Enhancing the performance of CNN-based blind image steganalysis approach using multi-GPU TESLA P100

240th ECS Meeting
Digital Meeting, Oct 10-14, 2021
We are going fully digital!
Attendees register for free!
REGISTER NOW

Enhancing the performance of CNN-based blind image steganalysis approach using multi-GPU TESLA P100

Eslam M Mustafa [1,2], Mohamed A Elshafey [1,3] and Mohamed M Fouad[1,4]

[1]Department of Computer Engineering, Military Technical College, Cairo, Egypt

E-mail: [2]eng.eslam.mtc@gmail.com, [3]m.shafey@mtc.edu.eg and [4]mmafoad@mtc.edu.eg

Abstract. Blind image Steganalysis is the binomial classification problem of determining if an image contains hidden data or not. Classification problems have two main steps: i) feature extraction step and ii) classification step. Traditional blind image steganalysis approaches use handcrafted filters for the first step and use classifiers such as support vector machine (SVM) for the second step. The rapid development of steganographic techniques makes it harder to design new effective handcrafted filters, which negatively affect the feature extraction step. Recently, Convolutional Neural networks (CNNs) are introduced as an auspicious solution for this problem. CNN-based steganalysis can automatically extract features from the input images without using handcrafted filters. Although considerable success has been achieved with CNNs, CNN-based applications are considered as time consuming applications. Accordingly, it is important to quicken the CNN-based steganalysis approaches training in order to make them more applicable. This paper suggested an implementation technique of the improved Gaussian-Neuron CNN (IGNCNN) steganalysis approach on GPUs. In this paper data parallelism concept is applied to the convolutional layers while model parallelism concept is applied to the fully connected layers. Results show that the proposed method provides better performance as compared with IGNCNN [1] by an average speed up factor of 1.4 X.

Keywords: Image steganalysis, convolutional neural network, deep learning, transfer learning, variable batch size, data parallelism, model parallelism, GPU, TESLA P100, CUDA

1. Introduction

Blind image steganalysis attempts to distinguish the presence of shrouded message in a digital image without any prior information on the steganographic technique used. Blind image steganalysis general approach employs machine learning techniques. Thence, stego and non-stego images are exposed to feature extractors to extract features, a classifier is trained on these features and new stego and non-stego images are used to evaluate the generated model. Support Vector Machines (SVM), Fisher Linear Discriminants (FLD)-based ensemble classifier [2] and artificial neural networks [3] are the most commonly used classifiers.

Features extraction and selection is acritical step for blind image steganalysis to obtain best results. Numerous procedures were utilized for this, for example, covariance matrix, similarity measures between pixels etc. These procedures are very difficult to be handcrafted with the rapid increase in the steganography techniques development. Contemporary blind image steganalysis approaches utilize deep learning represented in the convolutional neural networks (CNNs) [4, 3, 1, 5, 6] which automatically extract the needed features according to the input images.

Despite the remarkable success achieved by CNNs, CNN-based applications are time consuming. Accordingly, it is important to quicken the CNN-based image steganalysis approaches training in order to make them more applicable. Stochastic Gradient Descent

(SGD) [7] is used by deep learning (DL) tasks so as to achieve high performance. This paper presents a blind image steganalysis approach on GPUs in order to enhance the performance of the training step.

The rest of the paper is organized as follows: In Section 2 the related work is presented, while Section 3 presents the proposed approach. After describing experiments in Section 4, we present results and discussion. Conclusions are drawn in Section 5.

