on

(mostly lossless)

compression induced learning

Israel Statistics and Data Science Annual Conference

June 2026

Tsachy Weissman
Stanford University

my work

- information theory
- information processing
- data compression

what this talk is not about

- compression of neural networks
- compression of gradients while training
- KV cache compression
- etc.

outline

- apology
- lossless compression landscape
- compressors as Sequential Probability Assignments (SPAs)
- the LZ78 SPA: classical and modern twists
- example applications: classification and “Gen AI”
- LZ78 as a data generator
- the interplay between lossy compression and denoising
- textual transform coding
- some takeaways

(will try to accommodate a diversity of backgrounds)

apology in advance

there will be no justice to the (classical and recent) literature but it is served in:

- [SW26]
- [Omri et al. 25]
- [Gorle et al. 26]
- [SDHW26]
- [SOW26]
- [Ding et al. 26]
- [W23]

lossless compression landscape in practice

https://www.mattmahoney.net/dc/text.html

Large Text Compression Benchmark

Matt Mahoney
Last update: Oct. 8, 2024. history

This competition ranks lossless data compression programs by the compressed size (including the size of the decompression program) of the first 10^9 bytes of the XML text dump of the English version of Wikipedia on Mar. 3, 2006. About the test data.

The goal of this benchmark is not to find the best overall compression program, but to encourage research in artificial intelligence and natural language processing (NLP). A fundamental problem in both NLP and text compression is modeling: the ability to distinguish between high probability strings like *recognize speech* and low probability strings like *reckon eyes peach*. Rationale.

This is an open benchmark. Anyone may contribute results. Please read the rules first.

Open source compression improvements to this benchmark with certain hardware restrictions may be eligible for the Hutter Prize.

Benchmark Results

Compressors are ranked by the compressed size of enwik9 (10^9 bytes) plus the size of a zip archive containing the decompresser. Options are selected for maximum compression at the cost of speed and memory. Other data in the table does not affect rankings. This benchmark is for informational purposes only. There is no prize money for a top ranking. Notes about the table:

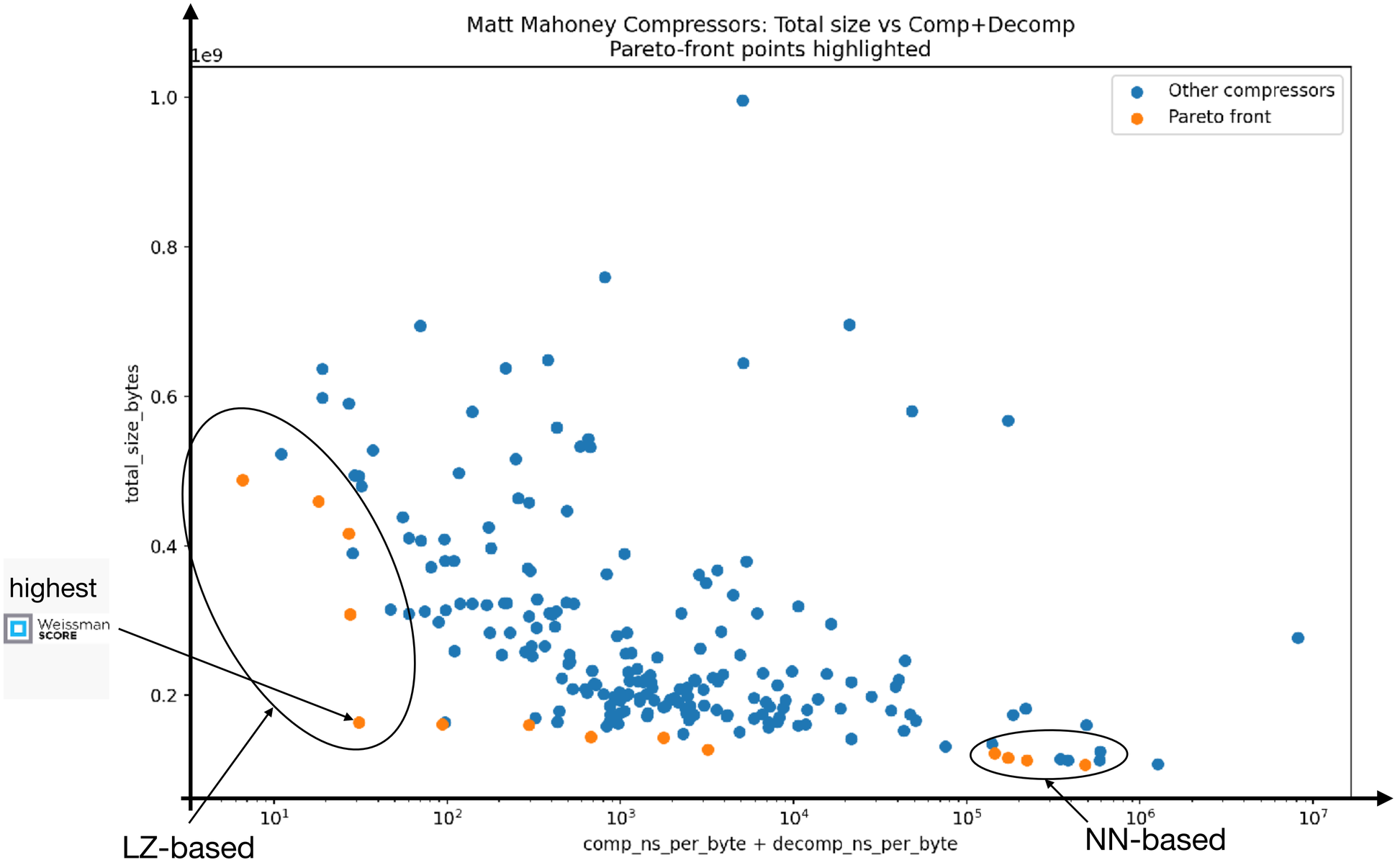
- **Program:** The version believed to give the best compression. A | denotes a combination of 2 programs.
- **Compression options:** selected for what I believe gives the best compression.
- **enwik8:** compressed size of first 10^8 bytes of enwik9. This data is used for the Hutter Prize, and is also ranked here but has no effect on this ranking.
- **enwik9:** compressed size of first 10^9 bytes of enwiki-20060303-pages-articles.xml.
- **decompresser size:** size of a zip archive containing the decompression program (source code or executable) and all associated files needed to run it (e.g. dictionaries). A letter following the size has the following meaning:
  - x = executable size.
  - s = source code size (if available and smaller).
  - d = size of a separate decompression program (separate from compression). For self extracting archives (SFX), the size is 0 because the decompresser and compressed data are combined into one file.

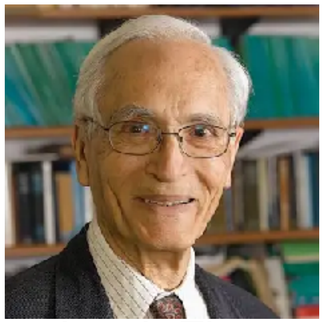
  For testing, if no zip file is supplied I create archives using InfoZIP 2.32 -9. (Prior to July 1, 2008 I used 7zip 4.32 -tzip -mx=9).
- **Total size:** total size of compressed enwik9 + decompresser size, ranked smallest to largest.
- **Comp:** compression rate in nanoseconds per byte on the largest file tested (e.g. seconds for enwik9). Speed is approximate and has no effect on ranking. A ~ means "very approximate". Not all tests are done on the same computer. Times reported are the smaller of process time (summed over processors if multi-threaded) or real time as measured with timer). If there is no note then the program was tested on a Compaq Presario 5440, 2.188 GHz, Athlon-64 3500+ in 32 bit Windows XP. An underlined time means that no better compressor is faster.
- **Decomp:** decompression time as above. If blank, decompression was not tested yet and ranking is pending verification that the output is identical. An underlined time means that no better compressor is faster.
- **Mem:** approximate memory used for compression in MB. Decompression uses the same or possibly less. There is some ambiguity whether a megabyte means 10^6 bytes or 2^{20} bytes. The approximation is course enough that it doesn't matter. I use peak memory as measured with Windows Task Manager during compression (so if you really want to know, 1 MB = 1,024,000 bytes :) Memory does not include swap or temporary files. An underlined value means that no better compressor uses less memory.
- **Alg:** compression algorithm, referring to the method of parsing the input into symbols (strings, bytes, or bits) and estimating their probabilities (modeling) for choosing code lengths. Symbols may be arithmetic coded (fractional bit length for best compression), Huffman coded (bit aligned for speed), or byte aligned as a preprocessing step.
  - Dict (Dictionary). Symbols are words, coded as 1 or 2 bytes, usually as a preprocessing step.
  - LZ (Lempel Ziv). Symbols are strings.
    - LZ77: repeated strings are coded by offset and length of previous occurrence.
    - LZW (LZ Welch): repeats are coded as indexes into a dynamically built dictionary.
    - ROLZ (Reduced Offset LZ): LZW with multiple small dictionaries selected by context.
    - LZP (LZ predictive): ROLZ with a dictionary size of 1.
  - on (Order-n, e.g. o0, o1, o2...): symbols are bytes, modeled by frequency distribution in context of last n bytes.
  - PPM (Prediction by Partial Match): order-n, modeled in longest context matched, but dropping to lower orders for byte counts of 0.
  - SR (Symbol Ranking): order-n, modeled by time since last seen.
  - BWT (Burrows Wheeler Transform): bytes are sorted by context, then modeled by order-0 SR.
  - ST (Sort Transform): BWT using stable sort with truncated string comparison.

#	Program	Compression Options	Compressed size enwik8	Compressed size enwik9	Decompresser size (zip)	Total size enwik9+prog	Time (ns/byte) Comp	Time (ns/byte) Decomp	Mem	Alg	Note
1.	nncp v3.2		14,915,298	106,632,363	628,955 xd	107,261,318	241871	238670	7600	Tr	88
2.	cmix v21	-t	14,623,723	107,963,380	281,307 sd	108,244,767	622949	630442	30950	CM	83
3.	fx2-cmix			110,351,665	0 xd	110,351,665		272872	6811	CM	97
4.	tensorflow-compress v4		15,905,037	113,542,413	55,283 sd	113,597,696	291394	290803	45360	LSTM	94
5.	cmix-hp 10 Jun 2021		15,957,339	113,712,798	0 xd	113,712,798	189420	194200	6873	CM	89
6.	fast-cmix			113,746,218	0 xd	113,746,218		121971	8027	CM	99
7.	starlit 31 May 2021		15,215,107	114,951,433	0 xd	114,951,433	173953	171682	10233	CM	89
8.	phda9 1.8		15,010,414	116,544,849	42,944 xd	116,587,793	86182	86305	6319	CM	83
9.	paq8px_v206fix1	-12L	15,849,084	124,696,418	402,949 s	125,099,359	281916	294847	28151	CM	93
10.	durilca'kingsize	-m13000 -o40 -t2	16,209,167	127,377,411	407,477 xd	127,784,888	1398	1797	13000	PPM	31
11.	cmve 0.2.0	-m2,3,0x7fed7dfd	16,424,248	129,876,858	307,787 x	130,184,645	1140801		19963	CM	81
12.	paq8hp12any	-8	16,230,028	132,045,026	330,700 x	132,375,726	37660	37584	1850	CM	41
13.	drt|emma 1.23		16,523,517	134,164,521	1,358,251 xd	135,522,772	73806	67097	3800	CM	81
14.	zpaq 6.42	-m s10.0.5fmax6	17,855,729	142,252,605	4,760 sd	142,257,365	6699	14739	14000	CM	61
15.	drt|lpaq9m	9	17,964,751	143,943,759	110,579 x	144,054,338	868	898	1542	CM	41
16.	mcm 0.83	-x11	18,233,295	144,854,575	79,574 s	144,934,149	394	281	5961	CM	72
17.	nanozip 0.09a	-cc -m32g -p1 -t1 -nm	18,594,163	148,545,179	783,642 x	149,328,821	1149	1141	32000	CM	74
18.	xwrt 3.2	-l14 -b255 -m96 -s -e40000 -f200	18,679,742	151,171,364	52,569 s	151,223,933	2537	2328	1691	CM	
19.	fp8 v3	-8	18,438,169	153,188,176	50,068 s	153,238,244	20605	22593	1192	CM	26
20.	WinRK 3.03	pwcm +td 800MB SFX	18,612,453	156,291,924	99,665 xd	156,391,589	68555		800	CM	10
21.	paq8pstr J	-m1700 -o16	19,055,092	157,007,383	42,019 x	157,049,402	3574	~3600	1700	PPM	
22.	zcm 0.93	-m8 -t1	19,572,089	159,135,549	227,659 x	159,363,208	421	411	3100	CM	48
23.	slim 23d	-m1700 -o12	19,077,276	159,772,839	69,453 x	159,842,292	5232	~5400	1700	PPM	
24.	bsc-m03 0.4.0	-b1000000000	20,293,393	160,258,936	105,456 xd	160,364,392	160	135	13000	BWT	96
25.	bwmonstr 0.02		20,307,295	160,468,597	69,401 x	160,537,998	331801	156147	590	BWT	30
26.	nanozipltcb 0.09		20,537,902	161,581,290	133,784 x	161,715,074	64	30	3350	BWT	40
27.	M03 1.1b	1000000000	20,710,197	163,667,431	50,468 x	163,717,899	457	406	5735	BWT	52
28.	bcm 2.03	-b1000k-	20,738,630	163,646,387	125,866 x	163,772,253	63	34	4096	BWT	98
29.	glza 0.10.1	-x -p3	20,356,097	163,768,203	69,935 s	163,838,138	8184	11.9	8205	Dict	67
30.	bsc 3.25	-b1000 -e2	20,786,794	163,884,462	74,297 sd	163,958,759	23	8	5000	BWT	96
31.	bbb	m1000	20,847,290	164,032,650	11,227 s	164,043,877	4574	2619	1401	BWT	
32.	pcompress 3.1	-c libbsc -l14 -s1000m	20,769,968	163,391,884	1,370,611 x	164,762,495	359	74	3300	BWT	48
33.	paq9a	-9	19,974,112	165,193,368	13,749 s	165,207,117	3997	4021	1585	CM	
34.	uda 0.300		19,393,460	166,272,261	11,264 x	166,283,525	25282	25174	180	CM	
35.	BWTmix v1	c10000	20,608,793	167,852,106	9,565 x	167,861,671	1794	690	5000	BWT	49
36.	lrzip 0.612	-z -L 9 -p 1	19,847,690	169,318,794	99,363 x	169,418,157	2987	2920	2700	CM	33
37.	cm4_ext		20,188,848	170,566,799	204,782 x	170,771,581	4123	4130	1906	CM	26
38.	M1x2 v0.6	7 enwik7.txt	20,723,056	172,212,773	38,467 s	172,251,240	711	715	1051	CM	26
39.	cmm4 v0.1e	96	20,569,034	172,669,955	31,314 x	172,701,269	2052	2056	1321	CM	
40.	lstm-compress v3		20,318,653	173,874,407	144,567 s	174,018,974	92342	91876	9	LSTM	83
41.	ccmx 1.30	7	20,857,925	174,142,092	15,014 x	174,157,106	1313	1338	1332	CM	
42.	bit 0.7	-p=5	20,823,204	174,425,039	62,493 x	174,487,532	2050	2100	663	CM	26
43.	ncomp 2.00	-mw -M320m	21,103,670	174,388,351	172,531 x	174,560,882	473	399	1643	BWT	26
44.	epmopt|epm r9	-m800 -n20 --fixedorder:12	19,713,502	174,817,424	141,101 x	174,958,525	3179	3376	600	PPM	
45.	WinUDA 2.91	mode 3 (194 MB)	20,332,366	174,975,730	17,203 x	174,992,933	23610	23473	194	CM	
46.	dark 0.51	-b333mf	21,169,819	175,471,417	34,797 x	175,506,214	533	453	1692	BWT	
47.	FreeArc 0.40pre-4	-mppmd:1012m:o13:r1	20,931,605	175,254,732	748,202 x	176,002,934	1175	1216	1046	PPM	
48.	hook v1.4	1700	21,990,502	176,648,663	37,004 x	176,685,667	741	695	1777	DMC	26
49.	7zip 4.46a	-m0=ppmd:mem=1630m:o=10 ...	21,197,559	178,965,454	0 xd	178,965,454	503	546	1630	PPM	23
50.	rings 2.5	-m8 -t1	20,873,959	178,747,360	240,523 x	178,987,883	260	163	2518	BWT	48
51.	pimple2		20,871,457	180,251,530	78,642 x	180,330,172	18474	17992	128	CM	
52.	ash 04a	/m700 /o10	19,963,105	180,735,542	11,137 x	180,746,679	6108	5853	700	CM	
	https://www.mattmahoney.net/dc/text.html#1165		22,729,140	180,732,702	19,809 s	180,752,591	1151	2444	5000	CM	71

