Chapter 6

Compression of Deep Convolutional Neural Network

In previous chapters, it is seen that classifying histopathological images after extracting the features from each block of VGGNet-16 improved the classification performance. However, in order to progress towards developing deep learning models for resource-constrained devices, we are looking for different ways to make deep neural networks faster and more energy-efficient. Therefore, we propose a lightweight CNN model, MobiHisNet, in this chapter.

6.1 Introduction

With the boom in internet growth and mobile communication systems, the Internet of Things (IoT) has become an important component of our daily life. Internet of Everything (IoE) is the next wave of internet growth, encompassing the interconnectivity of various technologies, processes, and people in a vast distributed network. The advent of 5G is beginning to transform our lives, making IoE a reality. The use of IoT and IoE based smart technologies in the healthcare sector has increased significantly in the past few years, which has led to the development of powerful diagnostic devices and enabled

new forms of medical treatment. The integration of AI, edge computing, and IoE has the potential to revolutionize the healthcare system, particularly in the field of cancer theranostics.

Cancer incidence rates are increasing in the developing world as a result of an increase in life expectancy. Among cancer, breast cancer is the most prevalent cancer in women. Although deep learning frameworks have proven their efficacy in the classification of breast cancer, these algorithms are computationally expensive and require huge parameters. This makes them less affordable for mobile devices and IoT. When compressing a neural network, the tradeoff is between network size and accuracy. In general, the smaller the network, the faster it runs (and the less battery power it requires), but the poorer the predictions. It is also observed in previous chapters that the use of the VGGNet-16 model for histopathological image analysis has the potential to improve breast cancer diagnosis. However, when compared to other CNNs, VGGNet-16 has a very large size and parameters. Also, after compressing the VGGNet-16 model, the accuracy of the model suffers greatly. Since providing a low-cost solution is highly valuable in the computing world, we are motivated to propose an efficient and lightweight CNN model for histopathological image classification based on MobileNet that can be utilized efficiently on edge devices and is much smaller than VGGNet-16 as well as other state-of-the-art CNNs while having higher accuracy. The unique feature of this study is the integration of a lightweight CNN for the automatic recognition of cancer images on resource-constrained mobile and IoT devices.

6.2 Motivation and Significant Contribution

Deep learning has already proved its success for wider applications in computer vision and medical image analysis. Various researchers have analyzed and demonstrated its applicability in medical diagnoses. Previous studies, specifically on the histopathological image classification of breast cancer, have many limitations, which motivated us to

engage in this work. The first limitation in the existing literature on histopathological image classification is of employing machine learning methods such as Random Forest and SVM that rely on domain-related features. However, some studies [216] have been conducted to mitigate this problem by incorporating the automatic feature extraction capability of CNN. However, they are computationally expensive and have enormous parameters; therefore, they are not applicable directly for resource-constrained devices. Recently in the edge computing world, providing a low-cost solution has been highly valuable. To the best of my knowledge, this is the first study to develop the model for histopathological image classification that works effectively on edge devices. With this motivation, we propose an efficient and lightweight CNN model for histopathological image classification by utilizing CNN MobileNet [217]. We have also checked the efficiency of the proposed lightweight model on IoT devices.

The major contributions of this chapter are as follows:

1. We propose "MobiHisNet", an efficient and lightweight deep learning model that uses a series of depth-wise separable convolutions to reduce computational parameters, similar to MobileNet, which has been used as a feature extractor for histopathological images. In addition, a deep neural network has been used for classification.
2. Further, we have performed post quantization to compress the model and reduce its inference time on the edge device. The model has been quantized with Floating Point 32 (FP32), Floating Point 16 (FP16), and Integer 8 (INT8). It helps in achieving an optimal solution in terms of inference time, memory storage, and memory requirements.
3. Finally, the proposed model was tested on a Raspberry Pi and three different smartphones to demonstrate its efficiency on a lightweight processor. We also validated its efficacy in terms of different performance metrics suitable for edge-based applications, such as accuracy, inference time, memory peak, model size,

model parameters, and FLOPs counts. Our proposal speeds up histopathological image classification significantly without sacrificing accuracy, allowing it to compete with other models.

6.3 Theoretical Background

Nowadays, researchers are interested in mobile-based vision applications and are making continuous efforts to design CNN based models to enhance the performance while reducing the computation and communication cost for quick response. Basically, mobile-based vision applications can be subdivided into model-based vision application [118], device-based mobile system [218] and edge computing based paradigm [219, 220].

So, in this section, we will describe edge computing paradigms, deep compression, and MobileNet in order to provide a deeper understanding of the proposed framework.

6.3.1 Edge-based computing paradigms

Edge computing [13] presents a new computing paradigm that allows computation to be performed at the edge of the internet. In this paradigm, substantial computing and storage resources are placed in close proximity to mobile devices, end-users, sensors, and IoT devices at the edge of the internet. Latency, bandwidth, confidence, and survival are improved by this physical proximity. "Edge" devices can be any computing and networking resources located along the path between data sources and cloud data centres. Any sensing equipment such as smartphones, PDAs, tablets and smartwatches that collect sensor information such as images, audio and video can be the data source. The cloud data centre is equipped with powerful servers capable of performing complicated data processing and computing. We can address the critical issues [221] of response time requirements, battery life constraints, bandwidth cost savings, as well as data security and privacy by using the computing capacity of the edge devices.

6.3.2 Deep compression

Deep learning models usually require huge parameter computations, which make the model computationally expensive and dependent on high-performance computational resources. Due to these limitations, the extension of deep networks on edge devices or mobile applications that are limited to battery and memory constraints is very challenging. Also, the level of difficulty also increases when medical data is considered. Therefore, efficient, lightweight deep models are required, which can be archived through compressing deep learning models via network pruning, quantization, etc. Network pruning is the process of removing redundant connections and retaining the only most informative connection as illustrated in Fig. 6.1.

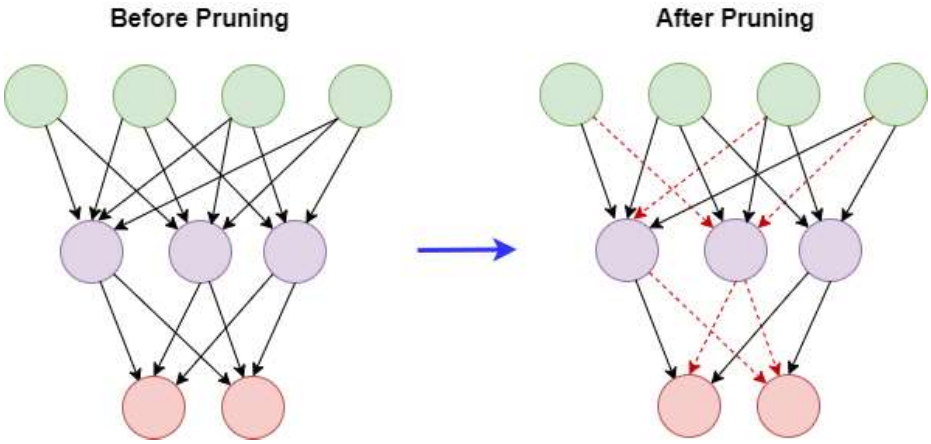


Figure 6.1: Pruning

This is studied widely in the earlier works [222, 223]. During the last few years, some works have also been reported to address the network pruning of state of the art CNN, as well as large scale CNNs [224] without the loss of accuracy. EdgeCNN [225] is also proposed to present an efficient network on edge devices. Similarly, in [224] authors have used network pruning in which all redundant connections with weights below a certain threshold were removed to compress the network, followed by network quantization which was applied along with Huffman coding to further compress the network. Mostly, high computations are due to convolutional layers, and large parameters are due to fully

