Research Paper Review *15/10/2024*

CSPNET: A New Backbone That Can Enhance Learning Capabilities OF CNNs

*Submitted as part of the **CSCI E-82** course offered by **Harvard Extension School**.*

By: Daud Waqas

Abstract:

Neural networks have enabled state-of-the-art approaches to achieve incredible results on computer vision tasks such as object detection. However, such success greatly relies on costly computation resources, which hinders people with cheap devices from appreciating the advanced technology. In this paper, we propose Cross Stage Partial Network (CSPNet) to mitigate the problem that previous works require heavy inference computations from the network architecture perspective. We attribute the problem to the duplicate gradient information within network optimisation. The proposed networks respect the variability of the gradients by integrating feature maps from the beginning and the end of a network stage, which, in our experiments, reduces computations by 20% with equivalent or even superior accuracy on the ImageNet dataset, and significantly outperforms state-of-the-art approaches in terms of AP50 on the MS COCO object detection dataset. The CSPNet is easy to implement and general enough to cope with architectures based on ResNet, ResNeXt, and DenseNet.

Source code is at https://github.com/WongKinYiu/CrossStagePartialNetworks

Link: https://paperswithcode.com/paper/cspnet-a-new-backbone-that-can-enhance

Context: I essentially chose this paper due to my curiosity on the potential for highly efficient compute for CNN models, especially in production (ie finalised model) environments. Other than the actual accuracy of the model itself, I believe this is the second factor that companies would deeply consider when it comes to any possible implementation of image classifier solutions.

The extension on this paper starts on the next page.

Extension *(from last paragraph onwards)*:

Overall, the implementation of cross stage partial network, or CSPNet, has significantly improved the general performance of convolutional neural networks (CNNs) when benchmarking. These performance gains were primarily observed in optimising memory usage and enhancing inference speed, particularly on devices with limited computational power. In order to maintain its relevancy with existing solutions, the research paper circulated around the implementation of CSPNet with the popular CNN backbones used today *(such as PeleeNet, ResNet, DenseNet, etc)*. After the implementation of CSPNet, the performance gains are actually quite substantial, frequently achieving a 50%+ reduction in memory traffic and a 10-20% increase in inference speed. The way CSPNet works is by "splitting gradient flow" *(ie. separating the feature map into two parts, where one part undergoes dense processing while the other bypasses it)* to avoid redundant gradient duplication. This design helps "maximise gradient difference" *(ie. ensuring that the gradients passing through different stages are distinct, which prevents excessive reuse of similar gradients)*. In simpler terms, CSPNet doesn't "discard" any feature map associated with the gradients of the CNN model, but rather reorganises how feature maps "flow" through the network; by splitting the path of the feature map gradients, CSPNet ensures that each path carries unique features, thus avoiding duplicative learning and mitigating unnecessary processing time *(refer to Figure 2, Figure 3 or Figure 5 for visual representation, since this is a somewhat difficult concept to grasp at first)*. While CSPNet achieves impressive gains on baseline CNNs, its methodology can limit performance on complex tasks. CSPNet optimises gradient flow by truncating and enhancing gradient diversity, which is ideal for general object detection and classification; though this could be troublesome in more complex applications like video tracking, multi-class segmentation, or macro feature identification. Future research could explore "adaptive feature fusion" to dynamically *adjust the extent at which gradient flow and feature map division is applied* based on task complexity, thereby avoiding counterproductive effects in more advanced CNNs.

Potential Improvements *(how I would take this study forward)*:

When it comes to potentially improving on this study, I would like to keep the focus on the main purpose of CSPNet in the first place, which was to increase compute and memory efficiency for CNNs. While this study would prove sufficient for those with large-scale data centres *(who want to "squeeze" more from their production CNN systems)*, it doesn't really cover how CSPNet would be implemented and correctly optimised for low-power hardware, such as mobile and IoT devices. Now, while the study does cover benchmarks with the NVIDIA Jetson TX2, which is a mobile GPU device, it *doesn't really discuss any potential "power-aware" techniques*. Power-aware adaptations could involve model "quantisation" *(reducing bit precision of operations)*, "network pruning" *(removing less essential weights)*, and "dynamic frequency scaling" to reduce power consumption of the model itself. In my opinion this would be a short but meaningful study as it would determine if combining CSPNet with the common "power-aware" techniques used today yields better results than solely implementing CSPNet, or solely implementing "power-aware" techniques. In this study, it may also be worth exploring "on-demand feature activation" which would theoretically allow CSPNet to adjust its computational load based on real-time needs *(thus saving power if the mobile or IoT device in question is processing data at varying levels of demand)*.