Compression Meets Sampling: On Energy-Efficient Random Variate Generation

Johann Ukrow*, **Anna Kazachkova***, **Nicolas Alder, Sven Köhler, Ralf Herbrich**
Hasso Plattner Institute
{johann.ukrow,anna.kazachkova,nicolas.alder,
sven.koehler,ralf.herbrich}@hpi.de

Abstract

Generating (pseudo-)random variates lies at the core of probabilistic machine learning and prediction algorithms and yet remains a major bottleneck due to its high computational and energy cost. In this paper, we introduce a general and scalable sampling strategy that enables fast and energy-efficient random variate generation from arbitrary distributions. Our approach is based on efficient lookup tables combined with a fast index sampling scheme. Using only a handful of fast and energy-efficient compute operations on simple array structures, we achieve superior speed, energy efficiency, and precision at near-optimal entropy cost compared to state-of-the-art techniques. Our method can be easily integrated into existing machine learning pipelines, outperforming commonly used samplers.

1 Introduction

Sampling from probability distributions is a fundamental yet computationally expensive operation in machine learning. On digital computers, sampling from arbitrary distributions is reduced to sampling from finite distributions due to fundamental constraints of finite precision and memory. All distributions, whether continuous or infinite discrete, must be discretized for computational implementation (see Appendix, Section A). Standard sampling algorithms in widely used libraries assume infinite precision arithmetic, where computation can be performed with arbitrarily precise real numbers Shamos [1978]; (Devroye [1986], Chapter 2, p.1, Assumption 1). Additionally, they assume the ability to generate infinitely precise samples from the real unit interval (Devroye [1986], Chapter 2, p.1, Assumption 2). However, actual implementations rely on finite floating-point representations and don't have access to exact samples from the real unit interval, causing generated distributions to deviate from their intended target distributions in uncontrolled ways. These deviations are often intractable to quantify, precluding theoretical guarantees about sampling accuracy.

This work introduces a new sampling approach for arbitrary distributions based on operations with lookup tables. Besides being a generic method, our approach is especially suitable for scenarios demanding precise sampling, situations where floating-point operations are either unavailable or too error-prone, and situations with low power supply. We summarize our contributions as follows:

1. We introduce an efficient random variate generator based on compressed lookup tables (cLUT), achieving an exponential compression ratio.
2. We compare a C implementation of cLUT against state-of-the-art approaches. It saves 30-40% sampling time and saves 25-50% of energy across a diverse set of distributions.
3. We compare against commonly used Python libraries, achieving 10-100$\times$ acceleration.

*Equal contribution.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Frontiers in Probabilistic Inference: Sampling Meets Learning.

2 Approach

We will describe a method to generate random variates from any finite discrete distribution, represented by n probabilities $p_1, \ldots, p_n \in [0, 1]$ of the n outcomes $x_1, \ldots, x_n \in \mathcal{X}$. Ideally, we would construct a lookup table containing duplicates of each outcome proportional to its probability:

$$\frac{\text{occurrences of } x_i \text{ in table}}{\text{table size}} = p_i. \tag{1}$$

Sampling would then reduce to uniformly selecting a random table index $I \sim \text{Uniform}\{1, \ldots, N\}$ and returning $S = \text{Table}[I]$, where N is the table size. See Figure 1a for an example lookup table.

In practice, memory constraints bound the table size N, limiting representable probabilities to multiples of $1/N$. Approximating probabilities to precision b bits requires quantizing each p_i to $f_i = \text{round}(p_i \cdot 2^b)$, yielding frequencies $\mathbf{f}=(f_1, \ldots, f_n)$ and a table of size $N = \sum_i f_i = 2^b$ (see the Appendix for error analysis). While approximation error decreases logarithmically with b, memory requirements grow exponentially, making high-precision sampling prohibitive. The basis of our main approach is a lossless compression strategy for the lookup tables and the following sampling scheme.