#	Program	Compression Options	Compressed size enwik8	Compressed size enwik9	Decompresser size (zip)	Total size enwik9+prog	Time (ns/byte) Comp	Time (ns/byte) Decomp	Mem	Alg	Note
154.	gzip 1.3.5	-9	36,445,248	322,591,995	38,801 x	322,630,796	101	17	1.6	LZ77	
155.	Info-ZIP 2.3.1	-9	36,445,373	322,592,120	57,583 x	322,649,703	104	35	0.1	LZ77	
156.	pkzip 2.0.4	-ex	36,556,552	323,403,526	29,184 xd	323,432,710	171	50	2.5	LZ77	
157.	jar (Java) 0.98-gcc	cvfM	36,520,144	323,747,582	19,054 x	323,766,636	118	95	1.2	LZ77	
158.	PeaZip	better, no integrity check	36,580,548	323,884,274	561,079 x	324,445,353	243	243	8	LZ77	28
159.	arj 3.10	-m1	37,091,317	328,553,982	143,956 x	328,697,938	262	67	3	LZ77	26
160.	lzgt3a		37,444,440	334,405,713	4,387 xd	334,410,100	1581	2886	2	LZ77	
161.	pucrunch	-d -c0	39,199,165	350,265,471	34,359 s	350,299,830	2649	463	2	LZ77	
162.	packARC v0.7RC11	-sfx -np	38,375,065	361,905,425	0 xd	361,905,425	1359	1486	23	CM	
163.	urban		38,215,763	362,677,440	4,280 s	362,681,720	381	450	6	o2	48
164.	lzop v1.01	-9	41,217,688	366,349,786	54,438 x	366,404,224	289	12	1.8	LZ77	
165.	lzw 0.2		41,960,994	367,633,910	671 s	367,634,581	3597	31	18	LZW	
166.	NTCompressor v1.0		41,295,546	370,152,396	3,620 x	370,156,016	173	117	74	LZ77	26
167.	lz4x 1.02	c4	41,958,112	372,068,437	48,609 x	372,117,046	79	1.4	114	LZ77	68
168.	arbc2z		38,756,037	379,054,068	6,255 sd	379,060,323	2659	2674	68	PPM	
169.	lz4 v1.2	-c2	42,870,164	379,999,522	49,128 x	380,048,650	91	6	20	LZ77	26
170.	lzss 0.02	cx	42,874,387	380,192,378	48,114 x	380,240,492	107	2.3	145	LZSS	63
171.	xdelta 3.0u	-9	44,288,463	380,302,725	107,985 x	380,410,710	1021	30	47	LZ77	
172.	brieflz 1.1.0		43,300,800	390,122,722	14,907 s	390,137,629	21	7.5	3	LZ77	48
173.	mtari 0.2		41,655,528	397,232,608	4,156 s	397,236,764	80	99	18	CM	26
174.	lzf 1.02	cx	45,198,298	406,805,983	48,359 x	406,854,342	68	2.2	151	LZ77	68
175.	srank 1.1	-C8	43,091,439	409,217,739	6,546 x	409,224,285	51	45	2	SR	
176.	QuickLZ 1.30b	(quick3)	46,378,438	410,633,262	44,202 x	410,677,464	48	12	3	LZ77	
177.	stz 0.7.2	-c2	47,192,312	416,524,596	41,941 x	416,566,537	14	13	3	LZ77	26
178.	compress 4.3d		45,763,941	424,588,663	16,473 x	424,605,136	103	70	1.8	LZW	
179.	lzrw3-a		48,009,194	438,253,704	4,750 x	438,258,454	38	17	2	LZ77	
180.	fcm1		45,402,225	447,305,681	1,116 s	447,306,797	228	261	1	CM1	
181.	runcoder1		46,883,939	458,125,932	5,488 s	458,131,420	140	156	4	o1	26
182.	data-shrinker 23Mar2012		51,658,517	459,825,318	3,706 s	459,829,024	14	4	2	LZ77	26
183.	lzwc_bitwise 0.7		46,639,414	463,884,550	4,183 x	463,888,733	123	134	71	LZW	26
184.	exdupe 0.3.3		53,717,422	478,788,378	1,092,986 x	479,881,364	27	5	1000	LZ77	48
185.	lzv 0.1.0		54,950,847	488,436,027	10,385 x	488,446,412	4	2.6	3	LZ77	48
186.	FastLZ Jun 12 2007		54,658,924	493,066,558	7,065 xd	493,073,623	18	13	1	LZ77	
187.	sharc 0.9.11b	-c2	53,175,042	494,421,068	81,001 s	494,502,069	15	14	6	LZP	28
188.	flzp v1		57,366,279	497,535,428	3,942 s	497,539,370	78	38	8	LZP	
189.	alba 0.5.1	cd	52,728,620	515,760,096	4,870 s	515,764,966	239	10	4	BPE	48
190.	lzpgt6		56,113,248	522,877,083	27,136 x	522,904,219	6	5	6	LZP	95
191.	snappy 1.0.1		58,350,605	527,772,054	23,844 s	527,795,898	25	12	0.1	LZ77	26
192.	bpe	5000 4096 200 3	53,986,667	532,250,688	1,037 sd	532,251,725	639	28	0.5	Dict	26
193.	kwc		54,097,740	532,622,518	15,186 x	532,637,704	438	145	668	Dict	26
194.	bpe2 v3		55,289,197	542,748,980	2,979 s	542,751,959	518	132	0.5	Dict	26
195.	fpaq0f2		56,916,872	558,645,708	3,066 x	558,648,769	222	207	0.4	o0	
196.	ghost	456 5	55,357,186	568,004,779	696 sd	568,005,475	172800	245	88000	Dict	100
197.	srr		61,657,971	579,352,307	1,472 s	579,353,779	80	59	1	SR	
198.	ksr	4	59,511,259	580,557,413	13,507 x	580,570,920	40050	7917	1700	SR	48
199.	lzbw1 0.8		67,620,436	590,235,688	21,751 x	590,257,439	15	12	55	LZP	26
200.	lzp2 0.7c		67,909,076	598,076,882	40,819 x	598,117,701	11	8	15	LZP	26
201.	NTFS	LZNT1	70,955,648	636,870,656	0	636,870,656	10	9	0.1	LZ77	28
202.	shindlet_fs		62,890,267	637,390,277	1,275 xd	637,391,552	113	103	0.6	o0	
203.	arb255		63,501,996	644,561,595	4,871 sd	644,566,466	2551	2574	1.6	o0	
204.	compact		63,862,371	648,370,029	3,600 sd	648,373,629	216	164	0.2	o0	
205.	TinyLZP 0.1		79,220,546	694,274,932	2,811 s	694,277,743	32	38	10	LZP	26
206.	smile		71,154,788	695,562,502	207 xd	695,562,709	10517	10414	0.6	MTF	26
207.	barf	(2 passes)	76,074,327	758,482,743	983,782 s	759,466,525	756	53	4	LZ77	
208.	arb2x v20060602		99,642,909	995,674,993	3,433 sd	995,678,426	2616	2464	1.6	o0b	

https://www.mattmahoney.net/dc/text.html





Jacob Ziv

Abraham Lempel

- LZ77 (sliding window)
- LZ78 (incremental parsing)

LZ compression achieves:

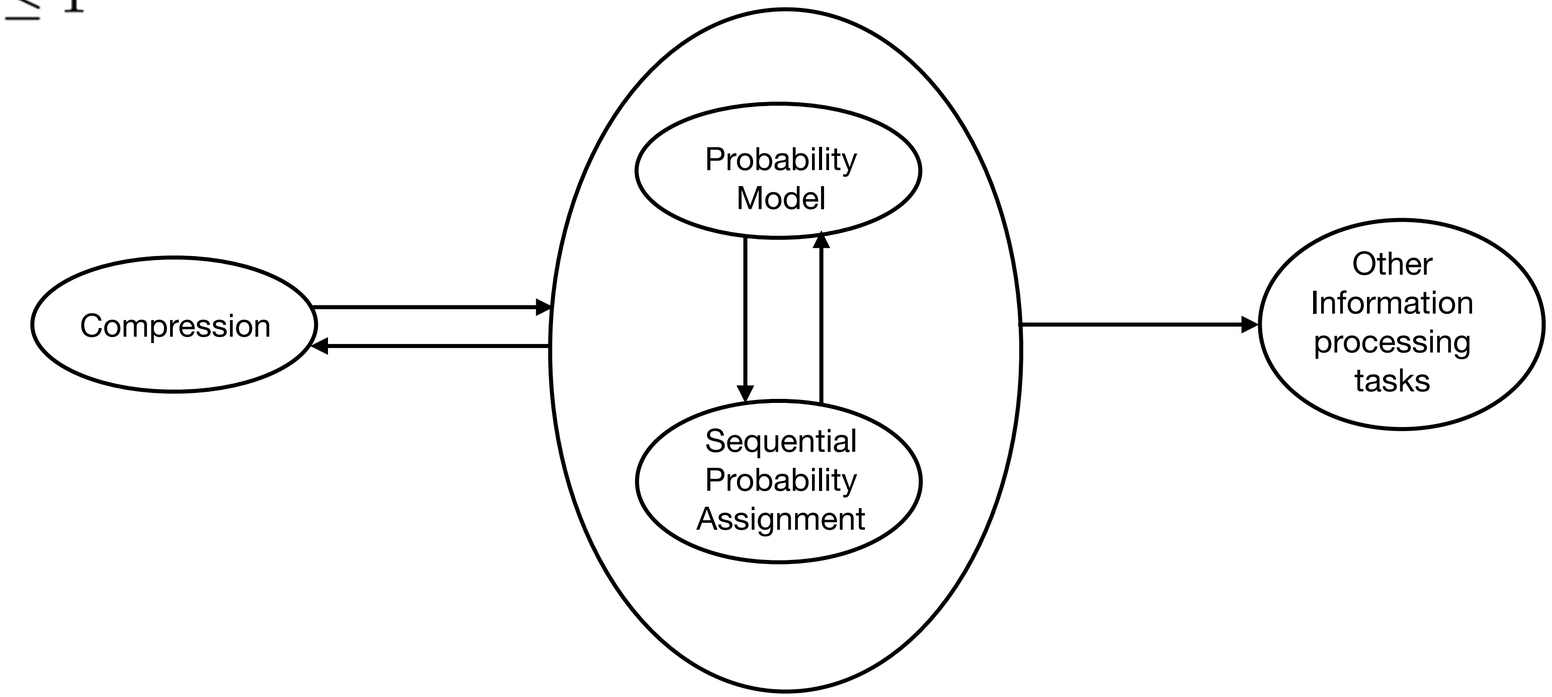
- the fundamental limits (entropy rate and finite-state compressibility)
- (near) linear complexity
- elegance
- state of the art performance on real data

why care about (lossless) compression?

in(formation) theory (101)

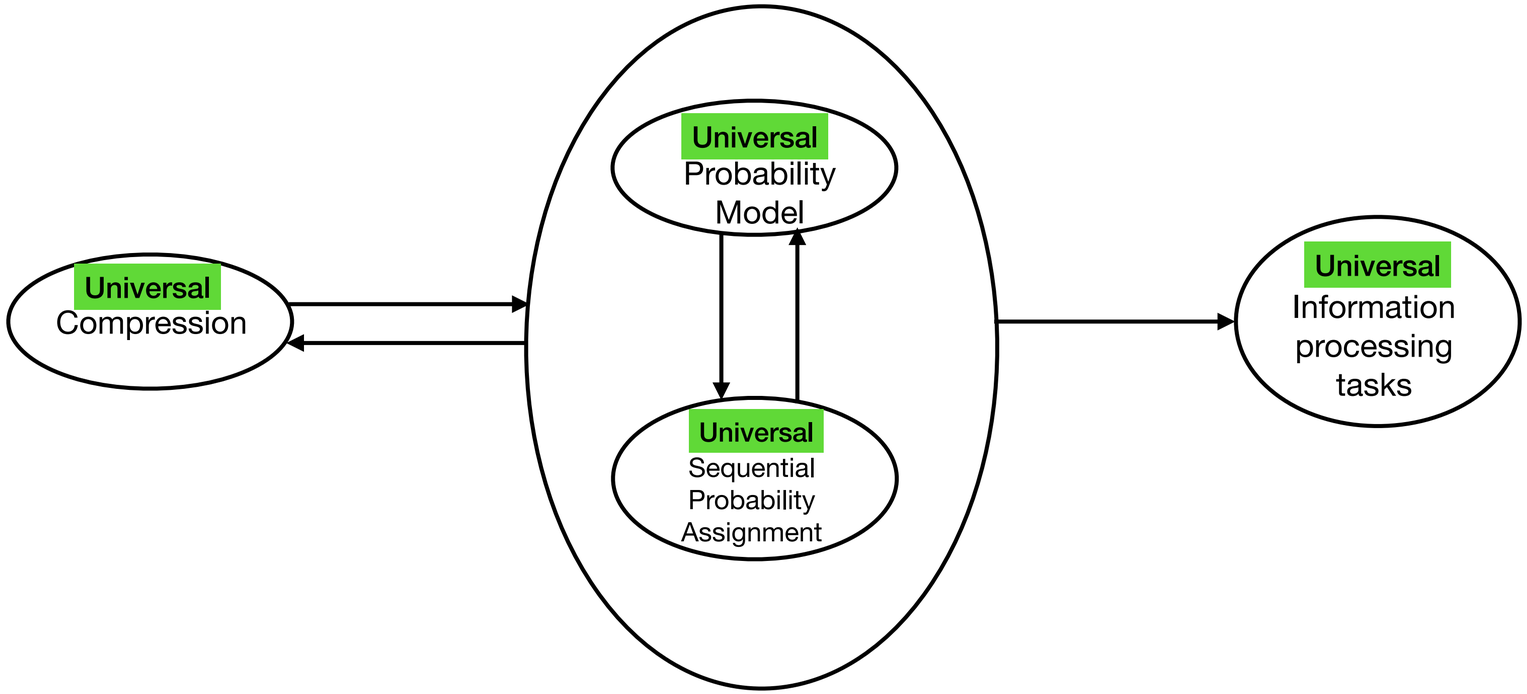
Kraft inequality:

$$\sum_{\text{data}} 2^{-\ell(\text{data})} \leq 1$$

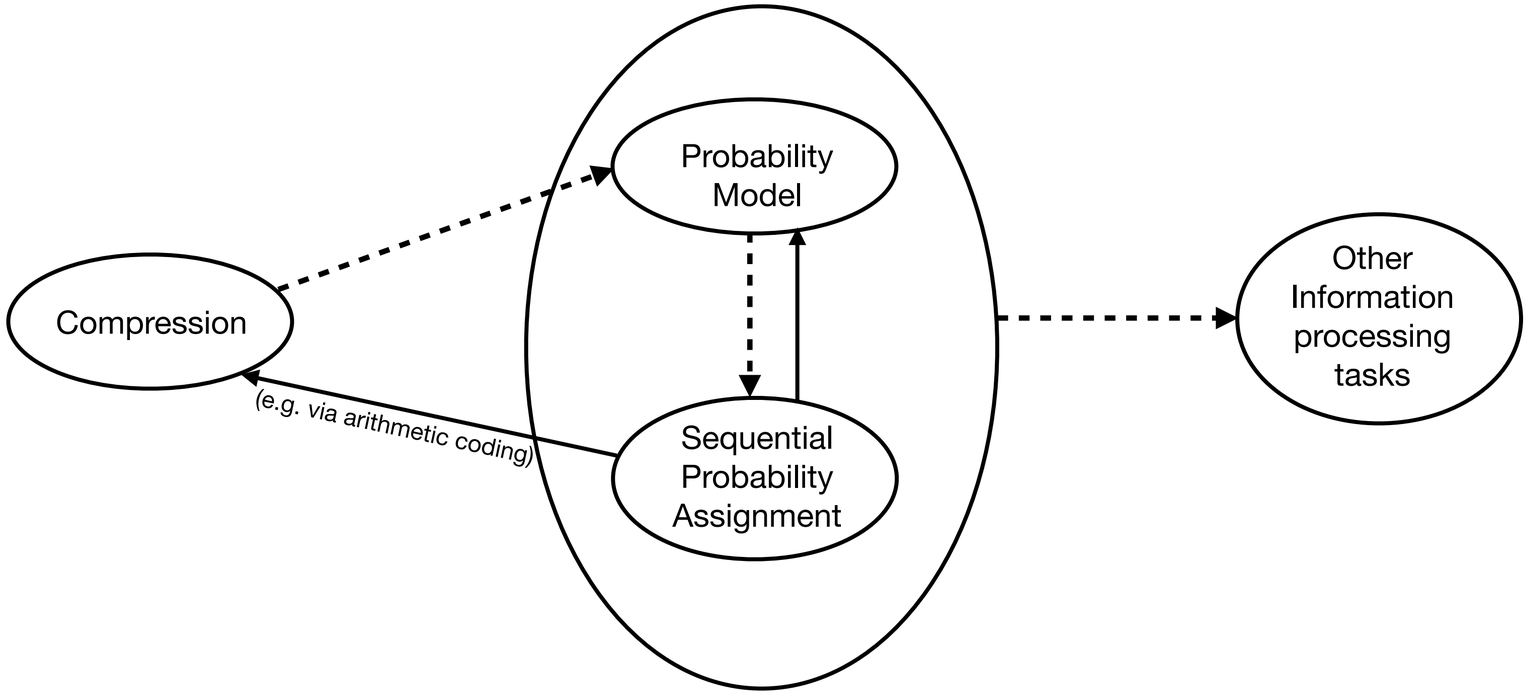


- (gets rediscovered periodically in other communities, cf., e.g., “Language modeling is compression”, ICLR 2024)

