ICIC Express Letters
Part B: Applications
Volume **13**, Number **9**, September **2022**

ICIC International ©2022 ISSN 2185-2766
pp. 991–1000

MEMORY AND RESOURCE MANAGEMENT FOR MOBILE PLATFORM IN HIGH PERFORMANCE COMPUTATION USING DEEP LEARNING

Dipak Raghunath Patil[1,*], Mukesh Kumar Gupta[2], Manish Sharma[3] and Malay Banerjee[4]

[1]Department of Computer Engineering
Amrutvahini College of Engineering Sangamner
S.P. Pune University
University Road, Ganeshkhind, Pune, Pune, Maharashtra 411007, India
*Corresponding author: dipak.patil@avcoe.org; patildipak87@gmail.com

[2]Department of Electrical Engineering
[4]Chair Professor
Suresh Gyan Vihar University
Gyan Vihar Marg, Jagatpura, Jaipur, Rajasthan 302017, India
{ mukeshkr.gupta; malay.banerjee }@mygyanvihar.com

[3]Department of Computer Science and Engineering
Jaipur National University
Jaipur-Agra Bypass Near New RTO Office, Jagatpura, Jaipur 302017, India
director.naac@jnujaipur.ac.in

Received February 2022; accepted May 2022

Abstract. *Since the last decade, memory management systems have been a concern in High-Performance Computing (HPC) and cloud computing. The number of supported workloads in HPC and Clouds environments causes the variation in-memory optimization methods. Due to the heterogeneity of memory and resource locality, data migration in the HPC becomes extremely crucial. In addition to the memory usage on cloud, eli-minate the unutilized objects as well as transmission cost is reduced. Nevertheless, the migration of memory over the network is a very expensive and time-consuming process. This paper describes, an efficient workload and memory management for mobile platforms using deep learning algorithms is described. The Deep Convolutional Neural Network (DCNN) has been carried out to improve the Quality of Service (QoS) parameters during the execution. A most apparent target circumstance of the HPC is the batch processing of a particular high-performance application through input data of the same size, with a greater degree of heterogeneity and productivity. In particular, HPC pays attention to speeding due to the memory system's technological modernization and cache. However, in the other direction, in Clouds, the focus is on maximizing resources' availability and allocation. The proposed algorithm reduces the network overhead by minimizing unnecessary data load from input file systems, and it has been selected according to the CNN training module. In experimental analysis, the system reflects the reduction of execution cost around 14% over the traditional data transmission CNN-TensorFlow. RESNET and VGGNet framework have been used to investigate the efficiency of the proposed system. RESNET demonstrates better results over VGGNet for both training and testing. The proposed RESNET provides **99.69% accuracy**, which is higher than other deep learning frameworks such as GoogleNet, and CaffeNet. It also produces a **low error of 0.31%**, which is lower than another deep learning framework.*

Keywords: Memory management, HPC computing, TensorFlow, Deep convolutional neural network

DOI: 10.24507/icicelb.13.09.991

1. **Introduction.**

In today's era mobile edge registering is relied upon a million-dollar industry with big business organizations arriving at $73 millions. The rise of organization complexity frameworks is due to the growth of on-demand and adjustable services. Network access suppliers should oblige traffic for web perusing, associated vehicles, real-time videos, web-based gaming, voice over IP and consistently on Internet of Things (IoT) gadget transmissions [1]. New imperatives presented by on-request benefits as recorded above require an extreme change of fixed and portable access organizations. There are significant benefits to deep learning across conventional machine learning. Initially, when data volumes are large, deep learning will deliver better efficiency. This implies that artificial intelligence will take full advantage of the enormous amount of IoT data obtained. Where data volumes are limited, conventional data mining techniques are superior. Even so, when storage capacity is extremely high, the output dramatically degrades. In comparison, through large data, deep learning demonstrates beneficial interoperability. Second, deep learning is less dependent on functional technology [2]. Heterogeneous file systems may accumulate differentiated knowledge categories that are spatial in nature. It is a challenging task to manually extract characteristics of heterogeneous files. To extract the features, conventional data found automatically expose architectures to hidden meanings. Fortunately, in a layer-wise way, deep learning independently extracts the features to describe input sequence with a hierarchical awareness campaign. Based on lower-level functions derived from previous layer, each layer defines greater features. Deep learning with convolutional hidden layers is one effective technique for maximizing the data. This pre-allocates a continuous chunk of memory and assumes care of operating system memory management. There are numerous variations in the implementation of the memory extraction and offload, to achieve small time complexity and competitive ratio [3]. This does not provide optimization for deep learning instruction, however. This research focuses on optimizing the allocation of memory within the GPU memory pool by leveraging the lifetime and size information of variables to achieve a better balance with low time complexity. Following are the key contributions of this work.

- This work is applied to various mobile platforms' heterogeneous file systems, which provides dynamic load balancing.
- RESNET-100 deep learning framework has been used to implement CNN with 100 convolutional layers with numerous optimization functions that provide effective memory compression.
- Resource matchmaking is another novelty of this work; according to the job size system automatically selects the virtual machine for data storage.

The paper is organized as follows. Section 2 describes the related work. In Section 3 system architecture is presented along with aim and modules. Results and discussion are explained in Section 4. Finally, the conclusion is presented in Section 5.

