mathematics

Article

A Comparative Analysis of Compression and Transfer Learning Techniques in DeepFake Detection Models

Andreas Karathanasis †, John Violos *,† and Ioannis Kompatsiaris

Information Technologies Institute, Centre for Research & Technology, Hellas, 57001 Thessaloniki, Greece; andrew.karathanasis@iti.gr (A.K.); ikom@iti.gr (I.K.)
* Correspondence: violos@iti.gr
† These authors contributed equally to this work.

Abstract: DeepFake detection models play a crucial role in ambient intelligence and smart environments, where systems rely on authentic information for accurate decisions. These environments, integrating interconnected IoT devices and AI-driven systems, face significant threats from DeepFakes, potentially leading to compromised trust, erroneous decisions, and security breaches. To mitigate these risks, neural-network-based DeepFake detection models have been developed. However, their substantial computational requirements and long training times hinder deployment on resource-constrained edge devices. This paper investigates compression and transfer learning techniques to reduce the computational demands of training and deploying DeepFake detection models, while preserving performance. Pruning, knowledge distillation, quantization, and adapter modules are explored to enable efficient real-time DeepFake detection. An evaluation was conducted on four benchmark datasets: "SynthBuster", "140k Real and Fake Faces", "DeepFake and Real Images", and "ForenSynths". It compared compressed models with uncompressed baselines using widely recognized metrics such as accuracy, precision, recall, F1-score, model size, and training time. The results showed that a compressed model at 10% of the original size retained only 56% of the baseline accuracy, but fine-tuning in similar scenarios increased this to nearly 98%. In some cases, the accuracy even surpassed the original's performance by up to 12%. These findings highlight the feasibility of deploying DeepFake detection models in edge computing scenarios.

Keywords: ambient intelligence; smart environments; deepfake models; compression; transfer learning; edge computing

MSC: 68T45

check for updates

Academic Editor: Antanas Cenys

Received: 31 January 2025
Revised: 28 February 2025
Accepted: 3 March 2025
Published: 6 March 2025

Citation: Karathanasis, A.; Violos, J.; Kompatsiaris, I. A Comparative Analysis of Compression and Transfer Learning Techniques in DeepFake Detection Models. *Mathematics* **2025**, *13*, 887. https://doi.org/10.3390/math13050887

1. Introduction

Ambient intelligence and smart environments are composed of interconnected IoT devices and AI systems that exchange information to deliver seamless user experiences [1]. However, DeepFakes—synthetic media created using advanced AI—pose a significant threat to these environments by introducing falsified data that can deceive both systems and users [2]. For instance, in smart surveillance systems, DeepFakes could be used to manipulate video feeds, leading to unauthorized access or false alarms [3]. Similarly, voice-controlled smart home devices could be deceived by DeepFake audio commands, resulting in unauthorized operations or breaches of privacy [4]. The integration of DeepFake detection models into these systems is essential to maintain security, ensure accurate human–computer interactions, and uphold user trust. Recent research has explored the application of DeepFake detection in IoT-based applications, emphasizing the need for robust safeguards against such vulnerabilities [5]. Additionally, studies have proposed frameworks that combine AI and IoT technologies to enhance surveillance security by effectively identifying individuals and detecting DeepFake-generated content [6]. By implementing DeepFake detection models, ambient intelligent systems can better protect against malicious activities, ensuring that automated decisions are based on authentic and reliable data.

Running and training DeepFake detection models on a smart environment comprising IoT and edge devices is essential for enabling real-time detection directly at the data source. This approach reduces the need for constant data transfers to centralized servers, minimizing latency and bandwidth usage [7]. It also enhances privacy and security by processing sensitive data locally, addressing the privacy concerns inherent in ambient intelligence systems, which are often criticized for their pervasive data collection and potential misuse [8].

Deep fake detection models leverage advanced neural networks like convolutional neural networks (CNNs), and as a consequence they demand substantial computational resources and memory, rendering them impractical for deployment on resource-constrained devices [9]. Neural network compression techniques such as pruning [10], quantization [11], knowledge distillation [12], and low-rank factorization [13] enable these models to have reduced computational demands while maintaining accuracy. These optimizations are crucial for deploying efficient, real-time detection systems in ambient intelligence environments, where devices must operate autonomously with limited hardware capabilities. Achieving this balance between efficiency and performance is crucial for lightweight inference and effective DeepFake threat detection.

While compressing neural networks reduces the resources needed to efficiently run DeepFake detection models, the challenge of training these models with reduced data, time, and computational demands persists. Rather than training a DeepFake detection model from scratch, it is more effective to utilize pre-trained neural networks that have already acquired rich, generalized patterns from extensive datasets. Transfer learning [14] enables these models to harness knowledge from related tasks, such as binary image classification, by reusing learned representations and fine-tuning pre-trained models. Additionally, the adaptation of transformer adapters [15] to deep fake detection CNNs has been explored to further improve transfer learning's effectiveness in addressing dynamic and evolving threats within ambient intelligence ecosystems.

In this article, compression and transfer learning techniques for DeepFake detection models are analyzed and evaluated. Specifically, the methods of pruning, quantization, knowledge distillation, and low-rank factorization were explored, assessing their effectiveness using four benchmark datasets: "Synthbuster", "140k Real and Fake Faces", "DeepFake and Real Images", and "ForenSynths". The evaluation was conducted using metrics such as accuracy, precision, recall, F1 score, and compression time. This research provides four key contributions:

- It demonstrates that compressed models can achieve performance levels comparable to uncompressed models, even when compressed to 40%, 30%, 20%, or 10% of the original model size.
- It proposes various approaches for applying transfer learning to DeepFake detection models, instead of training them from scratch, tackling the challenge of efficiently training models with limited resources and data.
- It conducted extensive evaluations across multiple benchmark datasets to establish the generalizability and robustness of compressed models for real-world DeepFake detection applications.

- It presents experimental results and provides an empirical analysis of how different types of synthetic image generators and image types impact the effectiveness of transfer learning.

The structure of this paper is as follows: Section 2 reviews related work in the fields of DeepFake detection, compression, and transfer learning in CNN models; Sections 3 and 4 describe the methodologies followed for compression and transfer learning for DeepFake detection, respectively. Section 5 reports the experimental results. Section 6 discusses the research implications, while Section 7 outlines future work, and Section 8 provides the conclusions.

2. A Critical Literature Review of Deep Fake Detection, Compression and Transfer Learning Techniques

Ambient intelligence refers to digital environments that are sensitive and responsive to the presence of people, integrating technologies seamlessly into daily life to enhance user experience [1]. Smart environments, including smart cities and smart homes, are a practical manifestation of ambient intelligence that utilize interconnected devices and systems to provide automated and adaptive services [16]. However, related research has shown that smart environments are vulnerable to the threats posed by DeepFake technology [6]. To address these critical risks, various DeepFake detection models have been developed. In the following subsection, the current state of the art in DeepFake models is explored. Additionally, to enhance the development and execution of these models, making them more efficient and suitable for deployment in resource-constrained smart environments, related work on compression techniques and transfer learning approaches is reviewed.

2.1. DeepFake Detection

The utilization of Visual Transformers (ViT) and CNNs are two of the most sophisticated techniques employed for the detection of DeepFake images. In [17], the authors conducted a comprehensive study evaluating the performance of CNNs and ViTs in DeepFake detection under various conditions. Their experiments included scenarios where models were trained and tested on a single type of DeepFake generation method, as well as multiple methods simultaneously. The robustness and generalization capabilities of these models were assessed through a cross-forgery analysis, which benchmarked how well a model trained on one type of forgery performed on different, unseen forgeries. The study concluded that ViTs demonstrated superior generalization, making them more robust against new and emerging DeepFake generation techniques. Similar conclusions for robustness can be made from other papers like [18–20]. Conversely, CNNs showed better performance in specialized tasks where the nature of the forgery was consistent and known.

In the paper [21], the authors introduced an innovative approach leveraging capsule networks for forgery detection. Capsule networks, which are adept at capturing spatial hierarchies, offer a significant advantage over traditional CNNs by preserving hierarchical relationships within data. Capsule networks consist of groups of neurons, called capsules, that encode various properties of objects and the spatial relationships and orientations of objects or features within an image. This capability is critical for identifying subtle inconsistencies in forged media. The authors extended the application of capsule networks from other computer vision tasks like [22,23] to include DeepFake detection use cases, such as replay attack detection. The experimental results demonstrated that capsule networks outperformed CNNs, showcasing higher accuracy and superior generalization abilities, especially in detecting novel and varied forgery techniques.

Although ViTs and CapsNets represent a commendable option for the task of DeepFake detection, they also come with significant drawbacks. ViTs typically have more parameters and require more computational resources than CNNs, as pointed out in [24]. This is due to the self-attention mechanism in ViTs, which scales quadratically with input size, compared to the localized receptive fields in CNNs. As for Capsule Networks, the work of [25] suggested that they are computationally intensive due to the absence of pooling operations. As a result, they must account for every aspect of the input image, including background noise. Additionally, their performance is inconsistent across different datasets and they struggle with distinguishing multiple instances of the same entity. This variability and the current lack of optimal implementations make CapsNets less reliable for time-critical and high-accuracy applications like DeepFake detection. These drawbacks of ViTs and CapsNets conflict with the overarching goal of efficient training and deployment in ambient intelligence environments, whereas CNNs, combined with compression and transfer learning, are considered a more suitable approach.

The paper in [26] presented a novel approach to DeepFake detection, which is named DeepFakeUCL and leverages unsupervised contrastive learning. This paper stands out by describing a method that does not rely on labeled DeepFake data for training. Contrastive learning is most commonly used as an unsupervised method [27–30], although it can also be used in a supervised setting, as showcased in [31]. The proposed approach comprises three main steps: data preprocessing, unsupervised training, and follow-up classification. Data preprocessing involves transforming images to focus on the face area, optimizing the utilization of facial data. Unsupervised contrastive learning is then employed, where images are augmented to generate paired versions, and a backbone network is trained to learn features by maximizing the similarity between these versions. Notably, the Xception network [32] is utilized as the encoder, and a projection head network enhances the efficiency of the contrastive loss function. Finally, for evaluation, the features extracted through unsupervised learning are utilized in a linear classification network. Moreover, ablation studies conducted in the paper shed light on the effectiveness of unsupervised contrastive learning compared to supervised methods; the impact of different data augmentation schemes, something very important for effective learning [33]; and the superiority of features learned by the encoder over those from the projection head.

Consistency learning has been applied to various image forensics tasks with varying levels of supervision, as demonstrated by [34–36]. In their study, the authors of [37] proposed a model that detects image manipulations, specifically splicing, by leveraging the consistency between different patches within an image. Their method involves calculating the consistency between patch pairs, generating response maps, and then producing a global consistency map for the entire image using mean shifting. This global consistency map allows for the computation of a global consistency score, which facilitates the identification and localization of spliced regions within the image. The paper in [38] proposed a novel approach called pair-wise self-consistency learning (PCL) for detecting face forgeries generated by stitching-based techniques and localizing the manipulated regions within images. PCL leverages the inconsistency of source features within modified images, offering a lightweight solution that can be easily integrated into existing backbone networks. To support PCL training, the authors introduced an innovative method called the inconsistency image generator (I2G), which dynamically generates forged images along with annotations of their manipulated regions. Experimental results across seven popular datasets demonstrated the competitiveness of PCL and I2G against state-of-the-art methods, establishing a strong baseline for future research.

The methods described in papers [26,38], while demonstrating promising results, encounter specific challenges. Notably, their success is highly dependent on the DeepFake data utilized. DeepFakeUCL relies heavily on the quality and diversity of data augmentations, while PCL depends on identifying inconsistencies within images. This presents a problem, because sophisticated DeepFake techniques may produce highly consistent and low-variation forgeries, thus hindering models from learning robust features. Consequently, generalization issues arise. With the use of GANs for generation of DeepFake materials, which constantly evolve and improve, as noted in [39], the introduction of images with new inconsistencies into PCL can degrade its performance. Similarly, if the training data for DeepFakeUCL do not represent a broad distribution of DeepFakes, an arduous task given the numerous DeepFake generation methods, as illustrated in [40], the model may fail to generalize effectively. Finally, the computational resources required for both methods are substantial. PCL necessitates the use of an I2G, which is computationally intensive, especially for larger images or higher resolutions. DeepFakeUCL also uses an extra unsupervised learning stage, which requires significant computational resources.