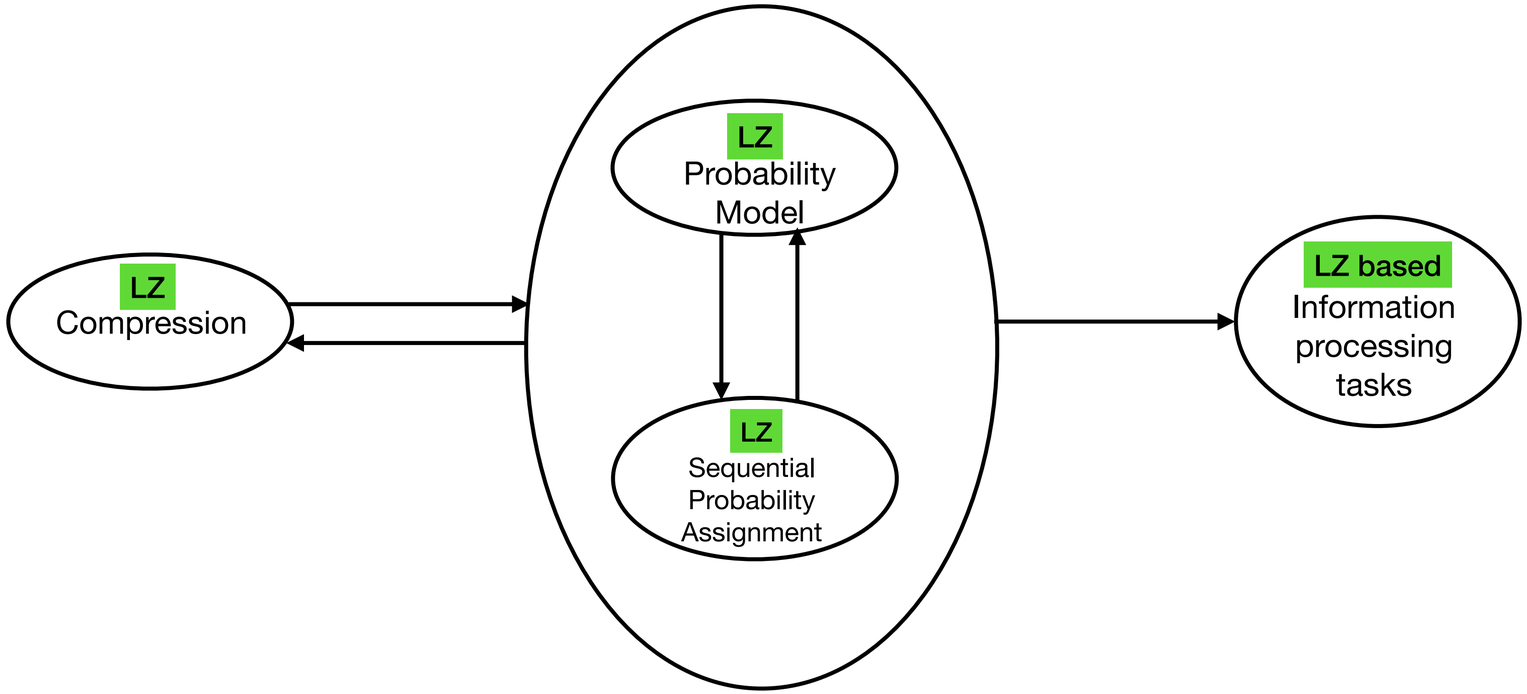
universality is also inherited



practically/algorithmically



LZ(78): both universality and practicality is inherited

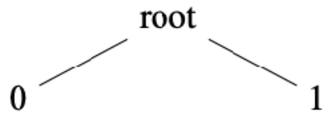


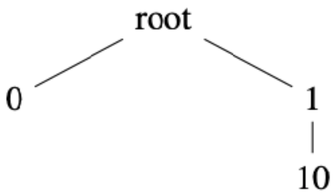
LZ78 induced Sequential Probability Assignment (SPA)

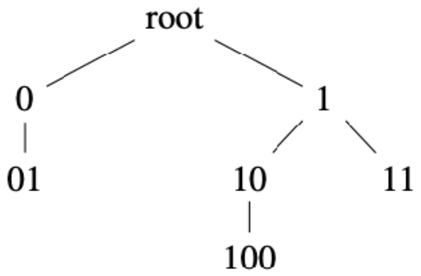
$$x^n = 01100110011$$

incremental parsing: 0,1,10,01,100,11

induces tree:







tree induces SPA

# of phrases in the parsing

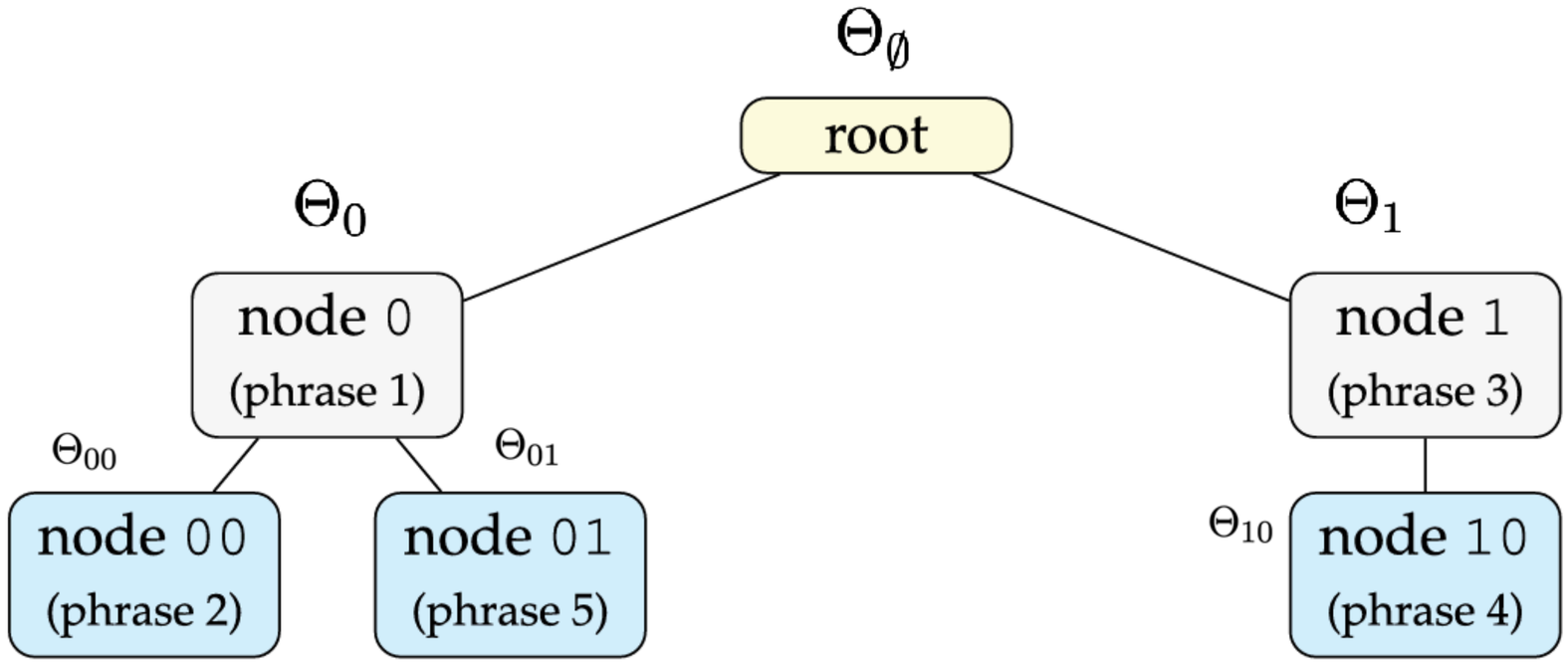
$$\ell_{LZ}(x^n) = \frac{C(x^n)\log C(x^n)}{n} + o(1)$$

N. Sagan and T. Weissman,
"A Family of LZ78-based Universal Sequential Probability Assignments",
Information Theory transactions, April 2026
[SW26]

[SW26] construction: Bayesian mixtures in LZ78 contexts

00011001 -> 0,00,1,10,01,…

$$\Theta_\cdot \sim f$$



[SW26]

example

when f is the Dirichlet prior:

$$q^f(x_t|x^{t-1}) = \frac{N(x_t|x^{t-1}) + \gamma}{(t-1) + \gamma A}$$

$$q^{LZ78,f}(a|x^{t-1}) = \frac{N(a|x^{t-1}, z_c(x^{t-1})) + \gamma}{\sum_{b \in \mathcal{A}} N(b|x^{t-1}, z_c(x^{t-1})) + \gamma A}$$

$z_c(x^{t-1})$ is the node in the LZ tree reached at time $t-1$

[SW26] optimum per-symbol log loss achievable by a finite-state SPA

(aka “finite-state predictability under log loss”)

Definition IV.1 (Finite-State SPA)**.** A finite-state SPA is such that the probabilities assigned depend solely on the current state of an underlying finite-state mechanism. Concretely, q is a M-state SPA if $\exists$ next-state function $g : [M] \times \mathcal{A} \to [M]$, prediction function $f : [M] \to \mathcal{M}(\mathcal{A})$, and initial state $s_1 \in [M]$ such that

$$\forall t \geq 1,\ q(\cdot|x^{t-1}) = f(s_t), \quad \text{and} \quad s_{t+1} = g(s_t, x_t).$$

The set of all M-state SPAs is denoted $\mathcal{F}_M$.

The optimal log loss of any finite-state SPA is defined as follows:

Definition IV.2 (Optimal Finite-State Log Loss)**.** The minimum log loss of a M-state SPA for sequence $\mathbf{x}$ is defined as

$$\lambda_M(\mathbf{x}) \triangleq \overline{\lim_{n\to\infty}}\, \lambda_M(x^n) \quad \text{for} \quad \lambda_M(x^n) \triangleq \min_{q\in\mathcal{F}_M} \frac{1}{n}\sum_{t=1}^{n} \log \frac{1}{q(x_t|x^{t-1})}.$$

The optimal finite-state log loss takes the number of states to infinity,

$$\lambda(\mathbf{x}) \triangleq \lim_{M\to\infty} \overline{\lim_{n\to\infty}}\, \lambda_M(x^n).$$

$\lambda_M(\mathbf{x})$ is monotonically non-increasing in M and bounded below, so the outer limit is guaranteed to exist.

[SW26]

optimum per-symbol log loss achievable by a Markov SPA

(aka “Markov predictability under log loss”)

The definition of the optimal Markov SPA log loss is analogous to that of the optimal finite-state SPA log loss:

Definition IV.6 (Optimal Markov Log Loss)**.** For any sequence $\mathbf{x}$, the optimal log loss of a k-order Markov SPA is

$$\mu_k(\mathbf{x}) \triangleq \overline{\lim_{n\to\infty}}\, \mu_k(x^n) \quad \text{for} \quad \mu_k(x^n) \triangleq \min_{q\in\mathcal{M}_k} \frac{1}{n}\sum_{t=1}^{n} \log \frac{1}{q(x_t|x^{t-1})}.$$

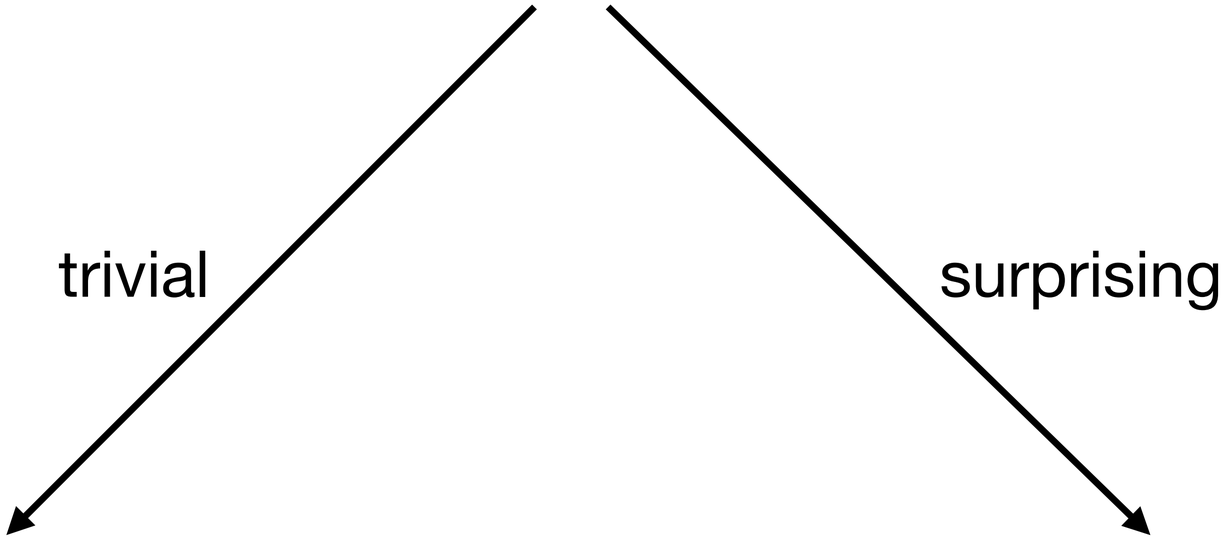
The optimal Markov SPA log loss is defined by taking the context length to ∞:

$$\mu(\mathbf{x}) \triangleq \lim_{k\to\infty} \mu_k(\mathbf{x}) = \lim_{k\to\infty} \overline{\lim_{n\to\infty}}\, \mu_k(x^n).$$

As with $\lambda(\mathbf{x})$, the outer limit is guaranteed to exist.

[SW26]

Theorem: $\lambda(\mathbf{x}) = \mu(\mathbf{x})$



$\lambda(\mathbf{x}) \leq \mu(\mathbf{x})$

$\lambda(\mathbf{x}) \geq \mu(\mathbf{x})$

[SW26]

universality of the LZ78 family of SPAs

Theorem V.1 (Universality of LZ78 SPA). *For prior distribution* Π *such that* $\mathrm{supp}(\Pi) = \mathcal{M}(\mathcal{A})$, $q^{LZ78,\Pi}$ *from Construction III.5 satisfies, for any individual sequence,*

$$\overline{\lim_{n\to\infty}} \frac{1}{n} \log \frac{1}{q^{LZ78,\Pi}(x^n)} \leq \lambda(\mathbf{x}).$$

universality in the stochastic setting (classical)

entropy rate: $\mathbb{H}(\mathbf{X}) = \lim_{n\to} \frac{1}{n} H(X^n)$

Theorem E.2. *Suppose* $\mathbf{X}$ *is a stationary process and the SPA* q *satisfies*

$$\overline{\lim_{n\to\infty}} \frac{1}{n} \log \frac{1}{q(x^n)} \leq \mu(\mathbf{x}),$$

for all individual sequences $\mathbf{x}$*. Then, if* $\mathbb{H}(\mathbf{X})$ *is the entropy rate of the stochastic process,*

$$\lim_{n\to\infty} \mathbb{E}\left[\frac{1}{n} \log \frac{1}{q(X^n)}\right] = \mathbb{H}(\mathbf{X}).$$

(equivalently: $\lim_{n\to\infty} \frac{1}{n} D(\text{True law of } X^n \| \text{law of } X^n \text{ under } q) = 0$)

When $\mathbf{X}$ is also ergodic: $\mu(\mathbf{X}) = \lambda(\mathbf{X}) = \mathbb{H}(\mathbf{X}) \quad a.s.$

result at the heart

Theorem III.11. *For any prior such that* $\mathrm{supp}(\Pi) = \mathcal{M}(\mathcal{A})$,

$$\lim_{n\to\infty} \max_{x^n} \left| \frac{1}{n} \log \frac{1}{q^{LZ78,\Pi}(x^n)} - \frac{C(x^n)\log C(x^n)}{n} \right| = 0.$$

where $C(x^n)$ is the number of phrases in the LZ parsing of x^n and

$$\ell_{LZ}(x^n) = \frac{C(x^n)\log C(x^n)}{n} + o(1)$$

initial experiments in [SW26]

Table I
RESULTS OF CLASSIFICATION EXPERIMENTS USING THE LZ78 SPA.