2. **Related Work.**

The state-of-the-art survey on deep learning and machine learning in mobile computing with automatic resource allocation, energy efficiency, and memory management for mobile environments is presented in this section. A concern of optimizing the benefit of cloudlets management solution that accepts computing orders from smartphone subscribes and fulfils these requests by exploiting active computing services in different frameworks [4]. Due to the uncertain advent of different mobile cloud user's active involvement and difficulty in the computational allocation of resources, optimizing such a management platform and given operating benefit is very difficult. The model does not require any descriptive statistics of the related control system, unlike the existing method, and is effective for execution in optimization and allocation practices. A new supervised learning method is applied to dealing with resource allocation supported by cloud computing. The novel approach [5] proposed to build a various deep learning frameworks implementation to reduce the memory consumption and network overhead during data transmission. Deep Learning (DL) methodology proposed as the constituent technique of machine learning can be incorporated into mobile computing methodologies to construct perceptive edge for dynamic, most straightforward task management and maintenance [6].

The implementation shows how to exploit computer vision techniques to increase hardware design and optimize computational complexity. For collaborative training, the machine learning system MXNet (or "mix-net") [7] is used. According to its low memory size, MXNet is preferably added to edge computers. The autonomous deep Q-learning approach [8] greatly increases the efficiency of computational offloading by minimizing service computational latency [9]. To access its enormous computing power, mobile fog interplays with conventional cloud. This segment explores the work on computing unloading in the cloud infrastructure. A complex allocation and deep classification algorithms focused on autonomous control of virtual resources depending on their ratio of minimum allocated resources, confidence and achieve reserved unused resources from the Virtual Networks (VNs). Instead, utilizing reinforcement learning algorithm based on the total service quality efficiency or resource usage of their subscribers, the VNs independently monitor their resource allocation [10].

A dynamic resource allocation model composed of mobile devices, cloudlets and public cloud, refers to as hybrid mobile cloud computing to schedule data-intensive applications in an integrated computing resource environment. Offloading heavy tasks to the cloud will efficiently decrease energy consumption [11]. On cloud infrastructure, dynamic resource allocation and memory for different frameworks the programmed asset and memory management was present, which is an important stage that empowers deep portable asset designation in frameworks [12]. From the literature review it is seen that there are still some flaws in those systems, such as high-power consumption during data transmission, heavy network overhead due to massive data selection, high memory consumption, and high time complexity due to single process overrun. Hence in this work, an efficient workload and memory management for mobile platforms using deep learning algorithms is described. The DCNN has been carried out to improve the Quality of Service (QoS) parameters during the execution.

3. System Architecture.

The aim of this work is to design and develop an approach for effective memory management for mobile technology with resource utilization using deep learning approach. An automatic memory management, effective resource utilization, reducing network overhead during data offloading between local devices to data server using deep learning algorithms is described.

Figure 1 illustrates three modules as described below.

1) *TensorFlow Library Module:* In the first module the interface is developed and customized on TensorFlow. It provide the access interfaces and customized for every deep learning tool.
2) *Data Processing Module:* This is the middleware module of system which maintains available resources. This module executes processing tasks with free memory space, and synchronizes deep learning process from the combined model to reach the target of code execution on system on chip. The memory distracter will automatically check memory request from application and provide available space in memory. That improves system efficiency for both private as well as public cloud environments.
3) *Cloud Storage Module:* This is used to train deep NN models, store the trained results, and all file dataset in the training processes. For the proposed work NVIDIA is used for deployment which has GTX 1080 GPUs.

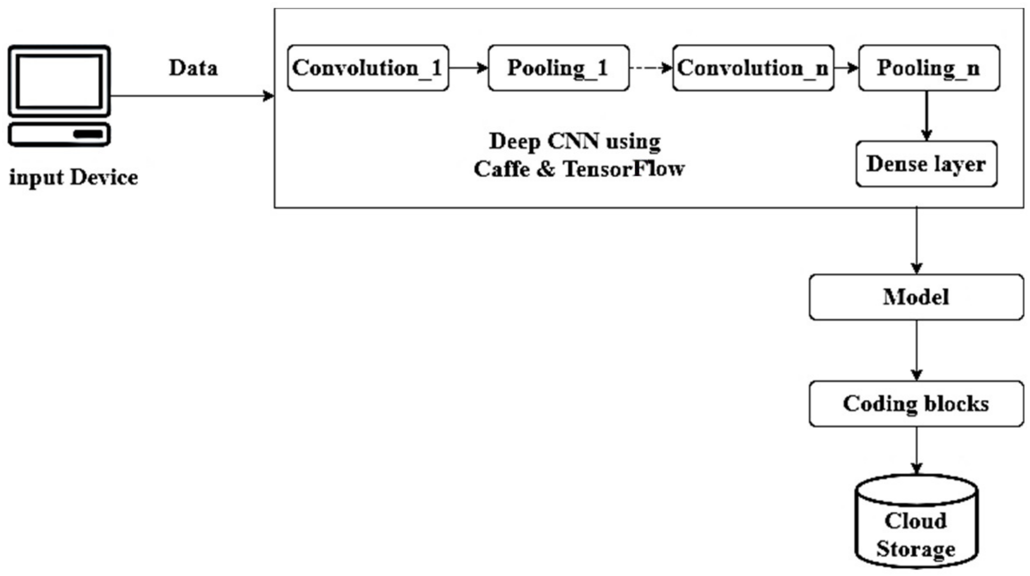

Figure 1. System architecture

Using this research work, the problems and limitation encountered earlier as enlisted below are solved.

