IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 27 April 2024, accepted 31 May 2024, date of publication 10 June 2024, date of current version 5 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3411776

SURVEY

Applications of Pruning Methods in Natural Language Processing

MARVA TOUHEED[1], UROOJ ZUBAIR[1], DILSHAD SABIR[1], ALI HASSAN[2], MUHAMMAD FASIH UDDIN BUTT[1,3], (Member, IEEE), FARHAN RIAZ[4], WADOOD ABDUL[5], AND RASHID AYUB[6]

[1]Department of Electrical and Computer Engineering, COMSATS University Islamabad, Islamabad 45550, Pakistan
[2]Department of Computer and Software Engineering, College of Electrical and Mechanical Engineering, National University of Sciences and Technology, Islamabad 44000, Pakistan
[3]Next-Generation Communications Research Group, COMSATS University Islamabad, Islamabad 45550, Pakistan
[4]School of Computer Science, College of Health and Science, University of Lincoln, LN6 7DL Lincoln, U.K.
[5]Department of Computer Engineering, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia
[6]Department of Science Technology and Innovation, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia

Corresponding author: Ali Hassan (alihassan@ceme.nust.edu.pk)

This work was supported by Researchers Supporting Project number (RSPD2024R1051), King Saud University, Riyadh, Saudi Arabia.

ABSTRACT Deep neural networks (DNN) are in high demand because of their widespread applications in natural language processing, image processing, and a lot of other domains. However, due to their computational expense, over-parameterization, and large memory requirements, DNN applications often require the use of substantial model resources. This strict requirement of latency and limited memory availability are hurdles in the device deployment of these technologies. Therefore, a common idea could be to mitigate the DNN-based models' size without any performance degradation using different compression techniques. During the last few years, a great deal of progress has been made in the field of Natural Language Processing (NLP) using deep learning approaches. The objective of this research is to offer a thorough overview of the various pruning methods applied in the context of NLP. In this paper, we review several recent pruning-based schemes used for converting standard networks into their compact and accelerated versions. Traditionally, pruning is a technique for improving latency, reducing model size, and computational complexity which is a viable approach to deal with the above-mentioned challenges. In general, these techniques are divided into two main categories: structural and unstructured pruning methods. Structural pruning methods are further classified into filter, channel, layer, block, and movement pruning. Whereas, neuron, magnitude-based, and iterative pruning lie in the category of unstructured pruning. For each method, we discuss the related metrics and benchmarks. Then recent work on each method is discussed in detail, which provides insightful analysis of the performance, related applications, and pros and cons. Then, a comparative analysis is provided to analyze the differences among approaches. Finally, the paper concludes with possible future directions and some technical challenges.

INDEX TERMS Pruning, convolution neural networks, natural language processing, DNN, model compression, acceleration.

I. INTRODUCTION

Deep neural networks (DNNs) have demonstrated remarkable success in a variety of domains, including image recognition, natural language processing (NLP), bio-informatics, and computer vision [1]. These DNN models have attained impressive performance in complex tasks including image classification, machine translation, sentiment analysis, and speech recognition. However, their success comes at the expense of high processing requirements, over-parameterization, and large memory footprints [1].

To address these challenges and facilitate the implementation of DNN models in environments with limited resources, researchers have explored various techniques for model

The associate editor coordinating the review of this manuscript and approving it for publication was Prakasam Periasamy.

compression and optimization. One such technique that has garnered considerable attention is pruning, which involves eliminating certain components from the network to reduce its size or complexity, such as weights, connections, or neurons, that are considered less important or redundant. The objective of pruning is to optimize the network's efficiency, reduce computational requirements, and potentially improve its generalization performance.

Although there are other methods of network size reduction, like weight quantization [2], which lessens the data type precision of weights without any accuracy degradation, the technique is often engaged for hardware implementation as it provides considerable efficiency for the network training and testing phases. In general, the pruning method completely removes the insignificant connections, weights, blocks, and even the layers with almost no accuracy loss. Therefore, the implications of mitigating the memory and processing requirements are far greater for the pruning.

Pruning is typically performed after a neural network has been trained or during the training process itself. Bellec et al. [3] devise a unique method for pruning neural networks after training. Their methodology involves iteratively eliminating connections with low importance and fine-tuning the remaining weights, resulting in highly sparse networks without sacrificing performance. Various pruning techniques and criteria can be applied to identify which components to prune. These techniques may involve analyzing weight magnitudes, gradients, activations, or other factors to determine the importance of individual components. By removing unnecessary or less impactful components, pruning can lead to a more compact network that requires fewer resources for training, inference, and deployment. Pruning can also help mitigate issues such as over-fitting and improve the network's interpretability and generalization capabilities. After pruning, the pruned network may undergo further fine-tuning or retraining to restore or even improve its performance. The overall goal of pruning is to achieve a more efficient and effective neural network by selectively removing redundant or less influential components while maintaining or improving its desired functionality.

In addition to allowing the use of NLP models on devices with limited resources such as smartphones and embedded systems, pruning also facilitates faster inference and reduced memory usage. The effectiveness and implications of pruning techniques in NLP have been extensively investigated in numerous research works. Han et al. [4] introduced the concept of deep compression, which involves a combination of Huffman coding, pruning and quantization for the compression of neural networks. Their study demonstrated that pruning can achieve significant compression (up to 90% on the weights) without a substantial loss in accuracy.

Islam and Alawad propose [5] a novel method for reducing the complexity of deep learning models in natural language processing (NLP) tasks, making them more suitable for deployment in resource-constrained environments. The approach combines compressive sensing and Bayesian learning to identify and represent the most important weights in the model in a compressed form. By stochastically pruning non-critical weights, the model's accuracy is preserved. The authors evaluate their approach on various NLP tasks, such as sentiment analysis and text classification, comparing it with other compression methods like weight pruning and knowledge distillation. Results indicate that their method can significantly reduce model complexity (90% compression) while maintaining high accuracy ($<$1% drop). The proposed technique offers a promising solution for deploying large language models in resource-limited settings, striking a favorable trade-off between model size and accuracy compared to other methods.

Molchanov et al. [6] built upon structured pruning with the Group-wise Brain Damage (GBD) algorithm, which prunes filters within convolutional layers based on their importance. GBD pruning significantly reduces the amount of parameters (up to 95%) while maintaining accuracy. Li et al. [7] improved upon magnitude pruning with iterative magnitude pruning, which combines many pruning and fine-tuning processes. Their experiments demonstrated that iterative magnitude pruning can achieve high sparsity (up to 80%) while maintaining comparable performance. Combining pruning techniques with other model compression methods has also garnered attention. Zhu and Gupta [8] proposed combining pruning with low-rank matrix factorization, achieving higher compression rates. Their experiments on language models demonstrated significant model size reduction (up to 20 times) with minimal performance loss.

In addition to model compression, pruning techniques have been utilized to increase the interpretability of many NLP models. Voita et al. [9] explored pruning as a tool for interpretability in neural machine translation (NMT) models. By pruning the attention mechanism, they were able to identify important features and gain insights into the decision-making process of the model. Pruning methods, in general, present potential approaches for lowering the size and computing complexities of DNNs in NLP. of DNNs in NLP. Through the exploration of various pruning methods and their combinations with other compression techniques, researchers aim to develop more efficient and compact models without compromising performance or interpretability.

Onan et al. [10] presented in the text is focused on the novel concept of text augmentation in the field of NLP. Text augmentation is a powerful concept that can significantly improve the performance of a wide range of downstream tasks. GTR-GA, the proposed approach, utilizes graph-based neural networks and genetic algorithms to create diverse and high-quality augmented text data. The model utilizes a graph attention network model, called HetGAPN, to obtain node representation of a heterogeneous graph over text features. The experimental results show that GTR-GA outperforms the text augmentation baselines ASC and TowEute and completes

close to the state of the art in various NLP tasks, including sentiment analysis, data scarcity and text classification.

The proposed work [11] resolves the challenge of a limited amount of annotated data for Turkish NLP and sentiment analysis. They suggest using text data augmentation as a solution by creating synthetic data to increase the diversity and volume of the annotated dataset. However, there has been a lack of research on this method for Turkish and other languages with fewer resources. To fill this gap, a new ensemble approach to text data augmentation for Turkish sentiment classification is introduced in the paper. This method combines specific and general transformations to enhance the training dataset. The effectiveness of this approach is demonstrated on the TRSAv1 dataset, showing superior results compared to existing data augmentation techniques.

The authors [12] address the issues of creating quality labeled data for training machine learning models, specifically in NLP. Human annotators may provide inconsistent manual annotations due to varying skill levels. To address this issue, a new framework called SRL-ACO is suggested in the paper. SRL-ACO integrates Semantic Role Labeling (SRL) and Ant Colony Optimization (ACO) to produce additional training data for NLP models. SRL identifies the roles of words while ACO generates new sentences maintaining these roles. This method improves the accuracy of NLP models by offering additional data without manual labeling. Results from experiments on seven text classification datasets, including sentiment analysis, are also discussed. Tests on seven datasets for text classification, such as sentiment analysis and identifying toxic text, show that SRL-ACO is effective in boosting the performance of classifiers in various NLP tasks.

The proposed research scheme [13] introduce a new hierarchical graph-based text classification model that incorporates contextual node embedding and BERT-based dynamic fusion to better understand the intricate connections between nodes in the hierarchical graph for more precise text classification. The model consists of seven steps: Linguistic Feature Extraction, constructing Hierarchical Nodes using domain-specific knowledge, Contextual Node Embedding, learning Multi-Level Graphs, Dynamic Text Sequential Feature Interaction, Attention-Based Graph Learning, and Fusion with BERT. By combining their framework with BERT, the researchers achieve improved classification results, as demonstrated by evaluations on standard datasets.

Onan [14] explore the sentiment analysis has evolved in computational linguistics, focusing on the success of deep neural network models such as CNNs and RNNs like LSTM and GRU. It also acknowledges the obstacles, like increased dimensions in feature space and the equal weighting of features in these models. To overcome these challenges, a new approach is suggested: a bidirectional convolutional recurrent neural network. This design uses bidirectional LSTM and GRU layers to understand both previous and upcoming information at the same time. It also includes a group-wise improvement system to focus on key elements and minimize less important ones. Additionally, convolution and pooling layers are used to identify advanced characteristics and decrease the dimensionality of features. Testing has shown that this design outperforms current results in sentiment analysis tasks.

This paper is organized into multiple sections. Section II describes the motivation behind the implementation of pruning in NLP-based applications. It also expresses the challenges related to their implementation. Related metrics and benchmarks utilized for the evaluation of pruning techniques are detailed in section III. Structured and unstructured pruning schemes are elaborated in section IV. The section provides the details of pruning methods. Further, section V compared the related literature review in the paper. While sections VI and VII give the conclusion and future work.

II. MOTIVATION AND CHALLENGES OF PRUNING IN NLP

A. MOTIVATION

- **Model Size Reduction** Pruning allows for the elimination of redundant or insignificant parameters, leading to significant reductions in model size and memory footprint [15].
- **Computational Efficiency** Pruned models require fewer computations during both training and inference, enabling faster processing and lower energy consumption [16].
- **Deployment on Resource-Constrained Devices** Compact pruned models are more appropriate for use on computing-constrained platforms, such as smartphones and embedded computers. [17].
- **Interpretability** Pruning can help identify the most important and relevant features or components of the model, aiding in model interpretation and understanding [18].

B. CHALLENGES

- **Impact on Model Performance** Pruning can potentially lead to a degradation in model performance, including accuracy, fluency, or coherence, depending on the pruning method and the pruning rate applied [19].
- **Transferability** Pruned models may not generalize well to unseen data or different tasks, necessitating careful fine-tuning or transfer learning strategies [16].
- **Fine-tuning Strategies** Determining the optimal fine-tuning approach to recover and retain model performance after pruning is an ongoing research challenge [20].
- **Robustness** Pruning methods need to be robust to variations in data distribution, input format, and linguistic phenomena to ensure consistent performance across different NLP tasks and domains [21].

In this paper, we examine the various pruning methods used in NLP and discuss their advantages, limitations, and associated challenges. By understanding these motivations

and challenges, we aim to provide insights into the cutting-edge pruning methods for NLP models.

III. METRICS AND BENCHMARKS

There are various metrics and benchmarks to evaluate the performance of pruning in natural language processing. Some of them are described below.

A. METRICS

- **Model Accuracy**
  This metric evaluates the pruned model's performance on a specific NLP task. To evaluate the effect of pruning on accuracy, it contrasts the performance of the pruned model with the original, unpruned model.
- **Parameter Reduction**
  This metric quantifies the extent to which parameter reduction is achieved through pruning, providing a measure of the size and complexity of the pruned model based on the total number of pruned parameters, connections, or units.
- **Inference Speed**
  This metric evaluates the speed or runtime of the pruned model during inference. It measures the time taken by the pruned model to process a given input compared to the unpruned model, highlighting the efficiency gained through pruning.
- **Memory Footprint**
  This metric assesses the reduction in memory usage achieved by pruning.In comparison to the unpruned model, it compares the amount of memory needed to hold the pruned model's weights, activations, and intermediate tensors.
- **Compression Ratio**
  This metric calculates the compression ratio achieved through pruning. The size ratio of the pruned model to the original model is quantified, providing an indication of the level of compression achieved.
- **Robustness**
  Reporting adversarial robustness in addition to accuracy [22]. Smaller neural networks are more susceptible to adversarial attacks [23]. Compressed pretrained language models are much less resistant on out-of-distribution (OOD) data in addition to adversarial resilience [24].

