IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 9, Issue 9, September 2020

DOI 10.17148/IJARCCE.2020.9902

A DNN-Based Application in Joint Mobile and Cloud Services Platform

Iing Muttakhiroh[1], Kazi Md Shahiduzzaman[2], Md. Rashed Ibn Nawab[3]

Student, School of Computer Science and Technology, Huazhong University of Science and Technology, China[1]

Assistant Professor, Department of Electrical and Electronic Engineering,

Jatiya Kabi Kazi Nazrul Islam University, Mymesingh, Bangladesh[2]

PhD Student, School of Computer Science and Technology, Northwestern Polytechnical University, China[3]

Abstract: Deep Neural Network (DNN) has been a great success in many areas recently. It has achieved an advanced performance in applications such as image classification, speech recognition, and time series forecasting. However, when the data is getting bigger, the performance computation on a single CPU becomes worse. One approach to tackle this challenge is to distribute the DNN workload into several machines. This research uses mobile cloud computing as a joint distributed environment to run the DNN application. An image recognition problem used as a test case will be conducted on both mobile and cloud platforms. Latency time and energy consumption measured in each DNN layer and used as the parameters to allocating efficiently the computational task of each layer in to suitable cores of mobile devices or cloud in Mobile Cloud Computing (MCC) environment. The joint platform can achieve improvements up to 73% in latency and 56% in energy consumption. As an addition, we apply lossless compression to reduce the influence of the communication between layer.

Keywords: Deep Neural Network, AlexNet, Image Recognition, Mobile Cloud Computing, Task Scheduling.

I. INTRODUCTION

Recently, Deep Neural Network (DNN) has been a great success in many areas, especially in computer vision, speech recognition, and time series forecasting. It has achieved an advanced performance, high accuracy and gives workable solutions [1][2][3]. Since it was proposing in 1998, DNN's topology evolution was improving quite rapidly [4][5]. There are some important components of DNN: 1) volume of data, 2) flexible model or topology, and 3) growing computing power. With the increasing number of parameters and training data, the DNN performance shall be improved dramatically. DNN is super computation [6]. It has many layers of computation and contains super huge data size and parameters. While so many researches built DNN application to solve actual problems, there are several types of research work on how to implement DNN the applications in real-world efficiently [7].

Further discussion related to DNN and exploration of advanced DNN has become a recent research trend. Almost 80% of research is related to improvising DNN performance [8]. Implementing DNN on end-user applications has spread rapidly. As we know that DNN is super computation, how to construct and implement DNN in end-user-applications is still facing significant challenges up until today. There are two common approaches to deploy DNN architecture in end-user-application, cloud only and mobile-only computation [9][10]. Running DNN only on a mobile platform is very challenging for some limitations related to its performance and battery. The same situation also happens if DNN is run on cloud infrastructure. Subscription cost, network dependency, and service congestions become critical issues. Recently, some researches were proposing distributed deep neural network over mobile and cloud services and the efficient way to implement the DNN inference and training mobile and cloud services platform [11]. However, several studies on implementing DNN computation over mobile cloud computing efficiently still face a substantial challenge up to date.

In another hand, Mobile Cloud Computing (MCC) has gradually grown up. In MCC environment, more data and computational task are increasingly produced, and it needs high-order graph to represent the complex relationship among them[12]. In[13], they construct a holistic optimization framework for mobile cloud task scheduling. They proposed a holistic model cloud optimization model including energy consumption, system reliability and Quality of Services (QoS). This research implemented AlexNet over mobile cloud computing environment. AlexNet is as one of the most favorable DNN topologies and very popular for image classification [14]. A test case will be conducted on a holistic optimization framework for mobile cloud task scheduling. They will measure latency time as the performance and energy consumption in each DNN layer and will allocating efficiently the computational task of each layer to suitable cores of mobile devices or cloud in Mobile Cloud Computing (MCC) environment. How to implement a DNN-based image recognition on a joint platform Mobile Cloud Computing (MCC) environment is our major focus. There are two things to explore in this

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 9, Issue 9, September 2020
DOI 10.17148/IJARCCE.2020.9902

research, First is how to construct a model for latency time and energy consumption for distributed an AlexNet topology in MCC environment then second thing is provide optimization formulations for allocating efficiently the computational task of each layer to compatible cores of mobile devices or cloud services in Mobile Cloud Computing (MCC) environment which can adapt to mobile battery limitations, cloud server load constraint and QoS.

The aims of this paper are as follows:

1. Change the holistic optimization framework for mobile cloud task scheduling to allocating efficiently every layer of DNN architecture under MCC environment
2. Construct model to measure latency time and energy consumption the joint platform DNN architecture base on the framework
3. Construct optimization formulation to give an optimal solution in allocating the computational layers to suitable cores of mobile devices or cloud in MCC with minimum latency time and energy consumption and maximize system reliability and QoS.

