MobileNets: Depth wise Separable Convolutions for Faster Network Construction.

SHADHIN.MD.YOUSUF ALI
dept. Computer Science
AIUB
Dhaka, Bangladesh
20-42783-1@student.aiub.edu

PARTHA MALAKAR
dept. Computer Science
AIUB
Dhaka, Bangladesh
20-42908-1@student.aiub.edu

MST. JANNATUL FERDOUS SUMI
dept. Computer Science
AIUB
Dhaka, Bangladesh
20-43007-1@student.aiub.ed

ARGHO PROSHAD SINGHIO
dept. Computer Science
AIUB
Dhaka, Bangladesh
20-42906-1@student.aiub.edu

INTRODUCTION

MobileNets use depth wise separable convolutions to reduce compute time and predict performance. MobileNet is a type of convolutional neural network designed for mobile and embedded vision applications. They are based on a streamlined architecture that uses depth wise separable convolutions to build lightweight deep neural networks that can have low latency for mobile and embedded devices. For a long time, neural networks have been constructed. Based on the learned data, they can anticipate newer outputs quite effectively. But most of the predictions made up until now have been based on textual information. In recent years, using neural networks to photos (and videos) has been a popular area of study. To perform various operations on images, such as image categorization and object detection, to mention a few, numerous strategies have been established, and significantly more advanced techniques are being created every year.

This research paper presents an innovative network architecture, MobileNets, designed to address the challenges of efficiency in mobile and embedded vision applications. By incorporating a carefully crafted network design and two essential hyper-parameters, width multiplier and resolution multiplier, MobileNets enable the creation of compact models with minimal latency. These models can be easily tailored to meet the specific design requirements of resource-constrained platforms.

This paper demonstrates a better neural network design to reduce the above-mentioned trade-off. Section 2 describes the prior work done in this domain in the past. Section 3 describes the MobileNet design and architecture. Section 4 describes the results obtained by training MobileNet on CIFAR-10[2] dataset. Section 5 describes the performance prediction strategy and the various computations required by the network. Section 6 closes with the conclusion of the network.

RELATED WORKS

The historic LeNet[3], which was the very first genuine convolutional neural network that delivered accurate results on a limited MNIST dataset, served as an inspiration for MobileNet's design. The creation of AlexNet[4] earlier this year, which took first place in the ImageNet challenge: ILSVRC 2012[5], gave researchers new incentive to create more effective networks. This network is built on serial design, where input and output are processed in a fixed order. In spite of the fact that the network had substantially more parameters due to the fully linked layers, which made computations much slower, it significantly reduced the top-1 and top-5 error rates during ILSVRC 2012. MobileNet shares a sequential design foundation with AlexNet, although it employs many less parameters.
The paper begins by acknowledging the historical impact of CNNs, particularly with the success of AlexNet in the ImageNet Challenge. It highlights the subsequent trend of increasing network depth and complexity to achieve higher accuracy. However, it also emphasizes that this pursuit of accuracy often overlooks the crucial aspect of efficiency, which is vital in real-world applications like robotics, self driving cars, and augmented reality.To address these challenges, the paper presents MobileNets—a network architecture designed explicitly for resource-constrained platforms. MobileNets leverage a carefully crafted network design and two critical hyper-parameters, namely the width multiplier and resolution multiplier, to construct small and efficient models with minimal latency. This approach allows MobileNets to meet the specific design requirements of mobile and embedded vision applications.

A. ADVANTAGE

The MobileNets architecture offers several advantages for mobile vision applications. MobileNets optimize resource use by using lightweight network structures and techniques[1]. MobileNets prioritize low latency for quick processing of visual data in time-critical applications. MobileNets are compact, allowing for efficient storage, faster transfer, and reduced bandwidth requirements. MobileNets provide flexibility through hyper-parameters like the width multiplier and resolution multiplier. These parameters allow developers to adjust the model size and computational complexity according to specific application requirements and hardware constraints. This customization capability ensures that MobileNets can be optimized for a wide range of target devices and use cases. MobileNets deliver competitive accuracy due to balanced design and optimization techniques. The advantages of MobileNets translate into practical benefits for real-world mobile vision applications.

