Multimedia Tools and Applications (2023) 82:12229–12243
https://doi.org/10.1007/s11042-022-13712-3

On edge deep learning implementation: approach to achieve 5G

Dhritiman Mukherje[1] · **Aman Anand**[1]

Received: 1 December 2020 / Revised: 25 April 2022 / Accepted: 24 August 2022 /
Published online: 21 September 2022

Abstract
Through 5G networks, mobile edge computing (MEC) brings the power of cloud computing, storage, and analysis closer to the end user. Innovative inventions in the domain of multimedia and others such as connected cars, large-scale IoT, video streaming, and industry robotics are made possible by improved speeds and reduced delays. On the other hand, in mobile edge computing, machine learning (ML) is leveraged to predict demand changes based on cultural events, natural disasters, or daily travel patterns, and it prepares the network by automatically scaling up network resources as required. Mobile edge computing and ML together allow seamless network management automation to decrease operating costs and boost user experience. In this paper, we discuss the state of art with in mobile edge computing with deep learning to server low-latency, real time application by providing application specific resource allocation. The experimental results have indicated significant amount of improvement in respond time while executing in low latency.

Keywords 5G · Fog computing · Edge computing · Deep learning

1 Introduction

In recent years, the number of Internet of Things (IoT) devices has significantly increased due to the exponential growth of wireless communication technology. More than 25 billion devices are expected to be connected to the Internet by 2020[8] and the total economic effect of the IoT by 2025 will be $3.9 trillion to $11.1 trillion annually [9]. Usually, IoT devices have minimal processing capacity and tiny memories. Sensors, cameras, smart fridges, and smart lights are examples of such resource-constrained IoT products. Large quantities of data are continually produced by IoT devices and sensors, which are of vital importance for many modern technological applications, such as autonomous vehicles. To obtain

✉ Dhritiman Mukherje
dmukherjee3@kol.amity.edu

Aman Anand
amananand972@gmail.com

[1] Department of Computer Science and Engineering, Amity University, Kadampukur, 24PGS(N), Kolkata, West Bengal, 700135, India

knowledge and make decisions, this knowledge must be fed to a machine learning system. Unfortunately, limitations in resource-scarce devices' computational capacities hinder the implementation of ML algorithms on them. The data is then deposited into remote computing facilities, most generally cloud servers, where computations are carried out. Transferring raw data to cloud storage increases the cost of communication, causes device responses to be delayed, and leaves any private information vulnerable to compromise. It is appropriate to consider processing data closer to its origin and sending only the required data to remote servers for further processing in order to resolve these problems [2].

Edge computing refers to computations conducted as near as possible to data sources rather than at distant, remote locations [1, 13]. This is accomplished by adding an edge computing system close to the resource-constrained devices where data is generated (or simply, edge devices). There are both computing and communication capabilities for Edge applications. Computations that are too intense for edge devices are sent to remote servers that are more efficient. Driven by the advent of new computer-intensive applications and the vision of the Internet of Things (IoT), an exponential rise in traffic volume and computational demands is anticipated for the evolving 5G network. End users, however, often have restricted storage capacities and finite processing resources, so it has recently become a common concern to run compute-intensive applications on resource-constrained devices. Mobile edge computing (MEC), a key technology in the evolving fifth generation (5G) network, can leverage mobile resources by hosting compute-intensive applications, processing large data before sending to the cloud, providing Radio Access Network (RAN) cloud computing capabilities in close proximity to mobile users, and providing RAN information with context-aware services. 5G mobile edge computing (MEC) is expected to be a multi-million - dollar market by 2024 with $73 million in enterprise deployments [18].

Through 5G networks, mobile edge computing (MEC) brings the power of cloud computing, storage, and analysis closer to the end user. Innovative inventions in the domain of multimedia and others such as connected cars, large-scale IoT, video streaming, and industry robotics are made possible by improved speeds and reduced delays. On the other hand, in mobile edge computing, machine learning (ML) is leveraged to predict demand changes based on cultural events, natural disasters, or daily travel patterns, and it prepares the network by automatically scaling up network resources as required. Mobile edge computing and ML together allow seamless network management automation to decrease operating costs and boost user experience [10]. A note related to terminology. Fog computing, a similar concept, describes an architecture where the 'cloud is extended' to be closer to the end-devices of the IoT, thereby enhancing latency and security through computing near the network edge [10]. The major difference is where the data is processed: in edge computing, data is processed directly on the devices to which the sensors are attached (or physically very close to the sensors on the gateway devices); in fog computing, data is processed further away from the edge, on devices connected via a LAN [6]. Layering ML on top of the mobile edge computing infrastructure allows efficiency-enhancing network functions at the edge to be automated. ML and mobile edge computing, together, can enable real-time applications with cloud computation at low-latency.

