Accelerating Collapsed Variational Bayesian Inference for Latent Dirichlet Allocation with Nvidia CUDA Compatible Devices

Tomonari Masada, Tsuyoshi Hamada, Yuichiro Shibata, and Kiyoshi Oguri

Nagasaki University
Bunkyo-machi 1-14, Nagasaki, Japan
{masada,hamada,shibata,oguri}@cis.nagasaki-u.ac.jp

Abstract. In this paper, we propose an acceleration of collapsed variational Bayesian (CVB) inference for latent Dirichlet allocation (LDA) by using Nvidia CUDA compatible devices. While LDA is an efficient Bayesian multi-topic document model, it requires complicated computations for parameter estimation in comparison with other simpler document models, e.g. probabilistic latent semantic indexing, etc. Therefore, we accelerate CVB inference, an efficient deterministic inference method for LDA, with Nvidia CUDA. In the evaluation experiments, we used a set of 50,000 documents and a set of 10,000 images. We could obtain inference results comparable to sequential CVB inference.

1 Introduction

In this paper, we present an application of general-purpose GPU to parameter inference for probabilistic document models. We accelerate an inference method, called *collapsed variational Bayesian (CVB) inference* [12], for a well-known Bayesian multi-topic document model, *latent Dirichlet allocation (LDA)* [3], by using Nvidia *compute unified device architecture (CUDA)* [1] compatible devices. First of all, we summerize our two main contributions. Our research is the first attempt to parallelize *CVB inference*, and also the first attempt to parallelize inference for LDA *by using GPU*. The rest of the paper is organized as follows. Section 2 includes the background of our research. Section 3 presents the details of parallelized CVB inference for LDA. Section 4 shows how to implement parallelized CVB inference by using Nvidia CUDA compatible devices. Section 5 provides the results of our experiments. Section 6 concludes the paper.

2 Background

2.1 Latent Dirichlet allocation

LDA [3] is a *Bayesian multi-topic* document model. The term *Bayesian* refers to probabilistic models where, after introducing prior distributions, posterior distribution is estimated not by a specific set of parameter values, but as a distribution over all possible parameter values. The meaning of the term *multi-topic* can be explained as follows. In document modeling, topic is often identified with a multinomial distribution defined over words, because semantic differences are reflected in what kind of words are frequently used. In LDA, a set of words constituting one document are drawn from more than one multinomials. Namely, documents are modeled as a mixture of *multiple topics*.

While LDA is originally proposed as a probabilistic model of documents, we can find its applications in various research fields [13][14]. However, due to complicated model structure, LDA requires acceleration when applied to datasets of large size. We can apply Expectation-Maximization (EM) algorithm to simpler document models, e.g. Dirichlet compound multinomial [7] and probabilistic latent semantic indexing [6]. EM algorithm can be efficiently implemented in a parallelized manner [4]. In contrast, we cannot use EM for LDA. Therefore, we propose an acceleration customized for LDA by using general-purpose GPU.

2.2 Collapsed variational Bayesian inference

The following three inference methods are so far proposed for LDA: variational Bayesian (VB) inference [3], collapsed Gibbs sampling [5], and CVB inference. In this paper, we focus on CVB inference due to the following two reasons. First, both VB and CVB inferences use variational method to obtain a tractable posterior distribution. While variational method introduces approximation, CVB inference can achieve less approximation than VB for LDA [12]. Second, collapsed Gibbs sampling incorporates randomization. Therefore, we should carefully determine when to stop inference iteration. In contrast, CVB and VB inferences are deterministic methods. It is relatively easy to decide when to terminate iteration.

Among these three inference methods, VB inference and collapsed Gibbs sampling have already been parallelized by using PC clusters [8][11][10]. However, both methods divide a given dataset into smaller subsets and process the subsets in parallel. As far as intermediate results are appropriately broadcasted in the course of inference, we can adopt this *coarse-grained* parallelization only by introducing negligible approximations. Our research focuses on another parallelism appearing in parameter update formula of CVB inference. Our method is based on *fine-grained* parallelism, whose details are outlined below.

