Convolution is a Database Problem!

Paul G. Brown
Paradigm4 Inc
281 Winter Street Suite 360
Waltham MA 02451 USA
pbrown@paradigm4.com

ABSTRACT

Among the most exciting developments in contemporary software are systems for the processing and analysis of very large image collections [16]. In this talk, we will propose that integrating these methods with scalable and reliable data management is an important challenge in DBMS research and development. We provide a brief introduction to some of these algorithms and review two inter-related problem areas; image processing, and deep learning. Then we move on to explain the role data management has to play in bringing these techniques out of narrow and relatively specialized domains. Our investigation of these use cases further motivates the inclusion of matrix operations within a scalable DBMS platform.

1. INTRODUCTION

Our work with SciDB [15] has found us focussing on problem domains where image analysis techniques play a critical role. For example, our collaboration with NASA earth scientists has involved using RADAR data to understand changes in extreme weather events [7], INPE are using SciDB to manage remote sensing data in order to better understand in tropical rainforest environments and global climate change [6] [5], and a team at Lincoln Labs using microscopy data to investigate the neurological function of mammal brains. What all of these applications share is that they rely on the application of sophisticated image processing algorithms to very large data sets to achieve their results.

Traditionally, researchers have used tools like MATLAB [12] or an open-source alternative like Octave [8] to implement the analytic portions of their workload and have relied upon file systems for data management. In each of the applications introduced above, however, the SciDB Array DBMS is being used to manage very large (multi-Terabyte) data sets. The decision to opt for a DBMS rather than a file system was motivated by several factors.

First, before any image analysis can be done, research teams are typically obliged to do considerable work on data integration, curation and quality control; all tasks made easier because of SciDB's support for high level, declarative queries instead of low level file APIs. Second, SciDB's shared nothing architecture and ACID transactions make it possible for multiple users to share access to very large data sets without duplicating it. Third, SciDB's support for built in linear algebra operations–parallelized dense matrix multiply, for example–make the platform capable of efficiently performing large scale image analysis computations. And finally, SciDB's extensibility has made it possible to integrate user-defined operators that perform specialized tasks like connected component labeling.

Very recently, researchers have made tremendous strides in the field of image analysis with the application of so-called "deep learning" techniques. These neural network approaches rely for their efficiency on cleverly orchestrated linear algebra operations. So far, these techniques have been applied to problems of image classification: given a large corpus of photographs of people, animals, indoor and outdoor scenes, classify each image according to its content, or label the faces in the images. Several experimental systems such as Caffe [9], Theano [2] [3] and cuda-convnet [11] implement frameworks for experimenting with these approaches.

In this presentation we will argue that taking these image processing techniques out of their current application context and applying them to scientific data sets will require integrating them with data management facilities. Moving data to code is always a bad idea, and modern DBMS platforms can be specialized to such tasks. In other words, we will argue that "Convolution is a Database Problem!"

We begin with a brief review of convolutions and the image processing tasks that they are used for; smoothing and filtering using image kernel convolutions. Then we briefly describe convolution neural nets, and explain how they are implemented in Caffe. Then we move on to identify the key features we propose adding to the DBMS layer: convolution operations, efficient matrix multiply, transforms and automatic optimization.

2. CONVOLUTION BASICS

In image processing a convolution is a procedure that takes two inputs–usually a smaller one called the *kernel* and a larger one we will refer to as the *data*, both of the same dimensionality–and produces as output a result that is (usually) the same size as the *data* input. The naive algorithm addresses itself to each cell in the *data*, finds a surrounding sub-region that is the same size as the *kernel*, and performs the following operation to compute a single output value for each cell in the *data* input. (We illustrate one step in the algorithm where the *kernel* is a 3 x 3 matrix.)

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{pmatrix} (1*i)+(2*h)+(3*g)+ \\ (4*f)+(5*e)+(6*d)+ \\ (7*c)+(8*b)+(9*a) \end{pmatrix}$$

In mathematical terms, a *convolution product* C[u] (which is usually annotated using the '$*$' symbol) of two functions

f and g is defined as:

$$C[u] = (f * g)(u) = \int_{-\infty}^{\infty} f(\gamma) g(u - \gamma) d\gamma$$

Intuitively, a mathematical convolution is an integral that expresses the amount of overlap of the g function as it is "shifted" over the other function f. Mathematical convolutions have applications in many fields of science and engineering, but in this talk we will focus on how image transformation algorithms are expressed as a kind of matrix operation which is loosely related to mathematical convolutions. In such applications, we can use the following definition for convolution:

$$C[n] = (f * g)(n) = \sum_{i=-\infty}^{\infty} f(i) g(n - i)$$

Which explains the 3x3 convolution example at the start of this section. An important consideration in convolution methods is what to do at the edges where there is insufficient data in the *data* input to supply a value for each entry in the *kernel*. Typically the output is slightly smaller than the input *data* array as cells on the edge are trimmed because the results they contain are not usable.