The rapid advancements in generative AI make synthetic media increasingly difficult to distinguish from real content. To address this challenge, detection methods generally fall into the three categories analyzed in [41]: CNNs, Transformers, and GANs. CNNs remain widely used, due to their efficiency in detecting subtle inconsistencies, while Transformers offer strong generalization, at a higher computational cost. GAN-based approaches enhance robustness through adversarial training but continue to evolve to counter increasingly sophisticated forgeries. The survey highlighted that CNNs offer adaptability for evolving DeepFake techniques, while Transformers excel in generalization but demand significant resources. GAN-based methods show promise in enhancing detection resilience. The challenge of DeepFake detection has also been addressed with hierarchical detection frameworks that can distinguish real, GAN-generated, and diffusion model images. The work used a generative approach and framed the problem as Visual Question Answering (VQA) instead of the regular binary classification [42]. A vision-language transformer approach was proposed that improves DeepFake detection by leveraging common sense reasoning, enhancing interpretability and generalization [43].

2.2. Compression of CNNs

Pruning and quantization aim to reduce the computational complexity and memory demands of deep neural networks (DNNs) and are considered the two most widely recognized compression techniques, according to a survey article [10]. Pruning techniques are categorized into various types, such as element-wise, channel-wise, and layer-wise pruning, each offering distinct advantages, depending on the specific architecture and application. Similarly, quantization techniques, including weight quantization and activation quantization, are characterized by their ability to decrease memory requirements and computational complexity by reducing parameter precision [44]. The potential for combining these techniques to achieve enhanced compression and efficiency has also been underscored, as demonstrated by [45]. Their work introduced a three-stage process for high compression rates. This process includes pruning to reduce parameters, quantization to compress the remaining weights, followed by fine-tuning to recover performance, and finally, the application of Huffman coding for further lossless compression.

Knowledge distillation [46] involves transferring knowledge from a complex "teacher" model to a simpler "student" model, aiding in model compression and task-specific performance. This student–teacher (S-T) learning paradigm is often employed alongside knowledge distillation and utilizes this framework to train the student model to emulate the teacher's output [12]. Furthermore, in the paper in [12], the authors explored applications of knowledge distillation and S-T in visual intelligence tasks like image classification and object detection, emphasizing benefits such as enhanced efficiency and performance on edge devices. While the strengths of knowledge distillation and S-T were noted, such as improved efficiency, careful consideration of model selection and the risk of over-fitting were also highlighted.

A method for parameter reduction in DNNs through a low-rank matrix factorization applied to the final weight layer was proposed in [13]. By decomposing the weight matrix into two lower-dimensional matrices, the authors claimed that this can achieve a substantial reduction in the number of parameters, ranging from 30 to 50%, without compromising classification accuracy. The approach was validated in both acoustic modeling and language modeling tasks, where the low-rank models not only matched but often exceeded the performance of their full-rank counterparts. Additionally, the low-rank models demonstrated more efficient training processes. A similar work [47] focused on compression of the convolutional layers of a network by constructing a low-rank basis of filters using two distinct methods. The first method reduces computation by approximating the original convolutional filters using a linear combination of fewer, separable basis filters. The second method restructures the convolution process into two stages with rectangular filters, optimizing efficiency by leveraging redundancies across input and output channels.

CNN compression remains a critical challenge, due to high computational costs and memory demands. Even though new methods have emerged, such as StarSPA [48], which leverages stride-aware sparsity compression to enhance inference speed and energy efficiency, and multi-objective evolutionary algorithms [49], which optimize pruning by balancing multiple performance indicators, recent surveys like [50] have reaffirmed that traditional techniques remain the most effective. The survey in [50] provided a comprehensive review of CNN compression, highlighting pruning, quantization, knowledge distillation, and low-rank matrix factorization as the primary techniques. Pruning remains the most widely used approach for removing redundant parameters, while quantization enables efficient deployment by reducing numerical precision. Knowledge distillation continues to be a key strategy for training smaller models, without significant performance loss, and low-rank matrix factorization offers an alternative for reducing model complexity. The survey also emphasized the increasing adoption of hybrid methods that combine multiple techniques to optimize efficiency and accuracy, reinforcing the continued relevance of these traditional approaches.

2.3. Transfer Learning in CNNs

Fine-tuning is the process of further training a pre-trained model on a specific task by adjusting its parameters with a smaller learning rate, allowing it to adapt, while retaining previously learned features [14]. In transfer learning, fine-tuning helps repurpose a model trained on a large dataset for a new but related task, improving the performance with limited data, computational resources, and training time. The paper in [51] investigated the effectiveness of fine-tuning pre-trained CNNs versus training from scratch across four medical-imaging applications, covering classification, detection, and segmentation tasks. The authors concluded that fine-tuning pre-trained CNNs generally matches the performance of CNNs trained from scratch and even outperforms them with limited training data. Additionally, they emphasized the robustness of fine-tuned models to smaller training set sizes and the efficacy of layer-wise fine-tuning, which focuses on adjusting the top layers of the network first. The study also found that the optimal fine-tuning strategy, shallow versus deep, varies depending on the specific application and data characteristics. Expanding on the concept of fine-tuning, ref. [52] proposed a method for adaptive fine-tuning in transfer learning. Their approach enhances transfer learning efficiency by dynamically selecting specific layers of a network to fine-tune based on every given data example, thereby optimizing the results.

Transfer learning and domain adaptation in computer vision applications were examined in the comprehensive survey in [53]. The paper distinguished transfer learning, which applies knowledge from one domain to another, from domain adaptation, which specifically addresses shifts in data distributions between source and target domains. It categorized approaches into homogeneous (same feature space) and heterogeneous (different feature spaces) methods, and reviewed both traditional shallow techniques and modern deep learning methods. The authors of [54] addressed the challenge of domain adaptation by defining bounds on the generalization error of classifiers when transferring knowledge from a source domain to a target domain and finding a common representation between the source and target domains. The work in [55] focused on cases where labeled data are scarce. The proposed method of structural correspondence learning adapts a classifier trained on a source domain with labeled data to a target domain with no labeled data, by learning a shared feature space from both domains using pivot features. This transformation allows the classifier to use both the original and transformed features for better performance on the target domain.

Transfer learning addresses the challenge of domain discrepancies in machine learning, where training and testing data often come from different distributions, limiting model generalization. The recent survey in [56] provided a comprehensive review of transfer learning approaches, categorizing them based on the number of source domains, the degree of supervision, and techniques for handling incomplete data. It highlighted key methodologies such as instance-based, feature-based, and parameter-based transfer learning, emphasizing their effectiveness in adapting models across domains. Beyond general transfer learning, [57] explored a hybrid CNN–LSTM architecture that leverages transfer learning for deepfake detection. By combining the spatial feature extraction of CNNs with the sequential modeling capabilities of LSTMs, the approach enhances detection accuracy on deepfake datasets. Similarly, a three-stage artificial rabbits optimization with transfer learning technique, employing a modified DarkNet-53 model for feature extraction and optimizing hyperparameters, was proposed to improve deepfake detection in biometric applications [58].

Recent research on transformer compression has yielded promising techniques, such as DistilBERT [59] and MobileBERT [60]. However, within the context of ambient intelligence and edge computing, CNN-based architectures continue to hold a practical advantage, due to their inherently smaller model sizes and computational efficiency [61–63]. As smart environments depend on resource-constrained edge devices, selecting models that balance accuracy and efficiency is critical.

Although vision transformers have demonstrated competitive performance in various computer vision tasks [64,65], their dominance over CNNs in DeepFake detection remains uncertain. In particular, research by [66] highlighted that compressed transformer models still require higher computational resources compared to CNNs, making them less practical for real-time deployment in intelligent systems. Furthermore, CNNs benefit from well-established compression techniques, including pruning, quantization, and knowledge distillation [45,67], which have been extensively refined for edge computing scenarios. In contrast, transformer architectures are still undergoing optimization for these constraints, limiting their immediate applicability to real-world ambient intelligence applications. Given these considerations, our approach prioritizes CNN-based models for DeepFake detection, ensuring efficient operation in smart environments. As noted by [5], the integration of effective detection technology into ambient intelligence systems is crucial for maintaining secure and trustworthy automated processes. By leveraging CNNs, we can enhance the reliability of smart surveillance, IoT security, and real-time anomaly detection, while maintaining computational efficiency in edge-based environments.

3. Compression of DeepFake Models

Model compression in machine learning encompasses a suite of techniques aimed at reducing the size and computational complexity of models, thereby enhancing their efficiency in terms of storage, memory usage, and computational resources. In the following subsections, the four major approaches to model compression will be examined in detail.

Before delving into the details, a summary of the models used in this study is presented. This study primarily focuses on a single baseline model for comparison, but also employs several student models with similar, smaller architectures for knowledge distillation. The uncompressed baseline DeepFake detection model, serving as the teacher or unpruned model, is a VGG-based architecture with approximately 4.5 million parameters, drawing inspiration from [68]. The student models follow a similar architecture. Table 1 provides a concise overview of the models. A detailed analysis of their performance, including ablation studies and comparisons across various metrics and datasets, is presented in Sections 5 and 8.

Table 1. Baseline and student models.

Model	Architecture	Layers	Parameter Count
Baseline/Teacher Model	VGG-based	8 Conv layers 3 FC layers	∼4.5 Million
Student Model 40%	VGG-based	7 Conv layers 3 FC layers	∼1.8 Million
Student Model 30%	VGG-based	6 Conv layers 2 FC layers	∼1.34 Million
Student Model 20%	VGG-based	7 Conv layers 2 FC layers	∼0.9 Million
Student Model 10%	VGG-based	5 Depthwise Separable Conv layers 2 FC layers	∼0.44 Million

3.1. Pruning

Pruning starts with a fully trained CNN, which is first trained to establish baseline performance on a DeepFake detection task. Then, L1-norm unstructured local pruning is applied, where parameters are selectively removed at predefined percentages based on their L1-norm values, as given in Equation (1).

$$||W||_1 = \sum_{i,j,\ldots} |W_{i,j,\ldots}| \tag{1}$$

where W is the tensor to be pruned, $||W||_1$ is defined as the sum of the absolute values of the weights, and $W_{i,j,\ldots}$ are the individual elements (weights) in the tensor. As shown in Figure 1, an equal number of parameters are removed from each layer and each filter of the given network following the L1-norm, which characterizes the method as local.

The method is also termed unstructured, because it involves the removal of individual weights within layers rather than entire filters, as depicted in Figure 1, and thus the L-1 norm is calculated separately for each weight. This selective pruning method reduces the model complexity, while retaining critical features. The most widely used approach is post-fine-tuning which uses a pruning mask, which is a binary tensor to indicate which parameters in a neural network should be pruned and which should remain active. The fine-tuning process is conducted after the network pruning to assess the performance recovery potential following parameter reduction.

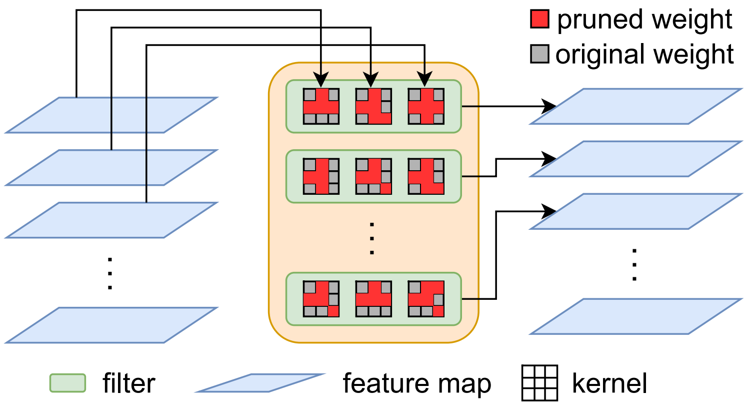

Figure 1. Pruning of convolutional neural networks.

3.2. *Knowledge Distillation*

Knowledge distillation is a model compression technique that involves transferring knowledge from a larger, pre-trained model (teacher) to a smaller model (student) to achieve similar performance with reduced computational complexity.

Response-based offline distillation is a method where the teacher model's outputs are used to train the student model, without updating the teacher's parameters. As illustrated in Figure 2, the distillation process begins by feeding the same data into both the teacher and student models, which then produce the logits of their predictions. These logits are used to calculate the loss, which is a weighted combination of two components: the cross-entropy loss and the Kullback–Leibler (KL) divergence loss.

The cross-entropy loss is calculated using the raw logits from the student model and the actual hard labels, as given in Equation (2)

$$H(p,q) = -\sum_{x\in C} p(x)\log q(x) \tag{2}$$

where C is the number of classes, $p(x)$ the true probability distribution, and $q(x)$ the student model's predicted probability distribution. The KL divergence loss, on the other hand, as shown in Figure 2, is computed using soft probabilities and soft targets, as given in Equation (3).

$$D_{KL}(z||q) = \sum_{x\in C} z(x)(\log z(x) - \log q(x)) \tag{3}$$

where C is the number of classes, $z(x)$ the teacher model's probability distribution, and $q(x)$ the student model's predicted probability distribution. The soft probabilities are obtained by applying the softmax function to the student model's logits, followed by a log function to soften the probabilities. The soft targets are generated similarly, using the logits from the teacher model. This process facilitates knowledge transfer from the teacher to the student model via the loss function.

