Estimating the Entropy of Natural Scenes from Nearest Neighbors using CUDA

Paul Ivanov

REDWOOD CENTER for Theoretical Neuroscience

Vision Science Graduate Program
Redwood Center for Theoretical Neuroscience
University of California, Berkeley

Summary

The poster provides an overview of nearest neighbor search for entropy estimation of natural scenes. We report a 53 fold speed increase between C and CUDA implementations of high dimensional nearest neighbor search, and discuss the advantages of using CUDA from Python with PyCUDA.

Motivation

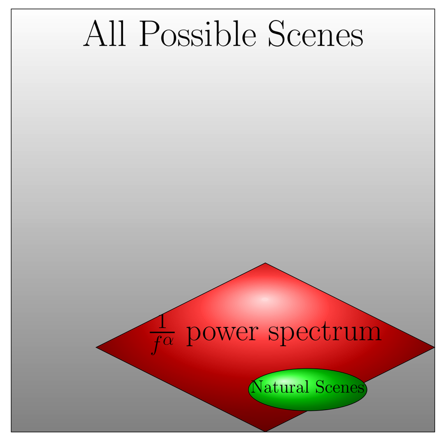

Figure 1: How large is the space of natural scenes?

Characterizing the statistics of natural scenes is an important area of vision research. For example, the entropy of images provides a measure of the information content available to the visual system and as such quantifies the demands placed on neural information processing mechanisms. From an applications perspective, entropy is the theoretical limit of compression – the lower bound on *any* compression scheme. Recently, Chandler and Field [2007] used an entropy estimation algorithm to binlessly estimate the entropy of small patches of natural images from the distribution of nearest-neighbor (NN) distances.

The approach described by Chandler and Field [2007] is limited by requiring NN calculations of an exponentially growing set. We overcome this limitation by porting the parallel brute force NN search to the GPU. This enables us to perform more extensive entropy and fractal dimensionality analyses on the van Hateren image database [van Hateren and van der Schaaf, 1998].

Review

Entropy is defined as : $H(\mathbf{X}) = -\sum_{x \in \mathbf{X}} p(x) \log_2 p(x)$

Entropy provides:

- a measure of information
- the uncertainty of a random variable
- the number of bits needed to describe a random variable
- lower bound on the number of bits needed for compression

Another useful interpretation: $H(\mathbf{X}) = \mathbf{E}\left[-\log_2 p(x)\right]$

For data with a small number of discrete states, entropy can be estimated by binning the data samples and using this empirical distribution as the joint probability $p(x)$. However, the binning approach quickly becomes computationally intractable for high dimensional data. Consider the case of image patches of pixels with 256 gray levels. Estimating $p(x)$ requires :

$$
\begin{aligned}
(256)^1 &= 256 && \text{bins for } 1 \times 1 \text{ pixel patches}\\
(256)^4 &= 4.295 \times 10^9 && \text{bins for } 2 \times 2 \text{ pixel patches}\\
(256)^9 &= 4.722 \times 10^{21} && \text{bins for } 3 \times 3 \text{ pixel patches}\\
(256)^{16} &= 3.403 \times 10^{38} && \text{bins for } 4 \times 4 \text{ pixel patches}\\
(256)^{25} &= 1.607 \times 10^{60} && \text{bins for } 5 \times 5 \text{ pixel patches}\\
(256)^{36} &= 4.973 \times 10^{86} && \text{bins for } 6 \times 6 \text{ pixel patches}
\end{aligned}
$$

Even for the modest case of 2×2 patches, the number of bins alone is onerous, yet the data necessary to obtain a reasonable estimate of the joint probability is even larger. The number of bins required for 6×6 image patches exceeds the number of atoms in the observable universe (10^{81}).

