

## OPTIMIZATION OF DISTRIBUTED CONVOLUTIONAL NEURAL NETWORK FOR IMAGE LABELING ON ASYNCHRONOUS GPU MODEL

JINHUA FU<sup>1,2</sup>, YONGZHONG HUANG<sup>1</sup>, JIE XU<sup>3,\*</sup> AND HUAIGUANG WU<sup>2</sup>

<sup>1</sup>State Key-Laboratory of Mathematical Engineering and Advanced Computing  
No. 62, Science Avenue, High-Tech Zone, Zhengzhou 450001, P. R. China  
yzhuangz@tom.com

<sup>2</sup>School of Computer and Communication Engineering

<sup>3</sup>School of Software  
Zhengzhou University of Light Industry  
No. 5, Dongfeng Road, Zhengzhou 450002, P. R. China  
zzuliteacher@126.com; \*Corresponding author: jiexuzz@tom.com

Received July 2018; revised November 2018

**ABSTRACT.** *The problem of image labeling consists in assigning semantic labels to every pixel in an image. Recently, given their capacity to learn rich features, many state-of-the-art techniques make use of Convolutional Neural Networks (CNNs) for dense image labeling tasks (e.g., multi-class semantic segmentation). In this paper, we propose the optimization of distributed CNN for image labeling on asynchronous GPU (Graphics Processing Unit) model. We deduce equations for ADMM (Alternating Direction Method of Multipliers) which has been widely used in variety of optimization problems, and apply the optimization to distributed convolutional neural networks training. Our proposed framework is designed and implemented in GPU-GPU communication depending on the asynchronous model. The results show that we could get good pixel accuracy of image labeling while having a fast labeling speed.*

**Keywords:** Image labeling, Convolutional neural networks, Asynchronous GPU model, Labeling speed

**1. Introduction.** In the computer vision field, the labeling task aims at getting pixel-wise dense labeling of an image in terms of semantic concepts such as tree, road, sky, water, building, and foreground objects. It consists of labeling each pixel in an image with the category of the object it belongs to, which is an important step towards image understanding. However, the scene labeling is very challenging, and it implies solving jointly detection, segmentation and multi-label recognition problems.

Image labeling has been approached with variety of methods in the past. Traditionally, CRF (Conditional Random Field) [1] and MRF (Markov Random Filed) [2] provide powerful tools for building models to label image segments. They are particularly well-suited to model local interactions among adjacent regions. Most methods rely on CRF [3], MRF [4] or similar types of graphical models [5, 6] to ensure consistency in labeling. They can compute a simultaneous labeling of image regions into semantic classes (e.g., tree, building, car) and geometric classes (sky, vertical, ground). These methods are augmented with different approaches from conventional vision literature, involving CRF or MRF over those superpixels and multilevel segmentation with class purity criterion. However,

they are limited in dealing with complex, global (long-range) interactions between regions. Some pre-segmentation methods of image into superpixels [7, 8] are exploited, which extract features from individual segments to get the most consistent labels for the image. However, their methods alone do not accurately pinpoint the object boundaries. Then, Liu et al. [9] show how the parameters of the label tree can be found using maximum likelihood estimation. This new probabilistic learning technique produces a label tree with significantly improved recognition accuracy. At the same time, Kae et al. [10] show how an RBM (Restricted Boltzmann Machines) can be added to the architecture to provide a global shape bias that complements the local modeling, which demonstrates its labeling performance for the parts of complex face images from the labeled faces. Although they allow for a finer segmentation and optimize the segmentation process with a global objective of class purity, they still have to improve the labeling accuracy and speedup labeling process.

As computing power increases, GPUs can solve large data parallel problems at a higher speed than the traditional CPU, while being more affordable and energy efficient than distributed systems [11]. Furthermore, GPU can enable concurrent visualization and interactive segmentation, where the user can help the algorithm to achieve a satisfactory segmentation result [12]. On the other hand, many state-of-the-art techniques make use of convolutional neural networks for dense image labeling tasks [13], which has recently become one of the most powerful learning tools in computer vision, and achieved good results in various computer vision tasks. Based on GPU hardware accelerated implementation, Pinheiro and Collobert [14] state that backpropagated gradients on image are equivalent to convolutional networks by Zeiler et al. [15] wherein, they reverse the layers and apply them on the generated code. Deep Convolutional Neural Networks (DCNNs) Feature Extractor (DCNN-FE) has been widely applied in many applications and achieved great success. It is true that CNNs handle much smaller number of parameters than the standard neural networks by sharing weights (convolution layers), but several state-of-the-art CNNs applications such as hyperspectral image segmentation [16, 17], image action recognition [18] and image retrieval [19] still require a lot of parameters. The large number of parameters make training hard, causing fatal issues such as overfitting and local optima. More recently, convolutional neural networks as important parts of DCNNs have achieved great successes in the field of computer vision [20]. However, convolution always takes much computation time in the DCNNs.