Before calculating the soft labels and probabilities, the logits are scaled using a temperature parameter T, set to 2, which controls the sharpness or smoothness of the resulting probability distributions. The KL divergence loss is also scaled by the square of the temperature (T^2) before it is incorporated into the weighted loss. This temperature scaling approach was recommended by the authors of the paper [69].

Lastly, the weights assigned to the cross-entropy loss and the KL divergence loss for the weighted combination of the total loss are given in Equation (4).

$$L_{total} = \frac{1}{T^2}(w\, D_{KL}(z||q) + (1-w)\, H(p,q)) \tag{4}$$

where T is the temperature for scaling, w the weight for the KL divergence loss, and $(1-w)$ the weight for the cross-entropy loss, dictating the importance attributed to each individual loss component.

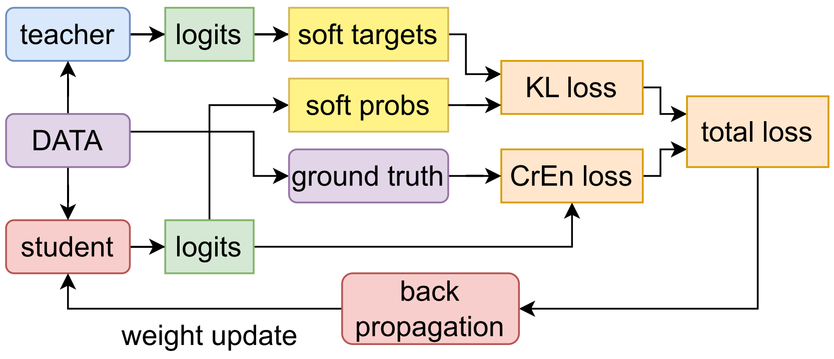

Figure 2. Knowledge distillation in the teacher–student framework.

3.3. Quantization

Quantization is a model compression technique aimed at reducing the size and computational requirements of neural networks, particularly focusing on CNNs in DeepFake detection. In this method, an original model can be utilized, trained for DeepFake detection as described in the previous sections and specifically applying quantization to its linear layers. This process entails reducing the parameter precision of these layers, from high-precision to low-precision. Figure 3 illustrates the quantization function, which converts the weights of fully connected (FC) layers from float32 to int8. Given a floating-point value x of a weight, the quantized value x_q is calculated based on Equation (5).

$$x_q = round(\frac{x - x_{min}}{\Delta})\Delta + x_{min} \tag{5}$$

where x_{min} is the minimum possible value of x, x_{max} is the maximum possible value of x, the *round* function rounds the scaled value to the nearest integer, and Δ is the quantization step size, calculated based on Equation (6).

$$\Delta = \frac{X_{max} - x_{min}}{2^b - 1} \tag{6}$$

where b is the number of bits used for quantization.

The main constraint on quantization is the notable limitation of GPU operability. This constraint not only impedes potential performance enhancements in contexts where GPU acceleration is pivotal but also carries implications for the utilization of specialized hardware [9]. Furthermore, an additional obstacle can be highlighted as the unquantizability of the model's convolutional layers, which obstructs the overarching objective of reducing the model size. This happens due to the lack of support for dynamic quantization in the convolutional layers in the available programming frameworks like Pytorch 2.6 quantization (https://pytorch.org/docs/stable/quantization.html (accessed on 2 March 2025)). Consequently, the compression of DeepFake models that can be achieved is limited, as only the linear layers benefit from reduced precision. Despite these challenges, the quantization method is still worth investigation.

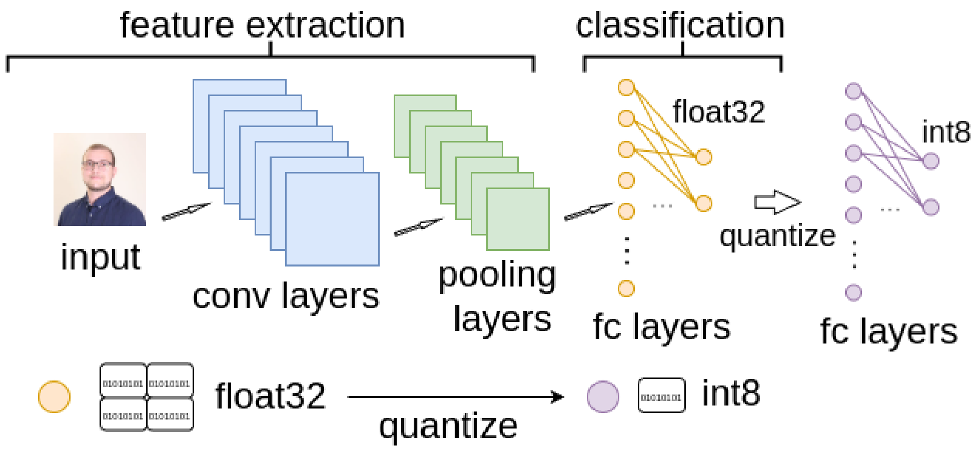

Figure 3. Quantization of deep neural network parameters.

3.4. Low-Rank Factorization

Low-rank factorization is a compression technique that approximates large weight matrices by decomposing them into the product of smaller matrices, significantly reducing the number of parameters. This process involves representing an original weight matrix W of size $m \times n$ as the product of two smaller matrices F and Z, where F is $m \times k$ and Z is $k \times n$, with k much smaller than m and n.

Low-rank factorization can be applied in a DeepFake detection model by using the Singular Value Decomposition (SVD) method to decompose the first linear layer, which is the largest in terms of parameter count. As shown in [70], this process involves representing an original weight matrix W of size $m \times n$ as given in Equation (7).

$$W = USV^T, U \in \mathbb{R}^{m\times m},\ S \in \mathbb{R}^{m\times n},\ V \in \mathbb{R}^{n\times n} \tag{7}$$

In order to reduce size the original weight matrix, W is approximated by keeping the k most significant singular vectors, as given in Equation (8).

$$\hat{W} = \hat{U}\hat{S}\hat{V}^T,\ \hat{U} \in \mathbb{R}^{m\times k},\ \hat{S} \in \mathbb{R}^{k\times k},\ \hat{V} \in \mathbb{R}^{n\times k} \tag{8}$$

The low-rank factorization method replaces the original large linear layer with two smaller layers, resulting from $\hat{U}$, $\hat{S}$, $\hat{V}$, with fewer parameters in total. These two layers, when multiplied together in a specific manner, approximate the function of the original layer. The new layers replace the original, as illustrated in Figure 4, where the bias of the original layer is assigned to the second new layer, and the first new layer is left without a bias.

To integrate these changes, a new model architecture is designed that accommodates the two new layers in place of the original large linear layer, and then a revised model is instantiated. Then, all parameter weights from the original model are transferred to the new one, and the two new linear layers are also populated with weights derived from the factorization results. Although this approach results in a reduced parameter count, the majority of the layers of a CNN model are convolutional and, as a consequence, not amenable to factorization, resulting in less prospective compression overall. Nevertheless, there may be specific instances where low-rank factorization could be beneficial, particularly in models where linear layers dominate and where the impact on performance is more manageable.

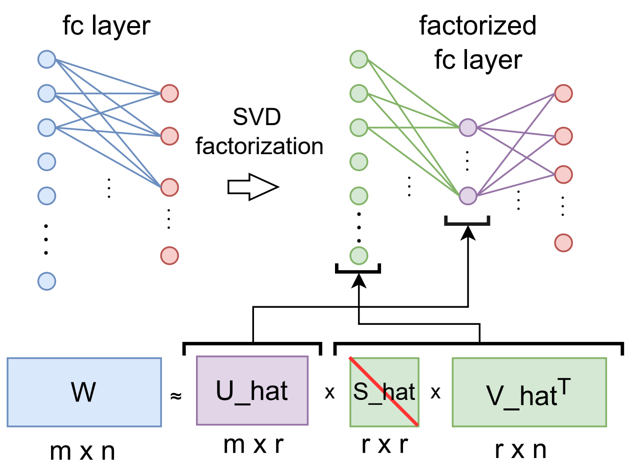

Figure 4. Low-rank factorization.

4. Transfer Learning in DeepFake Models

Transfer learning leverages pre-trained models on large-scale datasets for new, often smaller and more specific, tasks. The primary advantage of transfer learning lies in the ability to utilize the knowledge acquired by a model during its initial training phase on a broad and extensive dataset. This pre-existing knowledge can then be transferred to improve performance and accelerate learning on a new task that typically has less data available for training. By using pre-trained models as a starting point, transfer learning can significantly reduce the computational resources and time required to build effective neural networks for specialized applications.

In practice, transfer learning often involves fine-tuning a pre-trained model on a new dataset, allowing the model to adapt its learned features to the specifics of the new task. By leveraging models pre-trained on extensive visual datasets, developers can fine-tune these models to detect DeepFakes with relatively less data and computational effort. This transfer of knowledge not only enhances the efficiency of model training but also ensures that the detection algorithms achieve a high level of accuracy, which is essential for the intended applications. The following subsections discuss various approaches for applying fine-tuning to the compression techniques presented in the previous section. Additionally, the role of adapters in the context of pruning and knowledge distillation is explored.

4.1. Pruning with Fine-Tuning

Pruning with fine-tuning leverages a pre-trained deep learning model by first applying pruning to reduce its parameter count, followed by fine-tuning, as shown in Figure 5. Initially, the model is trained on a dataset from task A, learning general feature representations relevant to the source domain. After this pre-training phase, pruning removes redundant or less significant parameters, improving the model's efficiency. However, since the pruned model will work on a different task B, fine-tuning is performed using a dataset generated by task B to adapt the pruned model. During fine-tuning, most of the network's layers remain frozen, to transfer knowledge from the task A, while selected layers—typically higher-level ones—are re-trained to specialize on the task B.

This approach ensures that the model benefits from transfer learning, without excessive weight modifications that could cause catastrophic forgetting. Catastrophic forgetting is a phenomenon in deep learning where a neural network loses previously learned knowledge when trained on new data, as the weight updates for the new task overwrite important information from earlier tasks. Furthermore, a lower learning rate is used to stabilize updates, allowing gradual adaptation, while maintaining the efficiency gains from pruning. Additionally, the pruning mask is retained throughout fine-tuning, to enforce sparsity constraints, preventing pruned connections from being restored. Pruning masks are binary tensors used to indicate which parameters in a neural network should be pruned and which should remain active. By combining pruning with fine-tuning, deep learning models can be efficiently transferred to new datasets, while maintaining high performance and reduced computational complexity, making them well-suited for resource-constrained applications such as DeepFake detection.

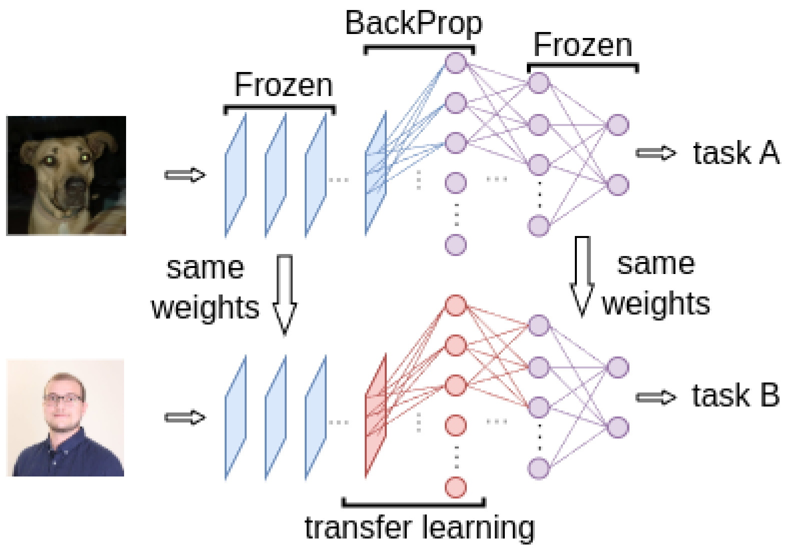

Figure 5. Transfer learning across different tasks.

4.2. Pruning with Fine-Tuning and Adapters

In this section, the pruning with fine-tuning approach for transfer learning is presented, which leverages adapters to enhance the performance of pruned CNN models. This method is inspired by the growing use of adapters in transformers, as highlighted in [15,71,72], a trend that has shown considerable promise for improving model performance, with minimal additional complexity.

Adapters serve as an additional layer within the DeepFake model architecture, as shown in Figure 6, and are specifically designed to introduce minimal complexity, while providing some performance gains. These adapters are strategically placed between hidden layers. This optimal placement likely relates to the complexity of the features detected at this stage in the network. The adapter layer, by intervening at certain points, can effectively balance between detecting very simple and highly complex features, thus providing significant performance improvements.