Dataset	MNIST	Fashion-MNIST	IMBD	Spam Emails
Accuracy (%)	75.36	72.16	75.62	98.12
Training Time (s)	14	15	16	14

initial experiments in [SW26] (cont.)

This is the moon with the fairest charge thee stay;
Which is not so.

Provost:
Art thou not
That which withal; you go you to Baptista; or, but I am not Lucentio,
Red the beasts, that Warwick,
And those that runatice of his auture of our straight and will play the his profane eyes, came to see thy servant so die.

LUCIO:
But what lives in Signior Gremio: fools
His will I may have auture of his auture of our common good time
Unto the Tower,
Give me thy hand as come enough
And then he shall not rather with the Lord Stand be so longer see that are fond, as thou say the orace to speak brother, or oints?

BIONDELLO:
I may more straight and will play the his punish his convey much this leavine art thou that will not should be thus sir, there's face.
Go you to Baptista; or, lo, here all abroad in the

Figure 1. 800 symbols generated via the LZ78 SPA, trained on the `tiny-shakespeare` dataset, seeded with the sequence "This."

Y. Omri, N. Sagan, E. Min, H. Choi, T. Moon and T. Weissman,
"Genomic Data Classification via Universal Compression”,
[Omri et al. 25]
ISMB 2025 and
submitted for journal publication
https://doi.org/10.21203/rs.3.rs-6363017/v1

[Omri et al. 25]

main idea

- compress the labeled data from each class jointly with the test sequence
- choose class with which the test sequence jointly compresses best
- when using LZ78:
  - training is linear in the training data
  - inference is linear in the test data
  - induced LZ78 tree gives a succinct and transparent summary of the class statistics

[Omri et al. 25]

on accuracy:

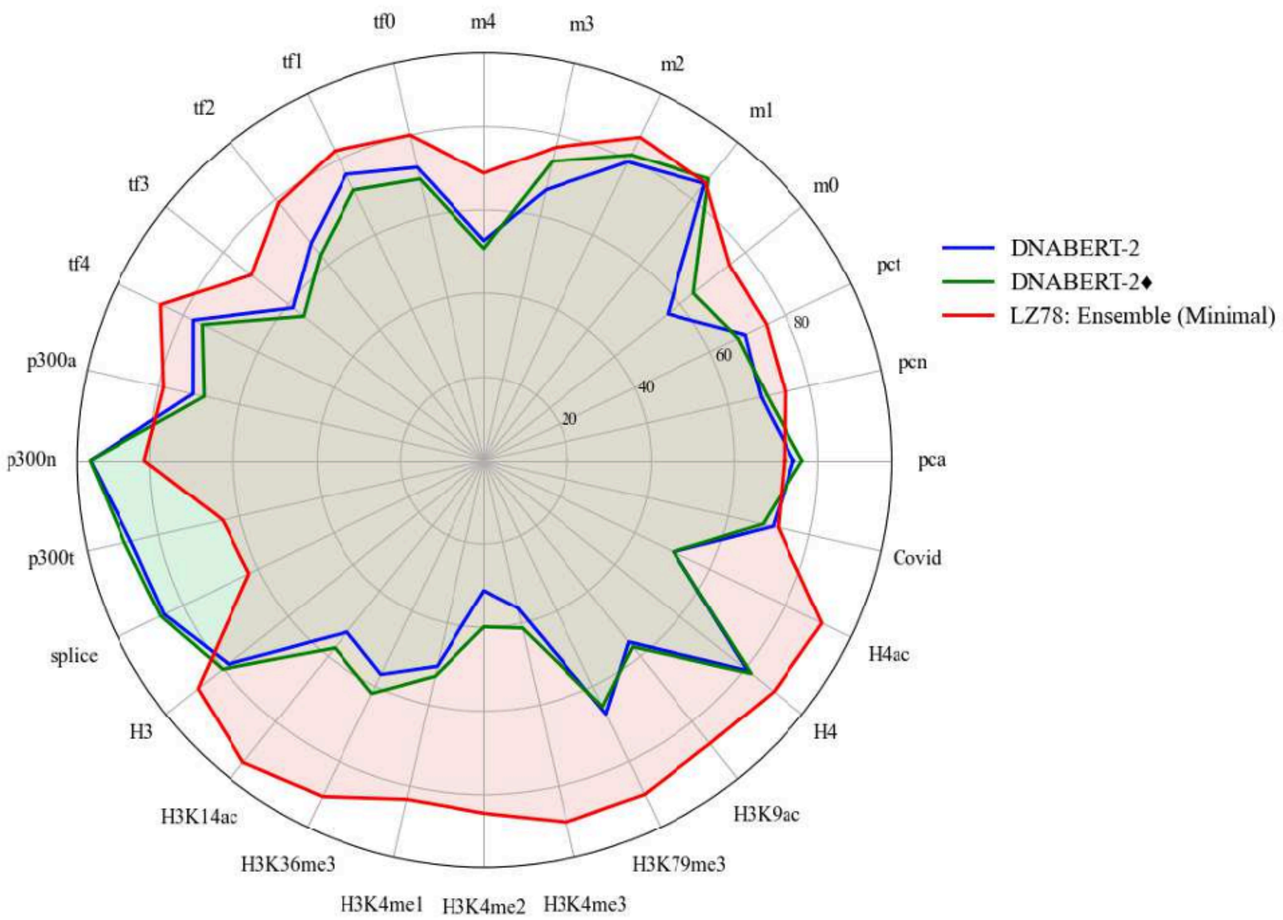
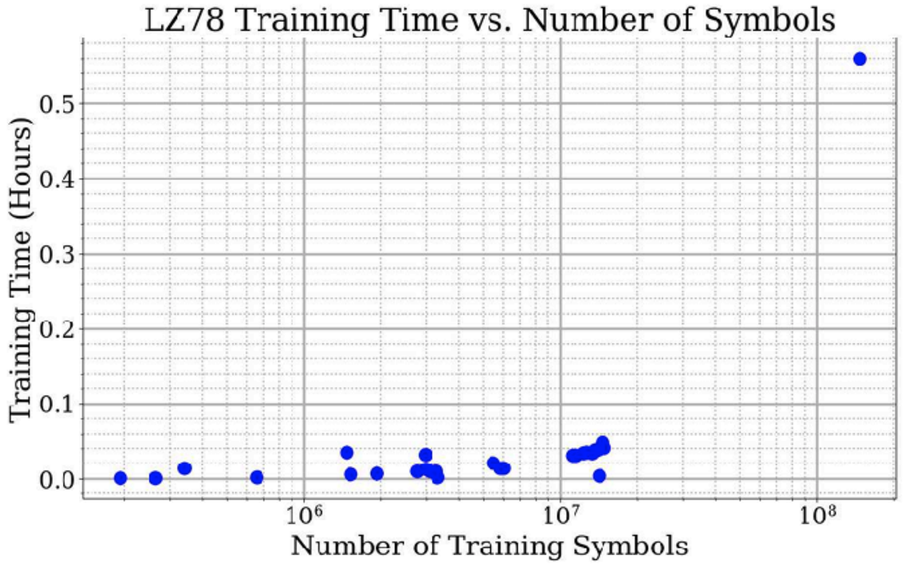


Fig. 2: Accuracy results on the GUE benchmark dataset

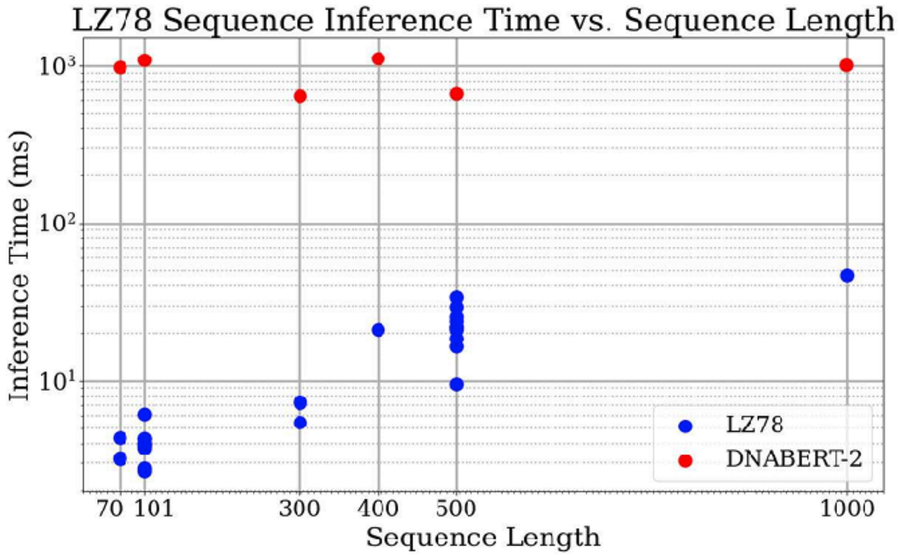
(GUE: Genomic Understanding Evaluation Suite)

[Omri et al. 25]

on compute:



(a) Efficiency of LZ78 Classifier Training; average training time of ≈2.3 minutes per dataset on a modern CPU.



(b) Inference time speedup of Using an LZ78-Based classifier compared to DNABERT-2.

[Omri et al. 25]

merits

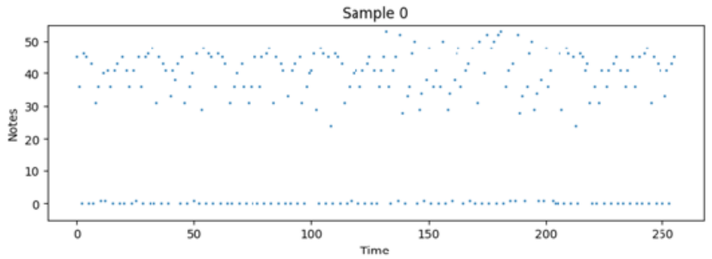
- improved accuracy
- dramatically faster
- requires substantially less training data
- transparent white-box logic
- theoretically justified

A. Gorle, C. Ding, S. Bhattacharya, D. Hasteer, N. Sagan and T. Weissman,
"LZMidi: Compression-Based Symbolic Music Generation",
ICASSP 2026
[Gorle 26]

[Gorle 26]

LZMidi

- **Goal**: Offer an alternative to large ML-based generators with a *universal*, *CPU-only* compression-driven generator (sampler)
- **Core idea**: Fast sampling from the sequential probability assignment induced from an LZ78 tree (compression)
- **Application**: Symbolic Music generation
- **Theoretical results**: Convergence to true source statistics

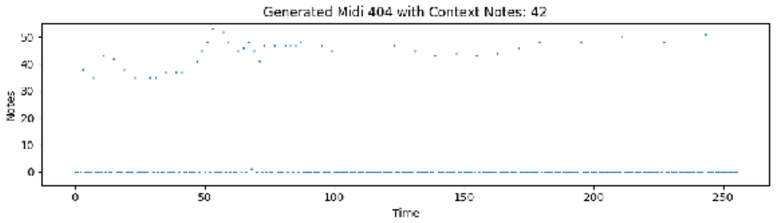


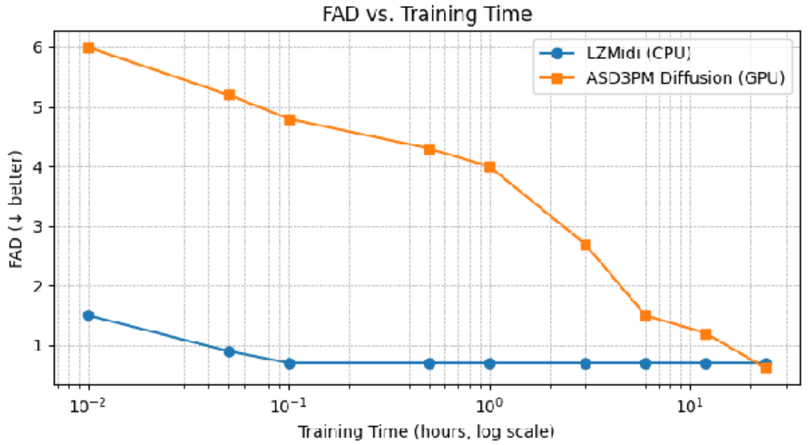
Training and Dataset:

- Lakh MIDI Dataset (LMD): 648,574 symbolic MIDI sequences (each 256 tokens over alphabet {0-89}, where 0=rest, 1=continuation, 2-89=pitches)
- 80/20 split: ~518 k sequences for training (used to build/update the LZ78 tree)
- Training: a simple one-pass through all training sequences to build the LZ tree, CPU-friendly

LZMidi highlights

- **Sampling/generation**: initialized by a single randomly selected symbol (a direct child of the LZ tree root) from the 90-symbol alphabet
- Reference sample on the right
- **Metrics (vs. deep models):**
  - FAD as low as ↓ 0.64 (*comparable to MusicVAE and fully-trained Diffusion baselines*)
  - Training: 30× faster, Generation: 300× faster (CPU only) versus neural baselines (GPU)
  - substantially outperforms compute-matched deep-learning baselines
- **Compute Efficiency:** Hits FAD ~ 0.7 within minutes on M1 CPU vs hours of GPU training for deep-learning models





	Train (s)	Generation (s/sample)	Model Size, MB	TE (kJ)	GE (J/sample)
LZMidi	**107.7**	**0.016**	**287.1**	**9.144**	**1.36**
ASD3PM	3480	5.4	306.2	2088	3240

Key Takeaway: Quality comparable to SOTA, but *minutes on CPU* instead of *GPU-days*

examples:

got us thinking

- about compressors and SPAs as information sources
- for some it’s very clear: Markov, CTW,…
- what about the LZ78 SPA as an information source?

N. Sagan, A. Dembo, M. Ho and T. Weissman,
"The LZ78 Source", Information theory transactions, 2026 [SDHW26]

question:
what are the distributional and entropic properties of a process generated from the LZ78 source?

[SDHW26]:

Theorem III.11 (Shannon-McMillan-Breiman-Type Result)**.** *Let* $\mathbf{X} \sim Q^{LZ,\Pi}$ *be generated from the LZ78 source with* $\operatorname{supp}(\Pi) = \mathcal{M}(\mathcal{A})$*. Then, almost surely,*

$$\lim_{n\to\infty} \frac{1}{n} \log \frac{1}{Q^{LZ,\Pi}_{X^n}(X^n)} = \mathbb{E}_{\Theta\sim\Pi}[H(\Theta)].$$

Theorem III.12 (Entropy Rate)**.** *The entropy rate of the LZ78 source, under any* Π*, is*

$$\lim_{n\to\infty} \frac{1}{n} H(X^n) = \mathbb{E}_{\Theta\sim\Pi}[H(\Theta)].$$

on the other hand:

Theorem III.13 (Finite-state Predictability)**.** *For* $\mathbf{X}$ *generated from the LZ78 source, the finite-context Markov predictability is almost surely*

$$\mu(\mathbf{X}) = H(\mathbb{E}_{\Theta\sim\Pi}[\Theta]).$$

so there's a "Jensen gap" (in stark contrast to stationary ergodic processes)

[SDHW26]:

divergence properties

When Π has full support, among the results of [6] is the universality of $Q^{\mathrm{LZ},\Pi}$ in the sense that

$$\lim_{n\to\infty} \frac{1}{n} D\left(P_{X^n} \| Q^{\mathrm{LZ},\Pi}_{X^n}\right) = 0 \quad \forall P \text{ stationary}.$$

In contrast:

Theorem III.6 (Divergence with Markovian Law: Pointwise Result)**.** *Let* $\mathbf{X}$ *be generated from the LZ78 source* $Q^{LZ,\Pi}$ *with* $\mathrm{supp}(\Pi) = \mathcal{M}(\mathcal{A})$*. For any* k^{th}*-order Markov law* $P \in \mathcal{M}_k$*, almost surely*

$$\lim_{n\to\infty} \frac{1}{n} \log \frac{Q^{LZ,\Pi}_{X^n}(X^n)}{P_{X^n}(X^n)} = \mathbb{E}\left[D\left(\mathbb{E}[\Theta] \,\middle\|\, P_{X_0|X_{-k}^{-1}}(\cdot|Y_{-k}^{-1})\right)\right] + H(\mathbb{E}[\Theta]) - \mathbb{E}[H(\Theta)], \tag{1}$$

where on the right side $\Theta \sim \Pi$ *and* $Y_{-k}^{-1} \overset{i.i.d.}{\sim} \mathbb{E}[\Theta]$.

