Implementation of ShuffleNet v1 & v2

Yann Debain debain@kth.se
Suhas Sheshadri suhass@kth.se
Harsha HN harshahn@kth.se

Abstract

In this project we showcase the nitty-gritty of a novel convolutional neural network architecture - ShuffleNet which is known for its computational efficiency. Two research papers were dedicated for versions of ShuffleNet; v1[1] and v2[2] each, both of which have been thoroughly discussed here. Through our experiments, we shall highlight the computational advantage gained by our elegant implementation using Tensorflow. Also, showcase the performance and complexity of our model over MobileNet[3]. ShuffleNet v1 introduces two novel operations, namely pointwise group convolution and channel shuffle. ShuffleNet v2 augments its predecessor's network structure with the introduction of channel split. These ShuffleNet units are employed on the tiny ImageNet for a classification task. Our results demonstrate the performance of this model with a very limited computational budget of around 1-10 MFLOPS. In addition, we also enlist the model performance for various configuration setup. Finally, we present a brief note on the key findings of our experiments made in our scientific endeavour.

1 Introduction

The most common approach towards solving major and complex visual recognition problems have been building deeper, larger and sophisticated architecture inclusive of convolution neural network layers [4,5]. Most accurate state-of-the-art CNN's have deeper layers and vast channels. On the flip side, these models incur massive computational budget and likely not suitable for portable embedded systems such as wearable devices and mobile phones. This computational limitation hinders the pace of further innovation and advancement in various related fields. More computations would consume more power and produce more heat which aggravates the issue. Thus, it is imperative to address this concern.

In this regard towards reducing the computational burden, a novel mechanism called ShuffleNet[1,2] has been proposed. ShuffleNet introduces a modern architecture which serves the functionality of convoluted operations but with significantly limited FLOPS for comparable performance. We have gone ahead further and considered ShuffleNet v2 as an innovative add-on in our demonstration. ShuffleNet units employ point-wise separable group convolution and channel shuffle. Our first task is to successfully implement ShuffleNet and apply this mechanism to solve the Tiny ImageNet classification task. Subsequently, we will examine the FLOPS (floating point operations) incurred, comparison over MobileNet and between versions, and evaluation of model performance for internal configuration setups and tabulate corresponding results.

2 Related Work

Our work is based on the following paper: *ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices* by Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin and Jian Sun [1]. Further supplemented with the paper: *ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design* by Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, Jian Sun[2].

Efficient Model Designs Introduction of deep neural networks and its massive success in computer vision tasks[4, 6, 7] has stimulated its research work. Efficient model designs[8] are vital to address the constraints in embedded space. GoogLeNet[5] and ResNet[9,10] introduced lower complexity models but with impressive performance. SqueezeNet[11] architecture significantly reduces parameters and computations. Since then, NASNet[12] model achieves comparable results to our ShuffleNet. C3AE[13] introduces compact model at 1/9 parameters of our model. Recently, efficient semantic segmentation techniques[14] were proposed using ShuffleNet V2 with atrous separable convolutions.

Group Convolution AlexNet[4] introduced group convolution for distributing computing over two GPUs. Xception[15] showed the effectiveness of depthwise separable convolution which inspired Inception series [16, 17]. MobileNet[3] introduced lightweight models utilizing depthwise convolution method to embark on state-of-the-art results. ShuffleNet generalizes both pointwise group convolution and depthwise separable convolution.

Channel Shuffle and Split Operation Prior to this work, channel shuffle techniques were seldom exploited. IGCNets[18] introduces interleaved group convnets which employs random shuffle approach. However, further investigation on channel shuffle is addressed here. Channel split approach in ShuffleNet v2 is a novel mechanism.

3 Dataset and Pre-processing

The main operations involved with data are Extract, Transform and Load. Firstly, to demonstrate ShuffleNet model, we have chosen Tiny ImageNet dataset comprising of 100,000 colour images encoded in jpeg. The same kind of dataset of size 10,000 images is considered for the validation purpose. These datasets were available on the CANVAS portal. OS file operations were performed to gather access to relevant data. Secondly, the transformation; each image has RGB components coded on 8 bits and of size 64 by 64 pixels. The training dataset consists of 200 classes with each class having a uniform image count of 500 while the validation dataset consists of 200 classes with 50 in each class.