B. LIMITATION

While MobileNets offer numerous advantages for mobile vision applications, there are also some limitations to consider: MobileNets reduce complexity to improve efficiency, but may not be suitable for complex tasks. MobileNets may sacrifice accuracy to optimize efficiency and reduce computational requirements, depending on application. MobileNets face difficulty in fine-grained visual tasks due to reduced model complexity. Careful tuning and optimization of hyper-parameters is necessary to achieve desired performance.

PROPOSED MODEL

Design: The basic building blocks of the encoder (body) design of MobileNet are depth wise separable convolutions. These are composed of depth wise convolution and pointwise convolution. The depth wise convolution applies a single filter to each input channel. The pointwise convolution applies a 1X1 convolution and then adds the result with the result of the depth wise convolution to obtain the net result. Depth wise separable convolution drastically reduces the computation and model size.

Training: MobileNet is trained on CIFAR-10 dataset. The dataset consists of 60000 32 x 32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. For data preprocessing, firstly, the input image is normalized by subtracting the mean and dividing by standard deviation. The image is then randomly flipped horizontally from left to right. Then, a random crop is applied to the image to convert it from 32 x 32 to 28 x 28. After the input images are preprocessed, they are sent to the MobileNet body, where firstly a standard convolution layer is applied, followed by 11 layers of depth wise separable convolutions. Inside the depth wise separable convolutions, after applying the convolution, batch normalization, followed by ReLU nonlinearity is applied, as shown in the right chart of Figure 5. In batch normalization, the outputs are scaled by subtracting mean and dividing my standard deviation for fixed sized batches. It allows each layer of a network to learn by itself a little bit more independently of other layers. ReLU (Rectified Linear Unit) provides a non-linearity to the network by making all negative values zero and passing all positive values unchanged. A SoftMax classifier classifies the outputs into the 10 classes of CIFAR-10. Cross Entropy loss is used to calculate the loss over the 60 epochs on which the network is trained. Furthermore, RMSProp is used for updating the weights. A hyperparameter, depth multiplier was used, which factors the number of input channels (M) and number of output channels (N) by a factor α. It helps in faster computation, but reduces the accuracy obtained.

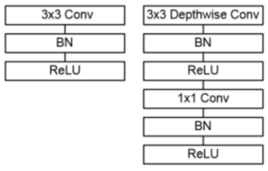
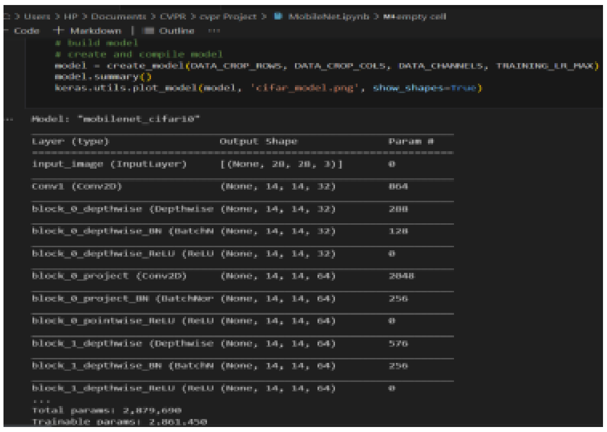


Figure01: Left: Standard Convolution Layer; Right: Depth wise Separable Convolution Layer.

Figure 1 shows the basic architecture of the standard and depth wise separable convolution layers, consisting of the convolution, followed by batch normalization [7] and ReLU[8] nonlinearity.