B. BENCHMARKS

To evaluate the performance of pruning techniques in NLP, several widely-used benchmarks have been employed. These benchmarks serve as standard tasks or datasets for assessing the effectiveness of pruning methods across different NLP domains. The following benchmarks are commonly used in the literature:

- **Text classification**
  The benchmark for text classification commonly includes datasets such as the IMDb movie reviews dataset [25], AG's News dataset, or the 20 Newsgroups dataset. These datasets consist of labeled text documents across various categories, and the accuracy measure is used to assess the classification performance of the pruned models.
- **Machine Translation**
  For machine translation tasks, the benchmarks often involve large-scale datasets such as the Workshop on Machine Translation (WMT) dataset [26] or the International Workshop on Spoken Language Translation (IWSLT) dataset. These datasets contain parallel sentences in multiple languages, and evaluation metrics such as Bilingual Evaluation Understudy (BLEU) score are used to assess the quality of translation produced by pruned models.
- **Sentiment Analysis**
  Sentiment analysis benchmarks commonly include datasets such as the Stanford Sentiment Treebank [27], the Amazon Product Reviews dataset, or the Twitter Sentiment Analysis dataset. These datasets contain text samples that have been labeled with sentiment polarity, and measures like F1 score or accuracy are used to assess how well the pruned models classify sentiment.
- **Named Entity Recognition**
  For named entity recognition tasks, popular benchmarks include the CoNLL-2003 dataset [28], the OntoNotes dataset, or the ACE (Automatic Content Extraction) dataset. These datasets provide labeled text documents with annotated named entities, and different metrics like precision, recall, and F1 score are used to evaluate the pruned models' effectiveness in identifying named entities.

IV. METHODS

Different pruning techniques exist in NLP with the goal of reducing the size and extent of neural network models. The paper discusses the pruning schemes that are summarized in Figure 1.

Here are some commonly used pruning methods along with their brief descriptions:

A. STRUCTURED PRUNING

Structured pruning [29], [30] [31], [32] refers to the removal of entire structures, such as filters, channels, or layers, from a neural network. This type of pruning maintains the overall structure and shape of the network while reducing its size and computational complexity. Structured pruning is particularly effective for models with convolutional or recurrent layers. Neural networks used in NLP, such as recurrent neural networks (RNNs) or transformer models, often consist of numerous parameters and layers, making them computationally expensive and memory-intensive. Structured Pruning is depicted in Fig 2.

Structure pruning aims to mitigate these issues by identifying and removing unnecessary components of the neural network while preserving its performance as much as possible. This process typically involves eliminating

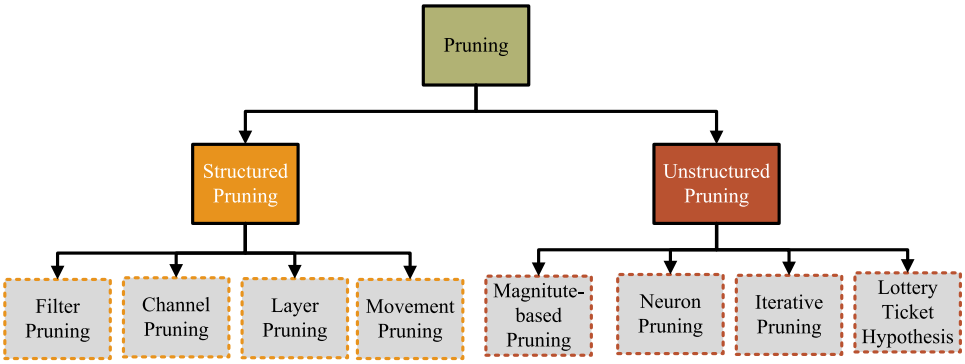


FIGURE 1. Hierarchical representation of various pruning approaches.

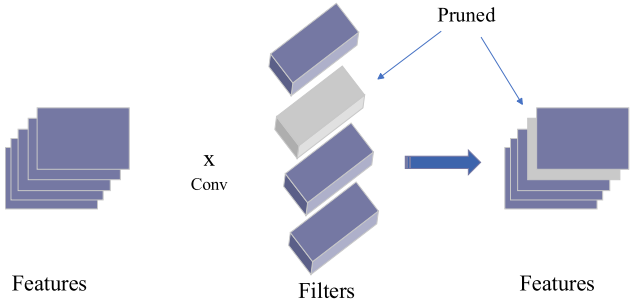


FIGURE 2. Typical structured pruning applies the criteria-based reduction of filter blocks [29].

connections between neurons, removing entire neurons or layers, or reducing the dimensionality of the network's weight matrices.

The goal of structure pruning is to obtain a more compact and efficient model that requires fewer computational resources for training and inference, while maintaining a similar level of accuracy or performance. Pruning can help the model operate more quickly, use less memory, and can be deployed on systems with little computing power, such as mobile phones or embedded systems.

Pruning techniques in NLP can be based on various criteria, including the magnitude of weights, their importance to the network's performance, or their contribution to the overall loss function. Several algorithms and heuristics have been proposed for structure pruning, such as magnitude-based pruning, iterative pruning, or group sparsity regularization.

Peng et al. [33] explore structured pruning techniques in self-supervised pre-trained models specifically designed for speech recognition and understanding. Self-supervised learning refers to training models on unsupervised tasks using the available data's inherent structure, without relying on manual annotations. Their objective was to reduce model size by removing redundant parameters while preserving speech recognition performance. The authors propose a structured pruning method to lessen the size and complexity of self-supervised pre-trained models while maintaining their performance on speech-related tasks. The key contributions and findings of the paper include the following:

1) Pruning Strategy: The authors present a specific pruning strategy tailored for self-supervised pre-trained models in the speech domain. This strategy identifies and removes less important structures or components, such as layers or neurons, based on their significance to the model's overall performance.
2) Performance Analysis: The paper evaluates the impact of structured pruning on speech recognition and understanding tasks. It assesses the model's accuracy and efficiency before and after pruning, demonstrating the potential benefits of structured pruning in reducing model size and computational requirements while preserving task performance.
3) Comparison with Baselines: The authors compare their proposed structured pruning approach with other baseline methods commonly used for model compressions, such as unstructured pruning or weight quantization. The comparison highlights the advantages and effectiveness of structured pruning specifically for self-supervised pre-trained models in the speech domain.

The results show significant compression and improved accuracy compared to the original model, as validated by experiments on LibriSpeech and SLURP datasets.

Ma et al. [29] proposes Large Language Model-Pruner (LLM-Pruner), a task-agnostic compression method, aiming to maintain the multi-task solving and language generation abilities of the original LLM while minimizing reliance on the extensive training dataset. LLM-Pruner adopts structural pruning, removing non-critical coupled structures based on gradient information to preserve most of the LLM's functionality. The pruned models can be efficiently recovered through tuning techniques in a short time and with minimal data. Experimental results on three LLMs, Large Language Model Meta AI (LLaMA), Vicuna, and General Language Model (ChatGLM), demonstrate that the compressed models perform well in zero-shot classification and generation tasks.

McCarley et al. [30] investigate the utilization of structured pruning in a Bidirectional Encoder Representations

from Transformers (BERT)-based Question Answering (QA) model. The primary objective is to reduce the model's size by removing redundant parameters while preserving its QA performance. They present a novel approach to optimize the computational efficiency of BERT-based question-answering models through structured pruning. BERT is a popular transformer-based approach for applications involving natural language processing. Deploying it on devices with limited resources or in situations where real-time inference is necessary, however, is difficult due to its vast size and computing requirements. The proposed method focuses on identifying and removing redundant parameters in BERT by leveraging the structured sparsity pattern present in the model's attention heads. By pruning attention heads that contribute minimally to the model's performance, significant model size reduction is achieved without sacrificing accuracy. When compared to the original BERT model, the suggested technique gains considerable pruning ratios while keeping comparable performance, which is demonstrated by testing on benchmarks for answering questions. This work contributes to the development of more efficient and deployable BERT-based question-answering systems.

Yang et al. [31] address the computational resource limitations associated with pre-trained language models used in NLP by introducing TextPruner which is a dedicated open-source toolkit developed to facilitate model pruning, aiming to enable efficient and straightforward compression of models. It provides structured post-training pruning techniques, such as vocabulary pruning and transformer pruning, for streamlined implementation. These methods enable the reduction of the model size without the need for retraining, thus making the pruning process more efficient. The toolkit is flexible and may be used for a variety of applications including pre-trained language models and NLP tasks. In addition to structured pruning, the authors propose a self-supervised pruning technique that does not need any kind of labeled data. This method allows for further reduction of the model size by removing unnecessary parameters without compromising performance. For the effective evaluation of TextPruner, the authors conduct experiments on several NLP tasks. The results demonstrate that TextPruner effectively reduces the model size without retraining, thus addressing the computational resource limitations. The toolkit proves to be valuable in enabling the wider application of pre-trained language models by making them more resource-efficient.

In their paper Wang et al. [32] explore the need for large language models and proposes a structured pruning approach to reduce their size without sacrificing performance. As language models have become larger, their resource requirements and latency have also increased, leading to higher costs. The authors address this issue by investigating model compression techniques. Their purposed method focuses on structured pruning, which entails parameterizing each weight matrix with a low-rank factorization and deleting rank-1 components selectively during training. By doing so, the authors achieve significant compression levels while outperforming unstructured and block-structured pruning techniques in language modeling tasks. Moreover, their approach offers notable speed improvements during both training and inference stages. The paper also highlights the applicability of their method to other aspects of large language models. They demonstrate its effectiveness in pruning adaptive word embeddings, which are crucial for language understanding. Furthermore, they apply their structured pruning approach to the BERT model and evaluate its performance on various downstream fine-tuning classification benchmarks.

1) FILTER PRUNING

Filter pruning is a method used in NLP to minimise DNN computational complexity and memory needs. In NLP tasks, like language translation or sentiment analysis, neural networks often consist of multiple layers with numerous filters capturing different linguistic features. Filter pruning involves accessing the value of each filter depending on criteria like weights or activation's and removing filters that contribute minimally to the network's performance. This process helps in reducing the model size, accelerating computations, and improving efficiency without significantly sacrificing the network's overall performance. By selectively removing redundant or less informative filters, filter pruning enables more streamlined and resource-efficient NLP models. Filter Pruning is depicted in Fig. 3.

Li et al. [35] discuss the problem of shallow neural networks being the preferable option for applications with limited processing and memory resources,in spite of DNN leading-edge capabilities. The research focuses specifically on the one-convolutional-layer CNN, which is frequently used for text categorization and other natural language processing (NLP) applications. In spite of that, it has been noted that CNNs may have trouble handling terms in the dataset that are unrelated to the task at hand, leading to sub-optimal performance. While integrating attention mechanisms into CNNs can alleviate this problem, it also consumes limited resources. To tackle this issue, the authors suggest a unique method to deal with misfitting by pruning words from the dataset that are redundant to the task. The suggested technique assesses each convolutional filter's performance based on its ability to generate features at the pooling layer. Subsequently, words identified by underperforming filters are pruned, eliminating irrelevant information for the task at hand. The experimental findings reveal that the proposed model surpasses the baseline CNN model with significant improvements. Furthermore, the proposed model achieves comparable or even superior performance, while requiring fewer parameters and floating-point operations (FLOPs). These characteristics make it an appealing option for scenarios with limited resources, such as mobile applications.

