A Survey on Deep Neural Network Compression : Challenges, Overview and Solutions

Reading By, Harsh Srivastava
B.Tech in Computer Science
IIT Patna
Patna, India
harshsrivasta243@gmail.com

Guided By, Saurabh Sharma
Ph.D, Computer Science
IIT Patna
Patna, India

Supervised By, Dr. Joydeep Chandra
Professor, Dept.of Computer Science
IIT Patna
Patna, India

***Abstract*—This document is a (summarized) reading of the paper titled ”A Survey on Deep Neural Network Compression : Challenges, Overview and Solutions” which is originally authored by Rahul Mishra, Hari Prabhat Gupta and Tanima Dutta.**

The Venue of Publication :
Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, India.

The link to the original paper is here :

https://github.com/harshsrivasta243/1901CS24_CS299_2021/blob/Papers/Paper2.pdf

I. Motivations and Introduction

Deep Neural Network (DNN) has gained unprecedented performance due to its automated feature extraction capability. However, the colossal requirement of computation, energy, and storage of DNN models make their deployment prohibitive on resource constraint IoT devices. Therefore, several compression techniques were proposed in recent years for reducing the storage and computation requirements of the DNN model. These techniques on DNN have utilized a different perspective for compressing DNN with minimal accuracy compromise.

DNN model involves the following components, including Convolutional Neural Network (CNN), Fully Connected (FC) layers, and RNN (Recurrent Neural network). A DNN model can have all these components at a time or any of its combination. The CNN extract spatial features and RNN identifies temporal features form the dataset. These features archives a high order performance in classification, regression and prediction of different tasks.

II. Problem(s) Addressed

Large deep models tend to achieve good performance in practice, because the over parametrization improves the generalization performance when new data is considered. Despite these advantages, the DNN models require a significant amount of resources. The huge resource requirements of the DNN model are a bottleneck for real-time inference and to run DNN model on browser-based applications. This reading addresses this issue and proposes different DNN compression techniques to mitigate these shortcomings of DNN model.

III. Baseline

A. Benefits of DNN Model Compression

We can achieve several benefits by compressing the DNN model over the traditional cumbersome DNN model. Some of these benefits are as follows:

- Storage capacity : By compressing the DNN model, we can preserve storage that facilitates the deployment of DNN model on Resource Constraint Devices (RCDs).
- Computation requirements : It would be beneficial to employ DNN compression for reducing the computation requirement.
- Earliness : The DNN compression mechanisms provide a higher degree of earliness in both training and inference phases.
- Privacy : It would be beneficial to employ in-situ processing using compressed DNN model on RCDs as it could help in preserving privacy and provides data security.
- Energy consumption : It enhances its suitability for the deployment of compressed DNN model on battery-operated IoT devices.

B. Categorization of DNN Compression Techniques

This section classifies the existing work on DNN compression in five broad categories, i.e., network pruning, sparse representation, bits precision, knowledge distillation, and miscellaneous techniques. All these methods are discussed in detail in the next section.

IV. Techniques of DNN Compression

A. Network Pruning

Deep network pruning is one of the popular techniques to reduce the size of a deep learning model by incorporating the removal of the inadequate components such as channels, filters, neurons, or layers to produce a lightweight model. The pruning is performed on a pre-trained model iteratively, such that only inadequate components are pruned from the model. We further categorize the network pruning techniques into the following categories :

1) Channel Pruning: Channel pruning is a concept of reducing the number of channels in the input supplied to the intermediate layers of the DNN model. The data supplied to the DNN model are initially channelized to form an appropriate input. An inference time approach was proposed to perform channel pruning by effectively removing the redundant channels from CNN. It is claimed to accelerate the deep CNN using two sequential steps, including, regression-based channel selection followed by estimation of least reconstruction error.

A combinatorial optimization problem was formed for network pruning which is solved using an evolutionary algorithm and attention mechanism. It was named as the Evolutionary NetArchitecture Search (EvoNAS).
Later on, an approach named VarGNet was proposed for reducing the operations in the CNN model that uses variable group convolution. The use of variable group convolution helps in solving the conflict between small computational cost and the unbalanced computational intensity. Further, angular distillation loss is adopted to improve the performance of the generated lightweight model.
Another DNN compression technique was proposed based on the streamlined architecture that uses depth-wise separable convolutions to build a lightweight model. The technique was named as MobileNets that estimates two hyperparameters, i.e., width multiplier and resolution multiplier. It allows the model developer to choose a small network that matches the resources restrictions (latency, size) for different applications.