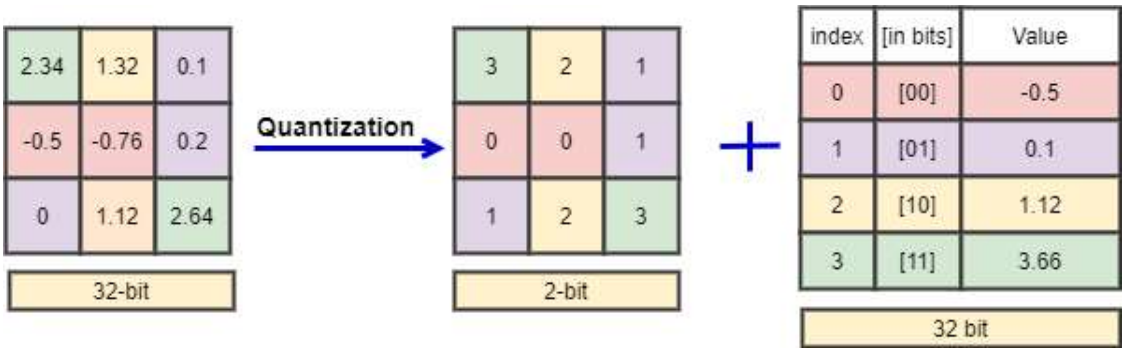


Figure 6.2: Quantization

connected layers; hence, the majority of work related to deep compression addresses either one of these issues.

Nowadays, quantization is applied on the pruned network that reduces the number of bits required to represent weights without interfering accuracy. For example, Fig. 6.2 demonstrates the quantization process by converting 32-bit representation into 2-bit representation. In this direction, quantization based model Q-CNN [226] is proposed where they carefully quantized both convolution layer and fully connected layer to present an accelerated, compressed CNN model. They have also shown that directly minimizing approximation error in the response of each layer leads to better quantization. The quantization process can be formulated as an optimization problem and works differently based on the objective function considered for optimization, which is described in [226]. The compression of the network is measured by using compression rate (C_R), which can be illustrated as: for a given k clusters $log_2(k)$ bits are needed to code the index. In the case of a network with m connections and n bits are needed to represent each connection. Limiting the connection to only k shared weights will result in C_R, which is presented by the following equation:

$$C_R = \frac{mn}{n \log_2(k) + kn} \tag{6.1}$$

In our study, we investigated the on-device inference performance using the quantization concept on a fine-tuned network based on MobileNet [217].

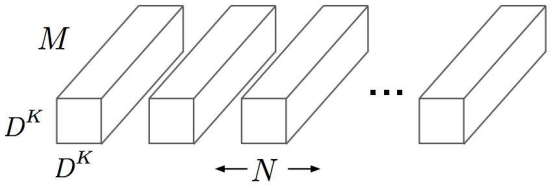
6.3.3 MobileNet

MobileNet [217] is a lightweight, efficient CNN designed for edge devices and mobile-vision applications. It effectively finds the tradeoff between latency/computation and accuracy by using a width multiplier. Depthwise separable convolution, which was originally introduced in [227] is the essential building block of efficient CNNs. MobileNet is also based on depthwise separable convolution that significantly reduces the parameters and computations. Since the network architecture for our study is based on MobileNet, we will first discuss depthwise-separable convolution to have a better understanding before moving on in this chapter.

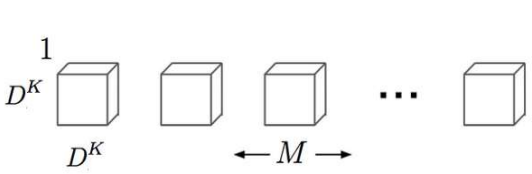
6.3.3.1 Depthwise-separable convolutions

The fundamental concept of depthwise-separable convolutions is similar to that of factorized convolutions. As a result, it subdivides standard convolution operations into two suboperations: depthwise convolution and pointwise convolution, as shown in Figure 6.3. As compared to standard convolution, which performs filtering and combining input into a new output set in a single step, the depthwise separable convolution divides it into two layers. Depthwise convolution is responsible for lightweight filtering by applying a single filter to each input channel, which is then combined by pointwise convolution by applying 1×1 convolution as shown in Figure 6.4. The size of the generated feature maps is the same in both standard convolution and depthwise separable convolution, but the number of parameters and computation is different. This factorization greatly reduces the computation and model size.

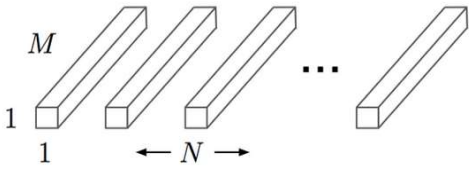
To understand the cost reduction in computation, let us assume M,N be the number of input channels and the number of output channels, respectively. D^K is the spatial dimension of the kernel K_r, which is assumed to be a square. We also assume that the output feature map has the same spatial dimensions D^F as the input and both the feature maps are in square shape. So, mathematically depth-wise convolution is defined



(a) Standard Convolution



(b) Depthwise Convolution



(c) Pointwise Convolution

Figure 6.3: Convolution operations

as the channel-wise $D^K \times D^K$ spatial convolution.While pointwise convolution is used to change the dimension that is 1×1 convolutions. Here kernel $K_r = D^K \times D^K$ and feature map $F = D^F \times D^F$ are used for further computations. In the case of standard convolution, standard convolution cost C_{std} is denoted as:

$$C_{std} = D^K.D^K.M.N.D^F D^F \tag{6.2}$$

However, in the case of overall operation cost (C_{dws}) for depth-wise separable convolutions is represented as :

$$C_{dws} = Depthwiseconvolutioncost + Pointwiseconvolutioncost \tag{6.3}$$

$$C_{dws} = D^k.D^k.M.D^F.D^F + M.N.D^F.D^F \tag{6.4}$$

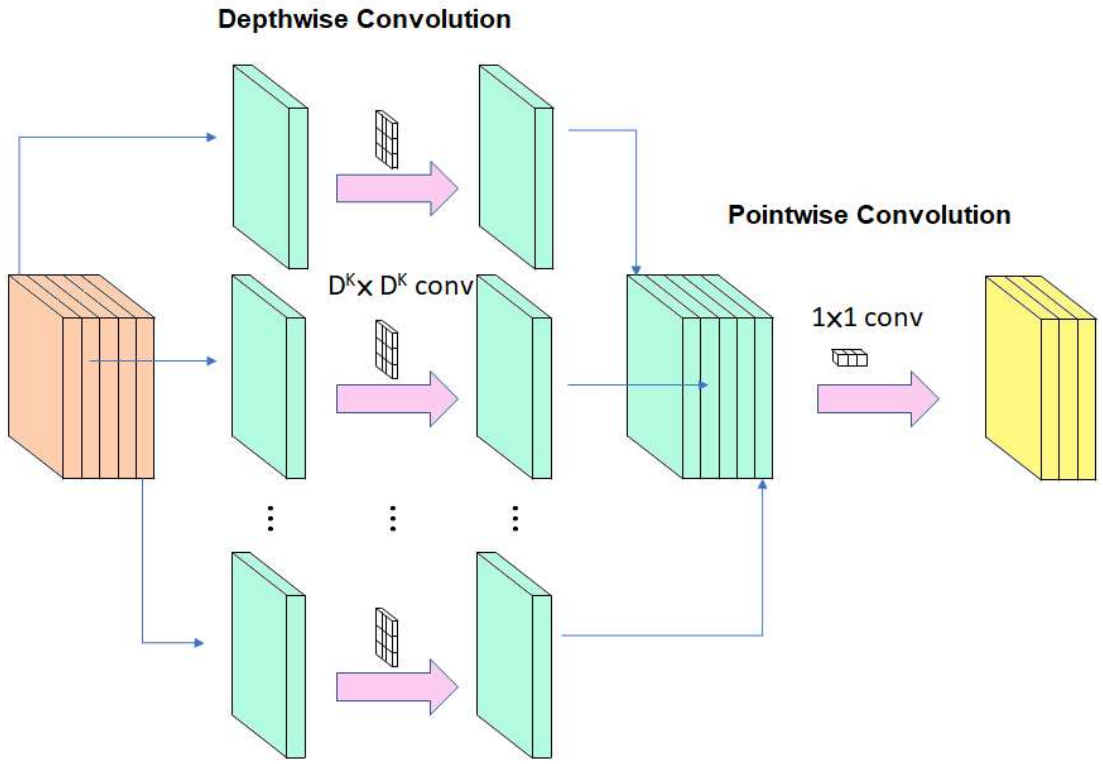


Figure 6.4: Illustration of depthwise separable convolution consisting of depthwise and pointwise convolution.

