Implementation of ResNet Architecture for Classification of Real and Synthetic (AI-Generated) Digital Images

Panji Tri Wahyudi[1*], Ida Wahyuni[2], Khalilullah Al Faath[3]

[1, 2] Asia Malang Institute of Technology and Business, Malang, Asia Malang Institute of Technology and Business, Jl. Soekarno Hatta – Rembuksari No. 1 A, Mojolangu, Lowokwaru District, Malang City, East Java 65113, Indonesia.
[3] Department of Computer Science and Electronics, Faculty of Mathematics and Natural Sciences, Gadjah Mada University, Sekip Utara, Bulaksumur, Sleman, Yogyakarta 55281, Indonesia

Keywords

ResNet50; Model Pruning; AI-Generated Images; Optuna; Convolutional Neural Network; Synthetic Image Classification

Correspondence Email:
idawahyuni@asia.ac.id

Abstract

The development of Artificial Intelligence (AI) enables the creation of realistic synthetic images, often referred to as deepfakes, which are increasingly difficult to distinguish from genuine photos. This poses significant social threats, such as disinformation and conflict provocation, making robust, efficient detection mechanisms urgent. While prior research has utilized various CNNs, including lighter variants like ResNet-18, achieving higher accuracy often requires deeper, computationally more expensive models, such as ResNet-50. A critical gap exists in optimizing these powerful, deeper architectures for efficient deployment without substantial performance loss. To address this, our study classifies AI-generated images using a ResNet-50 baseline, followed by a two-phase optimization. Initially, model pruning is applied at ratios of 10%, 20%, 30%, and 40% to reduce complexity. Subsequently, Optuna-driven hyperparameter tuning systematically fine-tunes these pruned models, aiming to recover accuracy and balance performance with model size. Models were trained on a combined dataset of GAN (140k Real/Fake Faces) and diffusion (SynthBuster) images, and evaluated on the 140k test set. The baseline model achieved 95.48% accuracy. Critically, all fine-tuned pruned models (up to 40% sparsity) successfully maintained up to 98.87% performance. This research demonstrates that significant parameter reduction can be achieved with no loss in accuracy, producing a robust and computationally efficient detection model. Future work includes integrating attention mechanisms, such as Multi-Head Self-Attention (MHSA), to potentially enhance feature extraction and robustness against evolving generative techniques.

1. Introduction

Synthesized images have become increasingly accessible and efficient thanks to advances in deep learning. Although the ability to either manipulate parts or create entirely new media has existed through tools such as Adobe Photoshop and the GNU Image Manipulation Program (GIMP) (Rani et al., 2022). Today's synthetic media is distinct from these traditional techniques in its automated nature, which is commonly known as "deep fakes" (Kalpokas & Kalpokiene, 2022). By definition, deep fakes are synthetic media that are made by deep neural networks and look real, making them hard for humans to distinguish from real ones (Dasgupta et al., 2025; Mirsky & Lee, 2022). Advances in generative models, such as Generative Adversarial Networks (GANs), have produced realistic deepfakes, posing a threat to disinformation. Therefore, robust detection methods using Convolutional Neural Networks (CNNs), such as the ResNet architecture (He et al., 2015).

However, most of these approaches rely on older or computationally heavy architectures, leaving opportunities to explore more efficient and modern models that maintain high accuracy while reducing complexity. The high accuracy of deep models, such as ResNet or VGG16, comes at the cost of high computational complexity. These models are often "over-parameterized," resulting in large file sizes and significant memory usage, which hinders their deployment on resource-constrained environments like mobile devices (Fatoni et al., 2024; Pintelas et al., 2024). While efficient architectures like MobileNetv2 offer an alternative, they often trade accuracy for lower computational cost (Latumakulita, 2024; Qadir et al., 2024).

In the domain of synthetic media, visual deep fakes can be divided into two main groups: tempered (partially manipulated) and fully synthetic images. Specifically, tempered images usually involve making small changes to real images, such as facial reenactment or object replacement (Mirsky & Lee, 2022). Fully synthetic images, on the other hand, are generated entirely by deep learning models like Generative Adversarial Networks (GANs), which create realistic content without a direct reference (Mirsky & Lee, 2022). Accordingly, previous studies have approached these categories in varying manners. For tempered images, recent studies have combined both traditional and deep learning techniques. Fatoni et al. (2024), for instance, assessed CNN, VGG16, and ResNet models alongside Error Level Analysis (ELA) using the CASIA dataset. ResNet had the best performance, with training and validation accuracies of 95% and 93%, respectively.