Authors in [7] have proposed a decentralized communication model with a smart device and vehicle base station servers. Further, they have formulated a combinational optimization graph problem based on the superposed-data collection problem. However, their research work addressed only the problem of finding communication link and strategy which minimizes the total energy consumed by smart devices. In the case of computation-intensive task-based real-time time applications, optimizing response time will be challenging. In

[15] authors mainly focuses on the deep learning overview and explain how that can be adopted in mobile edge network with computation and memory limitation. Further, they have also studied deep learning techniques and challenges in the 5G network. They only focus on the abstract view of the discussed study. In [16] authors have proposed a smart home architecture based on cloud computing and blockchain. They have implemented hashing and encrypted algorithms to provide integrity and confidentiality in the network. In case of computation-intensive device transactions, huge network delay can be incurred while getting results for computations. However, none of the above-discussed papers has implemented a real test case scenario that deals with challenges like response time, computation and memory constraints.

In this paper,

- An overview of 5G network is discussed.
- The power of machine learning implemented at the network edge to serve low-latency, real time application that are considered primary goal for designing the 5G network.
- Challenges like inherent resource constraint and computational limitations at edge device are identified and proposed solution are also addressed.
- Respond time is compared in low latency with different data sets.

2 Research methodology

The overall architecture of the proposed system shown at Fig. 1. At level 0, edge/IoT layer is present.

In edge/IoT layers edge devices can be considered as smartphones, sensors, tablets, laptops, moving cars etc. At top of the level 0 the fog layer is present. Fog layer can be considered as multiple interconnected fog controller. Fog controllers are the intelligent devices that also represents the cloud near to the ground. In this project we established the fog

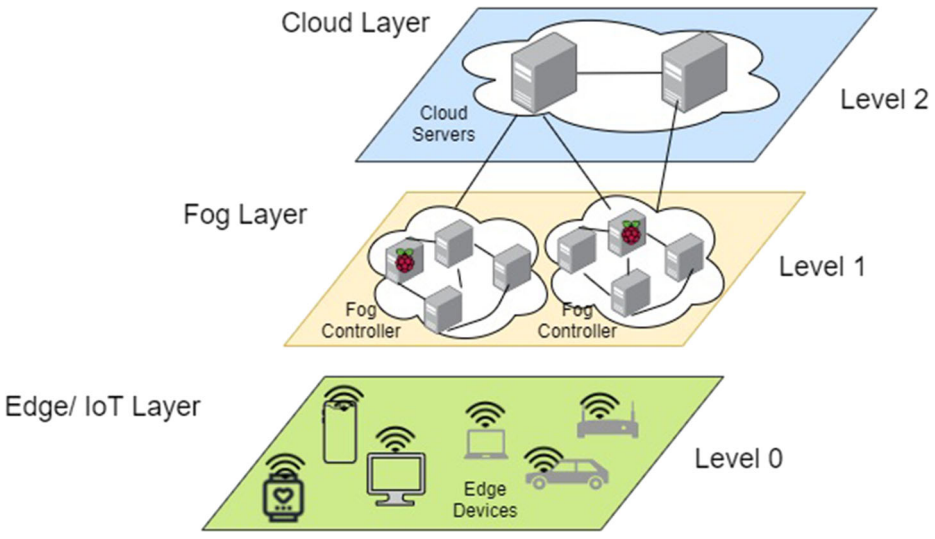

Fig. 1 The Overall architecture

environment inside the raspberry pi device and we have considered that device as a fog controller. At level 2 or top of the hierarchy remote cloud servers are present.

Figure 2 describes the overall flow diagram of the proposed system Now a day, smartphones are so powerful device that it can able to perform computational intensive task. In spite of that they are still suffering the inherent problem like limited battery power, memory uses etc. Hence, the challenge of performing entire deep learning operation on a mobile device often be computational intensive. In this study, this issue is resolved by performing training phase on a powerful GPU outside the mobile device. After successful completion the training phase the trained model deployed in mobile device to classify the images.