Thus, the computation reduction is computed as the ratio of C_{dws} to C_{std}.

$$\begin{aligned} \frac{C_{dws}}{C_{std}} &= \frac{D^k \cdot D^k \cdot M \cdot D^F \cdot D^F + M \cdot N \cdot D^F \cdot D^F}{D^k \cdot D^k \cdot M \cdot N \cdot D^F \cdot D^F} \\ &= \frac{1}{N} + \frac{1}{(D^k)^2} \end{aligned} \tag{6.5}$$

Input width of a layer and input image resolution of the network are controlled by a width multiplier γ and resolution multiplier δ, respectively. In that case, overall computation cost in the depthwise separable network can be written as:

$$C_{dws} = D^k.D^k.\gamma.M.\delta.D^F.\delta.D^F + \gamma.M.\gamma.N.\delta.D^F.\delta.D^F \tag{6.6}$$

In case of MobileNet baseline γ and δ are 1.

6.3.3.2 Linear bottleneck

Linear bottleneck is introduced in CNN to resolve the issue of information loss incurred by activation functions like rectified linear unit (ReLu) that performs nonlinear transformation of the input feature map. It increases the sparsity and reduces the chances of model overfitting, but it also causes large information losses for features with small channels. As a result, a linear bottleneck is required, so that feature maps are sent directly to the next convolutional layer after the pointwise convolutional layer and batch normalization without applying any nonlinear activation function.

6.4 Proposed Methodology

This section presents an overview of the proposed model "MobiHisNet" for histopathological image classification on the edge device. The development of the proposed model is a three-step process, as depicted in Fig. 6.5. In the first step, we propose the lightweight deep learning model for the histopathological image classification problem. The second step performs the post-training model optimization to reduce the resource requirement (such as storage size, runtime memory, power consumption, etc.), which enhances the inference time of the model on the edge device. Finally, the proposed MobiHisNet is evaluated on different smartphones and a Raspberry Pi to select the best-optimized model for histopathological image classification.

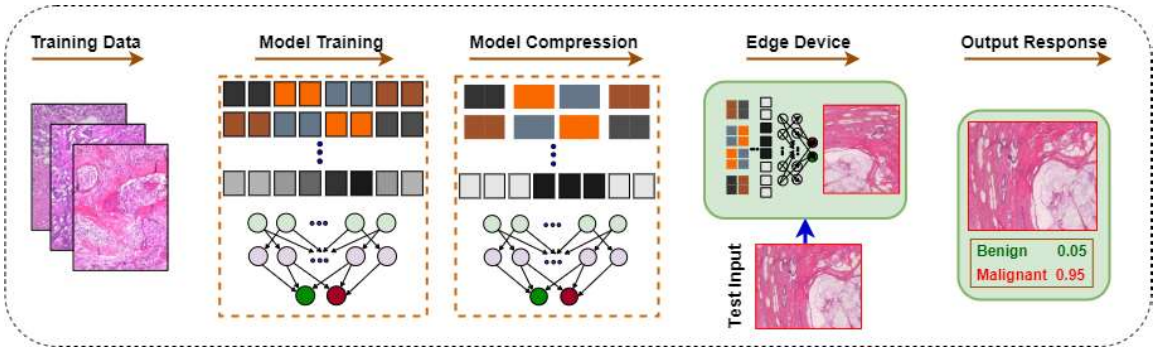


Figure 6.5: Proposed framework

6.4.1 Model architecture

The MobiHisNet is designed by utilizing the inline structure of MobileNet, as depicted in Fig. 6.6. Basically, the architecture of the model follows a two-step process: feature extraction and classification. Feature extraction is an essential part of any classification problem. It becomes even more critical for edge devices, where a highly computationally efficient method is required. Therefore, the feature extraction part is performed by utilizing the depthwise separable convolution, which combines two stacked operations. Firstly depthwise convolution applies the filter on each input channel, and second, pointwise convolution combines the output of depthwise convolution by applying 1×1 convolution.

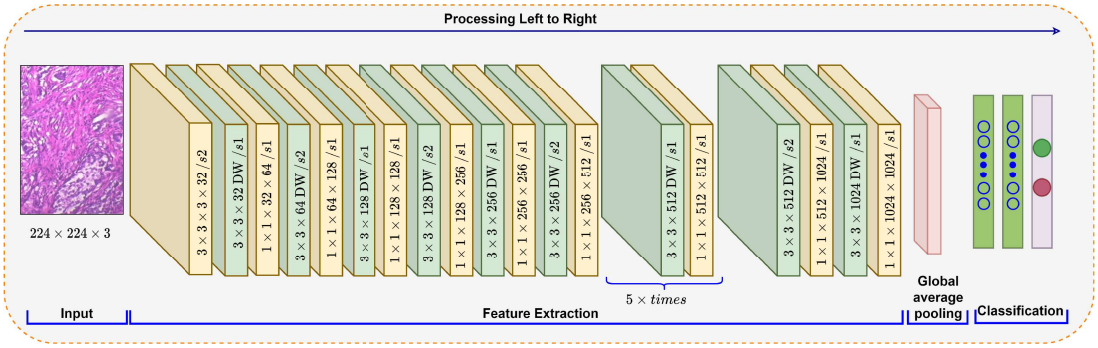


Figure 6.6: Proposed model architecture

Mathematically, we can explain the whole process by assuming that features maps are square metrics and output feature maps have the same spatial dimension as the input. Furthermore, to understand the cost reduction in computation, let us assume that M and N are the number of input channels and number of output channels, respectively. W^K is the spatial dimension of the kernel K_r, which is assumed to be a square. We also assume that the output feature map has the same spatial dimensions W^F as the input and both the feature maps are in square shape. So, mathematically, depth-wise convolution is defined as the channel-wise $W^K \times W^K$ spatial convolution. Pointwise convolution is applied to change the dimension that is 1×1 convolutions. Here kernel $K_r = W^K \times W^K$ and feature map $F = W^F \times W^F$ are used for further

computations. Therefore in the case of standard convolution, standard convolution cost C_{std} is denoted as:

$$C_{std} = W^K.W^K.M.N.W^F.W^F \tag{6.7}$$

However, the overall operation cost (C_{dws}) for depth-wise separable convolutions is represented as :

$$C_{dws} = \begin{cases} \text{Depthwise convolution cost} \\ + \\ \text{Pointwise convolution cost} \end{cases} \tag{6.8}$$

$$C_{dws} = W^k.W^k.M.W^F.W^F + M.N.W^F.W^F \tag{6.9}$$

Thus, the computation reduction is computed as the ratio of C_{dws} to C_{std} as shown below:

$$\begin{aligned} \frac{C_{dws}}{C_{std}} &= \frac{W^k \cdot W^k \cdot M \cdot W^F \cdot W^F + M \cdot N \cdot W^F \cdot W^F}{W^k \cdot W^k \cdot M \cdot N \cdot W^F \cdot W^F} \\ &= \frac{1}{N} + \frac{1}{(W^k)^2} \end{aligned} \tag{6.10}$$

Moreover, the input width of a layer and input image resolution of the network are controlled by a depth multiplier γ and resolution multiplier δ, respectively. In that case, overall computation cost in the depthwise separable network can be written as:

$$C_{dws} = W^k.W^k.\gamma.M.\delta.W^F.\delta.W^F + \gamma.M.\gamma.N.\delta.W^F.\delta.W^F \tag{6.11}$$

γ and δ are supposed to be 1 for the baseline MobileNet.