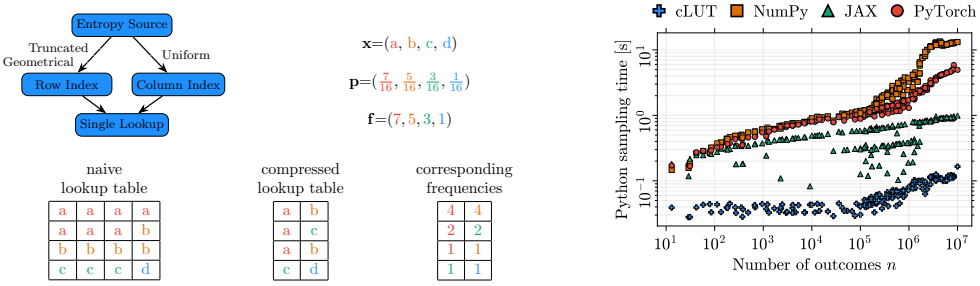
Compression scheme To tackle the prohibitive memory requirements of lookup tables, we propose to use the following compression scheme. Intuitively, the compressed lookup table can be viewed as a two-dimensional array consisting of $r + 1$ rows and 2^c columns, with $r, c \in \{0, \ldots, b\}$ satisfying $2^{r+c} = 2^b = N$. Each row i of the first r rows corresponds to a frequency of 2^{r-i}, where row indices run from 1 to r. The r+1-th row corresponds to the same frequency as the r-th row, namely $2^{r-r} = 1$. For an exemplary compression and the corresponding frequencies, see Figure 1a. This lossless compression scheme preserves the total frequencies while drastically decreasing the size of the lookup table. Compressing a naive lookup table with $N = 2^{r+c}$ entries to a compressed lookup table with $(r + 1) \cdot 2^c$ entries yields a compression ratio of $\rho = 2^r/(r + 1)$.

Sampling step To generate a sample $S \in \mathcal{X}$ using a compressed lookup table, we generate two indices independently: a row index $I \in \{1, \ldots, r+1\}$ and a column index $J \in \{1, \ldots, 2^c\}$. We sample the index I according to a truncated geometric distribution, and the column index J uniformly:

$$\mathbb{P}(I = i) = \max(2^{-i}, 2^{-r}) \text{ for } i = 1, \ldots, r+1, \text{ and } \mathbb{P}(J = j) = 2^{-c} \text{ for } \quad j = 1, \ldots, 2^c.$$

Therefore, we sample a table-index $(I, J) = (i, j)$ with probability $2^{-\min(i,r)-c}$. The column index J can be efficiently sampled using any uniform sampler. The row index I can also be sampled extremely efficient using the entropy optimal procedure detailed in Algorithm 1 in lines 3-9. A sample is then generated by returning the value stored in the compressed lookup table at that index:

$$S = \texttt{compressedTable}[I, J].$$



(a) Schematic and example lookup tables.

(b) Average sampling time.

Figure 1: **a.** A schematic of the sampling step of cLUT (topleft), the naive and compressed lookup table for an example distribution (bottom row) on $\mathbf{x}$ given by $\mathbf{p}$ (topright). The naive lookup table (left) contains each value according to its frequency $\mathbf{f}$ at a precision of $b = 4$ bits. The compressed lookup table (middle) stores the same distribution when considering the frequency scheme (right). **b.** Average wall time (seconds) to generate 10^7 samples from distributions of varying sizes $n \in [10^1, 10^7]$, compared against standard methods from popular machine learning libraries.

Algorithm 1 Sampling from compressed lookup tables

Require: number of samples K, compressed lookup table `compressedTable` of size $(r+1) \times 2^c$
Ensure: array of samples S

for $k = 1$ **to** K **do**
  $I \leftarrow 1$
  while $I < r + 1$ **do** *// Sample row index geometrically:*
    if `randomBit`$() = 1$ **then**
      break
    $I \leftarrow I + 1$
  $J \leftarrow \text{Uniform}\{1, \ldots, 2^c\}$ *// Sample column index uniformly.*
  $S[k] \leftarrow$ `compressedTable`$[I, J]$ *// Generate a sample from the distribution.*
return S

Table 1: Average wall time (in seconds, mean $\pm$ std) for generating 10^7 samples.

