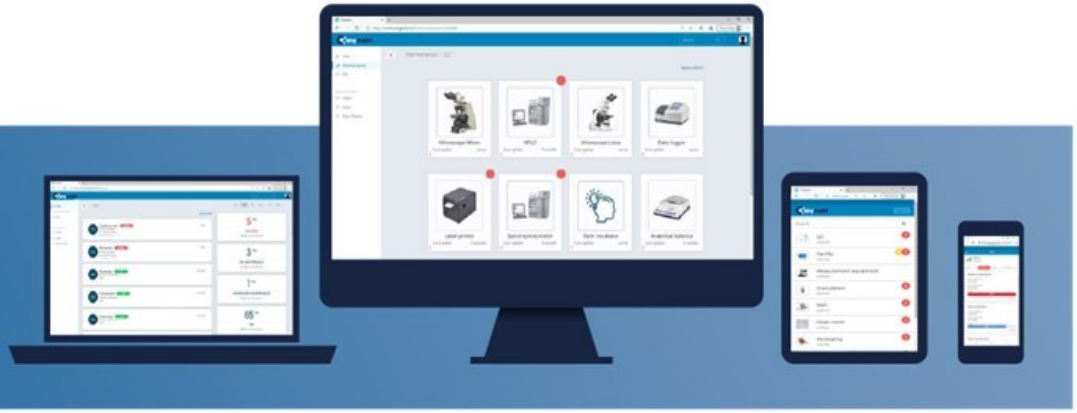
X

| | | | | |
|---|---|---|---|---|
| $Computation_{21}$ | $Computation_{22}$ | $Computation_{23}$ | $Fingerprint_2$ | $Computation_{24}$ |

| |
|---|
| $Expected$ result? |

anymaint

# INDUSTRY 4.0
# INNOVATIVE SOLUTION
# COLLABORATIVE MAINTENANCE



## Paying Customers

MEDITEC, OPKO, Allergan, TAT Technologies, stratasys, Caniel, Mul-T-Lock, amiad WATER SYSTEMS, elcon recycling center, CRONOS GROUP, NOVAmed, RAPHAEL BY TALIS, shamir, Man, FIELD PRODUCE Ltd, ALUBIN Aluminum Industries, Clal Concrete Products Ltd

## AnyMaint Software

- Unique solution of collaborative maintenance
- Machine learning technology
- All-in-one software solution for maintenance management
- Intuitive, simple and user-friendly interface
- Suitable for all industries and for all factory sizes
- Easy onboarding and fast implementation
- Directly reduces costs and immediately improves production process with no capital investment

## Unplanned Downtime

**$50BN losses each year**

## Predictive Maintenance

**Fast-growing Market**

**25.2% CAGR through 2025**

**We`re raising now , let`s collaborate !**

Contact Us: Vladimir Krasniansky, Founder/CEO vladimir@anymaint.com +972-52-6174879 www.anymaint.com