The proposed model uses a series of depthwise separable convolution operations followed by global average pooling to extract 1024 features from the input histopathological image, having a size of $224 \times 224 \times 3$ (150528 point features). Subsequently, two fully connected layers of size 1024 and 512 are added to produce a higher-level feature representation that is more appropriate for discriminating the classes. Finally, one more fully connected layer is added to perform the classification task.

6.4.2 Model compression

The main challenge in developing a model for edge devices is resource constraints, such as limited memory and computation power. Sometimes, they also have downloading bandwidth issues. The model size depends on the number of trainable parameters; the fewer are the parameters, the smaller the size. Moreover, the model with fewer parameters requires less power and memory than a bigger network with a high number of trainable parameters. Therefore, after training the proposed model, the next step is to optimize the model to support the above constraints of edge devices and improve the inference time. In literature, pruning and quantization are two popular techniques for this task.

Pruning reduces the number of parameters by eliminating redundant and insignificant connections between network nodes that are not performance-sensitive. Basically, the pruning approach tries to find out more informative nodes in the network that need to be preserved and eliminate the nodes with less or no significant contribution to the model output. This activity is usually performed based on information collected during the model's training or by re-training the model.

On the other hand, quantization reduces the model size by representing the weight matrix with a lesser bit representation. Typically, the DNN model is stored in a 32-bit floating-point representation. However, through quantization, it can be reduced to 16-bit, 8-bit, or even 1-bit floating point, as well as in integer representation. In our

proposed framework, we have adopted the quantization approach for model compression, and the best representation is selected for histopathological image classification via empirical analysis on edge devices.

6.5 Experimental Results and Analysis

This section demonstrates the performance evaluation of the proposed model on the BreakHis dataset. After thoroughly validating the efficacy of the proposed model on the BreakHis dataset, we applied it to the CMTHis dataset using a Raspberry Pi. First, we provide the implementation details, including hardware and software, followed by hyperparameters and evaluation metrics. Next, we present the experimental results on edge devices and also on high-end computer systems. Models training are performed on a GPU Node with a 2.4 GHz Intel-Xeon Skylake 6148 processor, 192 GB RAM and a 16 GB HBM2 NVIDIA V100 PCIe accelerator card.

6.5.1 Experimental setup

6.5.1.1 Hardware

The proposed model is tested on four resource-constrained devices, including three smartphones -Redmi 4A, OPPO F11, Samsung A51, and a Raspberry Pi. The Redmi 4A has a low hardware specification in comparison to the OPPO F11 and Samsung A51. Redmi 4A is equipped with a quad-core processor and 2GB memory, while the other two mobile devices have an octa-core processor with 6GB memory. The detailed hardware configuration is provided in Table 6.1.

6.5.1.2 Software

The proposed framework has used the Tensorflow-lite [228] library to perform the model compression with different quantization parameters. TensorFlow benchmark tool [229],

Table 6.1: Mobile devices used in the experiments

Mobile Device	OS Version	Processor	CPU	GPU	Memory(GB)
Redmi 4A (D1)	Android 6	Snapdragon 425	Quad-core	Adreno 308	2GB
Oppo F11 (D2)	Android 10	Mediatek MT6771	Octa-core	Mali-G72 MP3	6GB
Samsung A51 (D3)	Android 10	Exynos 9611	Octa-core	Mali-G72 MP3	6GB

a C++ binary, was used to profile the model and its component's execution time and maximum memory consumption on the target edge device. The benchmark tool has one warm-up run for each model in order to initialize the model and then profile all components execution time with 50 runs without internal delay. Furthermore, the mean values are used to calculate the profiled execution time. Devices D1, D2 and D3, have android versions of 5.0, 10, and 10, respectively.

6.5.1.3 Hyperparameter setting

Initially, in the proposed model, CNN layers of MobileNet are initialized with pre-trained weight on the Imagenet dataset, and other layers have been initialized with Xavier Uniform (XU) initializer [230]. Basically, the XU initializer draws the samples from a uniform distribution within limit $[-\sqrt{(6/(P+Q))}, \sqrt{(6/(P+Q))}]$, where P and Q are the input and output units in the weight tensor of each layer. We have used the SGD optimizer to train the proposed model with a learning rate of 0.0001 and batch size 32. Further, to select the best model for the histopathological image classification, we have considered the depth multiplier $\gamma \in \{1.0, 0.75, 0.50, 0.25\}$.

6.5.2 Evaluation metrics

The following metrics were used to assess the performance of this study:

1. **Accuracy**: It measures the correctly classified samples over total samples in the testing set. This evaluation metric tells about "how accurate the predicted output

of a model is".

2. **Inference time**: This is the time on the edge device taken by the model between accepting desired input and providing the evaluated output. It tells us how fast the model is.

3. **Memory Peak**: It is the measurement of maximum memory used by the model, anytime during inference.

4. **Model size**: It tells about the storage requirement for the model on the edge device. It affects device storage as well as downloading bandwidth.

5. **Model parameters**: It is the indicator of execution time, which works as the rule-of-thumb to estimate "how fast the model will be on edge devices" [231].

6. **FLOPs counts**: FLOPs is the acronym of Floating-point operations per seconds. It is a platform-independent measure of computation of the model. It also works as a rule of thumb to estimate "how fast the model is" [231].

6.5.3 Result analysis and discussion

This section presents an analysis of the results. In this chapter, we have compared the proposed model with six baseline models including VGGNet-16 [115], ResNet50 [103], Xception [232], InceptionV3 [233], MobileNetV1 [217], and EffNetB0 [234]. These baseline models have been designed by considering pre-trained models on the Imagenet dataset. Firstly, the top classification layer has been removed from the base pre-trained model, and then two additional layers (one global average pooling and one fully connected layer) have been added on the top of the base model. Moreover, we have frizzed all the baseline pre-trained model layers and trained only the last output layer to get the classification result.

6.5.3.1 Performance analysis

The proposed model has been evaluated on the BreakHis histopathological image dataset, with 40X, 100X, 200X, and 400X magnifications. Each magnification has been divided into five-folds, with each fold having a separate train and test sets. The models are evaluated separately on each fold. Table 6.2 demonstrates the proposed model's comparative study with other baseline models by considering the metrics accuracy, model size, FLOPs counts, and model parameters. The best value of the parameter is highlighted in Table 6.2. It is observed that the VGGNet-16 model is expensive in terms of FLOPs counts, while EffNetB0 is highly computationally efficient. ResNet50 is a large-size model having 23.59 million parameters and model size of 94.89 MB. However, ResNet50 is computationally effective compared to VGGNet-16 and Xception. MobileNetV1 is the most lightweight model among all the models. Furthermore, It was observed that the proposed model achieved the highest accuracy over all the magnifications compared to other baseline models. Moreover, the proposed model provided the highest accuracy on 200X magnification. We also observed that the proposed model is computationally faster when compared to VGGNet-16, ResNet50, Xception, and InceptionV3 by twenty-seven times, eight times, six times, and five times, respectively. Table 6.2 demonstrates that the proposed model outperformed all baseline models while having a moderate model size and FLOPs counts.