II. LITERATURE REVIEW

There are two common approaches to implementation DNN-based application in mobile platform, i) the DNN process compression in mobile platform only and ii) compute an entire process in single platform (mobile platform only or cloud platform only). Yet, recently there were several researches proposed new approach to implement the DNN process in the joint platform. In [9], the authors examine the status quo approach of cloud only and mobile only. They use 8 intelligent applications spanning computer vision, speech and natural language domains, all using state-of-the-art DNN as the core machine learning techniques. They design Neurosurgeon, a lightweight scheduler which automatically partition DNN computation between mobile devices and data centers at the granularity of neural network layers. They test Neurosurgeon on a state-of-the-art mobile development platform and show that it improves end-to-end latency by 3.1X on average and up to 40.7X, reduces mobile energy consumption by 59.5% on average and up to 94.7% and improves data center throughput by 1.5X on average and up to 6.7X. In this study, they partitioned DNN into cloud and mobile. They generalize the task from the mobile, then it predicts the model of each layer and determine at what point they will do the layer on the mobile device. In our study, the initial layer will be generated from mobile, while the next layer will be determined based on the cost taken, so there is a possibility that the results got are having a mobile-cloud-mobile or mobile-cloud-mobile-cloud-mobile pattern or maybe it has another pattern. Authors in [15], have introduced a benchmarking framework called "SyNERGY" to measure energy and time from 11 deep convolutional neural networks representations on embedded platforms such as NVidia Jetson TX1. They combined the Streamline Performance Analyzer with a standard deep learning framework such as Caffe and CuDNNv5, to examine how the current process of deep learning models on a specific layer at the same time. Namely the image processing tasks. In addition, they construct an initial multi-variable regression liner model to predict the energy consumption of the invisible neural network model based on several SIMD instructions executed and access to core memory from the TX1 CPU cores. With a common rate test error of 8.04 ± 5.96%.

The results of their study show that it is very possible to redefine the model to predict several SIMD instructions and access to main memory only from the Multiply Accumulate account from the application, then remove the need for actual measurements. They predict the results with an error rate with an average of 7.08 ± 6.0% over the actual energy measurement of all 11 networks tested except MobileNet. By including MobileNet, their average error rate increased to 17.33 ± 12.2. BranchyNet in [16], a novel deep neural network architecture changed with additional branches for classifiers. This architecture predicts results for many samples to exit the network faster through these branches. This research is one of the basic milestones in implementing DNN over mobile device and cloud services applications. They add an early exit to each branch, so this architecture eases the data load of the casual type and architecture. When the input received is very large, and requires a fairly high computing system, they can help this by an early exit branch of this BranchyNet architecture. They implemented this BranchyNet architecture on several well knowns DNN networks such as LeNet, AlexNet, and ResNet. The dataset they use MNIST and CIFAR10. Their research shows satisfactory results. This BranchyNet architecture can improve accuracy and significantly reduce DNN network computing inference time. Our research does not use this BranchyNet network, but we use the method they used to distribute a series of layers in this DNN architecture over mobile devices and cloud services. As we explained above, it has carried several studies related to implementing DNN over mobile devices and cloud services applications out with several approaches. However, there are disadvantages to the status quo approach, mobile-only approach or cloud-only approach. Some studies that trying to combine platforms to give more effective results are still being developed up today. This research continues to developing ideas for implementing DNN-based collaboration platform, mobile devices platform and cloud services platform. The fundamental idea of this research is how to allocate each layer of DNN computation into those two platforms efficiently base on the energy consumption and latency performance of each layer.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 9, Issue 9, September 2020
DOI 10.17148/IJARCCE.2020.9902

III. METHODOLOGY

A. DNN-Based Application: Since it was first proposing, Convolutional Neural Network (CNN) is used for digit recognition, and it gives very satisfying results. In 2012, AlexNet won ImageNet Challenge with an accuracy rate of 85% and since then the development of computer vision applications has developed very quickly and widely. We can take this conclusion base on the number of papers in the field of computer vision which increased dramatically in every year from 2012 to 2017[17]. From the data we got base on Google Scholar and arXiv, the graph of the amount of research in this field showed a very sharp increase. In 2012, the number of papers published was only around 577, then in 2013 there was a slight increase in number to 852, in 2014 the increase was quite significant, namely 1,349 and in 2015 the number of papers published became 2,261 and continued to increase in 2016 , the number became 3,627 and in 2017 there was a very sharp increase, 5,693 and this number continues to increase up to this day[18]. Base on this reason, we decide to use AlexNet topology for image recognition as a study case for DNN applications in our research.