In order to improve the efficiency of CNNs, many solutions [21, 22, 23] focusing on training algorithms and parallelism strategies have been proposed. Their results are considerably better than the traditional image feature extraction and segmentation, and show great potential of CNN used for image interpretation. An improved deep learning framework brief-net based on CNN [24] is presented, which provides higher accuracy of identification and classification for the image datasets considered. Also Maggiori et al. [25] establish the desired properties of an ideal semantic labeling CNN, and assess how those methods stand with regard to these properties for high-resolution aerial image labeling. Nowadays, in the applications such as online recognition [26], ocean emergency warning [27] and fast applications [28, 29], the main challenge of image segmentation and labeling is not only to give a high accuracy, but also it should take less computation time and integrate processing with existing GPU pipelines easily.

To accelerate the process of image labeling, we propose a novel asynchronous distributed CNN optimization using Alternating Direction Method of Multipliers (ADMM). Unlike previous works on distributed optimization for neural network training which rely solely on the primal optimization, we formulate the problem into a global consensus optimization and distribute the neural network training in a principled fashion. In this paper,

we present distributed CNN training framework based on ADMM, and design the data parallelization scheme applied to image labeling. We deduce equations for ADMM, which has been widely used in variety of optimization problems. Then we present the ADMM-based parameter update equations adapted to CNN training context and design distributed implementation based on asynchronous GPU model. Experimental results have demonstrated that the proposed method can achieve satisfactory labeling accuracy and speed.

The rest of this paper is organized as follows. Scene labeling and its implementation flow are given in Section 2. Optimization of convolutional neural network is described in Section 3. GPU-based distributed CNN training framework is deduced and designed in Section 4. At last, experimental results are discussed in Section 5, and finally conclusions are given in Section 6.

**2. Scene Labeling and Its Implementation Flow.** In the context of scene labeling, given an image  $I$  we are interested in finding the label of each pixel at location  $(i, j)$  in the image. More precisely, the network is fed with a squared context patch  $I_{i,j,k}$  surrounding the pixel at location  $(i, j)$  in the  $k^{\text{th}}$  image. The main difficulty of scene labeling is to select appropriate features and how to use contextual semantic information. Features provide a compact and concise representation for scene image. A good feature should be robust to accommodate intra-class variations and also provides good inter-class discriminativeness. Few other properties of good feature descriptors are locality, pose-invariance, scale-invariance, repeatability, etc. There is one example of scene labeling as shown in Figure 1.

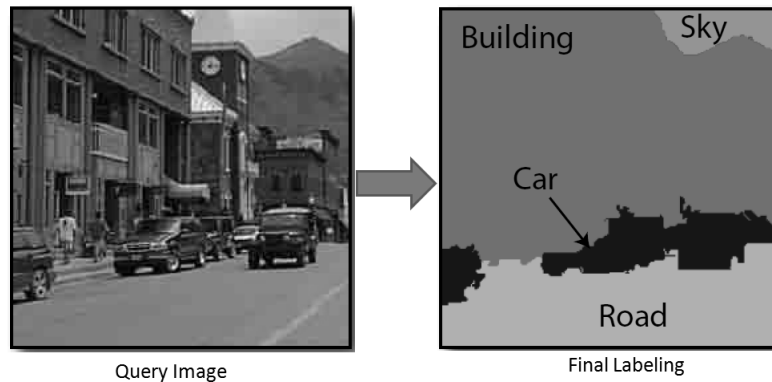


FIGURE 1. From the query image to scene labeling

From Figure 1, on the left there is one scene of city street for image labeling. Four categories of the object region on the right, that is, sky, building, car and road are segmented based on feature information and semantic concepts. Therefore, the scene labeling task consists of partitioning the different meaningful regions of one image and labeling pixels with their regions.

In order to get more accurate labeling information, it is necessary to use multi-scale semantic information, namely, the semantic features of close adjacent pixels, different objects in long range and interconnections between backgrounds. Recently, convolutional neural network has obtained good classification and detection results on several image datasets. So it has been paid more attention to extract features and produce better robustness of classification. However, the problem of scene labeling is not only a classification problem, but a structural prediction problem. The graph model structure can predict the structure better and establish semantic association on the basis of extracting

features. Therefore, combining the convolutional neural network and the graph model, such as conditional random fields, we can solve the problem of scene labeling.

In scene labeling, we use distributed convolutional neural networks to label each pixel. It utilizes different scales of context information. After coarse labeling of CNN, a fully-connected conditional random field corrects some labeling mistakes. Implementation flow of our scene labeling is shown in Figure 2.