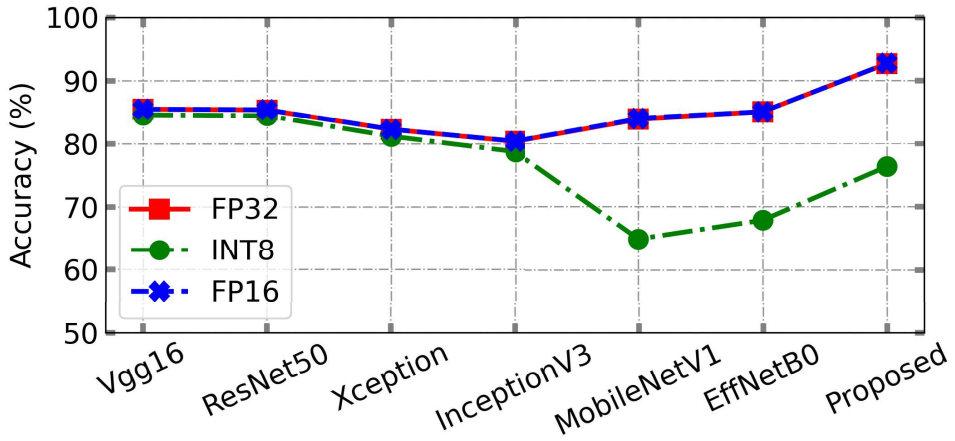
6.5.3.2 Effect of quantization

In our experimental work, we have utilized the TensorFlow python library to implement the model that usually represents the model weight parameters in a 32-bit floating-point. It is observed that model quantization with 16-bit floating-point reduces the model size approximately by half without a significant loss in accuracy value. In contrast, 8-bit integer representation minimizes the size of the model by roughly a quarter. We also analyzed that 8-bit quantization harms accuracy on the models having a size of less

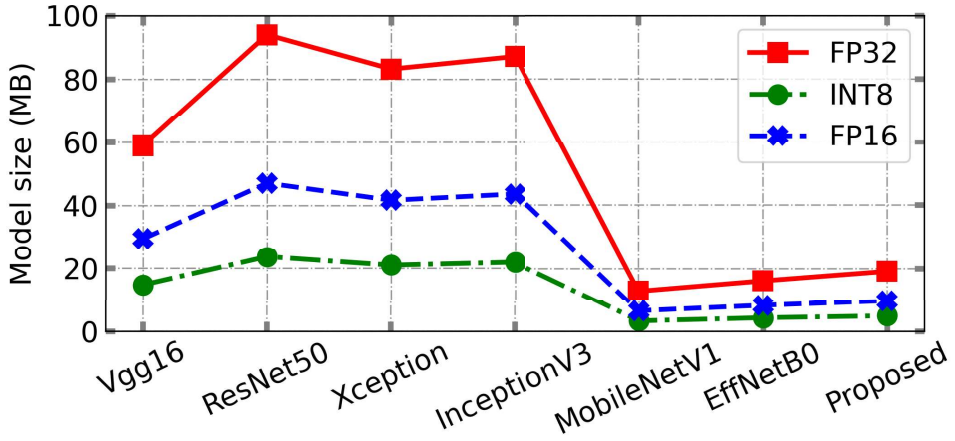
Table 6.2: Comparative analysis of proposed MobiHisNet with state-of-the-art pretrained models.

Model Name	Accuracy (%)								Model Size (MB)	Model parameters (million)	FLOPs (million)
	40X		100X		200X		400X				
	Mean	SD	Mean	SD	Mean	SD	Mean	SD			
VGGNet-16	82.90	1.24	86.87	2.09	84.87	2.07	83.19	2.00	58.94	14.72	30712.96
ResNet50	81.01	4.80	86.10	4.50	85.08	4.08	83.51	2.08	94.89	23.59	7724.35
Xception	78.93	2.99	81.51	3.85	82.43	1.23	80.49	1.61	83.87	20.87	9108.80
InceptionV3	80.01	1.60	80.32	3.83	80.90	0.96	79.92	2.22	88.10	21.81	5683.98
MobileNetV1	83.78	2.32	83.63	3.33	83.49	2.02	81.03	2.94	**13.19**	**3.23**	1135.49
EffNetB0	85.25	1.65	86.74	2.74	85.21	4.24	80.94	3.66	16.85	4.05	**780.37**
Proposed	**90.82**	3.49	**87.61**	3.14	**90.89**	3.57	**83.70**	2.65	38.79	4.80	1138.63

than 20 MB. It is because model size is already reduced by 3-5 times by utilizing depth-wise separable convolution operation, compared to VGGNet-16 and ResNet50 models. Hence, further reduction in the precision of weight matrix values causes a high fall in accuracy value. Histopathological image classification is a critical problem and requires a highly reliable prediction. Therefore, based on empirical analysis, 16-bit floating-point quantization of the proposed model is the right choice for histopathological image classification. Fig. 6.7 demonstrates the quantization effect on the model's accuracy as well as on model size.



(a)

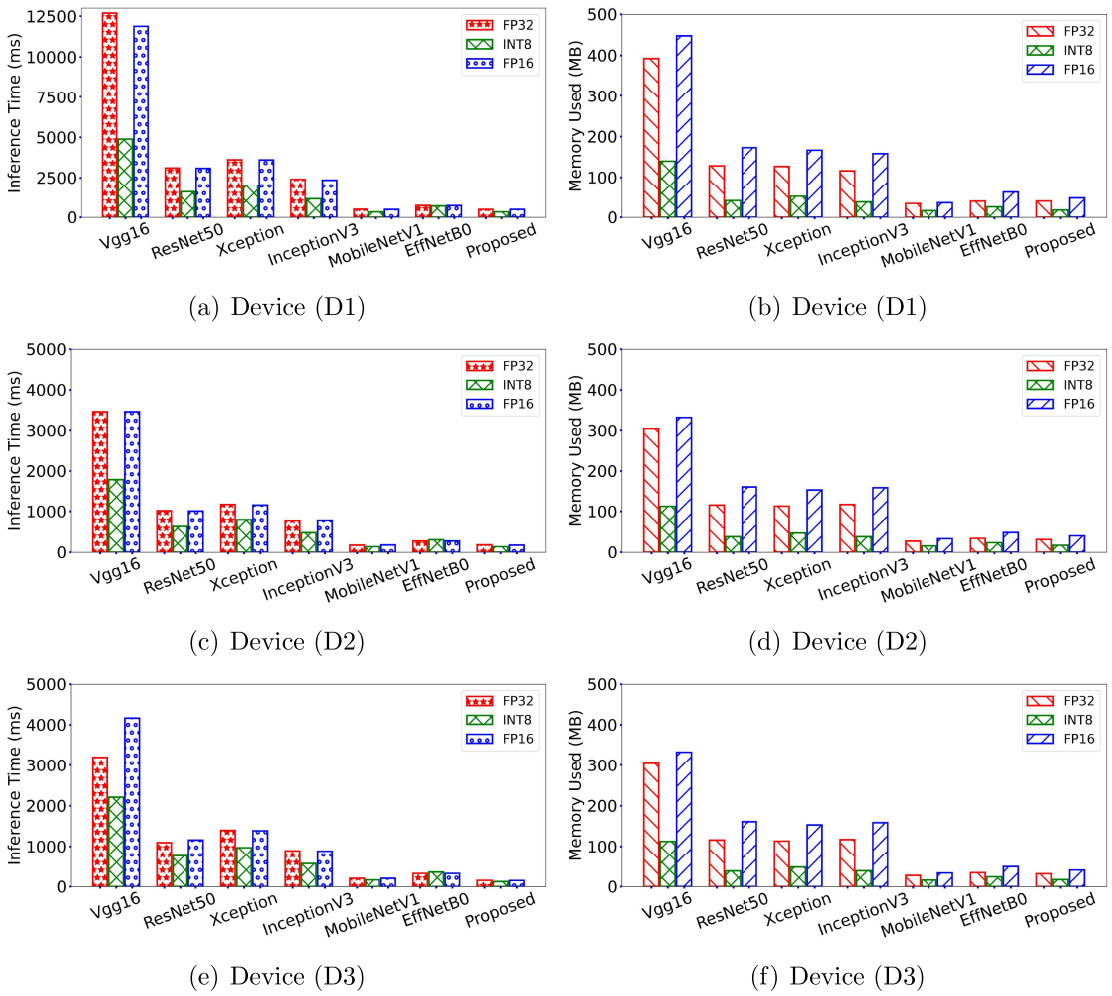


(b)

Figure 6.7: Effect of model compression on the accuracy, and model size.