As discussed due to limited memory capacity of mobile devices it sometimes become challenging to deploy trained model into the mobile devices. This issue has been resolved by

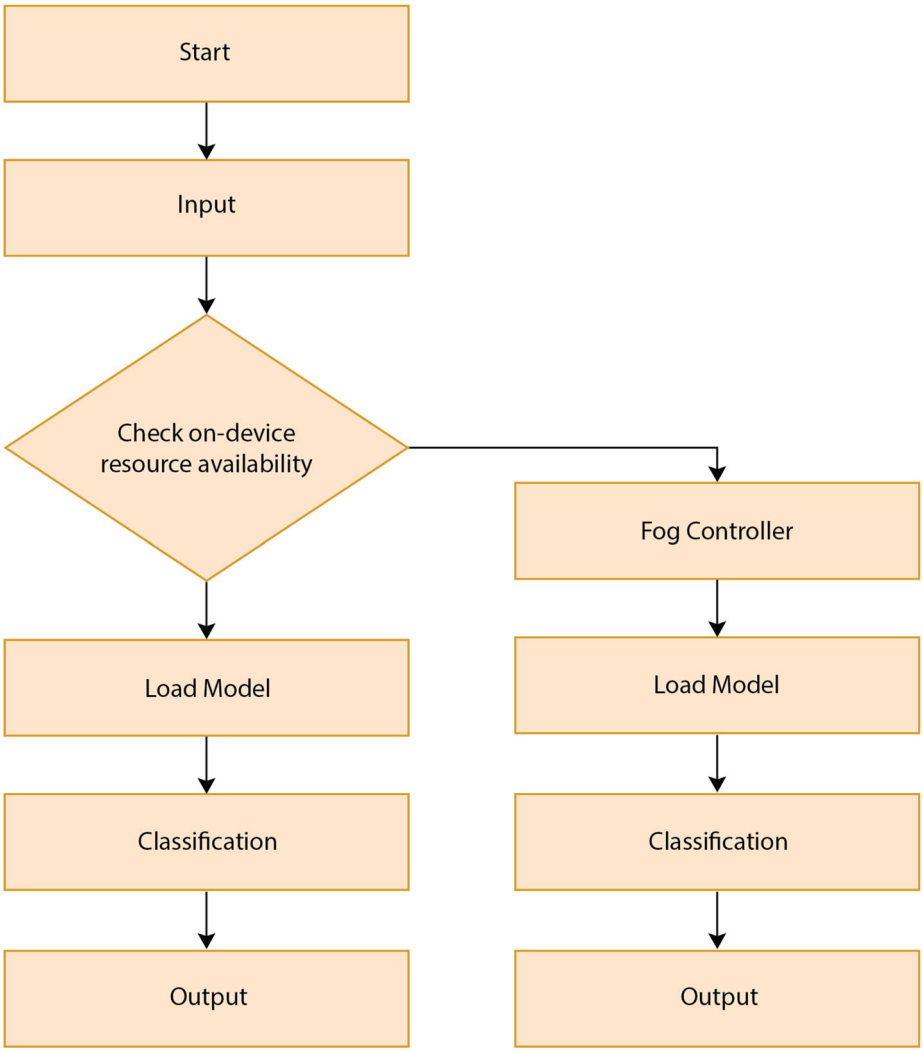

Fig. 2 The overall flow diagram of proposed system

offloading the entire task (i.e. image) to the fog controller depending on the available device resources. In that situation, only final classify result will display on the mobile devices. Hence, entire classification process will be initiated by the fog controller.

2.1 Transfer learning

When the dataset is significantly small transfer learning can be very effective tool to develop a large network without overfitting. Firstly, in transfer learning a base network is trained on a dataset. Furthermore, the learned features are transferred or repurposed to the second target network to be trained on the target dataset. The remaining layers are randomly initialized and trained. This process will work if the features are general and fit for the both base and target tasks. Transfer learning can be considered a powerful tool for training a large network when the target dataset is significantly smaller than the base dataset [19]. Deploying pre-trained model on similar dataset shown good image classification result [11, 14]. Few organizations are working on building such models such as Oxford VGG model [14], Google Inception Model and Microsoft ResNet Model [5]. These models can be integrated with existing research to bring better result.

2.2 Microsoft ResNet model

In recent past, among deep learning models CNN have become the leading architecture in the field of image classification and recognition. It uses multiple convolution layers for complex tasks like information processing, feature extraction, pattern analysis. Neural network suffers with problem like vanishing gradient [4] and degradation problem [17] while training deeper layer. So, researchers observed adding more layers eventually had an improper effort in final result.

