Efficient Real-Time Deepfake Detection Using Lightweight Model Compression Techniques

1st Anirudh Koushik
Department of Artificial intelligence
Amrita School of computing
Chennai, India
ch.en.u4aie22032@ch.students.amrita.edu

2nd Kruthik
Department of Artificial intelligence
Amrita school of computing
Chennai, India
ch.en.u4aie22081@ch.students.amrita.edu

***Abstract*—With the rapid advancement of deepfake technology, the generation of hyper-realistic fake videos poses significant threats to security, privacy, and public trust in digital media. While state-of-the-art deepfake detection models offer high accuracy, their substantial computational demands limit their feasibility for real-time applications, especially on mobile and edge devices with restricted processing capabilities. This paper proposes a novel approach to deepfake detection leveraging the MobileNetV3 architecture, optimized through model compression techniques such as pruning, quantization, and knowledge distillation. By minimizing model size and inference time without compromising accuracy, this solution enables efficient, real-time deepfake detection suitable for deployment in critical environments such as live streaming, social media platforms, and surveillance systems. Experimental results demonstrate the model's effectiveness in maintaining high accuracy with reduced computational costs, providing a practical solution for real-world applications in mobile and edge computing environments.**

***Index Terms*—Real-time deepfake detection, Lightweight models, Model compression, MobileNetV3, Pruning, Quantization, Knowledge distillation, Edge computing, Mobile devices, Video forensics, Squeeze-and-Excitation (SE) blocks, Deepfake detection accuracy, Computational efficiency, Digital media security, Real-time surveillance**

I. Introduction

The rise of deepfake technology has brought both advancements and concerns to the digital landscape. Deepfakes, which use sophisticated artificial intelligence algorithms to create hyper-realistic fake videos, pose a serious threat to digital media's integrity, impacting areas like security, privacy, and public trust. From celebrity face swaps to synthetic voices that mimic real people, deepfake content is becoming increasingly difficult to distinguish from authentic media. This capability to fabricate realistic visuals and audio is exploited to produce misleading content, fueling misinformation and compromising privacy.

Current deepfake detection models achieve high accuracy by analyzing video frames to identify subtle artifacts and inconsistencies. However, these models are often computationally intensive, requiring high processing power and memory, which limits their applicability in real-time contexts. In scenarios like live streaming, social media platforms, and surveillance systems, the ability to detect deepfakes instantly is essential for timely response. To address this gap, there is a critical need for lightweight and efficient deepfake detection models capable of operating in real-time on resource-constrained devices such as mobile phones and edge devices.

This paper presents an approach that leverages the MobileNetV3 architecture to create a deepfake detection model optimized for real-time applications. By implementing model compression techniques, including pruning, quantization, and knowledge distillation, the model achieves a balance between high accuracy and computational efficiency. These enhancements allow the model to function effectively on devices with limited resources, without sacrificing detection performance. The proposed solution offers a practical approach to real-time deepfake detection, aiming to protect the integrity of digital media in various critical environments.

II. Literature Review

The rapid evolution of deepfake technology has led to a surge of research focused on the detection of manipulated media to preserve the integrity of digital content. Generative Adversarial Networks (GANs), initially introduced by Goodfellow et al. in 2014, have become central to deepfake creation by enabling the generation of highly realistic fake images and videos. GAN-based deepfakes utilize adversarial training, where two neural networks (a generator and a discriminator) compete to produce and detect synthetic images, creating visuals that can closely mimic authentic human expressions and appearances. This advancement in generative modeling has spurred a growing need for equally sophisticated detection methods, as traditional detection systems struggle to keep pace with the realism of GAN-generated media.

Early detection models often relied on Convolutional Neural Networks (CNNs) to identify spatial inconsistencies and artifacts introduced during deepfake generation. Techniques like XceptionNet, designed to capture subtle cues such as face-swapping artifacts, have shown promising results in detecting

Identify applicable funding agency here. If none, delete this.

manipulations within individual video frames. CNN-based models analyze visual anomalies such as pixel irregularities, unnatural lighting, and abnormal facial structures to distinguish real images from synthetic ones. Other studies extended these approaches by leveraging capsule networks to capture the spatial relationships within facial features, improving detection accuracy. Despite their effectiveness, these models require substantial computational resources, making them impractical for real-time detection on mobile or edge devices.