# Outcomes:	$n \in [10^1, 10^5)$		$n \in [10^6, 10^7)$	
Method	**Sampling time (s)**	**Preprocessing time (s)**	**Sampling time (s)**	**Preprocessing time (s)**
NumPy	0.6680 ± 0.2650	0.0001 ± 0.0001	9.6248 ± 3.5823	0.0308 ± 0.0202
PyTorch	0.6073 ± 0.2436	0.0003 ± 0.0006	3.3768 ± 1.0615	0.1028 ± 0.0655
JAX	0.3647 ± 0.1277	0.2898 ± 0.0894	0.7982 ± 0.1815	0.6528 ± 0.0948
cLUT	0.0374 ± 0.0051	0.0006 ± 0.0011	0.1016 ± 0.0129	0.3925 ± 0.2219

Preprocessing step Before sampling, we must construct the compressed table. Conveniently, we do not have to construct the uncompressed lookup table. We rather construct the compressed lookup table directly from the binary expansion of the frequencies f_i. A value x_i appears in row j if and only if the j-th bit $f_i^{(j)}$ of f_i is one. The frequencies $\mathbf{f}$ can be adjusted to sum to exactly 2^b by using a sum-preserving rounding scheme, making our sampling procedure rejection-free. To improve the sampling speed, we ensure that all rows have uniform width as detailed in Algorithm 2.

3 Evaluation

To demonstrate the advantage of our approach, we compared it to state-of-the-art solutions. We evaluate all methods on a fixed set of synthetically generated distributions of sizes $n \in [10^1, 10^7]$ drawn from exponential distributions with varying parameters, with zero probabilities removed. The bit precision is set to $b = 16$ for $n \in [10^1, 10^4)$, $b = 20$ for $n \in [10^4, 10^6)$ and $b = 23$ for $n \in [10^6, 10^7]$. For more details on the evaluation see Appendix, Section B & C.

Sampling speed in Python We benchmarked our method against standard discrete sampling routines from widely used Python machine learning libraries: `RandomGenerator.choice()` from NumPy Harris et al. [2020], `multinomial()` from PyTorch Paszke et al. [2019], and `random.choice()` from JAX Bradbury et al. [2018]. As shown in Figure 1b, our method achieves a 10–100$\times$ speedup across a wide range of distributions. Table 1 reports a $10\times$ improvement for distributions with 10^1 to 10^5 entries, with the speedup growing to over $100\times$ already for distributions with 10^6 to 10^7 entries. One should note that the samplers in NumPy, PyTorch, and JAX build on the Inversion method Devroye [1986] and produce distributions that are only approximately similar to the desired distribution, whereas our proposed method produces exactly the specified distribution.

Sampling speed and energy consumption in C vs. SOTA We compare our cLUT sampler to the following SOTA sampling methods: the alias method Walker [1974], ALDR, and FLDR Draper and Saad [2025]. Figure 2 shows our results. Our cLUT method outperforms the Alias method, ALDR, and FLDR for almost all distributions except then very small ones ($n \leq 10^2$). For very small distributions, ALDR and FLDR appear to be the better choice. The preprocessing phase (look-up table creation) scales log-linear with the distribution size across all investigated methods. Our cLUT method shows the highest time demand for the preprocessing phase and the alias method the lowest one, but is then more time efficient in sampling.

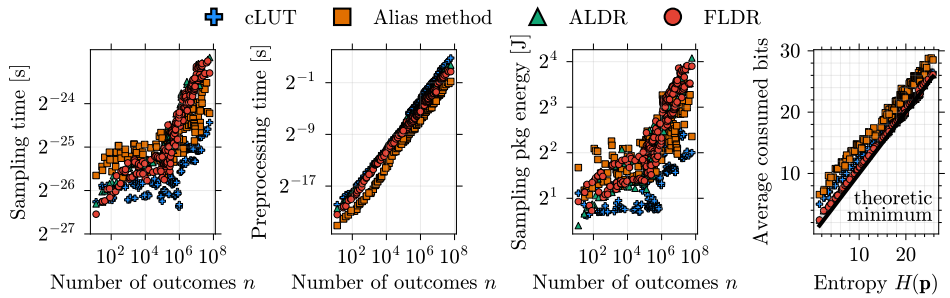


Figure 2: Comparison of our cLUT approach with existing state-of-the-art sampling methods. We ran 10^6 repetitions of the sampling operation and 10 repetitions of the preprocessing phase. Shown are (1) the mean wall time required for sampling and (2) preprocessing, as well as the (3) cumulative energy demand of the CPU socket of all sampling runs. Time and energy are shown on a log-log scale. The fourth subfigure shows the average consumed bits per sample from the entropy source.

Table 2: Averaged energy demand, wall time, and power draw of a single sampling operation.