2. Related Work

Blind image steganalysis tries to detect the presence of any hidden data in an image regardless the steganographic technique used to hide this data. As probability mentioned before, relevant features extraction and selection are the main challenge in such steganalysis approaches. In traditional blind image steganalysis approaches, such as Spatial Reach Model (SRM) steganalysis approach [8, 2], linear and non-linear filters are used to obtain the rich model of noise residuals. Then these models are assembled using ensemble classifiers to generate the final steganalysis model. [9] presents a blind image steganalysis technique based on the wavelet decomposition. This approach can be divided to three steps. The first step is to perform the wavelet decomposition, the second is extracting the entropy, energy and Probability Density Function (PDF) from the joint density matrix of sub-bands difference coefficients as features. At the last step an SVM classifier is used for classification. A new approach is presented in [10] based on the properties of the Gray Level Co-occurrence Matrix (GLCM). This approach trains a multilayer perceptron (MLP) neural network on new special features. These features are discovered using the GLCM of non stego images and stego images. The presented approach in [11] also use a multilayer perceptron but with backpropagation algorithm to predict the class of the input images. Detection of the embedded data is done using Preprocessed Vectors Diagonal Back Propagation Algorithm (PVDBPA). Class prediction is done using the Euclidean distance. Authors in [12] present a steganalysis approach depends on the probability of embedding data in the image pixels. This approach is effective, especially for low payloads. In [3] CNN is introduced as an impressing approach for image steganalysis called Gaussian-Neuron CNN (GNCNN). This approach has the ability of learning features automatically using its convolutional layers. The classification can be guided within the feature learning process due to the consolidation of feature extraction and classification steps. The results of this approach are slightly worse when compared with other state-of the-art steganalysis methods, especially for low payloads, which is the main drawback of this approach.

The training process of CNN is time-consuming. It may take days or weeks to finish a large-scale training project. Parallelizing CNN training is a must to speed-up the process. Stochastic Gradient Descent (SGD) is used by Deep learning (DL) tasks so as to achieve high performance. A considerable amount of literature has been published on SGD and CNN model parallelization.

In [13] Hogwild! parallelization approach is presented. This process allows the deep learning model (CNN) to access and overwrite the shared memory of the GPU which helps the model to reach a nearly optimal converge rate faster. The DistBelief parallelizing approach presented in [14] extend the work in [13] to obtain good results, especially with nonconvex models. Work in [13] is also extended in [15]. [15] uses asynchronous stochastic gradient descent with many GPUs (GPU A-SGD) to minimize the time consumed by the training process of a neural network. GPU A-SGD take the benefit of using model and data parallelism. In [16], GPU cluster is used to accelerate the training process. Image is split spatially. These splits are assigned to the GPUs. Synchronization at every convolutional layer is a must, so as to grantee that the training generates a consistent model. Authors in [17] present a CNN parallel-computing-based approach. This approach improves the model accuracy and accelerates the training.

In [18], an approach to select an effective range learning rates is presented. This range is obtained by training the model several times with different learning rates and calculate the loss then, choose the effective range. Suitable learning rate helps the CNN to converge faster. In [7], data parallelism method for CNN models is presented. In this method, the full model is assigned to each of used GPUs so as to allow each GPU to train the model in a different set of training data. This method helps to accelerate the training time of CNNs whoever, it needs weight synchronization. Model parallelism presented in [19] is another method to accelerate the training time of CNNs. In this method, model is split to parts so as to train this part on different GPU.

In [1] an image steganalysis approach called Improved Gaussian-Neuron CNN (IGNCNN) is presented. IGNCNN approach is based on transfer learning method. First the CNN model is pre-trained on images with high payload and learn feature representations from these images. Then the parameters of the pre-training CNN are transferred to a CNN called fine-tuning CNN. Finally, the fine-tuning CNN is trained on images with lower payload. Transfer learning method improves the detection accuracy of the stego-image with lower payload. Results of this improved version presents comparable results with other state-of the-art steganalysis methods. Whoever IGNCNN suffers from the following problems:

- IGNCNN uses a fixed learning rate which may negatively affect the performance and the detection accuracy.
- IGNCNN is implemented using one GPU to accelerate the training process. whoever, using single GPU prevents this approach from using model parallelism method.
- Single GPU implementation prevents IGNCNN from using data parallelism method.

Solving these previous problems helps IGNCNN image steganalysis approach to achieve high performance.

3. Proposed Approach

In this section, the proposed approach is introduced. This approach shows that, training a CNN-based image steganalysis approach can be accelerated using multiple GPUs, a cyclic learning rate, model parallelism and data parallelism methods. This section is divided into four subsections. The First subsection is concerned with the Improved Gaussian-Neuron CNN [1] network structure. The second subsection presents the training using cyclic learning rate. The third subsection shows the effect of model and data parallelism methods on training process. The fourth subsection describes the workflow scenario of the proposed approach.

