TSXor: A Novel Time Series Compression Algorithm

UNIVERSITY OF PISA
DEPARTMENT OF COMPUTER SCIENCE

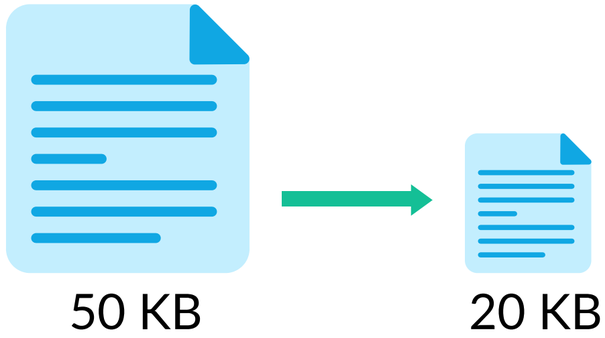
CANDIDATE
Andrea Bruno

SUPERVISORS
Franco Maria Nardini
Rossano Venturini

COMPRESSION - 1

The discipline of **data compression** deals with reducing the size of the data.

A data compression algorithm reads a stream of bits and outputs a shorter stream.



COMPRESSION - 2

Data are representations of a phenomenon, event, or fact, made through symbols or combinations of symbols.

Informations are the interpretations of a data, namely, the semantics we associate with data.

COMPRESSION - 3

Find redundancies

Discover hidden patterns within the data and try to eliminate them.

Shannon's Theorem

Entropy as a lower bound for compressing data

$$H(X) = -\sum_{c \in \Sigma} p_c \log_2(p_c)$$

COMPRESSION: WHY?

Environment

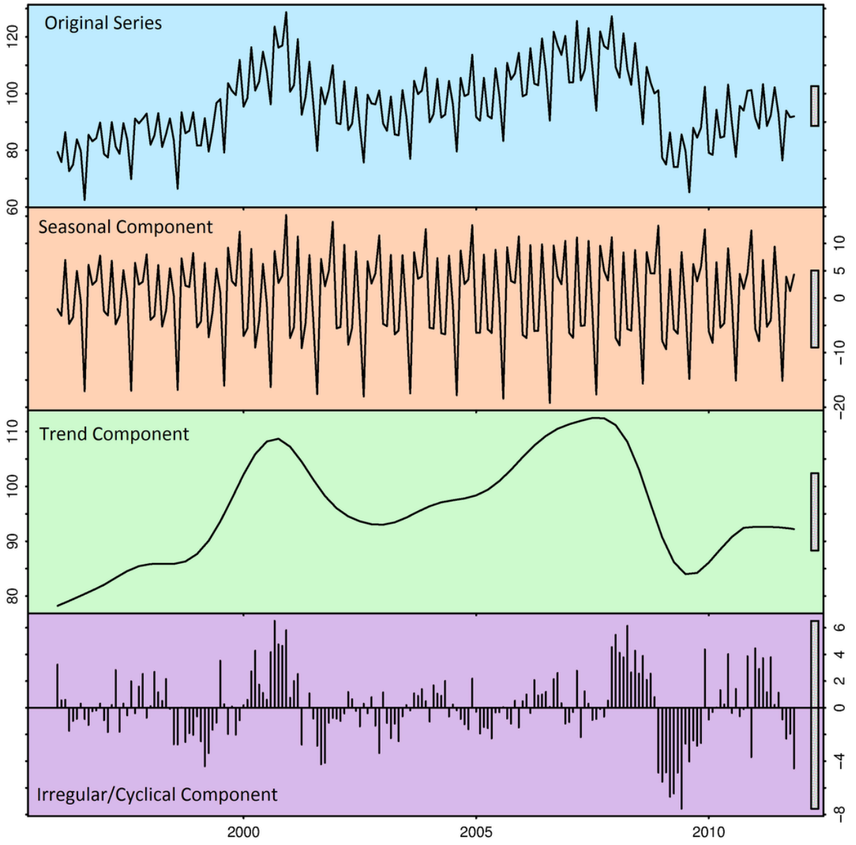
"There are 2.5 exabytes (10^{18}) of data created each day at our current pace"