ResNet was introduced as an innovative solution to the vanishing gradient problem. Residual Network (ResNet) is a convolution Neural Network(CNN) architecture which can enable hundreds or thousands of convolution layers. ResNet claimed the first position at ILSSVRC 2015 in image classification and detection and localization as well as winner of MS COCO 2015 detection and segmentation [12]. As we see in the Fig. 3 a skip/shortcut connection was added to add input x to the output to avoid vanishing gradient problem. Hence, the output $H(X) = F(X) + X$. The model instead of learning from the features tries to learn from the residual mapping:

$$F(X) = H(X) - X \tag{1}$$

The input for our model is RGB image. We use flower classification for the proof of work. The image will reach to the pre trained ResNet layer with pertained weight. The last layer of the model is fully connected dense layer with softmax activation. The two significant layer for the proposed model is pre- trained ResNet layer and the dense layer. The weights for the pre-trained model is to be imported and the last layer i.e. dense layer will learn from back propagation. The proposed model architecture shown in Fig. 4.

The Relu action function are being used in the ResNet model. RestNet50 is a 50-layer residual network architecture. It also has other variants such as ResNet101 and ResNet152. Software activation function is used in this proposed architecture. Softmax is kind of multiclass sigmoid function. Softmax is use for the multiclass classification. In this paper, we

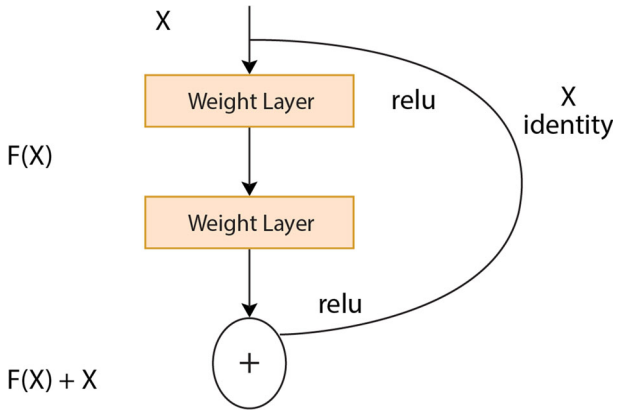

Fig. 3 ResNet Architecture

use softmax activation because we are classifying 5 classes of flowers. The mathematical presentation of the softmax activation function is as follows.

$$\sigma(z) = \frac{e^{Z_j}}{\sum_{j=1}^{k} e^{Z_j}} \tag{2}$$

2.3 Model conversion and integrating pre-trained model into the application

Tensorflow lite is designed to execute models efficiently on smartphones and other devices with minimal processing power and resources. Converting model results reducing the file size and introducing optimization without affecting the accuracy [3]. Tensorflow lite converter is a tool available in python API to convert pre trained model into tflite model. We used following coding snippet for conversation.

$$model = tf.keras.models.Load\_model(`pre - trained.h5')$$
$$converter = tf.lite.TFLiteConverter.form_keras)model(model);$$
$$tflite_model = converter.convert()$$
$$open(\backslash convertedmodel.tflite", \backslash wb").write(tflite_model)$$

Code snippet for the necessary steps to convert model for the smartphone device.

Figure 5 demonstrates the model conversion process of integration pre-trained model inside the edge device. Trained model deployed into the edge devices after successfully completion of training phase. This integration process brings the machine learning model from the cloud server onto the edge devices. However, ML model (pre-trained.h5) not designed to execute efficiently on mobile, IoT and other embedded devices [3]. In most of the cases, pre-trained model has to be converted into the suitable format and integrated onto edge devices. In this study, TFLiteConverter has been used to convert (.h5) format to (.tflite) model format [4].