2) CHANNEL PRUNING

Channel pruning in natural language processing (NLP) refers to a method for reducing the computational complexity and

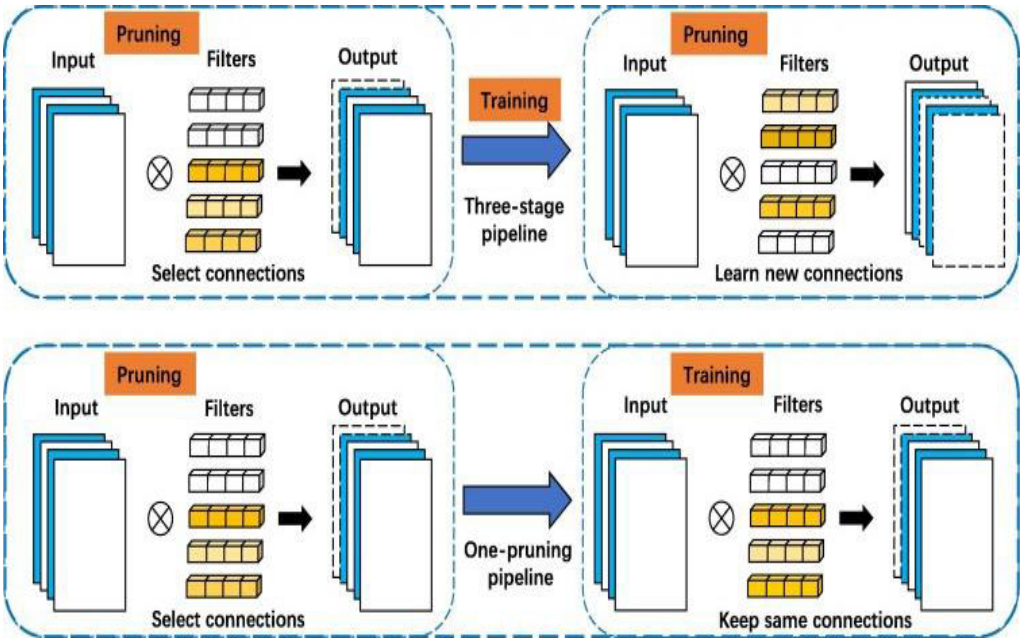


FIGURE 3. Filter pruning removes the insignificant filters, and retraining the network [34].

memory requirements of neural models by eliminating excess or redundant channels. In many tasks of NLP, like text classification or sentiment analysis, neural networks often employ convolutional layers to extract meaningful features from textual data. These convolutional layers consist of multiple channels, each responsible for detecting specific patterns or linguistic features. However, not all channels contribute equally to the network's performance, and some may even be redundant. Channel pruning aims to identify and remove these redundant channels, thereby reducing the model's overall complexity without significantly sacrificing accuracy. Channel pruning extends the idea of filter pruning to the entire channels of convolutional layers. A channel in a convolutional layer refers to the output of a single filter applied to the entire input. Instead of pruning individual filters, channel pruning prunes entire sets of filters (channels) from the network, leading to a more significant reduction in computational cost. In NLP models, channel pruning involves removing entire sets of learned features. The process of channel pruning is shown in Fig.4.

The process typically involves evaluating the importance or contribution of each channel through methods like magnitude-based pruning or sensitivity analysis. Pruned models can generate significant computational reductions, making them more effective for use in large-scale NLP applications or on devices with limited resources.

Yu and Wu [37] introduces Unified Pruning Framework for Vision Transformers (UP-ViTs), a unified pruning framework for vision transformers, to address issues like large model sizes, memory consumption, and computational costs. Existing vision transformer pruning methods involve token sampling, which hampers generalization and is challenging to apply to NLP tasks. UP-ViTs prunes channels in a unified

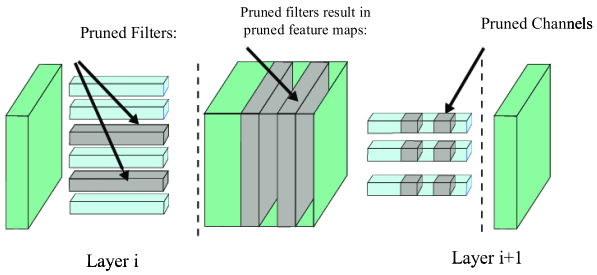


FIGURE 4. Channel pruning amputates the entire channel of filters [36].

manner, covering all transformer components. It evaluates the importance scores of each filter in a pre-trained ViT and removes redundant channels based on compression goals. The approach maintains token representation consistency, ensuring generalization to downstream tasks. UP-ViTs outperform previous ViTs with higher throughput and can be extended to transformers in NLP tasks, showing improvements on language modeling benchmarks. The pruning method involves evaluating performance changes after removing channels in specific components. Experimental results demonstrate the effectiveness of UP-ViTs in achieving high accuracy with compressed models while significantly reducing computational complexity for vision transformers.

Liu et al. [38] introduced TCAMixer, a lightweight Mixer model designed for edge devices to classify texts. It addresses the challenges posed by large-scale model sizes and expensive computing costs in deploying large pre-trained models. The TCAMixer incorporates a novel Triple Concepts Attention Mechanism, which abstracts textual concepts in a human-like manner. Comparing it to counterparts like pNLP-Mixer and HyperMixer, which are projection-based MLP-Mixer and hyper-network models, respectively, the

TCAMixer outperforms them significantly on several public datasets. For instance, the TCAMixer achieves 3% higher accuracy with a smaller model size of 0.177M. Additionally, the TCAMixer performs at 85% to 98.7% of the large pre-trained models' performance but occupies only 1/3000 to 1/2000 of their size on most test datasets. The proposed TCAMixer presents a promising solution for efficient text classification on resource-constrained devices while maintaining competitive accuracy compared to larger models.

3) LAYER PRUNING

Layer pruning involves removing entire layers from a neural network. This approach may be used to minimise the size and computing needs of various NLP models, such as RNNs or transformer-based models [39]. Layer pruning can be particularly effective when the network has multiple stacked layers, allowing for removal of unnecessary layers without significantly affecting performance.

Pruning criteria for layer pruning can be based on factors such as layer importance or the layer's impact on the network's output. For instance, layers with low contribution to the model's performance or minimal impact on the final predictions can be pruned. By removing redundant layers, the network can become more compact and computationally efficient.

Jordao et al. [40] introduce a technique called Discriminative Layer Pruning for Convolutional Neural Networks. This approach involves assessing the discriminative power of each layer in a CNN and subsequently removing the least significant layers. The authors additionally show that combining layer pruning with filter pruning in a cascading manner can lead to enhanced utilization of memory, removes 36.70% of FLOP count, and faster predictions. Layer pruning is one of the approaches used to decrease model complexity and increase model performance.

Fan et al. [41] introduce a layer-wise model pruning technique utilizing mutual information. The main motivation behind this research is the desire to better the capability of DNNs by reducing their computational complexity and memory footprint without sacrificing performance. The authors utilize mutual information as a measure to capture the statistical dependency between layers and the target output. By assessing the mutual information for each layer, they can identify and prune layers with lower mutual information, which indicates their potential redundancy or lesser importance in the network. The proposed layer-wise model pruning technique is conducted in a sequential manner, that starts from the input layer and iteratively progresses towards the output layer. At each iteration, the mutual information between the current layer and the target output is estimated and used as a criterion for pruning. The pruning process involves removing the least informative layers, which helps streamline the network and improve its efficiency. To evaluate the effectiveness of their approach, the authors ran trials on a variety of benchmark datasets, comparing the performance of their pruned models to that of the original complete models. The results of their experiments demonstrated that their layer-wise model pruning technique effectively reduces the computational complexity and memory requirements while maintaining competitive accuracy levels. When compared to the original models, the pruned models achieved considerable reductions in the number of parameters and FLOPs, with no appreciable deterioration in performance. Moreover, the authors explored the impact of different pruning ratios on the performance of the pruned models. They observed that even with aggressive pruning ratios, the pruned models maintained relatively high accuracy levels, indicating the usefulness of the proposed approach in achieving significant compression while preserving performance. The experiments involve WMT14 En-Fr and WMT14 En-DE datasets and include extra-large, large, base, and tiny models. The models are trained on 16 V100 graphics processing units (GPUs) with 32G memories, using Adam optimizer with specific parameters. Beam search is employed for evaluation, and BLEU scores, FLOPs, and practical speedup are reported for single models without ensembling. The results demonstrate that mutual information-based layer-wise model pruning outperforms these strategies in terms of attaining larger pruning ratios with equivalent or superior accuracy.

Peer et al. [42] introduce a method called Greedy-layer pruning for shrinking transformer models in NLP tasks. The authors aim to achieve a customizable tradeoff between performance and speed without the need for additional pre-training phases, unlike knowledge distillation methods. Greedy-layer pruning operates through iterative layer pruning using a greedy approach. The algorithm dynamically adjusts the model size for specific downstream tasks, allowing precise customization without sacrificing performance. The method focuses on reducing computational costs while maintaining high performance. Experimental results showcase the effectiveness of Greedy-layer pruning. For BERT and RoBERTa models, the approach achieves 95.3% and 95.4% performance retention, respectively, while pruning 50% of the layers. This demonstrates the ability of Greedy-layer pruning to significantly reduce the computational requirements of transformer models while preserving a considerable portion of their original performance. The paper highlights the advantages of Greedy-layer pruning over distillation methods. Unlike knowledge distillation, which typically involves additional pre-training phases, Greedy-layer pruning allows for dynamic adjustment of model size and performance retention during the fine-tuning process itself. This flexibility makes the method suitable for various NLP tasks where computational efficiency is a crucial factor.

4) STRUCTURED WEIGHT PRUNING

Structured Weight pruning involves removing unimportant weights from a neural network while preserving its structure. This technique is widely used for model compression and reducing computational requirements. In the field of NLP, structured weight pruning has been applied to various tasks, such as language modeling and sentiment analysis.

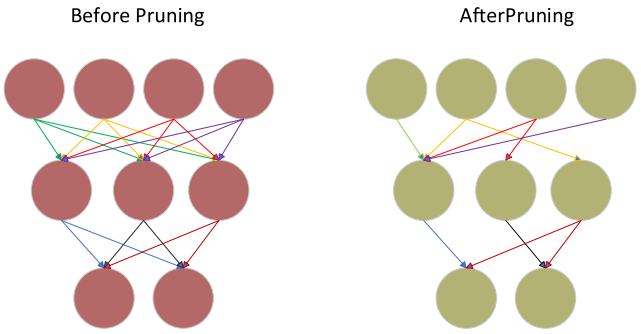


FIGURE 5. Structured weight Pruning removes the weight connections [31].

Weight pruning typically involves setting a threshold value and then removing weights that fall below this threshold. The threshold can be determined based on various criteria, such as the weight magnitude, importance scores, or sensitivity analysis.Once the threshold is set, any weights that fall below it are pruned, and the related connections are deleted from the network.

Pruning can be performed in an iterative manner, with multiple pruning rounds. During each round, a certain percentage or fixed number of the lowest magnitude or least important weights are pruned. After pruning, the network may undergo a retraining phase to fine-tune the remaining weights and mitigate any performance degradation caused by the pruning process.

Cho et al. [43] proposes Parameter-free Differentiable Pruning (PDP), an efficient and effective train-time pruning scheme for DNNs. PDP generates soft pruning masks for weights during training without requiring additional parameters, making it universal for various vision and natural language tasks. It achieves state-of-the-art results in model size, accuracy, and training cost. PDP outperforms existing algorithms in random, structured, and channel pruning tasks for vision and language models. For instance, it achieves higher accuracy in ImageNet1k for MobileNet-v1 at 86.6% sparsity and in Multi-Genre Natural Language Inference for BERT at 90% sparsity. Moreover, PDP can be applied to structured pruning like N:M pruning and channel pruning, outperforming state-of-the-art results in tasks such as 1:4 structured pruning of ResNet18 and channel pruning of ResNet50. Overall, PDP provides a powerful and efficient solution for model compression and optimization across a wide range of DNN architectures and tasks.

The technique is shown in fig. 5. Weight pruning offers several advantages. It minimises the memory footprint of the model, making it more suited for deployment on resource-constrained devices. Pruned models also require fewer computations, resulting in faster inference times. Moreover, weight pruning can enhance model interpretability by exposing the most critical connections and features.

5) BLOCK PRUNING

Block pruning is a technique that expands structured methods by taking into account blocks of any size and incorporates this structure into the movement pruning paradigm for fine-tuning. Instead of removing individual weights or neurons, the method entails identifying and removing whole blocks of the model, such as attention heads. This strategy keeps the model's performance while enabling more effective pruning. In Block pruning a large pre-trained transformer model is trained on the given task before putting this strategy into practice. Then, entire blocks of the model that are believed to be unnecessary for the task at hand are identified and eliminated using movement pruning algorithm.

Lagunas et al. [44] also introduce a brand-new metric known as ''block importance'' to assess how much each block contributes to the overall performance of the model. Then after pruning, the authors fine-tuned the pruned model on the same task, and compare its performance to the original, un-pruned model. Experiments explore classification and generation tasks, providing, among other things, a pruned model that is 2.4x quicker, 74% smaller BERT on SQuAD v1, with a 1% drop on F1, competitive in both speed and size with distilled models.

Pheng et al. [45] explores the use of column-balanced block-wise pruning to accelerate Transformer models on field-programmable gate arrays (FPGAs). While weight pruning has been studied for reducing model size on GPUs, its application on FPGAs has not been investigated. The authors create an FPGA acceleration engine for the Transformer's balanced block-wise matrix multiplication. The experimental findings show that the FPGA implementation achieves a latency of 10.35 ms for Transformer inference with a batch size of 32, delivering a 10.96x speedup over the central processing unit (CPU) platform and a 2.08x speedup over the GPU platform.

Li et al. [46] uses a hardware-friendly block-structured pruning method for efficient large-scale language representations based on Transformers. The need to increase the efficiency of DNNs by decreasing their computational complexity and memory footprint without losing performance is the driving force behind this study.