Resource Saving

Storage, power consumption, network bandwidth

TIME SERIES - 1

*"A **time series** is a list of observations recorded in order of time"*



TIME SERIES - 2

- **Univariate Time series**

  Single time-dependent variable

  $< T(n), V(n) >$

- **Multivariate Time series**

  Multiple time-dependent variables

  $< T(n), [V_1(n), V_2(n), \ldots, V_m(n)] >$

IEEE 754: FLOATING POINT

$$x = (-1)^{s} \times (1.b_{p-1}b_{p-2}\ldots b_0)_2 \times \beta^{e}$$

Sign ***s*** ∈ *{0,1}*

Mantissa with precision ***p***

Base ***β*** ≥ 2

Exponent ***e***, $e_{min} \leq e \leq e_{max}$

DOUBLE PRECISION FLOATING POINT

$$s \in \{0,1\}$$

$$\mathbf{p} = 52 \text{ bits}$$

$$\beta = 2$$

$$-1022 \leq e \leq 1023$$

FLOATING POINT COMPRESSION

High level of entropy

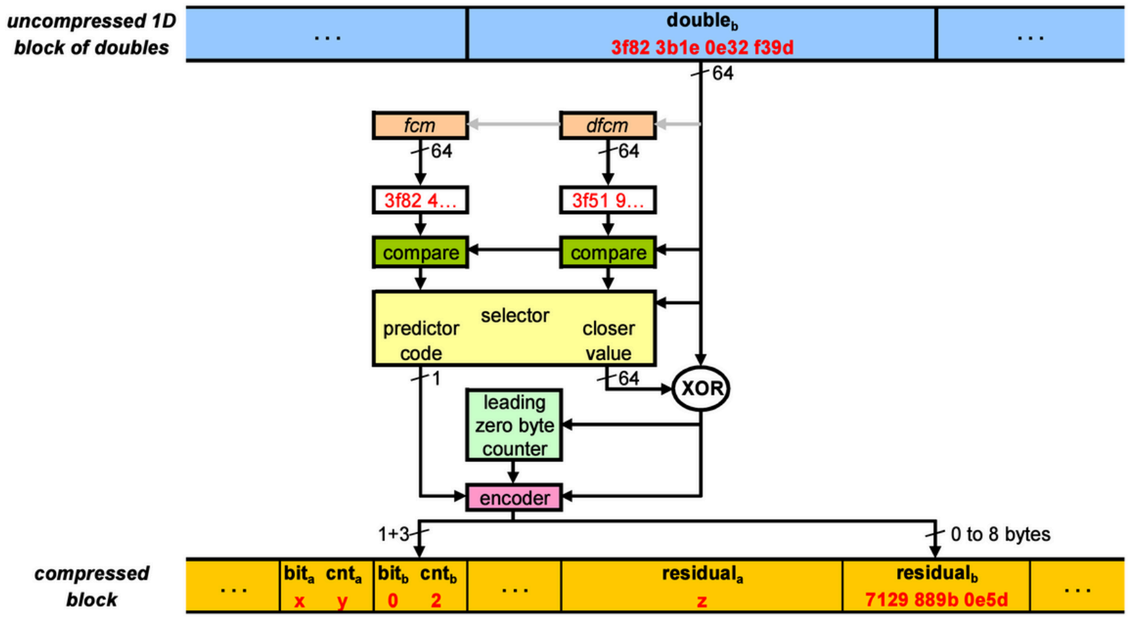
especially in the less significant bits of the mantissa.

XOR

VALUE	IEEE REPRESENTATION
3.25	01000000 00001010 00000000 00000000 00000000 00000000 00000000 00000000
8.625	01000000 00100001 01000000 00000000 00000000 00000000 00000000 00000000
XOR	00000000 00101011 01000000 00000000 00000000 00000000 00000000 00000000

leading zeros X bits trailing zeros

FPC: A HIGH-SPEED COMPRESSOR FOR DOUBLE-PRECISION FLOATING-POINT DATA



Main Features:

- XOR
- Hash-based predictions

GORILLA: A FAST, SCALABLE, IN-MEMORY TIME SERIES DATABASE

Two main algorithms:

1. Timestamps
2. Values

TIMESTAMPS COMPRESSION - 1

Algorithm 1 Gorilla time-stamps compression

1: Write the first time-stamp t_{-1} using 64 bits
2: Write the second time-stamp t_0 as a delta from t_{-1} using 14 bits
3: **for** each next time-stamp t_n **do**
4: $D = (t_n - t_{n-1}) - (t_{n-1} - t_{n-2})$
5: **if** D is zero **then**
6: Write a single *'0'* bit
7: **else if** $D \in [-63, 64]$ **then**
8: Write *'10'*
9: Write D using 7 bits
10: **else if** $D \in [-255, 256]$ **then**
11: Write *'110'*
12: Write D using 9 bits
13: **else if** $D \in [-2047, 2048]$ **then**
14: Write *'1110'*
15: Write D using 12 bits
16: **else**
17: Write *'1111'*
18: Write D using 32 bits
19: **end if**
20: **end for**

TIMESTAMPS COMPRESSION - 2

D	Tag bits	D bits	Total bits
0	*'0'*	0	1
[-63,64]	*'10'*	7	9
[-255,256]	*'110'*	9	12
[-2047,2048]	*'1110'*	12	16
>2048	*'1111'*	32	36

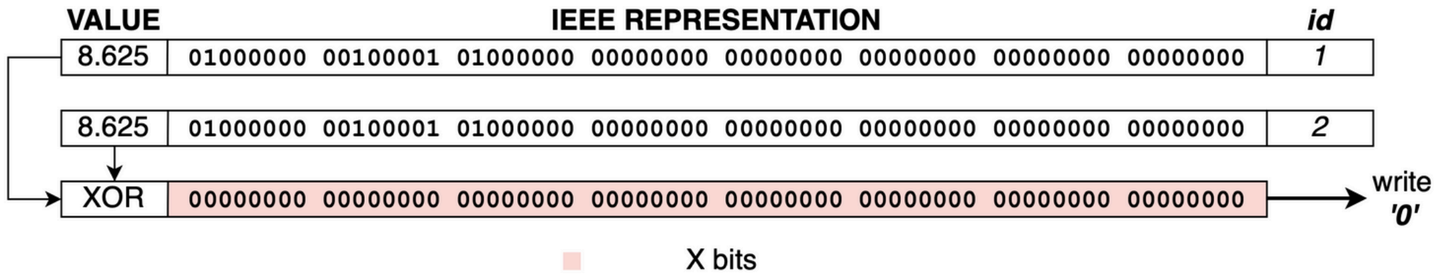
VALUE COMPRESSION

Algorithm 2 Gorilla values compression

1: Write the first value v_0 using 64 bits
2: $LZ_{n-1} \leftarrow 0$, $TZ_{n-1} \leftarrow 0$
3: **for** each next value v_n **do**
4: $X \leftarrow v_{n-1} \oplus v_n$
5: **if** X is zero **then**
6: Write a single *'0'* bit ▷ Case A
7: **else**
8: $LZ_n \leftarrow LeadingZeros(X)$
9: $TZ_n \leftarrow TrailingZeros(X)$
10: $len \leftarrow 64 - TZ_n - LZ_n$
11: **if** $(LZ_n = LZ_{n-1}) \wedge (TZ_n = TZ_{n-1})$ **then**
12: Write *'10'* ▷ Case B
13: $X \leftarrow X \gg TZ_n$
14: Write X using len bits
15: **else**
16: Write *'11'* ▷ Case C
17: Write LZ_n using 5 bits
18: Write len using 6 bits
19: Write X using len bits
20: **end if**
21: **end if**
22: $LZ_{n-1} \leftarrow LZ_n$, $TZ_{n-1} \leftarrow TZ_n$
23: **end for**

