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

Abhishek Kumar
Computer Science and Engineering
IIT PATNA,2nd year
Bihta, India

I. Introduction

In recent years, Deep Neural Network (DNN) based IoT applications have achieved dominance in different application domains, including smart home, agriculture, locomotion mode recognition, etc . It is because the DNN provides automatic feature extraction capability from a large amount of raw data. The DNN models not only eliminates the human intervention for calculating domain-related features but achieves excellent performance CNN and RNN are the components of DNNs models. 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.
Besides all these advantages, the DNN models require a significant amount of resources including, energy, processing capacity, and storage. This resource requirement reduces the suitability of DNN model for Resources constraint Devices (RCDs). Therefore, to overcome these shortcomings of DNN model, different DNN compression techniques have been proposed in the existing literature.

II. Methods and their Description

A. Network Pruning

Network Pruning is further divided into four categories depending on the existing work and they are and their descriptions are as follows:

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.
An approach (named VarGNet) is 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, they adopt angular distillation loss to improve the performance of the generated lightweight model. Another authors proposed a DNN compression technique based on the streamlined architecture that uses depthwise separable convolutions to build a lightweight model. The authors named the technique 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. In MobileNets, the depthwise convolution applies a single filter to each input channels. Further, 1×1 convolution is performed to combine inputs into a new set of output. MobileNets achieve faster inference by exploiting the sparsity of the dataset. The role of the width multiplier wd is to thin a network uniformly at each layer. For a given layer, the number of input channels M becomes (wd)M. The resolution multiplier rm is applied to the input image and the same multiplier subsequently reduces the internal representation of every layer. However, none of the existing work helps in diminishing the accuracy compromise due to channel pruning.

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. As it assumed that the increment in the number of filters, improves the distinguishable characteristics of the spatial features generated from the CNN model. However, this increment in the convolutional filters leads to a significant increase in the number of floating-point operations in the DNN model which increases its computations requirements. Therefore, the elimination of the unimportant filters plays a decisive role in reducing the computational requirements of the DNN model. The existing work on filter pruning are discussed as follows:
An author proposed a DNN compression technique that removes the sparsification in fully connected layer and convolutional filters. The main motive was to reduce the storage requirement of device and processor resource in both training and inference phases. The basic operational principle of this work is the hypothesis that computational and space complexity of DNN model can be considerably increase using sparsification of layers and convolutional filters.
Some authors established a trade-off between the number of parameters and performance compromise due to model compression. The basic principle of this mechanism is to prune parameters that result in little performance degradation. To estimate the importance of a parameter, they have used the criteria of performance loss to identify the redundancy. To achieve a compressed model with a small size, the number of filters preserved must be less as possible to achieve a considerable accuracy. The problem can be treated as to find a compact binary representation that can prune maximum filters while preserving the performance to a greater extent. This mechanism can be further improved by incorporating the low-rank estimation. An author proposed a mechanism named

DeepMon to provide deep learning inference on mobile devices. They claim to run the inference in limited time and provides energy efficiency using the Graphical Processing Unit (GPU) on the mobile device. The authors proposed an optimization mechanism 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. Thus, removing unimportant filters and connections for proving faster inference.

Authors speed up the test-time evaluation of the large DNN model, designed for object recognition tasks. The authors have exploited the redundancy present within the convolutional filters to derive approximations that significantly reduce the required computation. They have started compression of each convolutional layer by finding an appropriate low-rank approximation.

The filter pruning technique talked above have successfully reduced the number of floating-point operations in the CNN. However, the fully connected layers and recurrent layers also contribute a major portion of floating-point operations in the DNN model, which should not be ignored during DNN compression.

3) Connection Pruning: The number of input and output connections to a layer of DNN model determines the number of parameters. These parameters can be used to estimate the storage and computation requirement of the DNN model. Since the DNN model requires a large number of parameters in their operations; therefore, it is convenient to reduce the parameters by eliminating unimportant connection for different layers in the DNN model.

Some authors proposed an approach that helps in the deployment of deep learning models on IoT devices. Thus, named the approach as DeepIoT. The proposed approach 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. In DeepIoT, small dense matrices are obtained by performing compression of the different structures in the network. These dense matrices contain a minimal number of non-redundant elements, such as input and output dimension of a layer, filters, etc. The process of obtaining dense matrices must safeguard the network from higher accuracy compromise.

An author proposed an Energy efficient Inference Engine (EIE) 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. In EIE, the processing is performed using an array of processing elements. Each element simultaneously partitions the network in SRAM and perform the computation of their respective part. Some authors proposed a Multi-Tasking Zipping (MTZ) framework 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. Here, the authors induced a minimal change in error upon layer sharing.

Some authors developed a quantitative characterization approach for a deep learning model to make its deployment on embedded devices. It not only provides a better selection mechanism for deployment of DNN model on the embedded device but also guides the development of compression technique using a wiser connection pruning. In [39] authors proposed a zero-valued weight estimation from a network during training that is generated from ReLU operation. The authors named the approach as Sparse CNN (SCNN). It 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.