A block-based pruning framework (BLCR) meant to speed DNN execution on resource-limited computing platforms, notably mobile devices, is introduced by Ma et al. [47]. To achieve high sparsity while exploiting on-device parallelism, BLCR combines a flexible structured pruning approach with a reweighted regularisation method. It includes both CNNs and RNNs, with computation-intensive layers like convolutional and fully linked layers supported. Furthermore, the system incorporates compiler-based code optimisation to enable real-time DNN inference on mobile devices while maintaining accuracy.

6) MOVEMENT PRUNING

Movement pruning is a first-order weight pruning method which is mostly used to fine-tune the pertrained models. Movement pruning iteratively prune the weights by using least absolute movement which is the difference between

weights values in current and previous iterations. Movement pruning produces a smoother distribution of residual weights that spans the whole interval except for values near to zero. Sanh et al. [48] exploit movement pruning to introduce adaptive sparsity in neural networks which is achieved in two steps. 1: in training of network separate mask is used for each layer for determining the importance of each connection, initially the values of mask are set to all ones then the values of mask are updated based on gradients of loss function to prune the unimportant connections. 2: to recover the performance of the pruned network fine–tuning is performed. For this perturbations to the pruned connections during fine-tuning are introduced. These perturbations help the network explore alternative paths and improve its ability to recover from the pruned state. The authors undertake tests on several deep learning architectures, including feed-forward networks, convolutional neural networks, and recurrent neural networks, to evaluate their technique. The results reveal that Movement Pruning achieves substantial sparsity while preserving or even boosting network performance when compared to the original dense networks. The authors also compare their method to current pruning strategies and show that it outperforms them in terms of sparsity-performance trade-off.

The framework proposed by Joniak and Aizawa [49] focuses on identifying a subset of the model that exhibits reduced gender bias. To achieve this, attention heads in the transformer-based models are selectively pruned at either the block or entire head level. By pruning specific attention heads, the authors aim to mitigate the propagation of gender bias within the model's representations and predictions. The effectiveness of the approach is demonstrated through experimental evaluation. The authors evaluate the trimmed models' performance on gender bias-related tasks and benchmarks. The results showcase the efficacy of the framework in reducing gender bias, as the pruned models exhibit improved fairness and reduced biases in their predictions. Furthermore, the authors propose improvements to existing debiasing techniques, highlighting the potential of movement pruning as a valuable tool in mitigating bias. By selectively removing attention heads associated with gender bias, the framework provides a targeted approach to address and rectify biases in pre-trained language models. One crucial finding highlighted in the paper is the trade-off between bias mitigation and model performance. The authors reveal that higher model performance often correlates with increased gender bias. This observation sheds light on the inherent challenges in achieving both fairness and high accuracy simultaneously. It encourages academics and practitioners to carefully explore and balance these variables while designing and using transformer-based language models. The importance of this study rests in its investigation of gender bias in pre-trained language models and the development of a methodology that effectively tackles this prejudice through movement trimming. By selectively pruning attention heads, the authors demonstrate the potential to mitigate gender bias and improve fairness in language models. The findings contribute to ongoing discussions on bias in natural language processing and highlight the importance of developing debiasing techniques that go beyond post-hoc interventions.

B. UNSTRUCTURED PRUNING

Unstructured pruning is a NLP approach used to minimise the size and complexity of neural networks used for language modelling, text classification, machine translation, and other NLP applications. This pruning technique involves selectively removing individual weights or parameters from the network, typically based on their magnitudes or importance scores. By eliminating redundant or less significant parameters, unstructured pruning aims to achieve model compression, improved efficiency, and reduced computational requirements without significantly sacrificing performance. The process of unstructured pruning involves two main steps: identification and removal. In the identification step, each weight or parameter in the network is evaluated based on a predefined criterion, such as magnitude or sensitivity analysis. Magnitude-based pruning, for example, ranks the weights according to their absolute values, allowing the removal of those with the smallest magnitudes. Depending on the pruning approach, this rating can be done globally or layer-by-layer. Once the weights or parameters are ranked, the removal step involves discarding a certain percentage of the least important ones. This removal can be performed by setting the corresponding weights to zero or by completely eliminating the associated connections. In some cases, a threshold is applied to determine the cutoff point for pruning, allowing finer control over the sparsity level of the pruned model.

One of the primary advantages of unstructured pruning is its flexibility in targeting specific weights or parameters, making it suitable for reducing the model size while preserving important network structures. However, this flexibility comes at the cost of irregular sparsity patterns, as individual weights are pruned independently. Consequently, unstructured pruning can lead to inefficient memory access and inefficient deployment on hardware accelerators optimized for dense matrix operations. To address these issues, additional techniques such as structured pruning and weight sharing can be applied in conjunction with unstructured pruning [7], [30], [50], [51], [52], [53].

In the context of NLP, unstructured pruning can be applied to various components of neural network models, including word embeddings, recurrent connections, and fully connected layers. The main objective is to identify and eliminate parameters that are deemed less important or redundant, thereby reducing the model's memory requirements and computational demands while striving to maintain or minimize the impact on performance.

Unstructured pruning in NLP seeks to establish a compromise between model reduction and acceptable performance. It allows for more aggressive parameter removal compared to structured pruning methods but may also pose challenges

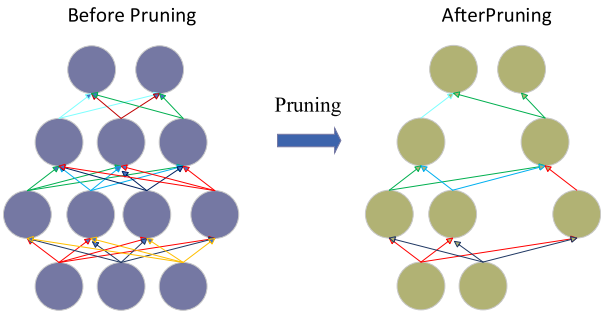


FIGURE 6. Unstructured Pruning reduces the network by irregular removal of neuron units [54].

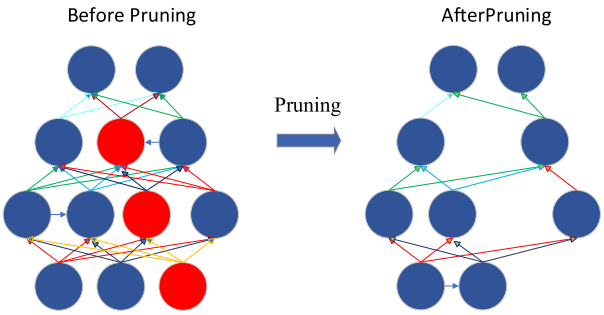


FIGURE 7. Neuron Pruning removes the neurons form the network [53].

such as potential accuracy loss or the need for careful fine-tuning.

Unstructured Pruning is depicted in Fig 6. Li et al. [7] present an unstructured weight pruning approach for CNNs. The authors offer a metric for measuring the relevance of filters and prune filters systematically depending on their importance. The results show that unstructured pruning can minimise the number of parameters while preserving performance.

Narang et al. [50] investigate the use of unstructured pruning in the context of recurrent neural networks (RNNs). The authors want to know how weight pruning affects RNNs and suggest a structured regularisation strategy to encourage sparsity in these networks. By exploring the potential of unstructured pruning, they aim to reduce the number of parameters in RNNs while minimizing the impact on performance. The study begins by highlighting the significance of sparsity in deep learning models, especially in scenarios where computational and memory resources are limited. RNNs, being widely used in various NLP tasks such as language modeling and sequence generation, can benefit from pruning techniques to alleviate the computational burden and improve efficiency. However, the unique recurrent structure of RNNs poses challenges in applying traditional pruning methods effectively. To address these challenges, Narang et al. propose a structured regularization technique that encourages sparsity in RNNs. This regularization technique introduces a penalty term based on the L1 norm of the weights, encouraging a large number of them to be set to zero. By incorporating this regularization term into the training objective, the authors aim to promote sparsity during the learning process. The experimental evaluation conducted by the authors showcases the effectiveness of unstructured pruning in reducing the number of parameters in RNNs. By applying the proposed structured regularization technique, they successfully induce sparsity in the RNNs without a significant loss in performance. The experiments demonstrate that the pruned RNNs achieve comparable or even improved performance on tasks such as language modeling and speech recognition. Furthermore, the authors examine how different sparsity levels affect the performance of pruned RNNs. They discover that even at high sparsity levels, trimmed models may outperform their dense counterparts. This discovery demonstrates the potential of unstructured pruning to dramatically lower RNN memory and compute needs while maintaining RNN efficacy in NLP workloads.

Chen et al. [51] addresses the challenges posed by unstructured sparsity after pruning in the implementation of deep learning models. The authors suggest a technique of compression that combines unstructured pruning with a unique weight permutation mechanism. The sparse weight matrix is further compressed into a tiny and dense shape by permuting it, maximising hardware resource utilisation. When compared to state-of-the-art approaches, the suggested method yields a 10.28x improvement in matrix compression rate. As a result, throughput and energy efficiency are increased by 2.12 and 1.57 times, respectively. The approach is named "Tight Compression" and offers a promising solution for efficiently compressing CNN models.

1) NEURON PRUNING

Neuron pruning aims to remove unnecessary neurons or units from neural network architectures. In the context of NLP, this technique involves eliminating unimportant neurons in language models. The process of neuron pruning typically involves identifying and removing neurons based on their importance or contribution to the overall network. Several criteria can be used to determine the importance of a neuron, such as its activation level, its impact on the network's output, or its connection weights. Neurons that are deemed less important can be pruned, eliminating their connections and reducing the overall complexity of the network. The process of Neuron pruning is described in Fig 7.

The pruning process can be performed iteratively, where pruning is done in multiple rounds. During each round, a certain percentage or fixed number of neurons with the lowest importance scores are pruned. After pruning, the network may undergo a retraining or fine-tuning phase to compensate for the removed neurons and regain its performance.Neuron pruning can offer several benefits. It can help reduce the memory footprint of the network, making it more efficient for deployment on resource-constrained devices. Additionally, pruning can lead to faster inference times by reducing the number of computations required. Pruned networks can also be more interpretable as the removal of redundant neurons can reveal the important connections and patterns in the model.

Hu et al. [52] propose a method called ''Network Trimming'' to design efficient DNNs. The approach involves iteratively pruning unimportant neurons based on their outputs, specifically targeting zero activation neurons that are deemed redundant. After pruning, the network is retrained using the pre-pruning weights as initialization. This pruning-retraining cycle is repeated to further reduce zero activations. Experiment findings on LeNet and VGG-16 show that the suggested technique achieves large compression ratios without losing accuracy and, in some cases, improves accuracy over the original network. Network Trimming is a data-driven technique to building efficient deep architectures.

Yu et al. [53] propose a method called Neuron Importance Score Propagation (NISP) for pruning deep CNNs. NISP seeks to simultaneously prune neurons based on minimising the reconstruction error of key responses in the final response layer (FRL), which is the second-to-last layer before classification, as opposed to previous approaches that focus on individual layers or successive layers. The method measures neuron relevance using feature ranking techniques, formulates network pruning as a binary integer optimisation problem, and develops a closed-form solution for pruning neurons in earlier layers. When tested on numerous datasets and CNN models, NISP achieves considerable acceleration and compression with low accuracy loss.

In [55] the authors explore the application of regularization in DNN pruning. They introduce a new scenario where the regularization gradually increases in strength, addressing two key problems in pruning: pruning schedule and weight importance scoring. The authors propose an L2 regularization variant with rising penalty factors, which leads to significant accuracy gains compared to traditional one-shot pruning methods. The growing penalty scheme also enables the utilization of Hessian information for more accurate pruning without the need for precise Hessian values.

2) MAGNITUDE BASED PRUNING

Magnitude-based pruning is the practise of pruning weights based on their magnitudes, with lower magnitudes regarded less relevant and trimmed. This strategy is based on the idea that tiny weights contribute less to the overall performance of the model. The model's size and computational complexity can be lowered by deleting these less significant weights. Magnitude-based pruning has been used successfully in NLP applications including machine translation and sentiment analysis.Magnitude-based pruning selectively removes weights with the smallest magnitudes (i.e., absolute values). See et al. [56] compress RNN models for NMT using magnitude-based pruning combined with retraining. After pruning, they further refine the pruned network through fine-tuning, aiming to achieve improved performance.

Li et al. [57] propose Optimization-based Layer-wise Magnitude-based Pruning (OLMP) as a unique strategy for compressing DNNs in their paper. OLMP aims to achieve effective model compression while controlling accuracy loss by automatically modifying layer-specific criteria. The method utilizes strong derivative-free optimization methods to find optimal solutions for threshold tuning as a constrained optimization problem. OLMP operates iteratively, pruning the network to reduce its size while meeting accuracy requirements and then fine-tuning the pruned model to recover any potential accuracy loss. Experimental evaluations on LeNet-style and AlexNet-style networks demonstrate that OLMP outperforms state-of-the-art approaches in terms of compression ratios while maintaining competitive accuracy levels. OLMP offers automated threshold tuning, eliminating the need for manual setting and leveraging powerful optimization algorithms for efficient pruning configurations. The layer-wise approach of OLMP contributes to more effective DNN compression and improved overall efficiency, making it a promising method for addressing the demand for efficient DNNs.