The adapters, as demonstrated in Figure 6, consist of four components: two convolution operations (a depthwise convolution and a pointwise convolution, which together form a depthwise separable convolution), a batch normalization, and a ReLU activation function. This composition is designed to ensure the adapter layer remains lightweight yet effective from a feature extraction perspective.

The pruning with fine-tuning and adapters approach begins by creating a new model instance capable of accommodating the adapter layer. For a pruned DeepFake model, a new instance is instantiated and the weights of the original layers are transferred to this new model. The original layers are frozen, meaning their parameters are not updated during the training of the adapters. Notably, as the pruned layers are not subjected to further training, modification and use of the pruning masks of the original pruned model instances is not needed, significantly reducing the method's complexity. This represents a subtle but crucial difference compared to other techniques previously discussed that require the training of pruned models.

The training is focused exclusively on the adapter layer. This selective training strategy ensures that the overall model remains compact and does not demand extensive additional training resources. The resulting model, when enhanced with adapters, can achieve improved performance, while preserving its compact size.

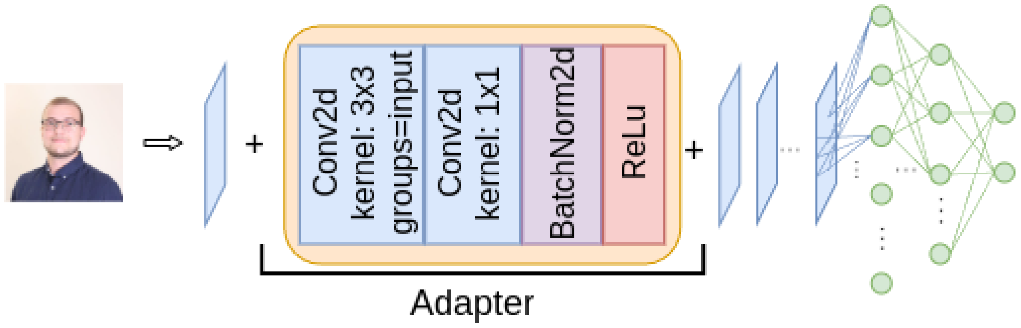

Figure 6. CNN with adapter module for transfer learning.

4.3. *Knowledge Distillation Using Data from a New Task*

In this approach, a pre-trained DeepFake model trained in task A serves as the teacher, gradually distilling knowledge into a smaller student model.This distillation process differs from the one described in Section 3.2, as it leverages a dataset from task B, which is separate from the dataset used to originally train the teacher model, as illustrated in Figure 7. Recognizing that the teacher model may exhibit lower performance on the dataset from task B, the weighting of the loss components in the distillation process are adjusted accordingly. This adjustment ensures greater emphasis on the loss derived from the actual labels.

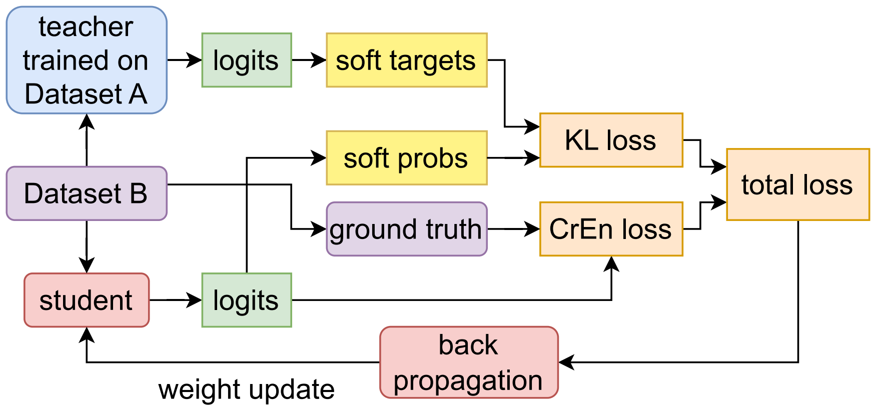

Figure 7. Knowledge distillation for transfer learning.

4.4. *Knowledge Distillation with Adapter Using Data from a New Task*

In Section 4.3, the use of knowledge distillation for transfer learning was discussed. Here, the distilled DeepFake models are enhanced by incorporating adapters, similarly to those used in pruned models, to improve the performance of the compressed model in the new task. These adapters consist of depthwise separable convolution (a combination of depthwise and pointwise convolutions), followed by batch normalization and ReLU activation, as shown in Figure 6. The adapter placement differs between pruned models and distilled models, since shallower models benefit more from the detection of features with different complexity. New model instances are created to accommodate the adapter layer, transferring and freezing the original weights to ensure only the adapter layer is trained, enhancing the DeepFake detection without altering existing knowledge. The result is slightly larger, yet still compact models with improved performance.

4.5. Quantization with Fine-Tuning

Quantization with fine-tuning enables effective transfer learning by first applying a selective knowledge transfer and then compressing the model for improved efficiency. The process begins with a pre-trained model, where a knowledge transfer is performed by fine-tuning only specific layers, while keeping the rest frozen. As illustrated in Figure 5, this approach focuses on updating the last convolutional and first linear layers using data from the task B, allowing the model to adapt to a different domain, while preserving previously learned features from the task A. Once the fine-tuning phase has enhanced the model's performance on the new data, quantization is applied to further optimize its efficiency. As described in Section 3.3 and shown in Figure 3, the quantization process reduces the precision of linear layers from float32 to int8, significantly decreasing the model size and computational requirements. However, this compression technique also introduces challenges in terms of hardware compatibility and inference speed, similarly to those discussed in Section 3.3. Despite these limitations, quantization with fine-tuning can be an effective strategy for transfer learning.

4.6. Low-Rank Factorization with Fine-Tuning

Transfer learning with low-rank factorization and fine-tuning integrates knowledge transfer and model compression to improve efficiency while adapting to a new task. The process starts with a pre-trained DeepFake model, where adaptation is achieved by fine-tuning selected layers using data from the task B, while keeping the rest of the network frozen. This targeted fine-tuning enables the model to learn environment-specific features, while retaining crucial knowledge acquired from the task A, ensuring a balance between adaptation and efficiency. Following the fine-tuning phase, low-rank factorization is applied to compress the model by decomposing weight matrices into lower-rank approximations, effectively reducing the number of parameters, while maintaining an approximation of the original functionality. This compression step enhances computational efficiency, reducing both memory usage and inference time. However, as highlighted in Section 3.4, this approach often leads to a notable performance drop, indicating potential limitations in its ability to generalize across different tasks.

5. Experimental Evaluation

This section presents the experimental evaluation of the compression and transfer learning approaches discussed in Sections 3 and 4. It outlines the datasets used, the evaluation metrics applied, and the experimental results, along with the key conclusions drawn from the analysis.

5.1. Datasets

For the experiments, five different datasets were used.

The first dataset comprised two distinct sub-datasets. The synthetic images were sourced from the test set of the "Synthbuster" dataset (https://zenodo.org/records/10066460 (accessed on 2 March 2025)) provided from the paper [73]. This set includes 9000 AI-generated images from nine different models, with 1000 images generated by each model. The models utilized are DALL·E 2, DALL·E 3, Adobe Firefly, Midjourney v5, Stable Diffusion 1.3, Stable Diffusion 1.4, Stable Diffusion 2, Stable Diffusion XL, and Glide. The categories of the images are indoor, outdoor, landscapes, people, objects, and buildings. These synthetic images were paired with authentic images from the "RAISE" dataset (http:

//loki.disi.unitn.it/RAISE/download.html (accessed on 2 March 2025)), which consists of 8153 high-resolution images. Both the DeepFake and real images were resized to 224 × 224 pixels to meet the desired dimensions. Subsequently, the dataset was split into a training set containing 60% of the images and a test set comprising the remaining 40%.

The second dataset was the "140k Real and Fake Faces" dataset (https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces (accessed on 2 March 2025)). This dataset comprises a total of 140,000 images. Of these, 100,000 images are allocated for training purposes, 20,000 images for validation, and the remaining 20,000 images for testing. Each image in the dataset has a resolution of 256 by 256 pixels.

The third dataset was "DeepFake and real images", which also pertains to real and DeepFake human photos (https://www.kaggle.com/datasets/manjilkarki/DeepFake-and-real-images (accessed on 2 March 2025)). The dataset consists of 190,335 images in total, with 140,002 images designated for training, 39,428 for validation, and 10,905 for testing. Each image has a resolution of 256 × 256 pixels.

The fourth dataset was named "ForenSynths" and provided by the authors of [74]. This dataset (https://github.com/PeterWang512/CNNDetection (accessed on 2 March 2025)) includes both DeepFake and real images of various types, with all DeepFake images generated through GANs. The dataset is divided into a training set, a validation set, and a test set. The training set contains a total of 720,119 images, with all DeepFake images produced by a single GAN. Similarly, the validation set comprises 8000 images, all generated by the same GAN. The test set, however, consists of 90,310 images, with DeepFakes produced by multiple different GANs. In the experiments, the images of human faces only were separated in order to directly compare the results to previous datasets. The images generated by each GAN have varying resolutions; therefore, for the purposes of this work, all images were resized to 224 × 224 pixels.

Additionally, a fifth dataset was utilized to evaluate transfer learning, which focused on a task distinct from DeepFake detection. Specifically, this dataset is designed for distinguishing between cats and dogs. An example pair of images can be seen below in Figure 8. In the transfer learning framework illustrated in Figure 5, this served as task A, while DeepFake detection corresponded to task B. This dataset is called dogs vs. cats, obtained from Kaggle and consists of images of cats and dogs for the task of binary classification (https://www.kaggle.com/datasets/salader/dogs-vs-cats (accessed on 2 March 2025)). This dataset is considerably smaller than the previous two and contains a total of 25,000 images. Specifically, 20,000 images are utilized for training and 5000 for testing. The sizes of the images vary, so a pixel resolution cannot be specified.

Figure 8. "Dogs vs. cats" dataset example.

5.2. Experimental Setup

The experimental setup for this study utilized a combination of Python 3.13 libraries and robust hardware to ensure optimal performance and efficiency. The main Python libraries employed included PyTorch for building and training the neural network models, PIL (Python Imaging Library) for image processing, the time module for tracking experiment duration, psutil for monitoring system and hardware resources, and scikit-learn for additional machine learning tasks and metrics. All experiments were conducted on Kaggle, a platform that offers free access to notebooks and accelerators such as GPUs and TPUs, and that is specifically designed for machine learning purposes. The experiments' source code is available for any kind of reproduction and reexamination in the first author's GitHub repository [75].

The specific hardware setup comprised two NVIDIA (Santa Clara, CA, USA) T4 GPUs provided by Kaggle. Each GPU was a TU104-895 model with the NVIDIA Turing architecture, operating at a base clock speed of 585 MHz. The GPUs included 2560 CUDA cores, delivering a peak FP32 performance of 8.1 TFLOPS and INT8 performance of 130 TOPS. Each GPU was equipped with 16 GB of GDDR6 memory, with a maximum memory clock speed of 5001 MHz. To fully leverage the computational power and memory capacity of both GPUs, the DataParallel container from the PyTorch framework was employed. This allowed the implementation of data parallelism at the module level, ensuring efficient utilization of the hardware resources and significantly enhancing the performance and speed of the model training processes.

The uncompressed DeepFake detection model, serving as the teacher or unpruned model, was a VGG-based architecture with approximately 4.5 million parameters [68]. It was trained on the datasets introduced in Section 5.1 and served as the baseline model for applying compression and transfer learning, which will be discussed in Section 5.4.

5.3. Evaluation Metrics

The evaluation metrics used were accuracy, precision, recall, and F1 score. These metrics quantified the models' ability to correctly identify DeepFakes and genuine images, as well as their robustness to false positives and false negatives. Additionally, training time was included as a metric to assess the computational efficiency of the models.

5.4. Outcomes and Discussion

This subsection presents a comprehensive analysis of the experimental outcomes obtained from applying various compression and transfer learning approaches to the DeepFake detection models. Section 5.4.1 investigates how different types of synthetic image generators and image variations influenced the compression effectiveness. In Section 5.4.2, the impact of compression and transfer learning is evaluated across the models trained and fine-tuned on distinct DeepFake datasets. Section 5.4.3 focuses on the effectiveness of compression when using datasets generated by multiple DeepFake models. Finally, Section 5.4.4 explores the potential of transfer learning between models trained on different dataset types, analyzing the benefits of knowledge distillation and pruning for improved cross-domain DeepFake detection.