Parameters to be estimated in CVB inference for LDA are indexed by documents, words, and topics. Let $j = 1, \dots, J$, $w = 1, \dots, W$, and $k = 1, \dots, K$ be the indexes of documents, words, and topics, respectively. Let γ_{jwk} denote parameters to be estimated for all document/word pairs j, w and for all topics k. Intuitively speaking, γ_{jwk} means how strongly word w in document j relates to topic k. An outline of one iteration, i.e. one dataset scan, of CVB inference for LDA is shown in Fig. 1. When the number of unique document/word pairs is M, time complexity of each iteration is $O(MK)$.

Our fine-grained parallelism originates from the fact that parameters corresponding to different topics can be updated independently. Namely, all values required for updating parameters indexed by a specific k are also indexed by the same k. Therefore, we can conduct K update computations in parallel, where K

```
for each document $j$ in a given document set
    for each word $w$ appearing in document $j$
        for each topic $k$
            update $\gamma_{jwk}$
        next
        normalize $\gamma_{jw1}, \ldots, \gamma_{jwK}$ so that $\sum_{k=1}^{K} \gamma_{jwk} = 1$ holds
    next
next
```

Fig. 1. An outline of one iteration of CVB inference for LDA.

is the number of topics. This is the fine-grained parallelism used by our method. However, $\sum_k \gamma_{jwk} = 1$ should hold for each document/word pair j, w. This normalization can be realized by a standard reduction (See Fig. 2) of $O(\log K)$ time. Therefore, time complexity of each iteration is reduced to $O(M \log K)$. Further, our fine-grained parallelism is orthogonal to coarse-grained parallelism, and thus both types of parallelism can be combined.

Recently, a new acceleration method of collapsed Gibbs sampling for LDA appears [9]. This method reduces time complexity by an algorithmic elaboration. Interestingly, our method shares the same intuition, because both methods try to reduce $O(K)$ factor of time complexity for LDA inferences.

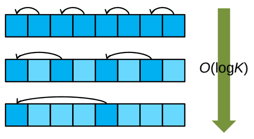

Fig. 2. A standard parallel reduction for parameter normalization

3 Details of Parallelized Inference

3.1 Task of CVB inference for LDA

We can describe LDA document model as a process of generating documents. First, we draw a topic multinomial distribution $Mul(\theta_j)$ for each document j from a symmetric Dirichlet prior distribution $Dir(\alpha)$. Second, we draw a word multinomial $Mul(\phi_k)$ for each topic k from another symmetric Dirichlet prior $Dir(\beta)$. Then, we can generate document j by repeating the following procedure as many times as the number of word tokens in document j: draw a topic from $Mul(\theta_j)$, and then draw a word from $Mul(\phi_k)$ when the drawn topic is k. This generative description leads to full joint distribution as follows:

$$p(\mathbf{x}, \mathbf{z}, \theta, \phi; \alpha, \beta) = \prod_k p(\phi_k; \alpha) \prod_j p(\theta_j; \beta) \cdot \prod_j \prod_i p(z_{ji}|\theta_j) p(x_{ji}|\phi_{z_{ij}}) \tag{1}$$

where x_{ji} is the random variable whose value is ith word token in document j, and z_{ji} is the random variable whose value is the topic used to draw ith word token in document j. In this paper, we say that topic k *is assigned to* a certain word token when the word token is drawn from a word multinomial $Mul(\phi_k)$.

From Eq. (1), we obtain posterior distribution $p(\mathbf{z}, \theta, \phi | \mathbf{x}; \alpha, \beta)$ as $p(\mathbf{x}, \mathbf{z}, \theta, \phi; \alpha, \beta) \,/ p(\mathbf{x}; \alpha, \beta)$. However, marginal likelihood $p(\mathbf{x}; \alpha, \beta)$ is intractable. Therefore, we approximate posterior by variational method. Let $q(\mathbf{z}, \theta, \phi)$ denote an approximated posterior. Then, log marginal likelihood can be lower bounded as:

$$\log p(\mathbf{x}; \alpha, \beta) \geq \int\int \sum_{\mathbf{z}} q(\mathbf{z}, \theta, \phi) \log \frac{p(\mathbf{x}, \mathbf{z}, \theta, \phi; \alpha, \beta)}{q(\mathbf{z}, \theta, \phi)} d\theta d\phi \tag{2}$$