6.5.3.3 On-device effect

The on-device effect of quantization has been analyzed on three resource-constrained devices. The details about the devices have been provided in Table 6.1. Further, two critical performances metrics, namely inference time and maximum memory requirement of the model, have been considered for comparative analysis. These metrics have been captured through the profiling tool Tensorflow Benchmark tool [229]. From Fig. 6.8 it is clearly observed that 8-bit integer representation is highly efficient and requires the lowest memory and inference time, compared to 16-bit and 32-bit representation.



(a) Device (D1) (b) Device (D1)

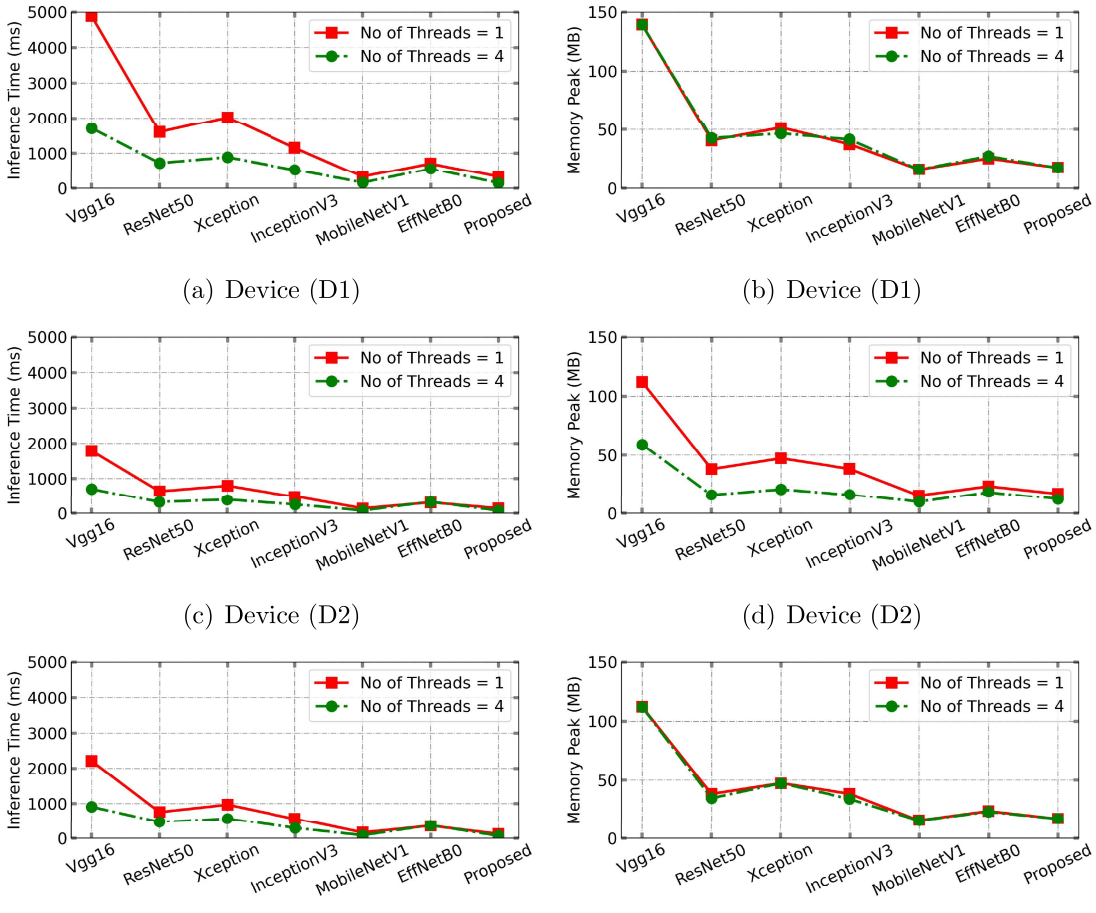
(c) Device (D2) (d) Device (D2)

(e) Device (D3) (f) Device (D3)

Figure 6.8: Effect of model compression on inference time as well as on maximum memory used.

It is also seen that the maximum memory utilization parameter for the model does not affect significantly on different devices, as shown in Fig 6.8(b), 6.8(d), and 6.8(f). In contrast, inference time highly depends on mobile hardware. The inference time of device D1 is very high compared to D2 and D3 as shown in Fig. 6.8(a), Fig. 6.8(c), and Fig. 6.8(e). This is because D1 has low hardware specifications as compared to other devices.

Further, the impact of the selection of threads has been demonstrated in Fig. 6.9. It is observed that the inference time of the models reduced approximately by half times on all the devices while considering four threaded executions in place of a single thread.



(a) Device (D1) (b) Device (D1)

(c) Device (D2) (d) Device (D2)

(e) Device (D3) (f) Device (D3)

Figure 6.9: Effect of thread selection on inference time as well as on maximum memory used.

However, the inference time of the model on device D1 is high compared to D2 and D3. Moreover, peak memory requirements have been decreased significantly on device D2 compared to devices D1 and D3. Thus, based on the above observations, it can be concluded that the thread selection depends on the hardware as well as software resources of the targeting device.

6.5.3.4 Ablation study

Ablation study has been performed by considering depth parameter, image level (different magnification levels) and model compression. First, the performance of depth parameter γ is analyzed for histopathological image classification. Table 6.3 demonstrates the impact of depth parameter γ on the proposed model. For $\gamma = 0.5$, it is observed that the model size, model parameters, and FLOPs counts have been reduced by 2.5, 2.5, and 3.8 times, respectively, without any loss in accuracy on 200X as well as on 400X magnifications. Fig. 6.10 represents the on-device performance of the proposed model with different values of γ. It is observed that for $\gamma = 0.75$, the inference time, as well as memory peak requirement, have been reduced by approximately 1.6 and

Table 6.3: Ablation study of proposed model with depth parameter γ

Proposed Model	**Accuracy**								**Model Size(MB)**	**Parameters (million)**	**FLOPs Counts (million)**
	40X		**100X**		**200X**		**400X**				
	Mean	SD	Mean	SD	Mean	SD	Mean	SD			
Proposed (γ=1.0)	90.82	3.49	87.61	3.14	90.89	3.57	83.70	2.65	38.79	4.80	1138.63
Proposed (γ= 0.75)	88.03	4.39	87.14	3.37	88.14	3.03	84.03	2.03	25.56	3.15	651.93
Proposed (γ= 0.50)	88.49	3.75	86.79	3.38	91.21	3.54	84.43	2.71	15.44	1.88	300.10
Proposed (γ= 0.25)	85.94	3.61	82.41	3.76	89.75	2.45	83.68	3.08	8.48	1.01	83.14

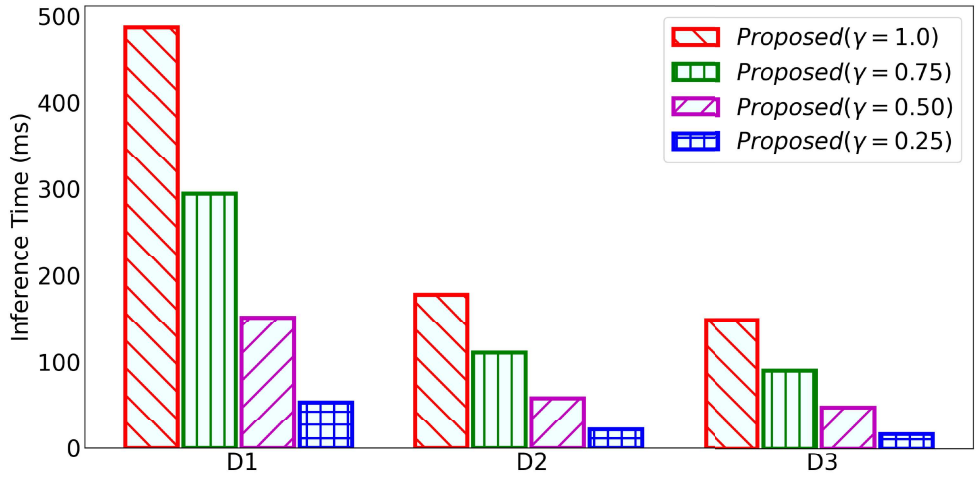
Table 6.4: Ablation study of proposed MobiHisNet with state-of-the-art pretrained models