5.4.1. Experimental Evaluation of Compression Using a DeepFake Detection Model Trained on Multiple Synthetic Image Types and Multiple DeepFake Generators

In the first set of experiments, an image synthetic detection model was trained using the "Synthbuster" and the "RAISE" datasets. The "Synthbuster" dataset contains synthetic images generated by multiple synthetic generator models. The experimental results, summarized in Table 2, indicated that models compressed through knowledge distillation and pruning—despite being reduced to just 10% of their original size—experienced less than a 1% drop in accuracy and F1 score. These results can also be verified in Figure 9, where some sample ROC curves are presented. When applying quantization, a 0.02% reduction in accuracy and F1 score was observed, while achieving only 28% compression of the original model size. The primary advantage of quantization is its instantaneous nature, as it directly converts model weights from float32 to int8. In contrast, knowledge distillation and post-pruning fine-tuning involve iterative processes that require significantly more time, depending on the extent of compression. Overall, the detection performance remained high, reaching approximately 92%, even with a diverse dataset that included various image types from multiple DeepFake generators. Notably, when the model was tested exclusively on human face images generated by a single DeepFake model, its performance improved significantly, as discussed in the next subsection.

Table 2. Compression using multiple synthetic image types and DeepFake generator models on "Synthbuster" and "RAISE" datasets.

Method	Full Model	40% Parameters	30% Parameters	20% Parameters	10% Parameters
Pruning	Precision: 0.9420 Recall: 0.9169 F-1 score: 0.9293 Accuracy: 0.9265 Train Time: 859 s	Precision: 0.9070 Recall: 0.9402 F-1 score: 0.9233 Accuracy: 0.9177 Train Time: 491 s	Precision: 0.8964 Recall: 0.9358 F-1 score: 0.9157 Accuracy: 0.9092 Train Time: 491 s	Precision: 0.9147 Recall: 0.9325 F-1 score: 0.9235 Accuracy: 0.9186 Train Time: 490 s	Precision: 0.9272 Recall: 0.9138 F-1 score: 0.9205 Accuracy: 0.9168 Train Time: 488 s
Knowledge Distillation	Precision: 0.9420 Recall: 0.9169 F-1 score: 0.9293 Accuracy: 0.9265 Train Time: 859 s	Precision: 0.9298 Recall: 0.9311 F-1 score: 0.9304 Accuracy: 0.9266 Train Time: 599 s	Precision: 0.9366 Recall: 0.9286 F-1 score: 0.9326 Accuracy: 0.9293 Train Time: 586 s	Precision: 0.9275 Recall: 0.9347 F-1 score: 0.9311 Accuracy: 0.9271 Train Time: 586 s	Precision: 0.9216 Recall: 0.9241 F-1 score: 0.9228 Accuracy: 0.9186 Train Time: 582 s
		Original Model		**Compressed Model**	
Quantization		Original size: 18.41 Mb Precision: 0.9420 Recall: 0.9169 F-1 score: 0.9293 Accuracy: 0.9265 Train Time: 859 s CPU inference time on 6833 images: 1979 s		Quantized size: 13.39 Mb Precision: 0.9420 Recall: 0.9166 F-1 score: 0.9291 Accuracy: 0.9263 CPU inference time for 6833 images: 1656 s	

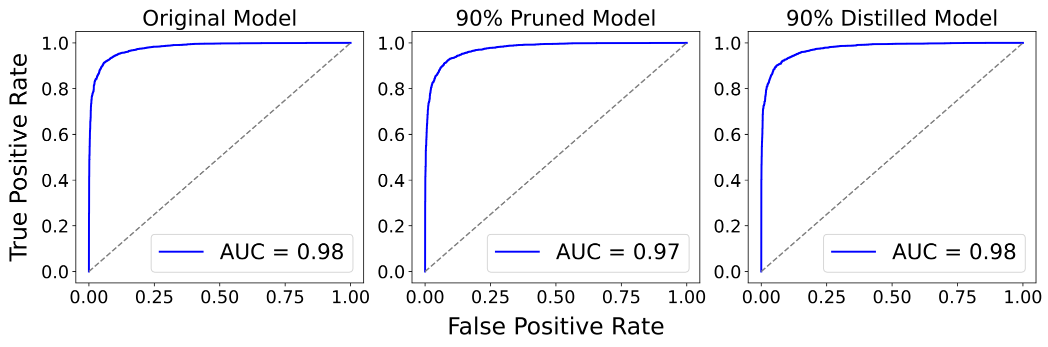

Figure 9. Sample ROC curves for Synthbuster dataset.

5.4.2. Experimental Evaluation of Compression and Transfer Learning Between Models Trained and Fine-Tuned on Different DeepFake Datasets

The results of the compression on the "140k Real and Fake Faces" dataset are presented in Table 3. In this set of experiments, each method was applied separately, without additional processing such as fine-tuning. These experiments were conducted to assess the performance of each method when applied independently. From the results, it can be observed that the quantization method maintained a high performance but achieved limited compression, close to 72%. Conversely, pruning exhibited significantly higher compression capabilities, although the performance deteriorated as the compression percentage increased. Notably, knowledge distillation demonstrated substantial superiority compared to the other methods. It exhibited excellent performance, achieving both high compression rates and high accuracy.

Table 3. Compression using a model trained on a single type of synthetic images and one DeepFake generator model on the "140k Real and Fake Faces" dataset.

Method	Full Model	40% Parameters	30% Parameters	20% Parameters	10% Parameters
Pruning	Precision: 0.9878 Recall: 0.9914 F-1 score: 0.9896 Accuracy: 0.9896 Train Time: 4382 s	Precision: 0.9671 Recall: 0.9507 F-1 score: 0.9588 Accuracy: 0.9592	Precision: 0.9296 Recall: 0.9141 F-1 score: 0.9217 Accuracy: 0.9224	Precision: 0.8577 Recall: 0.8371 F-1 score: 0.8473 Accuracy: 0.8491	Precision: 0.9418 Recall: 0.1134 F-1 score: 0.2024 Accuracy: 0.5532
Knowledge Distillation	Precision: 0.9878 Recall: 0.9914 F-1 score: 0.9896 Accuracy: 0.9896 Train Time: 4382 s	Precision: 0.9899 Recall: 0.9721 F-1 score: 0.9809 Accuracy: 0.9811 Train Time: 1862 s	Precision: 0.9845 Recall: 0.981 F-1 score: 0.9827 Accuracy = 0.9828 Train Time: 1825 s	Precision: 0.9928 Recall: 0.974 F-1 score: 0.9833 Accuracy: 0.9835 Train Time: 1913 s	Precision: 0.9856 Recall: 0.969 F-1 score: 0.9772 Accuracy: 0.9774 Train Time: 3195 s
	Original Model			**Compressed Model**	
Quantization	Size: 17.6 Mb Precision: 0.9878 Recall: 0.9914 F-1 score: 0.9896 Accuracy: 0.9896 CPU inference on 20,000 images: 5378 s			Size: 12.8 Mb Precision: 0.9875 Recall: 0.9914 F-1 score: 0.9894 Accuracy: 0.9894 CPU inference on 20,000 images: 3918 s	

In Table 4, the outcome of the next set of experiments is presented, related to knowledge transfer across models trained and fine-tuned with two different deep fake detection datasets. These experiments involved fine-tuning the pruned models derived from the "140k Real and Fake Faces" dataset with the images from the "DeepFake and real images" dataset. This fine-tuning significantly enhanced the models' performance on the evaluation with the "DeepFake and real images" dataset, while maintaining compression at the same levels. The knowledge distillation method once again emerged as the most effective approach, achieving a superior performance. Quantization with fine-tuning was implemented, as described in Section 4.5, resulting in negligible performance degradation; however, the level of compression remained insufficient. The proposed method of employing adapters on pruned and distilled models demonstrated improved performance in most instances, indicating its potential applicability. Various configurations of adapter placements were tested, leading to the conclusion that optimal results vary depending on model size and compression technique. This variability in performance depended on the adapter placement, as can be seen in the row "Pruning + adapter after last Conv2d layer" and the row "Pruning + adapter after fourth Conv2d layer" of Table 4. In these two experiments, the observation can be made that the two different adapter placements in the pruned models achieved unequal gains in performance. Ultimately, knowledge distillation, with or without the addition of adapters, proved to be the superior method compared to the other approaches.

Table 4. Transfer learning from a model trained on the "140k Real and Fake Faces" dataset to a model fine-tuned and evaluated on the "DeepFake and Real Images" dataset.

Method	40% Parameters	30% Parameters	20% Parameters	10% Parameters
Pruning + transfer	Precision: 0.8535 Recall: 0.8860 F-1 score: 0.8694 Accuracy: 0.8660 Train Time: 1881 s	Precision: 0.8477 Recall: 0.8800 F-1 score: 0.8635 Accuracy: 0.8599 Train Time: 1789 s	Precision: 0.8303 Recall: 0.8687 F-1 score: 0.8490 Accuracy: 0.8444 Train Time: 1790 s	Precision: 0.8116 Recall: 0.8443 F-1 score: 0.8276 Accuracy: 0.8229 Train Time: 1795 s
Knowledge Distillation	Precision: 0.9467 Recall: 0.9198 F-1 score: 0.9331 Accuracy: 0.9336 Train Time: 2715 s	Precision: 0.9537 Recall: 0.8867 F-1 score: 0.9190 Accuracy: 0.9213 Train Time: 2675 s	Precision: 0.9392 Recall: 0.9146 F-1 score: 0.9267 Accuracy: 0.9271 Train Time: 2583 s	Precision: 0.9534 Recall: 0.7605 F-1 score: 0.8461 Accuracy: 0.8607 Train Time: 2559 s
KD + adapter after last Conv2d layer	Precision: 0.9179 Recall: 0.9431 F-1 score: 0.9303 Accuracy: 0.9289 Train time: 1332 s	Precision: 0.9329 Recall: 0.9304 F-1 score: 0.9317 Accuracy: 0.9313 Train time: 1187 s	Precision: 0.9105 Recall: 0.9439 F-1 recall: 0.9269 Accuracy: 0.9250 Train time: 1183 s	Precision: 0.8845 Recall: 0.8872 F-1 score: 0.8859 Accuracy: 0.8849 Train time: 1188 s
Pruning + adapter after last Conv2d layer	Precision: 0.8633 Recall: 0.8801 F-1 score: 0.8716 Accuracy: 0.8695 Train time: 1865 s	Precision: 0.8582 Recall: 0.8831 F-1 score: 0.8705 Accuracy: 0.8676 Train time: 1918 s	Precision: 0.8540 Recall: 0.8461 F-1 score: 0.8500 Accuracy: 0.8497 Train time: 1912 s	Precision: 0.8424 Recall: 0.8102 F-1 score: 0.8260 Accuracy: 0.8281 Train time: 1890 s
Pruning + adapter after fourth Conv2d layer	Precision: 0.8748 Recall: 0.8845 F-1 score: 0.8796 Accuracy: 0.8781 Train time: 2415 s	Precision: 0.8720 Recall: 0.9011 F-1 score: 0.8863 Accuracy: 0.8836 Train time: 2462 s	Precision: 0.8861 Recall: 0.8617 F-1 score: 0.8738 Accuracy: 0.8746 Train time: 2458 s	Precision: 8745 Recall: 0.8343 F-1 score: 0.8539 Accuracy: 0.8563 Train time: 2415 s
	Transfered Model		**Compressed Model**	
Quantization	Size: 17.6 Mb Precision: 0.8526 Recall: 0.8996 F-1 score: 0.8755 Accuracy: 0.8711 Transfer time: 1959 s CPU inference on 10905 images: 2949 s		Size: 12.8 Mb Precision: 0.8523 Recall: 0.8998 F-1 score: 0.8754 Accuracy: 0.8710 CPU inference on 10905 images: 3036 s	

5.4.3. Experimental Evaluation of Compression Using a Dataset Generated by Multiple Types of DeepFake Models

The following experiments focused on human face images from the "ForenSynths" dataset, which were generated using multiple DeepFake generator models. Given that these images were produced by different AI models, each model's output was evaluated separately. The experimental results are summarized in Table 5. The DeepFake detection model used in this study was trained on images generated by ProGAN, which explains its high performance when evaluated on the ProGAN testing set. However, when tested on images produced by DeepFake, StarGAN, and WhichFaceIsReal, the model's performance dropped significantly. Notably, the performance decline was less severe on the StarGAN dataset compared to DeepFake and WhichFaceIsReal. This can be attributed to the fact that both ProGAN and StarGAN are GAN-based generators, allowing the model trained on one to partially generalize to the other.

Table 5. Compression using a model trained on data from a single DeepFake generator and evaluated on data from multiple DeepFake models using the "ForenSynths" dataset.