B. Collecting and Extracting Dataset: In order to train the machine, we need huge amount of data to make the model learn by themselves to identifying common feature related to the object[19]. ImageNet is a dataset that used in this image recognition problem[20]. ImageNet is an image dataset compiled for the WordNet hierarchy. Every concept that has meaning in WordNet has the possibility of being able to be translated into multiple words or words phrase, which is called "synonym set" or "synset". There are more than 100,000 synets in WordNet, which are mostly made up of nouns with around 80,000+. In ImageNet, they provide an average of 1000 images to describe each synset. Images from each concept are quality controlled or explained according to human language. ImageNet consists of more than 20,000 categories with a typical category, such as "dog" or "strawberry. The ImageNet project was inspired by the development of research on sentiment analysis for image and video in the field of computer vision. The thing that distinguishes ImageNet from other datasets is the annotation process in an image. It gives an annotation that indicates the presence or absence of a class of objects in an image, such as "there is a cat in this picture" or "no cat in this picture". Image level annotation gives a bounding box that marks an object. ImageNet uses a variety of WordNet schemes to categorize objects, for example there are 120 categories for cat types. This shows a very detailed and deep classification. ImageNet does not have copyright of images; this makes it very suitable for researches and educators who want to use images for educational purposes. ImageNet only provides thumbnails and URLs from images, in a manner similar to that done by image search engines. ImageNet dataset can be downloaded freely in http://image-net.org/download.php.

C. Optimization Framework for Mobile Cloud Task Scheduling: Proposed work in this research is to implement AlexNet over mobile cloud computing environment. A test case will be conducted on both mobile and cloud platforms. Latency time as the performance and energy consumption will be measured in each DNN layer and will allocating efficiently the computational task of each layer to suitable cores of mobile devices or cloud in an MCC environment.

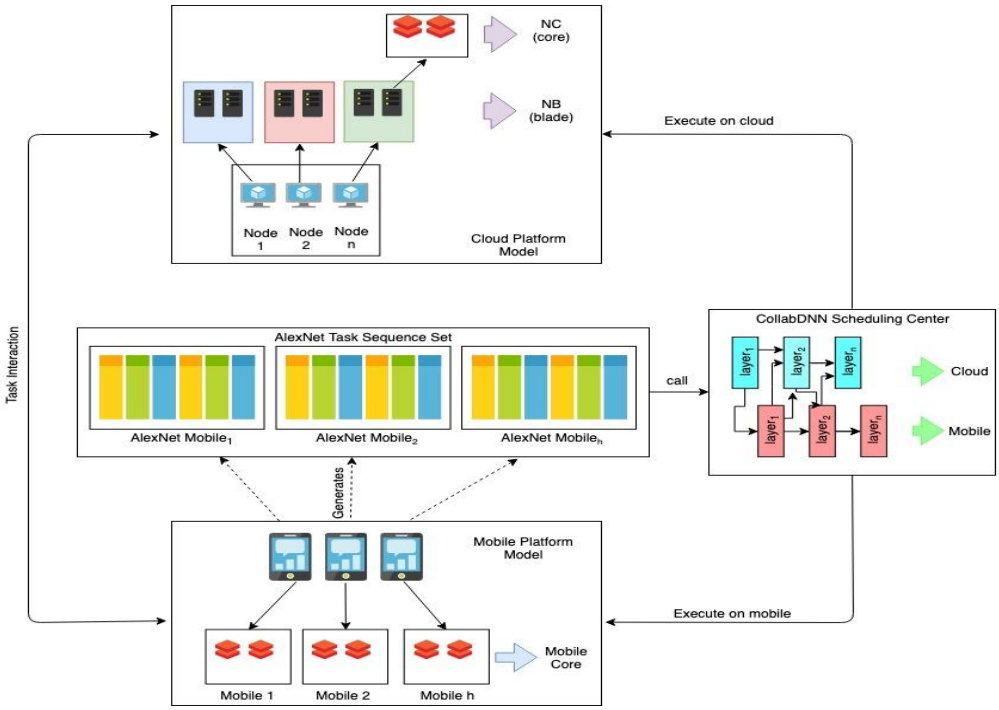

Fig 1. Basic functional diagram of task scheduling under Mobile Cloud Computing environment

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 9, Issue 9, September 2020
DOI 10.17148/IJARCCE.2020.9902

According to the Fig 1, here are the explanation of the system flow for the CollabDNN task scheduling under the MCC environment:

1. Each mobile device generates its own task that generates AlexNet Task Sequence Set (ATSS). AlexNet consists of several layers' computation.
2. CollabDNN Scheduling Center (CSC) calls the CollabDNN scheduling algorithm and allocating which layer will be executed on mobile and which layer will be executed on cloud base on the task allocation and frequency assignment solution.
3. If the allocation is the mobile device platform first, then mobile device will execute the allocated layer computation and then transmits the output of the layer as an input for next layer to the either continue in mobile device again or send to the cloud platform.
4. If the allocation is the cloud platform first, then it will select the specific cores and adjust the running frequency to the execute allocated layer computation then transmit the output of the layer as an input to either mobile device platform or continue execute the layer in the cloud platform.
5. The second platform (mobile platform/cloud platform) chosen will receive output from the previous layer and compute as an input for the current layer. The output from the current layer will be input for the next layer computation.
6. The mobile device platform executes the allocated layer computation on specific cores at adjusted frequency
7. The cloud platform (CP) selects the specific cores and adjusts the running frequency to execute the allocated layer computation
8. We describe the problem as the shortest path problem in a graph description and considering the layers as the nodes and the edge shows the cost among the node.