Corollary III.7 (Relative Entropy with Markovian Law)**.** *For* $Q^{LZ,\Pi}$ *with* $\mathrm{supp}(\Pi) = \mathcal{M}(\mathcal{A})$*, and any* k^{th}*-order Markov law* $P \in \mathcal{M}_k$*,*

$$\lim_{n\to\infty} \frac{1}{n} D\left(Q^{LZ,\Pi}_{X^n} \| P_{X^n}\right) = \mathbb{E}\left[D\left(\mathbb{E}[\Theta] \,\middle\|\, P_{X_0|X_{-k}^{-1}}(\cdot|Y_{-k}^{-1})\right)\right] + H(\mathbb{E}[\Theta]) - \mathbb{E}[H(\Theta)], \tag{2}$$

where on the right side $\Theta \sim \Pi$ *and* $Y_{-k}^{-1} \overset{i.i.d.}{\sim} \mathbb{E}[\Theta]$.

Remark III.8. In the spirit of Remark III.5, the expression in the right sides of (1) and (2) can be equivalently written as

$$D\left(P_Y \,\middle\|\, P_{X_0|X_{-k}^{-1}} \middle| P_{Y_{-k}^{-1}}\right) + I(\Theta; Y),$$

where: $\Theta \sim \Pi$, $Y \sim \Theta$, $Y_{-k}^{-1} \overset{\text{i.i.d.}}{\sim} Y$, $P_{X_0|X_{-k}^{-1}}$ is that associated with the $P \in \mathcal{M}_k$, and here we assume the notation of [11] for relative entropy between conditional distributions (kernels) averaged with respect to another distribution.

[SDHW26]: distributional properties

Corollary IV.11. *Applying Theorem IV.10 with single-sequence event A such that $A_1 = \cdots = A_r = \mathcal{M}(\mathcal{A})$, we obtain*

$$\frac{1}{n}\sum_{i=1}^{n} \delta_{X_i^{i+r-1}}(a^r) \xrightarrow{a.s.} \prod_{t\in[r]} \mathbb{E}[\Theta[a_t]] \quad \text{as } n \to \infty, \quad \forall a^r \in \mathcal{A}^r.$$

In words, the rth-order empirical distribution of the realized sequence from the LZ78 source converges, almost surely, to the law of r i.i.d. drawings from the PMF $\mathbb{E}[\Theta]$.

intuition

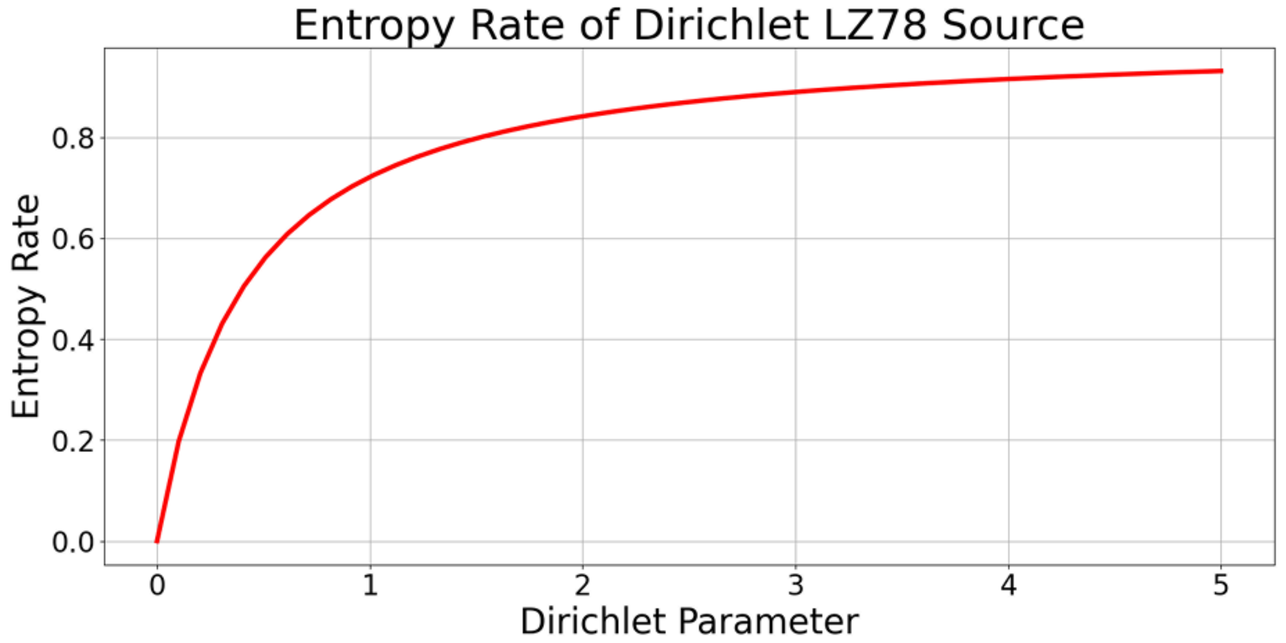
- the true model keeps track of the tree and the location of the process in it
- most of the time the process is in nodes visited a growing number of times so true model quickly learns their realized theta
- then “merely” need show the empirical distribution of the thetas is what one would expect
- any finite memory machine loses sync of where the process is in the tree and since things are locally iid its log loss is as good as that for predicting an iid source

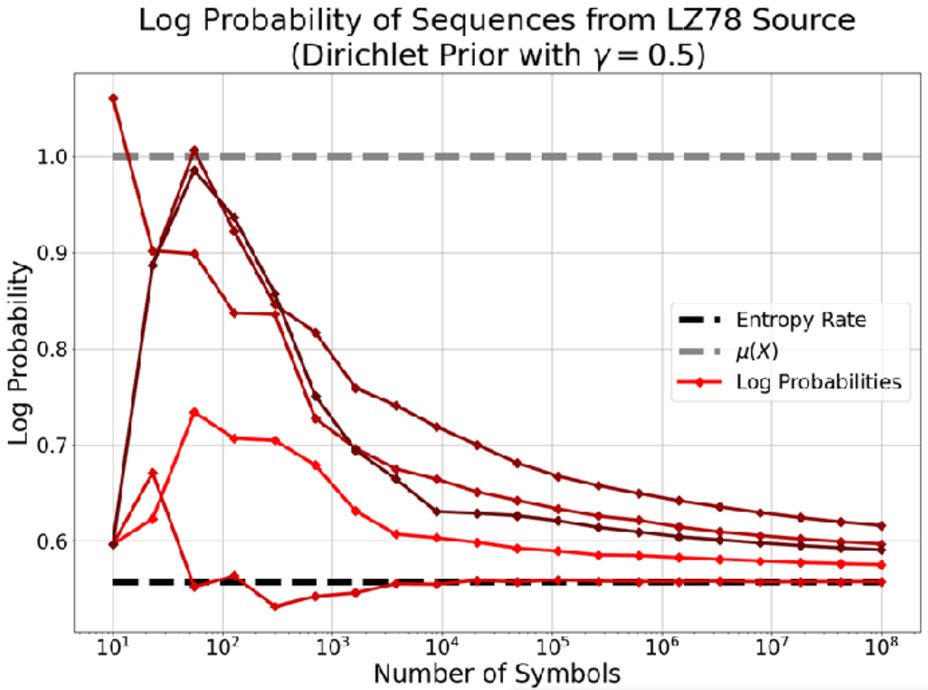
[SDHW26]:

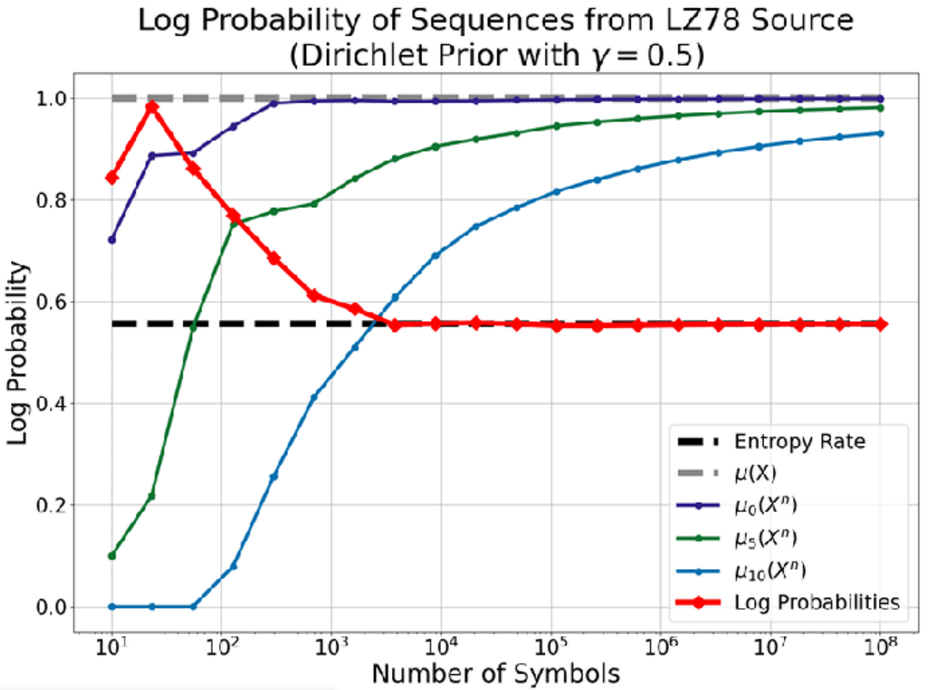
question:
what are the distributional and entropic properties of a process generated from the LZ78 source?

answer: the “LZ source”

- is non-stationary
- but ergodic
- is asymptotically “locally iid”
- is globally far from it
- has entropy rate < finite-state compressibility
- has the Shannon-McMillan-Breiman property
- divergent relative entropies


Entropy Rate of Dirichlet LZ78 Source
Entropy Rate
Dirichlet Parameter
0.0
0.2
0.4
0.6
0.8
0
1
2
3
4
5


Log Probability of Sequences from LZ78 Source
(Dirichlet Prior with $\gamma = 0.5$)
Log Probability
Number of Symbols
Entropy Rate
$\mu(X)$
Log Probabilities


Log Probability of Sequences from LZ78 Source
(Dirichlet Prior with $\gamma = 0.5$)
Log Probability
Number of Symbols
Entropy Rate
$\mu(X)$
$\mu_0(X^n)$
$\mu_5(X^n)$
$\mu_{10}(X^n)$
Log Probabilities

why is this interesting?

- provides a highly non-trivial ergodic information source with very different local vs global behavior
- can be useful for testing deep nets
- might be a ‘universal data source for training’
- can be harnessed for lossy compression

cf. upcoming international symposium on information theory (ISIT 2026):

The LZ78 Source and its Application to Evaluating In-Context Learning

Naomi Sagan
Electrical Engineering
Stanford University
Stanford, CA, USA
nsagan@stanford.edu

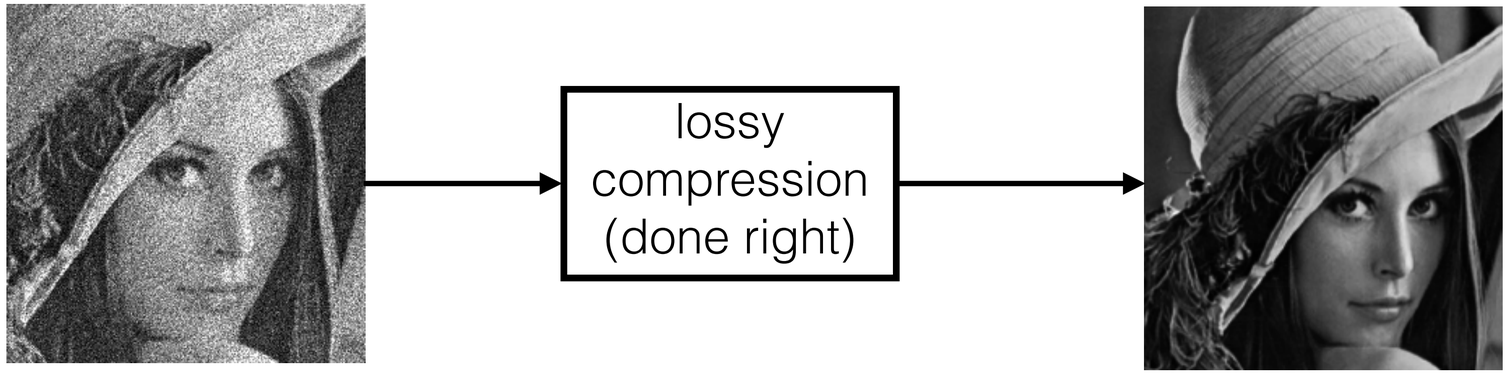
Amir Dembo
Statistics, Mathematics
Stanford University
Stanford, CA, USA
adembo@stanford.edu

Matthew Ho
Electrical Engineering
Stanford University
Stanford, CA, USA
matthho@stanford.edu

Tsachy Weissman
Electrical Engineering
Stanford University
Stanford, CA, USA
tsachy@stanford.edu

speaking of lossy compression

denoising via lossy compression



- “Occam filters”
- “Compresstimation”
- “Kolmogorov sampler”
- etc.

renewed recent interest due to prevalence of good “universal” lossy compressors

lossy compression of noisy signals for denoising (cont.)

Effect of lossy compression of quality scores on variant calling FREE

Idoia Ochoa ✉, Mikel Hernaez, Rachel Goldfeder, Tsachy Weissman, Euan Ashley

Briefings in Bioinformatics, Volume 18, Issue 2, March 2017, Pages 183–194,

Compression Detects Changes in Spiking Neural Data from Cortical Lesions

Alice Tor[1], Yuxin Wu[1], Stephen E Clarke[2,3], Lisa Yamada[2,1,3], Tsachy Weissman[1], Paul Nuyujukian[2,3,1,4], for the Brain Interfacing Laboratory

Lossy Compression of Noisy Data for Private and Data-Efficient Learning

Berivan Isik and Tsachy Weissman, *Fellow, IEEE*

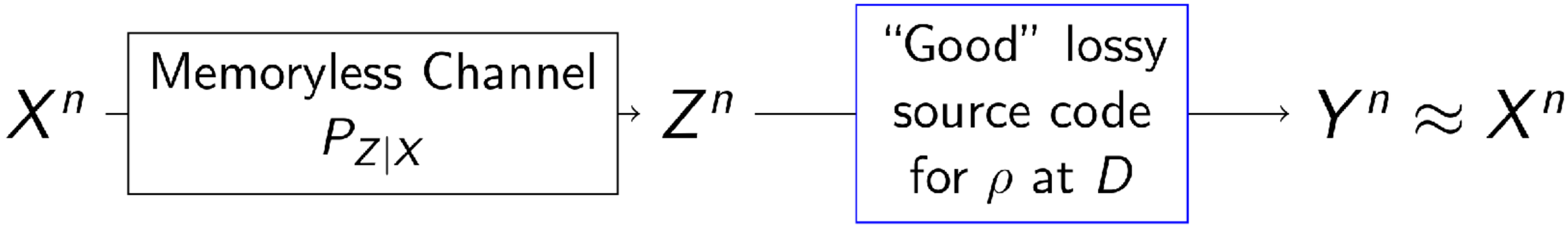
less storage, faster processing/learning, boosted accuracy in the downstream, enhanced privacy when it matters, …

[SOW26]: “The Performance of Compression-Based Denoisers”,

Dan Song, Ayfer Özgür and T. Weissman,

to appear in Information Theory Transactions

https://arxiv.org/abs/2512.14539



- Source X^n from fixed stationary ergodic source X.
- Channel $P_{Z|X}$ are fixed and known.