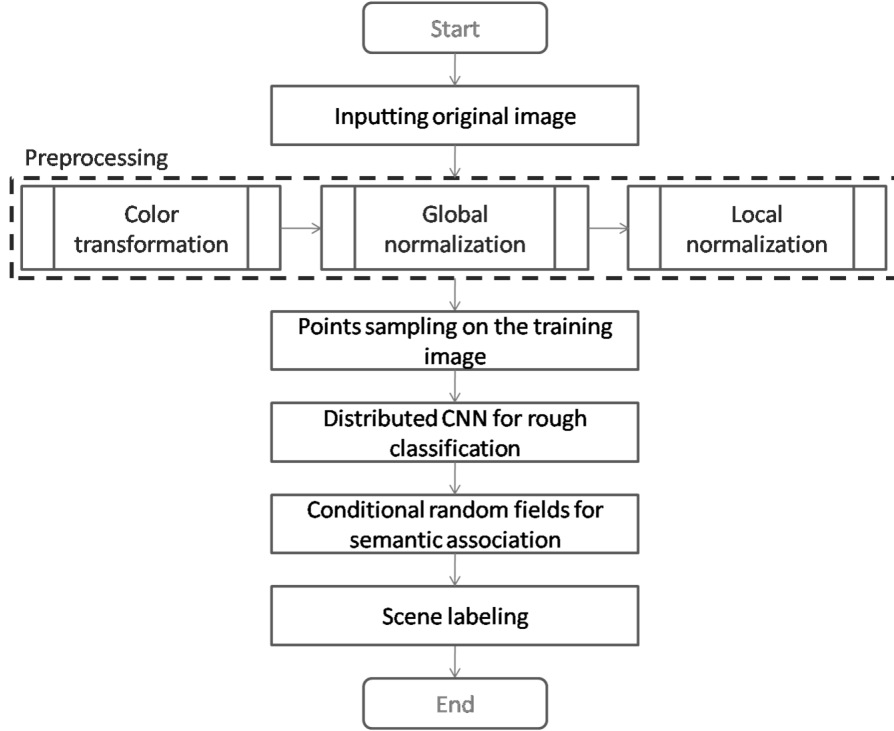


FIGURE 2. Whole implementation flow of scene labeling

From Figure 2, there are four major steps to accomplish the labeling task as follows.

- Preprocess of the original image: the original *RGB* (Red, Green and Blue) color channel is transformed into *YUV* color channel, where the *Y* component, also called *luma*, represents the brightness value of the color, and the *U* and *V* components are called *chroma values* or *color difference* values. The global normalization is processed on each channel. In the local region, subtraction normalization and division normalization are applied later.

- Points sampling: several points are randomly sampled on the training image to keep the training samples of different classes balanced.

- CNN training: during the training process, using distributed CNN, features are extracted around the sampling point in different scale windows. The rough classification is carried out by the CNN classifier.

- Scene labeling: based on classification, the full connection CRF is used for smooth processing, in which the edge potential cluster is the color and position relation. The vertex potential cluster is the number of categories, that is, the output of scene labeling via distributed CNN.

**3. Optimization of Convolutional Neural Network.** After given the definition of scene labeling and its implementation flow in Section 2, we introduce how to use CNN to implement the scene labeling and solve the complex dual problem in CNN by applying the projection gradient method in this section.