Method

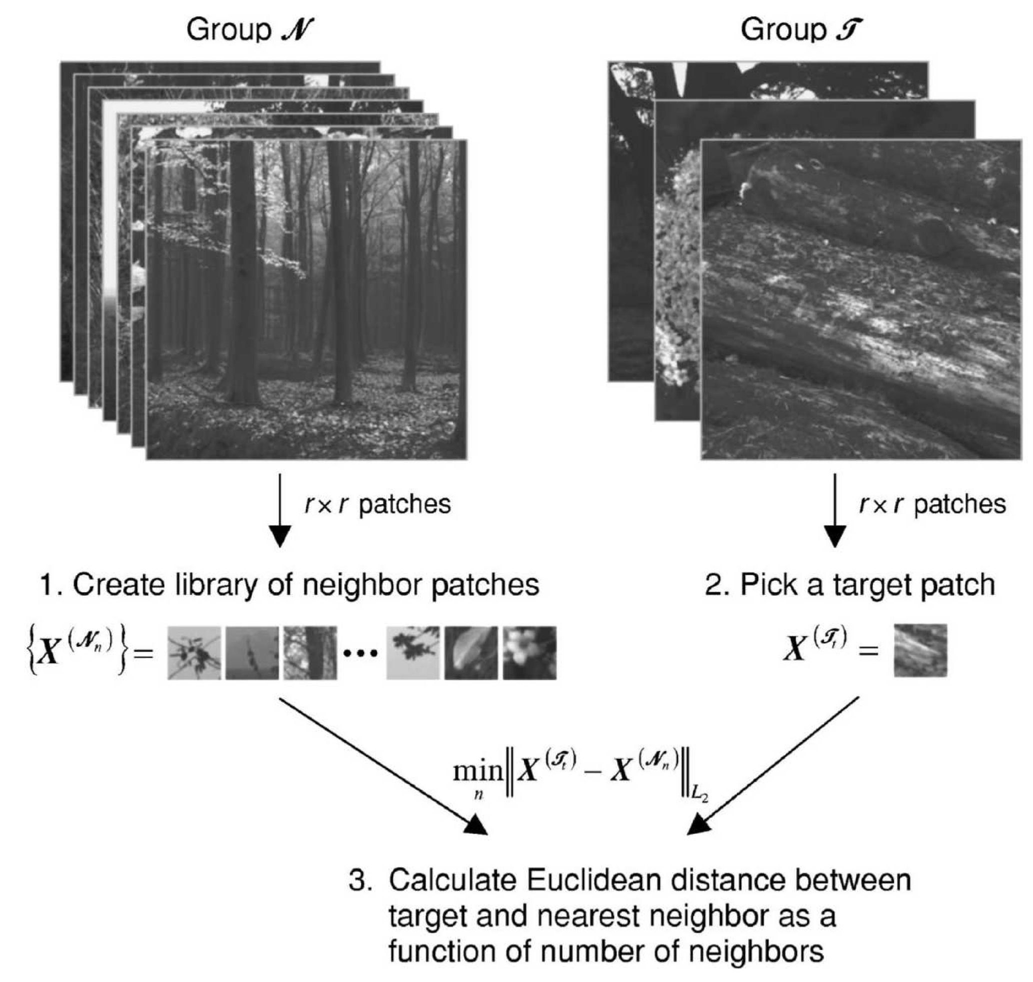

Figure 2: Diagram of the method (Figure 1 in Chandler and Field [2007]).

It has been shown that the average log NN distance ($\mathbf{E}[\log_2 D^*]$) can be used to estimate $\mathbf{E}[-\log_2 p(x)]$ without estimating $p(x)$ [Kozachenko and Leonenko, 1987]. This *binless* approach has been proven to be consistent and asymptotically unbiased.

Proximity Distribution (PD) : $\mathbf{E}[\log_2 D^*]$ as a function of the size of an exponentially growing set of neighbors. [Chandler and Field, 2007]

Relative Dimensionality (RD): negative reciprocal of the slope (first derivative) of the PD [Chandler and Field, 2007]. The dimensionality data appear to lie in for a given number of samples. Converges to intrinsic ("fractal") dimension.

Verification

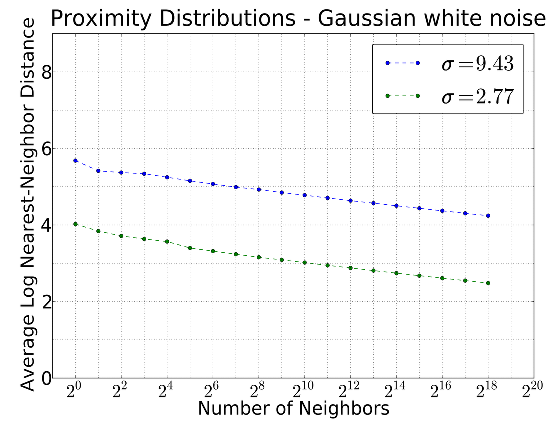

Figure 3: Proximity Distributions .