Model Name	Model Compression	Accuracy (%)							
		40X		100X		200X		400X	
		Mean	SD	Mean	SD	Mean	SD	Mean	SD
VGGNet-16	FP32	82.53	1.93	86.60	1.99	85.48	2.78	82.61	1.91
	INT8	82.94	1.46	86.57	2.02	84.57	2.13	82.56	1.07
	FP16	82.53	1.93	86.60	1.99	85.48	2.78	82.65	1.91
ResNet50	FP32	83.30	2.96	86.28	4.43	85.36	3.53	84.16	2.19
	INT8	83.34	2.66	85.85	4.23	84.44	3.36	83.53	2.35
	FP16	83.30	2.92	86.31	4.39	85.39	3.56	84.12	2.18
Xception	FP32	78.79	2.97	81.59	3.78	82.27	1.19	80.55	1.34
	INT8	77.49	4.01	80.20	3.66	81.17	1.27	80.38	0.80
	FP16	78.77	2.94	81.59	3.69	82.27	1.19	80.52	1.46
InceptionV3	FP32	80.43	0.88	80.79	3.12	80.35	1.15	79.66	1.66
	INT8	79.49	1.26	79.08	2.65	78.73	1.50	78.26	2.21
	FP16	80.46	0.88	80.73	3.18	80.38	1.27	79.69	1.71
MobileNet	FP32	84.41	2.20	83.76	3.27	83.89	1.80	81.28	2.84
	INT8	57.83	6.50	62.45	13.79	64.85	3.53	66.37	3.03
	FP16	84.36	2.36	83.70	3.41	83.99	1.82	81.31	2.87
EffNetB0	FP32	85.11	1.57	86.58	2.68	85.03	4.20	80.94	3.53
	INT8	42.83	11.23	66.64	1.53	67.82	4.93	65.37	3.56
	FP16	85.05	1.62	86.62	2.70	85.03	4.19	80.91	3.48
Proposed (γ=1.0)	FP32	91.36	2.04	89.98	2.40	92.67	1.95	85.95	1.95
	INT8	72.23	0.74	73.06	2.89	76.32	5.79	74.22	3.32
	FP16	91.42	1.99	89.93	2.40	92.70	1.87	85.84	1.88
Proposed (γ= 0.75)	FP32	89.85	2.96	88.19	2.34	89.23	1.97	84.62	1.04
	INT8	80.18	3.25	72.90	4.27	73.99	5.25	72.00	2.42
	FP16	89.77	2.91	88.33	2.27	89.19	1.93	84.49	0.87
Proposed (γ= 0.50)	FP32	89.93	2.68	87.47	2.55	91.50	2.67	84.11	2.03
	INT8	71.71	10.96	72.72	3.37	73.90	6.00	71.69	2.95
	FP16	89.94	2.75	87.50	2.49	91.51	2.68	84.15	2.09
Proposed (γ= 0.25)	FP32	87.43	2.23	84.29	3.35	89.55	2.19	83.41	2.32
	INT8	61.60	13.65	69.26	1.79	66.80	2.14	66.06	3.72
	FP16	87.31	2.16	84.42	3.31	89.59	2.06	83.46	2.19

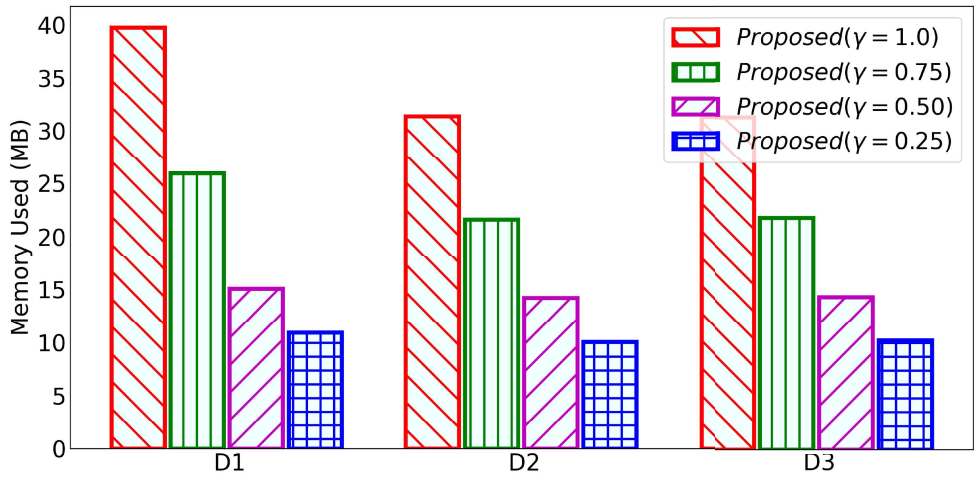
Table 6.5: Ablation study of model comparison on device

Model Name	Model Compression	Inference Time (ms)						Peak Memory (MB)			Model Size (MB)
		Device (D1)		Device (D2)		Device (D3)		Device (D1)	Device (D2)	Device (D3)	
		Mean	SD	Mean	SD	Mean	SD				
VGGNet-16	FP32	12,680.80	1,378.07	3,459.37	34.36	3,194.24	114.28	392.04	303.99	304.08	58.87
	INT8	4,882.87	4.55	1,783.76	13.78	2,206.30	3.93	139.38	112.11	112.22	14.74
	FP16	11,873.00	27.96	3,457.04	14.22	4,163.93	35.86	447.11	332.03	332.18	29.45
ResNet50	FP32	3,099.78	15.85	1,011.50	7.51	1,090.04	117.58	127.92	115.35	115.55	94.01
	INT8	1,594.93	11.05	635.19	3.77	755.21	1.21	40.88	37.98	38.20	23.64
	FP16	3,079.41	33.04	1,004.15	4.91	1,155.62	3.22	172.82	160.37	160.57	47.07
Xception	FP32	3,607.28	2.77	1,167.86	13.81	1,393.44	6.57	126.63	112.67	112.81	83.22
	INT8	2,019.40	1.80	793.85	15.54	966.20	1.31	51.50	47.23	47.39	20.96
	FP16	3,590.53	17.13	1,150.10	9.04	1,381.80	3.19	166.33	152.53	152.61	41.67
InceptionV3	FP32	2,397.17	4.22	773.22	4.76	888.38	4.50	116.03	116.55	116.72	87.18
	INT8	1,149.97	4.39	477.07	2.24	561.24	0.65	37.58	38.18	38.31	21.92
	FP16	2,341.26	2.25	777.29	3.16	880.12	1.75	157.74	158.28	158.34	43.67
MobileNet	FP32	492.26	10.55	173.73	1.57	198.11	0.80	33.72	26.93	27.02	12.82
	INT8	329.91	6.82	133.90	1.62	159.13	0.48	16.21	15.41	15.55	3.27
	FP16	484.85	14.73	176.52	0.58	199.11	0.32	35.63	33.19	33.29	6.44
EffNetB0	FP32	734.33	2.11	273.64	3.04	320.96	0.67	39.53	33.89	34.05	16.09
	INT8	705.19	0.83	305.08	2.56	354.06	0.77	25.50	23.30	23.47	4.25
	FP16	717.79	1.53	273.07	1.63	321.07	0.94	62.23	48.34	48.47	8.17
Proposed (γ=1.0)	FP32	487.33	1.02	178.11	0.63	149.11	1.51	39.74	31.36	31.28	19.12
	INT8	328.20	0.54	135.14	2.21	117.92	0.42	17.76	16.88	16.93	4.85
	FP16	486.29	7.64	173.99	0.99	147.13	0.37	47.53	40.48	40.47	9.59
Proposed (γ= 0.75)	FP32	294.76	1.11	111.27	0.66	89.78	0.18	25.91	21.59	21.73	12.52
	INT8	236.01	0.26	102.29	0.86	90.92	0.53	14.14	13.26	13.38	3.20
	FP16	291.57	0.33	116.53	6.28	90.26	0.25	31.95	27.64	27.86	6.29
Proposed (γ= 0.50)	FP32	151.72	0.67	58.10	0.31	47.08	1.33	15.05	14.21	14.28	7.49
	INT8	121.70	0.33	50.19	0.26	42.83	0.14	9.87	9.05	9.13	1.93
	FP16	148.22	0.17	57.15	0.90	47.07	0.17	18.63	17.75	17.95	3.77
Proposed (γ= 0.25)	FP32	53.31	0.09	21.50	0.14	17.25	0.11	10.91	10.03	10.19	4.03
	INT8	50.76	0.09	22.19	0.15	18.35	0.10	8.15	7.26	7.34	1.06
	FP16	52.94	0.11	21.92	0.17	17.57	0.12	12.89	11.98	12.08	2.04