In CVB inference, we assume that topic assignments z_{ji} are mutually independent and approximate posterior as:

$$q(\mathbf{z}, \theta, \phi) = p(\theta, \phi | \mathbf{z}, \mathbf{x}, \alpha, \beta) q(\mathbf{z}) = p(\theta, \phi | \mathbf{z}, \mathbf{x}, \alpha, \beta) \prod_j \prod_i q(z_{ji}; \gamma_{jx_{ji}}) \tag{3}$$

where $q(z_{ji}; \gamma_{jx_{ji}})$ is an approximated posterior probability of topic assignment to ith word in document j. A set of K parameters $\gamma_{jw} = \{\gamma_{jw1}, \ldots, \gamma_{jwK}\}$ for a fixed document/word pair j, w can be regarded as a topic multinomial distribution. γ_{jwk} means how strongly word w in document j relates to topic k. By using Eq. (3), the right hand side of Eq. (2) is reduced to $\sum_{\mathbf{z}} q(\mathbf{z}) \log \frac{p(\mathbf{x}, \mathbf{z}; \alpha, \beta)}{q(\mathbf{z})}$, where all model parameters θ, ϕ are marginalized out. This marginalization is denoted by the term *collapsed*. The task of CVB inference is to determine posterior parameters by maximizing $\sum_{\mathbf{z}} q(\mathbf{z}) \log \frac{p(\mathbf{x}, \mathbf{z}; \alpha, \beta)}{q(\mathbf{z})}$.

3.2 Parameter update formula

We use CVB inference accompanied with Gaussian approximation presented in [12]. In this paper, we only show resulting update formula. Three types of mean/variance pairs, defined below, are needed to update posterior parameters:

$$\begin{aligned}
M_{jk} &\equiv \sum_w n_{jw} \gamma_{jkw}, & V_{jk} &\equiv \sum_w n_{jw} \gamma_{jkw} (1 - \gamma_{jkw}) \\
M_{kw} &\equiv \sum_j n_{jw} \gamma_{jkw}, & V_{kw} &\equiv \sum_j n_{jw} \gamma_{jkw} (1 - \gamma_{jkw}) \\
M_k &\equiv \sum_{j,w} n_{jw} \gamma_{jkw}, & V_k &\equiv \sum_{j,w} n_{jw} \gamma_{jkw} (1 - \gamma_{jkw})
\end{aligned} \tag{4}$$

where n_{jw} is the number of tokens of word w in document j. By using these three types of mean/variance pairs, posterior parameters are updated as shown in Fig. 3, which is a detailed description of the innermost loop of Fig. 1. Fig. 3 shows that update computations for different ks can be executed in parallel. Only normalization in Step 3 requires $O(\log K)$ time based on a standard reduction technique in Fig. 2. Therefore, time complexity per iteration is $O(M \log K)$.

1. Subtract the contribution of γ_{jwk} from all three types of means and variances.

$$\begin{aligned} M_{jk} &\leftarrow M_{jk} - n_{jw}\gamma_{jwk}, & V_{jk} &\leftarrow V_{jk} - n_{jw}\gamma_{jwk}(1-\gamma_{jwk}) \\ M_{kw} &\leftarrow M_{kw} - n_{jw}\gamma_{jwk}, & V_{kw} &\leftarrow V_{kw} - n_{jw}\gamma_{jwk}(1-\gamma_{jwk}) \\ M_{k} &\leftarrow M_{k} - n_{jw}\gamma_{jwk}, & V_{k} &\leftarrow V_{k} - n_{jw}\gamma_{jwk}(1-\gamma_{jwk}) \end{aligned}$$

2. Update γ_{jwk}.

$$\begin{aligned} \gamma_{jwk} \leftarrow & (\alpha + M_{jk})(\beta + M_{kw})(W\beta + M_k)^{-1} \\ & \cdot \exp\{-\frac{V_{jk}}{2(\alpha+M_{jk})^2} - \frac{V_{kw}}{2(\beta+M_{kw})^2} + \frac{V_k}{2(W\beta+M_k)^2}\} \end{aligned}$$