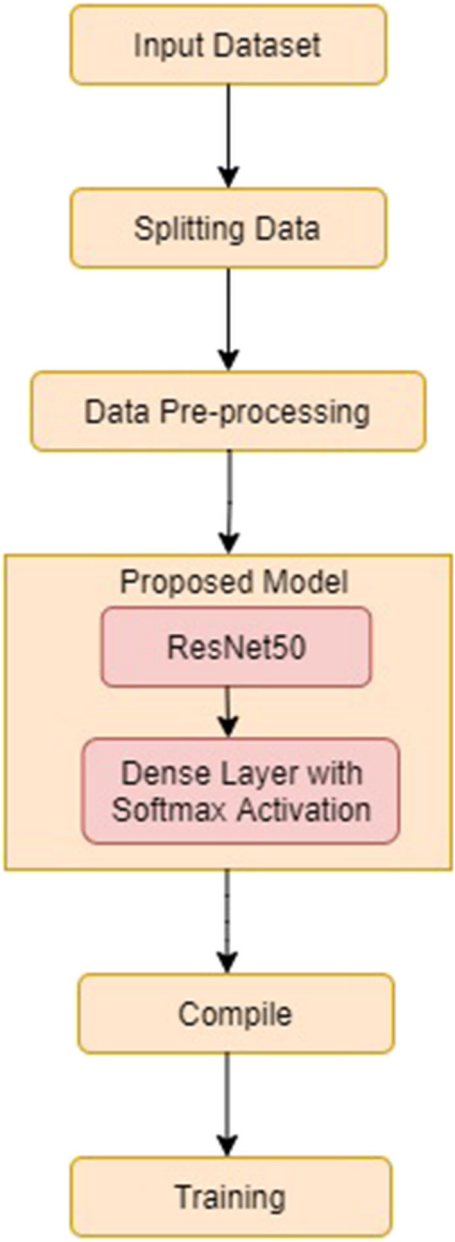

Fig. 4 Proposed model architecture

3 Experiment and result

The machine learning part of the project is conducted in a Google Colab notebook. In the backend for the deep learning environment, Keras 2.4.3 and Tensorflow 2.3.0 were used. Apart from that fog environment is established inside raspberry pi B+ model. Android Pi

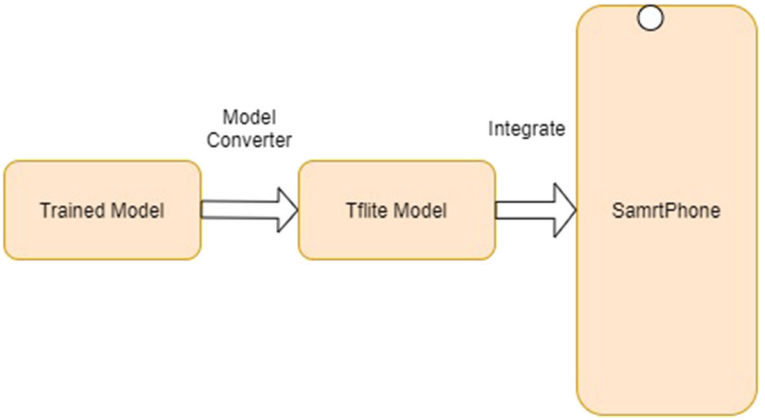

Fig. 5 Pre-trained model integration into the Edge Device

with 6GB RAM 4500mAH lithium-polymer battery configuration base smartphone is used for the edge devices. v9Tesla P100- 16 GB GPU is used for the training the model.

3.1 Dataset

We use a dataset of flower images to demonstrate the proof of work. The dataset is from Kaggle's flowers recognition. The dataset consists of 4242 images of flowers. Images are divided into five classes namely daisy, dandelion, rose, sunflower, tulip. These are about 800 images for each classes. Number of image in each class category are listed below Table 1:

The images were resized to 256 ×256, as original size 375×500 was too large to train in the tensorflow. Furthermore, the dataset was split into 80% of training and 20% of validation.

3.2 Spliting data

Total number of flower data in the data set are 4326. We split out dataset in 80:20 basic i.e. 80% of data used for the training purpose and rest of the data for the training purposes. It is well known practise of splitting data gets more than two-third of total data. This training data contains all the available classes of flowers in the dataset. Figure 1 shows the android application that is executing using devices' own resources. Figure 2 shows the output from model trained from ResNet 50.

Table 1 Data component in dataset

Image class	No of image
Daisy	769
Dandelion	1055
Rose	784
Sunflower	734
Tulip	984

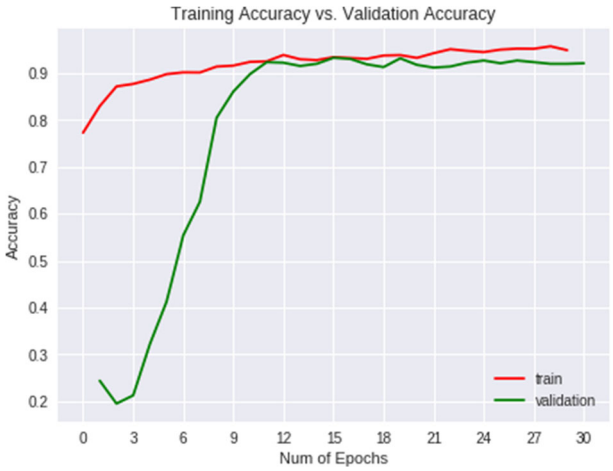

Fig. 6 Accuracy graph

3.3 Model training

In transfer learning the first base n layers are trained then the copy of that n layers transfers to the target network. The remaining layers of the target network are initialized randomly and trained towards the target network. Weights are updated by back propagating the errors and fine-tune the copied feature to the new task. The choice of fine-tune the first n layers depends on the size of the target dataset and number of parameters presents in first n layers. In case of small dataset and large parameters fine-tuning may cause overfitting hence the features are left frozen. On the other hand, overfitting can be avoided if the target dataset is large and number of parameters are large [19].