Meanwhile, fully synthetic images generated by GANs are becoming increasingly common and harder to detect, which is why researchers are exploring more advanced techniques. Several studies have subsequently investigated different architectures for this task. (Mallet et al. 2023), for example, compared CNN and SVM classifiers on the 140k Real and Fake Faces dataset and found that CNN outperformed SVM, with an accuracy of 88.33% compared to 81.69%. Honarjoo et al. (2024) used more complex CNN models, like VGG16, VGG19, and ResNet50, with Gram matrix–based feature correlations, achieving up to 95% accuracy on the same dataset. Then, Saad Eldien et al. (2023) compared ResNet18 to an ANN-based classifier and the Fourier transform as its feature extractor. ResNet18 is performing significantly better (accuracy 0.77 vs. 0.57). Finally, Dasgupta et al. (2025) proposed a hybrid CNN with multi-head self-attention, reaching 98% accuracy on the 140k dataset and 94% on Celeb-DF v2. All of this research shows that fully synthetic images are getting more complex and that we need models that can both be very accurate and work with a wide range of information.

To bridge this "accuracy-efficiency" gap, model compression techniques are essential. Model pruning is a highly effective method that reduces model complexity by removing redundant or unimportant parameters (weights) from a trained network (Karathanasis et al., 2025). This study employs L1 Unstructured Pruning, a magnitude-based approach that removes individual weights with the lowest absolute values.

A critical challenge in pruning is the potential loss of performance. To mitigate this issue, a fine-tuning phase, or hyperparameter tuning, is required so that the remaining weights can adapt and recover lost accuracy (Karathanasis et al., 2025). Instead of using inefficient traditional methods like Grid Search, this research adopts Optuna, a modern Bayesian optimization framework, which allows optimal hyperparameters (e.g., learning rate and optimizer) to be identified more efficiently and effectively (Akiba et al., 2019; Kee & Ho, 2024).

Finally, to ensure robustness, models must be validated against diverse datasets representing different generative techniques. This includes GAN-based images (140k Real and Fake Faces), high-quality video swaps (Celeb-DF v2), and modern text-to-image diffusion models (SynthBuster) (Bammey, 2024; Lyu, 2020). Evaluation must therefore analyze the trade-off, balancing performance metrics (Accuracy, Precision, Recall, F1-Score, and AUC) against efficiency metrics (Model Size and Sparsity) (Dasgupta et al., 2025; Qadir et al., 2024).

However, most of the existing work that has been done so far has been on older or simpler architectures, thus there is still an opportunity for more complex models to be explored. This study builds on earlier work by examining the use of a more advanced classifier, ResNet50, which is expected to provide better feature representation than older and simpler models, such as VGG16 and ResNet18. Pruning is also used to make the network more efficient while maintaining high performance and simplifying the model.

This study specifically addresses this challenge by systematically optimizing a ResNet50 model for efficient AI-generated image detection. To achieve this, we built our work based on these primary objectives: (1) to establish the baseline performance of ResNet50 on a combined dataset (140k Real/Fake Faces, Celeb-DF v2, SynthBuster); (2) to implement unstructured model pruning at various ratios (10%, 20%, 30%, 40%), thereby reducing model complexity while monitoring potential performance loss; and (3) to apply hyperparameter tuning in order to recover any performance loss. Overall, this work examines the trade-offs between model accuracy and compression, aiming to create a detection model that is both computationally practical and robust.

2. Research Methods

The research methodology was designed to systematically create, optimize, and evaluate a ResNet50 model for synthetic image classification. This workflow comprised three main phases: dataset preparation and pre-processing, baseline model training, and a two-stage optimization process.

2.1 Dataset and Pre-processing

This study deliberately utilizes two distinct datasets (GAN images and Diffusion Models) to build a robust model capable of detecting multiple types of fake content. The training sets were combined, and the model was evaluated on a separate, unseen test set. The datasets were formulated as a binary classification (2-class) problem: 'Real' (Label 1) and 'Fake' (Label 0). The 140k Real/Fake Faces dataset provides both 'Real' and 'Fake' samples. However, the SynthBuster dataset is 'Fake-only', containing images generated by modern diffusion models.