It's important to note that once the *kernel* matrix gets sufficiently large, more sophisticated algorithms are more efficient [13] [4]. Rather than naively computing a potentially very large number of moderately sized inner products, it is more efficient to compute the result as follows. If we say that $F\{f\}$ denotes a Fourier Transform of f, then the convolution theorem states that:

$$F\{f * g\} = F\{f\} * F\{g\}$$

From which we can derive the following.

$$(f * g) = F^{-1}\{F\{f\}.F\{g\}\}$$

The naive convolution algorithm has quadratic computational complexity. With the help of the convolution theorem and the Fast Fourier Transform algorithm, the complexity of the convolution can be reduced to $O(n.log(n))$. Parallelizing either of these algorithms is a relatively minor challenge–individual cells from the *data* must be used to compute a number of values in the output which means *data* cell values must be replicated in multiple partitions of the overall workload–but CPU optimization is more difficult–the low ratio between the arithmetic operations and memory accesses makes convolution a *memory bound* problem.

3. IMAGE PROCESSING

Convolution methods are used to perform a variety of image processing tasks, such as edge detection, smoothing and blurring. In many scientific applications the data comes from scans that are not collected under well controlled conditions, or are subject to inevitable distortion and error. Also, in many situations he data is being brought together from multiple sources. So before it can be analyzed, it must be aligned, calibrated and filtered.

Gaussian smoothing, for example, is a basic tool in image analysis, particularly when the data is noisy. Gaussian smoothing can be accomplished by using the following image convolution kernel.

$$convolve(image, \frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix})$$

In the context of scientific data sets we are often dealing with single logical objects that are very large because they have been created by combining a large number of individual scans. For example, the image processing team at INPE are performing their analysis using a unified 7TiB, spatio-temporal data set that consists of many years of MODIS satellite data. Moving data out of SciDB and into custom written application code would result in extremely poor performance.

It makes sense for the data layer to provide features allowing users to quickly implement their own filters. Doing so means extending the core DBMS with an operation that parallelizes the application of a matrix kernel over an input data array. We would also prefer it if the DBMS was able to pick the most efficient approach to computing the result choosing (for example) between moving all of a (small) *data* array to a single node, distributing *data* and replicating the *kernel*, or opting for the Fourier Transform method when the *kernel* was large.

4. CONVOLUTION NEURAL NETS

Convolution Neural Nets (CNNs) are a recently popular "deep learning" method that relies heavily on convolution techniques. The idea is motivated by the biological observation that the visual cortex is composed of a "grid" of photo-sensitive cells, each of which is sensitive to a small sub-region of the overall visual field. Each cell filters the signal from its own sub-region. Similarly, CNNs exploit the spatial co-locality in the image by tiling it into kernel sized sub-regions, and computing a convolution on each tile. In contrast with the image processing use of convolution described above, CNNs do not move the kernel across the data one step at a time, but rather in strides, where each stride reflects the size of the tile.

Experimental systems like Caffe compute convolutions by reducing the problem to dense matrix-matrix multiplication [1]. The approach taken is to first *lower* the large, multi-dimensional data input into a dense 2D matrix that arranges cells in each tile along a row. Then the multiple layers of the CNN are arranged so that the values in each local unit are arranged in columns of a second matrix. At this point, the application of the neural net to the data can be accomplished using a BLAS GEMM operation, something that the DBMS can parallelize very easily. Then the result of the GEMM is then *lifted* back into its output shape.

Convolution Neural Nets are today being used in a wide variety of analytic applications: image classification, document recognition, natural language processing, facial and handwriting recognition, and even drug discovery. To support scientific data analysis we would like to add to the DBMS support for CNN training–stochastic gradient descent–and classification–efficient, scalable application of tiled convolution. Again, query optimization would considerably enhance the DBMS's utility.