Method verified on data with known entropy.

For Gaussian white noise, $H(x) = \frac{1}{2}(\log_2 \pi e \sigma^2)$ bits.

From the Proximity Distribution curves (Figure 3), we can estimate the entropy (Figure 4a) and dimensionality (Figure 4b) of our data set.

When the relative dimensionality curve converges on the intrinsic dimensionality of the data ($k = 16$ for 4×4 patches), we have sampled enough data to accurately estimate the entropy.

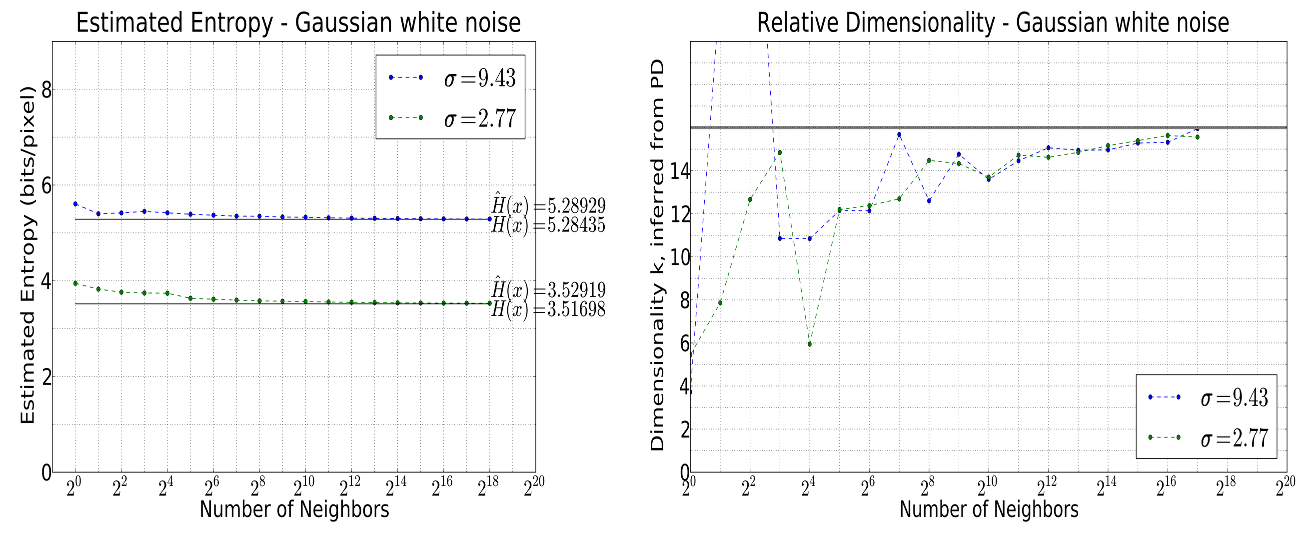

Figure 4: Estimated entropy (a) converges to the analytic result as the relative dimensionality (b) approaches the intrinsic dimension (gray line).