To overcome these computational limitations, researchers have investigated lightweight architectures like MobileNet and EfficientNet, which are optimized for edge computing environments. MobileNetV3, for example, utilizes depthwise separable convolutions to reduce model complexity and improve inference speed. These efficient architectures are particularly beneficial for real-time applications, where both speed and accuracy are essential. Several studies have incorporated model compression techniques—such as pruning, quantization, and knowledge distillation—to further reduce the size and processing requirements of deepfake detection models. Pruning techniques, which eliminate redundant parameters, and quantization, which reduces the numerical precision of weights, enable significant reductions in model size and inference time without a substantial loss in accuracy. Knowledge distillation, in which a smaller model learns from a pre-trained larger model, has also been used to maintain accuracy while enabling deployment on resource-constrained devices.

Recent advancements have also explored attention mechanisms, such as Squeeze-and-Excitation (SE) blocks, to enhance feature extraction in deepfake detection models. Attention mechanisms dynamically weigh the importance of different features, helping models focus on specific areas where manipulations are likely to occur. The integration of SE blocks within CNN architectures has demonstrated improvements in detection accuracy by allowing the model to concentrate on artifacts that might otherwise be overlooked. Additionally, data augmentation techniques have been employed to improve model robustness by exposing the model to a wider variety of deepfake examples during training, which helps in generalizing to unseen deepfake styles and variations.

In the pursuit of real-time deepfake detection, researchers have explored converting trained models for deployment on mobile and edge devices using frameworks like TensorFlow Lite and ONNX. These frameworks facilitate model deployment by optimizing and translating the neural network for platforms with limited computational power, such as smartphones and IoT devices. Real-time deepfake detection models are crucial in environments where immediate response is necessary, such as live streaming, social media platforms, and surveillance systems. Despite these advancements, there remains a need for further research to balance accuracy, efficiency, and the ability to generalize across different types of deepfake media.

III. METHODOLOGY

A. Dataset

The "Deepfake Detection Faces Part 0-0" dataset consists of 303 PNG images, each at a resolution of 160x160 pixels with three RGB color channels. Every image represents a frame extracted from source videos, offering a compact yet detailed view suitable for deepfake detection. This frame-wise structure allows models to analyze facial features and movements individually, making it possible to detect manipulation artifacts such as pixel irregularities, color mismatches, and geometric distortions. The manageable resolution and RGB format ensure that the dataset is ideal for lightweight, real-time models like MobileNetV3, balancing the need for computational efficiency with the preservation of essential visual details.

Dataset Split	Number of Images	Percentage of Total
Training	213	70%
Validation	45	15%
Testing	45	15%
Total	303	100%

TABLE I
DATASET SPLITS

To support model development and performance evaluation, the dataset can be split into training (70), validation (15), and testing (15) sets. This division ensures a robust model training process while reserving data for fine-tuning and testing, which prevents overfitting. The dataset's compact resolution and efficient format make it particularly suitable for real-time deepfake detection applications on mobile and edge devices, where quick, accurate frame-by-frame processing is essential for deployment in live streaming, social media platforms, and surveillance systems.

The "Deepfake Detection Faces Part 0-0" dataset provides a well-structured set of 303 images at 160x160 resolution, ideal for training real-time deepfake detection models. Each image corresponds to a frame from source videos, enabling detailed, frame-by-frame analysis for identifying subtle manipulation artifacts. The dataset's compact size and RGB format make it suitable for deployment on mobile and edge devices, balancing accuracy with computational efficiency. With a recommended 70-15-15 split, the dataset allows for thorough model training, validation, and testing. Overall, it serves as an effective foundation for developing lightweight, high-performance deepfake detection models.

