Structured Filter Pruning applied to Mask R-CNN: A path to efficient image segmentation

Yannik Frühwirth
Business Information Science
Baden-Wuerttemberg
Cooperative State University
(DHBW) Stuttgart
Rotebühlstr. 133
70197 Stuttgart
yannik.fruehwirth@web.de

ABSTRACT

This study explores the optimization of two-stage object recognition systems, which are integral to numerous applications, by leveraging advanced machine learning techniques. While such systems, including Mask R-CNN, achieve high recognition accuracy, they are often hindered by over-parameterization, excessive computational demands, and significant storagere quirements. To address these challenges, this research introduces a pruning method specifically designed for complex architectures like Mask R-CNN, aimed at reducing computing time, simplifying model complexity, and optimizing storage, all while maintaining detection accuracy.

The proposed method employs a Global Kernel Level Filter Pruning strategy, guided by the *L1-Norm*, to strategically remove non-essential parameters post-training. Experimental results demonstrate that this approach preserves recognition accuracy up to 50% pruning while achieving an 11.6% improvement in computing time on Graphics Processing Units and an 8% improvement on Central Processing Units. Furthermore, the method achieves a compression ratio of 1.47, reducing memory requirements by 33.5%, without compromising Average Precision, which remained at 0.32, equal to the unpruned model at this level.

These findings provide valuable insights into the efficiency optimization of Neural Networks, offering a practical and scalable solution for balancing accuracy, speed, and resource usage in complex architectures. This work contributes to advancing the state-of-the-art in Artificial Intelligence and opens new pathways for integrating complementary techniques such as quantization for further enhancements.

Keywords

Pruning, Filter Pruning, Mask R-CNN, Image segmentation, Two-stage detection

INTRODUCTION

Machine learning, driven by advanced computational power and Graphics Processing Units, has recently gained immense interest (Alpaydin 2016, (Shinde and Shah 2018), particularly in natural language processing, predictive analytics, and image processing (Shinde and Shah 2018). A key focus is object recognition, crucial for applications like autonomous vehicles (D. Feng et al. 2021). This technology's significance is evident in both academic research and public discourse, highlighting its increasing impact on daily life.

However, a significant challenge within this domain is addressing the complexity and computational demands of two-stage object recognition processes (Zou et al. 2019). These processes are characterized by high accuracy but suffer from issues such as over-parametrization (Canziani 2016), extended inference times (Chen et al. 2021), and substantial model storage requirements (Basili 2002). A notable gap in current research is the lack of insights regarding pruning strategies, specifically for complex, two-stage object recognition systems like Mask-RCNN, as evidenced by the scarcity of literature on this topic (Tzelepis et al. 2019, Aguiar Salvi and Barros 2021).

The primary objective of this research is to improve computing time for two-stage object recognition systems through the design and implementation of a tailored pruning approach. While computing time, measured as the change in inference time, is the main focus, the method also aims to address over-parameterization (quantified by the compression ratio) and reduce model memory size (measured as the change in memory size in megabytes). This is achieved by strategically manipulating the model parameters post-training to optimize performance across these metrics.

RELATED WORK

The research contributes primarily to Filter Pruning in the broader context of model compression. Scientific findings on model compression can be clustered into three categories:

Connection pruning

Connection pruning introduces sparsity in deep Neural Networks by eliminating redundant connections (Blalock et al. 2020), a vital aspect of model optimization. Pioneering techniques, such as those in (LeCun 1989) and (Hassibi and Stork 1992), used Taylor expansion for parameter significance assessment. However, these methods often require specialized hardware to leverage the resulting sparsity effectively. Recent advancements in connection pruning have focused on unstructured approaches, like the iterative pruning method in (Han 2016), which removes weights below a certain threshold. Although beneficial in fully connected layers, these methods do not typically lead to significant reductions in computational load in Convolutional Layers (Anwar 2017).