In order to read data efficiently, it is helpful to serialize our data and store it in a set of files that can each be read linearly. TfRecords file format is a simple format from Tensorflow for storing a sequence of binary records. It is recommended and favours data caching for further data-preprocessing. In tfrecords, each image is decoded from jpeg format, pre-processed with data-augmentation (crop, flip, brightness, contrast, hue, and saturation), resized to 56x56 pixels and encoded into features set of image data. Labels are one-hot encoded and stored. Validation and Training datasets are prepared in this fashion but without data-augmentation for the Validation set. For the training purpose, the training dataset is segmented into mini-batches whose size is configurable. Further, they are shuffled and stored. These steps allow faster, better generalization and easy data-loading for the later operations.

4 Methods

Main methods involved are the channel shuffle, ShuffleNet units of both versions: 1 and 2, the overall network architecture of our model and training process.

4.1 Channel Shuffle

Although group convolution significantly reduces the computational burden, it facilitates smaller feature map as outputs from a certain channel are only derived from a small portion of input channels. In other words, it blocks the information flow between channel groups which weakens the flow of representation. Channel shuffle involves the division of input channels into a certain number of groups followed by a stratified shuffle and then fed to successive layer. Channel shuffle enables cross-group information flow and allows a wider feature map between convnets layers.

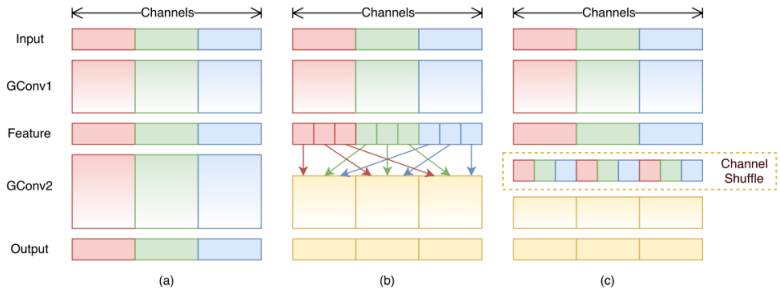

Figure 1: Channel shuffle with two stacked group convolutions (GConv). a) Two stacked Conv layers with two groups, no cross talk; b) input and output channels are fully related when GConv2 takes data from different groups after GConv1; c) Channel shuffle extended to b)

4.2 ShuffleNet Units

A unit consists of three layers: Point-wise group convolution, channel shuffle, Depth-wise separable convolution and again point-wise group convolution in a structured flow as shown in figure 2 below. Implementation of these units was made using tensorflow layer APIs, however, further operations were to be performed to achieve group convolution, channel shuffle and channel split to form shuffle net units.

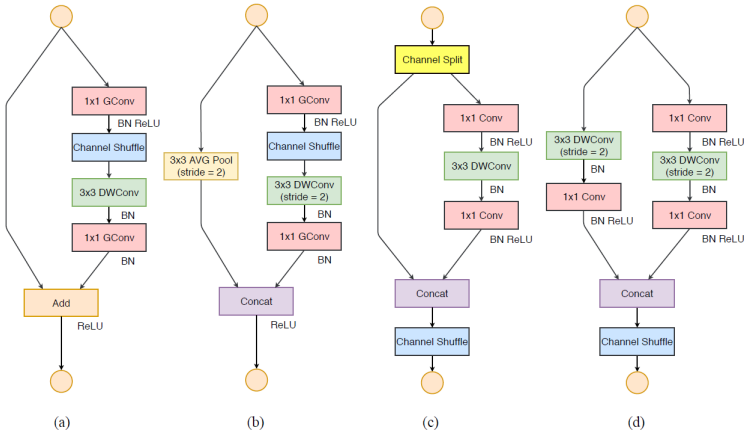

Figure 2: (a) : ShuffleNet v1 unit with point-wise group convolution (GConv) and channel shuffle; (b) : ShuffleNet v1 unit with stride = 2 for spatial sampling; (c) : ShuffleNet v2 with channel split; (d) : ShuffleNet v2 with spatial sampling

4.3 Network Architecture

ShuffleNet units are stacked in 3 stages followed by a global pool layer, a fully connected layer of 200 nodes each representing an output class and finally, softmax function on these which yields class probabilities. Details as shown below in Table 1. With the true labels being one-hot encoded, we can observe cross-entropy error with the predicted output and consequently, the loss is evaluated. Regularization is performed on each kernel by adding an L2 cost over their coefficients and therefore enhance the generalization of the model. Network architecture allows to customize groups size *g* and choose desired complexity by a scalar factor s. With the above table referenced as ShuffleNet 1x, then *S*x enables scaling the number of output channels size and thus, determines the complexity of the model.