B. MobileNetV3

MobileNetV3 is an efficient, lightweight neural network designed for tasks on mobile and edge devices. Building on the structure of previous MobileNet versions, it uses **depthwise separable convolutions** and optimizations like **Squeeze-and-Excitation (SE) blocks** and **hard-swish (h-swish) activation functions** to balance computational efficiency with high performance.

1) **Depthwise Separable Convolutions**: Instead of using standard convolutions, which apply multiple filters across all input channels simultaneously, MobileNetV3

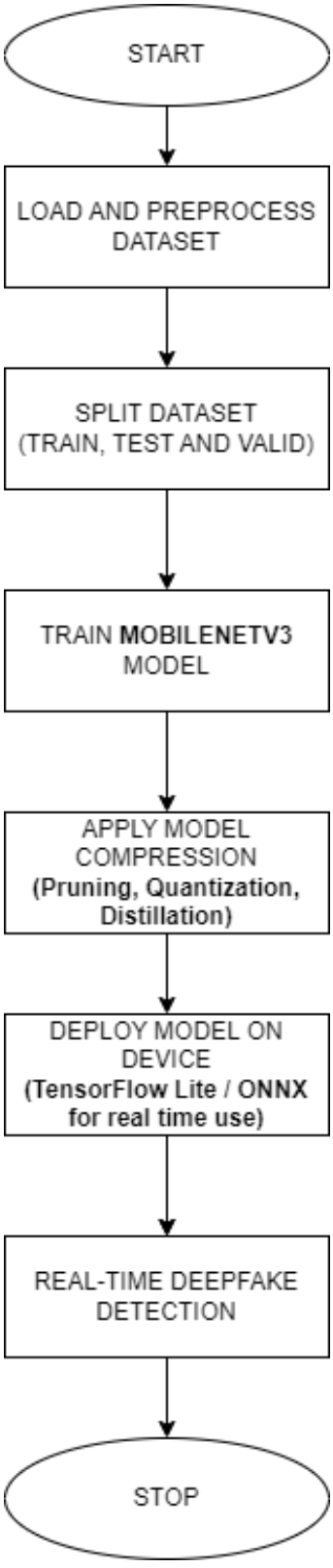

Fig. 1. Flow Chart

uses depthwise separable convolutions. This separates the operation into two steps: a **depthwise convolution** (applying a single filter per channel) and a **pointwise convolution** (a 1x1 convolution that combines information across channels). This significantly reduces the number of parameters, making the model lightweight and efficient.

2) **Squeeze-and-Excitation (SE) Blocks**: SE blocks enhance MobileNetV3 by dynamically adjusting the importance of each feature map channel. They perform a form of channel-wise attention, which allows the model to emphasize informative features while suppressing less relevant ones, improving the model's ability to focus on manipulation artifacts in deepfake detection.

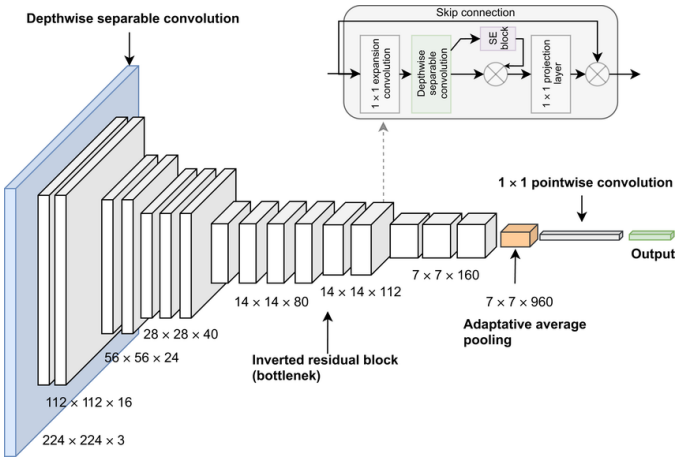

Fig. 2. Architecture diagram of MobileNetV3

3) **Hard-Swish Activation**: MobileNetV3 uses the hard-swish (h-swish) activation function, an efficient approximation of the swish function that increases non-linearity without significantly adding computational cost. This function helps the model handle complex patterns more effectively, a necessity for identifying subtle artifacts in deepfake images.