Method	GAN Type	Full Model	40% Parameters	30% Parameters	20% Parameters	10% Parameters
Pruning	**DeepFake**	Precision: 0.4909 Recall: 0.3690 F-1 score: 0.4213 Accuracy: 0.4923 Train Time: 2236 s	Precision: 0.5038 Recall: 0.4536 F-1 score: 0.4774 Accuracy: 0.5026 Train Time: 1326 s	Precision: 0.5116 Recall: 0.6080 F-1 score: 0.5557 Accuracy: 0.5130 Train Time: 1318 s	Precision: 0.5144 Recall: 0.8574 F-1 score: 0.6430 Accuracy: 0.5232 Train Time: 1319 s	Precision: 0.4974 Recall: 0.6605 F-1 score: 0.5675 Accuracy: 0.4958 Train Time: 1308 s
	progan	Precision: 0.9746 Recall: 0.96 F-1 score: 0.9672 Accuracy: 0.9675 Train Time: 2236 s	Precision: 0.9802 Recall: 0.995 F-1 score: 0.9875 Accuracy: 0.9875 Train Time: 1326 s	Precision: 0.9945 Recall: 0.915 F-1 score: 0.9531 Accuracy: 0.955 Train Time: 1318 s	Precision: 0.9747 Recall: 0.965 F-1 score: 0.9698 Accuracy: 0.97 Train Time: 1319 s	Precision: 0.9705 Recall: 0.99 F-1 score: 0.9801 Accuracy: 0.98 Train Time: 1308 s
	stargan	Precision: 0.7585 Recall: 0.8909 F-1 score: 0.8194 Accuracy: 0.8036 Train Time: 2236 s	Precision: 0.7656 Recall: 0.9674 F-1 score: 0.8548 Accuracy: 0.8356 Train Time: 1326 s	Precision: 0.9461 Recall: 0.8179 F-1 score: 0.8773 Accuracy: 0.8856 Train Time: 1318 s	Precision: 0.7951 Recall: 0.9804 F-1 score: 0.8781 Accuracy: 0.8639 Train Time: 1319 s	Precision: 0.7804 Recall: 0.9764 F-1 score: 0.8675 Accuracy: 0.8509 Train Time: 1308 s
	whichfaceisreal	Precision: 0.5860 Recall: 0.504 F-1 score: 0.5419 Accuracy: 0.574 Train Time: 2236 s	Precision: 0.6442 Recall: 0.393 F-1 score: 0.4881 Accuracy: 0.588 Train Time: 1326 s	Precision: 0.6998 Recall: 0.359 F-1 score: 0.4745 Accuracy: 0.6025 Train Time: 1318 s	Precision: 0.6289 Recall: 0.473 F-1 score: 0.5399 Accuracy: 0.597 Train Time: 1319 s	Precision: 0.6691 Recall: 0.441 F-1 score: 0.5316 Accuracy: 0.6115 Train Time: 1308 s
Knowledge Distillation	**DeepFake**	Precision: 0.4909 Recall: 0.3690 F-1 score: 0.4213 Accuracy: 0.4923 Train Time: 2236 s	Precision: 0.4909 Recall: 0.3018 F-1 score: 0.3738 Accuracy: 0.4936 Train Time: 1642 s	Precision: 0.5242 Recall: 0.2670 F-1 score: 0.3538 Accuracy: 0.5115 Train Time: 1609 s	Precision: 0.4852 Recall: 0.2803 F-1 score: 0.3554 Accuracy: 0.4906 Train Time: 1609 s	Precision: 0.5475 Recall: 0.7480 F-1 score: 0.6323 Accuracy: 0.5642 Train Time: 1599 s
	progan	Precision: 0.9746 Recall: 0.96 F-1 score: 0.9672 Accuracy: 0.9675 Train Time: 2236 s	Precision: 0.9846 Recall: 0.96 F-1 score: 0.9721 Accuracy: 0.9725 Train Time: 1642 s	Precision: 0.985 Recall: 0.985 F-1 score: 0.985 Accuracy: 0.985 Train Time: 1609 s	Precision: 0.9801 Recall: 0.99 F-1 score: 0.9850 Accuracy: 0.985 Train Time: 1609 s	Precision: 0.9756 Recall: 1.0 F-1 score: 0.9876 Accuracy: 0.9875 Train Time: 1599 s
	stargan	Precision: 0.7585 Recall: 0.8909 F-1 score: 0.8194 Accuracy: 0.8036 Train Time: 2236 s	Precision: 0.7259 Recall: 0.8639 F-1 score: 0.7889 Accuracy: 0.7688 Train Time: 1642 s	Precision: 0.7996 Recall: 0.8024 F-1 score: 0.8009 Accuracy: 0.8006 Train Time: 1609 s	Precision: 0.7564 Recall: 0.8344 F-1 score: 0.7935 Accuracy: 0.7828 Train Time: 1609 s	Precision: 0.8919 Recall: 0.9784 F-1 score: 0.9332 Accuracy: 0.9299 Train Time: 1599 s
	whichfaceisreal	Precision: 0.5860 Recall: 0.504 F-1 score: 0.5419 Accuracy: 0.574 Train Time: 2236 s	Precision: 0.5771 Recall: 0.479 F-1 score: 0.5234 Accuracy: 0.564 Train Time: 1642 s	Precision: 0.5948 Recall: 0.458 F-1 score: 0.5175 Accuracy: 0.573 Train Time: 1609 s	Precision: 0.6058 Recall: 0.521 F-1 score: 0.5602 Accuracy: 0.591 Train Time: 1609 s	Precision: 0.6403 Recall: 0.593 F-1 score: 0.6157 Accuracy: 0.63 Train Time: 1599 s
			Original Model		**Compressed Model**	
Quantization	**DeepFake**		Original size: 18.41 Mb Precision: 0.4909 Recall: 0.3690 F-1 score: 0.4213 Accuracy: 0.4923 Train Time: 2236 s CPU inference time on 5405 images: 1568 s		Compressed size: 13.39 Mb Precision: 0.4906 Recall: 0.3690 F-1 score: 0.4212 Accuracy: 0.4921 CPU inference time on 5405 images: 1470 s	
	progan		Original size: 18.41 Mb Precision: 0.9746 Recall: 0.96 F-1 score: 0.9672 Accuracy: 0.9675 Train Time: 2236 s CPU inference time on 400 images: 113 s		Compressed size: 13.39 Mb Precision: 0.9746 Recall: 0.96 F-1 score: 0.9672 Accuracy: 0.9675 CPU inference time on 400 images: 113 s	
	stargan		Original size: 18.41 Mb Precision: 0.7585 Recall: 0.8909 F-1 score: 0.8194 Accuracy: 0.8036 Train Time: 2236 s CPU inference time on 3998 images: 1154 s		Compressed size: 13.39 Mb Precision: 0.7577 Recall: 0.8904 F-1 score: 0.8187 Accuracy: 0.8029 CPU inference time on 3998 images: 1086 s	
	whichfaceisreal		Original size: 18.41 Mb Precision: 0.5860 Recall: 0.504 F-1 score: 0.5419 Accuracy: 0.574 Train Time: 2236 s CPU inference time on 2000 images: 606 s		Compressed size: 13.39 Mb Precision: 0.5865 Recall: 0.505 F-1 score: 0.5427 Accuracy: 0.5745 CPU inference time on 2000 images: 600 s	

These experiments highlighted that when the same type of synthetic data were used for both training and evaluation, performance remained exceptionally high, and the compression techniques could preserve high accuracy, even when the compressed model retained only 10% of the parameters of the original. However, evaluating the model on synthetic images generated by a different DeepFake method presents a new challenge. This finding underscores the necessity of transfer learning techniques to improve generalization across different synthetic data sources, motivating further experiments in this direction.

5.4.4. Experimental Evaluation on Transfer Learning Between Models Trained on Different Types of Datasets

Moving on, transfer learning was applied from the model trained on the “Cats vs. Dogs” dataset across a model fine-tuned and evaluated on the “DeepFake and real images” dataset. The goal of this experimental evaluation was to make the two tasks of transfer learning significantly different. The experimental outcomes are presented in Table 6. Although pruning with fine-tuning was highly resource-efficient and achieved a commendable performance, it proved less effective in terms of overall performance compared to knowledge distillation. Furthermore, the introduction of adapters enhanced the performance of both the pruned and distilled models across all experiments. Notably, while the pruned models experienced the greatest performance improvement from the adapters, the combination of knowledge distillation and adapters remained superior overall. As previously discussed, the quantization method retained the improved performance from the transfer learning applied to the original model; however, its compression was significantly lower compared to pruning and knowledge distillation.

Table 6. Transfer learning from a model trained on the “Cats vs. Dogs” dataset to a model fine-tuned and evaluated on the “DeepFake and Real Images” dataset.

Method	40% Parameters	30% Parameters	20% Parameters	10% Parameters
Pruning + transfer	Precision: 0.8531 Recall: 0.8486 F-1 score: 0.8509 Accuracy: 0.8502 Train Time: 1994 s	Precision: 0.8536 Recall: 0.8623 F-1 score: 0.8579 Accuracy: 0.8562 Train Time: 1850 s	Precision: 0.8477 Recall: 0.8355 F-1 score: 0.8416 Accuracy: 0.8416 Train Time: 1845 s	Precision: 0.8555 Recall: 0.8282 F-1 score: 0.8417 Accuracy: 0.8430 Train Time: 1899 s
Knowledge Distillation	Precision: 0.8928 Recall: 0.9073 F-1 score: 0.9000 Accuracy: 0.8984 Train Time: 2694 s	Precision: 0.8599 Recall: 0.9417 F-1 score: 0.8990 Accuracy: 0.8934 Train Time: 2525 s	Precision: 0.8878 Recall: 0.9136 F-1 score: 0.9005 Accuracy: 0.8983 Train Time: 2498 s	Precision: 0.8209 Recall: 0.8841 F-1 score: 0.8514 Accuracy = 0.8445 Train Time: 2653 s
KD + adapter after last Conv2d layer	Precision: 0.9045 Recall: 0.9200 F-1 score: 0.9122 Accuracy: 0.9108 Train time: 1467 s	Precision: 0.8934 Recall: 0.9280 F-1 score: 0.9104 Accuracy = 0.9080 Train time: 1348 s	Precision: 0.8819 Recall: 0.9344 F-1 score: 0.9074 Accuracy: 0.9039 Train time: 1373 s	Precision: 0.8536 Recall: 0.8834 F-1 score: 0.8682 Accuracy: 0.8650 Train time: 1261 s
Pruning + adapter after fourth Conv2d layer	Precision: 0.8496 Recall: 0.9078 F-1 score: 0.8778 Accuracy: 0.8727 Train time: 2409 s	Precision: 0.8739 Recall: 0.8719 F-1 score: 0.8729 Accuracy: 0.8721 Train time: 2394 s	Precision: 0.8600 Recall: 0.8827 F-1 score: 0.8712 Accuracy: 0.8685 Train time: 2396 s	Precision: 0.8749 Recall: 0.8789 F-1 score: 0.8769 Accuracy: 0.8757 Train time: 2375 s
	Transfered Model		**Compressed Model**	
Quantization	Size: 17.6 Mb Precision: 0.8486 Recall: 0.8507 F-1 score: 0.8449 Accuracy: 0.8565 Transfer time: 1958 s CPU inference on 10,905 images: 3262 s		Size: 12.8 Mb Precision: 0.8486 Recall: 0.8508 F-1 score: 0.8447 Accuracy: 0.8570 CPU inference on 10,905 images: 3148 s	

5.4.5. Experimental Evaluation on Low-Rank Factorization

Table 7 summarizes the results of compressing some deepfake detection models using low-rank factorization with an effective rank of 6. For the “Synthbuster + Raise” dataset, the compressed model exhibited a modest drop in accuracy close to 10%. In contrast, for the compression on the "140k Real and Fake Faces" and the transfer learning in the "DeepFake and Real Images" datasets, low-rank factorization resulted in a drastic performance degradation. The accuracies dropped to around 53% and 50%, respectively. These findings indicate that if we apply low-rank factorization with a low enough rank to achieve significant compression, essential features required for robust detection are stripped from the layers, resulting in high performance degradation in many instances. This is the reason we did not proceed with experiments using low-rank factorization and instead focused on the other three compression methods.

Table 7. Low-rank factorization results.