"Good" Codes

Definition

A sequence of codes $(\mathcal{C}_n, \phi_n)$ is *good* at distortion D if asymptotically achieves a point on the rate distortion curve:

$$\limsup_{n\to\infty} \mathbb{E}[\rho_n\left(Z^n, \phi_n\left(Z^n\right)\right)] \leq D.$$

$$\limsup_{n\to\infty} \frac{1}{n} \log\left|\mathcal{C}_n\right| \leq \lim_{k\to\infty} R\left(Z^k, D\right)$$

We will take $Y^n = \phi_n\left(Z^n\right)$.

[SOW26]:

Empirical Distribution

Let:

$$J \sim \mathrm{Unif}([n-2k-1]), J \perp\!\!\!\perp (X^n, Z^n, Y^n).$$

Define:

$$X^n = (X_1, \ldots, X_{J-1}, X_J, \ldots, X_{J+2k}, X_{J+2k+1}, \ldots, X_n)$$

$$Z^n = (Z_1, \ldots, Z_{J-1}, Z_J, \ldots, Z_{J+2k}, Z_{J+2k+1}, \ldots, Z_n)$$

$$Y^n = (Y_1, \ldots, Y_{J-1}, \underbrace{Y_J, \ldots, Y_{J+2k}}_{(\tilde{X}^{(n,k)}, \tilde{Z}^{(n,k)}, \tilde{Y}^{(n,k)})}, Y_{J+2k+1}, \ldots, Y_n).$$

Let $Q^{(n)}$ be the probability measure such that

$$(\tilde{X}^{(n,k)}, \tilde{Z}^{(n,k)}, \tilde{Y}^{(n,k)}) \sim Q^{(n)}_{X^k_{-k}, Z^k_{-k}, Y^k_{-k}}.$$

[SOW26]:

Theorem 2.2 *Suppose Y^n is the output of a good compressor for Z^n for distortion*

$$\rho(z, y) = -\log P_{Z|X}(z|y)$$

at distortion level

$$D = H(Z|X).$$

Then, for every k,

$$Q^{(n)}_{Z^k,Y^k} \overset{n\to\infty}{\Longrightarrow} P_{Z^k,X^k}$$

in words:

’good’ lossy compression of the noisy signal gives a ‘sample from the posterior’

[SOW26]:

Characterization of Loss

Theorem (Informal)

For any choice of bounded loss, the loss between X^n and Y^n achieved by a sequence of good codes for

$$X^n - Z^n - Y^n$$

approaches the loss between X_0 and X_0', where

$$P_{X_0'|\mathbf{Z}} = P_{X_0|\mathbf{Z}},$$

$$X_0 - \mathbf{Z} - X_0'.$$

in words: an *independent* sample from the posterior

[SOW26]:

Exact Characterization of Loss

Theorem

Suppose $(\mathbf{X}, \mathbf{Z})$ *are double-sided mixing,* $P_{Z|X}$ *has invertible channel matrix, and* $\mathcal{X}, \mathcal{Z}, \mathcal{Y}$ *are finite.*
Choosing distortion $\rho(z, y) = -\log p_{Z|X}(z \mid y)$ *and distortion level* $H(Z \mid X)$*, for any bounded loss* $\Lambda : \mathcal{X} \times \mathcal{Y} \to [0, \Lambda_{\max}]$*,*

$$\lim_{n\to\infty} \mathbb{E}[\Lambda_n(X^n, Y^n(Z^n))] = \mathop{\mathbb{E}}_{Z_{-\infty}^{\infty}} \left[\mathop{\mathbb{E}}_{(X',Y')|Z_{-\infty}^{\infty}} \left[\Lambda(X', Y') \right] \right], \quad (14)$$

where

$$(X', Y') \sim \left(P_{X_0|Z_{-\infty}^{\infty}} \right)^2. \quad (15)$$

- Loss is exactly characterized (the limit exists).
- Independent samples from the posterior $P_{X_0|Z_{-\infty}^{\infty}}$.

(note scheme is oblivious to loss function)

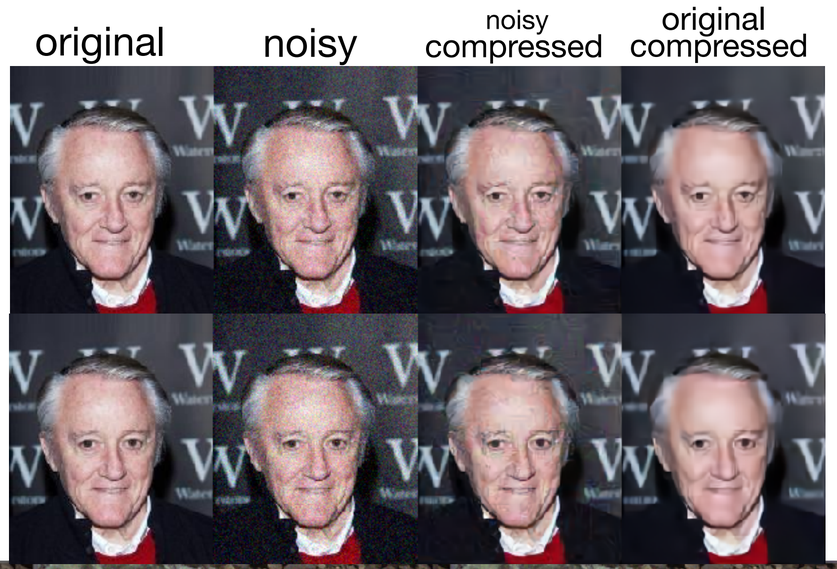
proof elements

- the r-d achieving test channel of the noisy source is the true posterior
- empirical distribution of good codes ~ the r-d achieving test channel
- new Markov lemma => a *conditionally independent* sample from the posterior

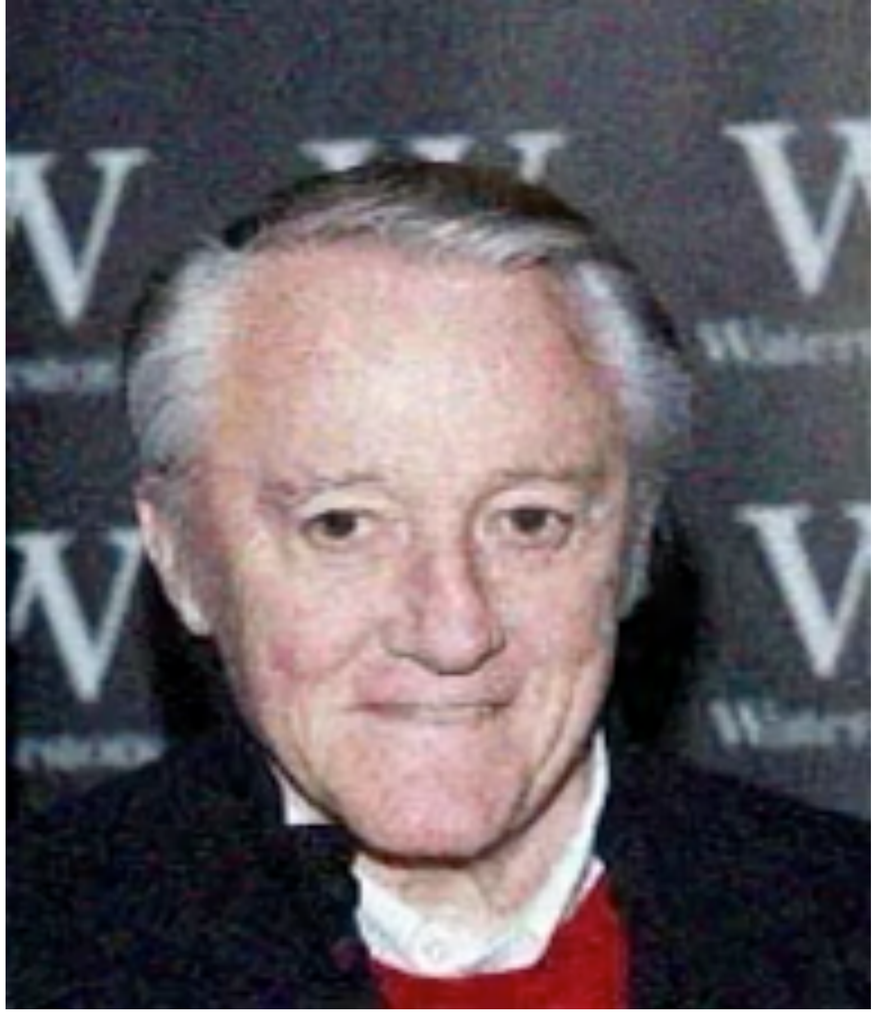
benefits

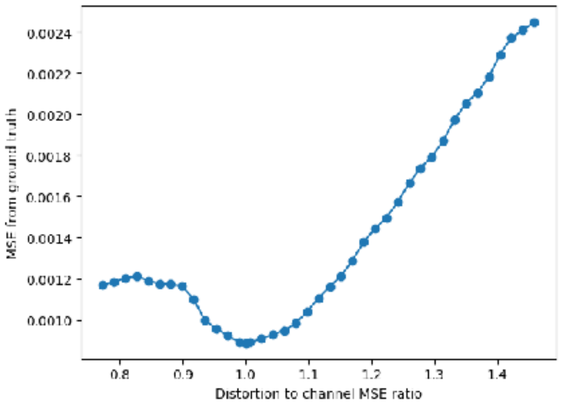
- generalizes, strengthens, and bridges gaps in the literature
- oblivious to the loss function under which denoising is judged
- yields schemes with “perfect perception” a la “rate-distortion-perception”

AWGN
+
squared
error
distortion



(also yields
state of the art
rate-distortion-
perception
performance)





Unsupervised Audio Denoising via Neural Lossy Compression

- **Motivation:** Neural codecs for audio are increasingly prevalent which much success. Neural compressors offer better rate-distortion performance than traditional compressors. Since audio is 1-dimensional, we can obtain a good lossy compressor with a small-scale network.
- **Approach:** We train a scale-hyperprior neural compressor on **noisy audio** samples to construct a good lossy code for the given noise mechanism. The learned compressor can be applied to the target noisy signal, which upon reconstruction, gives us a denoised signal. Thus, we obtain **joint audio compression and enhancement**.
- **Unsupervised Benefit:** There have been recent attempts at this joint problem, such as SoundStream [1] and ADNAC [2]. However, these approaches require access to clean-noisy audio pairs, which may not always be viable. Our approach only requires access to the noisy data.
- In addition to compression and denoising, we have a positive byproduct of bandwidth reduction.

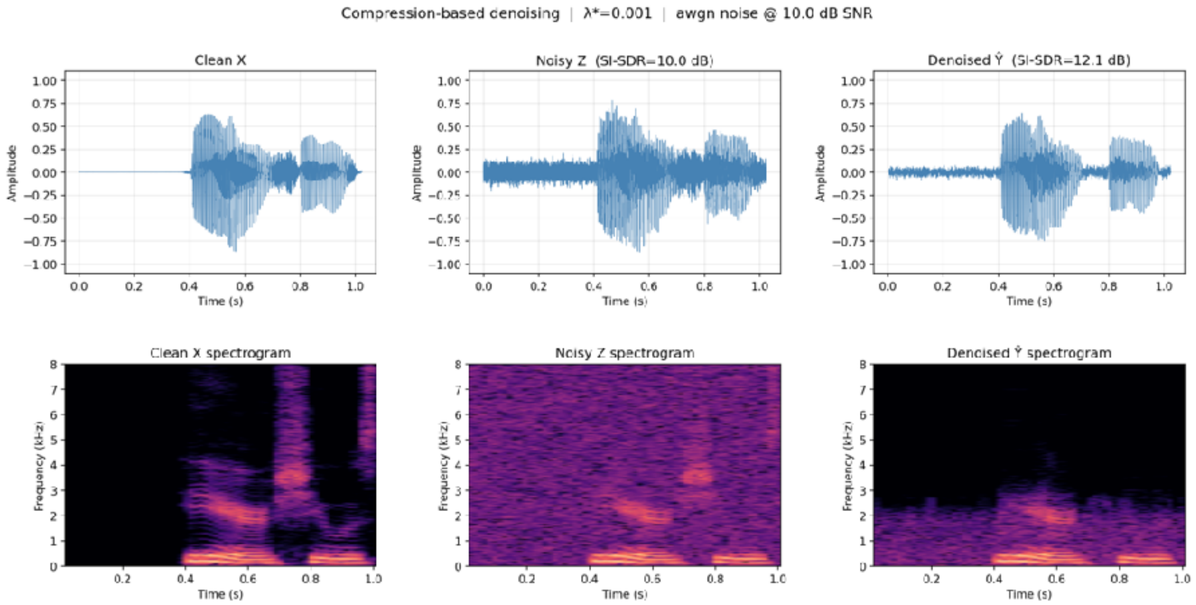


Figure: *Clean, noisy, and denoised signals at 10dB SNR AWGN noise.*

[1] "SoundStream: An End-to-End Neural Audio Codec," N. Zeghidour et al.
[2] “ADNAC: Audio Denoiser using Neural Audio Codec,” D. Jimon et al.

Results and Takeaways:

- **Compression and Denoising Performance:**
  - At 10dB SNR, we get ~4dB gain and ~12x compression.
  - At 5dB SNR, we get ~6dB gain and ~20x compression.
- **Classical Baselines:** We compare the denoising performance of our compressor to classical methods such as lowpass filtering, spectral subtraction, and Wiener filtering. In terms of SI-SDR, our approach outperforms these methods
- **Neural Audio Codecs:** We look at two off-the-shelf neural audio codecs used in practice – EnCodec and DAC – and compare their rate-denoising performance to our approach
  - EnCodec and DAC are neural codecs trained on, and meant for, clean audio data
  - Our approach gets a better denoising gain than these codec, while achieving a comparable bitrate regime
- **Specialised Audio Denoisers:** We also compare our denoising performance to Facebook Demucs [1], which is a specialized audio denoiser trained in a supervised way, and outperforms our denoising capabilities. However, it offers no compression, and should be treated as a denoising benchmark

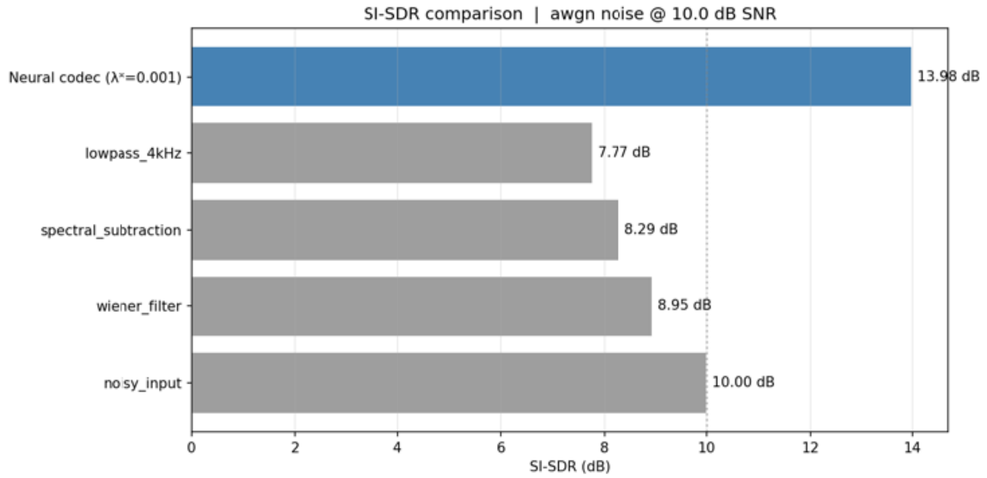


Figure: *Denoising performance at 10dB SNR against classical baselines.*

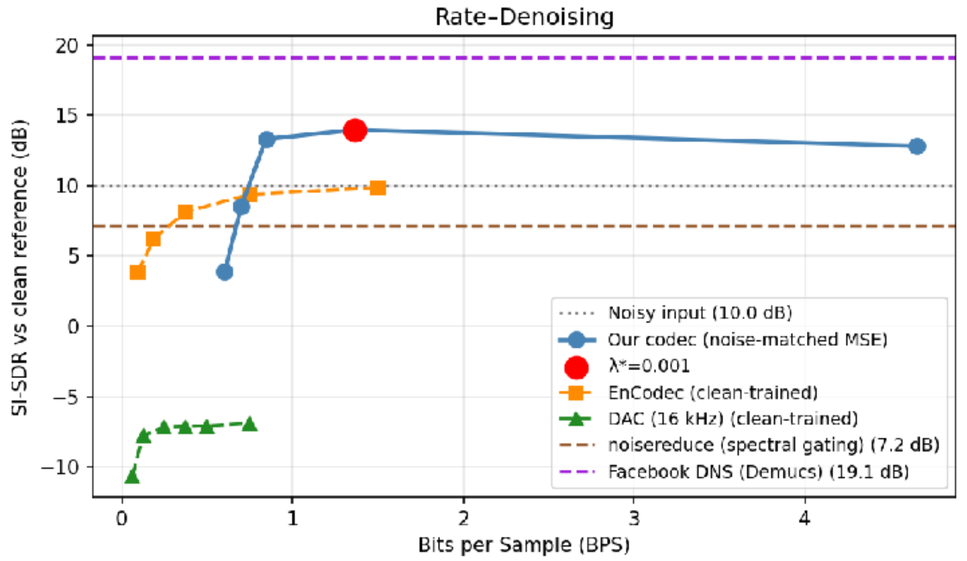


Figure: *Comparing rate-denoising performance at 10dB SNR against off-the shelf methods.*