Filter pruning

This method involves evaluating the importance of each filter within the network and pruning accordingly, followed by a crucial retraining phase to recuperate any loss in accuracy (Li et al. 2017).

In determining filter importance, various methodologies are employed. For instance, (Abbasi-Asl and Yu 2017) assesses filter significance by monitoring the impact of its removal on the model's accuracy. Similarly, (Li et al. 2017) utilizes the *L1-Norm* to determine filter importance, while (Hu et al. 2016) bases its evaluation on the activation of output Feature Maps from a subset of training data. These methods largely rely on hand-crafted heuristics.

Advancing beyond these, data-driven approaches for ranking filters have been proposed. One such method, as seen in (Liu et al. 2017), involves Channel Level Pruning, where a learnable scaling factor is attached to each channel. This factor is governed by the *L1-Norm* during training. Group sparsity has also emerged as a promising direction for Filter Pruning, with works like employing group lasso for this purpose (H. Zhou 2016, Wen et al. 2016). However, these techniques sometimes necessitate specialized hardware to optimize SpeedUp during inference (Anwar 2017).

Particularly relevant to the research are the approaches in (Molchanov et al. 2017), (Tzelepis et al. 2019) and (Singh et al. 2019), which, among other things, represents an innovation in filter classification through the use of absolute gradient values. This methods have demonstrated competitive results compared to more traditional brute-force methods, such as checking loss deviation for each filter.

Quantization

Quantization complements the pruning methods by reducing the memory and computational demands of a network. It involves converting network weights into a lower bit configuration (Cosman et al. 1993). Techniques like binarization (Rastegari et al. 2016) and ternary quantization (H. Zhou 2016) have been pivotal in this area. However, these methods sometimes necessitate special hardware for optimal implementation (Gholami et al. 2015). The combination of pruning and quantization, as seen in (Huang et al. 2017), highlights the potential for achieving significant model compression while maintaining performance.

RESEARCH METHODOLOGY

The methodology follows the Design Science Research approach (Peffers et al. 2007), which is known for its iterative nature, to experimentally evaluate the proposed compression method on various modern Neural Network architectures. This process is focused on the goal of optimizing compression, memory requirements and inference time for networks of different depths and widths in different domains. The project was divided into two distinct iterations, each of which produced a prototype that embodied the dynamic and adaptive progression that characterizes Design Science Research. The iterative process facilitates the refinement of the compression technique and allows to respond effectively to the insights gained at each stage.

Dataset and Algorithm

The widely used Microsoft COCO dataset (Lin et al. 2014) was selected for image classification and segmentation. The full scope of the subdataset train (80k images) (Lin et al. 2014) and subdataset validate (35k images) (Lin et al. 2014) are applied to the Mask R-CNN algorithm. Mask R-CNN is based on a Resnet50 backbone and is already pre-trained (Facebook 2020). The pruning is applied post-training. Recognition accuracy and computing time are measured using the minival sub-dataset (5k images) (Lin et al. 2014).

All implementations of the compression method are executed in PyTorch and CUDA, ensuring high performance and compatibility with modern graphics hardware. The inference time is tested on both graphics and central processing hardware to evaluate performance across diverse systems. Specifically, testing on the graphics hardware was conducted using the Nvidia H100 NVL, with a maximum memory capacity of 93 GB, while central processing hardware testing was performed on an Apple M1 chip with 16 GB of unified memory. Furthermore, to foster reproducibility and community engagement, the implementations are publicly available via GitLab.

Evaluation

The evaluation primarily focuses on computing time, with inference time serving as the key metric. Inference time is measured under consistent and controlled conditions, timing the model's forward pass during inference on both graphics and central processing hardware.

Additionally, the compression ratio is calculated, which is a direct measure of the reduction in network size:

$$\text{compression ratio} = \frac{\text{base model size}}{\text{pruned model size}}$$

Finally, the model's detection performance is evaluated using Average Precision for Intersection over Union thresholds ≥ 0.5. For the remainder of this paper, Average Precision refers to Average Precision computed at Intersection over Union 0.5. This metric is used to validate that the pruning method preserves detection accuracy.