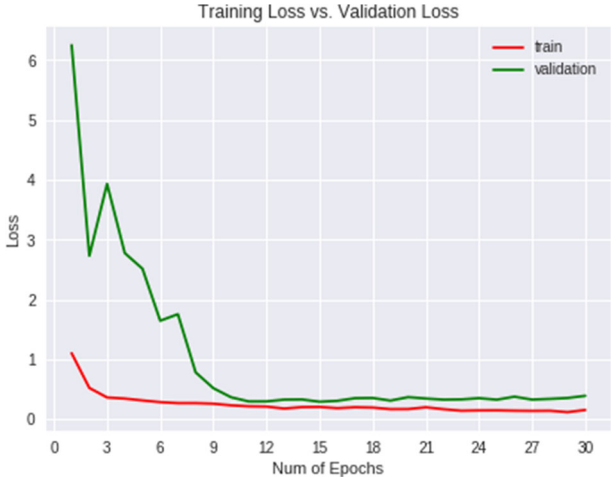

Fig. 7 Loss graph

Table 2 Training performance

Matrices	Data
Training Accuracy	95%
Validation Accuracy	93%
Training Loss	0.11
Validation Loss	0.13

3.4 Result

During training accuracy and loss matrices were measured. Tesla P100- 16 GB GPU is used for the training the model. Figure 6 shows the accuracy graph during the training of the dataset and Fig. 7 shows the loss graph during the training of dataset. Accuracy and loss data for training and validation are shown in Table 2.

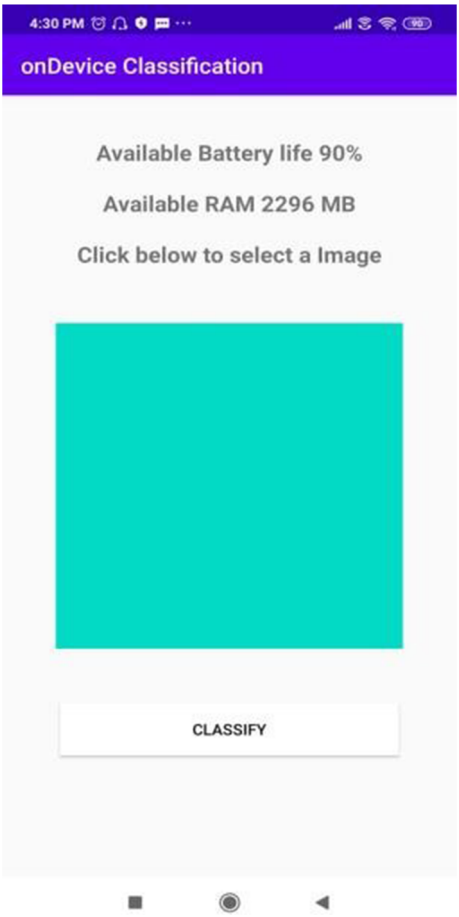

Fig. 8 Edge device resources availability before processing starts

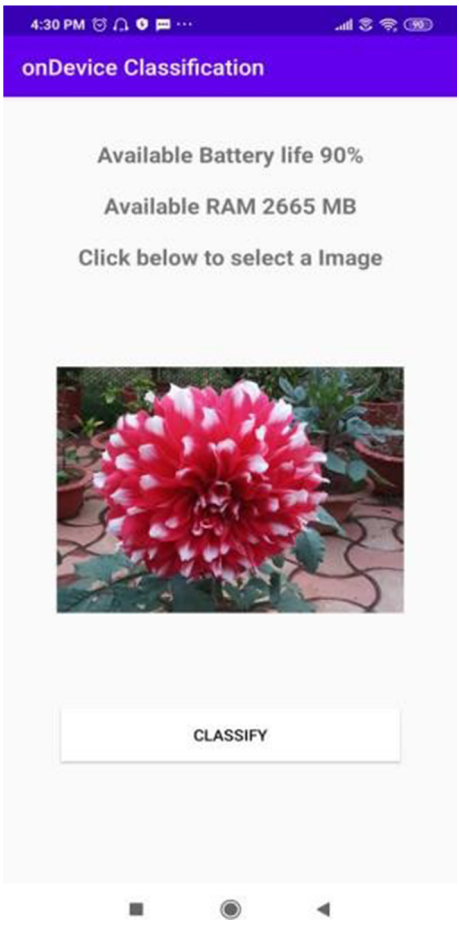

Fig. 9 Edge device resources availability while ongoing processing

Figures 8, 9 and 10 shown the results when we are implementing the classification task at the smartphones. The resource availability at the beginning of the task and at the end of the task are displayed at the top of the application. This resource availability information is the key part for making decision whether task (input image) will be uploaded to the fog controller or not.