3.1. Structure of IGNCNN image steganalysis approach

The IGNCNN approach [1] is based on transfer learning method. First the CNN model is pre-trained on images with high payload and learn feature representations from these images. Then the parameters of the pre-training CNN are transferred to a CNN called fine-tuning CNN. Finally, the fine-tuning CNN is trained on images with lower payload. Pre-training and fine-tuning CNNs have the same structure. They are consisted of a pre-processing layer (P1), five convolutional layers (C1, C2, C3, C4 and C5) and three fully connected layers (F1, F2 and F3) as shown in Fig 1.

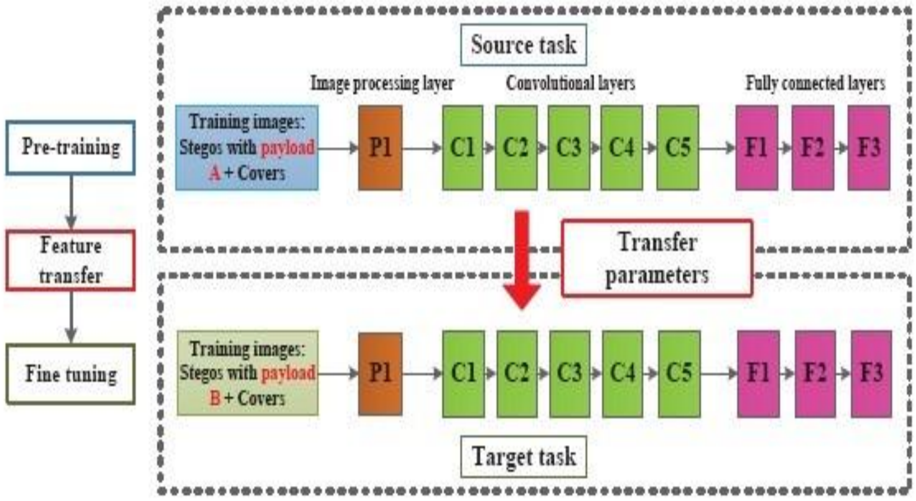

Figure 1. IGNCNN [1] Pre-training and fine-tuning CNN structure. A network with a structure similar to IGNCNN [1] structure is considered for the proposed approach. The following subsection is concerned with the cyclic learning rate and its effects on the training process.

3.2. Training using a cyclic Learning rate

There are hyper-parameters which have to be tuned in order to improve the performance of deep neural networks. The most important hyper-parameter is learning rate. Weights of Deep learning models is updated during training using an optimizer called a stochastic gradient descent optimizer. These weights are updated as shown in 1, where θ denotes weights, L denotes a loss function and ϵ_t denotes learning rate.

$$\theta^t = \theta^{t-1} - \epsilon_t \frac{\partial L}{\partial \theta} \tag{1}$$

Learning rate is the parameter that manages updating the weights of the network according to the loss gradient. It seems to be a good idea to use a low learning rate to make sure that no local minima are missed. Whoever it leads the network to take a long time to converge. In case of using a high learning rate, weight updates can be so big that the optimizer miss the minimum and makes the loss worse which is the reason that the network may neither converge nor diverge.

The proposed approach to select an effective range learning rates starts from a low learning rate, then the learning rate is increased for every batch. The loss gradient and the learning rate are recorded for each batch then plotted. It can be noticed that there is a range of learning rates in which the loss decreases fast [min_{LR} max_{LR}]. This range is denoted as effective learning rates (Cyclic learning rate [18]) as shown in Fig 2 where, min_{LR} and max_{LR} denote the minimum and the maximum learning rate in the effective learning rate range respectively.

A well-chosen cyclic learning rate helps the CNN to converge faster and improve its detection accuracy as well. The following subsection shows the effect of model and data parallelism methods on the training process.