3. Normalize $\gamma_{jw1}, \ldots, \gamma_{jwK}$ so that $\sum_k \gamma_{jwk} = 1$ holds.
4. Add the contribution of γ_{jwk} to all three types of means and variances.

$$\begin{aligned} M_{jk} &\leftarrow M_{jk} + n_{jw}\gamma_{jwk}, & V_{jk} &\leftarrow V_{jk} + n_{jw}\gamma_{jwk}(1-\gamma_{jwk}) \\ M_{kw} &\leftarrow M_{kw} + n_{jw}\gamma_{jwk}, & V_{kw} &\leftarrow V_{kw} + n_{jw}\gamma_{jwk}(1-\gamma_{jwk}) \\ M_{k} &\leftarrow M_{k} + n_{jw}\gamma_{jwk}, & V_{k} &\leftarrow V_{k} + n_{jw}\gamma_{jwk}(1-\gamma_{jwk}) \end{aligned}$$

Fig. 3. How to update parameters in CVB inference.

4 Implementation on CUDA

4.1 Nvidia CUDA compatible devices

Fig. 4 shows execution model of Nvidia CUDA compatible GPU devices. We only describe necessary details. The largest execution unit is called *grid*. In our case, one grid roughly corresponds to one iteration, i.e., one scan of dataset. However, when dataset is too large for graphics card RAM, called *device memory*, we divide dataset into subsets and process the subsets sequentially. In this case, one grid corresponds to processing of one data subset. Typical high-end graphics cards provide up to 1 GBytes device memory.

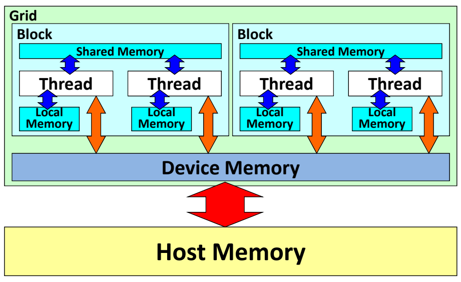

Fig. 4. Conceptual digram of Nvidia CUDA compatible devices.

Each grid contains hundreds of *blocks*, each of which corresponds to one document in our case. Therefore, we process as many documents as blocks in parallel. Each block includes hundreds of *threads*. Threads in the same block can communicate with each other by using a fast memory, called *shared memory*. Shared memory is as fast as *local memory* assigned to each thread. Only four clock cycles are required to both read from and write to shared memory or local memory. However, threads from different blocks should communicate via device memory, which requires hundreds of clock cycles to access. Further, shared memory size is only tens of KBytes per block in existing graphics cards. Therefore, efficient communication among threads can be realized only for small size data.

A typical usage of CUDA compatible devices is as follows: send data from host memory (i.e., CPU memory) to device memory, execute computations on GPU, and then write results back from device memory to host memory. Since data transfer between host memory and device memory requires quite a large number of clock cycles, threads should not access to host memory during computation. Therefore, we send data to device memory before launching a grid on GPU.

4.2 Implementation details of CVB inference for LDA

By considering features of CUDA described above, we implement a parallelized version of CVB inference for LDA. Different threads are responsible for computations relating to different topics. This is our fine-grained parallelization. We assume that K is less than the maximum number of threads per block. This assumption is admissible for most applications.

We have following variables in CVB inference: posterior parameters γ_{jwk} and three types of mean/variance pairs (M_{jk}, V_{jk}), (M_{kw}, V_{kw}), (M_k, V_k) as shown in Fig. 3. The number of posterior parameters is too large for shared memory. Therefore, we store the parameters on device memory. However, the same set of K parameters $\gamma_{jw1}, \ldots, \gamma_{jwK}$ are used for four consecutive steps in Fig. 3 with respect to a fixed document/word pair. This means that, before executing each sequence of these four steps, we can load the corresponding set of K parameters from device memory to shared memory. Normalization can also be efficiently conducted on shared memory.

We store the first type of mean/variance pairs (M_{jk}, V_{jk}) on local memory. There are three reasons. First, this type of pairs is not shared by threads from different blocks, because different blocks process different documents. Second, both local memory and shared memory are large enough to store all K mean/variance pairs (M_{j1}, V_{j1}), $\ldots$, (M_{jK}, V_{jK}) for a fixed j. Third, these pairs do not need to be shared among different threads even in the same block.