3.5 Respond time comparison

In the last part of our experiment respond time is measured with different data size while executing entire task inside the edge devices. Parameter details and corresponding values are listed in Table 3.

Figure 11 presents the impact of respond time for different data size. Here, responds time is modeled as time difference between completion and submission of task. It is discussed

Fig. 10 Edge device resources availability after completion of processing

that computational capabilities of edge devices resides closer to the data source. As, a consequence, network delay is negligible for edge devices. Furthermore, if the size of the data set is not varying the respond time will not differ significantly whether the application is executed in Fog or Edge.

Table 3 Parameter details of fog and edge

Parameter	Value
Bandwidth of Fog Server	34 Mbps
Number of Edge Devices	1 -5
Data size	20 - 40 KB

Fig. 11 Respond time comparison of edge devices for different data sizes

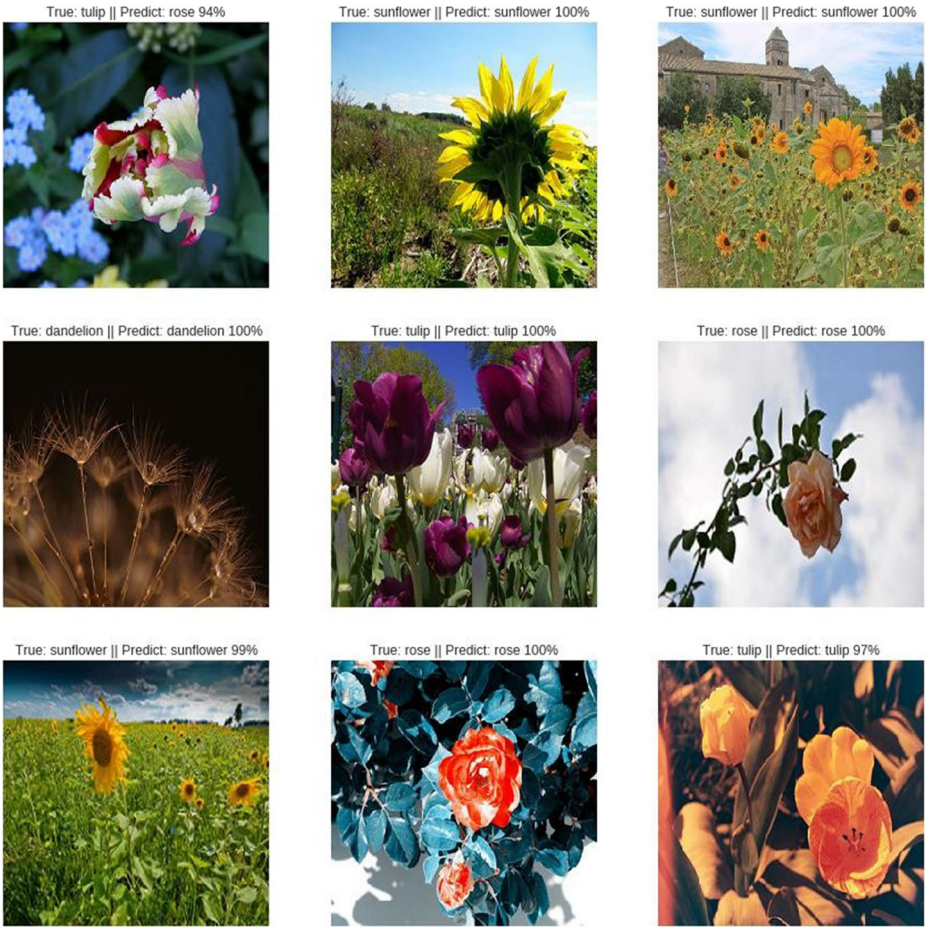

Fig. 12 Classification and accuracy of the trained model

4 Conclusion and future work

In the area of new computation intensive task and IoT it is predicted that the emerging 5G network will face an unprecedented growth in traffic volume. Mobile edge computing (MEC) or edge application, a key technology that can optimize mobile resource by offloading the task to the nearby server. Edge application is considered one of the most informative drivers for information delivery application. The combination of deep learning and edge computing has great potential to transfer knowledge with minimum delay. Many machine learning studies on mobile devices have been focus on cloud-based solution because limitation of memory and computation power and latency causes some issue in case of computational intensive tasks. In this study we present on device inference approach. As pertained model is deployed on the edge devices like smartphones it is not necessary to invoke remote server. Still, we consider the situation when remote server can take action. Instead of using cloud we used fog (i.e. close to user) server. In future we are focusing to extend our work in medical image processing domain where we can reduce the latency to get the response in minimum delay within radio access network (RAN) in close proximity to mobile users. The overall classification and accuracy of trained model shown in Fig. 12.

Acknowledgements On a grateful note, we want to acknowledge our sincere thanks to Dr. Pulak Konar, Associate Professor, Department of Mathematics, The ICFAI University, Tripura (Kamalghat, Mohanpur, West Tripura, India) for his association with this manuscript with useful discussion, valuable contribution and for constant support.

Funding This research received no specific grant from any funding agency in the public, commercial or not-for-profit sector.

Declarations

Conflict of Interests The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Butler B (2017) What is edge computing and how it's changing the network. Network World, Accessed: 21 July 2019