# Outcomes:	$n \in [10^3, 10^4)$			$n \in [10^7, 10^8)$		
Method	**Energy (nJ)**	**Time (ns)**	**Power (W)**	**Energy (nJ)**	**Time (ns)**	**Power (W)**
ALDR	263.451 ± 50.309	22.431 ± 1.250	11.836 ± 2.694	1223.520 ± 194.457	102.792 ± 12.066	12.168 ± 2.899
Alias method	319.804 ± 72.045	26.803 ± 2.934	12.156 ± 3.549	887.653 ± 185.627	55.946 ± 13.709	16.502 ± 3.897
FLDR	290.223 ± 47.274	21.268 ± 2.373	14.031 ± 3.770	1214.382 ± 177.125	101.404 ± 9.702	12.195 ± 2.753
cLUT	199.233 ± 38.579	15.475 ± 1.689	13.271 ± 4.091	450.155 ± 74.604	33.026 ± 4.880	14.188 ± 4.188

To demonstrate the energy-saving potential of our approach, we compare the energy demand of all implementations. Again, ALDR and FLDR have some advantage for smaller distributions, but our cLUT approach works best across all sizes. Although the energy demand roughly follows the same trend as the required time, the scale is not linear. This is because time is not an accurate indicator of the energy required by complex real-life computing systems. Rather, energy is the integral of the dynamic power demand over time.

Bit efficiency vs. SOTA Sampling algorithms are commonly evaluated based on the average number of i.i.d. $\mathrm{Bernoulli}(0.5)$ bits required to generate a single sample. The cLUT method requires an expected number of $b - r + 2 - 2^{-(r-1)}$ bits per sample. Empirical evaluations indicate that our method is close to the information-theoretic minimal cost of sampling ($-\sum_i p_i \cdot \log_2(p_i)$, see Knuth and Yao [1976]) and approaches the minimum for high-entropy distributions (see Figure 2).

4 Conclusion

We present cLUT, a new fast and energy-efficient sampling method for sampling from arbitrary distributions, based on operations with compressed lookup tables. Time to sample a distribution speeds up 10-100× compared to commonly used machine learning Python libraries. It saves up to 50% in energy compared to state-of-the-art methods. We have not vectorized or parallelized our implementation to improve understandability and facilitate comparison with other methods. However, our sampling method only requires a single index-based memory lookup and some arithmetic and bit-shift operations. This makes it better suited than competing approaches for single instruction, multiple data devices Flynn [1972], such as modern vector and graphics processing units, given a compatible streaming source of entropy.

References

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

H.-C. Chen and Y. Asau. On Generating Random Variates from an Empirical Distribution. *A I I E Transactions*, 6(2):163–166, June 1974. ISSN 0569-5554. doi: 10.1080/05695557408974949. URL http://www.tandfonline.com/doi/abs/10.1080/05695557408974949.

H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le. Rapl: memory power estimation and capping. In *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '10, page 189–194, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450301466. doi: 10.1145/1840845.1840883.

L. Devroye. *Non-Uniform Random Variate Generation*. Springer New York, New York, NY, 1986. ISBN 978-1-4613-8645-2 978-1-4613-8643-8. doi: 10.1007/978-1-4613-8643-8. URL http://link.springer.com/10.1007/978-1-4613-8643-8.

L. Devroye and C. Gravel. Random variate generation using only finitely many unbiased, independently and identically distributed random bits, Nov. 2020. URL http://arxiv.org/abs/1502.02539. arXiv:1502.02539 [cs].

T. L. Draper and F. A. Saad. Efficient rejection sampling in the entropy-optimal range. *arXiv preprint arXiv:2504.04267*, 2025.

M. J. Flynn. Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, C-21(9):948–960, 1972. doi: 10.1109/TC.1972.5009071.

T. S. Hao and M. Hoshi. Interval algorithm for random number generation. *IEEE Trans. Inf. Theor.*, 43(2):599–611, Sept. 2006. ISSN 0018-9448. doi: 10.1109/18.556116. URL https://doi.org/10.1109/18.556116.

C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

D. Knuth and A. Yao. Algorithms and complexity: New directions and recent results. Academic Press, 1976. Section: The complexity of nonuniform random number generation.

M. Lipp, A. Kogler, D. Oswald, M. Schwarz, C. Easdon, C. Canella, and D. Gruss. PLATYPUS: Software-based Power Side-Channel Attacks on x86. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021.