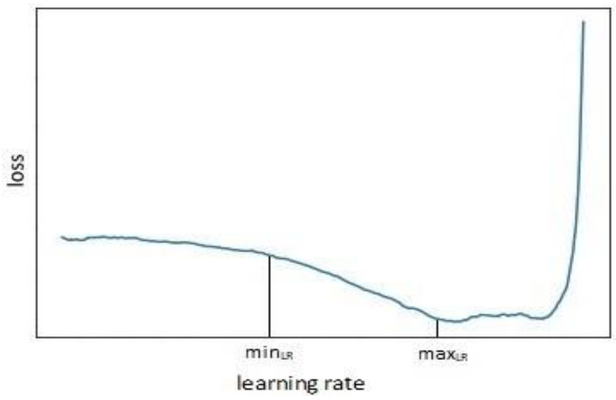

Figure 2. Loss variation due to changing learning rate in the proposed approach. The loss decreases in the beginning, then the training process starts diverging due to increasing the learning rate. min_{LR} and max_{LR} denote the minimum and the maximum learning rate in the effective learning rate range respectively.

3.3. Training using Model Parallelism and Data Parallelism

CNN training is a time-consuming process which may stand for days or weeks in case of big datasets. In order to accelerate the training process, parallelizing approaches is needed. Two parallelizing approaches can be used: data parallelism and model parallelism.

- Data parallelism:
  In this approach, the full model is assigned to each of used GPUs so as to allow each GPU to train the model in a different set of training data as shown in Fig 3.a. The gradients computed at each GPU are passed to the parameter server which calculate its summation that is sent to all GPUs to be sure that the model is consistent. This process is called weight synchronization.
- Model parallelism:
  In this approach, model is split to parts which are distributed to the GPUs. Each part is trained on different GPU. Each GPU updates the weights of its assigned model part as shown in Fig 3.b. Layers outputs and weight gradients are transferred among GPUs.

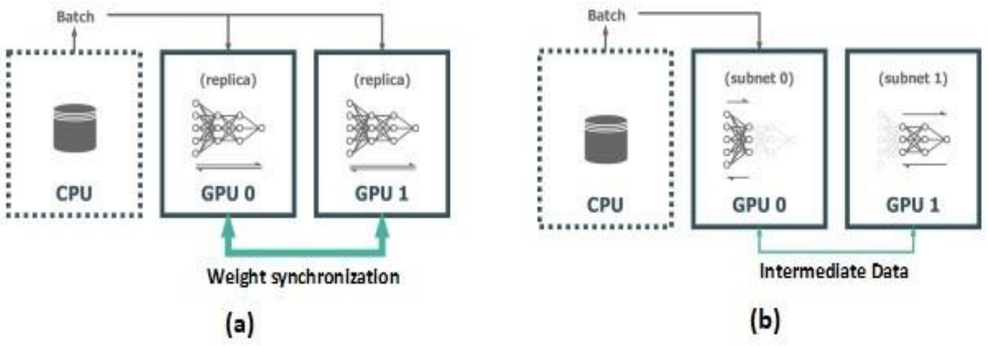

Figure 3. (a)Training process on data parallelism method in [7] using two GPUs. Each GPU is able train the model on a different set of training data. (b) Training process on model parallelism method in [19] using two GPUs. Each GPU is responsible for a part of the model.

3.4. workflow scenario of the proposed approach

Two GPUs are used to parallelize the IGNCNN image steganalysis approach. Based on the approach mentioned in subsection 3.2, a cyclic learning rate is used. About 100 different learning rates are tested to find the range of learning rates at which the loss decreases fast. A well-chosen cyclic learning rate helps the CNN to converge faster and improve its detection accuracy as well.
There are two kinds of layers in CNN: i) convolutional layers and ii) fully connected layers. Most of computations (about 90-95%) are computed by convolutional layers while Fully connected layers have high neuron activity (most of parameters) (about 95%). Based on that explanation, the proposed approach uses data parallelism with the convolutional layers and model parallelism with the fully connected ones instead of using data parallelism with all of them.