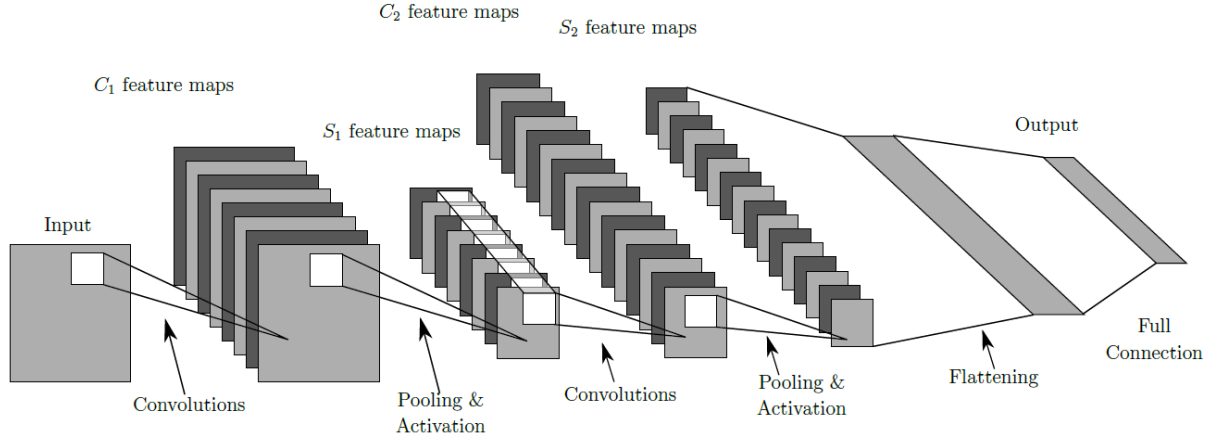


FIGURE 3. Model of convolutional neural network

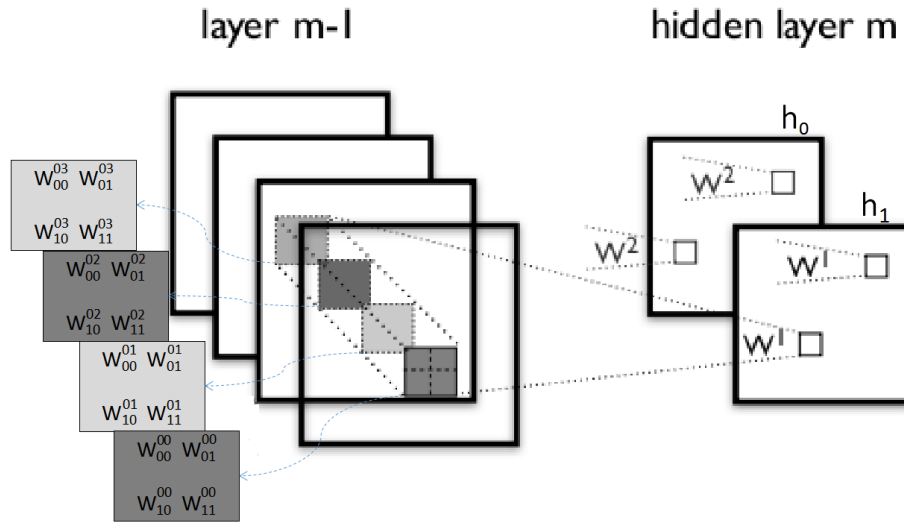


FIGURE 4. Example of a convolutional layer

Convolutional neural networks are state-of-the-art models for many image classification and labeling tasks, which are designed as shown in Figure 3. From Figure 3, our CNN consists of a series of sublayers, namely a convolution (filter bank) sublayer, a non-linearity sublayer, a local response normalization sublayer, and a feature pooling sublayer followed by an output. Each type of layer contains several feature maps, or groups of neurons, in a rectangular configuration.

To be specific, Figure 4 shows two layers of a CNN. Layer  $m - 1$  contains four feature maps. Hidden layer  $m$  contains two feature maps ( $h_0$  and  $h_1$ ). Pixels (neuron outputs) in  $h_0$  and  $h_1$  are computed from pixels of layer  $(m - 1)$  which fall within their  $2 \times 2$  receptive field in the layer below.

In this paper we would discuss the *Objective Function* of CNN and the back propagation algorithm to compute the gradient with respect to the parameters of the model in order to use gradient based on optimization. The act of training a CNN is a (mainly) supervised learning problem. It is the task of adjusting the output function to match what is reflected in the training image. The first step to perform this task is quantifying how well the current output matches the desired output. We call this quantifier the *Objective Function*, but it is equivalently referred to as a *Loss Function* or a *Cost Function*. Once we have found such a fitting measure, we can rephrase the training of a neural network as a

mathematical optimization problem. It is our objective to adjust the weights such that they minimize the *Loss/Cost/Objective Function*.

The goal of the CNN is to collectively solve the following convex optimization problem

$$\text{minimize } f(w) \quad \text{subject to } Aw = b \quad (1)$$

where  $f$  is the *Objective Function* and  $w$  is a primal variable.  $A$  is an  $r \times n$  matrix with rows  $a_i^T$ , and  $b \in \mathbb{R}^r$  is a vector with components  $b_i$ .

In some situations, the partial dual problem is considered and is still referred to as the dual. In particular, consider relaxing only the inequality constraints of the problem Equation (1), yielding a dual problem of the form

$$g(\lambda) = \inf\{\mathcal{L}(w, \lambda)\} \quad (\text{Dual function}) \quad (2)$$

where the dual function  $\mathcal{L}$  is given by

$$\mathcal{L}(w, \lambda) = f(w) + \lambda^T(Aw - b) \quad (\text{Lagrangian}) \quad (3)$$

where  $\lambda$  is a dual variable.

To solve the dual problem, we can now apply the projection gradient method, which is adapted to handle maximization. The projected gradient method for the dual problem has the new form. Since  $\nabla_\lambda g(\lambda) = Aw^+ - b$  where  $w^+ = \arg \min_w \mathcal{L}(w, \lambda)$ , the update equations can be written as follows:

$$w^{k+1} \leftarrow \arg \min \mathcal{L}(w, \lambda^k) \quad (w\text{-minimization step}) \quad (4)$$

$$\lambda^{k+1} \leftarrow \lambda^k + \eta (Aw^{k+1} - b) \quad (\text{Dual update}) \quad (5)$$

where  $\eta$  is an appropriate step size.

Finally, we can solve the dual problem in CNN by alternately repeating the above update equations Equation (4) and Equation (5) until convergence would result in the optimal solution.

**4. GPU-Based Distributed CNN Training Framework.** The optimization of convolutional neural network is given in Section 3, the detailed solution of the convex optimization problem in CNN has been acquired. So in Section 4, we represent the distributed implementation of CNN via improved ADMM algorithm in Section 4.1 and design the asynchronous peer-to-peer GPU model for the distributed implementation in Section 4.2.