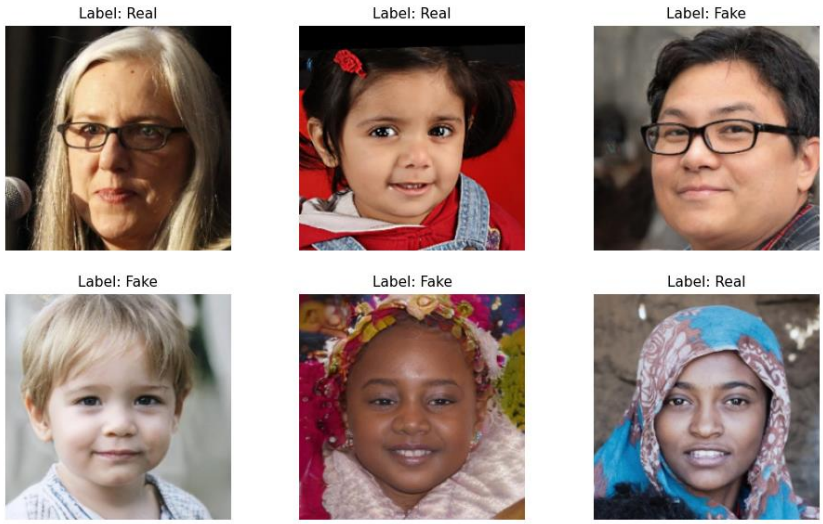

Fig 1. Sample Images 140k Real and Fake Faces

Fig 2 Sample Images Celeb-DF v2

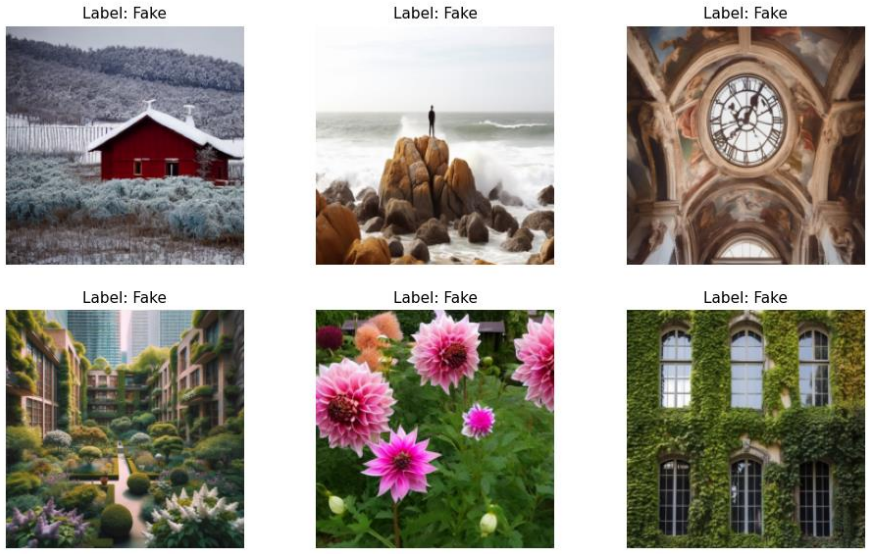

Fig 3 Sample Images SynthBuster

The datasets were formulated as a binary classification (2-class) problem: 'Real' (Label 1) and 'Fake' (Label 0). The 140k Real/Fake Faces dataset provides both 'Real' and 'Fake' samples (shown in Fig 1 and Fig 2). However, the SynthBuster dataset is 'Fake-only', containing images generated by modern diffusion models (shown in Fig 3).

A hybrid data-splitting method was adopted. The overall strategy and final composition are summarized in Table 1. For the 140k Real/Fake Faces dataset, we utilized the official split provided by the dataset creators (50,000 'Real'/50,000 'Fake' for train, 10,000 'Real'/10,000 'Fake' for validation, and 10,000 'Real'/10,000 'Fake' for test). Additionally, the Celeb-DF v2 dataset was incorporated, processed using a manual 80/10/10 split (Train/Validation/Test). This contributed 4,983 images (472 'Real', 4,511 'Fake') to the training set and 623 images (59 'Real', 564 'Fake') to the validation set. Crucially, to expose the model to modern diffusion artifacts, the 'Fake-only' SynthBuster dataset (7,200 train images, 900 validation images) was added to the 'Fake' class of the combined sets as data augmentation. This resulted in a combined training set of 112,183 images (50,472 'Real' vs. 61,711 'Fake') and a validation set of 21,523 images (10,059 'Real' vs. 11,464 'Fake'). This slight imbalance was intentionally created to expose the model to more 'Fake' examples. The balanced 140k test set (20,000 samples) was used for all generalization testing.