IV. OPTIMIZATION FLOWCHART AND ALGORITHM

The main idea of this research is how to implement DNN application in joint platform, mobile and cloud efficiently. This research using Dijkstra Algorithm to decide the optimal solution from the shortest path graph representation problem. There are three constraints to considering base on the optimization scenario: energy limitation, latency performance and QoS. The flow of the optimization framework described in Fig 2:

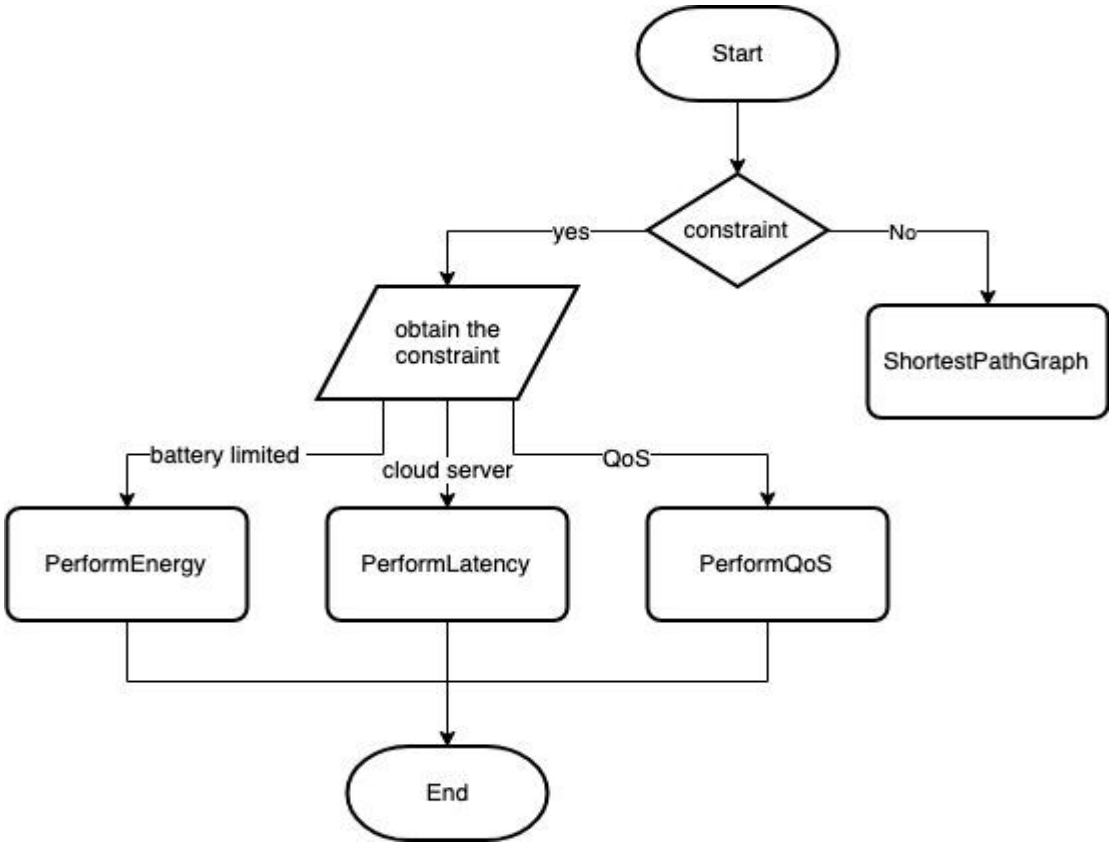

Fig 2. System flow optimization framework

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 9, Issue 9, September 2020
DOI 10.17148/IJARCCE.2020.9902

The Algorithm as below:

Input:
N: number of layers in the DNN
$\{L_i|i=1,\cdots N\}$: layers in the DNN
$\{D_i|i=1,\cdots N\}$: data each size at each layer
$f, g(L_i)$: regression models predicting the latency and power of executing L_i
K: current datacenter load level
B: current wireless network uplink bandwidth
PU: wireless network uplink power consumption
Procedure: PARTITION DECISION

For each i in $1,\cdots N$ **do**

$TM_i \leftarrow f_{\text{mobile}}(L_i, K)$// Latency of mobile execution
$TC_i \leftarrow f_{\text{cloud}}(L_i, K)$// Latency of cloud execution
$PM_i \leftarrow g_{\text{mobile}}(L_i)$// power of mobile execution
$TU_i \leftarrow D_i/B(L_i)$// transfer latency
$Tp \leftarrow f_{\text{mobile-cloud}}(L_i, K)$//computation of mobile cloud
$TCom \leftarrow f_{\text{cloud,mobile}}(L_i, K)$//communication of cloud mobile