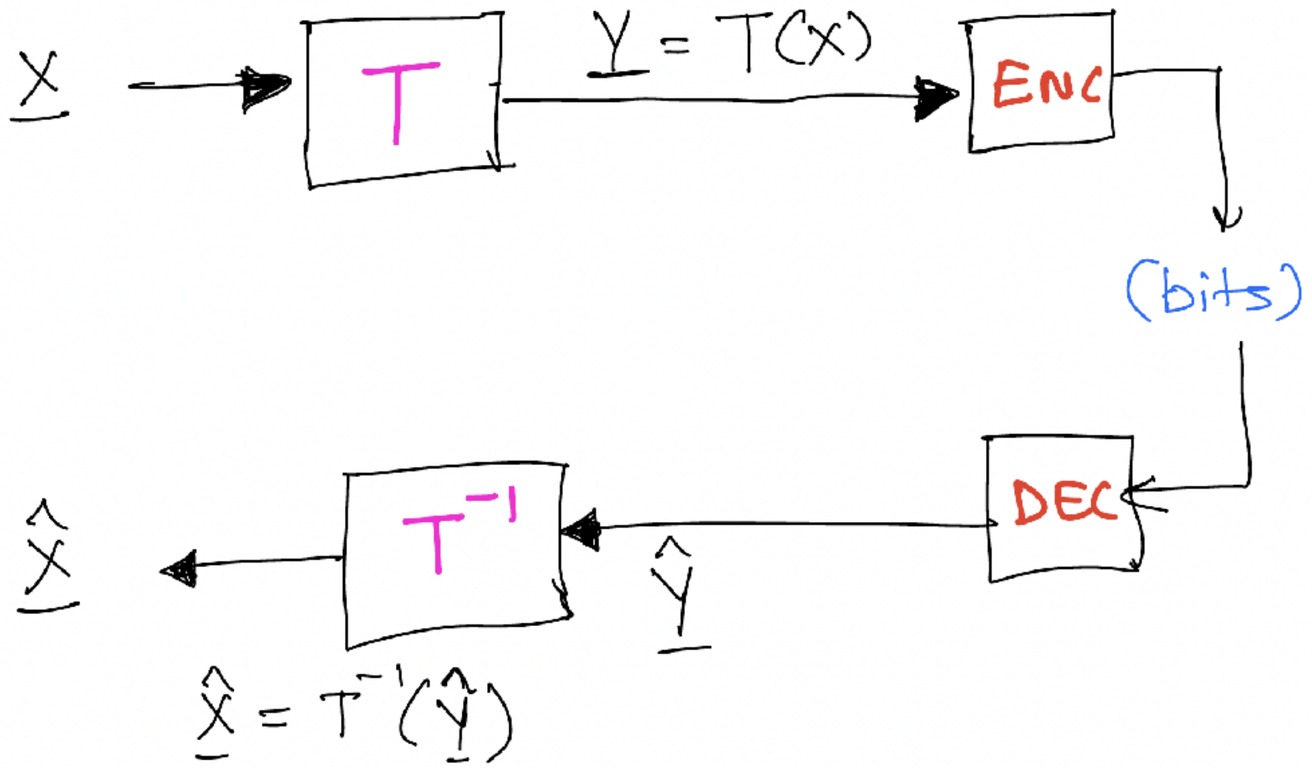
[1] “Real Time Speech Enhancement in the Waveform Domain,” A. Défossez et al.

C. Ding, A. Rao Gorle, J. Jeong, N. Sagan and T. Weissman,
"Information-computation trade-offs in non-linear transforms",
Philosophical Transactions of the Royal Society A, 2026
[Ding et al. 26]

among other highly non-linear transforms, considers:

- neural
- textual
- LZ

transform coding 101



elements of a good transform

- basis functions have biological/physical/conceptual significance
- sparsity
- transform coefficients related simply (e.g. independent, uncorrelated, etc.)
- invertibility (information lossless)
- smoothness of forward and inverse transform (w.r.t. relevant metrics)
- manageable complexity (both forward and inverse transform)
- induced benefit: yields a wieldy and natural model for your data

quest for a new transform

- T. Weissman, “Toward textual transform coding”, IEEE BITS, Vol 3, Issue 2, June 2023, [W23]

- what resonates with us?
- what is a ‘thing’?
- what lights up our (biological) neural nets?

what are the right “harmonics”?

words

- are what we've invented/evolved to refer to and describe concepts/things that resonate with/are meaningful to us

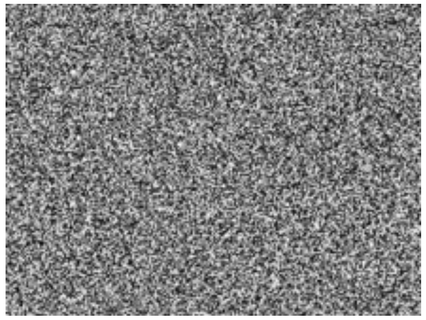
analogy toward a “textual transform”

Fourier transform	Textual transform
harmonics	objects for which we have words
frequencies	words
amplitude/phase of harmonics	size/location/orientation of objects
precision of amplitude/phase	precision of size/location
frequency resolution	number of words

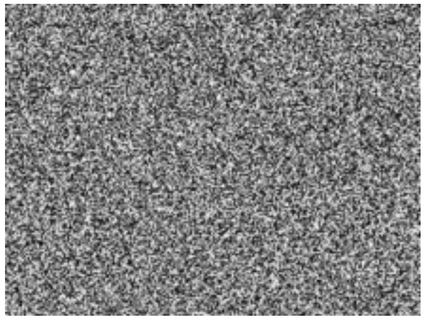
(some cartoonish sketches of)

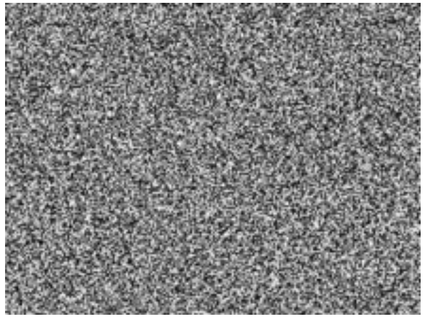
examples for what such a transform might look like

white noise

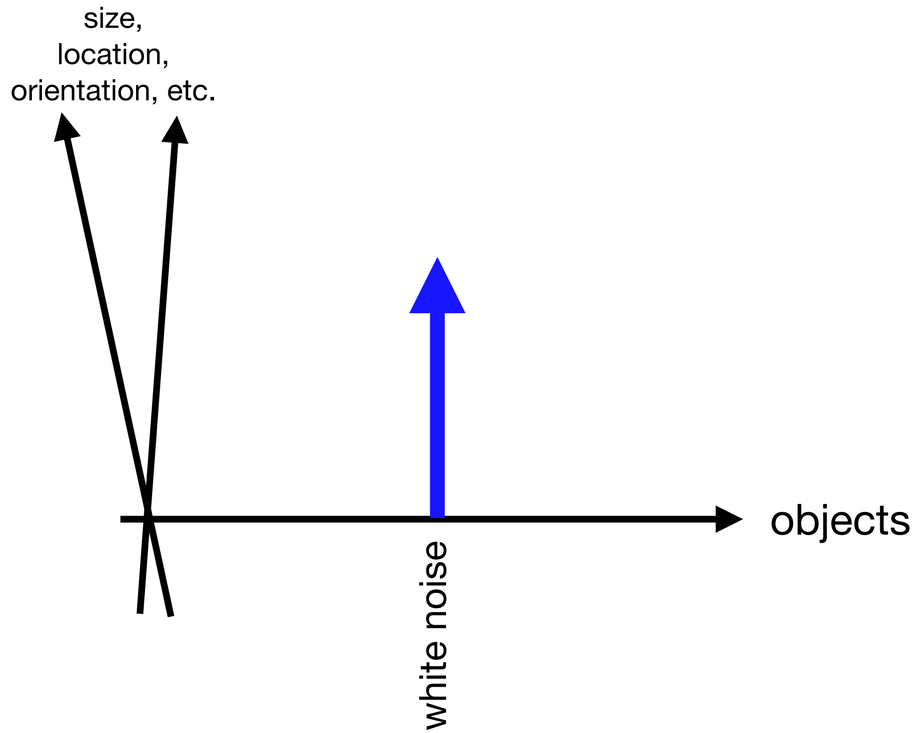


FFT, etc.

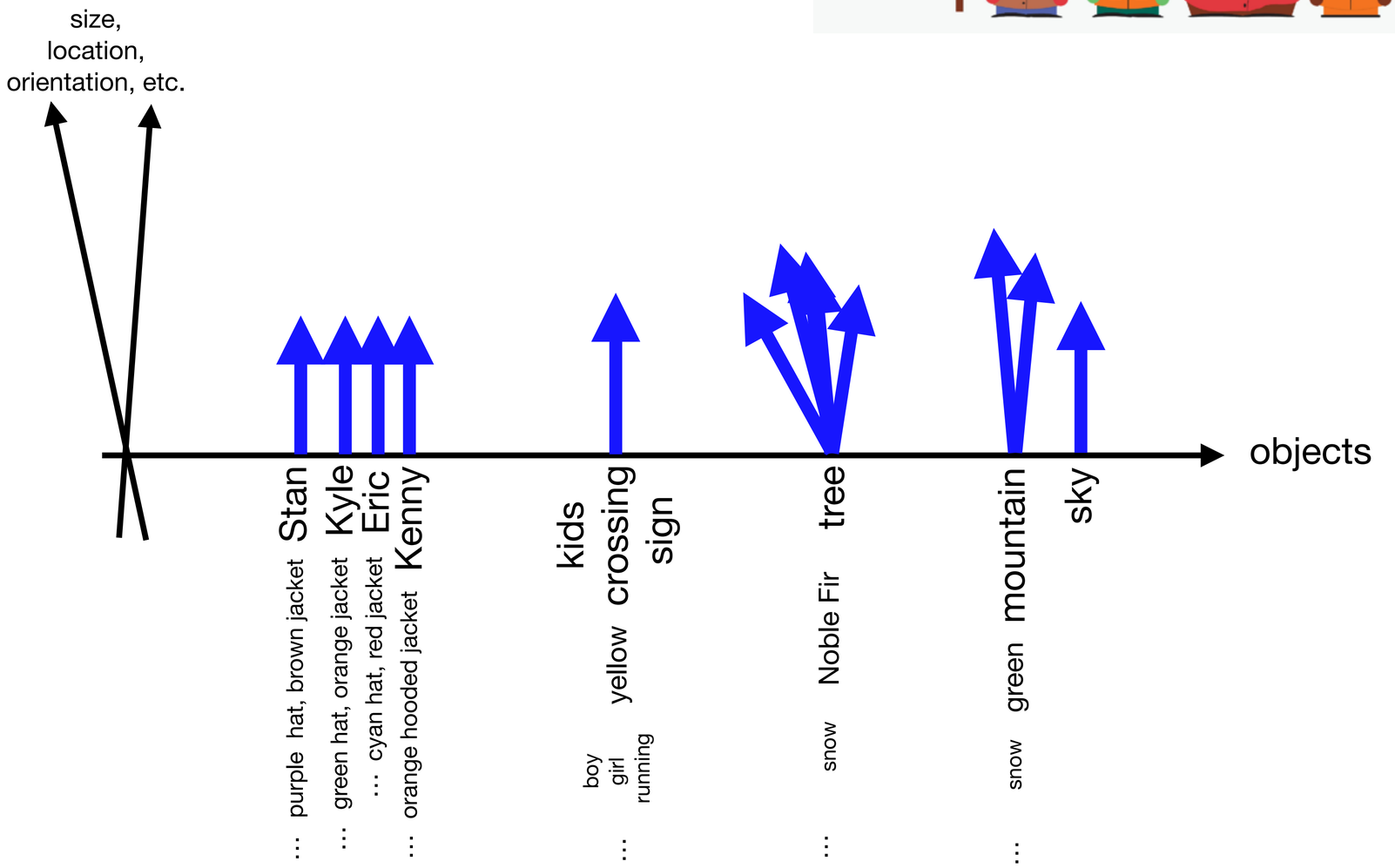




Textual Transform



South Park image



representation of an image in the textual transform domain

- how many objects (up to prescribed number L for cognitive load)
- list of what they are (one word each)
- their sizes, locations, orientations (at prescribed resolution R)
- more words of description for each and how they relate to each other (at prescribed resolution W)

in other words:
essentially what a verbal description would look like
(confined to a given budget of words/sentences)

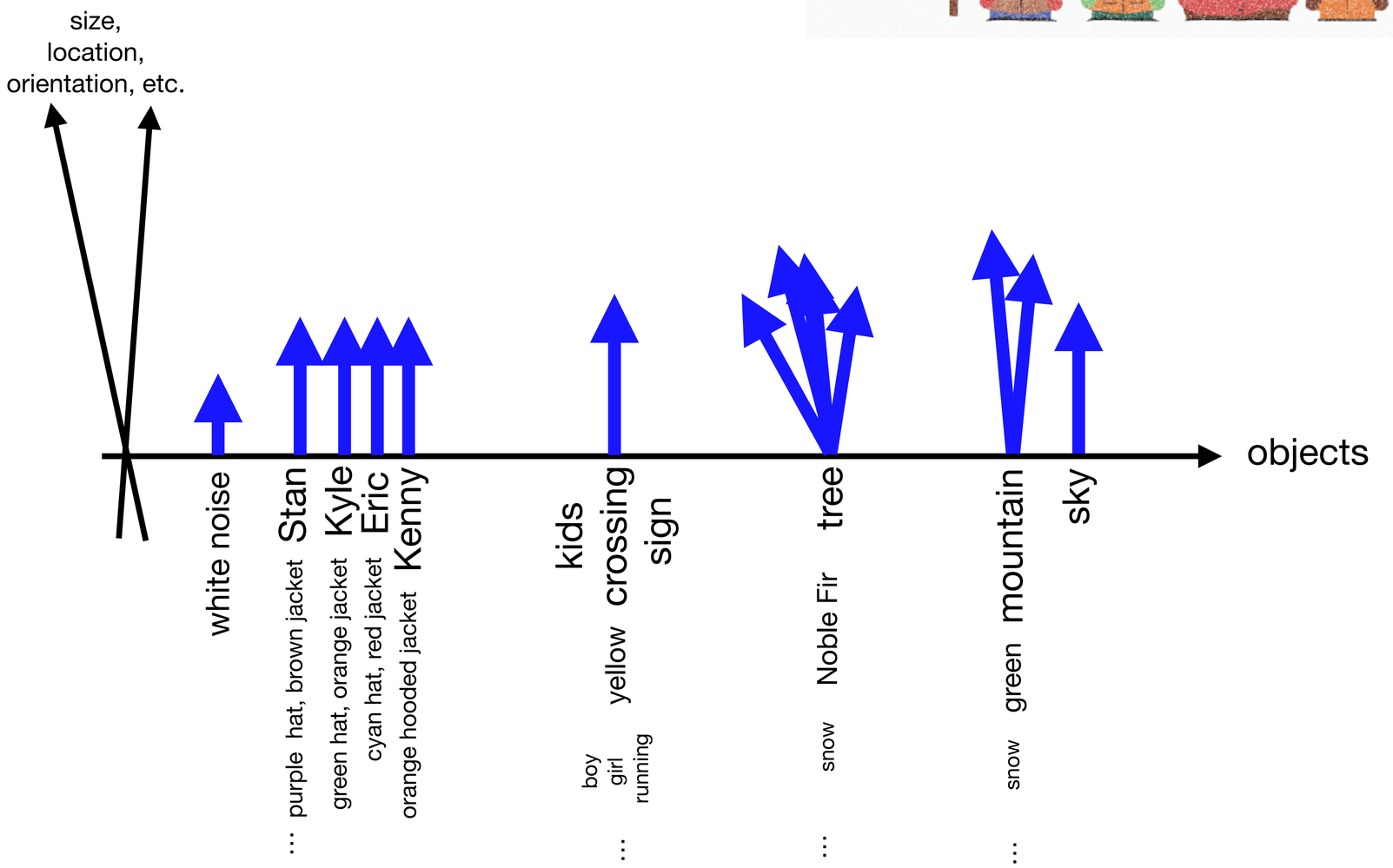
textual transformers

- transformers (Large Language and visual models), diffusion models, etc.
- computationally effective textual transforms
- humans 
- ongoing: text emerging from extreme lossy compression

a bit on denoising and lossy compression via the textual transform:

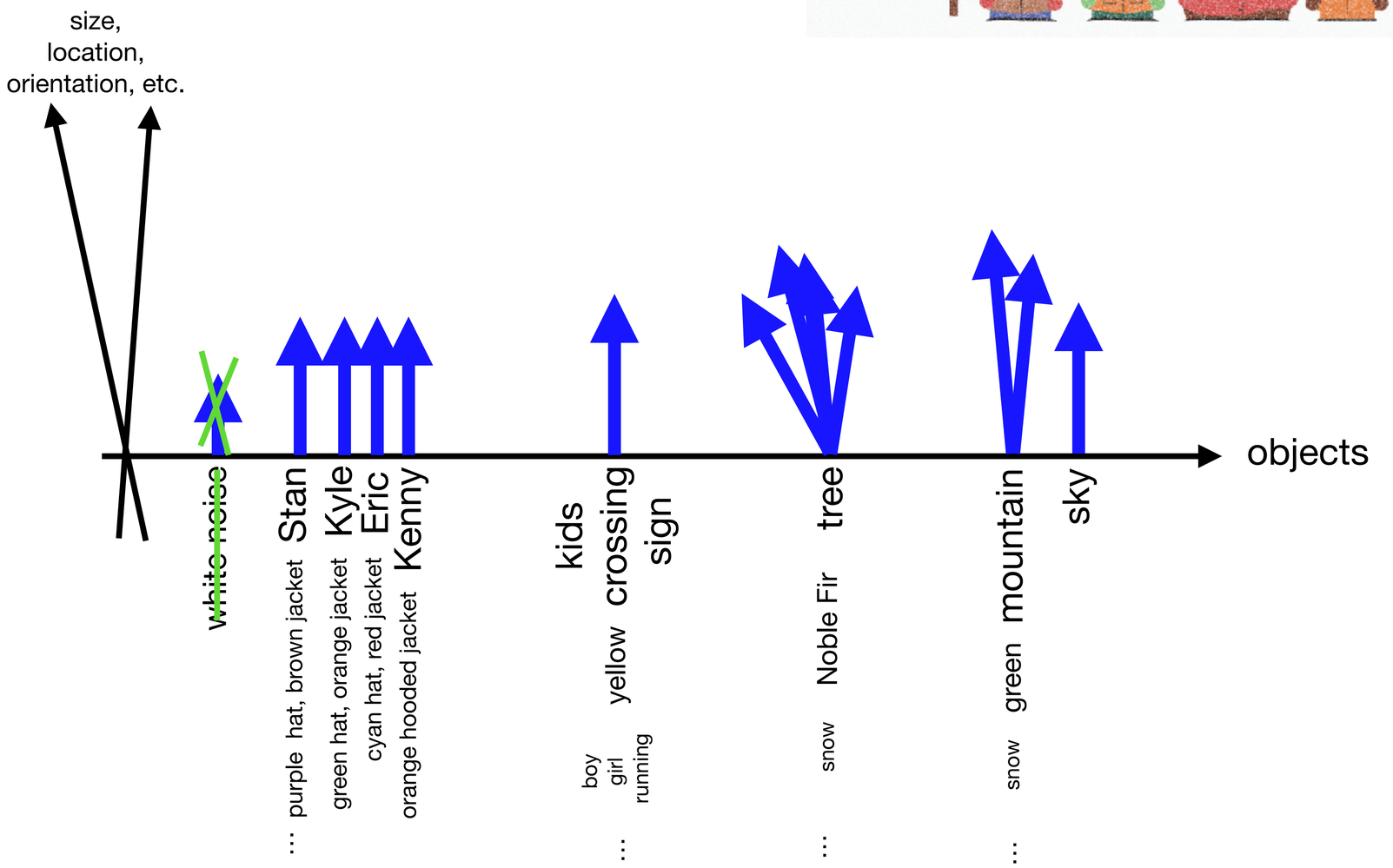
noisy

South Park image



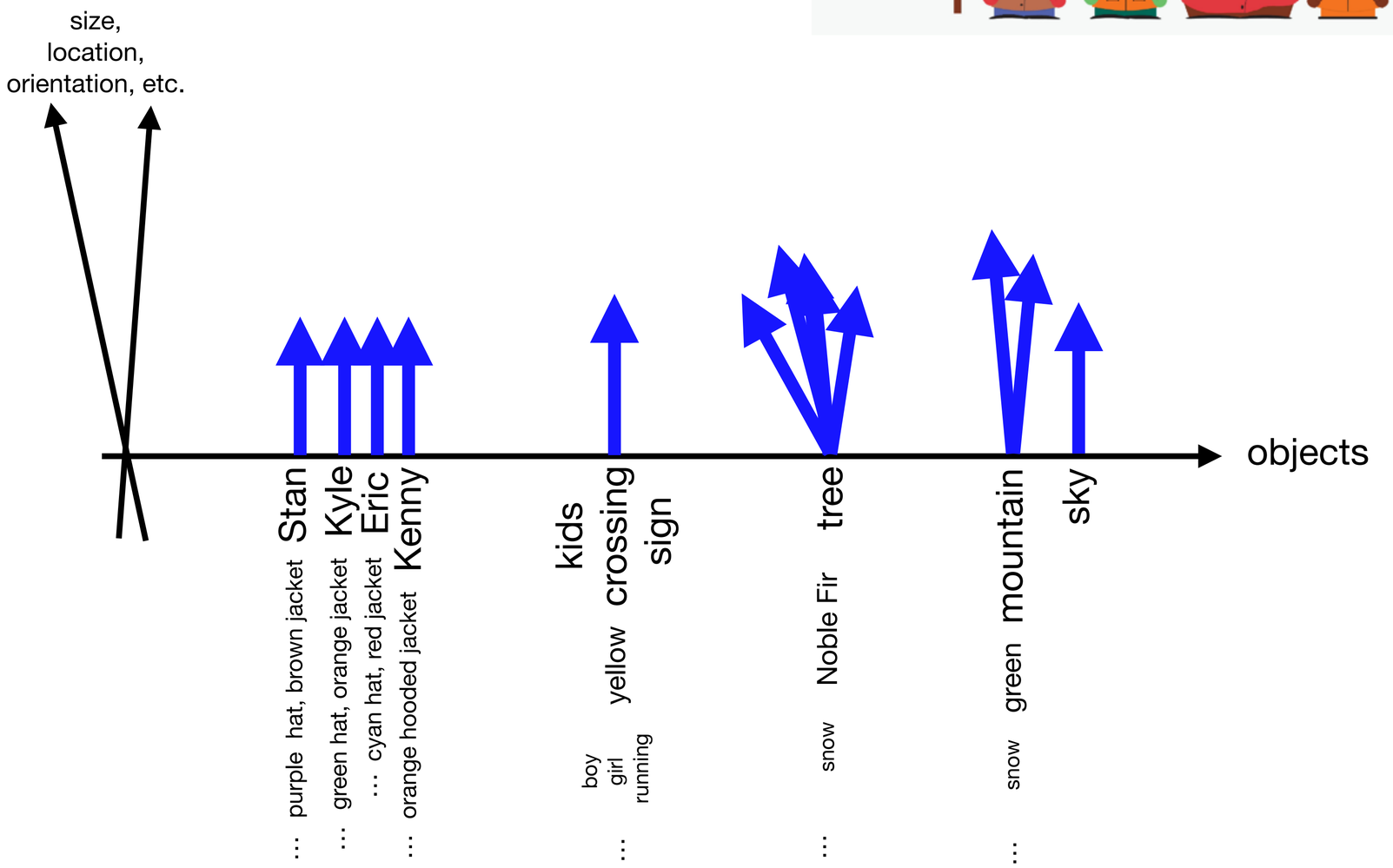
denoising the

South Park image



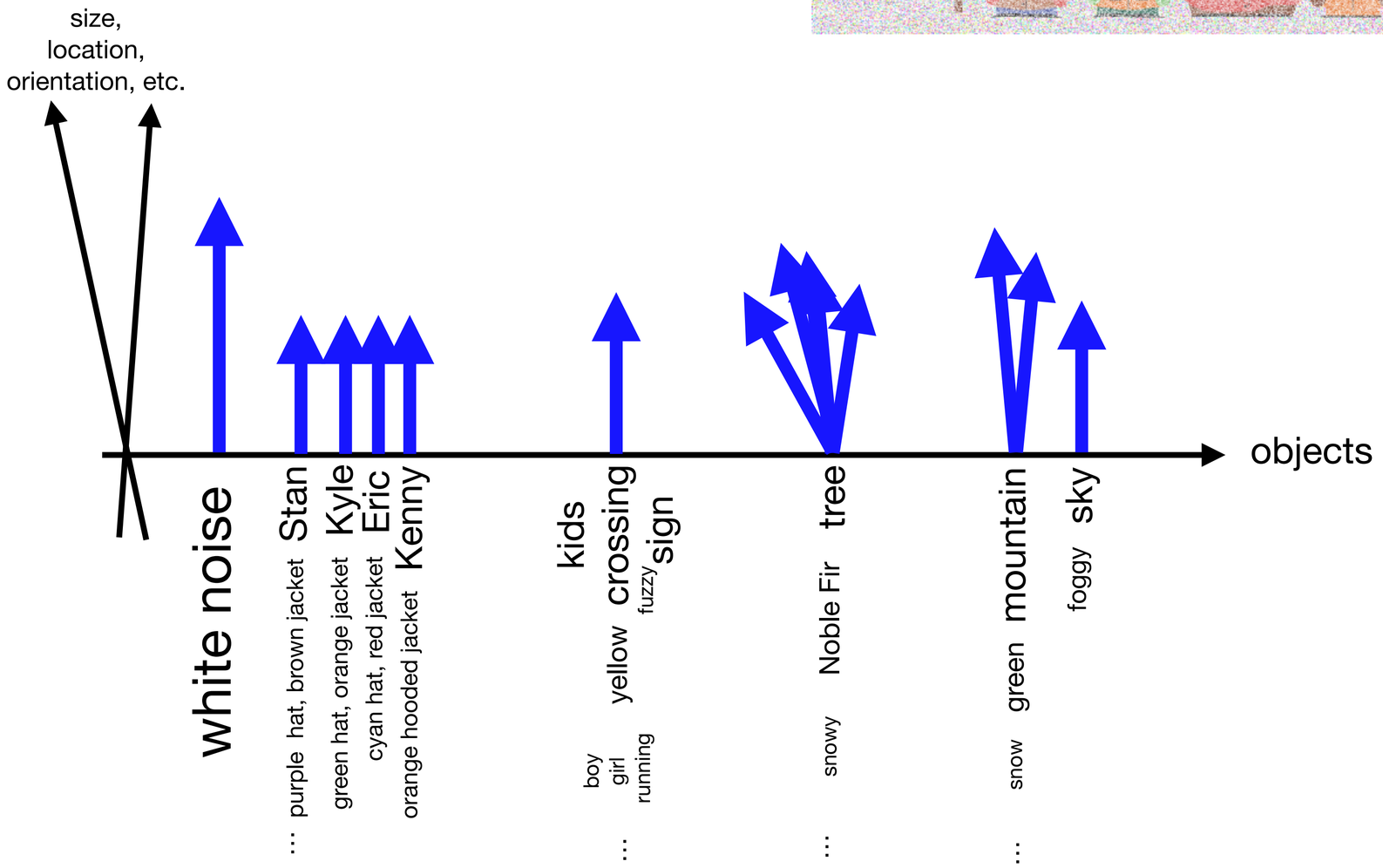
denoised

South Park image



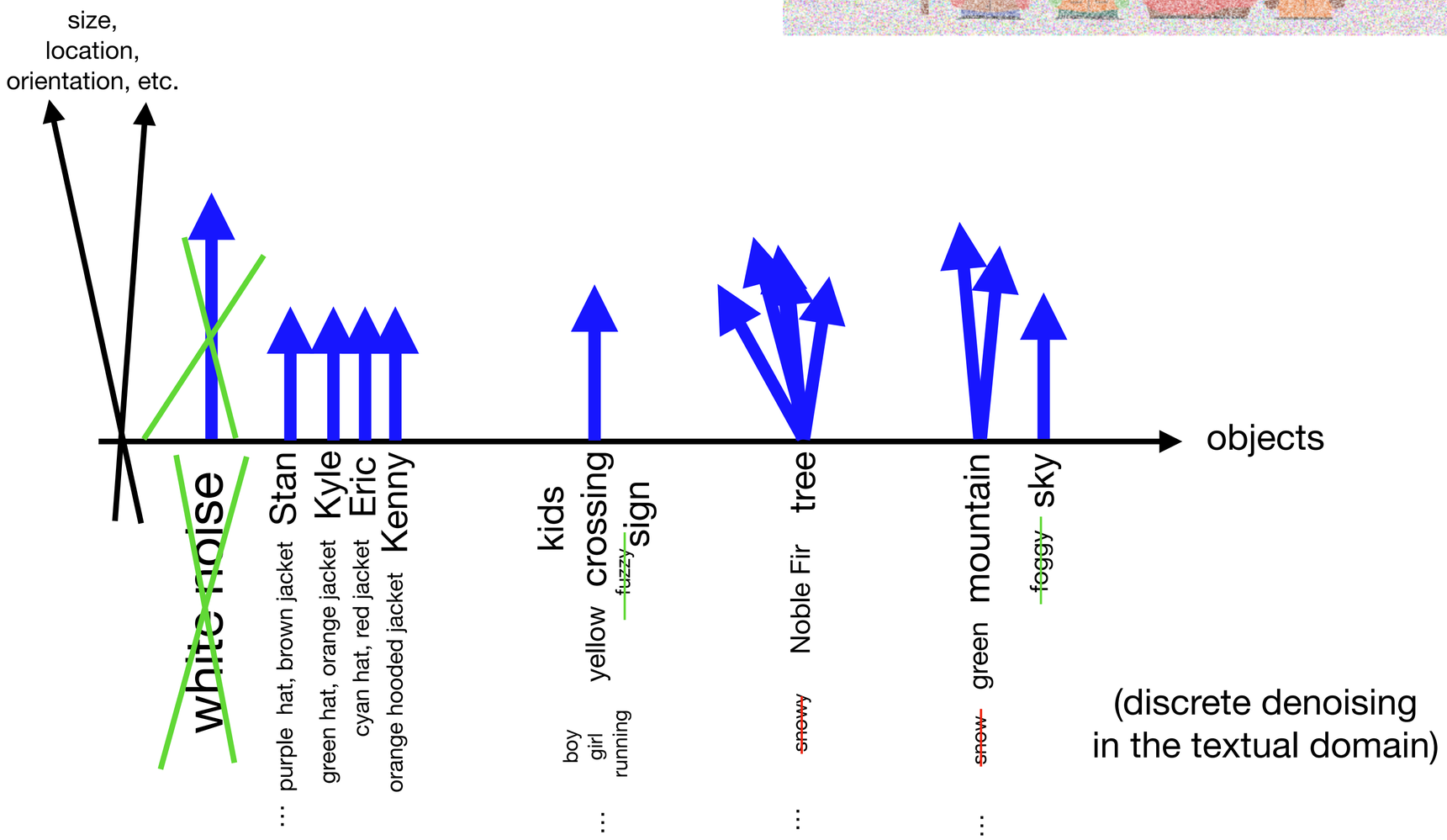
noisier

South Park image



denoising the noisier

South Park image



[Ding et al. 26]: **goes way beyond the cartoons to yield state of the art lossy compression and denoising**

via the textual transform + visual sketches

some takeaways

- NNs, transformer and diffusion based architectures have shown us what’s achievable
- existing compressors such as LZ hint that NNs are highly suboptimal from a compute perspective
- “plenty of room at the bottom” for new architectures
- at compute regimes >> compressors and << NNs
- need go back to the drawing board
- seek more inspiration from humans and other (not necessarily neuronal) biological systems
- new architectures could give rise to new types of intelligence

thanks