Users > HP > Documents > CVPR > cvpr Project > MobileNet.ipynb > M+empty cell
Code
Markdown
Outline

```
# build model
# create and compile model
model = create_model(DATA_CROP_ROWS, DATA_CROP_COLS, DATA_CHANNELS, TRAINING_LR_MAX)
model.summary()
keras.utils.plot_model(model, 'cifar_model.png', show_shapes=True)
```

```
Model: "mobilenet_cifar10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_image (InputLayer)     [(None, 28, 28, 3)]       0
_________________________________________________________________
Conv1 (Conv2D)               (None, 14, 14, 32)        864
_________________________________________________________________
block_0_depthwise (Depthwise (None, 14, 14, 32)        288
_________________________________________________________________
block_0_depthwise_BN (BatchN (None, 14, 14, 32)        128
_________________________________________________________________
block_0_depthwise_ReLU (ReLU (None, 14, 14, 32)        0
_________________________________________________________________
block_0_project (Conv2D)     (None, 14, 14, 64)        2048
_________________________________________________________________
block_0_project_BN (BatchNor (None, 14, 14, 64)        256
_________________________________________________________________
block_0_pointwise_ReLU (ReLU (None, 14, 14, 64)        0
_________________________________________________________________
block_1_depthwise (Depthwise (None, 14, 14, 64)        576
_________________________________________________________________
block_1_depthwise_BN (BatchN (None, 14, 14, 64)        256
_________________________________________________________________
block_1_depthwise_ReLU (ReLU (None, 14, 14, 64)        0
...
Total params: 2,879,690
Trainable params: 2,861,450
```

Figure02: Proposed model build.

Type / Stride	Filter Shape	Input Size
Conv / s2	3 x 3 x 3 x 32	28 x 28 x 3
Conv dw / s1	3 x 3 x 32 dw	14 x 14 x 32
Conv / s1	1 x 1 x 32 x 64	14 x 14 x 32
Conv dw / s1	3 x 3 x 64 dw	14 x 14 x 64
Conv / s1	1 x 1 x 64 x 128	14 x 14 x 64
Conv dw / s1	3 x 3 x 128 dw	14 x 14 x 128
Conv / s1	1 x 1 x 128 x 256	14 x 14 x 128
Conv dw / s1	3 x 3 x 256 dw	14 x 14 x 256
Conv / s1	1 x 1 x 256 x 512	14 x 14 x 256
5 x Conv dw / s1	3 x 3 x 512 dw	14 x 14 x 512
5 x Conv / s1	1 x 1 x 512 x 512	14 x 14 x 512
Conv dw / s1	3 x 3 x 512 dw	14 x 14 x 512
Conv / s1	1 x 1 x 512 x 1024	14 x 14 x 512
Conv dw / s2	3 x 3 x 1024 dw	14 x 14 x 1024
Conv / s1	1 x 1 x 1024 x 1024	7 x 7 x 1024
Avg Pool / s1	Pool 7 x 7	7 x 7 x 1024
FC / s1	1024 x 10	1 x 1 x 1024
Softmax / s1	Classifier	1 x 1 x 10

Table 1: MobileNet Architecture

As shown in Table 1, the MobileNet body (encoder) consists of a standard convolution layer, followed by 11 layers of depth wise separable convolutions. For the head (decoder) part, this is followed by a global average pool. A fully connected dense layer is then added to the network. A SoftMax function is applied to this layer to generate results among the 10 classes of CIFAR-10 dataset.

RESULTS

Depth Multiplier	Training Accuracy	Validation Accuracy
1	92.50%	84.52%
0.75	90.82%	80.69%

Table 2: Results of Experiments Performed

Since CIFAR-10 is not that big of a dataset, having depth multiplier to reduce accuracy while not reducing much of time does not seem apt. As a result, depth multiplier=1 is chosen and the curves showing training and validation accuracy and training and validation loss are plotted. The curves are shown in Figure 3