Estimated Entropy

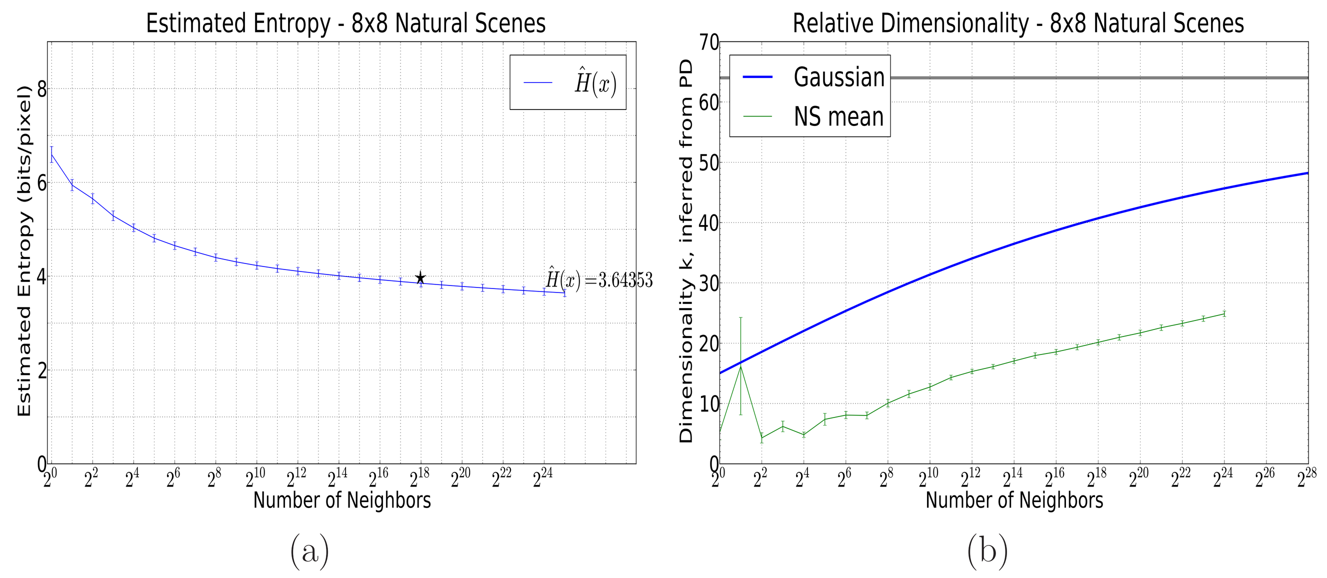

Figure 5: Estimated entropy for 8×8 image patches extracted from natural scenes.

⋆ Note: 2^{18} is the largest number of neighbors used in previous work [Chandler and Field, 2007]. We compute to 2^{25} neighbors, as there are fewer than 2^{26} non-overlapping 8×8 patches in the van Hateren database.

The RD (Figure 5a) has not converged to the intrinsic dimension ($k = 64$ for 8×8 patches, we do not have enough data to confidently estimate the entropy. As in Chandler and Field [2007], we fit a curve to our RD data (Figure 6b).and using this curve extrapolate our entropy estimate to 2^{300} samples (Figure 6a).

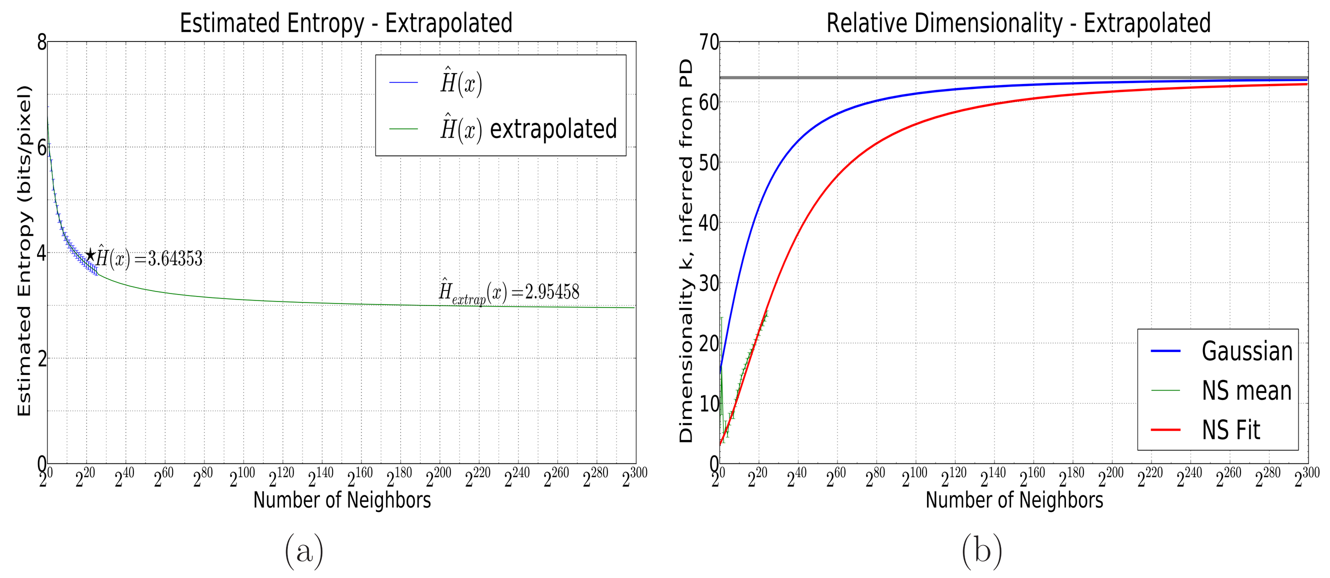

Figure 6: Extrapolated entropy estimate for 8×8 image patches from natural scenes.