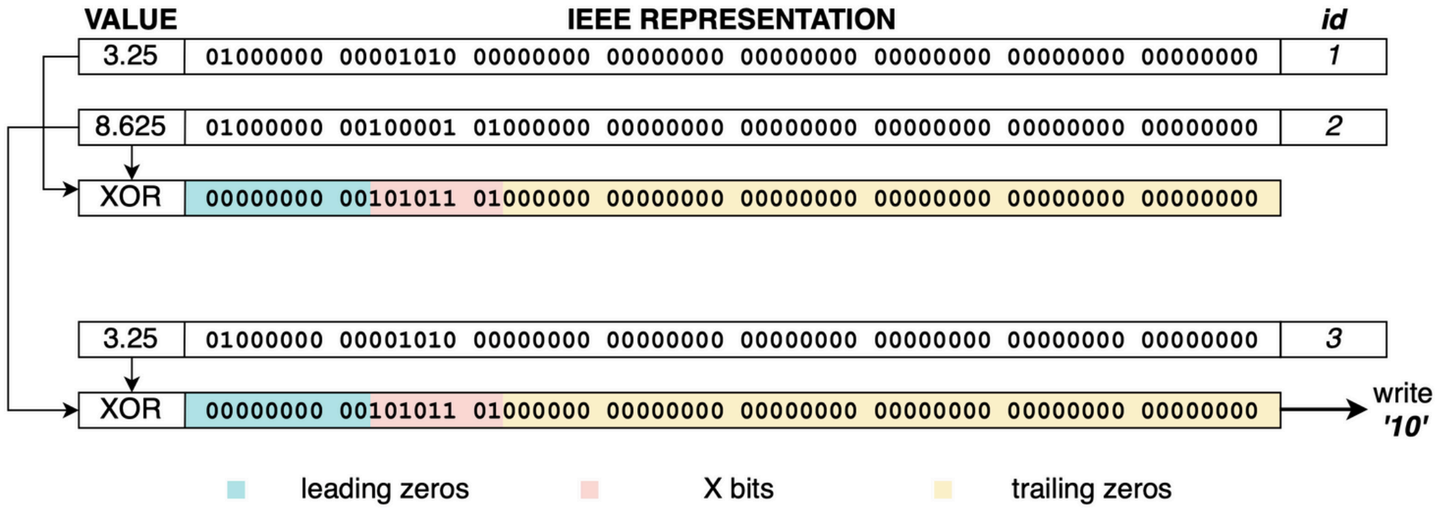
VALUE COMPRESSION - A

5: **if** X is zero **then**
6: Write a single *'0'* bit ▷ Case A



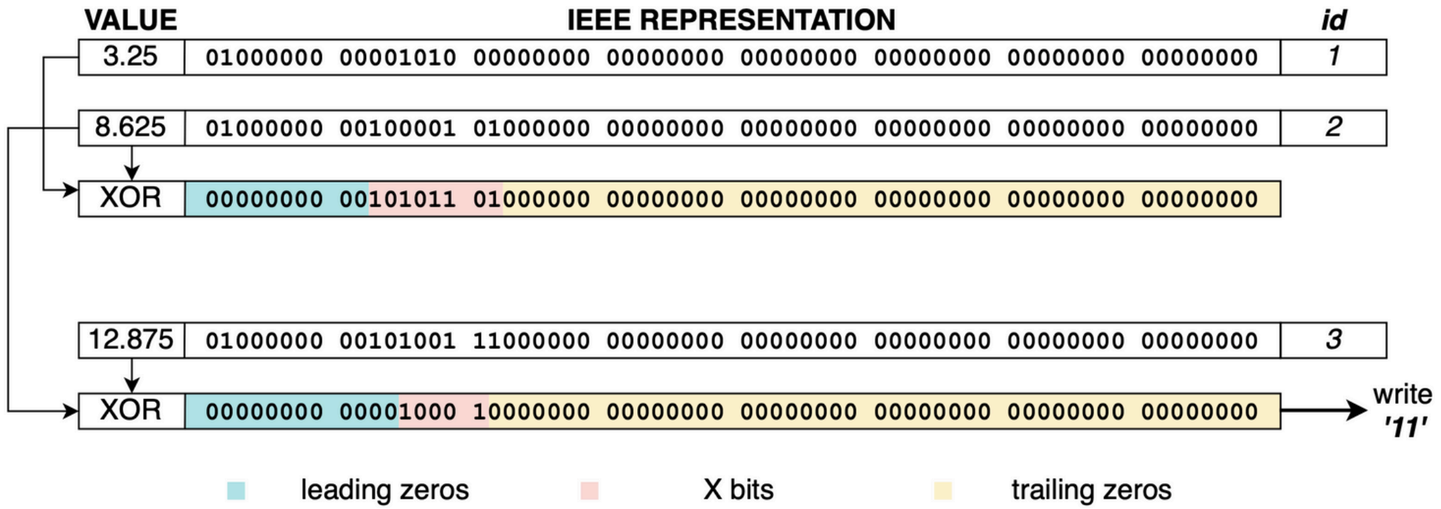
VALUE COMPRESSION - B

11: **if** $(LZ_n = LZ_{n-1}) \wedge (TZ_n = TZ_{n-1})$ **then**
12: Write *'10'* ▷ Case B
13: $X \leftarrow X \gg TZ_n$
14: Write X using len bits



VALUE COMPRESSION - C

```
15:     else
16:         Write '11'                          ▷ Case C
17:         Write LZ_n using 5 bits
18:         Write len using 6 bits
19:         Write X using len bits
```



TSXor

Main features:

1. Timestamps same as Gorilla
2. New byte-aligned compressor for floating point values

FLOATING POINTS: AN OBSERVATION

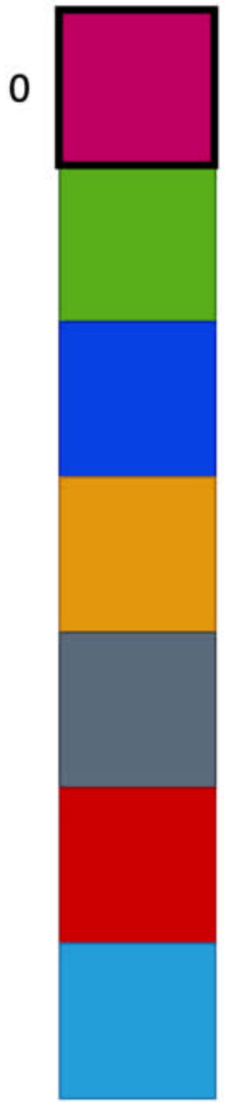
CLOSE VALUES

```
11.3    01000000 00100110 10011001 10011001 10011001 10011001 10011001 10011010

11.5    01000000 00100111 00000000 00000000 00000000 00000000 00000000 00000000
```

FAR VALUES

```
-6.6    11000000 00011010 01100110 01100110 01100110 01100110 01100110 01100110

-3.8    11000000 00001110 01100110 01100110 01100110 01100110 01100110 01100110
```

TSXor: COMPRESSION - 1

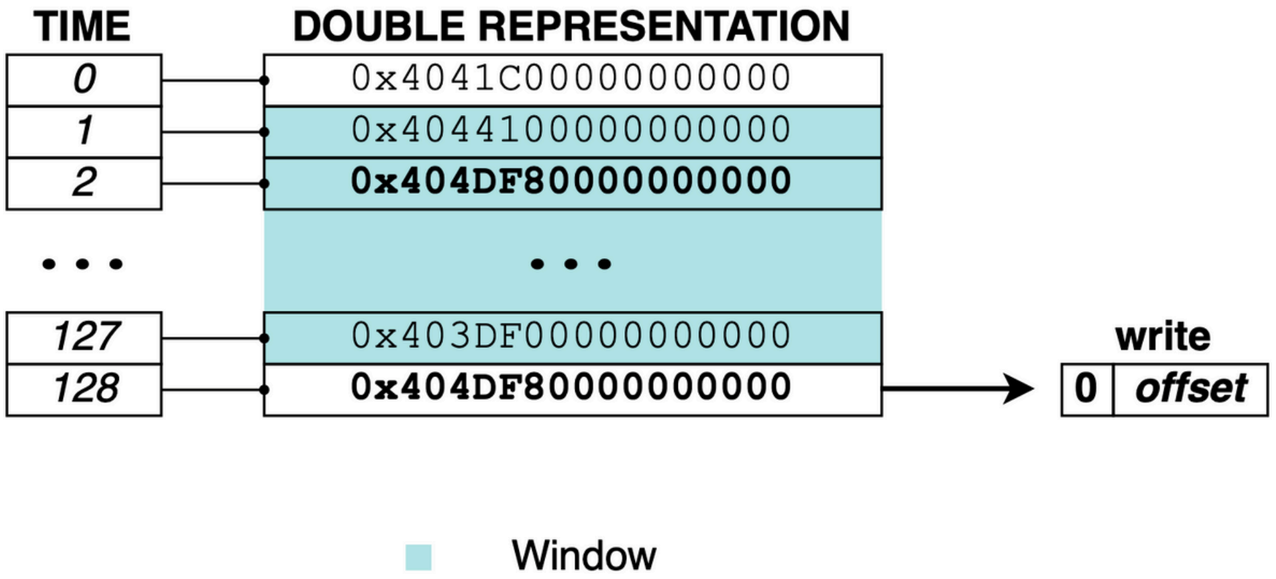


1: **for** each first value $v_0 \in feature_k$ **do**
2: **Write** v_0 using 64 bits.
3: add v_0 in $windows_k$
4: **end for**
5: **for** each next value $v_i \in feature_k$ **do**
6: **Compress** v_i
7: add v_i in $windows_k$
8: **end for**

TSXor: COMPRESSION - 2

1: **if** $v_i \in window_k$ **then** ▷ Case A
2: $offset \leftarrow getIndexOf(v_i)$
3: **Write** (*'0'* $\sqcup$ $offset$) using 1 byte
4: **else**
5: $p \leftarrow getCandidate(v_i, k)$
6: $X \leftarrow v_i \oplus p$
7: $LZ \leftarrow LeadingZerosBytes(X)$
8: $TZ \leftarrow TrailingZerosBytes(X)$
9: **if** $LZ + TZ > 1$ **then** ▷ Case B
10: $offset \leftarrow getIndexOf(p)$
11: **Write** (*'1'* $\sqcup$ $offset$) using 1 byte
12: $len_X = 8 - LZ - TZ$
13: $head = TZ \ll 4 \mid len_X$
14: **Write** $head$ using 1 byte
15: $X = X \gg TZ$
16: **Write** X using len_X bytes
17: **else** ▷ Case C
18: **Write** *255* using 1 byte
19: **Write** v_i using 8 bytes
20: **end if**
21: **end if**