- High network overhead during the data transmission which leads to increased time complexity as well as error rate.
- High level hardware resource dependency.
- Accuracy issues for object or various file system detection.
- Too much response time required to process the existing data.

4. Results and Discussion.

4.1. Code implementation.

In the experimental analysis various deep learning frameworks with inbuilt libraries are used. The entire system carries python implementation with around 1500 Line of Codes (LoC), including the libraries for TensorFlow. All implementation work is done on NVIDIA GFORCE GTX (TITAN) with 12GB GDDR5X, Intel(R) Xeon ES 1607V 2.3GHz with 16GB DDR3-1866 ECC.

4.2. Dataset and deep models.

During implementation phase mobile files system including heterogeneous dataset is used, including .txt files, log files, image objects and some supportive files. Flicker is another dataset used for processing large image objects in execution. For deep framework, some CNN models like AlexNet, VGGNet, GoogleNet, RESNET, CaffeNet are selected.

4.3. Memory optimization.

Figures 2-6 represent the variation of size of the blocks in MB with number of blocks where block is smallest amount of data retrieved.

Figure 2 depicts the per-iteration time approximately stays consistent as depth extensions, around 30% over regular iteration cost. It presents time of written repetition on epoch size 100 and batch size is 1280. In this phase TensorFlow is used for entire data processing that effectively reduces the optimization time. Figure 3 describes time and memory required for execution when on epoch size 100 and batch size is 4096.

Figure 4 demonstrates the memory and cost consumption using GPU execution with TensorFlow. It reduces the memory consumption around 35%, while 40% communication cost more than regular execution. It takes default epoch size as well as batch size as 1280.

Figure 5 demonstrates the memory and cost consumption using GPU execution with TensorFlow. It reduces the memory consumption around 30%, while 35% communication cost more than regular execution. It takes default epoch size as well as batch size as 4096.