2) Filter Pruning: The convolutional operation in the CNN model incorporates a large number of filters to improve its performance under different processes of classifications, regressions, and predictions. The elimination of the unimportant filters plays a decisive role in reducing the computational requirements of the DNN model.

An efficient framework, namely ThiNet, was hence proposed for accelerating the operation of the CNN model using compression in both training and testing phases. ThiNet uses a greedy algorithm for solving the optimization problem.
A mechanism named DeepMon was then proposed to provide deep learning inference on mobile devices. It was claimed to run the inference in limited time and provides energy efficiency using the Graphical Processing Unit (GPU) on the mobile device. An optimization mechanism was also proposed for processing convolutional operation on mobile GPU. The mechanism utilizes the internal processing structure of CNN incorporating filters and the number of connections to reuse the results.

3) Connection Pruning: The number of input and output connections to a layer of DNN model determines the number of parameters. Connection pruning reduces the parameters by eliminating unimportant connection for different layers in the DNN model. Later on, a connection pruning technique was proposed that first prunes the DNN model by learning only important connection using dropout. Next, the learned weights are quantized using clustering technique, and further Huffman coding is applied to reduce the storage requirement.The goal is to reduce the storage and bandwidth of the DNN model with minimal accuracy compromise.

An approach was proposed named DeepIoT that helps in the deployment of deep learning models on IoT devices. It is capable of performing compression on commonly used deep learning structures, including CNN, fully connected layers, and recurrent neural network. DeepIoT incorporates dropout techniques to remove unimportant connections from the network.
Energy efficient Inference Engine (EIE) was proposed to perform DNN based inference on the compressed network obtained after pruning of redundant connections and weight sharing among different elements. The EIE accelerates the sparse matrix multiplication by adopting weight sharing without losing the accuracy of the model.
A Multi-Tasking Zipping (MTZ) framework was proposed that automatically merges multiple correlated DNN models to perform cross model compression. In MTZ, the compression is performed using layer-wise neuron sharing and subsequent weight update.
Sparse CNN (SCNN) is a DNN compression architecture that improves performance and provides energy efficiency. SCNN uses both weight and activation sparsity, which enhances the power of the compressed DNN. SCNN also employ an effective dataflow mechanism inside the DNN that maintains the sparse weights and activations in the DNN compression. It further reduces unimportant data transmission and storage requirement.

4) Layer Pruning: In this method, some selected layers from the network are removed to compress the DNN model. The layer pruning is highly utilized for deploying DNN model on tiny computing devices, where, we need an ultra-high compression of DNN model. The primary issue with layer pruning is the loss of the semantic structure of the DNN model that generates low-quality features. However, layer pruning results in higher accuracy compromise due to structural deterioration of the DNN model, which should be mitigated to enhance the utility of layer pruning.

FINN is a framework that builds fast and flexible heterogeneous architecture. It utilizes a set of optimization for mapping binarized neural networks to the hardware. The authors have performed the implementation of different DNN components, including, convolutional, pooling, and fully connected layers.

B. Sparse Representation

The sparsity in the representation of the DNN model is exploited to reduce both storage and processing requirement of the DNN model. The sparsity in the DNN model persists in the weight matrices due to the following two reasons :

- The value of stored weights is zero or near to zero. It could be beneficial to remove these weights to reduce the computation and storage requirements.
- The value of maximum stored weights are alike that provides a convenience to replace multiple weights having single connections with single weight having multiplex connections.

These methods could be further categorized as follows :

1) Quantization: The weight quantization reduces the storage requirement of the DNN model, along with the computation requirement. A weight quantization technique was proposed to compress the deep neural network by curtailing the number of bits required for representing the weight matrices.The inference is performed using integer arithmetic.
The integer arithmetic provides higher efficiency than floating-point operation and requires a lower number of bits for representation. A training step was further designed that preserves the accuracy compromise during replacement of floating-point operation with integer operations. The proposed approach thus solve the tradeoff between on-device latency and accuracy compromise due to integer operations.