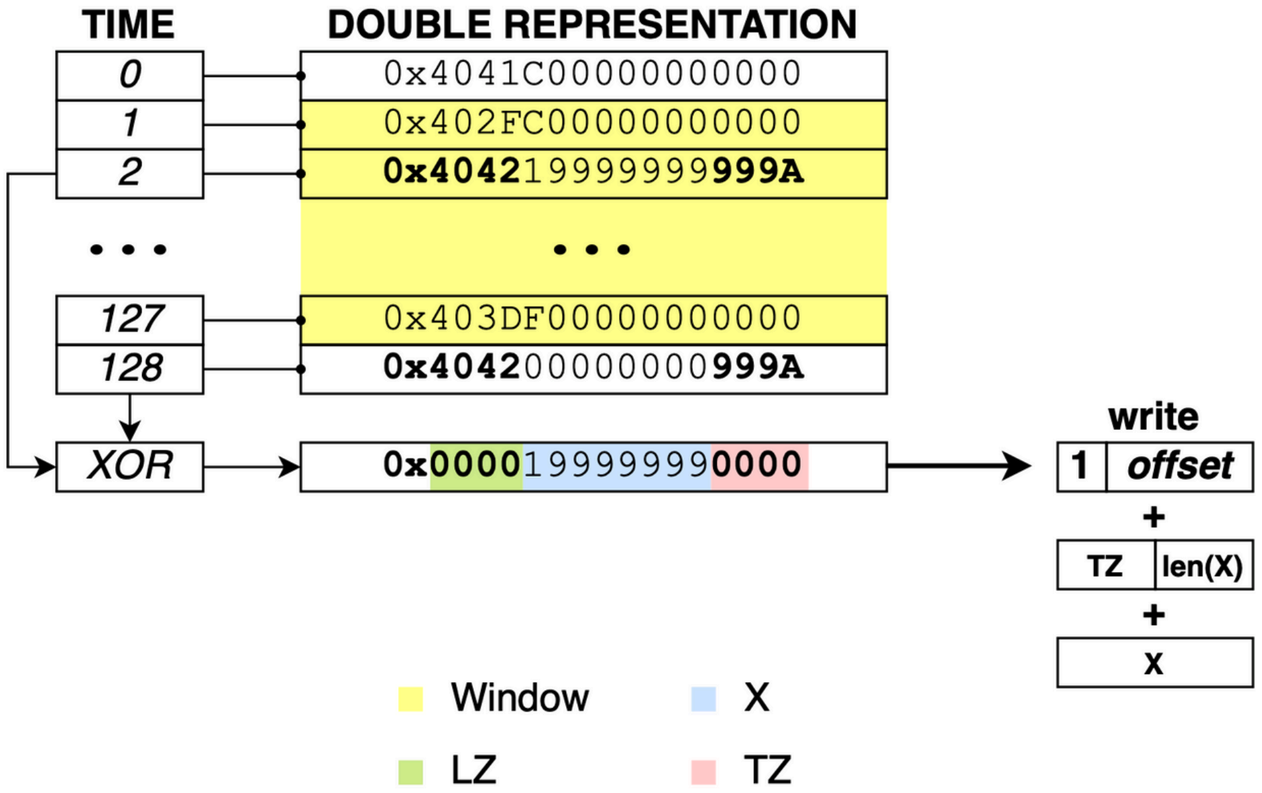
TSXor: CASE A


TIME
DOUBLE REPRESENTATION
0
1
2
0x4041C00000000000
0x4044100000000000
0x404DF80000000000
127
128
0x403DF00000000000
0x404DF80000000000
write
0
offset
Window

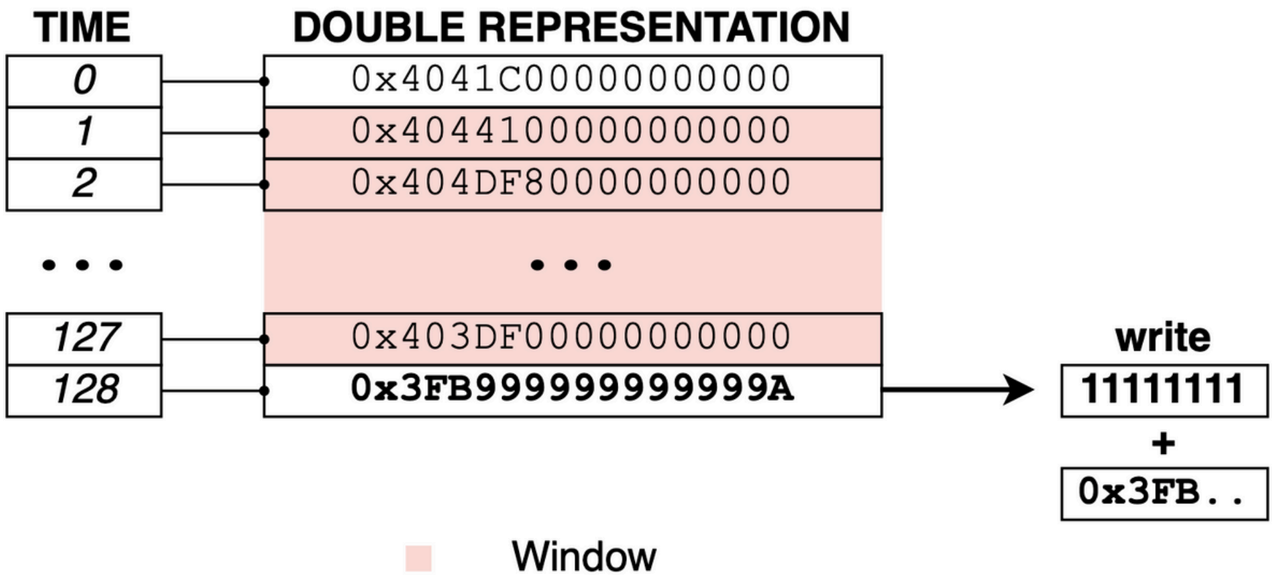
TSXor: getCandidate

1: **procedure** GETCANDIDATE(v, k)
2: **for** $i \leftarrow 1, window_k.size()$ **do**
3: $A[i] = v \ \oplus \ window_k[i]$
4: $B[i] = LeadingZerosBits(A[i]) + TrailingZerosBits(A[i])$
5: **end for**
6: **return** $max(B)$
7: **end procedure**