These metrics are critical in gauging the trade-off between model efficiency and performance, ensuring that the compression method achieves the optimal balance for practical deployment.

EXPERIMENTS

Filter Pruning at Channel Level

Approach

The initial approach involved systematically deactivating filters by nullifying channels. This was based on the calculation of the *L0-Norm* for each channel, which counts non-zero values and returns the result (M. Feng et al. 2013). Channels were then sorted ascendingly based on their *L0-Norm*, and a predetermined percentage of the least important channels were pruned.

Referring to Table 1, the results demonstrate that the pruning approach becomes ineffective for pruning proportions exceeding 10%, as the Average Precision (AP) drops to 0. While pruning leads to a reduction in inference time for both the Graphics Processing Unit (GPU) and the Central Processing Unit (CPU), the loss in detection accuracy outweighs the computational benefits. This sharp decline underscores the limitations of the current pruning strategy, suggesting it is unsuitable for practical deployment.

The failure of the method is likely due to the collapse of output feature maps, where essential information required for accurate detection is eliminated during the pruning process.

Table 1: Filter Pruning at Channel Level

Base Model	**Pruning Proportion [%]**	**Inference Time on GPU [ms]**	**Inference Time on CPU [ms]**	**AP**
Mask R-CNN	*0*	22.80	2128.03	0.31
Mask R-CNN	*10*	27.36	2236.58	0.18
Mask R-CNN	*20*	24.47	2165.98	0.03
Mask R-CNN	*30*	16.30	1963.02	0.00
Mask R-CNN	*40*	13.28	1837.71	0.00
Mask R-CNN	*50*	13.28	1801.18	0.00
Mask R-CNN	*60*	13.33	1836.08	0.00
Mask R-CNN	*70*	13.21	1746.75	0.00
Mask R-CNN	*80*	13.06	1793.32	0.00

The findings highlight the inadequacy of this pruning approach for complex architectures, as the severe reduction in Average Precision negates the benefits of reduced model size and computational efficiency. These results emphasize the need for alternative pruning strategies that can achieve model compression while preserving detection accuracy.

Filter Pruning on Global Kernel Level

Deficit analysis

The rapid decline in recognition accuracy was attributed to the collapse of subsequent layers caused by structured Filter Pruning at the Channel Level. It was clear that the method should not completely nullify the Feature Maps of filter outputs, essential for subsequent calculations.

Approach

The second iteration involved atomic-level manipulation. Similiar to the researches by (Li et al. 2017) and (Kumar et al. 2021), every parameter within the convolution kernels was considered for pruning, with the least important parameters identified using the *L1-Norm* method, which calculates the sum of vector sizes (Kumar et al. 2021). This allowed for a more nuanced pruning approach where fewer values might be pruned in some kernels compared to others.

Results

The pruning approach demonstrated improved computational efficiency, particularly in terms of inference time, with some trade-offs in detection accuracy as reflected in Average Precision (AP). Table 2 provides a comprehensive overview of these results.

The inference time on the Graphics Processing Unit showed a consistent reduction as pruning proportions increased. At 50% pruning, the inference time decreased from 22.8 milliseconds for the base model to 20.15 milliseconds, representing an 11.6% improvement in computational efficiency. The reduction continued with further pruning, reaching 17.31 milliseconds at 80% pruning, marking a total 24.1% improvement compared to the base model. Similarly, the inference time on the Central Processing Unit exhibited minor improvements. At 50% pruning, the inference time decreased from 2128.03 milliseconds for the base model to 1958.89 milliseconds, yielding an 8.0% improvement. Further pruning resulted in a consistent reduction, with the inference time reaching 1830.48 milliseconds at 80% pruning, amounting to a total 14.0% improvement compared to the base model.