Dataset	Original Model	Compressed Model
Synthbuster + Raise Dataset	Precision: 0.9420 Recall: 0.9169 F-1 score: 0.9293 Accuracy: 0.9265 Train time: 859 s	Precision: 0.8821 Recall: 0.8802 F-1 score: 0.9474 Acc: 0.8219 Rank: 6
"140k Real and Fake Faces"	Precision: 0.9878 Recall: 0.9914 F-1 score: 0.9896 Accuracy: 0.9896 Train Time: 4382 s	Precision: 1.0 Recall: 0.0725 F-1 score: 0.1351 Accuracy: 0.5362 Rank: 6
"DeepFake and Real Images"	Precision: 0.8526 Recall: 0.8996 F-1 score: 0.8755 Accuracy: 0.8711 Transfer time: 1959 s	Precision: 1.0 Recall: 0.0103 F-1 score: 0.0205 Accuracy: 0.5016 Rank: 6

5.4.6. Discussion

These experiments indicated that knowledge distillation achieved the highest performance, while pruning with fine-tuning offered a faster compression process. Both methods could maintain high accuracy, even when reducing the model to just 10% of its original parameters. Quantization provided an efficient and immediate compression technique; however, its compression rate was limited to 72% of the original model's parameters. These findings align with previous studies, such as [76], which highlighted the effectiveness of knowledge distillation in model compression, while maintaining accuracy. Similarly, ref. [77] demonstrated that pruning, when combined with fine-tuning, provides a strong trade-off between compression and performance, making it a practical approach for deployment on resource-constrained devices. Furthermore, the efficacy of knowledge distillation is heavily influenced by the teacher model's performance. In instances where the teacher model's performance was suboptimal, it was more advantageous to emphasize the loss from the ground truth over the loss from the teacher for better results during the distillation process. This observation is supported by [78], who analyzed the impact of teacher model quality on knowledge distillation and suggested modifications in the loss function to mitigate negative transfer effects, as well as [77].

The effectiveness of pruning and knowledge distillation can be attributed to the prevalent overparameterization in deep learning models. Many neural networks contain an excess of parameters that do not significantly contribute to performance. Pruning takes advantage of this property by identifying and removing less essential parameters, leading to a more efficient model, without severe accuracy degradation. Instead of harming performance, pruning optimizes the model by retaining the parameters that contribute most to learning, while discarding redundant ones. Prior research has highlighted that overparameterization not only facilitates initial training but also allows for effective model compression, without substantial accuracy loss [79,80]. In contrast to pruning, knowledge distillation relies on a different mechanism to enable compression, while retaining model performance. Instead of reducing the parameter count directly, KD transfers knowledge from a high-capacity teacher model to a smaller student model. Through this process, the student model captures the essential patterns learned by the teacher during its training, enabling it to achieve a similar performance, despite its highly reduced complexity [69,81].

The quantization method effectively maintained performance, but the overall compression achieved was not on par with knowledge distillation and pruning, making direct comparisons less meaningful. Additionally, the inability to execute quantized DeepFake models on processing accelerators such as GPUs restricts the applicability of this method in general scenarios. However, in this specific use case, which is DeepFake material detection on edge devices, where accelerators are rarely encountered, most practical scenarios will require inference in small low-end CPUs, making this method still valuable. The limitations of PyTorch quantization on GPU execution have been discussed in prior works, such as ref. [82].

Quantization maintains model performance by employing a numerical approximation strategy. Unlike pruning or knowledge distillation, which alter the model's structure, quantization preserves all parameters, while reducing their precision. This approach minimizes the impact on model accuracy, only introducing small rounding errors, instead of complete parameter removal, though this results in less storage and inference reduction compared to pruning or knowledge distillation.

Experiments were also conducted on low-rank factorization, which are not presented in the tables. Low-rank factorization reduced the size of the original models to a certain extent but frequently resulted in significant performance degradation. This substantial performance degradation limits the utility of this method, making it highly context-dependent. This is in line with findings from [83], who demonstrated that while low-rank approximations can reduce the size of deep networks, they often introduce significant accuracy drops. This made us not present the experimental outcomes in the article, but they are included in the GitHub repository of the paper [75]. Furthermore, it was concluded that the limited compression achieved necessitates its use in conjunction with other compression methods to attain the substantial compression that is commonly needed in DeepFake edge computing scenarios.

The primary reason for the failure of low-rank factorization in certain cases lies in its approach to compression. Unlike pruning or knowledge distillation, which selectively preserve crucial components of the model, low-rank factorization applies a general approximation technique that does not account for the individual importance of specific parameters. Instead, it attempts to reconstruct the model's functionality using fewer basis components, which can result in a significant loss in representational power when the approximation fails to capture critical patterns. Prior research has shown that, while low-rank factorization can be effective under controlled conditions, it often struggles to maintain accuracy when applied to highly complex models [84].

In terms of transfer knowledge, the findings indicate that the position of the adapter significantly influences the performance of the method. For deeper models with more layers, the adapter typically needs to be placed closer to the middle layers to maximize its effect and consequently enhance model performance. This suggests that recognizing moderately complex features is more effective than very simple or highly complex ones.

Therefore, in deeper pruned DeepFake models, the adapter was positioned after the fourth convolutional layer, which is approximately in the middle of the model. Conversely, in smaller DeepFake models that lacked complexity and depth, such as most of the student models used in the knowledge distillation method, placing the adapter at the end of the convolutional layers, the deepest position possible, yielded the greatest benefit. A study by ref. [85] supported this observation, highlighting that adapter placement plays a crucial role in transfer learning performance. Finally, the experiments revealed that adapters are only effective for feature extraction. Adapters consisting of linear layers did not demonstrate any noticeable performance gains in any of the experiments, and pruning emerged as the most effective and resource efficient technique in terms of both compression and performance for the DeepFake tasks.

The findings of this study have significant implications for the deployment of DeepFake detection models in real-world ambient intelligence applications. By systematically evaluating and optimizing compression techniques, this research provides a roadmap for designing lightweight, efficient, and scalable detection systems suitable for edge computing and IoT environments. The ability to deploy robust DeepFake detection models on resource-constrained devices is crucial for maintaining security in smart surveillance, digital identity verification, and multimedia authentication. Furthermore, as DeepFake technologies continue to evolve, the integration of these detection mechanisms into real-time decision-making systems will be essential for preventing misinformation and ensuring the integrity of AI-driven interactions. These contributions highlight the broader impact of model compression strategies beyond theoretical performance improvements, reinforcing their role in practical, security-critical applications.

6. Research Implications

This study contributes to existing knowledge by providing a detailed analysis of widely used compression techniques, specifically in the context of DeepFake detection. By evaluating knowledge distillation, pruning, quantization, and low-rank factorization, it highlights both their effectiveness and potential drawbacks, offering insights into how these limitations can be mitigated when handling synthetic DeepFake material. From a practical perspective, the findings serve as a concise set of guidelines for designing lightweight yet high-performing models suitable for deployment on IoT and edge devices within ambient intelligence systems. However, certain limitations in this work should be acknowledged, including constraints related to dataset size and quality, lack of variation in model architectures, and the extensive range of possible sub-variations within each compression method. These factors may influence the generalizability of the results, suggesting opportunities for future research to explore a broader range of datasets and methodological refinements.

7. Future Work

Looking ahead, future research will explore hybrid approaches that combine multiple compression techniques, such as integrating pruning with knowledge distillation or adaptive quantization, to further enhance DeepFake detection capabilities. Additionally, emerging technologies like neural architecture search [86] will be investigated to optimize model performance, while maintaining efficiency. Another crucial direction involves expanding the evaluation across diverse and larger datasets, including multimodal DeepFake detection scenarios, to improve the robustness and generalizability of compressed models. Finally, given the practical constraints of edge and IoT deployment, future work will focus on real-world testing and hardware-aware optimizations [87] to ensure efficient inference in resource-limited environments, further strengthening the security and trustworthiness of ambient intelligence ecosystems.

8. Conclusions

The research findings demonstrate that DeepFake detection models can be significantly compressed, reducing computational demands, without a substantial loss in performance, making them well-suited for real-time deployment in resource-constrained edge computing environments. Instead of training models from scratch, leveraging transfer learning with pre-trained neural networks proved to be a more efficient approach, preserving model accuracy, while reducing training time and resource consumption. Furthermore, the study highlights how different synthetic image generators impact transfer learning effectiveness, underscoring the need for adaptive model optimization. These advancements are crucial for integrating DeepFake detection into smart cities, IoT networks, and intelligent surveillance systems, where computational efficiency, localized processing, and security are paramount.

Author Contributions: Conceptualization, J.V. and A.K.; methodology, J.V.; software, A.K.; validation, A.K.; formal analysis, J.V.; investigation, J.V.; resources, A.K.; data curation, A.K.; writing—original draft preparation, J.V. and A.K.; writing—review and editing, J.V.; visualization, A.K.; supervision, J.V.; project administration, I.K.; funding acquisition, I.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the European Union's Horizon Europe research and innovation program under grant agreement No. 101120237 (ELIAS).

Data Availability Statement: All data used in this research are publicly available and were sourced from publicly available datasets or repositories. No additional data were generated or collected specifically for this study.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CNNs	Convolutional Neural Networks
DNNs	Deep Neural Networks
FC	Fully Connected
I2G	Inconsistency Image Generator
KL	Kullback–Leibler
PCL	Pair-wise self-Consistency Learning
S-T	Student-Teacher
SVD	Singular Value Decomposition
ViT	Visual Transformers

References

1. Bimpas, A.; Violos, J.; Leivadeas, A.; Varlamis, I. Leveraging pervasive computing for ambient intelligence: A survey on recent advancements, applications and open challenges. *Comput. Netw.* **2024**, *239*, 110156. [CrossRef]
2. Mitra, A.; Mohanty, S.P.; Corcoran, P.; Kougianos, E. iFace: A Deepfake Resilient Digital Identification Framework for Smart Cities. In Proceedings of the 2021 IEEE International Symposium on Smart Electronic Systems (iSES), Jaipur, India, 18–22 December 2021; pp. 361–366. [CrossRef]
3. Nagothu, D.; Schwell, J.; Chen, Y.; Blasch, E.; Zhu, S. A study on smart online frame forging attacks against Video Surveillance System. In Proceedings of the Sensors and Systems for Space Applications XII, Baltimore, MD, USA, 15–16 April 2019; SPIE: Paris, France, 2019; Volume 11017, pp. 176–188. [CrossRef]
4. Bilika, D.; Michopoulou, N.; Alepis, E.; Patsakis, C. Hello Me, Meet the Real Me: Audio Deepfake Attacks on Voice Assistants. *arXiv* **2023**, arXiv:2302.10328. [CrossRef]
5. Sridevi, K.; Kumar, K.S.; Sameera, D.; Garapati, Y.; Krishnamadhuri, D.; Bethu, S. IoT based application designing of Deep Fake Test for Face animation. In Proceedings of the 2022 6th International Conference on Cloud and Big Data Computing, New York, NY, USA, 18–20 August 2022; ICCBDC '22, pp. 24–30. [CrossRef]