TSXor: CASE B


TIME
DOUBLE REPRESENTATION
0
1
2
0x4041C00000000000
0x402FC00000000000
0x4042199999999999A
127
128
0x403DF00000000000
0x4042000000000999A
XOR
0x0000199999999000
write
1
offset
+
TZ
len(X)
+
X
Window
X
LZ
TZ

TSXor: CASE C


TIME
DOUBLE REPRESENTATION
0
1
2
0x4041C00000000000
0x4044100000000000
0x404DF80000000000
127
128
0x403DF00000000000
0x3FB999999999999A
write
11111111
+
0x3FB..
Window

TSXor: DECOMPRESSION

1: $head \leftarrow$ **Read** 1 byte
2: **if** $head < 128$ **then** ▷ Case A
3: $v_i \leftarrow window_k[head]$
4: **else if** $head == 255$ **then** ▷ Case C
5: $v_i \leftarrow$ **Read** 8 byte
6: **else** ▷ Case B
7: $block \leftarrow$ **Read** 1 byte
8: $TralingZeros_{bytes} \leftarrow$ leading 4 bits of $block$
9: $len_X \leftarrow$ trailing 4 bits of $block$
10: $xor \leftarrow$ **Read** len_X bytes
11: $xor \leftarrow xor \ll (8 * TralingZeros_{bytes})$
12: $offset \leftarrow$ remove first bit from head
13: $v_i \leftarrow xor \ \oplus \ window_k[offset]$
14: **end if**
15: **return** v_i

EXPERIMENTAL RESULTS: DATASETS

	Rows	Features			Unique Float		Unique Int	
	#	#	Float	Int	Avg	%	Avg	%
AMPds2	14629292	11	5	6	281	0,002	1342944	9,180
Bar Crawl	14057564	4	3	1	2332836	16,595	13	0,000
Max-Planck	473353	32	32	0	2559	0,541	-	-
Kinect	733432	80	80	0	301239	41,073	-	-
Oxford-Man	143397	19	18	1	120869	84,290	31	0,022
PAMAP	3127602	44	44	0	11956	0,382	-	-
UCI Gas	2841954	18	18	0	17776	0,635	-	-

COMPRESSION SPEED (MB/s)

	FPC	Gorilla	TSXor
AMPds2	339,3	**703,7**	66,6
Bar Crawl	423,7	**466,5**	28,7
Max-Planck	313,4	**870,6**	51,7
Kinect	166,3	**696,1**	17,1
Oxford-Man	170,3	**630,3**	15,4
PAMAP	181,6	**521,4**	45,0
UCI Gas	286,9	**654,3**	21,9

DECOMPRESSION SPEED (MB/s)

	FPC	Gorilla	TSXor
AMPds2	411,3	666,5	**1.173,6**
Bar Crawl	436,1	447,4	**709,7**
Max-Planck	355,3	858,7	**1.057,0**
Kinect	287,1	635,7	**665,5**
Oxford-Man	221,8	573,7	**604,5**
PAMAP	223,9	487,4	**949,3**
UCI Gas	454,9	578,4	**642,4**

COMPRESSION RATIO

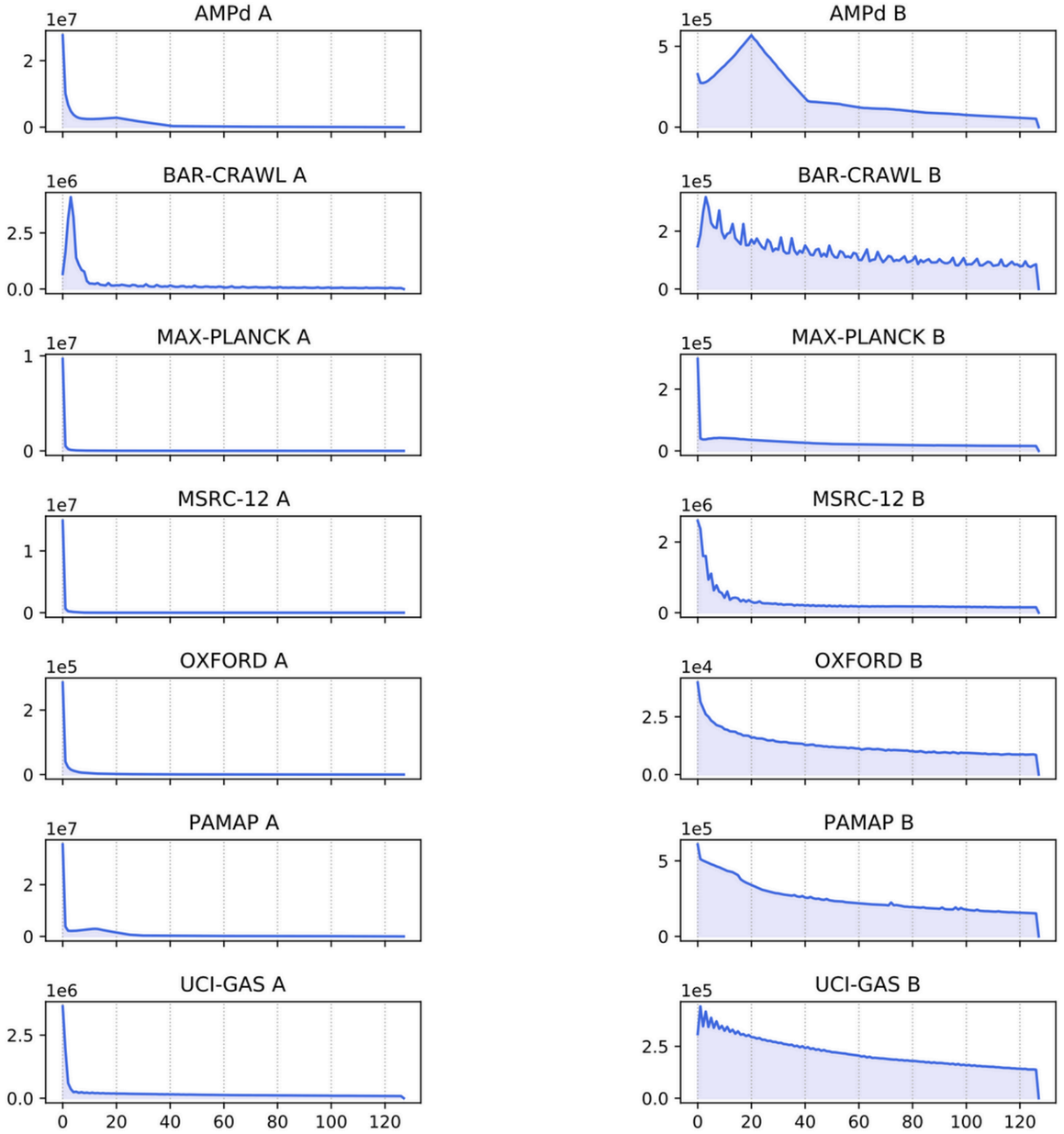
	FPC	Gorilla	TSXor
AMPds2	1.10x	2.03x	**6.39x**
Bar Crawl	1.20x	1.44x	**2.36x**
Max-Planck	1.06x	2.97x	**4.84x**
Kinect	1.09x	**1.41x**	1.37x
Oxford-Man	1.06x	1.28x	**1.30x**
PAMAP	1.01x	1.38x	**4.85x**
UCI Gas	1.19x	1.23x	**3.50x**