2) Multiplexing: When weight matrices have similar values of weights. These weights are replaced with a single weight with multiplexed connections. A lightweight neural multiplexer was proposed. The input to the multiplexer is the raw data and resource budget, which jointly determine the model that should be called to perform inference.
The lightweight data with a low budget of resources are inferences on the mobile device, whereas, hard data with high resource budget resources inferences at the Cloud. In the multiplexing approach, multiple models are multiplexed from small to large. The proposed approach is output to a binary vector that indicates the inference is either on the mobile device or Cloud.
A multi-branch CNN architecture was proposed, which multiplexes different recognition and prediction task simultaneously using a single backbone network. The proposed mechanism involves sequential steps, i.e., first, a CNN based backbone network is trained and then different branches of the network are trained by freezing the training of the backbone network.

3) Weight Sharing: In weight sharing, inspite of storing multiple weights for the DNN model, it would be convenient to share the pre-existing weights. The weights stored by any layer could be used by the next layer. The technique of sharing training process involves sharing of training data. However, this data sharing during the model training also leads to privacy compromise.
Deep learning-based modeling and optimization framework was proposed for resource constraint devices. The proposed framework achieves considerable accuracy while consuming limited resources. The framework follows a multitasking learning principle for training shared DNN model. Here, the hidden layers are shared among each other, where, similar weights are associated with multiple links after elimination.

C. Bits Precision

Bits precision in the DNN compression technique reduces the number of bits required for storing the weights in the weight matrices W. The number of bits required for representing weight matrices is suppressed for reducing the storage and computation. The transformation from float-to-integer simultaneously reduces storage and computation requirements. However, it comes up with the challenge of conversion complexity from float to integer along with a higher accuracy compromise. This method could be further categorized into :

1) Estimation using integer: The most straightforward strategy of reducing computation through bits precision is replacing floating-point operations with the integers. Here, the only complexity is to convert floating values to the integer. A mechanism was proposed for opportunistically accelerating the inference performance of the DNN model, named Neuro.ZERO. Four accelerating mechanisms are adopted, i.e., extended inference, expedited inference, ensemble inference, and latent training.
A quantization mechanism was proposed, which uses integer arithmetic despite floating points operations. It relies on the facts that multiplications are inefficient if the operands are wide (floating points 32 bits) and eliminating multiplication leads to performance degradation.Therefore, it could be beneficial to adopt integer multiplication inspite of floating-point operations to reduce the temporary storage for computation.

2) Low Bits Representation: It generalizes the concept of bits precision for using any number of bits for representing weights instead of 8-bits integers only. In the concept of Quantized Neural Networks (QNNs), a DNN network is designed that requires very low precision weight and activations during inference. This QNN scheme highly reduces memory requirements, access latency, and replace floating point operations with bit-wise operations. Therefore, colossal power efficiency is observed.

3) Binarization: Binarization refers to the process of converting floating-point operations into binary operations for reducing the storage and processing requirement of the DNN model. A training method was introduced that incorporates binary activations and weights. The resultant network is called Binary Neural Network (BNN). During the training of the model; the gradient is estimated using binary activations and weights. Binarization helps in reducing storage requirements and provides energy efficiency. Here, the floating-point operations in the DNN model are replaced with binary operations. Two binarization techniques, i.e., deterministic and stochastic, were hence explained.

A framework for building a fast and flexible DNN compression mechanism, named FINN was presented. It provides a heterogeneous architecture that achieves high order flexibility. The framework is a parametric architecture for dataflow and optimized method for classification on limited resource device.

A compact DNN architecture was proposed, named as cDeepArch, where a task is divided into multiple subtasks for reducing the resource requirement of the DNN model. The decomposition of the task in cDeepArch efficiently utilizes the limited storage and processing capacity during the execution of the task. For reducing the DNN model, cDeepArch adopted layer separation, delayed pooling, integration of activation, and binarization for reducing the computation.

D. Knowledge Distillation

The term knowledge distillation is defined as the process of transferring the generalization ability of the cumbersome model (teacher) to the compact model (student) to improve its performance. It is further classified into three parts, i.e., logits transfer, teacher assistant, and domain adaptation. All are briefed below:

1) Logits Transfer: Logits transfer is the simplified approach for knowledge distillation from teacher to student model. Firstly, the teacher model is trained on a given dataset. Next, the logit vectors (output of DNN model before the softmax layer) of the teacher model acts as a soft target for training student model.