6. Bethu, S.; Trupthi, M.; Mandala, S.K.; Karimunnisa, S.; Banu, A. AI-IoT Enabled Surveillance Security: DeepFake Detection and Person Re-Identification Strategies. *Int. J. Adv. Comput. Sci. Appl.* **2024**, *15*, 1013. [CrossRef]
7. Chen, J.; Ran, X. Deep Learning With Edge Computing: A Review. *Proc. IEEE* **2019**, *107*, 1655–1674. [CrossRef]
8. Caire, P.; Moawad, A.; Efthymiou, V.; Bikakis, A.; Le Traon, Y. Privacy challenges in ambient intelligence systems. *J. Ambient. Intell. Smart Environ.* **2016**, *8*, 619–644. [CrossRef]
9. Hadidi, R.; Cao, J.; Xie, Y.; Asgari, B.; Krishna, T.; Kim, H. Characterizing the deployment of deep neural networks on commercial edge devices. In Proceedings of the 2019 IEEE International Symposium on Workload Characterization (IISWC), Orlando, FL, USA, 3–9 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 35–48.
10. Liang, T.; Glossner, J.; Wang, L.; Shi, S.; Zhang, X. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing* **2021**, *461*, 370–403. [CrossRef]
11. Choukroun, Y.; Kravchik, E.; Yang, F.; Kisilev, P. Low-bit Quantization of Neural Networks for Efficient Inference. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), Seoul, Korea, 27–28 October 2019; pp. 3009–3018. [CrossRef]
12. Wang, L.; Yoon, K.J. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 3048–3068. [CrossRef]
13. Sainath, T.N.; Kingsbury, B.; Sindhwani, V.; Arisoy, E.; Ramabhadran, B. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 26–31 May 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 6655–6659.
14. Vrbančič, G.; Podgorelec, V. Transfer Learning With Adaptive Fine-Tuning. *IEEE Access* **2020**, *8*, 196197–196211. [CrossRef]
15. Moosavi, N.S.; Delfosse, Q.; Kersting, K.; Gurevych, I. Adaptable adapters. *arXiv* **2022**, arXiv:2205.01549.
16. Bica, I.; Chifor, B.C.; Arseni, S.C.; Matei, I. Multi-Layer IoT Security Framework for Ambient Intelligence Environments. *Sensors* **2019**, *19*, 4038. [CrossRef]
17. Coccomini, D.A.; Caldelli, R.; Falchi, F.; Gennaro, C.; Amato, G. Cross-forgery analysis of vision transformers and cnns for deepfake image detection. In Proceedings of the 1st International Workshop on Multimedia AI Against Disinformation, Newark, NJ, USA, 27–30 June 2022; pp. 52–58.
18. Bai, Y.; Mei, J.; Yuille, A.L.; Xie, C. Are transformers more robust than cnns? *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 26831–26843.
19. Naseer, M.M.; Ranasinghe, K.; Khan, S.H.; Hayat, M.; Shahbaz Khan, F.; Yang, M.H. Intriguing properties of vision transformers. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 23296–23308.
20. Wang, Z.; Bai, Y.; Zhou, Y.; Xie, C. Can cnns be more robust than transformers? *arXiv* **2022**, arXiv:2206.03452.
21. Nguyen, H.H.; Yamagishi, J.; Echizen, I. Capsule-forensics: Using capsule networks to detect forged images and videos. In Proceedings of the ICASSP 2019–2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 2307–2311.
22. Hinton, G.E.; Sabour, S.; Frosst, N. Matrix capsules with EM routing. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
23. Sabour, S.; Frosst, N.; Hinton, G.E. Dynamic routing between capsules. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1–11.
24. Maurício, J.; Domingues, I.; Bernardino, J. Comparing vision transformers and convolutional neural networks for image classification: A literature review. *Appl. Sci.* **2023**, *13*, 5521. [CrossRef]
25. Patrick, M.K.; Adekoya, A.F.; Mighty, A.A.; Edward, B.Y. Capsule networks–a survey. *J. King Saud-Univ.-Comput. Inf. Sci.* **2022**, *34*, 1295–1310.
26. Fung, S.; Lu, X.; Zhang, C.; Li, C.T. Deepfakeucl: Deepfake detection via unsupervised contrastive learning. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Virtual, 18–22 July 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–8.
27. Li, J.; Zhou, P.; Xiong, C.; Hoi, S.C. Prototypical contrastive learning of unsupervised representations. *arXiv* **2020**, arXiv:2005.04966.
28. Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G. A simple framework for contrastive learning of visual representations. In Proceedings of the International Conference on Machine Learning, Virtual, 13–18 July 2020; pp. 1597–1607.
29. He, K.; Fan, H.; Wu, Y.; Xie, S.; Girshick, R. Momentum Contrast for Unsupervised Visual Representation Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 9729–9738.
30. Misra, I.; Maaten, L.v.d. Self-supervised learning of pretext-invariant representations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 6707–6717.
31. Khosla, P.; Teterwak, P.; Wang, C.; Sarna, A.; Tian, Y.; Isola, P.; Maschinot, A.; Liu, C.; Krishnan, D. Supervised contrastive learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 18661–18673.
32. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. *arXiv* **2017**, arXiv:1610.02357. [CrossRef]

33. Wang, X.; Qi, G.J. Contrastive learning with stronger augmentations. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 5549–5560. [CrossRef]
34. Wu, A.; Zheng, W.S.; Lai, J.H. Unsupervised person re-identification by camera-aware similarity consistency learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 6922–6931.
35. Zhou, D.; Bousquet, O.; Lal, T.; Weston, J.; Schölkopf, B. Learning with local and global consistency. *Adv. Neural Inf. Process. Syst.* **2003**, *16*, 1–7.
36. Mayer, O.; Stamm, M.C. Forensic similarity for digital images. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 1331–1346. [CrossRef]
37. Huh, M.; Liu, A.; Owens, A.; Efros, A.A. Fighting fake news: Image splice detection via learned self-consistency. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 101–117.
38. Zhao, T.; Xu, X.; Xu, M.; Ding, H.; Xiong, Y.; Xia, W. Learning self-consistency for deepfake detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 15023–15033.
39. Zhang, T. Deepfake generation and detection, a survey. *Multimed. Tools Appl.* **2022**, *81*, 6259–6276. [CrossRef]
40. Dagar, D.; Vishwakarma, D.K. A literature review and perspectives in deepfakes: Generation, detection, and applications. *Int. J. Multimed. Inf. Retr.* **2022**, *11*, 219–289. [CrossRef]
41. Edwards, P.; Nebel, J.C.; Greenhill, D.; Liang, X. A Review of Deepfake Techniques: Architecture, Detection and Datasets. *IEEE Access* **2024**, *12*, 154718–154742. [CrossRef]
42. Zhang, Y.; Colman, B.; Guo, X.; Shahriyari, A.; Bharaj, G. Common sense reasoning for deepfake detection. In Proceedings of the European Conference on Computer Vision, Milan, Italy, 29 September–4 October 2024; Springer: Cham, Switzerland, 2024; pp. 399–415.
43. Guarnera, L.; Giudice, O.; Battiato, S. Mastering deepfake detection: A cutting-edge approach to distinguish GAN and diffusion-model images. *ACM Trans. Multimed. Comput. Commun. Appl.* **2024**, *20*, 1–24. [CrossRef]
44. Rokh, B.; Azarpeyvand, A.; Khanteymoori, A. A Comprehensive Survey on Model Quantization for Deep Neural Networks in Image Classification. *ACM Trans. Intell. Syst. Technol.* **2023**, *14*, 97:1–97:50. [CrossRef]
45. Han, S.; Mao, H.; Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv* **2015**, arXiv:1510.00149.
46. Tsanakas, S.; Hameed, A.; Violos, J.; Leivadeas, A. A light-weight edge-enabled knowledge distillation technique for next location prediction of multitude transportation means. *Future Gener. Comput. Syst.* **2024**, *154*, 45–58. [CrossRef]
47. Jaderberg, M.; Vedaldi, A.; Zisserman, A. Speeding up convolutional neural networks with low rank expansions. *arXiv* **2014**, arXiv:1405.3866.
48. Pham, N.S.; Shin, S.; Xu, L.; Shi, W.; Suh, T. Starspa: Stride-aware sparsity compression for efficient cnn acceleration. *IEEE Access* **2024**, *12*, 10893–10909. [CrossRef]
49. Lian, Y.; Peng, P.; Jiang, K.; Xu, W. Multi-objective compression for CNNs via evolutionary algorithm. *Inf. Sci.* **2024**, *661*, 120155. [CrossRef]
50. Lopes, A.; dos Santos, F.P.; de Oliveira, D.; Schiezaro, M.; Pedrini, H. Computer vision model compression techniques for embedded systems: A survey. *Comput. Graph.* **2024**, *123*, 104015. [CrossRef]
51. Tajbakhsh, N.; Shin, J.Y.; Gurudu, S.R.; Hurst, R.T.; Kendall, C.B.; Gotway, M.B.; Liang, J. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE Trans. Med. Imaging* **2016**, *35*, 1299–1312. [CrossRef] [PubMed]
52. Guo, Y.; Shi, H.; Kumar, A.; Grauman, K.; Rosing, T.; Feris, R. Spottune: Transfer learning through adaptive fine-tuning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4805–4814.
53. Csurka, G. Domain adaptation for visual applications: A comprehensive survey. *arXiv* **2017**, arXiv:1702.05374.
54. Ben-David, S.; Blitzer, J.; Crammer, K.; Pereira, F. Analysis of representations for domain adaptation. *Adv. Neural Inf. Process. Syst.* **2006**, *19*, 1–9.
55. Blitzer, J.; McDonald, R.; Pereira, F. Domain adaptation with structural correspondence learning. In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2006; pp. 120–128.
56. Gholizade, M.; Soltanizadeh, H.; Rahmanimanesh, M.; Sana, S.S. A review of recent advances and strategies in transfer learning. *Int. J. Syst. Assur. Eng. Manag.* **2025**, *16*, 1–40. [CrossRef]
57. Al-Dulaimi, O.A.H.H.; Kurnaz, S. A hybrid CNN-LSTM approach for precision deepfake image detection based on transfer learning. *Electronics* **2024**, *13*, 1662. [CrossRef]
58. Alazwari, S.; Alsamri, M.O.J.; Alamgeer, M.; Alabdan, R.; Alzahrani, I.; Rizwanullah, M.; Osman, A.E. Artificial rabbits optimization with transfer learning based deepfake detection model for biometric applications. *Ain Shams Eng. J.* **2024**, *15*, 103057. [CrossRef]

59. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv* **2019**, arXiv:1910.01108.
60. Sun, Z.; Yu, H.; Song, X.; Liu, R.; Yang, Y.; Zhou, D. Mobilebert: A compact task-agnostic bert for resource-limited devices. *arXiv* **2020**, arXiv:2004.02984.
61. Howard, A. Mobilenets: Efficient convolu-tional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
62. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. *arXiv* **2016**, arXiv:1602.07360.
63. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
64. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16×16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
65. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-end object detection with transformers. In Proceedings of the European Conference on Computer Vision, Milan, Italy, 29 September–4 October 2020; Springer: Cham, Switzerland, 2020; pp. 213–229.
66. Violos, J.; Papadopoulos, S.; Kompatsiaris, I. Towards Optimal Trade-Offs in Knowledge Distillation for CNNs and Vision Transformers at the Edge. In Proceedings of the 2024 32nd European Signal Processing Conference (EUSIPCO), Lyon, France, 26–30 August 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 1896–1900.
67. Kinnas, M.; Violos, J.; Karapiperis, N.I.; Kompatsiaris, I. Selecting Images With Entropy for Frugal Knowledge Distillation. *IEEE Access* **2025**, *13*, 28189–28203. [CrossRef]
68. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2015**, arXiv:1409.1556. [CrossRef]
69. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
70. Masana, M.; Van De Weijer, J.; Herranz, L.; Bagdanov, A.D.; Alvarez, J.M. Domain-adaptive deep network compression. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 4289–4297.
71. Chen, Z.; Duan, Y.; Wang, W.; He, J.; Lu, T.; Dai, J.; Qiao, Y. Vision transformer adapter for dense predictions. *arXiv* **2022**, arXiv:2205.08534.
72. Wang, R.; Tang, D.; Duan, N.; Wei, Z.; Huang, X.; Ji, J.; Cao, G.; Jiang, D.; Zhou, M. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv* **2020**, arXiv:2002.01808.
73. Bammey, Q. Synthbuster: Towards detection of diffusion model generated images. *IEEE Open J. Signal Process.* **2023**, *5*, 1–9. [CrossRef]
74. Wang, S.Y.; Wang, O.; Zhang, R.; Owens, A.; Efros, A.A. CNN-generated images are surprisingly easy to spot... for now. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 8695–8704.
75. Karathanasis, A. Andreaskarathanasis/Compression-Transfer-of-DeepFake-Models. 2024. Original-Date: 2025-03-6T05:44:03Z. Available online: https://github.com/andreaskarathanasis/Compression-Transfer-of-DeepFake-Models (accessed on 2 March 2025).
76. Hong, Y.W.; Leu, J.S.; Faisal, M.; Prakosa, S.W. Analysis of model compression using knowledge distillation. *IEEE Access* **2022**, *10*, 85095–85105. [CrossRef]
77. Meng, L.; Qiao, G.; Zhang, X.; Bai, J.; Zuo, Y.; Zhou, P.; Liu, Y.; Hu, S. An efficient pruning and fine-tuning method for deep spiking neural network. *Appl. Intell.* **2023**, *53*, 28910–28923. [CrossRef]
78. Alballa, N.; Canini, M. Practical Insights into Knowledge Distillation for Pre-Trained Models. *arXiv* **2024**, arXiv:2402.14922.
79. Gale, T.; Elsen, E.; Hooker, S. The state of sparsity in deep neural networks. *arXiv* **2019**, arXiv:1902.09574.
80. Frankle, J.; Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv* **2018**, arXiv:1803.03635.
81. Gou, J.; Yu, B.; Maybank, S.J.; Tao, D. Knowledge distillation: A survey. *Int. J. Comput. Vis.* **2021**, *129*, 1789–1819. [CrossRef]
82. Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv* **2018**, arXiv:1806.08342.
83. Denton, E.L.; Zaremba, W.; Bruna, J.; LeCun, Y.; Fergus, R. Exploiting linear structure within convolutional networks for efficient evaluation. *Adv. Neural Inf. Process. Syst.* **2014**, *27*, 1–9.
84. Idelbayev, Y.; Carreira-Perpinán, M.A. Low-rank compression of neural nets: Learning the rank of each layer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 8049–8059.
85. Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; Gelly, S. Parameter-efficient transfer learning for NLP. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 2790–2799.

86. Elsken, T.; Metzen, J.H.; Hutter, F. Neural architecture search: A survey. *J. Mach. Learn. Res.* **2019**, *20*, 1–21.
87. Marculescu, D.; Stamoulis, D.; Cai, E. Hardware-aware machine learning: Modeling and optimization. In Proceedings of the 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Diego, CA, USA, 5–8 November 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–8.