$G, S, F = constructGraph(N, L_i, D_i, B, PU)$

If no constraint **then**
return

$$scheduleTask = ShortestPathGraph(G, S, F)$$

If OptTarget==latency **then**
return

$$\arg\min_{j=1,\cdots,N} \sum_{i=1}^{j} TM_i + \sum_{k=j+1}^{N} TC_k + TU_j)$$

else if OptTarget==energy **then**
return

$$\arg\min_{j=1,\cdots,N} \sum_{i=1}^{j} TM_i \times PM_i + TU_j \times PU)$$

elseif QoS== maxQoS **then**
return

$$\arg\min_{j=1,\cdots,N} (\sum_{i=1}^{d} Tp + \sum_{k=j+1}^{N} TCom)$$

Fig 3: Algorithm of the proposed optimization framework

V. EVALUATION METHOD

A. System Platform Model: In the framework proposed, we are constructed four models platform : Mobile platform model, cloud platform model, AlexNet task representation model and CollabDNN task allocation representation model. Those platforms are connected each other and explained as below:

B. Mobile Platform Model: The mobile platform consists of h (number of) mobile devices, and each mobile devices has four components as below:

Table 1. Mobile Platform Model Attribute

Component	Description
J_h	Number of cores
F_h^{max}	Maximum frequency set on each h^{th} mobile devices
BA_h^{user}	Limited of remained battery capacity of the h^{th} mobile device
QoS_h^{user}	Limited of user service on each h^{th} mobile devices

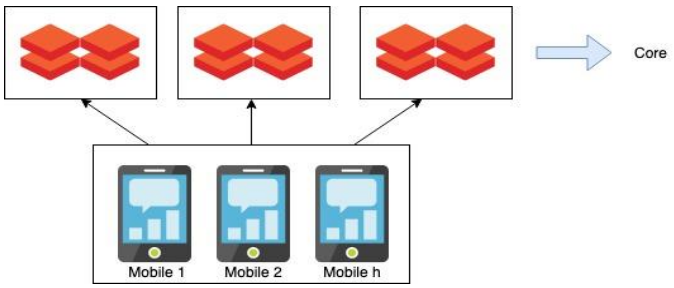

Fig 4. Mobile Platform Model

C. Cloud Platform Model: The cloud data center platform has three attributes as below:

Table 2. Cloud Platform Model Attribute

Attributes	Description
I	Total number of cores in cloud data center
F^{max}	Maximum frequency set of cores in cloud data center
D	Heat circulation matrix of cloud data center

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 9, Issue 9, September 2020
DOI 10.17148/IJARCCE.2020.9902

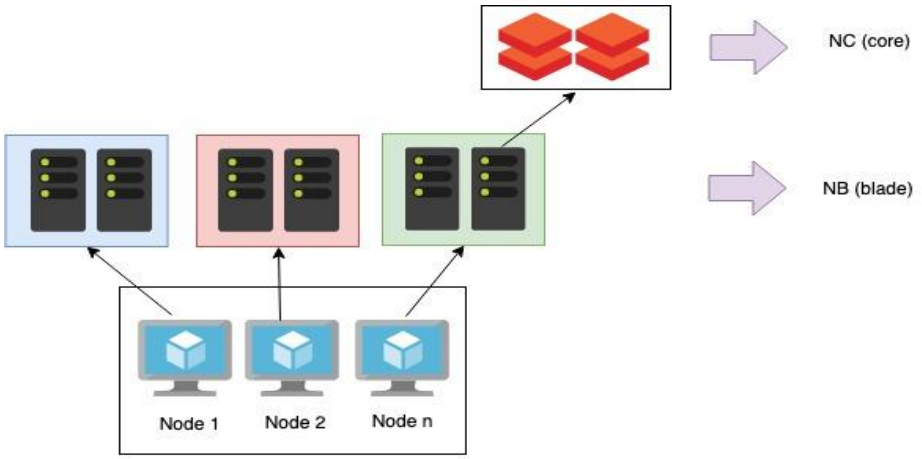

Fig 5: Cloud Platform Model

VI. ALEXNET TASK SEQUENCE SET PLATFORM MODEL

We propose AlexNet as a graph that execute sequentially with a linear topology as depicted in Fig 6:

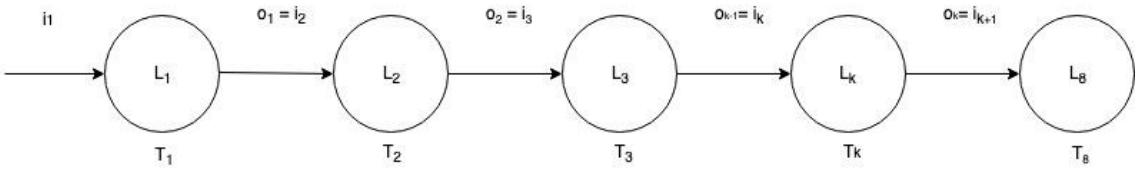

Fig 6. Computation model AlexNet in linear topology

Layers are executed sequentially, with output data generated by previous layer as an input to next layer. We denote the input and output data size of lth layer as il and ol respectively. Furthermore, the flow of the detail AlexNet task sequence set model can describe as picture in Fig 7:

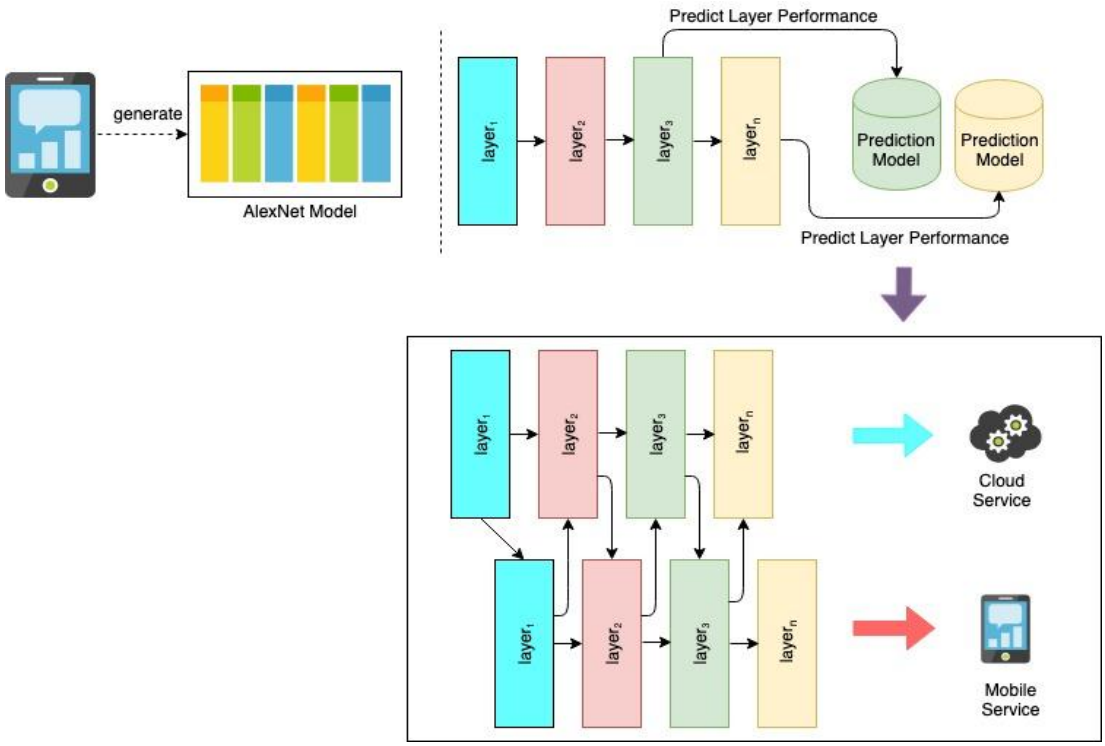

Fig 7. AlexNet task sequence set model

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 9, Issue 9, September 2020
DOI 10.17148/IJARCCE.2020.9902

CollabDNN task scheduling platform can be describe in graph representation as below:

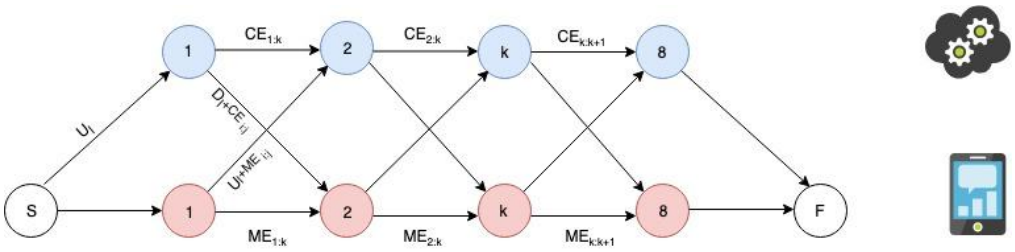

Fig 8. Graph representation AlexNet Joint Platform

A. Data Size and Parameters Layers of The Dnn-Based Application: AlexNet has total 8 layers, consist of 5 convolutional layers and 3 fully connected layers and huge number of parameters. We can calculate the total number parameters in AlexNet network is the sum of all parameters in the all those convolutional layers and all full-connected layers. It gives us the huge size as: 62,378,344. The table followed shows a summary of the calculation of the data size and the total number parameter of AlexNet architecture:

Table 3. Summary of dataset and parameters of AlexNet

Layer Name	Tensor Size	Weights	Biases	Parameter
Input Name	227 x 227 x 3	0	0	0
Conv – 1	55 x 55 x 96	34,848	96	34,944
Max Pool – 1	27 x 27 x 96	0	0	0
Conv – 2	27 x 27 x 256	614,000	256	614,656
Max Pool – 2	13 x 13 x 256	0	0	0
Conv – 3	13 x 13 x 384	884,736	384	885,120
Conv – 4	13 x 13 x 384	1,327,104	384	1,327,488
Conv – 5	13 x 13 x 256	884,736	256	884,992
Max Pool – 3	6 x 6 x 256	0	0	0
FC – 1	4096 x 1	37,748,736	4,096	37,752,832
FC – 2	4096 x 1	16,777,216	4,096	16,781,312
FC – 3	1000 x 1	4,096,000	1000	4,097,000
Output	1000 x 1	0	0	0
				62,378,344

VII. EXPERIMENTAL RESULTS

We compared the best case of single platform, mobile-only or cloud-only with joint platform, for inference and training phase. Communication cost and batch size are increase linearly. The result shows that all the back-propagation will be executed on the mobile device and weights are not transmitted from the cloud to the mobile. The collaboration platform can achieve improvements up to 73% in latency and 56% in energy consumption during inference. The result also shows the unique pattern for scheduling. The computation in DNN is divided between mobile and cloud, and the optimal solution while optimizing for latency is in Wi-Fi network. We can see the AlexNet pattern as inference phase of AlexNet follows a mobile-cloud pattern. In this inference phase, some first layers executed in mobile and the rest layer executed in cloud platform. The last layer of AlexNet, fully connected (FC) layer which has small data size, should be done in cloud for the low communication cost. The training phase of AlexNet follows a mobile-cloud-mobile pattern. Some first layers executed in mobile, then some layer in middle executed in cloud and the rest layers executed in cloud.

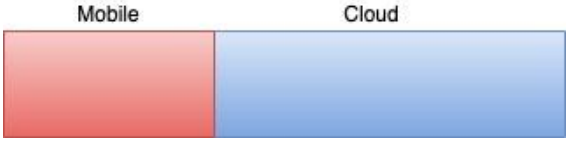

Fig 9. AlexNet latency efficient inference

Fig 10. AlexNet latency efficient training

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 9, Issue 9, September 2020
DOI 10.17148/IJARCCE.2020.9902

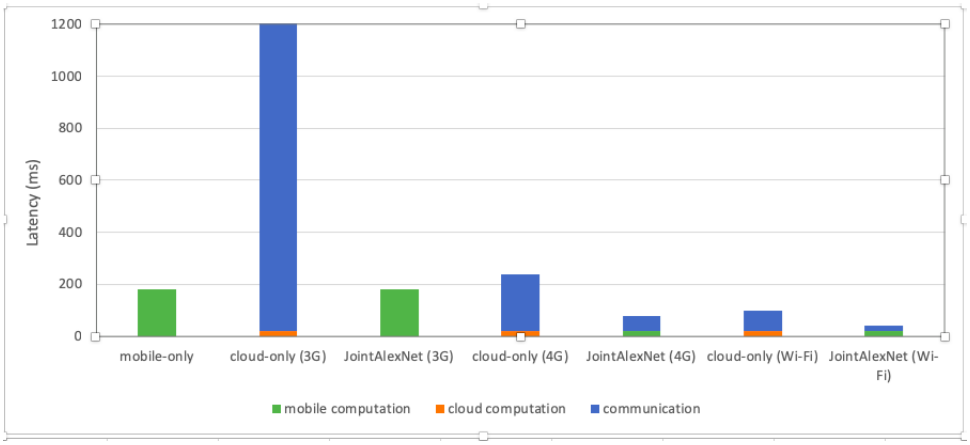

Fig 11. Execution time of AlexNet optimized for performance

In these figures below, we can see that execution time and energy analysis for AlexNet utilized in cloud servers. The communication costs are dominating the cloud-only approach. Total execution time for communication 3G, 4G, and Wi-Fi are 99%, 93%, and 81%, respectively as we can see from the Fig 11 and Fig 12. This proportion also valid to energy consumption. We can compare the latency and energy of the communication for those mobile-only approach, the result shows that AlexNet in mobile-only approach is better than cloud-only approach in all mobile networks condition. In Fig 11 we can see the execution time of AlexNet for latency performance for each platform and mobile energy consumption of AlexNet for each platform in Fig 12.