Lee et al. [58] introduce Layer-Adaptive Magnitude-based Pruning (LAMP) to address the challenge of selecting layer-wise sparsity in neural network pruning. They emphasize the importance of optimal sparsity selection for reducing model complexity and enhancing efficiency. Existing approaches often use manual heuristics or computationally intensive hyperparameter search. LAMP overcomes these issues by proposing a data-driven solution. It introduces an importance score that combines weight magnitudes and model-level L2 distortion, aiming for an optimal balance between sparsity and network performance. LAMP eliminates the need for manual tuning and heavy computation, utilizing the importance score to guide pruning automatically, reducing complexity in the process.

Hong et al. [59] provide Multi-objective Magnitude-based Latency-Aware Pruning (MMLAP), a new layer-wise magnitude-based pruning strategy for compressing DNNs that takes latency into account. MMLAP tries to minimise DNN inference latency by reducing network connections. Unlike previous methods, MMLAP directly captures latency and employs a novel multi-objective evolutionary algorithm to optimise both the accuracy and latency efficiency of the compressed networks during the hyper-parameter tuning of layer-wise magnitude-based pruning (LMP). The experimental findings show that MMLAP is competitive with known LMP approaches and that multi-objective optimisation is successful in generating Pareto-optimal compressed networks in terms of accuracy and latency. Li et al. [60] introduce a stage-wise magnitude-based pruning (SW-MBP) technique tailored for RNNs in NLP tasks. The authors recognize the unique characteristics of RNNs, which consist of recurrent connections that play a crucial role in capturing sequential dependencies in NLP tasks. In order to account for this recurrent structure, SW-MBP groups connections based on their intersection with recurrent neurons. This grouping allows for a more accurate and targeted pruning approach, as it considers the impact of pruning on both feedforward and recurrent connections. The pruning process in SW-MBP is conducted in stages. At each stage, connections within a group are pruned based on their magnitude. Magnitude-based

pruning identifies connections with low magnitudes that contribute less to the overall network performance. By removing these connections, the network's complexity is significantly reduced without sacrificing performance. To achieve the best pruning results, an optimization-based pruning approach is employed within each group. This approach enhances the optimization of the connection selection process. The optimization process ensures that the most redundant and least informative connections are pruned, leading to a more efficient and streamlined network. Experimental results shows the supremacy of SW-MBP over conventional RNN pruning methods. The proposed technique achieves impressive connection pruning rates of up to 96.84%, indicating a significant reduction in the number of connections while maintaining high precision indicators on testing datasets. This demonstrates how well SW-MBP works in lowering the computational difficulty and memory requirements of RNNs in NLP applications.

3) ITERATIVE PRUNING

Iterative pruning is a technique used in natural language processing (NLP) to minimize the size and computing demands of neural network models used for NLP applications. The process of iterative pruning involves a gradual selection and removal of less important components, such as channels, weights, or neurons, from the network. The objective is to iteratively prune the model while minimizing any negative impact on its performance. Iterative pruning is often used in natural language processing (NLP) models that use architectures such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs) for tasks such as text classification, sentiment analysis, machine translation, or natural language comprehension.

The iterative pruning procedure typically encompasses the following steps:

1) Attempting: In this stage, a subset of components, such as channels or neurons, is pruned from the network. The effect of this pruning on the model's performance is evaluated to estimate the resulting accuracy decrease.
2) Selecting: A selection technique is used to determine the layers, channels, or components that have the least influence on the overall network performance. These less important components are prioritized for further trimming.
3) Pruning: The selected components are then removed from the network, effectively reducing its size and computational complexity. The pruning criteria may involve factors like importance scores, activation levels, or weight magnitudes.

Following the pruning stage, the network is typically fine-tuned or retrained to restore its performance. This fine-tuning process enables the network to adapt to the pruned architecture and regain its accuracy.

Mallaya and Lazebnik [61] provide PackNet, a method for incorporating several tasks into a single DNN without encountering catastrophic forgetting. The method is inspired by network pruning techniques, which find and remove redundancies in big networks to provide a place for new tasks to be learned. The authors systematically integrate several jobs into a single network via iterative pruning and re-training, guaranteeing minimum performance degradation and storage overhead. PackNet optimizes for each individual job, as opposed to earlier approaches that rely on proxy losses to retain accuracy on older tasks. Extensive testing on various network designs and large-scale datasets shows that PackNet is more resilient to catastrophic forgetting than previous techniques. Notably, the scientists successfully combined three fine-grained classification tasks into a single VGG-16 network trained on ImageNet, attaining accuracies equivalent to individually trained networks for each job.

Rajaraman et al. [62] present a technique for identifying COVID-19 pulmonary symptoms using chest X-rays using iteratively trimmed deep learning model ensembles. The investigation focuses on the Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2) virus, also known as the novel Coronavirus (2019-nCoV) that causes COVID-19. On publically accessible chest X-ray datasets, the researchers train and test a proprietary convolutional neural network and pre-trained models to learn modality-specific feature representations. They use and fine-tune their knowledge to improve performance and generalization when identifying chest X-rays as normal, indicating bacterial pneumonia, or exhibiting COVID-19-related abnormalities. The best-performing models undergo repeated pruning, a technique of lowering network complexity by deleting less relevant connections or parameters, to improve complexity and memory efficiency. To improve classification performance, the predictions of these trimmed models are pooled using various ensemble procedures. Experiment results show that the weighted average of the best-performing pruned models enhances performance substantially, with an accuracy of 99.01% and an area under the curve of 0.9972 in identifying COVID-19 discoveries on chest X-rays.

4) LOTTERY TICKET HYPOTHESIS

A randomly initialized, dense neural network comprises a subnetwork that is initialized in such a way that, when trained in isolation, it can equal the original network's test accuracy after training for no more than the same number of iterations. The Lottery Ticket Hypothesis in NLP proposes that inside a large neural network, a sparse subnetwork known as the ''winning ticket'' can achieve equivalent or higher performance than the original dense network. This hypothesis is based on the idea that during the network's initialization, only a small fraction of the connections and weights are truly necessary for optimal performance. In the context of NLP, the lottery ticket hypothesis has been applied to transformer-based language models like BERT and GPT. The process begins with the random initialization of a large neural network. The network is then trained on a specific NLP task using standard training procedures. After training, a pruning algorithm is used to identify and

remove unimportant connections or weights based on certain criteria, such as weight magnitude. The pruned network is reinitialized, and a fine-tuning process is applied to recover its performance.

Chen et al. [63] investigate the presence of trainable and transferrable subnetworks within large pre-trained models like BERT in the context of NLP. It highlights the emergence of the lottery ticket hypothesis, which suggests that smaller subnetworks within these models can achieve full accuracy when trained in isolation and can be transferred to other tasks. The authors want to integrate these two findings by investigating if such trainable and transferable subnetworks exist in pre-trained BERT models. They do a number of tests on a variety of downstream tasks to find matched subnetworks with sparsities ranging from 40% to 90%. The study demonstrates that subnetworks discovered on the masked language modeling task, which is also used to pre-train BERT, have universal transferability. This means that they can be effectively applied to different downstream tasks. However, subnetworks found on other tasks show limited transferability, or in some cases, no transferability at all. This distinction suggests that the nature of the task plays a crucial role in determining the transferability of the subnetworks within the BERT architecture.

Behnke and Heafield [64] focus on the attention mechanism in transformer architectures, crucial for natural language processing tasks. Previous research reveals that certain attention heads lack confidence in their judgments and can be trimmed, but directly removing them leads to model quality reduction. To address this, the research employs the lottery ticket hypothesis, removing attention heads early in training. This hypothesis suggests trainable subnetworks within large networks can achieve similar performance. By leveraging this idea, unnecessary attention heads in transformers are identified and pruned. Experiments in machine translation, like Turkish to English, show promise. The pruned model demonstrates a slight -0.1 change in the BLEU score, a key translation quality metric, indicating effective trimming. Ding et al. [65] examine the use of the lottery ticket concept to build lightweight voice recognition models. Rather than creating these models from scratch, practitioners frequently opt to condense big pre-trained speech models. According to the lottery ticket theory, very sparse subnetworks inside bigger models may be trained independently without losing performance. The researchers undertook comprehensive tests on several voice recognition models, including CNN-LSTM, RNN-Transducer, and Transformer models. They confirmed the existence of very sparse winning tickets, which are subnetworks that function similarly to the full models. These winning tickets weighed less than 20% of the complete models, with the lightest ticket retaining just 4.4% of the weights.

V. COMPARISON OF DIFFERENT PRUNING TECHNIQUES

The following discussion provides a comparison among some recent trends in pruning in the field of natural language processing. Table 1 summarizes this comparison including the advantages and disadvantages of the pruning techniques.

Yoon et. al. [66] proposed the TextPruner, an open-source toolkit for pruning Natural Language Processing related pre-trained models. It offers a structural pruning that is utilized for vocabulary and transform pruning. Vocabulary pruning enhances efficiency and mitigates the network size by removing the tokens in pre-trained models that are not related to downstream tasks. In transform pruning, it removes the attention heads and neurons in feed-forward units based on the already computed importance score. On the Cross-lingual Natural Language Inference (XNLI) corpus using the XLM-RoBERTa model, the scheme achieves up to 41.3% model size. The proposed technique achieves up to 1.90x speed up which is almost a 50% increase in processing speed. However, the technique is self-supervised lacks flexibility in fine-grain pruning, and loses accuracy after pruning.

Peer et. al. [42] proposed the Greedy-layer pruning method which removes the layers of the network based on some predefined fine-grain performance criterion. The technique is evaluated on the GLUE benchmark using BERT, and RoBERTa transformer models. The method achieves mitigation from 67 M to 82 M parameters as compared to 108M parameters with inference time from 70 to 73 seconds on CPU. This speed-up is 1.7 seconds for GPUs.

Li et. al. [67] proposed a filter-pruning mechanism using Convolution Neural Networks (CNNs). The approach is applied to a text dataset of cancer pathology reports by cancer type for text classification. Like other pruning techniques, the method focuses on the importance of each filter for pruning by associating a utility score with a filter. After activation of a filter for a training set is passed through the max pooling operation, variation in resultant values across all training sets is used to calculate the utility score of each filter. The method achieved a reduction of a third less network weight with 7% enchantment in the micro-averaged F1-score and 22% in the macro-averaged F1-score.

Xu and Hu [46] proposed block pruning for BERT, RoBERTa, and DistilBERT models. The method rearranged the weight matrix into a row or column blocks format and performed the pruning of blocks based on the $l_2 norm$ value against a predefined value. This structure pruning supports parallel processing. The technique achieves 5x weight pruning with negligible accuracy degradation on the GLUE benchmark. Additional 1.79x compression is achieved using knowledge distillation for little accuracy loss.

Wang [68] explore the fairness of the Natural Language Processing (NLP) model due to various compression methods. The considered compression methods are pruning and knowledge distillation on the GPT2 model. The paper mainly discussed the contradictory hypotheses of Memorization, and Winer takes it all. Memorization deliberates the loss of unwanted and biased traits of corpus due to compression methods. Compression methods mitigate the standard model into a smaller model introducing bias itself in the second hypothesis. The results demonstrate the empirical evidence

TABLE 1. Comparison of different pruning techniques.

Technique	Definition	Advantages	Disadvantages	Metrics and Benchmarks	Performance values
Structured Pruning [66]	Removal of entire structured components (filters, channels, or layers) to reduce model size and computational complexity.	Preserves network structure, and efficient resource utilization.	Limited flexibility in fine-grained pruning, potential loss in accuracy.	Cross-lingual Natural Language Inference (XNLI) corpus as the text classification dataset.	After pruning, the model size is decreased by around 60% while maintaining acceptable performance.
Layer Pruning [42]	Pruning entire layers of a neural network to reduce the model size and computational costs.	Efficient reduction in model size, simplified network architecture.	May lead to information loss and limited flexibility in preserving fine-grained details.	BERT and RoBERTa	The difference in performance value is 0.5 for pruning BERT and 1.0 percentage points for pruning RoBERTa.
Filter Pruning [67]	Pruning individual filters within a layer to reduce the model size and improve efficiency.	Reduces parameter count, computational cost, and memory footprint.	May result in information loss, a potential impact on performance.	F1-score and Model size reduction.	The algorithm resulted in a nearly 0.07 increase in the micro-averaged F1-score and reduced the network weights by 33.33%.
Structured Weight Pruning [46]	Removing specific weights in a structured manner to reduce the model size and computational complexity.	Preserves network structure, and efficient resource utilization.	Limited flexibility, potential performance impact.	General Language Understanding Evaluation (GLUE) benchmark tasks.	Achieves up to 5.0× with zero or minor accuracy degradation.
Block Pruning [68]	Pruning blocks of connected weights or neurons to reduce the model size and computational requirements.	Reduces model size and computational cost and preserves local connectivity.	May result in information loss and potential performance degradation.	Perplexity	As a result of the pruning, computation speed improves from 1.2x to 1.5x.
Movement Pruning [49]	Pruning filters or channels based on their movement statistics to reduce the model size and improve efficiency.	Efficient reduction in model size, improved computational efficiency.	Limited flexibility, potential impact on performance.	GLUE, bias performance.	The better the model performs, the more bias it contains.
Unstructured Pruning [69]	Removing individual weights or connections without adhering to any specific pattern or structure.	Fine-grained reduction in model size, improved computational efficiency.	Loss of original network structure, potential for irregular connectivity.	Speed	3.4x faster than the same dense calculation at 90% sparsity and 5.4x faster at 95% sparsity.
Magnitude-based Pruning [60]	Pruning weights based on their magnitudes, removing those with lower magnitudes.	Simple to implement, reduced model size and computational cost.	Potential loss in accuracy, and sensitivity to weight initialization.	Precision.	Up to 96.84% of connections are trimmed with little or no loss of precision indicators.

about knowledge distillation being less biased and toxic while model compression may be viewed as regularization.