In [1], IGNCNN is trained using one GPU. In this paper IGNCNN is parallelized using two Tesla P100 GPUs as shown in Fig 4 in order to fairly compare the results. The proposed approach is implemented as follows: first, a batch of size 128 is assigned to each GPU in order to extract the features. Then the first GPU broadcast its feature maps to the second. The classification step is performed using the two GPUs on the feature maps of the first GPU which means that model parallelism is applied in the classification step. In order to avoid any of the GPUs to be idle pipelining is adopted by injecting the second batch to the model concurrently. After calculating the loss using feature maps of the first GPU, partial backpropagation is performed in order to update the parameters of the fully connected layers. The previous step is repeated for the second GPU feature maps. Finally, the backpropagation is performed to the convolutional layers and the CNN model is ready to be trained the rest batches. These steps are shown in Fig 5.

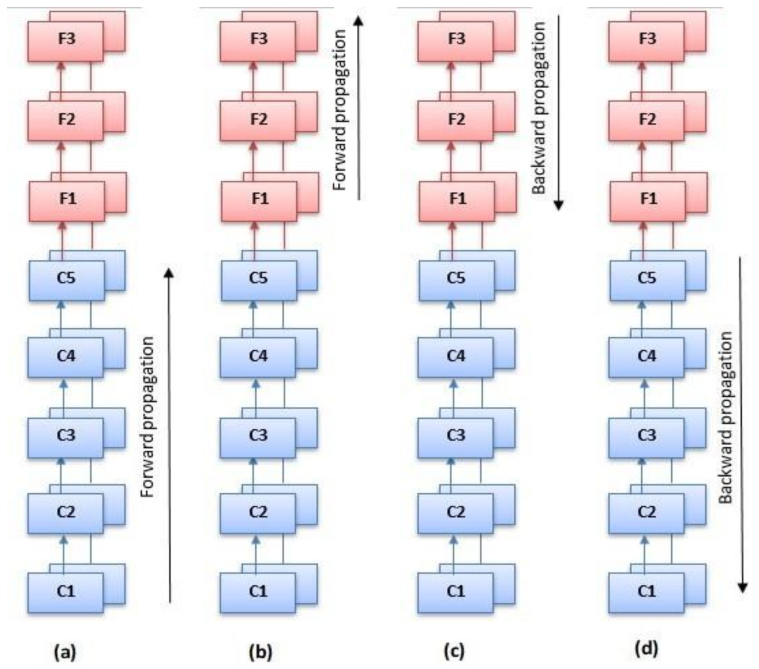

Figure 4. Illustration of the forward and backward propagation of IGNCNN [1] using two GPUs. The five convolutional layers and the three fully-connected layers are represented with blue and red rectangles respectively. Forward propagation for convolutional and fully-connected layers is shown in a and b while their backward propagation is shown in c and d, respectively.