The connection pruning technique provides a high order compression of the DNN model in contrast with channel pruning and filter pruning. However, a major concern of connection pruning technique is the elimination of important connection during DNN compression, needs more emphasis, such as pruning and splicing.

4) Layer Pruning: In this, 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.. The primary issue with layer pruning is the loss of the semantic structure of the DNN model that generates low-quality features.

The prior studies on layer pruning for DNN compression are elaborated as follows. Some authors presented a framework that builds fast and flexible heterogeneous architecture, named FINN. They utilized a set of optimization for mapping binarized neural networks to the hardware. The authors have performed the implementation of different DNN components, including, con- volutional, pooling, and fully connected layers. The implementation was conducted in such a manner that meet out the performance demand of the users. They claim to perform millions of classifications per second with microsecond latency on embedded devices.Some authors compared the performance of different classification approaches and proposed a pruning method for deploying DNN model on embedded devices. They have taken three embedded devices, including smartphone, smartwatch, and Raspberry Pi, for deploying compressed LSTM model. They claim to achieve significant accuracy by performing compression up to 25%.

The above discussed methods have reduced both storage and computation requirement of DNN model by providing an

ultra highlevel pruning. However, the 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.

B. Sparse Representation

Sparse representation is further divided into 3 parts which are as follows:

1) Quantization: This section covers the sparse representation technique that involves weight quantization in the DNN model. The weight quantization reduces the storage requirement of the DNN model, along with the computation requirement.

An authors proposed a weight quantization technique to compress the deep neural network by curtailing the number of bits required for representing the weight matrices. Here, authors try to reduce the number of weights that should store in the memory. In doing so, the same weights are eliminated, and multiple connections are drawn from a single remaining weight.

An author proposed a quantization technique, where, 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. Authors further design a training step 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.

The quantization technique discussed above for compressing DNN model cover the model reduction by providing optimal arrangements of weight matrices. However, the negative consequences of weight quantization and its complexity of estimation are missing for the existing work.

2) Multiplexing: The multiplexing plays a vital role in reducing the storage requirement of the DNN model. The multiplexing is highly effective when weight matrices have similar values of weights. These weights are replaced with a single weight with multiplexed connections. The overview of the existing literature on multiplexing for DNN compression are as follows:

Some authors proposed a lightweight neural multiplexer. 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.

Some authors proposed a multi-branch CNN architecture, which multiplexes different recognition and prediction task simultaneously using a single backbone network. The proposed mechanism involves sequential steps, i.e first, the authors have trained a CNN based backbone network then train different branches of the network by freezing the training of the backbone network. The authors proved the multiplex approach that preserves both time and energy of the system while achieving significant accuracy on an embedded device.

The existing works helps in DNN compression for reducing storage and computation requirements. However, none of the work tried to multiplex the weights, which can be an effective solution for reducing the storage of the DNN model.

3) Weight Sharing: Weight sharing proves to be the important criteria for reducing both storage and computation requirements of the DNN model. In weight sharing, inspite of storing multiple weights for the DNN model, it would be convenient to share the pre-existing weights.

Some authors emphasized the technique of sharing training process that involves sharing of training data. However, this data sharing during the model training also leads to privacy compromise. Therefore, the authors proposed a method that discloses a few samples of the training data, which are sufficient for a data analyst to get insight into the DNN model. To reduce the discloser of the training data, authors have used the max-margin approach that extracts most identifiable training data. These identifiable data significantly contributes to obtaining a decision boundary.

Some authors proposed deep learning-based modeling and optimization framework 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.

The above discussed technique helps in reducing storage and computation requirement of the DNN model. However, the complexity of weight sharing also comes into the picture, while, representing DNN model using minimal weights.

C. Bit Precision

Bit Precision is further divided into 3 categories which are as follows:

1) Estimation using Integer: This section covers very simple 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. The existing work on bits precision using integer are elaborated as follows:

Some authors proposed a mechanism of opportunistically accelerating the inference performance of the DNN model, named Neuro.ZERO. They adopt four accelerating mechanisms, i.e extended inference, expedited inference, ensemble inference, and latent training. Further, the authors

proposed two algorithms for applying these accelerating mechanisms. They have adopted integer arithmetic by replacing floating-point operations. The authors emphasized on facilitating runtime adaptations, where, accuracy during runtime suppose to increase when the resources are available. Some authors proposed a quantization mechanism, 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.

The above techniques in this section however lack in estimating the amount of memory we can preserve through integer estimation.

2) Low Bits Representation: This section extends the concept of the integer representation to reduce the storage and computation of DNN model. It generalizes the concept of bits precision for using any number of bits for representing weights instead of 8-bits integers only. In this , the authors design a DNN network that requires very low precision (e.g. 1 bits, 4 bits, and so on) weight and activations during inference. While training the quantized weights and activations helps in estimating the gradient. This QNN scheme highly reduces memory requirements, access latency, and replace floating point operations with bit-wise operations.