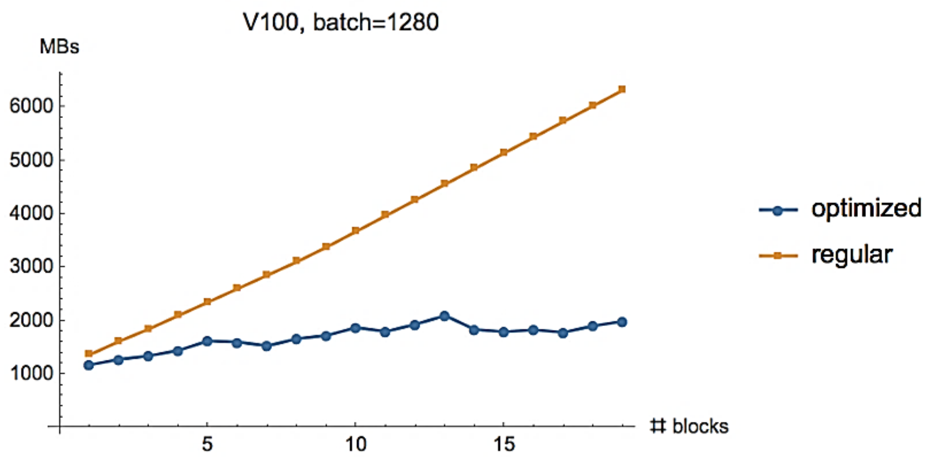

FIGURE 2. Iteration time per memory optimization

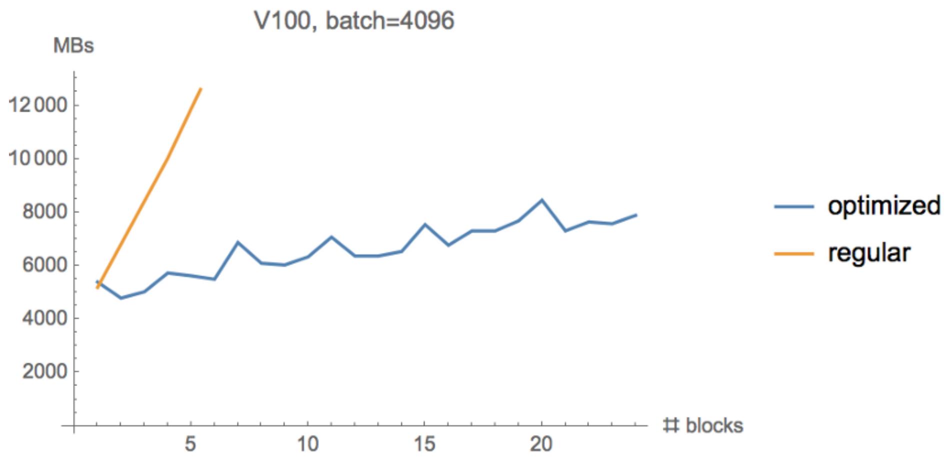

FIGURE 3. Time evaluation per memory size when batch size is 4096

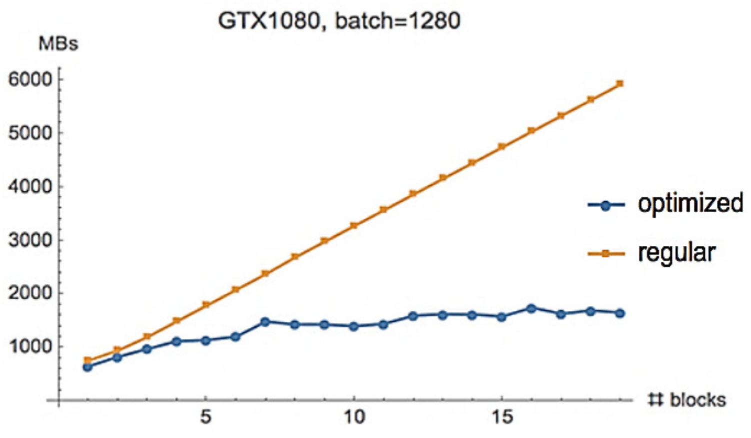

FIGURE 4. Memory and time evaluation when batch size is 1280 with GPU

Using the resources in this kit, one can trade off most of this memory used with computation to allow data work into storage more easily. For feed-forward prototypes, it was hard to squeeze more than 10x new designs into our GPU, with just a 20 percent improvement in computing time.

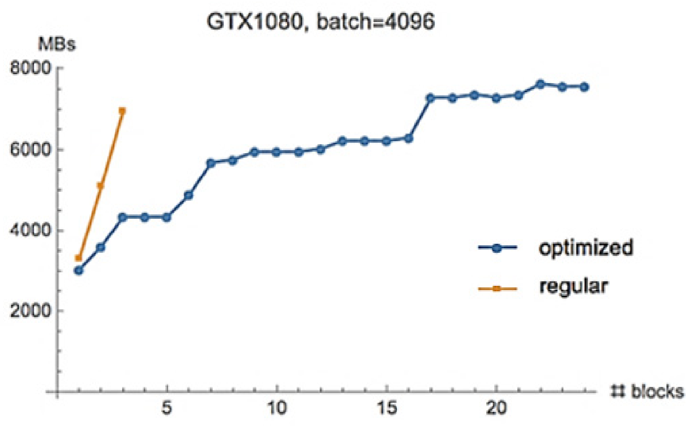

FIGURE 5. Memory and time evaluation with batch size is 4096 with GPU

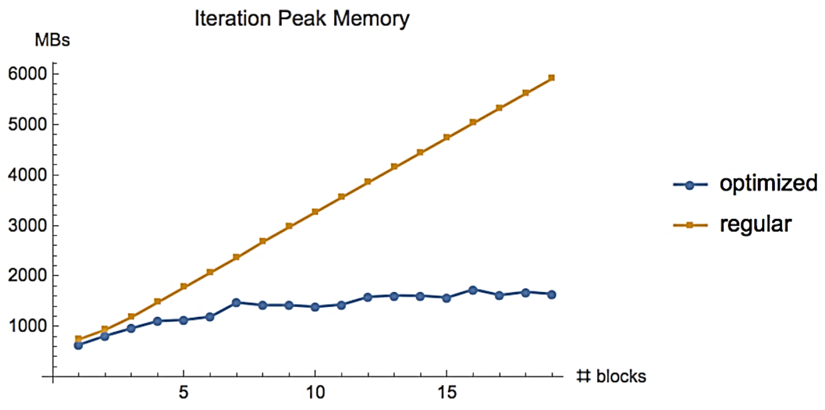

FIGURE 6. Testing memory usage and running time for RESNET with different numbers of layers. Batch size 1280, GTX1080.

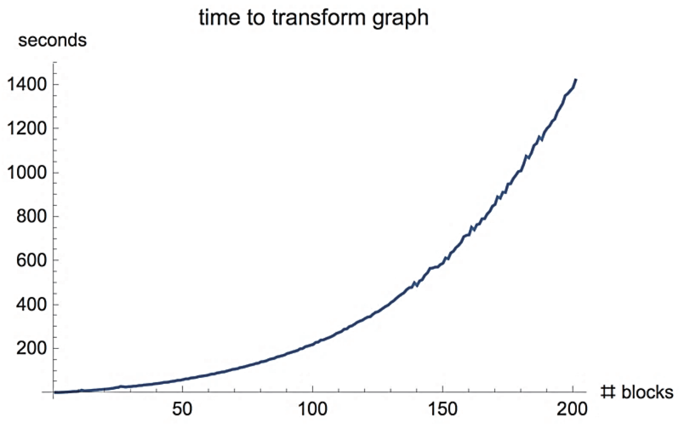

FIGURE 7. Number of data compiling time by system using TensorFlow

The memory intensive aspect of training deep neural networks is to calculate the regression of the failure through linear regression. It is feasible to compute the gradient at minimal cognitive cost by clustering nodes in the computation graph defined by our model and re-computing the portions of the graph in between other nodes during back-propagation. When teaching deep bite neural networks composed of n layers, one can minimize the memory consumption to O(sqrt(n)) in this manner, at the expense of one incremental forward move. This repository includes an application of this feature in TensorFlow, using the TensorFlow graph editor to dynamically rewrite the calculation graphs of the backward pass. Figure 7 depicts the time to compile the graph can be slow. In particular, RESNET with 200 blocks takes 30 mins to compile.