Yang et al. [69] proposed the code generator (SparseRT) for hardware implementation of unstructured pruning e.g., GPUs. Traditionally, structured pruning introduces more loss of accuracy as par unstructured pruning. However, their efficient hardware implementation remains a question. The proposed technique attempted to take advantage of unstructured sparsity to increase the efficiency of sparse linear algebra operations for transformer using the WMT English-to-German 2014 dataset BLEU score. The technique achieved 95% sparsity with 6.6 points accuracy loss.

Joniak and Aizawa [49] proposed the debiasing of the Language model using Movement Pruning for gender. The pruning of Attention heads and entire square blocks of the BERT model is utilized to reduce the bias by finding the subset of the existing Language model. Sentence Encoder Association Test (SEAT) and Stereotype Score metric are used for measuring bias. The finding determined the performance and bias trade-off. A model having high performance also has a high gender bias.

Li et. al. [60] presented stage-wise pruning of recurrent neural networks (RNN) for NLP. Based on the magnitude pruning method, the feed-forward layers and RNN layers are removed to reduce the model size. The method enables the network pruning of 96.84% connections without any significant accuracy loss.

VI. CONCLUSION

Pruning techniques provide valuable solutions for improving efficiency and performance in NLP tasks. The choice of pruning technique depends on specific requirements and trade-offs. Structured pruning techniques strike a balance between efficiency and preservation of network structure, while unstructured pruning techniques enable fine-grained reduction but may require additional strategies for performance preservation. Magnitude-based pruning offers a simple approach with efficient results. By effectively applying pruning techniques, NLP models can achieve significant reduction in size and computational complexity without compromising performance, thus paving the way for more efficient NLP applications in resource-constrained environments.

We have summarized some of the recent works on pruning of Natural Language Processing–based deep learning networks. This section provides more details about the possible comparison of these pruning methods. In order to provide the best pruning method is not simple. There is no

clear criterion to describe the best approach. However, their applicability is application-dependent.

- Structural ways of pruning like filter, channel, and weight pruning are more convenient for applications that require a compact version of the pre-trained model.
- The applications that require stable accuracy and reduced model size, structural pruning provides reasonable mitigation in network size with minimal accuracy degradation.
- Structural pruning is more hardware-friendly than other pruning techniques. Organized network sparsity is easy to exploit in memory and processing implementation.
- Unstructured pruning is reasonable where applications require acceleration and no compromise on performance.

Although all the above-mentioned techniques are orthogonal to each other, however, various pruning methods can be combined to maximize the punning advantage with minimum accuracy loss. In the case of Convolution Neural Networks (CNNs) that combine the two dimension convolution for feature extraction and fully connected layer for classification, filter or layer pruning and weight pruning can be applied.

VII. FUTURE WORK

This section summarizes the techniques' challenges and possible directions for future work.

A. CONFIGURATION OF EXISTING MODELS

Most NLP-based tasks are implemented using well-defined RNN, BERT, and LSTM models that imply these networks have a fixed configuration like their architecture and hyper-parameters. However, to handle more complex tasks, plausible configurations for compressed models need to be explored.

B. HYBRID PRUNING TECHNIQUES

Hybrid pruning techniques expand beyond a single pruning strategy. They use several pruning techniques and may merge them with additional compression approaches such as quantization or knowledge distillation. This multi-pronged approach intends to achieve greater levels of model compression (lowering size and computing cost) while limiting performance loss in NLP tasks.

Hybrid pruning can drastically reduce model size and computational cost, making NLP models more suitable for use on resource-constrained devices like smartphones. Hybrid pruning can improve NLP task processing speed by lowering model size and perhaps performing lower precision computations (quantization). Combining various pruning strategies may raise the risk of performance degradation. Careful examination and fine-tuning are essential to achieve the best combination of compression and accuracy. Following are the examples of hybrid pruning techniques employed for NLP:

1) **Structured Pruning and Magnitude-based Pruning**
   This approach might first remove whole filters or channels based on their low significance (structured pruning). The remaining network might then be pruned to remove individual weights with very low magnitudes. This combination takes advantage of the benefits of both methods: structured pruning for coarse-grained reduction [70] and magnitude-based pruning for finer-grained optimization [71]. Guo and Li [72] introduces a hybrid pruning technique that combines coarse- and fine-grained strategies to balance accuracy and computational efficiency in neural networks. Initially, coarse-grained pruning recognizes channels for removal while maintaining an acceptable accuracy decrease, followed by fine-grained pruning, which deletes weights below predicted thresholds, lowering network size and computational load. This hybrid technique outperforms single pruning strategies in models like as AlexNet and ResNet, reducing FLOPs by 60% and parameter count by almost 80% on the CIFAR-10 dataset.
2) **Filter Pruning and Quantization**
   Filter pruning removes irrelevant filters, but quantization decreases the precision of remaining weights (e.g., from 32 to 8 bits) [73]. Because of the lower precision computations, this hybrid strategy has the potential to reduce model size significantly and speed up inference times. Liu et al. [74] introduced a technique that uses dynamic and sparse graph (DSG) structures in deep neural networks (DNNs) to achieve compressive memory and faster execution in both the training and inference stages. Unlike previous studies, which primarily optimize for inference, this method addresses the training challenge by selectively activating a small number of neurons with high selectivity using dimension-reduction search (DRS) and ensuring batch normalization (BN) compatibility via double-mask selection (DMS). The experimental findings show considerable memory savings (1.7-4.5x) and operation reductions (2.3-4.4x) with negligible accuracy loss across several benchmarks.
3) **Knowledge Distillation and Unstructured Pruning**
   Knowledge distillation is the process of moving knowledge from a large, pre-trained teacher model to a smaller student model. Following distillation, unstructured pruning might further compress the student model without dramatically reducing performance. The teacher model serves as a safety net for the student, allowing for more severe pruning. Because edge devices lack computing capacity for DNNs, Kim et al. [75] introduced a unique compression approach known as PQK. PQK uses pruning, quantization, and knowledge distillation (KD) to produce efficient models. Unlike standard techniques, PQK constructs a teacher network using pruned weights in order to develop a better student network without prior training. This method, which includes iterative pruning and quantization, followed by teacher-student training within the same network, successfully tackles resource

restrictions, as proven in keyword spotting (KWS) and image recognition tasks.

Exploration of various pruning strategies for NLP-based tasks according to applications and the study of their merging effect with other compression techniques on the amount of compression and performance of deep learning models (e.g., combining the quantization, pruning, or knowledge distillation) must be conducted. Investigating the synergistic impact of these strategies might improve the efficiency and performance of NLP models even more.

C. TRANSFERABILITY OF PRUNED MODELS

There are various pre-trained models available in the literature that have been fine-tuned for NLP related tasks. Some of them are discussed below:

1) **BERT for Named Entity Recognition (NER)** Bidirectional Encoder Representations from Transformers (BERT) models, which were previously trained for tasks like as language modeling and sentence classification, have been pruned and fine-tuned for NER tasks [76]. They have obtained competitive performance in identifying named items such as person names, organization names, and places by removing specific layers or parameters and fine-tuning the model using NER datasets.
2) **GPT for Text Summarization** Generative Pre-trained Transformers (GPT) models, which are recognized for producing coherent text, have been pruned and fine-tuned for text summarization tasks [77]. They developed efficient models capable of producing short summaries of larger texts by deleting extraneous parameters and fine-tuning summarization datasets.
3) **ALBERT for Sentiment Analysis** A Lite BERT (ALBERT) models are a more compact form of BERT models produced by parameter reduction approaches, have been used in applications such as sentiment analysis [78]. Researchers have created algorithms that can successfully assess the sentiment of a movie by reducing ALBERT further or utilizing it as a starting point for fine-tuning. Utilizing the Albert model trained on Stanford's ''movie review dataset,'' [78] achieved 89.05% accuracy in sentiment analysis, outperforming traditional LSTM and GRU models by 3%.
4) **DistilBERT for Question Answering** DistilBERT, a distilled version of BERT, has been modified for question-answering jobs [79]. Authors build models ideal for tasks such as extracting responses from text passages by pruning BERT and distilling its knowledge into a smaller model such as DistilBERT, while preserving competitive performance and decreasing computing cost.
5) **Transformer-Based Models for Machine Translation** Machine translation tasks have been performed using transformer-based models such as BERT or GPT, following pruning and fine-tuning [80]. They have produced encouraging results by customizing the models to translate text across different languages, using the information gained during pre-training and applying it to translation tasks.

Exploration of the transferability of pruned models across different NLP tasks or domains must be performed. Understanding how pruning in one task can be leveraged to benefit other related tasks or domains can lead to more efficient model design and deployment.

D. ROBUSTNESS AND GENERALIZATION

Pruning may affect a model's robustness and generalization in a variety of ways, with empirical evidence indicating both positive and negative impacts depending on the pruning technique, task, and other factors. The positive impacts of pruning are discussed below:

1) Pruning can serve as a type of regularization, lowering model complexity and preventing overfitting. Pruning might encourage the model to focus on the most informative characteristics, resulting in better generalization [17].
2) Pruning can help the model learn simpler and more interpretable representations of the input. Simplified representations frequently result in greater generalization since the model is less likely to remember noise or irrelevant features from the training data. Han et al. presented the ''Deep Compression'' approach, which compresses neural networks using pruning, quantization, and Huffman coding. They showed that compressed models might achieve similar or even higher accuracy than original models on a variety of image classification tasks while being much smaller and more computationally efficient [4].
3) Pruning decreases the size of the model, resulting in greater computing efficiency. This efficiency may result in speedier training and inference durations, allowing the model to generalize more successfully across diverse datasets or tasks [6].

The negative impacts of pruning are enumerated below:

1) Loss of Information: Pruning can remove connections or parameters that are significant for the model's performance on unseen data. If pruning is too aggressive or not done appropriately, it might result in the loss of critical information and diminished generalization [81].
2) Certain pruning approaches may produce models that are sensitive to initial parameter values or pruning thresholds. This sensitivity might make it difficult to attain consistent performance across different runs or datasets, reducing the model's resilience [82].
3) The effect of pruning on resilience and generalization varies depending on the task and dataset. A model that performs well on one task after pruning may not generalize as well to another task or dataset, emphasizing the significance of task-specific assessment [82].

The effects of pruning on model robustness and generalization capabilities must be explored as pruning may introduce new challenges, such as sensitivity to adversarial

attacks or over-fitting to specific training data. Research should focus on techniques that enhance model robustness and generalization performance after pruning.