However in this method , selecting an optimal number of bits for any estimation is a tedious task.

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.
Some authors presented a compact DNN architecture, where a task is divided into multiple subtasks for reducing the resource requirement of the DNN model. The authors named the approach as cDeepArch. The decomposition of the task in cDeepArch efficiently utilizes the limited storage and processing capacity during the execution of the task. The authors emphasized that the unorganized compression of the DNN models results in unreliable and low performance for recognizing multiple tasks. Further, the authors established a quantitative measure to estimate the reduction in resources and available resources. cDeepArch incorporates offline training followed by the execution of the task on user smartphone. It also preserves the privacy of sensitive information. Here, to provide privacy of the sensitive information, it is converted into image format for training a DNN model. For reducing the DNN model, cDeepArch adopted layer separation, delayed pooling, integration of activation, and binarization for reducing the computation.

The proposed binarization approaches achieve a high-order compression but with performance degradation. Thus, it could be beneficial to mitigate this degradation for the successful adaptation of binarization techniques in DNN compression.

D. Knowledge Distillation

Knowledge Distillation is further divided into 3 categories which are as follows:

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 D. Next, the logit vectors (output of DNN model before the softmax layer) of the teacher model acts as a soft target for a training student model. The knowledge distillation using logits transfer improves the generalization ability of the student model and helps in improving its performance. Different logits transfer mechanism are described as follows:
Some authors proposed a knowledge distillation technique for improving the performance of a compressed DNN model using the generalization ability of the pre-trained cumbersome model. The cumbersome model contains the ensembled output of multiple models, thus achieve higher generalization ability. The deployment of the cumbersome model on an embedded device is impractical due to substantial resource requirements. To overcome such limitations, the small model is deployed on the embedded device, with accuracy improvement using the cumbersome model. The principle objective is to distil the knowledge from large model to small model by training small model in such a way that it achieves similar performance as the cumbersome model. While performing the transmission of the generalization ability of the cumbersome model to the small compressed model, the class label probabilities will act as a soft target for training the small model.
Some authors have designed a private model compression framework, named private model compression framework (RONA). The authors have highlighted the need for DNN compression to provide significant privacy to the user by analysing the collected data on the limited resource devices. Using knowledge distillation, the accuracy of the compressed model is improved by jointly using a hint, distillation, and self learnings. In performing knowledge distillation, the cumbersome model is carefully perturbed for enforcing a high level of privacy. Therefore, the authors try to meet two crucial goals simultaneously, i.e, model compression and preserve privacy.

The existing work on logits transfer in knowledge distillation improves the performance of the compressed DNN model. However, the logits of original pre-trained DNN model sometimes lead to overfitting, which leads to poor generalization ability of the compressed model.

2) Teacher Assistant: Some authors studied the concept of knowledge distillation from a different perspective. The

authors introduced teacher assistant in between teacher and student model during knowledge distillation. This insertion of teacher assistant has the main motive to reduce the gap between student and teacher model. They claim, 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. The authors also suggest the futuristic approach of inserting multiple teaching assistants for providing high order compression with minimal accuracy compromise.

The above approach will unfortunately increase the training time by nearly two-fold. First, the training of teacher assistant is performed using teacher model followed by training of student model using teacher assistant. Despite significant improvement in the student model performance, the complexity of training could be a loophole of the proposed approach.

3) Domain Adaptation: The knowledge distillation techniques discussed in the previous sections do not consider the consequence of domain disparity while training student model using logits of the teacher. The domain disparity leads to poor generalization ability of the student model. To handle domain disparity, different domain adaptation techniques have been proposed one of which is given below:
Some authors emphasized on the problem of domain adaptation that is encountered during the training of the DNN model on embedded Internet of Things (IoT) devices. The authors discovered the DNN model achieves prominent result on the high-end machine but suffers from degradation in accuracy and efficiency on IoT devices. They proposed a framework 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. The authors use cross-domain distillation 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. In other words, the training phase addresses the domain adaptation using cumbersome DNN model running on a high-end machine and testing phase is performed using a simplified DNN model deployed on the embedded device. Further, the authors proposed an algorithm that simultaneously optimizes different loss function for handling domain adaptation.

The domain adaptation technique proposed above have successfully covered the domain disparity problem. However, the gap between the student model and teacher model sometimes hampers the performance of the domain adaptation technique. Thus, it could be beneficial to adopt the teacher assistant in domain adaptation problem to get overall benefits.

III. Conclusion

All these categories seem to be the predominating area of research that encourages effective utilization of DNN model on resource constraint devices. As, demand for IoT based applications is proliferating, which enables data scientists to incorporate DNN models in IoT. Therefore, 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 overview of different categories of DNN compression provides several research possibilities. Even though several work have been done in the DNN compression, there are still a lot more to be uncovered.