2) Teacher Assistant: The insertion of teacher assistant has the main motive to reduce the gap between student and teacher model. It is claimed that when the size of the student model is fixed, then we can not employ a large teacher model. In other terms, the gap between teacher and student model beyond a limit leads to improper knowledge transmission from student to teacher. To mitigate such difficulty in improving the performance of the student model through knowledge distillation, the teacher assistant plays a decisive role.

3) Domain Adaptation: Domain disparity coould be eminent while training student model using logits of the teacher. The domain disparity leads to poor generalization ability of the student model.

A knowledge distillation technique was proposed that helps in shrinking the gap between cumbersome teacher and compact student model, this technique was called as ShrinkTeaNet. The objective of Shrink-TeaNet is simple to train a few parameter student models using cumbersome teacher model.

A framework was designed for Mobile Domain Adaptation (MobileDA) that can learn the transferrable features. The learning is performed in such a way that the structure of the DNN model remains simplified as possible. Cross-domain distillation was used for training student model on IoT devices using cumbersome teacher model on the high-end machine. The main objective of the MobileDA is to perform training on simplified DNN model that can handle the domain shift problem along with achieving significant accuracy.

E. Miscellaneous

This section covers the DNN compression techniques that do not meet the different categorization criteria discussed in the previous sections. These are listed below:

1) DeepSense: DeepSense uses CNN and RNN based feature extraction to avoid the difficulty in designing manual features. As it is not always convenient to find highly robust features that can accommodate noise in the sensory data and rapidly changing user behaviour. DeepSense uses parallel execution of multiple sensors data on DNN models that decrease its complexity to run on mobile devices.

2) FastDeepIoT: The framework is named as FastDeepIoT that provides an accuracy preserving DNN compression for IoT devices. FastDeepIoT exploits the non-linear relation between the structure of DNN model and its execution time. The objective behind the FastDeepIoT is to develop a DNN model that check the conditions which are responsible for non-linearity without any knowledge of library and hardware.

3) AdaDeep: The tradeoff between performance and resources of the DNN model was explored on the embedded device. Here, the authors determine the user-specified needs to estimate the appropriate DNN model for its deployment on the embedded device. This framework was referred to as AdaDeep. The framework automatically specifies a combination of compression techniques for a given DNN model.

4) NestDNN: A DNN framework was presented named NestDNN. The framework emphasized on the availability of dynamic resources on mobile devices. It indicates the DNN model running on a mobile device faces resource scarcity when different applications are running on the mobile device. Therefore, it is beneficial to select optimal resources while performing inference on the mobile device. NestDNN has the main objective of providing on-device computation of DNN model without Cloud support.

5) DeepEye: The DeepEye provide a matchbox size computational unit having attached camera sensor. The small size device can process the image captured by the camera in the same way the image processed at high-end machines (Cloud). On the matchbox size device, the DNN model runs successfully to perform image recognition with significant accuracy in short duration.

6) DeepApp: A deep reinforcement learning framework was proposed named DeepApp. It determines the mobile device usage pattern using historical data of the applications being used by the user. The features from the historical data are extracted using compressed DNN model. Further, the compressed DNN classifier recognizes the different usage pattern.

7) DeepFusion: A unified framework named DeepFusion was proposed that incorporates values of multiple sensors for recognition of different IoT based task. DeepFusion parallelly combines spatial features extracted from CNN for each sensory values. This parallelisation reduces the complexity of the DNN model and makes it suitable for its deployment on mobile devices.

8) ShuffleNet: Two crucial operations were incorporated, i.e., pointwise group convolution and channel shuffle for compressing DNN model. The proposed approach was named as ShuffleNet that reduces the power consumption by reducing the floating-point operations without accuracy compromise.

V. Results and Future Directions

We come up with five broad categories, i.e., network pruning, sparse representation, bits precision, knowledge distillation, and miscellaneous techniques. All these categories seem to be the predominating area of research that encourages effective utilization of DNN model on resource constraint devices. It is beneficial to provide a thorough overview of the DNN compression technique that can meet out the limited storage and processing capacity available at the resource constraint IoT devices.

The following researches could prove to be milestones in future :

- Recurrent neural networks
- DNN compression techniques face performance degradation. It is still a challenge to improve the performance even better
- The dynamic resources on tiny computing devices need more exploration.