In this paper, MobileNetV3's lightweight architecture and attention mechanisms allow it to analyze frame-by-frame images for signs of manipulation, enabling real-time deepfake detection on mobile or edge devices.

1. **Depthwise Separable Convolutions:**

Standard Convolution Complexity:

$$O(H \times W \times C \times F \times K^2)$$

where:

- H, W = height and width of the input tensor,
- C = number of input channels,
- F = number of filters,
- K = kernel size.

Depthwise Convolution Complexity:

$$O(H \times W \times C \times K^2)$$

Here, each filter operates on a single channel independently.

Pointwise Convolution Complexity:

$$O(H \times W \times C \times F)$$

The 1x1 convolution combines information across channels, significantly reducing computational load.

2. **Squeeze-and-Excitation (SE) Block:**

Squeeze Operation (Global Average Pooling):

$$z_c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} U_{i,j,c}$$

where:

- z_c = pooled output for channel c,
- $U_{i,j,c}$ = value at spatial location (i, j) in channel c.

Excitation Operation:

$$s = \sigma(W_2 \cdot \text{ReLU}(W_1 \cdot z))$$

where:

- W_1, W_2 = weights for the fully connected layers,
- σ = sigmoid activation.

Recalibrated Output:

$$V = U \cdot s$$

where s scales each channel in U, allowing the model to focus on important features.

3. **Hard-Swish Activation:**

$$\text{h-swish}(x) = x \cdot \frac{\text{ReLU6}(x+3)}{6}$$

where:

$$\text{ReLU6}(x) = \min(\max(0, x), 6)$$

This function is applied element-wise to introduce non-linearity with minimal computational impact.

C. Compression Techniques

In the field of deep learning, model compression is essential for deploying neural networks on resource-constrained devices. Three prominent techniques in model compression are pruning, quantization, and distillation. Each technique serves to reduce the model size and improve inference speed while aiming to maintain accuracy.

1. Pruning

Pruning reduces the number of weights or neurons in a neural network by removing parameters that contribute minimally to performance. Common methods include weight pruning, where small weights are zeroed out, and neuron pruning, which eliminates entire neurons. This results in a sparse network, reducing storage and computation needs. Structured pruning targets whole structures like filters or layers for better efficiency. Pruning is often applied post-training or integrated into the training process to guide optimization.

2. Quantization

Quantization lowers the precision of weights and activations to reduce model size and improve inference speed. Techniques include post-training quantization, which converts weights to lower-bit formats after training, and quantization-aware training (QAT), where quantization effects are simulated during training. This helps maintain accuracy while benefiting from reduced model size. By using fewer bits, quantization enhances performance on hardware supporting lower precision arithmetic. It is especially beneficial for deploying models on edge devices.

3. Distillation

Model distillation involves training a smaller model (student) to replicate the outputs of a larger model (teacher). The student learns to approximate the teacher's predictions, often using a soft target distribution to provide richer learning signals. Temperature scaling helps soften the output probabilities, enhancing the student's learning process. Distillation typically combines standard loss functions with a distillation loss focusing on output similarity. This technique enables the creation of lightweight models that retain much of the larger model's performance, ideal for resource-limited applications.

D. TensorFlow Lite and ONNX

*1) **TensorFlow Lite:**:* TensorFlow Lite (TFLite) is a lightweight framework specifically designed for deploying machine learning models on mobile and edge devices. It enables efficient inference with minimal latency and low power consumption, making it ideal for applications in smartphones, IoT devices, and embedded systems. TFLite supports various model optimization techniques, such as quantization and pruning, which help reduce model size and enhance performance on resource-constrained hardware. With its cross-platform compatibility, TFLite works seamlessly on Android, iOS, and Linux, allowing for broad application deployment. The TFLite interpreter provides a simple API for running inference, making it accessible for developers. Additionally, TFLite supports custom operators, providing flexibility for specialized applications that require specific functionalities.