For the Central Processing Unit, a similar trend was observed, though the reductions were less pronounced. At 50% pruning, the inference time decreased from 2731 milliseconds to 2428 milliseconds, corresponding to an 11.1% improvement. At 80% pruning, the inference time further decreased to 2386 milliseconds, resulting in a total 12.6% improvement compared to the base model.

The detection accuracy, as measured by Average Precision (AP), remained stable up to 50% pruning, maintaining values between 0.31 and 0.34. Beyond this point, the Average Precision began to decline significantly, dropping to 0.22 at 60% pruning and experiencing a sharp fall to 0.05 at 70% pruning. At 80% pruning, the Average Precision reached 0.00, indicating a complete loss of detection capability. This decline suggests that high pruning rates lead to a collapse of individual filter kernels, disrupting subsequent computations and feature map generation.

Table 2: Filter Pruning on Global Kernel Level

Base Model	**Pruning Proportion [%]**	**Inference Time on GPU [ms]**	**Inference Time on CPU [ms]**	**AP**
Mask R-CNN	*0*	22.80	2128.03	0.31
Mask R-CNN	*10*	22.74	2263.18	0.31
Mask R-CNN	*20*	23.67	2263.18	0.31
Mask R-CNN	*30*	24.10	1982.79	0.34
Mask R-CNN	*40*	21.73	2103.54	0.33
Mask R-CNN	*50*	20.15	1958.89	0.32
Mask R-CNN	*60*	19.97	1873.57	0.22
Mask R-CNN	*70*	18.45	1853.44	0.05
Mask R-CNN	*80*	17.31	1830.48	0.00

Filter pruning at the channel level proved to be ineffective. The collapse of filters resulted in unusable detections, rendering a comparison of computing time irrelevant. In contrast, the proposed approach, filter pruning on the global kernel level, demonstrated its effectiveness by maintaining consistent detection accuracy at a pruning rate of 50% of all parameters in the convolutional layers, along with a 11.6% improvement in computing time.

Furthermore, the base model requires 179 megabytes of memory, while the pruned model reduces this requirement to 119 megabytes. This corresponds to a 33.5% reduction in memory usage, making the pruned model significantly more efficient in terms of storage while preserving detection performance.

CONCLUSION

Objective

The study embarked on addressing a critical issue in the realm of Neural Networks, particularly focusing on the complexity and efficiency of two-stage object recognition methods like Mask R-CNN. The challenge lays in reducing computing time, over-parametrization and decreasing model storage size without compromising the high recognition accuracy inherent to these methods. The existing corpus of literature demonstrates a conspicuous paucity of insights into the application of pruning techniques within the ambit of intricate, two-stage object recognition frameworks. This research endeavor specifically targets this lacuna, with a focus on elucidating the implications of such techniques when applied to the Mask R-CNN architecture. The objective is to enrich the academic discourse by providing a comprehensive analysis of pruning strategies in complex Neural Networks, thereby bridging the identified knowledge gap.

Results

The research introduced a novel concept of Filter Pruning at the Global Kernel Level. This approach strategically identifies and eliminates the least significant parameters within the convolutional kernels of Mask R-CNN using the *L1-Norm*. This method represents a significant advancement in network optimization, effectively reducing the network's complexity and computational time while preserving the crucial accuracy required for object recognition tasks. The findings highlight the potential of precise, kernel-focused pruning as a powerful strategy to enhance the efficiency of complex Convolutional Neural Network architectures.

Key results of this study include maintaining high recognition accuracy up to 50% pruning, achieving an 11.6% improvement in computational time on the Graphics Processing Unit and an 8% improvement on the Central Processing Unit, while delivering a total compression ratio of 1.47. At this pruning level, the Average Precision remained at 0.32, equivalent to that of the unpruned model, demonstrating the effectiveness of this approach in preserving detection performance.