J. Lumbroso. Optimal Discrete Uniform Generation from Coin Flips, and Applications, Apr. 2013. URL http://arxiv.org/abs/1304.1916. arXiv:1304.1916 [cs].

G. Marsaglia. Generating discrete random variables in a computer. *Communications of the ACM*, 6(1):37–38, Jan. 1963. ISSN 0001-0782, 1557-7317. doi: 10.1145/366193.366228. URL https://dl.acm.org/doi/10.1145/366193.366228.

G. Marsaglia, W. W. Tsang, and J. Wang. Fast generation of discrete random variables. *Journal of Statistical Software*, 11:1–11, 2004.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

F. Saad, C. Freer, M. Rinard, and V. Mansinghka. The Fast Loaded Dice Roller: A Near-Optimal Exact Sampler for Discrete Probability Distributions. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, pages 1036–1046. PMLR, June 2020. URL `https://proceedings.mlr.press/v108/saad20a.html`. ISSN: 2640-3498.

K. Schwarz. Darts, Dice, and Coins, Dec. 2011. URL `https://www.keithschwarz.com/darts-dice-coins/`.

M. I. Shamos. *Computational Geometry*. Yale University., 1978. Ph.D. dissertation.

T. Uyematsu and Y. Li. Two algorithms for random number generation implemented by using arithmetic of limited precision. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E86A:2542–2551, Oct. 2003.

A. J. Walker. New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters*, 10(8):127–128, 1974.

A Details on Non-Finite Distributions

Many distributions relevant to machine learning belong to the class of continuous, real-valued, univariate distributions, with the Gaussian distribution as a prominent example. These distributions are discretized in a computational setting, as hardware can only represent a finite set of values.

A natural discretization proceeds as follows. Let a distribution on $\mathbb{R}$ be specified via its cumulative density function F. To discretize it on a finite support $\mathcal{X} \subset \mathbb{R}$ ($|\mathcal{X}| < \infty$), e.g., the set of representable numbers in the IEEE 754 16-bit floating-point format, we define the probability mass function $p : \mathcal{X} \to [0, 1]$ of the discretized distribution by

$$p(x) \coloneqq \frac{1}{c}\Big[F\Big(\frac{x + x_+}{2}\Big) - F\Big(\frac{x + x_-}{2}\Big)\Big], \quad \forall x \in \mathcal{X},$$

where $x_+ \coloneqq \min\{y \in \mathcal{X} : y > x\}$ is the next number to the right of x in $\mathcal{X}$, and $x_- \coloneqq \max\{y \in \mathcal{X} : y < x\}$ is the next number to the left.

Special care is required for the extrema of $\mathcal{X}$. Let $x^{\max} \coloneqq \max \mathcal{X}$ and $x^{\min} \coloneqq \min \mathcal{X}$. The next numbers beyond these limits can be defined in two ways, depending on how you would like to attribute the probability mass of the tails:

1. *Finite tail extension:*

$$x_+^{\max} \coloneqq x^{\max} + \frac{x^{\max} - x_-^{\max}}{2},$$
$$x_-^{\min} \coloneqq x^{\min} - \frac{x_+^{\min} - x^{\min}}{2},$$

   which requires a normalization constant $c = 1 - F(x_+^{\max}) + F(x_-^{\min})$ to ensure that the discretized probability mass function sums to one.

2. *Infinite tail extension:* $x_+^{\max} \coloneqq +\infty$ and $x_-^{\min} \coloneqq -\infty$, in which case $c = 1$ suffices.

Discrete distributions with infinite support, such as the Poisson distribution over $\mathbb{N}_{\geq 0}$, also require truncation to be represented in finite precision. A common approach is to apply a cutoff.

Since memory constrains impose a boundary on the size N of any lookup table, a lookup table might suffer from the inability to represent certain probabilities, such as very small or irrational probabilities, e.g., $p_i = \sqrt{1/2}$. In these cases we fill the table according to the frequencies

$$f_i \coloneqq \mathrm{round}(p_i \cdot 2^b) \in \mathbb{N}_{\geq 0}, \quad i \in \{1, \ldots, n\},$$

where $\mathrm{round}(\cdot)$ is an arbitrary sum-preserving rounding scheme. The approximation error of a distribution stored in a lookup table with probabilities $f_i/2^b$ directly depends on the precision b, as an upper bound on the KL divergence can be expressed as a function of $\min_{1 \leq i \leq n} f_i$ (see Theorem 1).