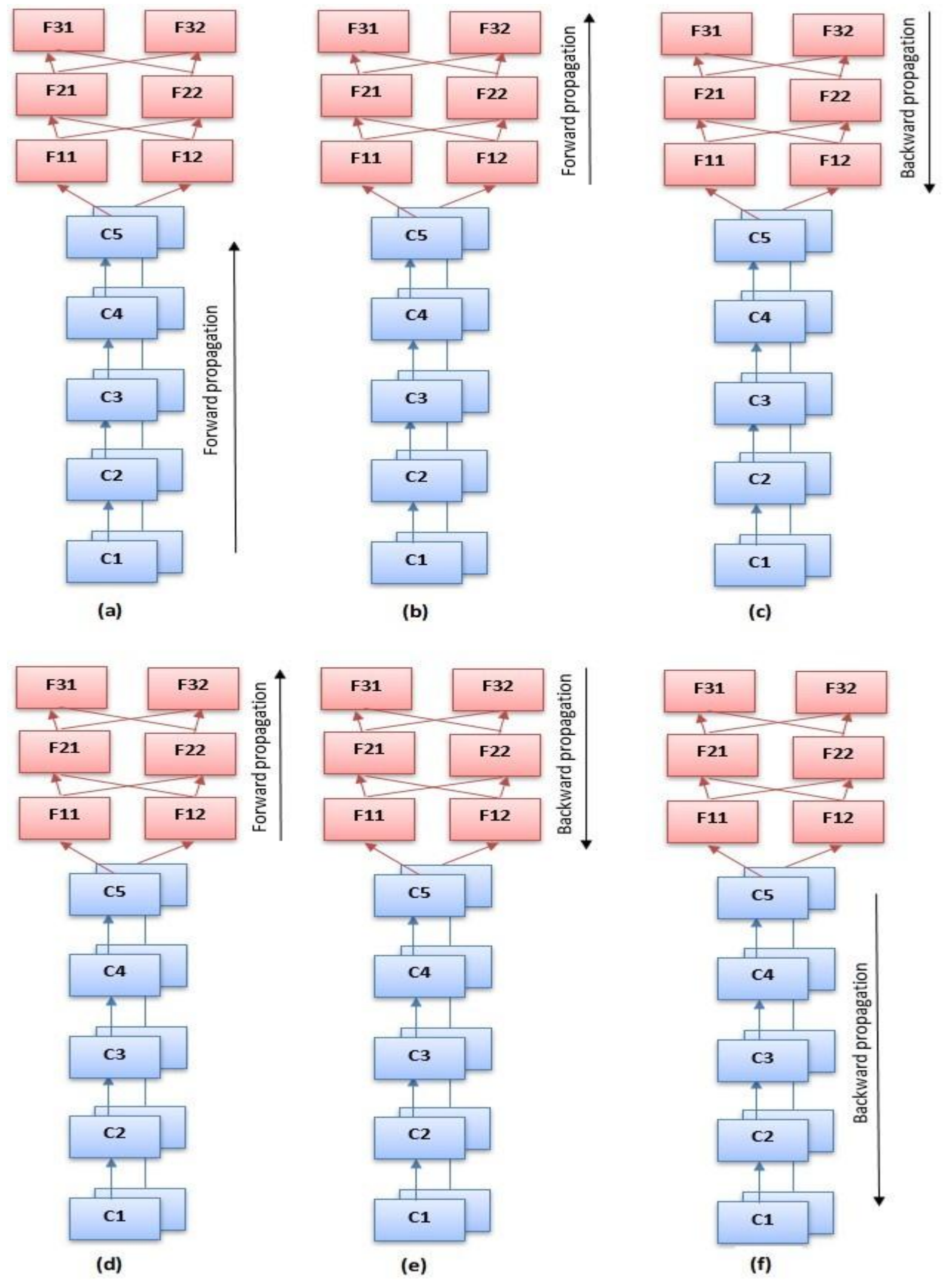

Figure 5. Illustration of the forward and backward propagations of the proposed approach using two GPUs. The five convolutional layers and the three fully-connected layers are represented with blue and red rectangles respectively. Two-way data parallelism in the five convolutional layers is represented with layers stacked on top of one another, while two-way model parallelism in the three fully-connected layers is represented with layers laid out next to one another. The standard two passes are replaced here with six passes (a, b, c, d, e and f).

4. Experiments and Discussion

In this section, first, dataset description is presented in subsection 4.1. Then, the implementation setup is mentioned in subsection 4.2. Finally, results and discussion are presented in subsection 4.3.

4.1. Dataset Description

The standard BOSSbase 1.01 image dataset [20] is used for all experiments. It is consisted of 10000 gray-level images each of size 512 × 512. Seven cameras are used to acquire these images. The dataset images are exposed to WOW steganographic algorithm [21] with payloads of 0.3 and 0.4 bpp (bits per pixel). Three experiments are performed. In the first experiment, the dataset images are resized to the size of 256 × 256 pixels. In the second experiment, in order to represent the whole image, 10 patches of size 256 × 256 are extracted from each 512 × 512 image. These 10 patches are the center patch, the four corners and their flip version. In the third experiment, dataset images are cropped to generate four patches of size 256 × 256. In each experiment, image patches are fed to the CNN to predict its class. The predictions of the 256 × 256 patches are averaged to estimate the original image class. For all the experiments, dataset is split as follows, 70% to a training set, 10% to a validation set, and 20% to a test set, respectively. The CNN pre-training is performed on pairs of images. Each pair contains non-stego and stego-image with payload 0.4 bpp. Like the pre-training, the fine-tuning is performed on pairs of images. But each pair contains non-stego and stego-image with payload 0.3 bpp.