**4.1. Design of distributed CNN.** Considering separability of the *Objective Function* and improved ADMM algorithm, CNN training in this paper can be designed for distributed implementation. We assume that  $W$  is a two dimensional parameter matrix,  $X = \{x_i\}_{i=1}^N$  is the set of training scene image and  $Y = \{y_i\}_{i=1}^N$  is the set of corresponding class features of scene image. We can represent the final  $L_2$ -regularized loss in terms of a linear sum of individual losses over the whole training scene images. The image dataset  $D$  is separated into  $m$  subsets  $\{D_j\}_{j=1}^m$ , and also the new *Objective Function* of CNN  $F$  could be split into  $m$  disjoint terms as follows:

$$\begin{aligned} F(W; D) &= \frac{1}{N} \sum_{i=1}^N l(W; x_i, y_i) + \frac{\lambda}{2} \|W\|_F^2 \\ &= \frac{1}{N} \sum_{j=1}^m \sum_{i \in D_j} l(W; x_i, y_i) + \frac{\lambda}{2} \|W\|_F^2 \\ &= \sum_{j=1}^m \left[ F_j(W_j; D_j) + \frac{\lambda}{2m} \|W\|_F^2 \right] \end{aligned}$$

In order to exploit the GPU parallelization scheme, each of terms  $m$  should be fed into one of the GPU processors. Although we divide the process into  $m$  separate terms independently, all GPU processors solve the same problem actually. So combining ADMM with the global consensus optimization problem by introducing the consensus variable  $Z$ , we can acquire the *Objective Function* optimization problem of convolutional neural network as follows

$$\begin{aligned} & \underset{W_j}{\text{minimize}} \quad \sum_{j=1}^m \left[ F_j(W_j; D_j) + \frac{\lambda}{2m} \|W_j\|_F^2 \right] \\ & \text{subject to} \quad W_j = Z, \quad j = 1, \dots, m \end{aligned}$$

Then the augmented Lagrangian of the *Objective Function* can be obtained as follows

$$\mathcal{L}_\rho(W, Z, \Lambda) = \sum_{j=1}^m \left[ F_j(W_j, D_j) + \frac{\lambda}{2m} \|W_j\|_F^2 \right] + \text{tr}(\Lambda_j^T (W_j - Z)) + \frac{\rho}{2} \|W_j - Z\|_F^2 \quad (6)$$

where  $\rho$  is the dual variable of augmented Lagrange,  $\Lambda_j$  is the  $j^{\text{th}}$  dual multiplier matrix of Lagrange, and  $\text{tr}(\Lambda_j^T (W_j - Z))$  is the sum of all the matrix elements after multiplying  $\Lambda_j$  with  $W_j - Z$ .

In order to find a minimizer of the constrained problem Equation (6), the improved ADMM algorithm uses three iterative computations for the  $j$ -th data subset as follows.

1) Make use of a descent method to solve the following performance promoting problem,

$$W_j^{k+1} = \arg \min_W \mathcal{L}_{\rho,j}(W, Z^k, \Lambda_j^k) \quad (7)$$

2) Find the analytical expressions for the solutions of the following sparsity promoting problem,

$$Z^{k+1} = \arg \min_Z \mathcal{L}_\rho(W_j^{k+1}, Z, \Lambda_j^k) \quad (8)$$

3) Update the dual variable  $\Lambda$  using a step-size equal to  $\rho$ , in order to guarantee that the dual feasibility condition is satisfied in each ADMM iteration,

$$\Lambda_j^{k+1} = \Lambda_j^k + \rho (W_j^{k+1} - Z^{k+1}) \quad (9)$$

Later we use  $\rho U_j$  to replace  $\Lambda_j$ , and simplify the update equations, which is called scaled version of ADMM as follows

$$W_j^{k+1} = \arg \min_W F_j(W; D_j) + \frac{\lambda}{2m} \|W\|_F^2 + \frac{\rho}{2} \|W - Z^k + U_j^k\|_F^2 \quad (10)$$

$$Z^{k+1} = \frac{1}{m} \sum_{j=1}^m (W_j^{k+1} + U_j^k) \quad (11)$$

$$U_j^{k+1} = U_j^k + (W_j^{k+1} - Z^{k+1}) \quad (12)$$

Finally, we compute  $W_j^{k+1}$  using the back propagation of CNN via gradient descent method as follows

$$\frac{\partial}{\partial W_{j,pq}^l} \mathcal{L}(W, Z^k, \Lambda^k) = a_{j,q}^l \delta_{j,p}^{l+1} + \left( \frac{\lambda}{m} + \rho \right) W_{j,pq}^l + \rho (U_{j,pq}^l - Z_{pq}^l) \quad (13)$$

$$\frac{\partial}{\partial b_{j,pq}^l} \mathcal{L}(W, Z^k, \Lambda^k) = \delta_{j,p}^{l+1} + \rho (b_{j,pq}^l - Z_{pq}^l + U_{j,pq}^l) \quad (14)$$

$$W_{j,pq}^l = W_{j,pq}^l - \alpha \frac{\partial}{\partial W_{j,pq}^l} \mathcal{L}(W, Z, \Lambda) \quad (15)$$

**4.2. Asynchronous peer-to-peer GPU model.** Our proposed framework can be implemented in the way depending on the communication strategy among GPU processors. The asynchronous GPU model is the *peer-to-peer model*, where client processors inter-communicate directly to transfer parameters without the server processor.

Instead of reporting  $W$  and  $U$  to the server and receiving the updated  $Z$  from the server as in the *client-server* model,  $Z$  is defined in all the processors, and is updated locally. When the client reaches certain period, it gathers up the  $U + W$  values from every client, and computes  $Z$  by using Equation (11). Figure 5 shows how the processor 0 gathers up parameters from other processors.

