Video Content Analysis at Google, Inc.

Video content analysis is the basis for categorizing videos and enabling search by content. Recently there has been a growing interest in using sparse-coding methods to extract motion features in video in support of video content analysis. Several approaches rely on solving an L_1-regularized least-squares optimization problem both to learn and then subsequently to apply the basis vectors which are used to extract these features. The number and size of the basis vectors required for video are substantially larger than the corresponding basis vectors learned from static images. This is due to the wide diversity of visual transformations characteristic of video. In this application, Jacket and GPUs were used to improve performance by substantially accelerating the solution of the L_1-regularized least-squares optimization problem.

The authors re-implemented a version of the coordinate-descent algorithm of Friedman *et al* [2007] developed by Raina *et al* [2009]. Raina et al implemented their version of coordinate descent in C and in CUDA. The authors took the C code and rewrote it as vectorized MATLAB. The vectorized code clearly mirrors the linear-algebra formulation of the algorithm. The transition to MATLAB enabled the use of the Jacket platform for the prototyping and mapping of the algorithm onto GPU resources. The Raina *et al* CUDA code exploited potential structure in the original coordinate-descent algorithm, but serially coded each input vector. The vectorized code running under Jacket exploited both opportunities for parallelism. Moreover, the authors were able to experiment with exploiting additional opportunities for parallelism by making simple changes to the vectorized code, changes that would require a major rewrite of the CUDA code. Ease of use coupled with quite remarkable improvements in performance made the Jacket system extremely attractive.

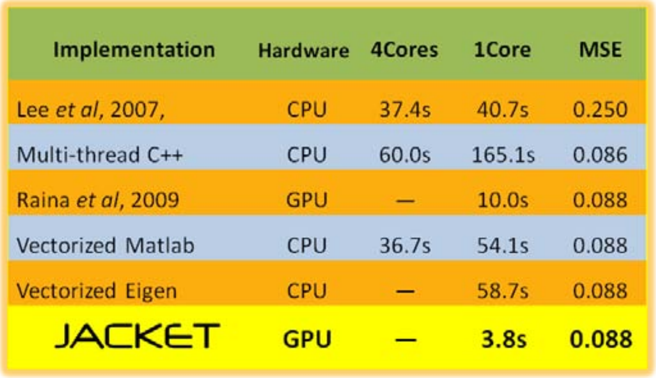
> **"Ease of use coupled with quite remarkable improvements in performance made the Jacket system extremely attractive."**

The research compared the results from six implementations each based on one of two algorithms:

I. Feature-sign algorithm [Lee *et al*, 2007]:
1. Implementation in MATLAB

II. Coordinate-descent algorithm [Friedman *et al*, 2007]:
2. Multi-threaded implementation in C++
3. CUDA implementation by [Raina et al, 2009]
4. Vectorized re-implementation of [2] in MATLAB
5. Vectorized re-implementation of [2] in Eigen/C
6. Same code as [4] but running under Jacket

The flexibility of Jacket made it easy to experiment with how best to use the GPU. As a result, it is not surprising that Jacket and GPUs outperformed the other implementations.



Implementation	Hardware	4Cores	1Core	MSE
Lee *et al*, 2007,	CPU	37.4s	40.7s	0.250
Multi-thread C++	CPU	60.0s	165.1s	0.086
Raina *et al*, 2009	GPU	—	10.0s	0.088
Vectorized Matlab	CPU	36.7s	54.1s	0.088
Vectorized Eigen	CPU	—	58.7s	0.088
JACKET	**GPU**	**—**	**3.8s**	**0.088**

Note: All runs were performed using 864 basis vectors, each vector with a spatial extent of 13 × 13 and a temporal extent of 7 frames, and 1024 input vectors. The reported times are averaged over multiple runs. All implementations used a sample of 1024 input vectors extracted from the first thirty seconds of a video. This is the basis for computing descriptors used to categorize videos in our experimental video categorization research.

Authors: Thomas Dean (Google), Rich Washington (Google) and Greg Corrado (Stanford)
Speedup: 10- to 20-times versus multi-core CPU implementations
System: MATLAB R2009a, an NVIDIA GeForce GTX 280, in a Dell Precision workstation with Two Intel 2.40 GHz Core 2 Quad Q6600

Jacket is a software platform specifically designed for engineers, scientists, and analysts who need maximum application performance, with minimal programming difficulty, while leveraging all technical computing resources available, including laptops, desktops, servers, clusters, and the Cloud. The Jacket platform consists of a runtime and language processing system that automatically optimizes existing applications or new algorithms for GPU computing.

---

[1] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani, "Pathwise coordinate optimization," Annals of Applied Statistics, vol. 1, no. 2, pp. 302–332, 2007.

[2] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in Advances in Neural Information Processing Systems 19, B. Schölkopf, J. Platt, and T. Hofmann, Eds. Cambridge, MA: MIT Press, 2007, pp. 801–808.

[3] R. Raina, A. Madhavan, and A. Ng, "Large-scale deep unsupervised learning using graphics processors," in Proceedings of the 25th Annual International Conference on Machine Learning, 2009.