GORILLA vs TSXor

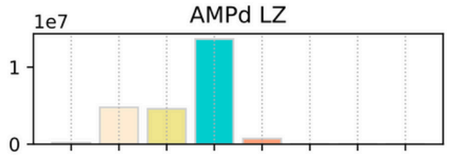
	A (%)	B (%)		C (%)	
	1 bit		bytes		bytes
AMPds2	17,17	82,8	4,76	0,03	4,18
Bar Crawl	1,19	70,45	5,77	28,35	8,44
Max-Planck	63,84	36,1	7,31	0,06	7,08
Kinect	25,35	70,4	7,39	4,25	8,33
Oxford-Man	10,46	87,54	7,11	2,00	8,15
PAMAP	25,85	51,04	7,41	23,11	8,33
UCI Gas	7,12	92,65	7,20	0,22	8,32
Average	**21,57**	**70,14**	**6,71**	**8,29**	**7,55**

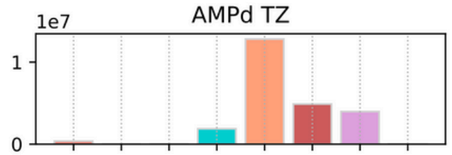
	A (%)	B (%)		C (%)
	1 byte		bytes	9 bytes
AMPds2	84,87	14,87	3,19	0,26
Bar Crawl	50,53	28,25	5,53	21,22
Max-Planck	77,93	21,94	4,15	0,13
Kinect	28,01	62,95	7,66	9,04
Oxford-Man	17,44	59,44	6,94	23,12
PAMAP	75,95	23,13	3,63	0,92
UCI Gas	45,36	54,63	3,57	0,01
Average	**54,30**	**37,89**	**4,95**	**7,81**