Table 1: ShuffleNet v1 architecture on Tiny ImageNet Classification

Layer	Output Size	Ksize	Stride	Repeat	Output Channels (g groups)				
					g = 1	g = 2	g = 3	g = 4	g = 8
Image	56x56				3	3	3	3	3
Conv1	56x56	3x3	1	1	24	24	24	24	24
MaxPool	28x28	3x3	2	1	24	24	24	24	24
Stage 2	14x14		2	1	144	200	240	272	384
	14x14		1	1	144	200	240	272	384
Stage 3	7x7		2	1	288	400	480	544	768
	7x7		1	3	288	400	480	544	768
Stage 4	4x4		2	1	576	800	960	1088	1536
	4x4		1	1	576	800	960	1088	1536
GlobalPool	1x1	4x4							
Fully Connected					200	200	200	200	200

4.4 Training

Prior to training, parameter initialization is vital for the training to be efficient and converge faster and hence following configuration decision has been made. All kernels were Xavier initialized while biases were initialized to zero vectors to ensure the range of the gradient's mean and variance to remain centred and consistent.

During training, the stochastic gradient descent approach has been chosen. We began with Adam optimizer, however, momentum-based Nesterov optimizer has found to be better for our model. Thus, here learning rate follows Nesterov accelerated gradient technique to have faster convergence. In addition, we use a drop learning rate of factor 0.1 at each time the validation accuracy plateaus. Consequently, accuracy is noted after each batch.

To ensure regularization in the model and avoid over-fitting, we add a regularization term in the loss computation for the kernels weights in the model to have the total cost.

5 Experiments

For the purpose of demonstrating the computational benefits of ShuffleNet architecture, we are mainly going to evaluate our models on the Tiny ImageNet classification task. For various model configuration, we will evaluate the model's performance attributes such as complexity and accuracy. Further, we shall provide a comprehensive comparison of ShuffleNet with its successor ShuffleNet v2 and its counterpart: MobileNet - which so far the most efficient (for the comparison the ShuffleNet architecture with g = 3 is a good trade-off between complexity and accuracy as shown in Table 3 and 4, therefore this architecture will be our basic architecture).

In order to quantify the computational benefits gained in ShuffleNet over MobileNet, we implemented MobileNet model and ran the experiments. We have tabulated the performance attributes mainly complexity and accuracy of ShuffleNet v1 model and MobileNet model. Shown in table 2. Notably, ShuffleNet complexity has outperformed the MobileNet by a significant margin (around 2 times) with comparable accuracy. These results confirm the computational merit of ShuffleNet.

Table 2: ShuffleNet vs MobileNet on Tiny ImageNet Classification

Model	Complexity (MFLOPs)	Max Accuracy - Top 1 (%)
ShuffleNet x1 (g = 3)	**10.9**	**52**
MobileNet x1	23.8	50
ShuffleNet x0.5 (g = 3)	**3.1**	**45**
MobileNet x0.5	6.4	43
ShuffleNet x0.25 (g = 3)	**1**	36
MobileNet x0.25	1.8	36

Table 3: Accuracy vs. number of groups g on Tiny ImageNet Classification

Model	Max Accuracy - Top 1 (%)				
	g = 1	g = 2	g = 3	g = 4	g = 8
ShuffleNet x1	50	51	**52**	51	51
ShuffleNet x0.5	**45**	43	**45**	44	44
ShuffleNet x0.25	34	34	36	36	**37**

Following observations can be made from Table 3 shown above. Accuracy of the classification has shown logarithmic growth with respect to the rise in complexity. It's also notable that the accuracy is associated with the shuffle group size.

In order to discern the impact of ShuffleNet channel group size over the complexity, we trained the model with three different sets of output channel sizes of ShuffleNet viz. x0.25, x0.5, and x1. As shown in Table 4 below. We observe that for a chosen output channel size, variations in group size has negligible influence over the complexity; however, xS exponentially heightens the complexity.