The second type of mean/variance pairs (M_{kw}, V_{kw}) should be shared by the threads processing the same word. However, the same word can appear in many different documents. Therefore, we store these mean/variance pairs on device memory. Further, multiple threads may access the same pair at the same moment when they occasionally process the same word simultaneously. Mutual exclusion is required here, because all mean/variance pairs are not only read, but also modified, as shown in Fig. 3. Preliminary experiments show that atomic functions prepared for CUDA largely increase execution time. Therefore, we do not implement mutual exclusion. This leads to an introduction of approximation, because non-atomic read and write operations from different threads are not ordered appropriately. Further, some of these operations may fail. Preliminary experiments show that this approximation is negligible as long as the number of blocks per grid is small enough. Based on our experiences, the number of blocks per grid should be at most 16. Therefore, at most $512 \times 16 = 8192$ threads run in parallel. With larger number of threads, inference will not proceed correctly.

The third type of mean/variance pairs (M_k, V_k) should also be shared by threads from different blocks. However, we store this type of mean/variance pairs on local memory and reduce access to device memory. As a result, different blocks use different values. However, this type of mean/variance pairs is only indexed by topics. Namely, these means and variances are obtained by summation over document/word pairs. We can expect that only negligible discrepancy will be observed among this type of mean/variance pairs localized to different blocks.

Only posterior parameters are written back from device memory to host memory after a launch of a grid. Three types of means and variances are computed based on Eq. (4) from scratch on CPU by using all posterior parameters. In many realistic applications, we will divide an input dataset into smaller subsets and process them on GPU sequentially, because device memory is not large enough. Therefore, a single launch of grid updates only a small part of posterior parameters. When we compute means and variances by using all posterior parameters including those which are not updated in the preceding launch of grid, approximations introduced into our parallelized inference will be reduced.

5 Experiments

5.1 Settings

In this paper, we evaluate inference quality by *test data perplexity*. We use one half of the tokens of each word in each document as training data for CVB inference. Another half is used as test data for perplexity evaluation. Test data perplexity is computed based on test data probability which is defined as $p(\mathbf{x}_{test}) = \prod_j \prod_w \left(\sum_k \frac{\alpha + M_{jk}}{K\alpha + \sum_k M_{jk}} \frac{\beta + M_{kw}}{W\beta + M_k} \right)^{n'_{jw}}$, where n'_{jw} is the number of test data tokens of word w in document j. Then, test data perplexity can be defined as $\exp\{-\log p(\mathbf{x}_{test}) / \sum_{j,w} n'_{jw}\}$. After a large enough number of iterations, test data perplexity reaches a minimum, which is not guaranteed to be a global minimum in CVB inference. We evaluate inference quality by test data perplexity after convergence. When our parallelized CVB inference gives test data perplexities comparable to sequential version, our parallelization is admissible.

All experiments are conducted on a PC equipped with Intel Core2 Quad CPU Q9550 at 2.83GHz and with 8 GBytes host memory. As an Nvidia CUDA compatible device, we used Leadtek WinFast GTX 260 with 896 MBytes device memory, where shared memory size per block is 16 KBytes, maximum number of threads per block is 512, and clock rate is 1.24 GHz. Based on preliminary experiments, we set the number of blocks in a grid to 16. With larger number of blocks per grid, inference will not proceed correctly due to too many simultaneous non-atomic accesses to device memory as described in Section 4.2. The number of threads per block is set to 512. Therefore, $512 \times 16 = 8192$ threads can run in parallel. With this setting, 20 GFLOPS was achieved for computations on GPU in our experiments. When K is far less than the number of threads per block, we can assign more than one documents to each block. For example, when $K = 64$, we can process $512/64 = 8$ documents in parallel per block.