5. DATA MANAGEMENT CHALLENGES

The convolution methods we have described here all rely on an underlying collection of linear algebra operations: matrix transformation, dense matrix multiply, the Fast Fourier Transform, convolution, etc. These are computational building blocks that are not available in business-centric DBMSs that rely on SQL, or currently fashionable NoSQL DBMSs. For the most part, developers using convolution methods rely on custom code for their application and a distributed file-system for storage. To judge from the benchmarks and competitions that are used to assess the efficacy and performance of these tools [14] [10], the emphasis is on grouping or classifying individual images from a (large) file based collection. Consequently, most of the effort being directed at performance and scalability has focussed on the problem of parallelizing a large number of smaller operations [16].

Data found in scientific data analysis is a composite of a large number of individual images. In scientific data analysis we're often focused on a relatively small number of much larger data objects. For example, SciDB is being used to build logically single data objects–a microscopy scan of a rat's brain, for example–at multi-Terabyte scale. Scientific users will specify the data they wish to subject to processing or analysis based on meta-data stored in the same database as the images. For example, someone working with remote sensing data may want to try a new convolution based filter on data for a particular geographic region. Or a neurologist may with to apply a CNN to compare counts of cancerous cells in scans taken from a control and an experimental group of rats.

From the perspective of a DBMS engine, we would also like to liberate users from the need to know precisely *how* their processing is being performed. In our section introducing convolution operations, we pointed out that there are (at least) two approaches, with different trade-offs in performance: for small kernels, naive computation is often good enough, while for larger kernels Fourier methods are preferred. Interestingly, in addition to performance, there are trade-offs in numerical precision between these two approaches: an additional challenge for the DBMS optimizer.

6. SUMMARY AND CONCLUSIONS

In this paper we have reviewed a number of recently and not so recently developed techniques for image processing and analysis, and have described their use in the context of scientific data management applications. We have explained why these methods are not well adapted for existing DBMS platforms: leading to a situation where almost all of the work being done involves custom code applied to large collections of files in a file-system. We have explained our case for extending an array DBMS like SciDB to support a collection of simple, granular operations that can be used to implement these techniques, and explained the advantages that would accrue to end users by doing so.

7. REFERENCES

[1] F. Abuzaid, S. Hadjis, C. Zhang, and C. Ré. Caffe con troll: Shallow ideas to speed up deep learning. *CoRR*, abs/1504.04343, 2015.

[2] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[3] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[4] R. Bracewell. *The Fourier Transform and Its Applications*. Electrical engineering series. McGraw Hill, 2000.

[5] G. Camara. e-sensing: Big earth observation data analytics for land use and land cover change information. Technical report, Instituto Nacional de Pesquisas Espaciais, 2015.

[6] G. Camara. Geoinformatics and environmental modelling for supporting global land use change research. Technical report, University of Munster, 2015.

[7] K. Doan, A. Oloso, K.-S. Kuo, and T. L. Clune. Performance comparison of big-data technologies in locating intersections in satellite ground tracks. In *ASE BigData/SocialInformatics/PASSAT/BioMedCom Conference*, December 2014.

[8] J. W. Eaton, D. Bateman, S. Hauberg, and R. Wehbring. *GNU Octave Version 4.0 User's Guide*. Free Software Foundation, Boston, Massachusetts, 2015.

[9] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *CoRR*, abs/1408.5093, 2014.

[10] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[12] MATLAB. *MATLAB Users Guide Version R2016b*. The MathWorks Inc., Natick, Massachusetts, 2015.

[13] K. Pavel and S. David. Algorithms for efficient computation of convolution. In D. G. Ruiz, editor, *Design and Architectures for Digital Signal Processing*. InTech, 2013.

[14] L. e. a. Roux. Mitosis detection in breast cancer histological images an icpr 2012 contest. *PMC. Web*, 2013.

[15] M. Stonebraker, J. Becla, D. J. DeWitt, K. T. Lim, D. Maier, O. Ratzesberger, and S. B. Zdonik. Requirements for science data bases and scidb. In *CIDR 2009, Fourth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2009, Online Proceedings*, 2009.

[16] R. Wu, S. Yan, Y. Shan, Q. Dang, and G. Sun. Deep image: Scaling up image recognition. *CoRR*, abs/1501.02876, 2015.