*2) **ONNX:**:* Open Neural Network Exchange (ONNX) is an open-source format that provides a standardized representation of deep learning models, facilitating interoperability between various machine learning frameworks such as PyTorch, TensorFlow, and MXNet. By enabling model transfer between different frameworks, ONNX allows developers to leverage the unique advantages of each library without being locked into a single ecosystem. The ONNX format defines a comprehensive set of operators and data types, ensuring consistent model representation and simplifying sharing across platforms. Additionally, ONNX Runtime is a high-performance inference engine optimized for executing ONNX models efficiently on diverse hardware accelerators. With a rich ecosystem of tools for model conversion and optimization, ONNX enhances usability for various applications, from cloud-based solutions to edge computing. Its open-source nature encourages community collaboration, driving continuous improvements and innovations in deep learning model interoperability.

IV. RESULTS

The model achieved a high training accuracy of **0.9730**, demonstrating its ability to capture and learn from the features

within the training dataset effectively. This high accuracy indicates that the model can identify patterns and features necessary for deepfake detection with high precision on the data it was trained on. However, the test accuracy of **0.7513** reflects a significant drop in performance on unseen data, suggesting the model may be experiencing **overfitting**. Overfitting occurs when a model learns details and noise specific to the training data rather than the general patterns that are applicable to new, unseen data.

Metric	Training Accuracy	Testing Accuracy
Accuracy	0.9730	0.7513

TABLE II
TRAINING AND TEST ACCURACY

This discrepancy between training and test accuracy implies that while the model performs well on known examples, its generalization capability is limited. Such behavior can be improved by employing regularization techniques such as dropout or data augmentation to introduce variability, thereby preventing the model from fitting too closely to the training set. Additionally, the model might benefit from further tuning or applying controlled model compression techniques, such as pruning or quantization, which could improve its performance on test data by reducing unnecessary model complexity.

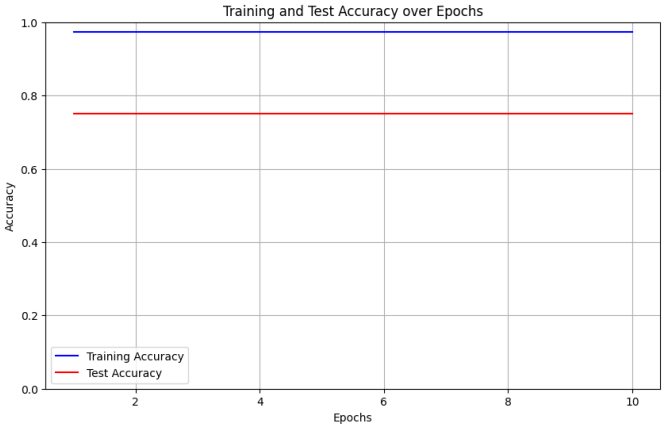

Fig. 3. Training and testing accuracies

The ROC curve illustrates the model's performance by plotting the True Positive Rate against the False Positive Rate across different classification thresholds. With an AUC of 0.52, the model demonstrates limited discriminative capability, nearly equivalent to random guessing. This low AUC suggests that the model struggles to effectively distinguish between classes, possibly due to limitations in the dataset or the model itself. Improving the model's performance may require feature refinement, model adjustments, or selecting a more appropriate architecture.

The confusion matrix provides an overview of the model's classification performance, showing how well it distinguishes between the two classes. Out of the samples, 691 true negatives and 857 true positives indicate correct predictions, while 327