Theorem 1 (KL-Divergence of approximated distribution)**.** *The KL-Divergence between a distribution on* $\mathbf{x}=(x_1 \ldots, x_n)$ *given by the associated probabilities* $\mathbf{p}=(p_1, \ldots, p_n)$ *and the distribution approximated to a precision of* $b \in \mathbb{N}_{>0}$ *bits given by the frequencies* $\mathbf{f}=(f_1, \ldots, f_n)$ *is bounded by*

$$D_{KL}\big(\mathbf{p} \,||\, \mathbf{f}\big) \leq \log\Big(1 + \frac{1}{2\kappa}\Big),$$

where $\kappa \coloneqq \min_{1 \leq i \leq n} f_i$.

Proof. Write

$$p_i = f_i \cdot 2^{-b} + \delta_i,$$

with $\delta_i \in [-2^{-b-1}, 2^{-b-1}]$. Then, the KL-Divergence is given by

$$\begin{aligned}
D_{\mathrm{KL}}\big(\mathbf{p} \,||\, \mathbf{f}\big) &= \sum_{i=1}^{n} p_i \log \frac{p_i}{f_i \cdot 2^{-b}} \\
&= \sum_{i=1}^{n} p_i \log \frac{f_i \cdot 2^{-b} + \delta_i}{f_i \cdot 2^{-b}} \\
&= \sum_{i=1}^{n} p_i \log \left(1 + \frac{\delta_i \cdot 2^b}{f_i}\right) \\
&\leq \sum_{i=1}^{n} p_i \log \left(1 + \frac{2^{-b-1} \cdot 2^b}{\min_i f_i}\right) \\
&= \log \left(1 + \frac{1}{2 \min_i f_i}\right) \cdot \sum_{i=1}^{n} p_i \\
&= \log \left(1 + \frac{1}{2\kappa}\right)
\end{aligned}$$

where $\kappa \coloneqq \min_{1 \leq i \leq n} f_i$. We used that $\delta_i \leq 2^{-b-1}$ and $f_i > \min_i f_i$ for all i. □

Note that $\kappa = \min_{1 \leq i \leq n} f_i = \min_{1 \leq i \leq n} \mathrm{round}(p_i \cdot 2^b)$ and therefore $D_{\mathrm{KL}} \in \mathcal{O}(\log(1 + 2^{-b}))$. Clearly, $D_{\mathrm{KL}} \to 0$ for $b \to \infty$. However, while approximation error decreases logarithmically with precision b, the lookup table size N required to store all values $\mathbf{x}$ with their respective frequencies $\mathbf{f}{=}(f_1, \ldots, f_n)$ grows exponentially in b:

$$N \coloneqq \sum_{i=1}^{n} f_i = 2^b.$$

B Implementation Details

A pseudo code of the cLUT preprocessing algorithm is shown in Algorithm 2. We implemented it in C and reused the computed data structures in Python. To do so, we created a foreign function library that conveniently interfaces between C and other languages.

Algorithm 2 Constructing a compressed lookup table

Require: probability distribution given by values $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ and frequencies $\mathbf{f} = (f_1, f_2, \ldots, f_n) \in \mathbb{N}^n_{\geq 0}$
Ensure: compressed lookup table `compressedTable` of size $(r+1) \times 2^c$

$N \leftarrow \sum_{i=1}^{n} f_i$
$b \leftarrow \log_2(N)$
$e \leftarrow \max\{v \in [0, b] : \sum_{j=0}^{w} \sum_{i=1}^{n} f_i^{(j)} \cdot 2^{v-b-1} \leq 1 \quad \forall w \in [0, b]\}$
$c \leftarrow b - r$
$\mathbf{d} = \texttt{distribute}(\mathbf{f}, r, c)$
`compressedTable` $= [\,]$
for $i = 1$ to $r{+}1$ **do**
    for $j = 1$ to n **do**
        for $k = 1$ to $\mathbf{d}_{ji}$ **do** `compressedTable`.append(v_j)
return `compressedTable`

Like the reference implementation of ALDR and FLDR Draper and Saad [2025], we used bit operations, compiler intrinsics and linearized arrays where possible to ensure fast computation. We extended the existing SOTA implementations to also work with 64-bit input values to make them comparable with our test distributions.

