ISSN: 2424-8460(O)
2251-2608(P)

Original Research Article

Research on the optimization of computer vision algorithms for large - scale models

Haolan Li

Western Institute of Computing Technology, Chongqing, 401120, China

Abstract: This paper delves into the optimization of computer vision algorithms for large - scale models. With the rapid development of deep learning in computer vision, large - scale models have shown remarkable performance in various tasks. However, their high computational demands and memory requirements pose significant challenges. This research explores techniques such as model compression, acceleration strategies, and efficient data handling to optimize these algorithms. Through the analysis, it demonstrates the effectiveness of these optimization methods in improving the efficiency and performance of large - scale computer vision models, making them more applicable in real - world scenarios.

Keywords: Computer vision; Large-scale models; Algorithm optimization; Model compression; Acceleration

1. Introduction

Computer vision has witnessed a revolutionary transformation with the advent of deep learning. Large - scale models, such as convolutional neural networks (CNNs) and their variants, have achieved state - of - the - art performance in tasks like image classification, object detection, and semantic segmentation. These models, with their complex architectures and numerous parameters, can capture intricate patterns in visual data. However, the increasing size and complexity of these models come at a cost. They require substantial computational resources, both in terms of processing power and memory, which limits their deployment in resource - constrained environments, such as mobile devices and edge computing platforms.

2. Challenges in large - scale computer vision models

2.1. High computational complexity

Large - scale computer vision models typically consist of multiple convolutional layers, fully - connected layers, and other complex operations. The number of floating - point operations (FLOPs) in a modern CNN for image classification can be in the billions or even trillions. For example, a standard ResNet - 152 model has a large number of convolutional filters that need to be convolved with the input images at each layer. This high computational complexity leads to long processing times, especially when dealing with high - resolution images or large datasets[1].

2.2. Memory requirements

Storing the model parameters, intermediate feature maps, and activation values requires a significant amount of memory. In large - scale models, the number of parameters can range from millions to billions. For instance, the GPT - 3 language model, which has some similarities in terms of parameter scale to large - scale computer vision models, has 175 billion parameters. In computer vision, a large - scale object detection model

might have hundreds of millions of parameters, and the intermediate feature maps generated during inference can also consume a vast amount of memory. This becomes a bottleneck when deploying these models on devices with limited memory, such as smartphones or embedded systems.

2.3. Data - intensive nature

Computer vision models rely on large amounts of labeled data for training. Collecting, preprocessing, and storing this data is a resource - intensive task. Moreover, training on large datasets requires high - speed data access and transfer, which can be a challenge in some environments. Additionally, the data distribution in real - world scenarios can be complex and often non - stationary, making it difficult for models to generalize well[2].

3. Optimization techniques for computer vision algorithms in large - scale models

3.1. Model compression

3.1.1. Pruning

Pruning is a technique that involves removing unimportant connections or neurons in the model. In a neural network, many connections may have very small weights, indicating that they contribute little to the overall model performance. By pruning these connections, the model size can be reduced without significantly degrading the accuracy. For example, in a CNN, we can prune the connections between convolutional filters and neurons in the subsequent layers. There are different pruning methods, such as magnitude - based pruning, where connections with weights below a certain threshold are removed, and second - order derivative - based pruning, which takes into account the sensitivity of the model to the removal of a particular connection.

3.1.2. Quantization

Quantization reduces the precision of the model's weights and activations. In traditional neural network implementations, weights and activations are often represented using 32 - bit floating - point numbers. However, research has shown that a lower - precision representation, such as 8 - bit integers or 16 - bit floating - point numbers, can still maintain a high level of accuracy. For instance, in integer quantization, the weights are mapped to a discrete set of integer values, which reduces the memory footprint of the model. This is particularly useful for deployment on hardware that has better support for integer operations, such as some mobile GPUs and neural network processing units (NPUs)[3].

3.1.3. Knowledge distillation

Knowledge distillation involves training a smaller "student" model to mimic the behavior of a larger "teacher" model. The teacher model, which is usually a pre - trained large - scale model, has learned a rich set of features. The student model is trained to match the soft labels (the output probabilities of the teacher model) on the training data, rather than just the hard labels (the true class labels). This way, the student model can capture some of the knowledge of the teacher model while being much smaller in size.