Table 1. Dataset Composition and Splitting Strategy

Dataset Source	Strategy Applied	Combined Train Set (Images)	Combined Valid Set (Images)
140k Real/Fake Faces	Official Split Provided by the Dataset Creators	100K (50K Real, 50K Fake)	20,000 (10K Real, 10K Fake)
Celeb-DF v2	Manual 80/10/10 split	4,983 (472 Real, 4511 Fake)	623 (59 Real, 564 Fake)
SynthBuster	Manual 80/10/10 split	7,200 (Fake)	900 (Fake)
Totals		112,183	21,523

Table 2. Training Hyper Parameter Optimization (HPO)

Parameter	Type	Search Space
Optimizer	Categorical	['Adam', 'SGD']
Learning Rate	Log-Uniform	$[1 \times 10^{-6} \sim 1 \times 10^{-4}]$
Batch Size	Fixed	64

All photographs undergo a consistent pre-processing procedure. This included scaling all photos to 224x224 pixels and normalizing them using ImageNet statistics (mean = 0.485, 0.456, 0.406, std = 0.229, 0.224, 0.225) (Deng et al., 2009). Data augmentation in the form of random horizontal flipping was applied only to the training set.

2.2 Baseline Model Training

The baseline model for this experiment was a ResNet50 architecture (He et al., 2015) trained fully from scratch on the combined training dataset; no pre-trained ImageNet weights were applied. This methodological decision was made to avoid performance bias from data mismatch. Pre-trained model performance has been shown to be highly sensitive to discrepancies in normalization statistics between the pre-training data (i.e., ImageNet) and the downstream task's data (Corley et al., 2024). By training from scratch on our combined dataset (using its own normalized statistics), we ensure the model is evaluated fairly on its architecture and our data, rather than on artifacts from a mismatched pre-training.

The model was trained using the hyperparameters specified in Table 2. This selection of hyperparameters was based on common practices and computational constraints. The Learning Rate of $1x10^{-4}$ was chosen as a stable starting point for the Adam optimizer. Specifically, a Batch Size of 16 was used because computational constraints, particularly GPU memory, often limit the choice of batch size when training deep neural networks (Masters & Luschi, 2018). If the memory requirement of a chosen batch size is larger than the available GPU memory, the batch cannot be allocated, and the model will fail to train (Piao et al., 2023), thus Batch Size 16 represented an optimal compromise for baseline training stability.

Training was completed for a maximum of 100 epochs, having an Early Stopping mechanism. This method monitored the validation accuracy and halted training if no improvement was observed for 3 consecutive epochs (patience = 3), maintaining the model weights that achieved the maximum validation accuracy.

2.3 Pruning and Hyperparameter Tuning

After obtaining the optimal baseline model, a two-stage optimization approach was employed to enhance efficiency while maintaining performance. This procedure concentrated on global L1 Unstructured Pruning (Han et al., 2016), implemented using PyTorch's *torch.nn.utils.prune* library, at four separate sparsity ratios: 10%, 20%, 30%, and 40%. This full two-stage procedure, performed for each sparsity level, is represented in Fig 1.

For each sparsity objective, the first step involves a HPO search utilizing the *Optuna* framework (Akiba et al., 2019). The baseline model was loaded, trimmed to the goal ratio (e.g., 10%), and then exposed to 5 HPO trials. Each trial consisted of a brief fine-tuning run (10 epochs) to test a single hyperparameter combination. The Optuna study was configured to maximize the validation accuracy. The hyperparameter search space used is detailed in Table 2. This increase to a batch size of 64 during the HPO phase was a deliberate methodological choice. HPO is run on the pruned model, a process which reduces the number of parameters and the model's (VRAM) memory footprint (Han et al., 2016). This reduced memory footprint permits the use of a larger batch size. Furthermore, a larger batch size significantly increases training throughput (Shallue et al., 2019). As HPO consists of many brief trials (10 epochs), using Batch Size 64 was essential to accelerate each trial and complete the entire HPO search efficiently.