E. HARDWARE-FRIENDLY PRUNING

Pruning methods differ in their hardware compatibility, and applying them on different devices might provide a variety of issues. These are discussed below:

1) **Structure Pruning:** This strategy removes whole layers or sub-networks from the model. It can be possibly hardware-friendly because it simplifies the overall design. However, changing hardware to remove whole layers or sub-networks can be challenging, especially with fixed-function hardware such as Application-Specific Integrated Circuits (ASICs) or FPGAs [83].
2) **Filter Pruning:** Filter pruning involves removing complete filters (also known as kernels or channels) from convolutional layers depending on their significance ratings. This approach can minimize computational complexity and the pruned networks can be immediately installed on off-the-shield platforms without the need for specialized hardware or libraries, this technique is hardware-friendly [84].
3) **Layer Pruning:** Layer pruning removes entire layers from the network based on their relevance. Hardware compatibility is determined by the hardware's ability to accept various network depths dynamically. It could be difficult on hardware with fixed architectures [85].
4) **Channel Pruning:** Channel pruning, like filter pruning, removes all channels (or individual feature maps) from convolutional layers. Hardware implementation issues include maintaining irregular tensor forms and effectively skipping channels during inference [86].
5) **Movement Pruning:** Movement pruning aims to reduce the amount of connections that cause movement in the activation region during training. Hardware compatibility is dependent on the ability to efficiently skip or decrease connections dynamically during inference, which may need specific hardware support. Movement-pruned models are quite sparse and make for effective storage. In simple terms, sparse matrices are matrices with a large number of zeros inside without particular hardware enhancements. Hardware manufacturers have recently released chips made especially for sparse networks [48].
6) **Unstructured Pruning:** Unstructured pruning removes individual weights or connections from the network. While it can achieve high compression rates, hardware support for unusual sparsity patterns may be difficult, particularly with hardware designed for regular tensor operations. As unstructured pruning isolates software from hardware and ignores the computing and continuous transmission properties of data flow in hardware, it is not hardware-friendly [86].
7) **Magnitude-based Pruning:** This strategy removes weights with small magnitudes, presuming they are less essential. Magnitude-based pruning requires less memory usage and reduce computation, it is thought to be hardware-friendly and is appropriate for use on edge devices with constrained resources. However, for effective sparse matrix computations, specialist hardware support might be needed [60].
8) **Iterative Pruning:** Iterative pruning involves repeatedly removing and fine-tuning the model. The irregular sparsity patterns produced by the iterative pruning process can result in inefficient memory access and computation on hardware accelerators. Individual weights are pruned by the algorithm, leading to non-structured sparsity, which can be challenging for hardware to optimize. Because the process is iterative, there are less options for parallelization, which makes it less appropriate for hardware acceleration. Nonetheless, the approach can be modified to make use of methods like sparse matrix compression and block-based pruning to increase hardware efficiency [87].
9) **Neuron Pruning:** Neuron pruning involves removing whole neurons from fully connected layers depending on their value. Neuron pruning reduces network parameters significantly speeds up processing and uses less energy by removing entire neurons that are not useful. Additionally, it minimizes weight matrices' dimensions, making hardware implementation more effective [88].
10) **Lottery Ticket Hypothesis:** The hypothesis proposes that sparse, trainable sub-networks (winning tickets) exist inside dense networks and may be identified by pruning. Hardware compatibility relies on the capacity to effectively identify and train sparse sub-networks, which may need specific hardware support for sparse tensor operations and dynamic network reconfiguration [89].

Hardware implementation of NLP-based tasks in different edge devices, mobiles, and robotics still has very stringent constraints of limited memory and processing. How to propose the pruning methods that fully exploit available hardware is the crucial aspect that needs to be addressed. Further, evaluating hardware constraints and designing pruning techniques that align with specific hardware architectures must be explored, as customizing the pruning strategies for a particular hardware platform can maximize the efficiency and performance gains by effectively leveraging the hardware capabilities.

F. ADAPTIVE PRUNING FOR EVOLVING NLP HARDWARE

Future research on adaptable pruning methods should concentrate on dynamically adjusting the pruning strategy based on the hardware platform. For example, prioritizing sparsity for CPUs, exploiting specific architectures like tensor processing units (TPUs) and evolving characteristics of NLP models, such as adapting to new architectures

like transformers. This would guarantee the best model compression and performance for both current and future NLP hardware and model environments.

G. ETHICAL CONSIDERATIONS IN PRUNING FOR SENSITIVE NLP APPLICATIONS

Pruning approaches must be ethically considered in sensitive NLP applications such as healthcare and criminal justice. Pruning techniques must be properly executed to prevent exacerbating biases, compromising privacy, or producing unfair results. Transparency, accountability, and thorough review are required to ensure the proper use of pruned models, to prevent unexpected results, and to maintain ethical norms in sensitive domains.

REFERENCES

[1] S. V. Naik, S. K. Majjigudda, S. Naik, S. M. Dandin, U. Kulkarni, S. M. Meena, S. V. Gurlahosur, and P. Benagi, "Survey on comparative study of pruning mechanism on MobileNetV3 model," in *Proc. Int. Conf. Intell. Technol. (CONIT)*, Jun. 2021, pp. 1–8.

[2] A. H. Zadeh, I. Edo, O. M. Awad, and A. Moshovos, "GOBO: Quantizing attention-based NLP models for low latency and energy efficient inference," in *Proc. 53rd Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2020, pp. 811–824.

[3] G. Bellec, D. Kappel, W. Maass, and R. Legenstein, "Deep rewiring: Training very sparse deep networks," 2017, *arXiv:1711.05136*.

[4] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.

[5] M. M. Islam and M. Alawad, "Stochastically pruning large language models using sparsity regularization and compressive sensing," in *Proc. Great Lakes Symp. VLSI*, Jun. 2023, pp. 63–68.

[6] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," 2016, *arXiv:1611.06440*.

[7] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. Peter Graf, "Pruning filters for efficient ConvNets," 2016, *arXiv:1608.08710*.

[8] M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," 2017, *arXiv:1710.01878*.

[9] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, "Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned," 2019, *arXiv:1905.09418*.

[10] A. Onan, "GTR-GA: Harnessing the power of graph-based neural networks and genetic algorithms for text augmentation," *Expert Syst. Appl.*, vol. 232, Dec. 2023, Art. no. 120908, doi: 10.1016/j.eswa.2023.120908.

[11] A. Onan and K. Filiz Balbal, "Improving Turkish text sentiment classification through task-specific and universal transformations: An ensemble data augmentation approach," *IEEE Access*, vol. 12, pp. 4413–4458, 2024.

[12] A. Onan, "SRL-ACO: A text augmentation framework based on semantic role labeling and ant colony optimization," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 35, no. 7, Jul. 2023, Art. no. 101611.

[13] A. Onan, "Hierarchical graph-based text classification framework with contextual node embedding and BERT-based dynamic fusion," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 35, no. 7, Jul. 2023, Art. no. 101610, doi: 10.1016/j.jksuci.2023.101610.

[14] A. Onan, "Bidirectional convolutional recurrent neural network architecture with group-wise enhancement mechanism for text sentiment classification," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 5, pp. 2098–2117, May 2022.

[15] J.-M. Attendu and J.-P. Corbeil, "NLU on data diets: Dynamic data subset selection for NLP classification tasks," 2023, *arXiv:2306.03208*.

[16] Z. Fan, Q. Zhang, P. Abillama, S. Shoouri, C. Lee, D. Blaauw, H.-S. Kim, and D. Sylvester, "TaskFusion: An efficient transfer learning architecture with dual delta sparsity for multi-task natural language processing," in *Proc. 50th Annu. Int. Symp. Comput. Archit.*, Jun. 2023, pp. 1–14.

[17] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," 2017, *arXiv:1710.09282*.

[18] N. Rethmeier, "Efficient, adaptable and interpretable NLP," Ph.D. thesis, Univ. Copenhagen, Copenhagen, Denmark, 2023.

[19] K. Ramesh, A. Chavan, S. Pandit, and S. Sitaram, "A comparative study on the impact of model compression techniques on fairness in language models," in *Proc. 61st Annu. Meeting Assoc. Comput. Linguistics*, 2023, pp. 15762–15782.

[20] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "Nisp: Pruning networks using neuron importance score propagation," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, Sep. 2018, pp. 9194–9203.

[21] T. Furlanello, Z. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1607–1616.

[22] M. Omar, "Backdoor learning for NLP: Recent advances, challenges, and future research directions," 2023, *arXiv:2302.06801*.

[23] C. Westbrook and S. Pasricha, "Adversarial attacks on machine learning in embedded and IoT platforms," 2023, *arXiv:2303.02214*.

[24] M. Du, S. Mukherjee, Y. Cheng, M. Shokouhi, X. Hu, and A. H. Awadallah, "What do compressed large language models forget? Robustness challenges in model compression," 2021, *arXiv:2110.08419*.

[25] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proc. 49th Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2011, pp. 142–150.

[26] O. Bojar et al., "Findings of the 2016 conference on machine translation," in *Proc. 1st Conf. Mach. Translation*, vol. 2, 2016, pp. 131–198.

[27] R. Socher et al., "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. EMNLP*, vol. 1631, Jan. 2013, pp. 1631–1642.

[28] M. Tkachenko and A. Simanovsky, "Named entity recognition: Exploring features," *KONVENS*, vol. 292, pp. 118–127, Sep. 2012.

[29] X. Ma, G. Fang, and X. Wang, "LLM-pruner: On the structural pruning of large language models," 2023, *arXiv:2305.11627*.

[30] J. S. McCarley, R. Chakravarti, and A. Sil, "Structured pruning of a BERT-based question answering model," 2019, *arXiv:1910.06360*.

[31] Z. Yang, Y. Cui, and Z. Chen, "TextPruner: A model pruning toolkit for pre-trained language models," 2022, *arXiv:2203.15996*.

[32] Z. Wang, J. Wohlwend, and T. Lei, "Structured pruning of large language models," 2019, *arXiv:1910.04732*.

[33] Y. Peng, K. Kim, F. Wu, P. Sridhar, and S. Watanabe, "Structured pruning of self-supervised pre-trained models for speech recognition and understanding," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2023, pp. 1–5.

[34] Y. Lian, P. Peng, and W. Xu, "Filter pruning via separation of sparsity search and model training," *Neurocomputing*, vol. 462, pp. 185–194, Oct. 2021.

[35] Q. Li, P. Li, K. Mao, and E. Y.-M. Lo, "Improving convolutional neural network for text classification by recursive data pruning," *Neurocomputing*, vol. 414, pp. 143–152, Nov. 2020.

[36] X. Ma, S. Lin, S. Ye, Z. He, L. Zhang, G. Yuan, S. Huat Tan, Z. Li, D. Fan, X. Qian, X. Lin, K. Ma, and Y. Wang, "Non-structured DNN weight pruning is it beneficial in any platform?" 2019, *arXiv:1907.02124*.

[37] H. Yu and J. Wu, "A unified pruning framework for vision transformers," *Sci. China Inf. Sci.*, vol. 66, no. 7, pp. 1–2, Jul. 2023.

[38] X. Liu, H. Tang, J. Zhao, Q. Dou, and M. Lu, "TCAMixer: A lightweight mixer based on a novel triple concepts attention mechanism for NLP," *Eng. Appl. Artif. Intell.*, vol. 123, Aug. 2023, Art. no. 106471.

[39] H. Sajjad, F. Dalvi, N. Durrani, and P. Nakov, "On the effect of dropping layers of pre-trained transformer models," *Comput. Speech Lang.*, vol. 77, Jan. 2023, Art. no. 101429.

[40] A. Jordao, M. Lie, and W. R. Schwartz, "Discriminative layer pruning for convolutional neural networks," *IEEE J. Sel. Topics Signal Process.*, vol. 14, no. 4, pp. 828–837, May 2020.

[41] C. Fan, J. Li, X. Ao, F. Wu, Y. Meng, and X. Sun, "Layer-wise model pruning based on mutual information," 2021, *arXiv:2108.12594*.

[42] D. Peer, S. Stabinger, S. Engl, and A. Rodríguez-Sánchez, "Greedy-layer pruning: Speeding up transformer models for natural language processing," *Pattern Recognit. Lett.*, vol. 157, pp. 76–82, May 2022.

[43] M. Cho, S. Adya, and D. Naik, "PDP: Parameter-free differentiable pruning is all you need," 2023, *arXiv:2305.11203*.

[44] F. Lagunas, E. Charlaix, V. Sanh, and A. M. Rush, "Block pruning for faster transformers," 2021, *arXiv:2109.04838*.

[45] H. Peng, S. Huang, T. Geng, A. Li, W. Jiang, H. Liu, S. Wang, and C. Ding, "Accelerating transformer-based deep learning models on FPGAs using column balanced block pruning," in *Proc. 22nd Int. Symp. Quality Electron. Design (ISQED)*, Apr. 2021, pp. 142–148.

[46] B. Li, Z. Kong, T. Zhang, J. Li, Z. Li, H. Liu, and C. Ding, "Efficient transformer-based large scale language representations using hardware-friendly block structured pruning," 2020, *arXiv:2009.08065*.

[47] X. Ma, G. Yuan, Z. Li, Y. Gong, T. Zhang, W. Niu, Z. Zhan, P. Zhao, N. Liu, J. Tang, X. Lin, B. Ren, and Y. Wang, "BLCR: Towards real-time DNN execution with block-based reweighted pruning," in *Proc. 23rd Int. Symp. Qual. Electron. Design*, 2022, pp. 1–8.

[48] V. Sanh, T. Wolf, and A. Rush, "Movement pruning: Adaptive sparsity by fine-tuning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 20378–20389.

[49] P. Joniak and A. Aizawa, "Gender biases and where to find them: Exploring gender bias in pre-trained transformer-based language models using movement pruning," 2022, *arXiv:2207.02463*.

[50] S. Narang, E. Elsen, G. Diamos, and S. Sengupta, "Exploring sparsity in recurrent neural networks," 2017, *arXiv:1704.05119*.

[51] X. Chen, J. Zhu, J. Jiang, and C.-Y. Tsui, "Tight compression: Compressing CNN model tightly through unstructured pruning and simulated annealing based permutation," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Sep. 2020, pp. 1–6.