Figure 8 describes how time should be increased, when data load has enlarged for various deep models. Figure 9 shows how data processing time varies for various deep models without TensorFlow. It sometimes depends on current heterogeneous configuration and CPU process utilization.

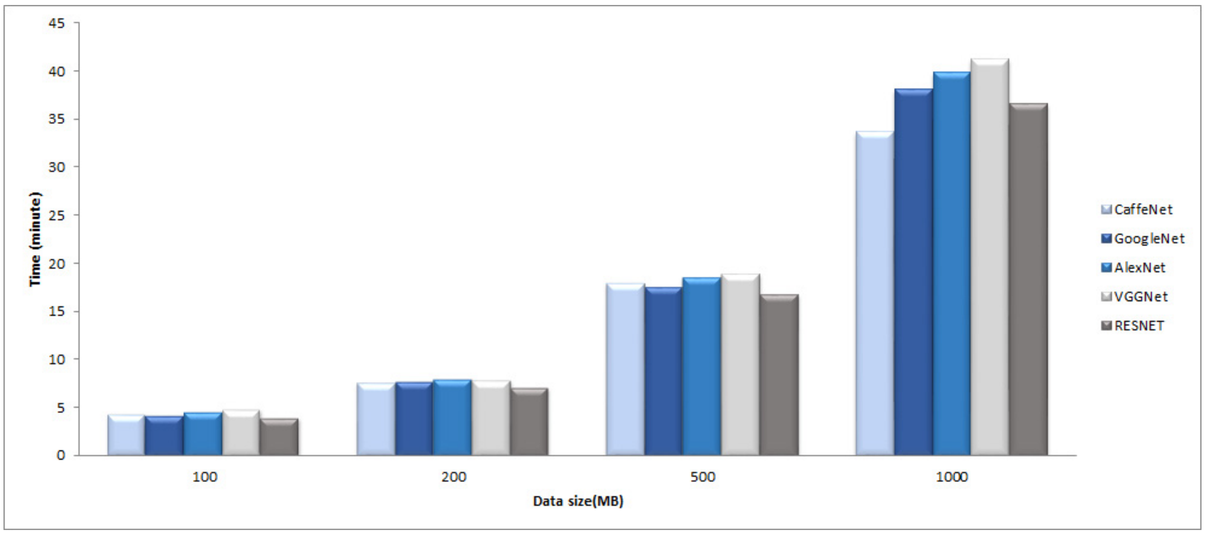

FIGURE 8. Data processing time for various deep models using TensorFlow

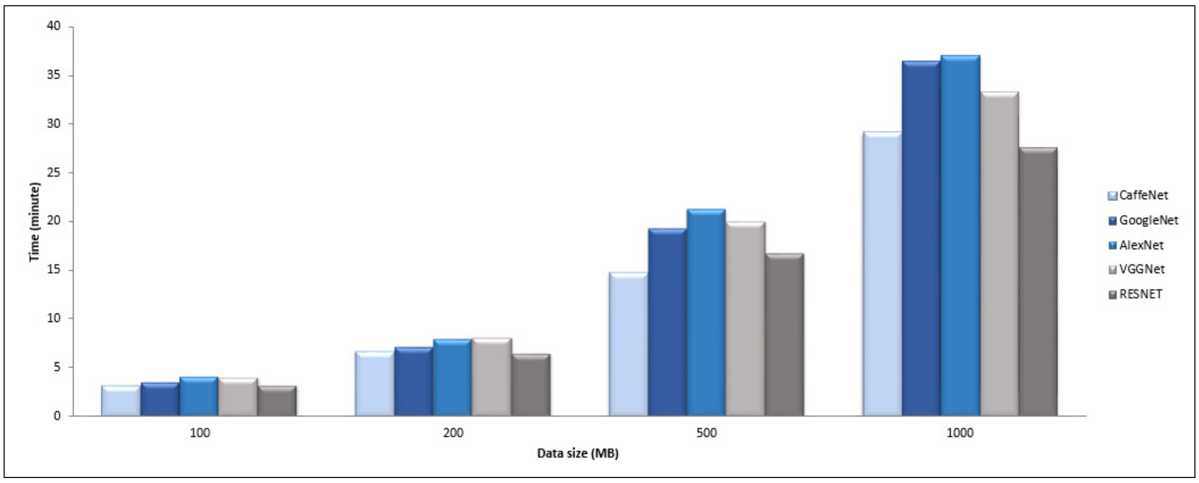

FIGURE 9. Data processing time for various deep models without TensorFlow

4.4. **Memory management.** The RESENT (32, 50, 101 and 152) version has been used for experimental investigation of the proposed system including 4G network. The major factors are execution time (including data processing, data uploading and downloading, etc.), memory consumption shown in Figure 10 to Figure 14.

Figure 10 demonstrates a default time reduction by the proposed methodology after using all deep modelling with TensorFlow. Based on that experiment it is concluded that RESNET can reduce around 30% time than default execution time.