5.2 Datasets

We used a document set and an image set for comparison experiments. The document set consists of 56,755 Japanese newswire articles from Mainichi and Asahi newspaper Web sites. Dates of articles range from November 16, 2007 to May 15, 2008. By using morphological analyzer *MeCab* [2], we obtain 5,053,978 unique document/word pairs. The number of word tokens is 7,494,242, among which 1,666,156 are used for perplexity evaluation. The number of words is 40,355 after removing rare words and stop words. For this dataset, we tested two settings for the numbers of topics, i.e., $K = 64$ and $K = 128$.

The image set is *10,000 test images* by Professor J.-Z. Wang [16] [15]. Since our aim is not to propose a new image processing method, we used a simple feature extraction. Regardless image size, we divide each image into $16 \times 16 = 256$ non-overlapping rectangle regions of the same width and height. Further, we uniformly quantize RGB intensities and reduce the number of colors from 256^3 to $8^3 = 512$. Then, we count the frequency of quantized colors in each of 256 rectangle regions. Consequently, all images are represented as a frequency distribution defined over $256 \times 512 = 131,072$ features. After removing features appearing too frequently, we obtain $113,181$ features, which is the number of words in LDA. Exact number of images in this dataset, i.e., the number of documents, is 9,908. The number of unique document/word pair is 17,127,235, and the number of tokens is 84,741,744, among which 40,941,092 tokens are used for perplexity evaluation. For this dataset, we set the number of topics to 64.

5.3 Results

We run 10 trials of both parallel and sequential CVB inferences for LDA starting from randomly initialized posterior parameters. We present test data perplexities for the first 64 iterations, because further iterations give no significant changes. Table 1 provides test data perplexity and execution time after 64 iterations. Further, Fig. 5 shows test data perplexity versus execution time.

Space complexity scales with K in CVB inference for LDA. Therefore, we need more frequent data transfers between host memory and device memory for larger K. Table 1 shows that we could only achieve $\times 4.6$ acceleration for $K = 128$ in comparison with $\times 7.3$ for $K = 64$ with respect to the same document set. However, CUDA could provide comparable perplexities for our document set. For Professor Wang's image set, we even achieved less perplexities,

Table 1. Test data perplexity and execution time after 64 iterations. Mean values for 10 trials are presented. We also show standard deviation for test data perplexity.

	document set, $K = 64$	document set, $K = 128$	image set, $K = 64$
without CUDA	1300.0 ± 10.7 (2h 06m 56s)	1001.0 ± 9.3 (4h 12m 49s)	5604.1 ± 30.3 (7h 10m 24s)
with CUDA	1295.2 ± 17.2 (17m 19s; $\times 7.3$)	1005.4 ± 8.7 (54m 25s; $\times 4.6$)	5015.4 ± 38.2 (2h 30m 50s; $\times 2.9$)

because parallelized inferences may occasionally find better descent directions. Similar arguments can be found in a previous paper proposing parallelized collapsed Gibbs sampling for LDA [10]. For this image set, we provide a complete result, where we use full dataset for inference, at the Web site of an author (`http://www.cis.nagasaki-u.ac.jp/~masada/LDAimage/page1.html`).

Fig. 5 intuitively shows that we could obtain comparable perplexities in much shorter time with CUDA. Based on these results, we can conlude that our parallelization is in fact a promising computational improvement.

6 Conclusions

In this paper, we propose an acceleration of collapsed variational Bayesian inference for latent Dirichlet allocation with Nvidia CUDA compatible GPU devices. We could obtain inference results of quality comparable to original sequential version in much shorter time only by introducing negligible approximations.

We are now implementing our method on a cluster of PCs equipped with graphics cards. In the near future, this combination of our fine-grained parallelization and exisiting coarse-grained parallelization will achieve further acceleration of collapsed variational Bayesian inference for latent Dirichlet allocation.