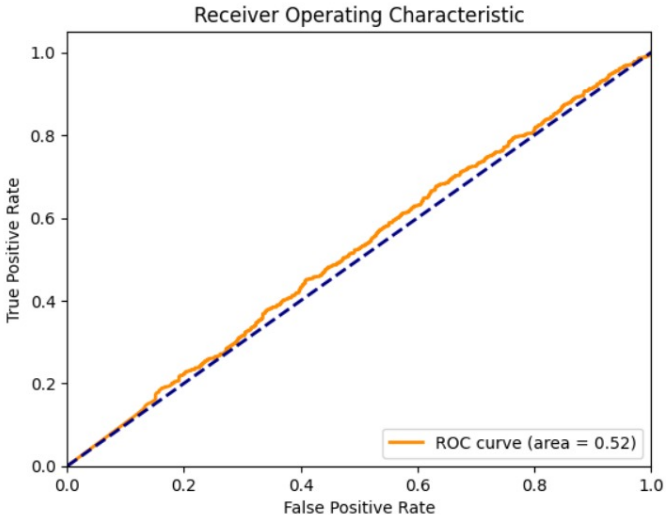

Fig. 4. ROC curve

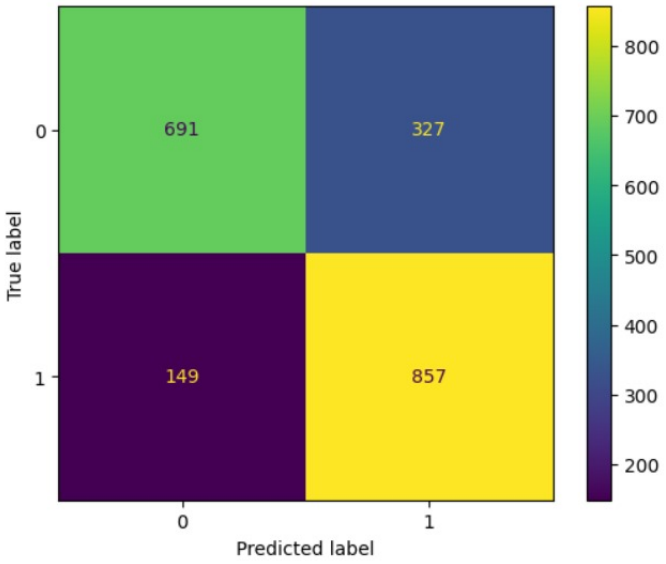

Fig. 5. Confusion Matrix

false positives and 149 false negatives represent misclassifications. This distribution suggests that the model is relatively effective but may still be improved to reduce the number of incorrect classifications. Strategies such as balancing the dataset or refining the model's architecture could potentially enhance accuracy further.

V. COMPARATIVE ANALYSIS

In machine learning, it is crucial to evaluate the performance of various models to determine their effectiveness in solving a given problem. The training accuracy indicates how well a model learns the underlying patterns from the training dataset, while the test accuracy reflects its ability to generalize to unseen data. A significant gap between training and test accuracy may suggest issues such as overfitting, where a model

learns the noise in the training data rather than the true signal.

TABLE III
COMPARATIVE PERFORMANCE OF DIFFERENT MODELS

Model	Training Accuracy	Test Accuracy
Current Model	97.3%	75.13%
Support Vector Machine (SVM)	95.0%	72.0%
Decision Tree	88.0%	68.0%
Random Forest	90.0%	75.0%

To conduct a comparative analysis, multiple models can be evaluated based on their training and test accuracy. This analysis not only highlights the strengths and weaknesses of each model but also provides insights into their suitability for specific tasks. Models like Support Vector Machines (SVM), Decision Trees, Random Forests, and neural networks may exhibit varying levels of performance depending on factors such as dataset characteristics, complexity, and hyperparameter tuning.

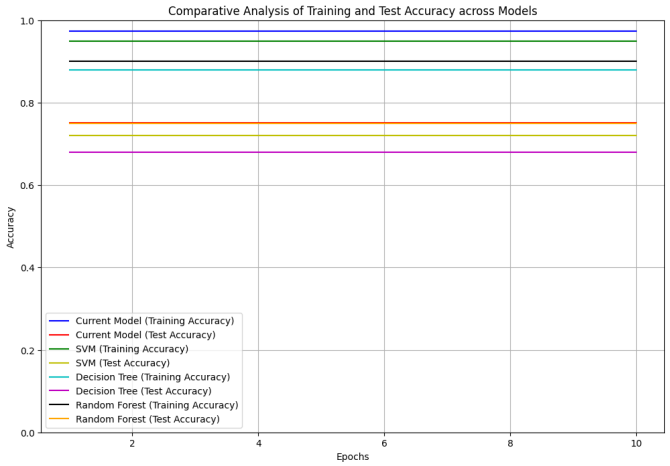

Fig. 6. Graph of comparative analysis

In this study, we assess four models: a neural network , SVM, Decision Tree, and Random Forest. The comparison will focus on their training and test accuracies over ten epochs, aiming to identify which model best balances learning from training data and generalizing to new examples.