Figure 11 illustrated the data uploading time required for five different deep learning frameworks by using TensorFlow. The RESNET provides slightly effective results than other deep learning frameworks. It provides around 20% effective results more than others. The compressive efficiency evaluation has been done on mobile computing file systems that is initially considered as input of system.

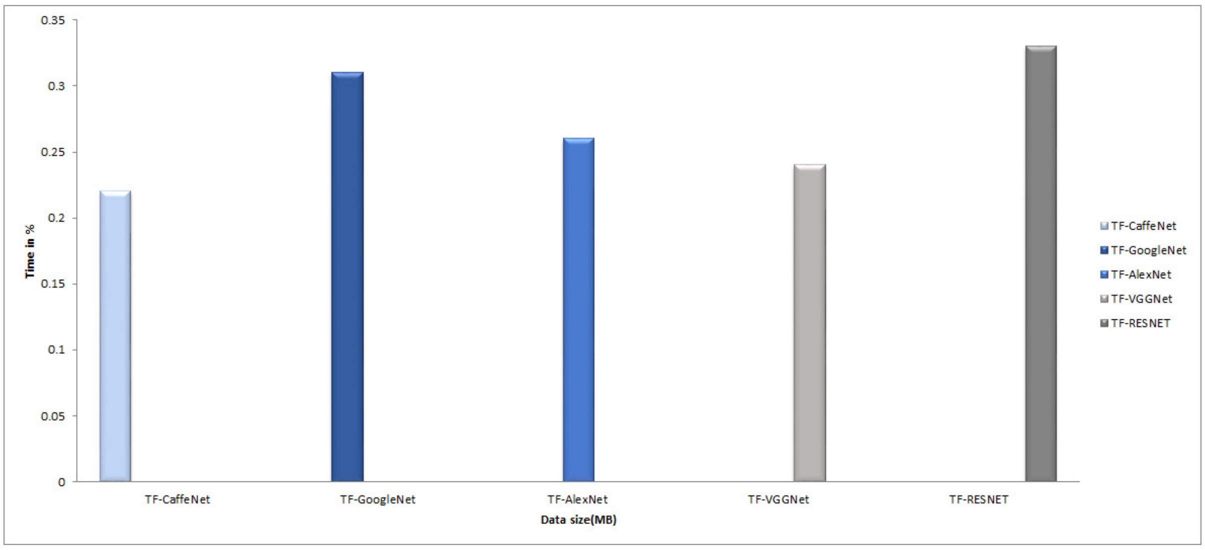

FIGURE 10. Default time reduction of deep learning model using TensorFlow

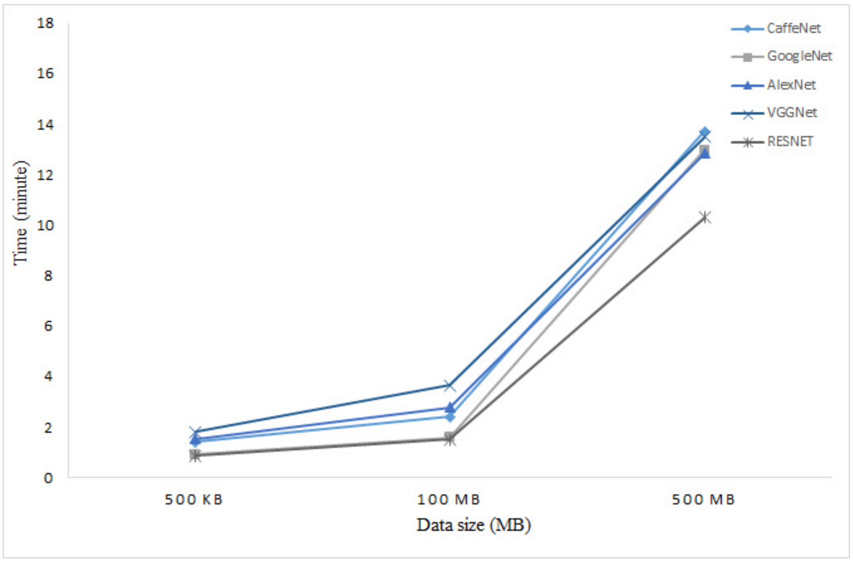

FIGURE 11. Data uploading TensorFlow with different deep learning models

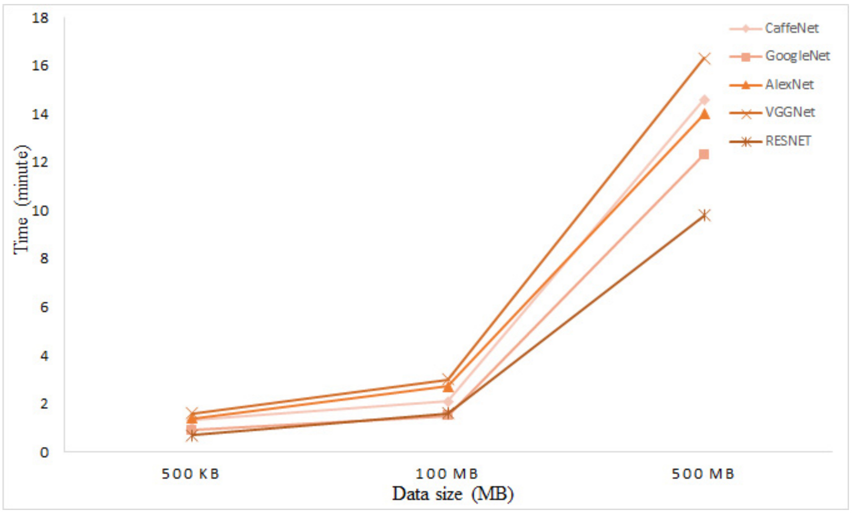

FIGURE 12. Data downloading TensorFlow with different deep learning models

Figure 12 describes the data downloading time required for five different deep learning frameworks by using TensorFlow. The RESNET provides slightly effective results than other deep learning frameworks. It provides almost 16.5% effective results more than others. The overall execution demonstrates success of system with heterogeneous data type or file system including various deep learning frameworks. The entire process could take large amount of cost for execution, and sometimes generates data dimensionality problem like data leakage.

Figure 13 demonstrates the data processing success rate with all frameworks. It shows RESNET gives higher accuracy over the other deep learning frameworks. Similarly, data reduction and data leakage are another problem that have occurred during execution. Figure 14 depicted the error rate of all frameworks using CNN. The data leakage has measured for all frameworks presented in Figure 14. It is observed that the CaffeNet produced high data leakage and RESNET and VGGNet generated low data leakage problem.