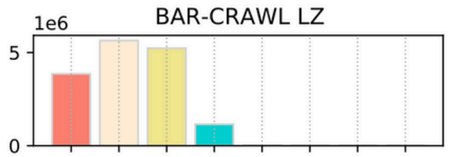
OFFSET: CASE A vs CASE B

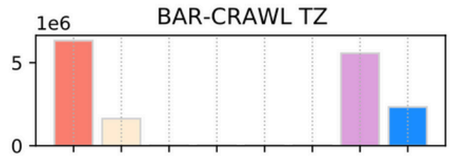


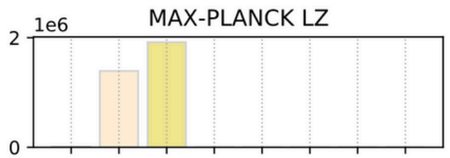
LEADING AND TRAILING ZERO BYTES

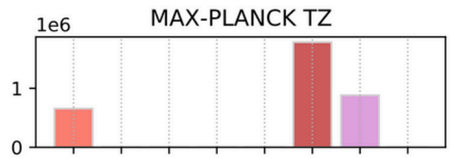


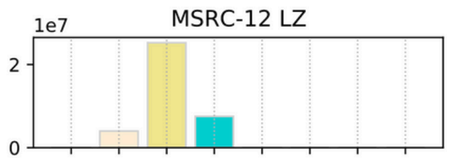


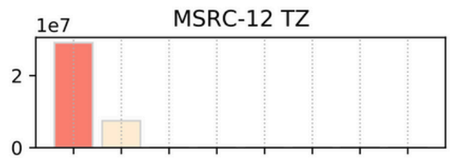


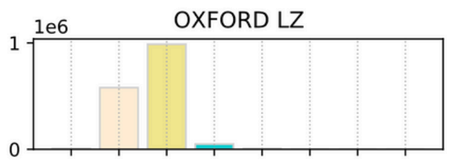


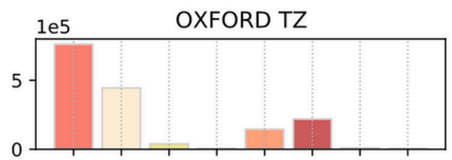


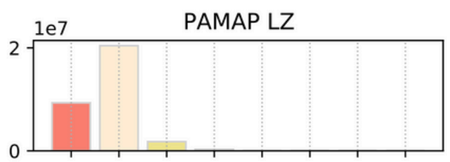


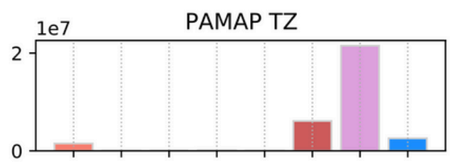


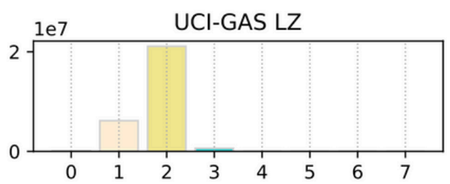


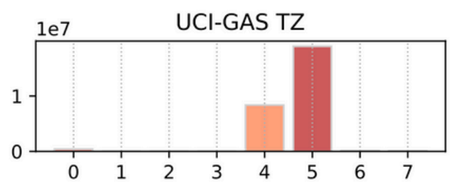






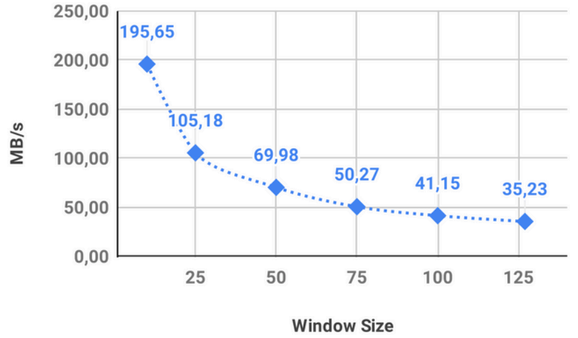




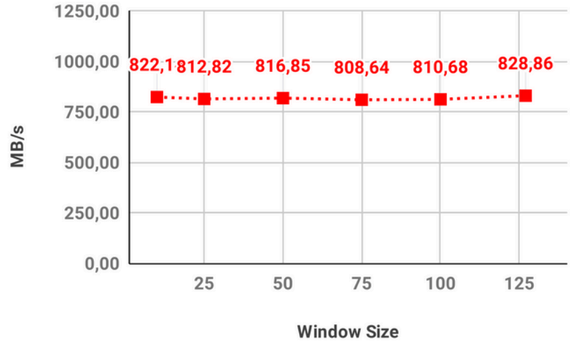


WINDOW SIZE

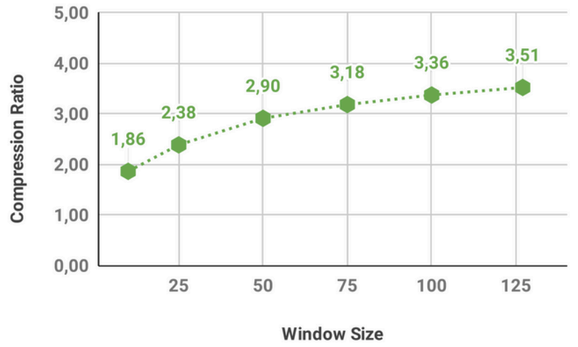
COMPRESSION SPEED (MB/s)						
Name	**10**	**25**	**50**	**75**	**100**	**127**
AMPds2	194,25	127,81	109,97	87,23	75,63	66,59
Bar Crawl	215,72	100,97	61,14	42,38	34,32	28,74
Max-Planck	295,95	164,22	106,93	75,29	61,29	51,74
Kinect	164,32	76,01	43,51	27,98	21,34	17,14
Oxford-Man	156,54	69,70	39,00	25,12	17,98	15,43
PAMAP	193,28	125,82	85,39	63,18	52,20	45,05
UCI Gas sensor array	149,46	71,73	43,91	30,71	25,30	21,93
AVERAGE	195,65	105,18	69,98	50,27	41,15	35,23



DECOMPRESSION SPEED (MB/s)						
Name	**10**	**25**	**50**	**75**	**100**	**127**
AMPds2	860,49	962,26	1134,77	1142,57	1155,33	1173,65
Bar Crawl	849,04	779,14	728,31	699,68	700,32	709,68
Max-Planck	1038,54	1018,75	1005,76	1012,72	1032,36	1057,00
Kinect	803,41	756,42	719,01	694,06	674,99	665,47
Oxford-Man	739,64	665,97	619,18	595,19	569,24	604,54
PAMAP	748,87	866,18	899,15	905,57	923,00	949,28
UCI Gas sensor array	715,02	641,00	611,77	610,69	619,52	642,40
AVERAGE	822,14	812,82	816,85	808,64	810,68	828,86



COMPRESSION RATIO						
Name	**10**	**25**	**50**	**75**	**100**	**127**
AMPds2	2,74	4,17	5,67	6,05	6,26	6,39
Bar Crawl	1,62	1,81	2,02	2,16	2,27	2,36
Max-Planck	2,80	3,22	3,79	4,23	4,56	4,84
Kinect	1,32	1,33	1,34	1,35	1,36	1,37
Oxford-Man	1,18	1,23	1,26	1,28	1,29	1,30
PAMAP	1,87	3,02	3,79	4,27	4,60	4,85
UCI Gas sensor array	1,45	1,87	2,46	2,89	3,22	3,50
AVERAGE	1,86	2,38	2,90	3,18	3,36	3,51



RASPBERRY PI

	Compr. Speed (MB/s)			*Decomp. Speed (MB/s)*		
	FPC	**Gorilla**	**TSXor**	**FPC**	**Gorilla**	**TSXor**
AMPds2	-	-	-	-	-	-
Bar Crawl	-	-	-	-	-	-
Max-Planck	29.1	**75.5**	5.8	30.3	84.3	**123.0**
Kinect	-	**53.8**	-	-	49.0	-
Oxford-Man	16.8	**45.7**	1.6	21.6	44.3	**72.1**
PAMAP	-	-	4.3	-	-	**95.0**
UCI Gas	24.6	**39.8**	2.2	39.8	39.8	**93.3**

FUTURE WORK

- Improve TSXor
- Machine Learning

THANKS!