Implications

This study provides key insights for optimizing Neural Networks, introducing a post-training compression technique for Mask R-CNN that enhances algorithm refinement and efficiency. The findings reveal that pruning Feature Map output channels offers limited benefits, whereas fine-tuning filter kernels at a granular level is more effective and adaptable for similar two-stage recognition methods. This approach not only reduces computing time, model size and complexity but also maintains high recognition accuracy, ensuring its practicality for real-world applications.

Furthermore, the proposed Filter Pruning strategy significantly enhances the model's suitability for complementary compression techniques, particularly quantization. By eliminating redundant parameters and structuring the model more efficiently, this method opens the door to substantial improvements in computing time when combined with quantization. Together, these methods create a synergistic pathway for optimizing resource usage while maintaining detection performance, making this approach highly relevant for deployment in resource-constrained environments, such as edge devices and real-time systems.

FUTURE WORK

The field of object recognition, especially with complex architectures like Mask R-CNN, is on a trajectory of continuous evolution and expansion. This growth trajectory underscores the pressing need for effective compression methods that can adeptly manage the intricacy and expansiveness of these systems. As the utilization of object recognition methodologies escalates, it's anticipated that the significance of algorithms such as Mask R-CNN will correspondingly rise, marking a pivotal juncture in the field's advancement.

Future research in this domain is poised to traverse several critical paths. Firstly, there is a compelling need to pioneer new compression approaches. These novel strategies should ideally harness the latest developments in machine learning and Artificial Intelligence, specifically tailored to address the unique challenges posed by intricate Neural Network architectures. The creation of these innovative methods is paramount to keeping pace with the escalating complexity and capabilities of these systems.

Furthermore, there is a significant opportunity to refine and optimize existing compression procedures. This optimization could focus on multiple fronts, including enhancing efficiency, minimizing computational demands, and striking a more effective balance between model size, processing speed, and accuracy. Fine-tuning these elements is crucial to ensure that compression techniques maintain their relevance and efficacy, especially as technology rapidly advances and network architectures grow increasingly complex.

Another vital area of focus is the application of compression techniques to a diverse array of algorithms within object detection. Moving beyond Mask R-CNN, this research would extend the reach and applicability of these compression methods, making them useful across a broader spectrum of object detection applications. By encompassing a variety of algorithms, this research endeavor can significantly contribute to the overall functionality and utility of object detectors.

Finally, this work demonstrates that pruning not only reduces over-parameterization but also prepares the model for quantization, a complementary compression technique with significant potential for further reducing memory and computational requirements. By removing redundant parameters through pruning, the model structure becomes better suited for lower-precision quantization, enabling even more substantial gains in efficiency. Future studies should explore the combined impact of pruning and quantization, particularly in resource-constrained environments such as edge devices, where lightweight models are paramount.

REFERENCES

Abbasi-Asl, R. and B. Yu 2017. "Structural Compression of Convolutional Neural Networks Based on Greedy Filter Pruning". In: *arXiv preprint arXiv:1705.07356.*

Aguiar Salvi, A. de and R. C. Barros 2021. "Model Compression in Object Detection". In: *2021 International Joint Conference on Neural Networks (IJCNN).* IEEE, pp. 1–8.

Alpaydin, E. 2016. *Machine learning, The new AI.* Cambridge, MA: MIT Press.

Anwar, S. et al. 2017. "Structured Pruning of Deep Convolutional Neural Networks". In: *ACM Journal on Emerging Technologies in Computing Systems* 13.3, pp. 1–18.

Basili, V. R. 2002. "The role of experimentation in software engineering: past, current, and future". In: *Proceedings of IEEE 18th International Conference on Software Engineering.* 1, pp. 1–8.

Blalock, D. et al. 2020. "What is the State of Neural Network Pruning?" In: *Proceedings of Machine Learning and Systems 2020* 1.1, pp. 1–18.

Canziani, A. et al. 2016. *An Analysis of Deep Neural Network Models for Practical Applications.* arXiv.

Chen, L. et al. 2021. "Knowledge from the original network: restore a better pruned network with knowledge distillation". In: *Complex Intelligent Systems* 8.1, pp. 1–10.