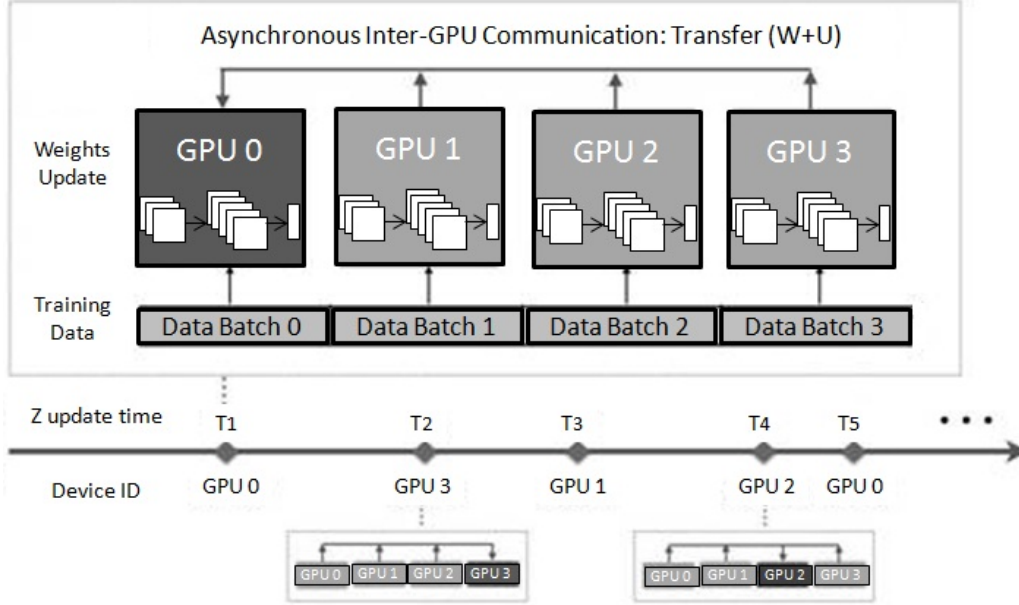


FIGURE 5. Asynchronous peer-to-peer GPU model

From Figure 5, in the peer-to-peer model, we assign the number of processors as  $m = 4$ . We classify the training data into four data batches, and transfer data batch 1, data batch 2 and data batch 3 to three GPU processors for proceeding asynchronously in different processors. Then the processor 0 can update its local  $Z$  by gathering up the  $U + W$  values from three clients via inter-GPU communication. The  $Z$  computing process is repeated periodically, and is proceeded asynchronously in different processors.

**5. Experiment and Results.** Based on GPU platform, we have implemented the efficient scene labeling using distributed convolutional neural networks on asynchronous peer-to-peer GPU model. Our final labeling is tested on a computer equipped with experiment on Intel i5-4670k@3.4Ghz, 16GB DDR3@1800Mhz and Nvidia GeForce GTX 780 Ti 3GB with 2880 CUDA Cores. We have realized the method C++ programming in Visual Studio 2016. We have achieved relatively high speed of labeling and our results clearly show the effective implementation of our technique.

We use the Stanford Background Dataset [30] to test, train and validate our proposed networks. The Stanford Background Dataset consists of 715 images having an approximate size of  $320 \times 240$  pixels, and each pixel labels into one of the following categories: sky, tree, road, grass, water, building, mountain, foreground, unknown. In addition, we add one more category: undefined, to take account of the variable image size. If the image is of size  $300 \times 240$ , then the last 20 columns of the resultant image (of size  $320 \times 240$ )

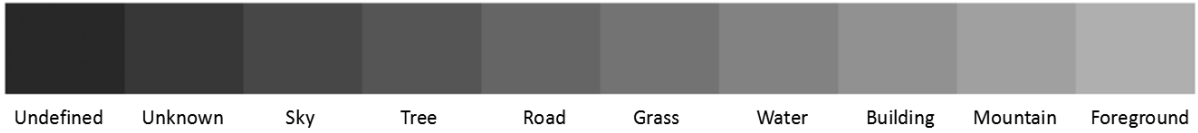


FIGURE 6. Ten categories when labeling scene image



FIGURE 7. One scene labeling result. (L-R): input image, ground truth labels, output learnt labels.

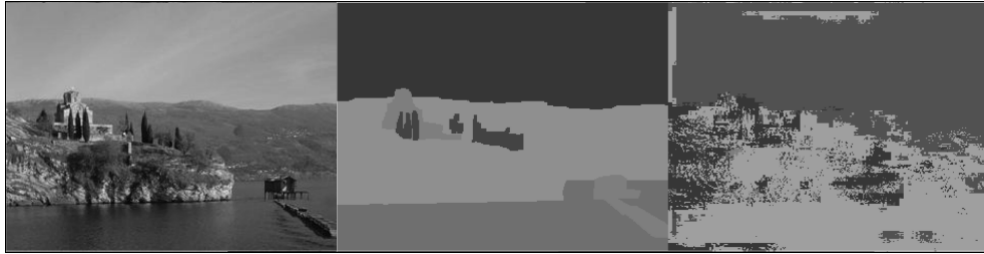


FIGURE 8. Another scene labeling result. (L-R): input image, ground truth labels, output learnt labels.

are labeled as undefined. Thus, our goal is to classify each pixel in an image into one of the ten categories.