4.2. Implementation Setup

Experiments are performed on device with two Intel Xeon Silver, 128 GB RAM and two Tesla P100 GPUs each with 3,584 CUDA cores. Convolutional layers weight decay is 0 and 0.01 for the fully connected layers. Momentum of 0.9 and mini batch size of 128 are used. Three experiments are performed as mentioned in subsection 4.1 where each experiment is performed twice. The first time it performed using a fixed learning rate of 0.001 while the second time is performed using A cyclic learning rate starts at 0.01 and ends at 0.00001.

4.3. Results and Discussion

The proposed approach using a fixed learning rate is referred to as *Proposed 1*. The proposed approach using a cyclic learning rate is referred to as *Proposed 2*. Results of both proposed approaches are compared to the improved GNCNN in [1], which is denoted as (IGNCNN).

- The first experiment is done by training both proposed approaches and IGNCNN on the dataset images resized to a size of 256 × 256 pixels. Results of this experiment is shown in Table 1.
- The second experiment is done by training the proposed approaches and IGNCNN on the dataset images split to ten 256 × 256 pixels patches. Results of this experiment is shown in Table 2.
- The third experiment is done by training both proposed approaches and IGNCNN on the dataset images cropped to four 256 × 256 pixels patches. Results of this experiment is shown in Table 3.

Results of the three experiments show that the cyclic learning rate (*Proposed 2*) helps the CNN to converge faster. This fast convergence leads to minimizing the training time by an average of 17% and 28.7% when compared with fixed learning rate (*Proposed 1*) and IGNCNN respectively. Results also show that splitting the dataset to ten patches of size 256 × 256 (Experiment 2) takes about 1 day to finish the training process which is the longest training time when compared with experiment 1 and experiment 3. Whoever, Experiment 3 improves the detection accuracy by an average increase of 2% when compared with experiment 1 and experiment 3.

Table 1. time of training CNN-based steganalysis approach on WOW 512 × 512 stego-images with payload of 0.3 bpp with pre-training on WOW stego-images with payload of 0.4 bpp. each 512 × 512 image is resized to an image of size 256 × 256

	IGNCNN[1]	*Proposed*1	*Proposed*2
Training time of 256 × 256 image	0.59 sec	0.51 sec	**0.42 sec**
Pre-training time on dataset (payload = 0.4)	1.76 h	1.51 h	**1.26 h**
Fine-tuning time on dataset (payload = 0.3)	1.52 h	1.31 h	**1.09 h**
Total time	3.28 h	2.82 h	**2.35 h**

Table 2. time of training CNN-based steganalysis approach on WOW 512 × 512 stego-images with payload of 0.3 bpp with pre-training on WOW stego-images with payload of 0.4 bpp. each 512 × 512 image is split to 10 images of size 256 × 256

	IGNCNN[1]	*Proposed 1*	*Proposed 2*
Training time of a 256 × 256 image	0.59 sec	0.51 sec	**0.42 sec**
Training time of a 512 × 512 image	5.91 sec	5.06 sec	**4.22 sec**
Pre-training time on dataset (payload = 0.4)	17.06 h	14.616 h	**12.18 h**
Fine-tuning time on dataset (payload = 0.3)	15.77 h	13.44 h	**11.20 h**
Total training time	32.83 h	28.14 h	**23.45 h**

Table 3. time of training CNN-based steganalysis approach on WOW 512 × 512 stego-images with payload of 0.3 bpp with pre-training on WOW stego-images with payload of 0.4 bpp. each 512 × 512 image is cropped to four images of size 256 × 256

	IGNCNN[1]	*Proposed*1	*Proposed*2
Training time of 256 × 256 image	0.59 sec	0.51 sec	**0.42 sec**
Training time of 512 × 512 image	2.36 sec	2.03 sec	**1.69 sec**
Pre-training time on dataset (payload = 0.4)	6.82 h	5.86 h	**4.88 h**
Fine-tuning time on dataset (payload = 0.3)	6.33 h	5.41 h	**4.50 h**
Total training time	13.15 h	11.27 h	**9.38 h**