[52] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," 2016, *arXiv:1607.03250*.

[53] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "NISP: Pruning networks using neuron importance score propagation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2018, pp. 9194–9203.

[54] S. Preite, "Deep question answering: A new teacher for distilBERT," M.S. thesis, Alma Mater Studiorum Università di Bologna, Bologna, Italy, 2019.

[55] H. Wang, C. Qin, Y. Zhang, and Y. Fu, "Neural pruning via growing regularization," 2020, *arXiv:2012.09243*.

[56] A. See, M.-T. Luong, and C. D. Manning, "Compression of neural machine translation models via pruning," 2016, *arXiv:1606.09274*.

[57] G. Li, C. Qian, C. Jiang, X. Lu, and K. Tang, "Optimization based layer-wise magnitude-based pruning for DNN compression," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2383–2389.

[58] J. Lee, S. Park, S. Mo, S. Ahn, and J. Shin, "Layer-adaptive sparsity for the magnitude-based pruning," 2020, *arXiv:2010.07611*.

[59] W. Hong, P. Yang, Y. Wang, and K. Tang, "Multi-objective magnitude-based pruning for latency-aware deep neural network compression," in *Proc. 16th Int. Conf.*, 2020, pp. 470–483.

[60] G. Li, P. Yang, C. Qian, R. Hong, and K. Tang, "Stage-wise magnitude-based pruning for recurrent neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 1, no. 1, pp. 1–15, May 2022.

[61] A. Mallya and S. Lazebnik, "PackNet: Adding multiple tasks to a single network by iterative pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7765–7773.

[62] S. Rajaraman, J. Siegelman, P. O. Alderson, L. S. Folio, L. R. Folio, and S. K. Antani, "Iteratively pruned deep learning ensembles for COVID-19 detection in chest X-Rays," *IEEE Access*, vol. 8, pp. 115041–115050, 2020.

[63] T. Chen, J. Frankle, S. Chang, S. Liu, Y. Zhang, Z. Wang, and M. Carbin, "The lottery ticket hypothesis for pre-trained BERT networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 15834–15846.

[64] M. Behnke and K. Heafield, "Losing heads in the lottery: Pruning transformer attention in neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2020, pp. 2664–2674.

[65] S. Ding, T. Chen, and Z. Wang, "Audio lottery: Speech recognition made ultra-lightweight, noise-robust, and transferable," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–18.

[66] Z. Yang, Y. Cui, and Z. Chen, "Textpruner: A model pruning toolkit for pre-trained language models," 2022, *arXiv:2203.159961*.

[67] H.-J. Yoon, S. Robinson, J. B. Christian, J. X. Qiu, and G. D. Tourassi, "Filter pruning of convolutional neural networks for text classification: A case study of cancer pathology report comprehension," in *Proc. IEEE EMBS Int. Conf. Biomed. Health Informat.*, Jun. 2018, pp. 345–348.

[68] G. Xu and Q. Hu, "Can model compression improve NLP fairness," 2022, *arXiv:2201.08542*.

[69] Z. Wang, "SparseRT: Accelerating unstructured sparsity on GPUs for deep learning inference," 2020, *arXiv:2008.11849*.

[70] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis.*, Feb. 2017, pp. 5058–5066.

[71] W. He, M. Wu, M. Liang, and S.-K. Lam, "CAP: Context-aware pruning for semantic segmentation," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2021, pp. 959–968.

[72] C. Guo and P. Li, "Hybrid pruning method based on convolutional neural network sensitivity and statistical threshold," *J. Phys. Conf. Ser.*, vol. 2171, no. 1, Jan. 2022, Art. no. 012055.

[73] S. Wu, G. Li, F. Chen, and L. Shi, "Training and inference with integers in deep neural networks," 2018, *arXiv:1802.04680*.

[74] L. Liu, L. Deng, X. Hu, M. Zhu, G. Li, Y. Ding, and Y. Xie, "Dynamic sparse graph for efficient deep learning," 2018, *arXiv:1810.00859*.

[75] J. Kim, S. Chang, and N. Kwak, "PQK: Model compression via pruning, quantization, and knowledge distillation," 2021, *arXiv:2106.14681*.

[76] Y. Chang, L. Kong, K. Jia, and Q. Meng, "Chinese named entity recognition method based on BERT," in *Proc. IEEE Int. Conf. Data Sci. Comput. Appl. (ICDSCA)*, Oct. 2021, pp. 294–299.

[77] N. Alexandr, O. Irina, K. Tatyana, K. Inessa, and P. Arina, "Fine-tuning GPT-3 for Russian text summarization," in *Lecture Notes in Networks and Systems*. Cham, Switzerland: Springer, 2021, pp. 748–757.

[78] Z. Ding, Y. Qi, and D. Lin, "Albert-based sentiment analysis of movie review," in *Proc. 4th Int. Conf. Adv. Electron. Mater., Comput. Softw. Eng. (AEMCSE)*, Mar. 2021, pp. 1243–1246.

[79] O. Nordström, "Unstructured pruning of pre-trained language models tuned for sentiment classification," M.S. thesis, KTH, Math. Statist., 2022.

[80] A. Raganato and J. Tiedemann, "An analysis of encoder representations in transformer-based machine translation," in *Proc. EMNLP Workshop BlackboxNLP: Analyzing Interpreting Neural Netw.*, 2018, pp. 287–297.

[81] M. Mousa Pasandi, M. Hajabdollahi, N. Karimi, and S. Samavi, "Modeling of pruning techniques for deep neural networks simplification," 2020, *arXiv:2001.04062*.

[82] X. Chen, J. Mao, and J. Xie, "Comparison analysis for pruning algorithms of neural networks," in *Proc. 2nd Int. Conf. Comput. Eng. Intell. Control*, 2021, pp. 50–56.

[83] Z. Liu, Q. Liu, S. Yan, and R. C. C. Cheung, "An efficient FPGA-based depthwise separable convolutional neural network accelerator with hardware pruning," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 17, no. 1, pp. 1–20, Mar. 2024.

[84] F. Yu, C. Han, P. Wang, R. Huang, X. Huang, and L. Cui, "HFP: Hardware-aware filter pruning for deep convolutional neural networks acceleration," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 255–262.

[85] H. Li and L. Meng, "Hardware-aware approach to deep neural network optimization," *Neurocomputing*, vol. 559, Nov. 2023, Art. no. 126808.

[86] X. Sui, Q. Lv, L. Zhi, B. Zhu, Y. Yang, Y. Zhang, and Z. Tan, "A hardware-friendly high-precision CNN pruning method and its FPGA implementation," *Sensors*, vol. 23, no. 2, p. 824, Jan. 2023.

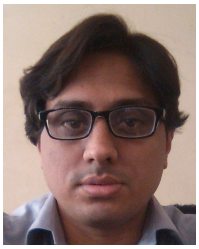
[87] G. Castellano, A. M. Fanelli, and M. Pelillo, "An iterative pruning algorithm for feedforward neural networks," *IEEE Trans. Neural Netw.*, vol. 8, no. 3, pp. 519–531, May 1997.

[88] W. Guo, H. E. Yantır, M. E. Fouda, A. M. Eltawil, and K. N. Salama, "Towards efficient neuromorphic hardware: Unsupervised adaptive neuron pruning," *Electronics*, vol. 9, no. 7, p. 1059, Jun. 2020.

[89] T. Chen, X. Chen, X. Ma, Y. Wang, and Z. Wang, "Coarsening the granularity: Towards structurally sparse lottery tickets," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 3025–3039.

MARVA TOUHEED received the degree in computer systems engineering from Mirpur University of Science and Technology (MUST), Azad Kashmir. She is currently pursuing the M.S. degree in computer engineering with COMSATS University Islamabad. Her research interests include natural language processing, deep learning, convolution neural networks, generative adversarial networks, super-resolution, and image processing.

UROOJ ZUBAIR received the B.Sc. degree in computer systems engineering from Mirpur University of Science and Technology (MUST), Azad Kashmir. She is currently pursuing the Master of Science degree in computer engineering with COMSATS University Islamabad. Her research interests include natural language processing, compressed sensing deep learning, convolution neural networks, CUDA optimization, super-resolution, and image processing.



DILSHAD SABIR received the degree in computer engineering from COMSATS University Islamabad (CUI), Islamabad, and the M.S. and Ph.D. degrees in computer engineering from the National University of Sciences and Technology, Islamabad, in 2022. He is currently a Faculty Member with CUI. His research interests include the Internet of Things (IoT), machine learning, deep learning, correlation pattern recognition, convolution neural networks, and network security.

ALI HASSAN received the B.E. and M.S. degrees in computer engineering from the College of Electrical and Mechanical Engineering, National University of Sciences and Technology (NUST), Pakistan, in 2004 and 2007, respectively, and the Ph.D. degree in electrical engineering from the University of Southampton, U.K., in 2012. He is currently with the College of Electrical and Mechanical Engineering, NUST, as a Professor, and the Head of Department of the Department of Computer and Software Engineering. His research interests include the application of machine learning to speech and image processing in the domains of speech, texture classification, and biomedical engineering. He has established research linkages with several international partners and is also one of the Heads of the Centre for Machine Learning and Biomedical Engineering. These research linkages have resulted in several international research funding for the faculty members and the students. He has over 100 international publications.

MUHAMMAD FASIH UDDIN BUTT (Member, IEEE) received the B.E. degree from the National University of Science and Technology (NUST), Pakistan, in 1999, the M.Sc. degree in computer engineering from the Center for Advanced Studies in Engineering, University of Engineering and Technology (UET), Taxila, Pakistan, in 2003, with a focus on digital communication/computer networks, and the Ph.D. degree in electronics and electrical engineering from the Communications Research Group, School of Electronics and Computer Science (ECS), University of Southampton, U.K., in June 2010, a with focus on wireless communication systems. In 2021, he was a Postdoctoral Researcher of machine learning for the massive Internet of Things with the Next Research Wireless Research Group, School of ECS, University of Southampton, focusing on smart cities applications. He joined COMSATS University Islamabad (CUI), Pakistan, in 2002, where he is currently a Tenured Professor with the Department of Electrical and Computer Engineering and the Head of the NGCRG Laboratory. He is an Innovative and Experienced Researcher with a proven track record of prestigious peer-reviewed journals and conferences. He has published over 50 research papers in various reputed journals and conference proceedings. He has successfully supervised five Ph.D. and 22 M.Sc. students in the field of telecommunications engineering. His research interests include radio over fiber technologies, physical-layer security, channel coding, cooperative cognitive radio networks, applications of machine learning and artificial intelligence in various fields, and efficient hardware implementation of high throughput decoders.

FARHAN RIAZ received the B.E. degree from the National University of Sciences and Technology (NUST), Pakistan, the M.S. degree from the Technical University of Munich, Germany, and the Ph.D. degree from the University of Porto, Portugal. He is currently a Senior Lecturer with the School of Computer Science, University of Lincoln, U.K. He is also a Senior Consultant of data science with Troon Technologies LLC. Prior to these engagements, he was an Associate Professor with NUST. His research interests include biomedical signal and image processing, applied machine learning, and computer vision.

WADOOD ABDUL received the Ph.D. degree in signal and image processing from the University of Poitiers, France, in 2011. Currently, he is a Professor with the Department of Computer Engineering, College of Computer and Information Sciences, King Saud University. He developed the Communications Laboratory by Lucus Nulle and the Biometrics Laboratory funded by ZKTeco at King Saud University. His research interests include multimedia security, biometrics, agriculture applications, privacy, medical image processing, and video understanding, where he is working on several externally funded research projects. He has published over 100 papers in well-reputed conferences and journals. He received the Best Faculty Award from the College of Computer and Information Sciences, King Saud University, in 2017.

RASHID AYUB received the double master's (M.C.A. and M.Sc.) degree in mathematics and computer science, in 1996, and the Ph.D. degree in computer science and applications. His Ph.D. thesis entitled "Extension of Bounded Rationality in Artificial Intelligence to Sensing Activity Through Limited Resource Sensing Problem." He is currently a Researcher with the Science Technology and Innovation Department, National Plan for Science and Technology (NPST) Setup, King Saud University, Riyadh, Saudi Arabia. He has more than two decades of working experience both in industry and academia in India and abroad. He was also a Trainer by United Nations Training Program on "Integrated Management Information System (iMIS) Based on ERP," UNEP Office, Bangkok, Thailand. He was also an MIS Officer with the Ministry of Environment and Forest, New Delhi, India, and developed "MIS Vision Document for Compliance Era of 2004–2007" covering aspects, such as e-based mechanisms for dissemination of information, knowledge management, and technology transfer. He has published several research papers in international journals and refereed conferences. His research interests include the Internet of Things (IoT), deep learning, artificial intelligence, machine learning, cloud computing, neural networks, and fuzzy logic and their applications in the areas of e-healthcare, smart cities, and bio-informatics.