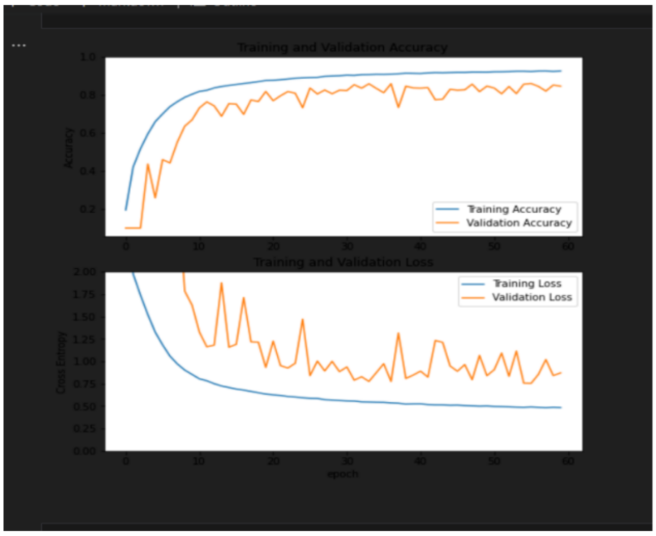
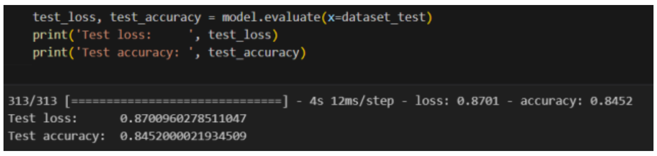


Figure 3. Training & Validation Accuracy & Loss Curves



```
    test_loss, test_accuracy = model.evaluate(x=dataset_test)
    print('Test loss:     ', test_loss)
    print('Test accuracy: ', test_accuracy)

313/313 [==============================] - 4s 12ms/step - loss: 0.8701 - accuracy: 0.8452
Test loss:      0.8700960278511047
Test accuracy:  0.8452000021934509
```

Figure 4. The final validation accuracy thus obtained is 84.52%

Implementation: The input feature map is initially stored in external memory. It is then moved to the internal memory, where the computations are performed. For every layer as shown in Table 1, the filter coefficients are brought to the internal memory from the external memory via the DDR bus, and are moved back to the external memory once the computations are performed.

CONCLUTION

The amount of time it takes to train the neural network versus the size of the input image is a significant trade-off. To lessen this trade-off, the MobileNets-introduced depth wise separable convolutions appear promising. Important design choices were made to ensure that the network doesn't take too long to run and that there are an adequate number of depth wise separable layers to boost validation accuracy. In order to boost the network's speed, the depth multiplier is employed as a hyperparameter. The network's performance time was predicted using calculations of data transfer and compute times. As the field of mobile vision advances, there exist opportunities for further research and improvement. One potential avenue for exploration involves integrating MobileNet with other state-of-the-art techniques, such as attention mechanisms or neural architecture search. By combining MobileNet's efficiency with these advancements, researchers can potentially enhance its performance even further. Continued efforts to optimize and advance convolutional neural networks for mobile applications will contribute to the development of practical and innovative mobile vision systems, ultimately benefiting a wide range of users and applications.

REFERENCES

[1] A. Howard, et. al., “MobileNets: Efficient Convolutional Neural Networks for Mobile VisionApplications,” arXiv:1704.04861, 2017.

[2] A. Krizhevsky, “Convolutional Deep Belief Networks on CIFAR-10,” Unpublished manuscript, 2010.

[3] Y. LeCun, et. al., “Gradient-based Learning Applied to Document Recognition,” Proceedingsof the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[4] A. Krizhevsky, et. al., “ImageNet Classification with Deep ConvolutionalNeural Networks,” In Advances in neural information processing systems, pages 1097– 1105, 2012.

[5] O. Russakovsky, et. al., “ImageNet Large Scale Visual Recognition Challenge” International Journal of Computer Vision, 115(3):211–252, 2012.

[6] L. Bottou, “Large-scale Machine Learning with Stochastic Gradient Descent,” inProc. 19th Int. Conf. Comput. Statist., 2010.

[7] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” arXiv:1502.03167, 2015.

[8] V. Nair and G. Hinton, “Rectified Linear Units Improve Restricted Boltzmann Machines”, ICML, 2010

Authors contribution Group Information:

Group Information:

NAME	ID	CONTRIBUTION
PARTHA MALAKAR	20-42908-1	25%
MD. YOUSUF ALI SHADHIN	20-42783-1	25%
MST. ZANNATUL FERDOUS SUMI	20-43007-1	25%
ARGHO PROSHAD SINGHO	20-42906-1	25%