Table 4: Complexity vs. number of groups g on Tiny ImageNet Classification

Model	Complexity (MFLOPs)				
	g = 1	g = 2	g = 3	g = 4	g = 8
ShuffleNet x1	11	11	10.9	10.8	11.6
ShuffleNet x0.5	3	3.1	3.1	3.2	3.6
ShuffleNet x0.25	0.9	0.9	1	1	1.2

Finally, we are able to implement the ShuffleNet v2 and the following comparisons were made. Version 2 takes higher computations and delivers incremental accuracy gain. Version 2 seems to share comparable complexity and accuracy with MobileNet than with its predecessor version 1.

Table 5: ShuffleNet vs ShuffleNet v2 on Tiny ImageNet Classification

Model	Complexity (MFLOPs)	Max Accuracy - Top 1 (%)
ShuffleNet x1 (g = 3)	**10.9**	52
ShuffleNet v2 x1	29	**54**
ShuffleNet x0.5 (g = 3)	**3.1**	45
ShuffleNet v2 x0.5	5.8	**46**

6 Conclusion

On the whole, we introduced a novel mechanism - ShuffleNet, an extremely efficient convolutional neural network. Excellent structure involving point-wise group convolution and channel shuffle massively reduced the computational burden. With our experiments, we demonstrated superior characteristics of the ShuffleNet units over MobileNet architecture. In addition, we conducted experiments to evaluate the ShuffleNet model performance for various internal configuration setups. The careful observation indicated better accuracy with increased output channels at each layers requiring higher complexity. ShuffleNet v2 model's accuracy and complexity found to be comparable with MobileNet. All these comprehensive experiments verify the effectiveness of our model. Further scope of advancement in this lane would be to explore the opportunities to enable wider feature mapping and better flow of image feature representation - both of these are potential approaches towards boosting accuracy. We hope this work could inspire future work of network architecture design that is computational friendly.

7 References

1. Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin and Jian Sun. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices, 2017.
2. Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, Jian Sun. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design, 2017.
3. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, 2017.
4. A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.
5. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1–9, 2015.
6. O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3156–3164, 2015
7. S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems, pages 91–99, 2015.
8. K. He and J. Sun. Convolutional neural networks at constrained time cost. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5353–5360, 2015.
9. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016.
10. K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In European Conference on Computer Vision, pages 630–645. Springer, 2016.
11. F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J.Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5 mb model size, 2016
12. B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition.
13. Chao Zhang, Shuaicheng Liu, Xun Xu, Ce Zhu. C3AE: Exploring the Limits of Compact Model for Age Estimation, 2019.
14. Sercan Türkmen, Janne Heikkilä. An efficient solution for semantic segmentation: ShuffleNet V2 with atrous separable convolutions, 2019.
15. F. Chollet. Xception: Deep learning with depthwise separable convolutions. arXiv preprint arXiv:1610.02357, 2016.

16. C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inceptionv4, inception-resnet and the impact of residual connections on learning. arXiv preprint arXiv:1602.07261, 2016.
17. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2818–2826, 2016.
18. Ting Zhang, Guo-Jun Qi, Bin Xiao, Jingdong Wang. Interleaved Group Convolutions for Deep Neural Networks, 2017.

Appendix

A short note on the reflections on the intended outcomes mainly on the pivotal knowledge and skills gained in this deep learning domain, on how they were acquired, and on its demonstration.

- Yann Debain:
  - Familiarity and ability to write network architectures and train them and run experiments with a recognized deep learning software package such as TensorFlow and tools such as GCP and TensorBoard.
  - Familiarity with a popular and efficient neural network (ShuffleNet v1/v2 and MobileNet) involved the complexity and intricacy of its network structure.
  - Familiarity with a popular training algorithm (SGD + Momentum + Nesterov, Adam) and understanding of how it works.
- Harsha HN:
  - Exposure and understanding on how to write network architectures and train them and run experiments with a popular industry standard deep learning software package such as TensorFlow and cloud platform such as GCP.
  - Familiarity with novel and popular CNN architectures such as ShuffleNet v1 and v2 and its components such as Depth-wise Convnet and Group Convnet. Also, about involved complexity and trade-offs.
  - Practical hands-on with training sophisticated DNN involving SGD, momentum, Nesterov and the model hyperparameter and parameter settings.
- Suhas Sheshadri:
  - Analysing a paper and reproducing their network architectures and investigating them with industry standard tools and libraries like GCP and Tensorflow.
  - Understanding the core logic behind ShuffleNet by replicating it. Extending it to ShuffleNet v2 and also comparing with MobileNet.
  - Understand and implement popular training algorithms and the effects of parameter tuning.