2. Chen Y, Wu A, Bayoumi MA, Koushanfar F (2013) Editorial low-power, intelligent, and secure solutions for realization of internet of things. IEEE J Emerg Select Topics Circ Syst 3(1):1–4
3. Converting Trained Model to TFLite Model https://www.tensorflow.org/lite/guide/get_started
4. Habibzadeh M, Jannesari M, Rezaei Z, Baharvand H, Totonchi M (2018) Automatic white blood cell classification using pre-trained deep learning models: Resnet and inception. In: Tenth international conference on machine vision (ICMV 2017), vol 10696. International Society for Optics and Photonics, p 1069612
5. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
6. Iorga M, Feldman LB, Barton R, Martin M, Goren NS, Mahmoudi C (2018) Fog computing conceptual model. https://doi.org/10.6028/NIST.SP.500-325. Special Publication (NIST SP) - 500-325
7. Li W, Xu H, Li H, Yang Y, Sharma PK, Wang J, Singh S (2019) Complexity and algorithms for superposed data uploading problem in networks with smart devices. IEEE Int Things J 7(7):5882–5891
8. Mahdavinejad MS, Rezvan M, Barekatain M, Adibi P, Barnaghi P, Sheth AP (2018) Machine learning for internet of things data analysis: a survey. Digit Commun Netw 4(3):161–175

9. Manyika J, Chui M, Bisson P, Woetzel J, Dobbs R, Bughin J, Aharon D (2015) The internet of things: mapping the value behind the hype. Technical report, McKinsey and Company, 6
10. McClellan M, Cervelló-Pastor C, Sallent S (2020) Deep learning at the mobile edge: opportunities for 5G networks. Appl Sci 10.14:4735
11. Quattoni A, Collins M, Darrell T (2014) Transfer learning for image classification with sparse prototype representations. In: 2008 IEEE conference on computer vision and pattern recognition 2008 Jun pp 1–8. IEEE. Processing Systems 27 (NIPS '14), NIPS Foundation
12. Ren S et al (2016) Faster r-cnn: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39.6:1137–1149
13. Rothe R, Timofte R, Gool LV (2015) DEX: deep EXpectation of apparent age from a single image. In: 2015 IEEE International conference on computer vision workshop (ICCVW), pp 252–257
14. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556
15. Singh S, Jeong YS, Park JH (2016) A survey on cloud computing security: issues, threats, and solutions. J Netw Comput Appl 75:200–222
16. Singh S, Ra IH, Meng W, Kaur M, Cho GH (2019) SH-BlockCC: a secure and efficient Internet of things smart home architecture based on cloud computing and blockchain technology. Int J Distrib Sensor Netw 15(4):1550147719844159
17. Wichrowska O, Maheswaranathan N, Hoffman MW, Colmenarejo SG, Denil M, de Freitas N, Sohl-Dickstein J (2017) Learned optimizers that scale and generalize. In: Proceedings of the 34th international conference on machine learning, vol 70, pp 3751–3760. JMLR.org.
18. Wood L (2019) 5G optimization: mobile edge computing, APIs, and network slicing 2019–2024; technical report for research and markets: Dublin, Ireland, 22 October
19. Yosinski J, Clune J, Bengio Y, Lipson H (2014) How transferable are features in deep neural networks? arXiv:1411.1792