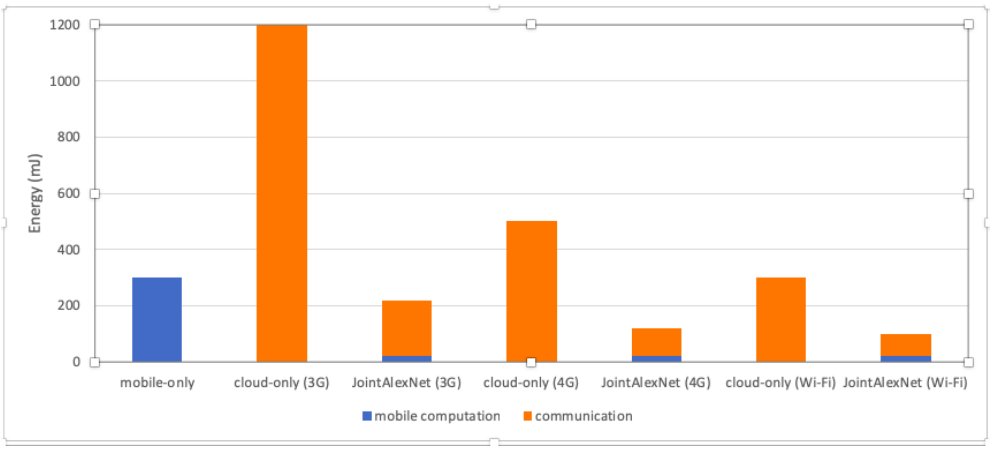

Fig 12. Mobile energy consumption of AlexNet optimized for energy

VIII. CONCLUSION

In this research, we construct AlexNet-based application on joint platform mobile and cloud services under the complex mobile cloud computation environment. The problem how to distribute the computation layer to the suitable core in cloud or mobile represented in shortest path problem graph. Each layer performed as a node and latency performance and energy consumption of each layer are considered as a cost among the nodes. We also add constraints mobile battery and latency performance in mobile and cloud and QoS. CollabDNN shows different pattern result between inference phase and training phase. For inference phase, some first layer executed in mobile and the rest executed in cloud, but in training phase some first layer executed in mobile, then some next layer executed in cloud and the rest layer executed in mobile again. This result might be different for different topology.

Overall, result of this research shows that joint platform mobile and cloud approach is give better performance and optimal solution by considering latency and energy consumption compare status-quo approaches, cloud-only or mobile-only. However, there are still some challenges that can become future work to improve performance efficiently, including:

1. Using heuristic approaches such as: ant colony or swarm intelligent, to solve shortest path search problems in task allocation for each computing layer under the complexity of MCC environment
2. Using reinforcement learning to predict partition models in each layer
3. Add some constraints to give better performance and applying those methods for other DNN topologies

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 9, Issue 9, September 2020

DOI 10.17148/IJARCCE.2020.9902

REFERENCES

[1] T. Ben-Nun and T. Hoefler, "Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis," 2018.
[2] D. Marmanis *et al.*, "Artificial Generation of Big Data for Improving Image Classification: A Generative Adversarial Network Approach on SAR Data," 2017.
[3] Y. Demir, "Face Recognition Based on Convolutional Neural Network," pp. 376–379, 2017.
[4] G. Huang and K. Q. Weinberger, "Densely Connected Convolutional Networks."
[5] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," *Proc Deep Learn. …*, pp. 611–620, 2011.
[6] T.A.L.Ben-nun, T.Hoefler, & E.T.H. Zurich, "1 Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis," 2018.
[7] B. Qi, M. Wu, and L. Zhang, "A DNN-Based Object Detection System on Mobile Cloud Computing," 2017.
[8] L. Deng, D. Yu, and J. Platt, "Scalable stacking and learning for building deep architectures," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, pp. 2133–2136, 2012.
[9] Y. Kang *et al.*, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *Proc. Twenty-Second Int. Conf. Archit. Support Program. Lang. Oper. Syst. - ASPLOS '17*, pp. 615–629, 2017.
[10] E. Alhamad, Mohammed; Dillon, Tharam; Chang, "Conceptual SLA Framework for Cloud Computing," *IEEE*, vol. 3, no. 3, pp. 741–751, 2010.
[11] E. Li, Z. Zhou, and X. Chen, "Edge Intelligence : On-Demand Deep Learning Model Co-Inference with Device-Edge Synergy," pp. 1–10, 2018.
[12] A. Cichocki, "Tensor Networks for Big Data Analytics and Large-Scale Optimization Problems," pp. 1–36, 2013.
[13] J. Ren, Y. Pan, A. Goscinski, and R. A. Beyah, "A Tensor-Based Holistic Edge computing Optimization Framework for the internet of things," *IEEE Netw.*, vol. 32, no. 1, pp. 6–7, 2018.
[14] A. Krizhevsky and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," pp. 1–9.
[15] C.F.Rodrigues, G.Riley, & M. Lujan, "Fine-Grained Energy & Performance Profiling framework for Deep Convolutional Neural Network," 2018.
[16] B. Mcdanel, "BranchyNet : Fast Inference via Early Exiting from Deep Neural Networks."
[17] Y. Bi, R. Bhatia, and S. Kapoor, "Intelligent Systems and Applications: Proceedings of the 2019 Intelligent Systems Conference (IntelliSys) Volume 1," *Adv. Intell. Syst. Comput.*, vol. 1037, no. October 2019, pp. 1–1316, 2020.
[18] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing Neural Network Architectures using Reinforcement Learning," pp. 1–18, 2016.
[19] G. Smoluk, "Google net," *Mod. Plast.*, vol. 57, no. 3, pp. 62–63, 1980.
[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2015.