Down, down, do your dance, do your dance: CupidShuffle

Matt Krzus
`matt@lastfrontiertechnologies.com`

Abstract

With the recent popularity of Vision Transformers, much of the focus has been on the novel aspect of the field. This paper looks to simply expand on one of the influential edge-focused models, The ShuffleNet. We propose CupidShuffle, a ShuffleNetV2 variant replacing the initial with a Window-based Transformer and also removing 7 ShuffleV2Unit layers while still keeping the round-trip shuffle in tact by switching from [3, 7, 3] to [1, 4, 1]. The proposed architecture keeps ShuffleNetV2's predictive power, while decreasing the total model size. And probably the most important contribution, this paper also attempts to keep it short and concise.

1 Introduction

Vision Transformers have become incredibly popular following advancements in Generative Networks field. Some have incredible utility to solve very pressing problems for those of us without tres commas worth of GPUs. It has become incredibly difficult to reinvent the wheel, and rather than doing so, the attempt of this paper is simply to utilize the novelty of the embedding space of the transformer while combining brutal compute efficiency of ShuffleNetv2.

2 Related Work

ShuffleNet[1] is a computation-efficient architecture designed specially for devices with very limited computing power. The novelty is applying group convolutions and then shuffling the groups so that information flows across channels. This group splitting followed by cross-channel information flow is what makes the network so efficient. In **ShuffleNetV2** [2], the authors present a number of guidelines for computation efficiency: G1) Equal channel width minimizes memory access cost (MAC) G2) Excessive group convolution increases MAC G3) Network fragmentation reduces degree of parallelism and G4) element-wise operations are non-negligible. Based on those 4 principles, the authors conclude that: d 1) use equal channel width convolutions; 2) avoid group convolutions if possible, 3) reduce network fragmentation; and 4) reduce element-wise operations. The authors then improve upon the ShuffleNet model and introduce an operator called *channel-split* for their ShuffleNetV2 architecture. This channel-split works such that at the beginning of each unit, the input feature channels are split into two channel-based branches; where one branch remains as an identity and the other branch consists of three convolutions with the same input and output channels. The major change here is that the two 1x1 convolutions of ShuffleNet are now no longer group-wise. The two branches are then concatenated where the *channel-shuffle* operation is then used to enable communication between the two branches.

Whereas ShuffleNetV2 looked to exploit MAC specific efficiencies, **Shuffle Transformer**[3] looked to focus simply on the *shuffle* operation entirely. The authors replaces the ShuffleUnit with a Window-based Multi-head Self-Attention (WMSA) unit that also does information shuffling for cross information flow. The WMSA structure uses windows that are arranged to evenly partition the

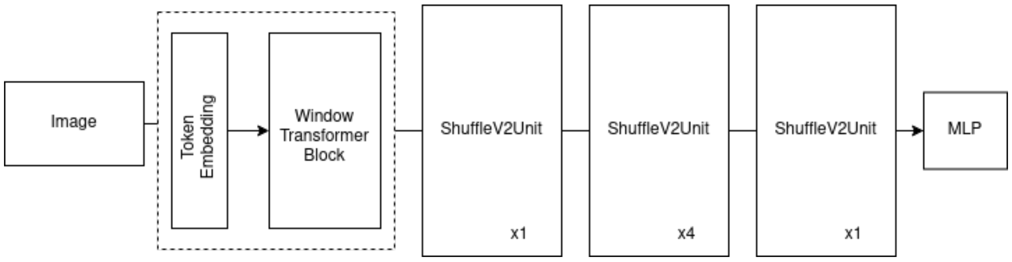

Figure 1: CupidShuffle architecture: token embedding into a single Window-based Multihead Self-Attention (WMSA) where the token-embedding is the same size as the first channel of the first shuffleunit layer to adhere to G1 and 3 total ShuffleV2Unit blocks of a 1-4-1 size

Table 1: Small model family performance on CIFAR 100

network	params	top1 err	top5 err
mobilenet	3.3M	34.02	10.56
mobilenetv2	2.36M	31.92	09.02
squeezenet	**0.78M**	30.59	8.36
shufflenet	1.0M	29.94	8.35
shufflenetv2	1.3M	30.49	8.49
cupidshuffle128	1.3M	29.75	8.46
cupidshuffle64	1.1M	**29.54**	8.61
cupidshuffle28	1.03M	29.55	**8.34**

image in a non-overlapping manner, and while more computation friendly than standard Multi-head Self-Attention (MSA), the receptive field is limited within the window. To address the lack of information flow between fields, the authors propose a spatial shuffle operation which mimics the channel shuffle operation of the original ShuffleNet paper. The result is a much more computation efficient transformer network, but still not adherent to the goals set forth in **ShuffleNetV2** [2].

3 CupidShuffle

In order to abide by the practical guidelines proposed in [2] We replace the initial convolutional block of ShuffleNetV2 with a Window-based Multi-head Self-Attention (WMSA) module from the Shuffle Transformer[2]. Here, in order to adhere to G1, we propose augmenting the tokenization scheme from [2] by changing the inter-channel and embedding dimensions to have the same grouping (HWC), which this conveniently becomes same grouping as the input to the first ShuffleV2Unit. The standard ShuffleNetV2 Unit is then used according the the 1.0 ratio from the paper, but instead of using a 3 blocks in the first layer, 7 blocks in the second layer, and 3 blocks in the third layer, we use a 1-4-1 format that still allows for 6 shuffles total, the minimum number of shuffles required to have information flow between each channel and back to itself again. The intuition here was to take as many good things that very thoughtful people have done and simply make an attempt to limit the size of the network so that it would be applicable for low power scenarios.

4 Formats

Provided is the training code and a tvm, compiled shufflenet variant that will do forward inference. One thing to keep in mind is that on some variants of CUDA, we won't have access to THRUST and thus, some of the efficiencies of TVM won't be of use to us (like when we want to perform NMS for YOLO for example); so we completely remove NMS from the architecture and allow NMS to be executed outside TVM's compiler. Secondly, we want to provide a framework with a focus on CPU/ARM deployment and not GPU deployment. Thirdly, none of this really matters, go spend time with your kids. Thank you for coming to my TED talk. Code available at: https://github.com/KayneWest/cupidshuffle

References

[1] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutionalneural network for mobile devices. In Proceedings of the IEEE conference on computer vision and patternrecognition, pages 6848–6856, 2018. 1, 3, 4

[2] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, Jian Sun. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design

[3] Zilong Huang, Youcheng Ben, Guozhong Luo, Pei Cheng, Gang Yu, Bin Fu. Shuffle Transformer: Rethinking Spatial Shuffle for Vision Transformer

[4] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Meghan Cowan, Haichen Shen, Leyuan Wang, Yuwei Hu, Luis Ceze, Carlos Guestrin, Arvind Krishnamurthy. TVM: An Automated End-to-End Optimizing Compiler for Deep Learning https://arxiv.org/abs/1802.04799