We use the first 429 images out of all the 715 images for training, and equally split up the remaining (143 each) for validation and testing. Although this may appear to be a small dataset, we are training for each pixel and so, effectively, the number of training, validation and test points is 32947200, 10982400 and 10982400 respectively (since image size is  $320 \times 240$ ). To visualize our results, each of ten labels is given a unique grey scale color as shown in Figure 6.

We evaluate our results based on per pixel and per class accuracy. We achieve an average per pixel accuracy of only 93% on the validation data and 95% on the test data after training for about 90 epochs. Figure 7 and Figure 8 show two scene labeling results on the basis of per pixel accuracies.

As shown in Figure 7 and Figure 8, The location features are important in discrimination few classes from other. For example, ‘sky’ generally appears in top portion of images, ‘grass’, ‘water’ and ‘road’ appear in bottom portion of images, and ‘foreground objects’ appear in center portion of images. The labeling result is quite good and is close to the true object in scene. Also the boundaries seem to be correct and average accuracy of scene labeling is relatively high.

In order to show the effectiveness of our method, we give an objective evaluation for validation accuracies for our model across training iterations as shown in Figure 9.

From Figure 9, we achieve a validation accuracy of approximately 93%. Especially in some categories such as sky, road and water, we have obtained a better labeling effect. The

Class	Accuracy	Class	Accuracy
Undefined	0.98	Grass	0.92
Unknown	0.97	Water	0.97
Sky	0.96	Building	0.83
Tree	0.92	Mountain	0.85
Road	0.97	Foreground	0.92

FIGURE 9. Per class accuracies

TABLE 1. Labeling speed comparison via different methods

Different scenes	Pinheiro's method [14]	Jiang's method [21]	Our method
Scene in Figure 7	25s	19.8s	4.7s
Scene in Figure 8	23s	17.4s	2.0s

LogLoss value, which is the *Objective Function* to minimize to fit a log linear probability model to a set of labeled images, achieved using this model is approximately 0.08 and it shows the model is reasonable.

Then, we implement the method for high-performance image labeling. which achieves the state-of-the-art performance. To give an objective evaluation, we implement different labeling for comparison with two methods [14, 21] as shown in Table 1.

From Table 1, we obtain a relatively faster labeling speed as a result of GPU acceleration. When the entire distributed convolutional neural networks can be put on many single GPUs instead of single CPU cluster, everything is much faster: communication latency is reduced, bandwidth is increased, and size and power consumption are significantly decreased, which make speed of image labeling faster. In addition, a very large number of features can be extracted from a super-pixel. However, increasing the number of features would increase the computation time for extracting the features, time of training the classifier, and time of predicting the labels after training.

**6. Conclusions.** Instead of naively distributing the computation over multiple processors, we attempt to apply one of the distributed optimization techniques to training convolutional neural networks, and implement an efficient scene labeling on GPU. Our framework is based on the combination of ADMM and global consensus optimization. We divide ten categories when labeling scene, and use different grey scale colors to visualize our labeling results. From labeling visualization, validation accuracy and LogLoss value, we can acquire efficient scene labeling with faster labeling speed.

To our knowledge, this is very useful to apply ADMM to neural network training, so it seems meaningful to further explore the problem. We also plan to experiment on different datasets with larger networks where the communication overhead of the naive data parallelism becomes worse.

**Acknowledgment.** This work was supported by the National Natural Science Foundation of China (No. 61672470), the Intergovernmental Special Program of National Key-point Research Project of China (Nos. 2016YFE0100300, 2016YFE0100600), and the Science and Technology Foundation of Henan Province Education Department of China (No. 14A520061).