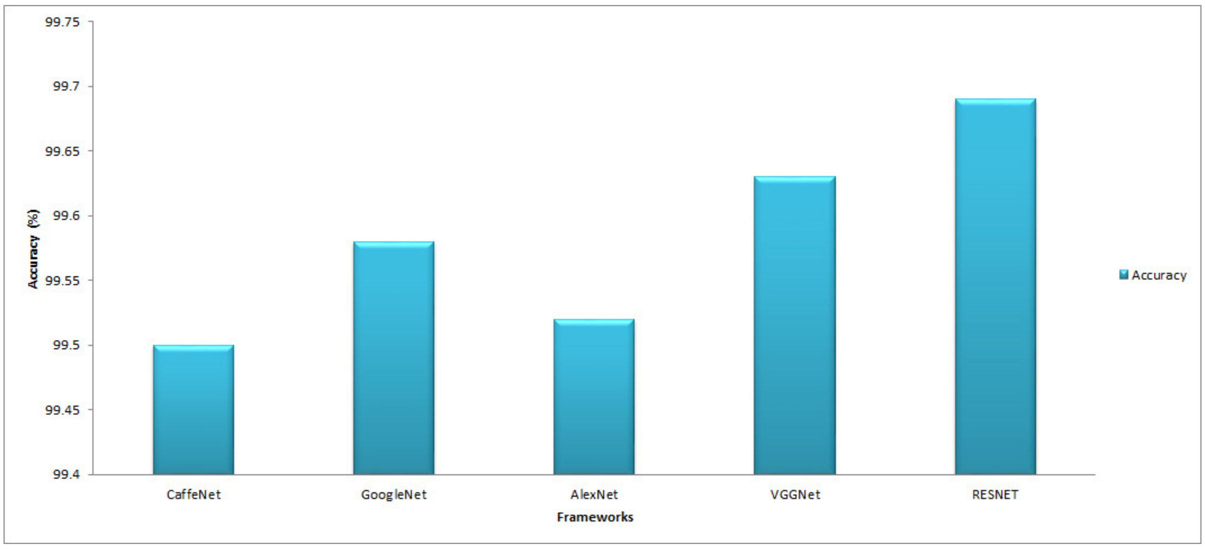

Figure 13. Accuracy for data processing with various deep learning frameworks

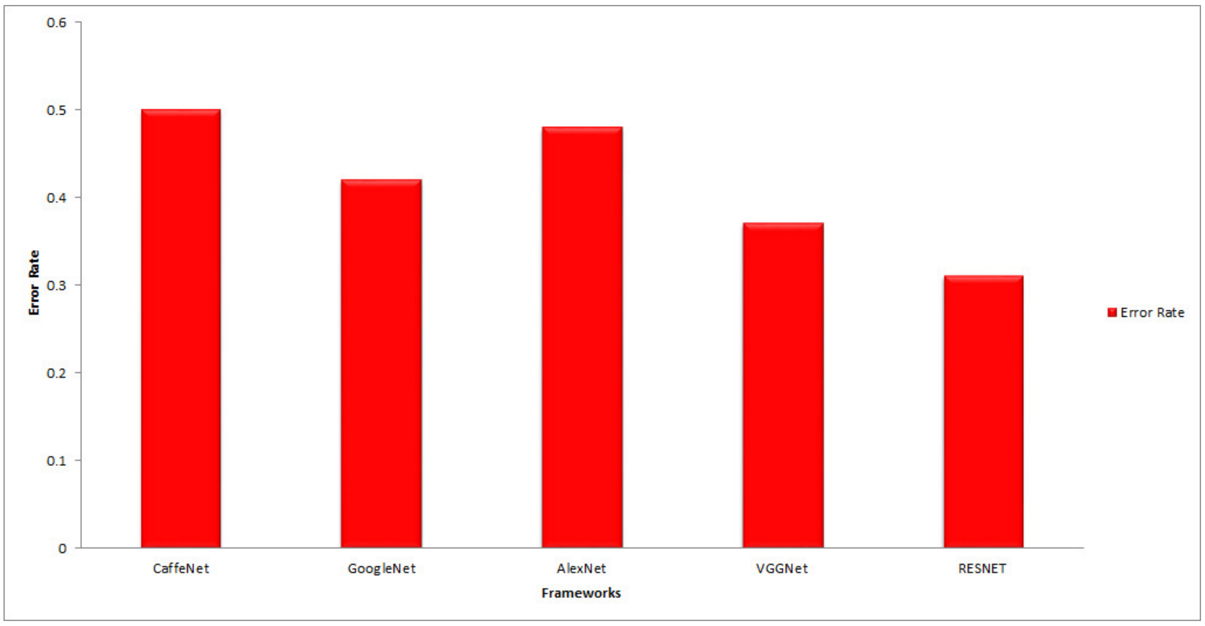

Figure 14. Error rate for data processing with various deep learning frameworks

5. **Conclusions.**

The experiments investigated the performance, good configurability and original deep learning process of the system by installing the application with TensorFlow frameworks. In this work the system's overheads are also assessed along with existing system constraints. According to the overall analysis, the RESNET provides **99.69% accuracy**, which is higher than other deep learning frameworks such as GoogleNet, and CaffeNet. It also produces a **low error of 0.31%**, which is lower than another deep learning framework. The overall extensive experiments show the proposed model can reduce almost **2.2% upload and download time using RESNET** with a similar dataset. For future direction, apply ensemble deep learning classifiers for reducing memory problem which occurs sometimes, when input data is complex or imbalance.

Acknowledgement. We would like to thank all the anonymous reviewers for their insightful comments and suggestions that helped us to improve the quality of this paper.

REFERENCES

[1] M. G. Alam, M. Hassan, M. Z. Uddin, A. S. Almogren and G. Fortino, Autonomic computation offloading in mobile edge for IoT applications, *Future Generation Computing System*, vol.90, pp.149-157, 2019.

[2] D. Frajberg, C. Bernaschina, C. Marone and P. Fraternali, Accelerating deep learning inference on mobile systems, in *Artificial Intelligence and Mobile Services – AIMS 2019. Lecture Notes in Computer Science*, D. Wang and L. J. Zhang (eds.), Cham, Springer, DOI: 10.1007/978-3-030-23367-9_9, 2019.

[3] Z. Ali, S. Khaf, Z. H. Abbas, G. Abbas, F. Muhammad and S. Kim, A deep learning approach for mobility-aware and energy-efficient resource allocation in MEC, *IEEE Access*, vol.8, pp.179530-179546, 2020.