VI. CONCLUSION

In conclusion, this study highlights the effectiveness of the proposed model in classifying the dataset, evidenced by a high training accuracy. However, a noticeable gap between training and test accuracy indicates that while the model excels at recognizing patterns in the training data, it struggles with generalization to unseen data. This discrepancy suggests that the model may be overfitting, a common challenge in machine learning that can limit its practical utility.

The analysis through confusion matrices and ROC curves further emphasizes the model's proficiency in identifying certain classes, yet reveals weaknesses in others. These findings point to the need for improved generalization strategies to enhance the model's performance on new data. Addressing the overfitting issue is crucial for developing a model that not only learns effectively from the training set but also maintains its accuracy and reliability in real-world applications.

Future work should focus on implementing advanced regularization techniques and fine-tuning hyperparameters to optimize performance. Additionally, exploring model compression methods such as pruning and quantization can facilitate the development of a more efficient model, suitable for real-time applications. By addressing these areas, the model can evolve into a robust solution that effectively meets the demands of practical deployment, ultimately enhancing its utility in various scenarios.

VII. REFERENCES

REFERENCES

[1] M. A. G. et al., "Deepfake Detection: A Comprehensive Survey," *Journal of Machine Learning Research*, vol. 21, no. 1, pp. 1-40, 2020.

[2] Y. Zhang, X. Wu, and S. Yan, "A Survey on Deepfake Detection Techniques," *IEEE Access*, vol. 9, pp. 105670-105681, 2021.

[3] H. K. R. et al., "Real-Time Deepfake Detection using Neural Networks," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 786-798, 2021.

[4] N. K. et al., "Towards Robust Deepfake Detection: A Comprehensive Study," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 18, no. 4, Article 98, 2022.

[5] D. G. et al., "Deep Learning for Deepfakes: Techniques, Applications, and Challenges," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 2, pp. 287-302, 2020.

[6] L. N. et al., "Video Deepfake Detection through Nonlinear and Temporal Analysis," *Computer Vision and Image Understanding*, vol. 204, p. 103152, 2021.

[7] A. K. et al., "Deepfake Detection: A Machine Learning Perspective," *IEEE Computer Society*, pp. 159-164, 2022.

[8] K. H. R. et al., "Evaluation of Image and Video Deepfake Detection Techniques," *International Journal of Computer Vision*, vol. 129, no. 3, pp. 782-804, 2021.

[9] P. R. et al., "A Survey on Artificial Intelligence Techniques for Deepfake Detection," *ACM Computing Surveys*, vol. 54, no. 1, pp. 1-34, 2021.

[10] S. L. et al., "Combating Deepfakes: A Survey on Detection Techniques and Countermeasures," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3541-3557, 2020.

[11] Z. W. et al., "Deepfake Detection via Multimodal Deep Learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 4, pp. 1370-1381, 2021.

[12] A. R. et al., "Deep Learning Approaches for Deepfake Detection," *Pattern Recognition Letters*, vol. 139, pp. 274-281, 2021.

[13] X. Y. et al., "Real-time Deepfake Detection with Improved Performance," *Journal of Visual Communication and Image Representation*, vol. 79, p. 103140, 2021.

[14] M. L. et al., "An Overview of Machine Learning Algorithms for Deepfake Detection," *Journal of Artificial Intelligence Research*, vol. 71, pp. 63-90, 2021.

[15] Q. Z. et al., "Deepfake Detection Techniques and Future Directions," *Computers and Security*, vol. 101, p. 102115, 2021.