## REFERENCES

- [1] J. Jiang and Z. Tu, Efficient scale space auto-context for image segmentation and labeling, *IEEE Conference on Computer Vision and Pattern Recognition*, pp.1810-1817, 2009.
- [2] S. Kumar, R. Gupta, N. Khanna and J. Sd, Text extraction and document image segmentation using matched wavelets and MRF model, *IEEE Trans. Image Processing*, vol.16, no.8, pp.2117-2128, 2007.
- [3] L. Yu, J. Xie and S. Chen, Conditional random field-based image labeling combining features of pixels, segments and regions, *IET Computer Vision*, vol.6, no.5, pp.459-467, 2012.
- [4] R. Song, Y. Liu, R. R. Martin and P. L. Rosin, MRF labeling for multi-view range image integration, *Asian Conference on Computer Vision*, Springer-Verlag, pp.27-40, 2010.
- [5] J. I. Gómez and N. P. D. L. Blanca, Labeling still image databases using graphical models, *Iberian Conference on Pattern Recognition and Image Analysis*, pp.330-337, 2009.
- [6] C. Farabet, C. Couprie, L. Najman and Y. LeCun, Learning hierarchical features for scene labeling, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.35, no.8, pp.1915-1929, 2013.
- [7] J. Tighe and S. Lazebnik, SuperParsing: Scalable nonparametric image parsing with superpixels, *Proc. of European Conference on Computer Vision*, pp.352-365, 2010.
- [8] I. Jebari and D. Filliat, Color and depth-based superpixels for background and object segmentation, *Procedia Engineering*, vol.41, pp.1307-1315, 2012.
- [9] B. Liu, F. Sadeghi, M. Tappen, O. Shamir and C. Liu, Probabilistic label trees for efficient large scale image classification, *IEEE Computer Vision and Pattern Recognition*, pp.843-850, 2013.
- [10] A. Kae, K. Sohn, H. Lee and E. Learned-Miller, Augmenting CRFs with Boltzmann machine shape priors for image labeling, *IEEE Conference on Computer Vision and Pattern Recognition*, pp.2019-2026, 2013.
- [11] E. Smistad, T. L. Falch, M. Bozorgi, A. C. Elster and F. Lindseth, Medical image segmentation on GPUs – A comprehensive review, *Medical Image Analysis*, vol.20, no.1, pp.1-18, 2015.
- [12] W. Song, Y. Tian, S. Fong et al., GPU-accelerated foreground segmentation and labeling for real-time video surveillance, *Sustainability*, vol.8, no.10, p.916, 2016.
- [13] D. Malmgrenhansen and M. Nobeljorgensen, Convolutional neural networks for SAR image segmentation, *IEEE International Symposium on Signal Processing and Information Technology*, pp.231-236, 2016.
- [14] P. Pinheiro and R. Collobert, Recurrent convolutional neural networks for scene labeling, *Computer Vision and Pattern Recognition*, no.1, pp.82-90, 2013.
- [15] M. D. Zeiler, G. W. Taylor and R. Fergus, Adaptive deep convolutional networks for mid and high level feature learning, *Proc. of IEEE International Conference on Computer Vision*, pp.2018-2025, 2011.
- [16] F. I. Alam, J. Zhou, W. C. Liew and X. Jia, CRF learning with CNN features for hyperspectral image segmentation, *IEEE Geoscience and Remote Sensing Symposium*, pp.6890-6893, 2016.
- [17] Y. Luo, L. Yang, L. Wang and H. Cheng, Efficient CNN-CRF network for retinal image segmentation, *International Conference on Cognitive Systems and Signal Processing*, pp.157-165, 2016.
- [18] Y. Lavinia, H. H. Vo and A. Verma, Fusion based deep CNN for improved large-scale image action recognition, *IEEE International Symposium on Multimedia*, pp.609-614, 2017.
- [19] W. M. Huang, P. Wei and J. H. Liang, Application of CNN-based hashing in image retrieval, *Computer Engineering & Design*, 2017.
- [20] J. Zhao, W. Guo, S. Cui, Z. Zhang and W. Yu, Convolutional neural network for SAR image classification at patch level, *IEEE Geoscience and Remote Sensing Symposium*, pp.945-948, 2016.
- [21] W. Jiang, Y. Chen, H. Jin et al., A novel GPU-based efficient approach for convolutional neural networks with small filters, *Journal of Signal Processing Systems*, pp.1-13, 2016.
- [22] H. Dong, T. Li, J. Leng, L. Kong and G. Bai, GCN: GPU-based cube CNN framework for hyperspectral image classification, *IEEE International Conference on Parallel Processing*, pp.41-49, 2017.
- [23] A. Shustanov and P. Yakimov, A method for traffic sign recognition with CNN using GPU, *International Conference on Signal Processing and Multimedia Applications*, pp.42-47, 2017.
- [24] Q. Wang, X. Li and D. Xu, An improved deep learning framework brief-net based on convolutional neural networks, *ICIC Express Letters*, vol.11, no.8, pp.1323-1330, 2017.
- [25] E. Maggiori, Y. Tarabalka, G. Charpiat et al., High-resolution aerial image labeling with convolutional neural networks, *IEEE Trans. Geoscience & Remote Sensing*, pp.1-12, 2017.

- [26] S. Sen, D. Shao, S. Paul, R. Sarkar and K. Roy, Online handwritten Bangla character recognition using CNN: A deep learning approach, *Intelligent Engineering Informatics*, vol.695, pp.413-420, 2018.
- [27] Y. Wang, L. Fu, K. Liu et al., Stable underwater image segmentation in high quality via MRF model, *IEEE Oceans*, pp.1-4, 2016.
- [28] G. Wang, W. Li, M. A. Zuluaga et al., Interactive medical image segmentation using deep learning with image-specific fine-tuning, *IEEE Trans. Medical Imaging*, 2017.
- [29] A. Guzman-Urbina, A. Aoyama and E. Choi, A polynomial neural network approach for improving risk assessment and industrial safety, *ICIC Express Letters*, vol.12, no.2, pp.97-107, 2018.
- [30] *Stanford Background Dataset*, <http://dags.stanford.edu/projects/scenedataset.html>.