1.5 times, respectively, overall the devices. For $\gamma = 0.25$, inference time and memory peak requirement have been reduced by 8.5 and 3.3 times, respectively. Moreover, the improvement in inference time and memory requirement is significantly high on device D1 as compared to devices D2 and D3. Thus, based on this ablation analysis, it is found that $\gamma = 0.5$ and 16-bit quantization are preferable parameters for the proposed model, which balances the accuracy, inference time, and peak memory requirement.



(a)



(b)

Figure 6.10: Comparison of on device inference and maximum memory utilization.

Finally, detailed ablation results have been presented in Table 6.4 and Table 6.5.

Table 6.4 demonstrates ablation results of the model performance on compression with FP32, FP16, INT8 quantizations for histopathological image classification. The models are evaluated with 5-fold cross-validation, and the mean accuracy with standard deviation is recorded in Table 6.4. Table 6.5 demonstrates the ablation results of on-device performance of the models. Basically, it displays model inference time and memory peak that are measured through a profiling tool over 50 runs without internal delay. These measures are evaluated for model compression with FP32, FP16, INT8 quantizations.

6.5.4 Performance of the MobiHisNet on Raspberry Pi

After evaluating the proposed model's performance for histopathological image classification on three mobile devices on the BreakHis dataset, we further used the resource constraint IoT device, Raspberry Pi, to show the proposed model's efficacy on a lightweight processor using the BreakHis and CMTHis datasets. Fig 6.11 demonstrates the working of a histopathological image classification system.

Table 6.6 shows the performance of the proposed model on Raspberry Pi B+ with

Table 6.6: Results of the proposed model with FP16 on Raspberry Pi

BreakHis Dataset								
	40X		**100X**		**200X**		**400X**	
	Acc (%)	**Time (ms)**	**Acc (%)**	**Time (ms)**	**Acc (%)**	**Time (ms)**	**Acc (%)**	**Time (ms)**
γ=1.0	89.76	582.80	88.12	583.10	90.19	586.75	84.19	588.55
γ=0.5	89.60	180.34	86.94	180.95	88.43	181.15	83.79	181.65
CMTHis Dataset								
γ=1.0	87.62	583.50	88.95	584.20	85.42	587.55	80.29	590.40
γ=0.5	86.90	181.54	88.10	182.10	84.56	182.45	79.95	182.95

*Here Time refers to inference time

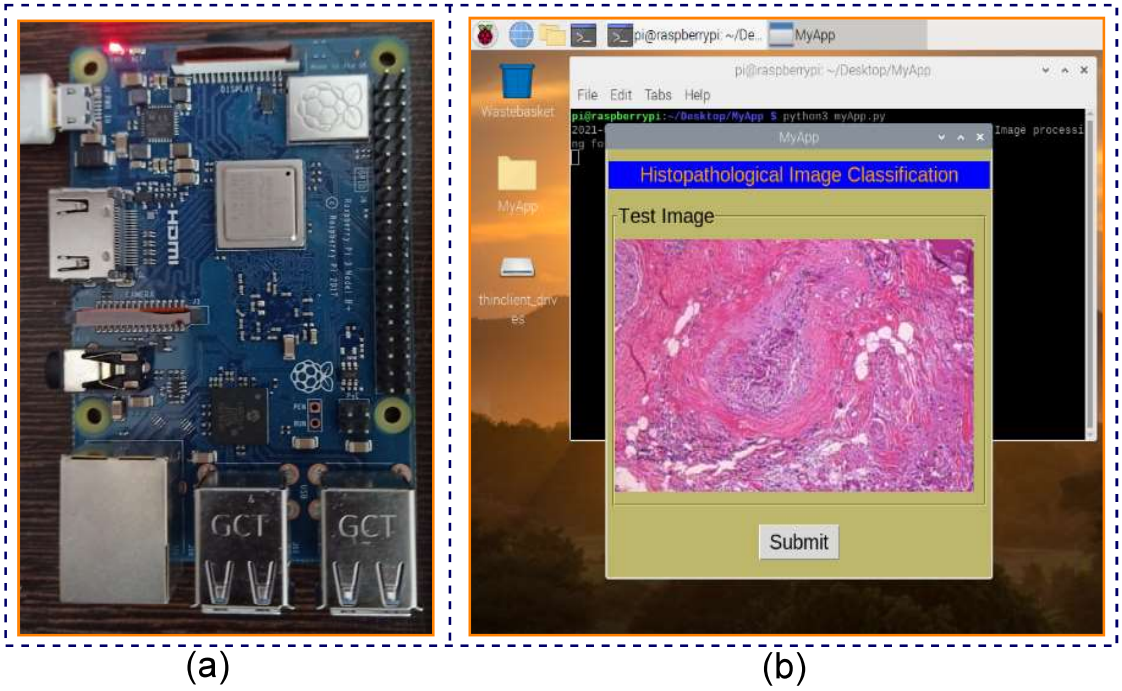


Figure 6.11: (a) Raspberry Pi 3 B+ Rev 1.3 (1.4 GHz quad-core BCM2835 ARMv7 CPU with 1 GB RAM) Connected to Power Supply (b) Histopathological image testing process on Raspberry Pi.

BreakHis and CMTHis datasets. The model has been tested by considering compression FP16 with depth parameters $\gamma \in (1.0, 0.5)$. It is observed that the proposed model with $\gamma = 0.5$ demonstrates decent performance for all magnifications and takes about 182 ms to inference the test image. The experimental results on different resource-constrained devices such as mobile devices and Raspberry Pi show the excellent efficacy of the proposed model for histopathological image classification.

6.6 Summary

Increased computational costs and high-resource requirements due to enormous parameters make deep learning models less affordable for edge-based applications. In the last few years, the demand for edge-based and IoT applications has increased, and it has become a necessity for various domains such as agriculture and medicine. In this

chapter, we have focused on the specific application of a deep learning model for diagnosing breast cancer in histopathological images. For breast cancer classification, we have proposed "MobiHisNet", an efficient, lightweight CNN model based on MobileNet. By applying the concept of depthwise separable convolution in the proposed framework for histopathological image classification, we have observed significant reductions in CNN parameters and computation costs. In order to design an efficient, lightweight framework, first, we fine-tuned the model to rectify the problems of histopathological image classification. This was followed by model compression, which was performed to get an optimal solution with respect to inference time, memory storage and memory requirements in the RAM. To evaluate the efficacy of our proposed model, we deployed it on four resource-constrained devices, including three different smartphones and a Raspberry Pi, and subsequently validated the model on the BreakHis dataset. This demonstrates the ability of MobiHisNet to run on lightweight and portable processors. The proposed model was extensively tested and evaluated comprehensively with respect to the accuracy, model parameters, memory peak, computation in terms of inference time, and FLOPs. Compared with the other state-of-the-art, pre-trained models for histopathological image classification, the proposed method demonstrated superior performance. Moreover, the proposed model is computationally less expensive, with fewer parameters and computations. This enhances the image classification process, thereby making it more viable for production purposes and applications on edge devices.