C Evaluation Setup

All measurements were taken on a standard laptop equipped with an Intel i7-1255U CPU and 16 GiB DDR4 memory running Ubuntu Linux. Modern CPUs provide hardware counters that monitor the current power and energy demand. On the x86_64 platform *Running Average and Power Limit* (RAPL) David et al. [2010] counters provide energy readings at a 1 ms resolution. RAPL is organized into different power domains, representing different parts of the system. For this work, we focus on the CPU domains *cores* and *package* (pkg). The latter includes the former and additionally other parts of the CPU socket, such as caches and the memory controller. There are several factors that make energy measurements noisy, apart from default hardware noise. They include background activities, battery charging, artificial noise against side-channel attacks for security reasons Lipp et al. [2021], etc. We limit these influences by disabling CPU security features, keeping the laptop charged, providing additional warm-up rounds, and measuring multiple iterations. Additionally, we set a constant CPU frequency and CPU core to get meaningful energy readings, as detailed in the Appendix, Section F.

All samplers are implemented in C, as it provides a lower execution overhead compared to, e.g., Python. This allows for proper assessment of the actual costs of the algorithm. To ensure comparability, we apply the same degree of fair but not over-engineered optimization across these methods. We avoid multi-threading or (auto) vectorized code and use identical compiler flags. All methods use the identical entropy source.

D Related Work

We review methods for sampling from arbitrary finite discrete distributions.

Knuth and Yao Knuth and Yao [1976] established the theoretical foundation for exact discrete sampling using discrete distribution generating (DDG) trees. Their seminal result shows that any optimal sampling algorithm requires between $H(\mathbf{p})$ and $H(\mathbf{p}) + 2$ bits per sample, where $H(\mathbf{p}) = \sum_i -p_i \log_2(p_i)$ is the Shannon entropy. While entropy-optimal, DDG trees typically require exponential memory in the distribution precision. Lumbroso [2013] overcame this limitation for uniform and Bernoulli distributions with a linear-memory implementation, but the approach does not generalize to arbitrary distributions. The generic interval algorithm Hao and Hoshi [2006] achieves linear memory usage while consuming at most $H(\mathbf{p}) + 3$ bits per sample. However, implementations require expensive binary searches at each sampling step, limiting practical efficiency Devroye and Gravel [2020], Uyematsu and Li [2003]. Saad et al. [2020] present the FLDR algorithm that combines entropy-optimal sampling with rejection sampling, achieving an uppper bound of $H(\mathbf{p}) + 6$ bits per sample. Draper and Saad [2025] improved this to $H(\mathbf{p}) + 2$ bits with faster sampling speed for the ALDR algorithm, though at increased memory cost. Building on Marsaglia [1963], Marsaglia et al. [2004] proposed compressed lookup tables for discrete sampling. However, their compression scheme requires conditional branching and searches across multiple tables during sampling, reducing efficiency. In contrast, our approach uses a single compressed table with direct indexing, eliminating conditional overhead. The Alias method Walker [1974] preprocesses distributions into probability and alias arrays, enabling sampling via one uniform random variable and one coin flip (see Schwarz [2011] for an in-depth explanation). While being fast, it produces approximate samples and lacks controllable error bounds. The Index method Chen and Asau [1974] uses preprocessed index tables to guide inversion-based approximate sampling, but still requires search operations. Devroye [1986] provides an excellent overview and detailed analysis of the mathematics of random sampling, and presents many approximate samplers building on the Real RAM model Shamos [1978].

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made match the results of the evaluation section and are further explained in the appendix.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes, limitations (continous distributions, performance on small distributions compared to competitors, preprocessing overhead) are discussed both in the introduction and the evaluation.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: In the appendix, all theorems are proven.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Both the algorithm as well as the evaluation setup are detailed extensively in the appendix and code will be provided.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code and data will be provided.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: In the evaluation setup section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Standard deviations are provided and mentioned in the tables.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Detailed in the evaluation setup section in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: To the best of our knowledge, the code of ethics is not validated and anonymity is preserved.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The environmental effect of inefficient sampling algorithms is discussed in the introduction and the potential of reduction in energy consumption is clearly stated and further explained in the appendix.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: There is no risk, as the sampling yields the same results as previous exact samplers, just more efficient.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Authors are credited in code and manuscript.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Details communicated in the appendix.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Not applicable.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Not applicable.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: LLMs not involved.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.