3.2. Acceleration strategies

3.2.1. Hardware - accelerated computing

Leveraging specialized hardware is an effective way to accelerate computer vision models. Graphics processing units (GPUs) have been widely used in deep learning due to their high parallel processing capabilities. GPUs can perform multiple operations in parallel, such as matrix multiplications, which are fundamental to

neural network computations. For example, in a CNN, the convolutional layers involve a large number of matrix multiplications between the convolutional filters and the input feature maps. GPUs can significantly speed up these operations. In addition, neural network processing units (NPUs) have been developed specifically for accelerating neural network computations. NPUs are optimized for the types of operations commonly found in neural networks[4].

3.2.2. Algorithm - level acceleration

Optimizing the algorithms themselves can also lead to significant speed - ups. For example, using fast convolutional algorithms, such as the Winograd algorithm, can reduce the number of FLOPs required for convolutional operations. The Winograd algorithm re - arranges the convolutional computation in a way that reduces the number of multiplications and additions. Another approach is to use sparse matrix operations. Since pruning often results in sparse matrices (matrices with a large number of zero elements), using algorithms that can efficiently handle sparse matrices can accelerate the computations. For instance, sparse matrix - vector multiplications can be performed much faster than their dense counterparts, which can speed up the forward and backward passes in a neural network.

3.3. Efficient data handling

3.3.1. Data augmentation

Data augmentation is a technique that creates new training data from the existing dataset by applying various transformations, such as rotation, flipping, zooming, and adding noise. This not only increases the size of the effective training dataset but also helps the model to generalize better. For example, in object detection, rotating the training images can expose the model to different orientations of the objects, making it more robust to real - world variations.

3.3.2. Active learning

Active learning is a sampling strategy where the model actively selects the most informative data points for labeling. Instead of randomly sampling data for training, active learning algorithms identify data points that are most likely to improve the model's performance. For example, in a semi - supervised setting, the model can select unlabeled data points that are close to the decision boundary (the boundary between different classes in the feature space). By labeling and adding these data points to the training set, the model can learn more effectively with fewer labeled data, reducing the cost of data collection and labeling.

4. Future directions

4.1. Hybrid optimization approaches

Future research will likely focus on combining different optimization techniques in a more intelligent way. For example, integrating model compression with hardware - specific acceleration techniques. A model that has been pruned and quantized can be further optimized for a particular NPU architecture by adjusting the computational graph to take advantage of the NPU's hardware - specific features. This hybrid approach can lead to even greater improvements in performance and efficiency.

4.2. Online optimization

In real - world applications, the data distribution may change over time. Online optimization techniques,

which can adapt the model in real - time as new data arrives, will become increasingly important. For example, in a surveillance system, the types of objects being monitored may change seasonally or due to changes in the environment. Online optimization algorithms can update the model without having to retrain it from scratch, ensuring that the model remains accurate and efficient.

4.3. Explainable AI in optimization

As large - scale computer vision models become more complex, understanding how the optimization techniques affect the model's behavior and performance is crucial. Explainable AI techniques can be applied to optimization to provide insights into why a particular optimization method works or fails. For example, in model compression, understanding which connections or neurons are most important for the model's performance can help in designing more effective pruning strategies.

5. Conclusion

Optimizing computer vision algorithms for large - scale models is essential for overcoming the challenges of high computational complexity, memory requirements, and data - intensive nature. Through techniques such as model compression, acceleration strategies, and efficient data handling, significant improvements in efficiency and performance can be achieved. The case studies presented in this paper demonstrate the practical applicability of these optimization methods in real - world scenarios. As the field of computer vision continues to evolve, further research in hybrid optimization approaches, online optimization, and explainable AI in optimization will open up new possibilities for making large - scale computer vision models more powerful and accessible.

About the author

Lihaolan (1987.06-)
Gender: male,
Nationality: Han nationality,
Native place: Liaoning,
Education: graduate student,
Title: Senior Engineer,
Research direction: artificial intelligence, model optimization

References

[1] Yang L ,Driscol J ,Gong M , et al. TGGLinesPlus: A Robust Topological Graph-Guided Computer Vision Algorithm for Line Detection From Images [J]. Transactions in GIS, 2025, 29 (1): e70015-e70015.

[2] Tian Y ,Zhu F . Application of computer vision algorithm in ceramic surface texture analysis and prediction [J]. Intelligent Systems with Applications, 2025, 25 200482-200482.

[3] Liu W ,Chen J ,Lyu Z , et al. Automatic tile position and orientation detection combining deep-learning and rule-based computer vision algorithms [J]. Automation in Construction, 2025, 171 106001-106001.

[4] Zhang H . An innovative approach to lighting design: implementing computer vision algorithms for dynamic light environments [J]. International Journal of System Assurance Engineering and Management, 2025, (prepublish): 1-13.