[4] W. Fang, X. Yao, X. Zhao, J. Yin and N. Xiong, A stochastic control approach to maximize profit on service provisioning for mobile cloudlet platforms, *IEEE Trans. Systems, Man, and Cybernetics: Systems*, vol.48, no.4, pp.522-534, 2018.

[5] S. Goudarzi, M. H. Anisi, H. Ahmadi and A. L. Musavian, Dynamic resource allocation model for distribution operations using SDN, *IEEE Internet of Things Journal*, vol.8, no.2, pp.976-988, 2021.

[6] J. Zhang, J. Xiao, J. Wan, J. Yang, Y. Ren, H. Si, A. L. Zhou and H. Tu, A parallel strategy for convolutional neural network based on heterogeneous cluster for mobile information system, *Mobile Information System*, vol.2017, pp.1-12, 2017.

[7] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang and Z. Zhang, MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems, *arXiv.org*, arXiv: 1512.01274, 2015.

[8] T. Li, X. Zhu and X. Liu, An end-to-end network slicing algorithm based on deep Q-learning for 5G network, *IEEE Access*, vol.8, pp.122229-122240, 2020.

[9] L. Wang and Z. Yuan, Efficient task offloading strategy for low-energy base station groups in mobile edge computing, *International Journal of Innovative Computing, Information and Control*, vol.17, no.5, pp.1531-1548, DOI: 10.24507/ijicic.17.05.1531, 2021.

[10] Y. Cui, Y. Liang and R. Wang, Resource allocation algorithm with multi-platform intelligent offloading in D2D-enabled vehicular networks, *IEEE Access*, vol.7, pp.21246-21253, 2019.

[11] M. Alkhalaileh, R. Calheiros, Q. Nguyen and B. Javadi, Dynamic resource allocation in hybrid mobile cloud computing for data-intensive applications, in *Green, Pervasive, and Cloud Computing. GPC 2019. Lecture Notes in Computer Science*, R. Miani, L. Camargos, B. Zarpelão, E. Rosas and R. Pasquini (eds.), Cham, Springer, DOI: 10.1007/978-3-030-19223-5_13, 2019.

[12] D. Patil and M. Sharma, Dynamic resource allocation and memory management using machine learning for cloud environments, *International Journal of Advanced Trends in Computer Science and Engineering*, vol.9, no.4, pp.5921-5927, 2020.