Cosman, P. C. et al. 1993. "Using vector quantization for image processing". In: *Proceedings of the IEEE* 81.9, pp. 1326–1341.

Facebook 2020. *From Research to Production.* `https://pytorch.org/`. Retrieved: 05.05.2020.

Feng, Di et al. 2021. "A Review and Comparative Study on Probabilistic Object Detection in Autonomous Driving". In: *IEEE Transactions on Intelligent Transportation Systems* 1, pp. 1–20.

Feng, M. et al. 2013. *Complementarity formulations of l0-norm optimization problems.* Industrial Engineering and Management Sciences. Technical Report.

Gholami, A. et al. 2015. "A Survey of Quantization Methods for Efficient Neural Network Inference". In: *Proceedings of the IEEE International Conference on Computer Vision.* Santiago, pp. 1440–1448.

H. Zhou, et. al 2016. "Less Is More: Towards Compact CNNs". In: *ECCV.* Springer, pp. 662–677.

Han, S. et. al 2016. "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding". In: *International Conference on Learning Representations.* San Juan, pp. 1–14.

Hassibi, B. and D. Stork 1992. "Second Order Derivatives for Network Pruning: Optimal Brain Surgeon". In: *Neural Information Processing Systems 1992*, pp. 164–171.

Hu, H. et al. 2016. "Network Trimming: A Data-Driven Neuron Pruning Approach Towards Efficient Deep Architectures". In: *arXiv preprint arXiv:1607.03250.*

Huang, J. et al. 2017. "Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors". In: *IEEE Conference on Computer Vision and Pattern Recognition 2017*, pp. 1–21.

Kumar, A. et al. 2021. "Pruning filters with L1-norm and capped L1-norm for CNN compression". In: *Appl Intell* 51, pp. 1152–1160. DOI: `10.1007/s10489-020-01894-y`.

LeCun, Y. et al. 1989. "Optimal Brain Damage". In: *Advances in Neural Information Processing Systems.* Ed. by D. Touretzky. Morgan-Kaufmann, pp. 1–8.

Li, H. et al. 2017. "Pruning Filters for Efficient ConvNets". In: *International Conference on Learning Representations.* Toulon, pp. 1–13.

Lin, T.-Y. et al. 2014. "Microsoft COCO: Common Objects in Context". In: *European Conference on Computer Vision.* Springer, pp. 740–755.

Liu, Z. et al. 2017. "Learning Efficient Convolutional Networks Through Network Slimming". In: *ICCV.* IEEE, pp. 2755–2763.

Molchanov, P. et al. 2017. "Pruning Convolutional Neural Networks for Resource Efficient Inference". In: *ICLR.*

Peffers, K. et al. 2007. "A Design Science Research Methodology for Information Systems Research". In: *Journal of Management Information Systems* 24.3, pp. 45–77.

Rastegari, M. et al. 2016. "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks". In: *ECCV.* Springer, pp. 525–542.

Shinde, P. P. and S. Shah 2018. "A Review of Machine Learning and Deep Learning Applications". In: *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA).* IEEE, pp. 1–6.

Singh, P. et al. 2019. "Stability Based Filter Pruning for Accelerating Deep CNNs". In: *Winter Conference on Applications of Computer Vision 2019.* IEEE. Waikoloa Village, pp. 1–9.

Tzelepis, G. et al. 2019. *Deep Neural Network Compression for Image Classification and Object Detection.* n.d.: n.p..

Wen, W. et al. 2016. "Learning Structured Sparsity in Deep Neural Networks". In: *NIPS*, pp. 2074–2082.

Zou, Z. et al. 2019. *Object Detection in 20 Years: A Survey.* n.d.: n.p.

Contact

Mail: yannik.fruehwirth@web.de
Code: `https://gitlab.com/yannik_2f/structuredFilterPruningForMaskRCNN`