References

1. NVIDIA CUDA, `http://www.nvidia.com/cuda`
2. MeCab, `http://mecab.sourceforge.net/`
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet Allocation. In: Advances in Neural Information Processing Systems 14, pp. 601–608 (2001)
4. Chu, C.T., Kim, S.K., Lin, Y.A., Yu, Y., Bradski, G.R., Ng, A.Y., Olukotun, K.: Map-Reduce for Machine Learning on Multicore. In: Advances in Neural Information Processing Systems 19, pp. 306–313 (2006)
5. Steyvers, M., Smyth, P., Rosen-Zvi, M., Griffiths, T.L.: Probabilistic Author-Topic Models for Information Discovery. In: the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 306–315 (2004)
6. Hofmann, T.: Probabilistic Latent Semantic Indexing. In: the 22nd International Conference on Research and Development in Information Retrieval, pp. 50–57 (1999)
7. Madsen, R.E., Kauchak, D., Elkan, C.: Modeling Word Burstiness Using the Dirichlet Distribution. In: the 22nd International Conference on Machine learning, pp. 545–552 (2005)

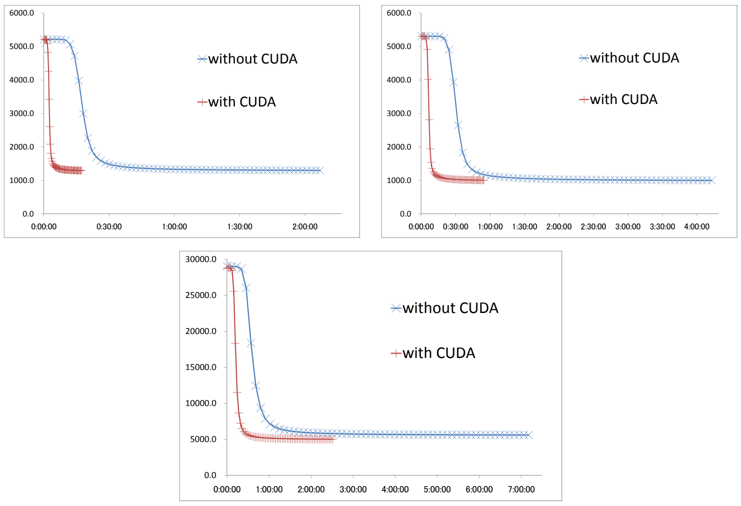

Fig. 5. Test data perplexity, averaged over 10 trials, versus execution time. Results for the newswire article set is presented in top row, where left and right panel correspond to $K = 64$ and $K = 128$, respectively. Results for the image dataset is given in bottom panel. CVB inferences with CUDA can achieve comparable (or even less) test perplexities in far shorter time for all settings.

8. Nallapati, R., Cohen, W.W., Lafferty, J.D.: Parallelized Variational EM for Latent Dirichlet Allocation: An Experimental Evaluation of Speed and Scalability. In: ICDM Workshop on High Performance Data Mining, pp. 349–354 (2007)
9. Porteous, I., Newman, D., Ihler, A.T., Asuncion, A., Smyth, P., Welling, M.: Fast Collapsed Gibbs Sampling for Latent Dirichlet Allocation. In: the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 569–577 (2008)
10. Newman, D., Ascuncion, A., Smyth, P., Welling, M.: Distributed Inference for Latent Dirichlet Allocation. In: Advances in Neural Information Processing Systems 20, pp. 1081–1088 (2007)
11. Newman, D., Smyth, P., Steyvers, M.: Scalable Parallel Topic Models. Journal of Intelligence Community Research and Development (2006)
12. Teh, Y.W., Newman, D., Welling, M.: A Collapsed Variational Bayesian Inference Algorithm for Latent Dirichlet Allocation. In: Advances in Neural Information Processing Systems 19, pp. 1378–1385 (2006)
13. Wang, X.G., Grimson, E.: Spatial Latent Dirchlet Allocation. In: Advances in Neural Information Processing Systems 20, pp. 1577–1584 (2008)
14. Xing, D.S., Girolami, M.: Employing Latent Dirichlet Allocation for Fraud Detection in Telecommunications. Pattern Recog. Lett. 28, 1727–1734 (2007)
15. Li, J., Wang, J.Z.: Automatic Linguistic Indexing of Pictures by a Statistical Modeling Approach. IEEE Trans. Pattern Anal. Mach. Intell. 25, No. 9, 1075–1088 (2003)
16. Wang, J.Z., Li, J., Wiederhold, G.: SIMPLIcity: Semantics-sensitive Integrated Matching for Picture LIbraries. IEEE Trans. Pattern Anal. Mach. Intell. 23, No. 9, 947–963 (2001)