GPU speedup

- Calculation for 2^{18} neighbors takes ≈ 3 hours on CPU
- We achieve the same in 6 *minutes* on a 8800GTX
  - (4 minutes on GTX 295 with no code changes)
- One 2^{25} neighbor run would take 16 days on CPU
- Same was done in 12 hours 48 minutes on 8800GTX
  - (7 hours 40 minutes on GTX 295 with no code changes)

C implementation ran on 2.4 GHz Intel Core2 Quad CPU (Q6600) (using one core).
All comparisons use 4096 target patches. Each patch is 64 dimensional (8×8).

N	pyCUDA (8800GTX)	pyCUDA (GTX 295)	C (gcc -O)	speedup (8800 GTX)	speedup (GTX 295)
4096	0.144 s	0.089 s	3.76 s	25.95	42.25
8192	0.270 s	0.159 s	7.52 s	27.80	47.30
16384	0.521 s	0.299 s	15.03 s	28.83	50.27
32768	1.029 s	0.583 s	30.04 s	29.17	51.53
65536	2.047 s	1.146 s	60.16 s	29.39	52.50
131072	4.025 s	2.276 s	120.83 s	30.02	53.09
262144	8.036 s	4.508 s	242.13 s	30.13	53.79
524288	16.064 s	9.003 s	484.50 s	30.16	53.81
1048576	32.093 s	17.989 s	969.00 s	30.19	53.87

Table 1: Speed Comparison Chart.

thanks, PyCUDA!

When I first started with CUDA, I was slowed down by the overhead of keeping track of different versions of kernel code and Makefiles and found it difficult to traverse the parameter space of my kernels.

PyCUDA [Kloeckner, 2009] let me concentrate on writing compute kernels, instead of keeping track of makefiles, with code generation and compilation conveniently abstracted away.

Workflow for porting to the GPU:

- write a *trusted* implementation (python)
- write a test suite that probes the input space and compares results of trusted and GPU implementation:random input, different dimensions, corner cases (nose)
- write *a* compute kernel (GPU implementation) that *passes* test suite
- optimize parameters and feel secure when test suite passes (pycuda)
- if performance satisfactory - done.
- else - think of a different organization for memory usage and write a new kernel

"Evolution" of my kernels (on 8800GTX):

- shared memory (load and reduction) - 8x total speedup
- more efficient reduction - 15x total speedup
- texture instead of load from global memory - 25x total speedup
- two textures interleaved - 30x total speedup (53x on GTX 295)

Contributions

- Implemented brute force NN search using PyCUDA.
  - 53x faster than C
- Estimated entropy and dimensionality of 8×8 patches using the *entire* van Hateren database.
- Achieved 2.95 bits per pixel estimate for natural scenes.
  - slightly higher than extrapolated estimate in Chandler and Field [2007].

References

Damon M Chandler and David J Field. Estimates of the information content and dimensionality of natural scenes from proximity distributions. *J Opt Soc Am A Opt Image Sci Vis*, 24(4): 922941, April 2007.

Andreas Kloeckner. PyCUDA, 2009. URL `http://mathema.tician.de/software/pycuda`.

L.F. Kozachenko and N.N. Leonenko. On statistical estimation of entropy of a random vector. *Probl. Inform. Transm.*, 23:9–16, 1987.

J H van Hateren and A van der Schaaf. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings of the Royal Society B: Biological Sciences*, 265(1394):359366, March 1998.

Acknowledgments

Thanks to Bruno Olshausen for providing guidance and encouragement. This work was supported by NEI Grant T32 EY007043 and NSF Grant IIS-0705939.

Contact information: Paul Ivanov, Redwood Center for Theoretical Neuroscience, University of California, Berkeley – Email: **pivanov@berkeley.edu** Web: *http://redwood.berkeley.edu/*