The comparison between the proposed approach and the IGNCNN in [1] shows that using model parallelism with the fully-connected layers and data parallelism with the convolutional decreasing the training time of the CNN used. the cyclic learning rate as well has an important role in minimizing the detection error and helping the CNN model to converge to better minima faster than the fixed learning rate. The comparison also shows that the proposed approach achieves a better performance than IGNCNN [1]in terms of accuracy and time metrics

5. Conclusion

This paper presents an implementation method of a CNN-based blind image steganalysis approach on multiple GPUs, so as to quicken the training of this approach. Data parallelism is more effective when applied to the convolutional layers while model parallelism is more effective when applied to the fully-connected layers. The cyclic learning rate helps CNN model to converge to better minima faster. It also affects the accuracy positively. However, this cyclic learning rate have to be well chosen to avoid being stuck on a plateau region. Results show that the presented approach outperforms competing approach IGNCNN [1] by an average improvement of 28.72% in training time regardless the whole image representation method (resizing, cropping or splitting). the presented approach provides better performance as compared to the improved GNCNN implementation method with an average speed up factor of 1.4X and with an average improvement of 4% in detection error.

References

[1] Qian Y, Dong J, Wang W and Tan T 2016 *IEEE Intern. Conf. on Image Processing* pp 2752–2756
[2] Kodovsky` J, Fridrich J J and Holub V 2012 *IEEE Trans. Information Forensics and Security* 432–444
[3] Qian Y, Dong J, Wang W and Tan T 2015 *Media Watermarking, Security, and Forensics* vol 9409 pp 94090J–94090J
[4] Pibre L, Pasquet J, Ienco D and Chaumont M 2016 *Electronic Imaging* 2016 1–11
[5] Salomon M, Couturier R, Guyeux C, Couchot J F and Bahi J M 2017 *European Research in Telemedicine* **6** 79–92
[6] Xu G, Wu H Z and Shi Y Q 2016 *IEEE Signal Processing Letters* **23** 708–712
[7] Krizhevsky A 2014 *Computing Research Repository*
[8] Fridrich J and Kodovsky J 2012 *IEEE Trans. on Information Forensics and Security* **7** 868–882
[9] Zong H, Liu F l and Luo X y 2012 *Digital Investigation* **9** 58–68
[10] Ghanbari S, Keshtegary M, Ghanbari N, Branch Z and Azad I 2012
[11] Verma A K 2014 *International Journal of Multidisciplinary Consortium* **1**
[12] Tang W, Li H, Luo W and Huang J 2016 *IEEE Trans. on Information Forensics and Security* **11** 734–745
[13] Recht B, Re C, Wright S and Niu F 2011 *Advances in neural information processing systems* pp 693–701
[14] Dean J, Corrado G, Monga R, Chen K, Devin M, Mao M, Senior A, Tucker P, Yang K, Le Q V *et al.* 2012 *Advances in neural information processing systems* pp 1223–1231
[15] Oyama Y, Nomura A, Sato I, Nishimura H, Tamatsu Y and Matsuoka S 2016 *IEEE Intern. Conf. on Big Data (Big Data)* pp 66–75
[16] Coates A, Huval B, Wang T, Wu D, Catanzaro B and Andrew N 2013 *Intern. Conf. on Machine Learning* pp 1337–1345
[17] Zhou J, Chen W, Peng G, Xiao H, Wang H and Chen Z 2017 *Intern. Conf. on Progress in Informatics and Computing (PIC)* pp 71–76
[18] Smith L N 2017 *IEEE Winter Conf. on Applications of Computer Vision* pp 464–472
[19] Harlap A, Narayanan D, Phanishayee A, Seshadri V, Devanur N, Ganger G and Gibbons P 2018 *arXiv preprint arXiv:1806.03377*
[20] Bas P, Filler T and Pevny´ T 2011 *Information Hiding* pp 59–70
[21] Holub V and Fridrich J 2012 *IEEE Intern. Workshop on Information Forensics and Security* pp 234–239