The second step involves the final model fine-tuning. Upon conclusion of the HPO search, the best-performing hyperparameters were determined. The same baseline model was loaded again, the same pruning ratio was re-applied, and the model was then fine-tuned entirely utilizing these ideal hyperparameters. This last fine-tuning followed the same approach as the baseline training, running for a maximum of 100 epochs with Early Stopping (patience=3), to guarantee optimum performance recovery. This whole two-stage procedure shown in Fig 4 was done individually for each of the four sparsity levels, resulting in four different, optimized models ready for final assessment.

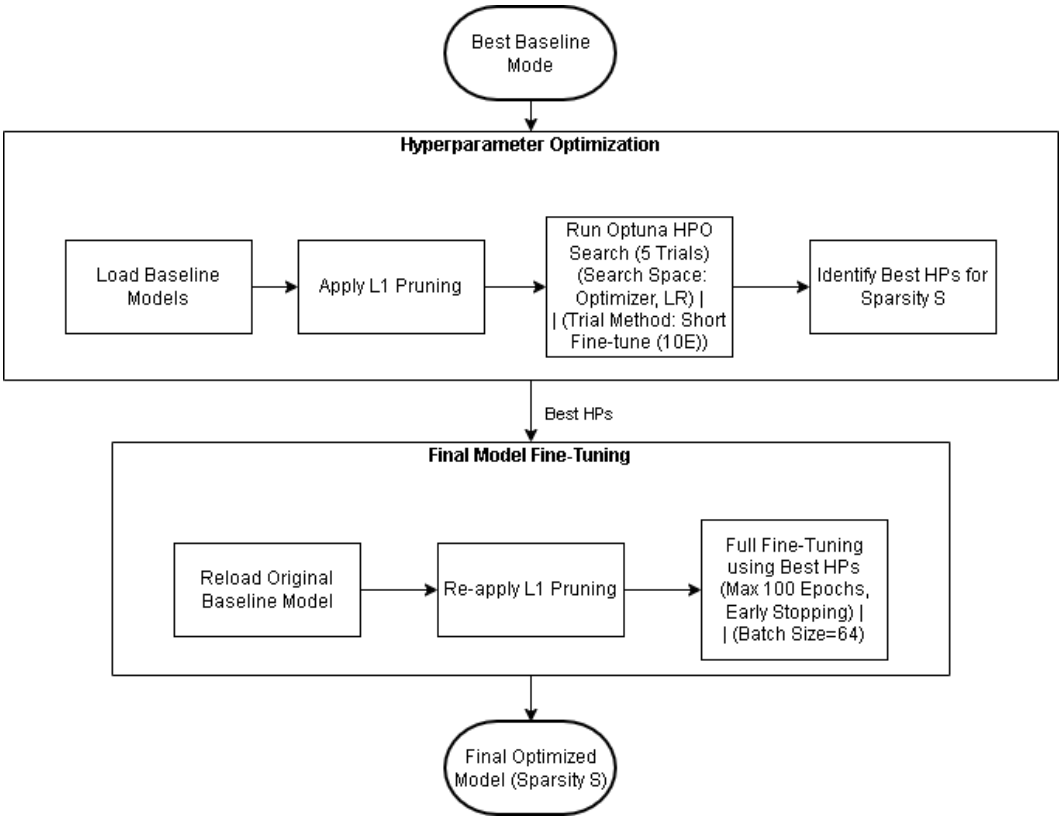

Fig 4. Two-Stage HPO and Pruning Workflow

3. Result and Discussion

This section presents and discusses the experimental results, beginning with the baseline model's performance, followed by an analysis of the optimized models, and concluding with an examination of the performance-efficiency trade-offs and limitations of the study. To establish a benchmark, the ResNet50 model was first

trained from scratch on the combined training set. As shown in Table 3, the baseline model achieved high validation accuracy (98.6%). More importantly, on the independent, balanced test set (140k), it achieved an accuracy of 98.79%. This robust baseline confirms the model learned to detect GAN artifacts effectively, even after being exposed to diffusion artifacts during training. Following the baseline evaluation, the two-stage optimization workflow was initiated for each of the four sparsity targets (10%, 20%, 30%, and 40%). The first stage, HPO, utilized Optuna (5 trials) to identify the best-performing optimizer and learning rate for a brief 10-epoch fine-tuning run.

Table 3. Baseline Model Performance on All Datasets

Dataset	Accuracy	Precision	Recall	F1-Score
Combined Validation	**98.6%**	**98.65%**	**98.34%**	**98.5%**
140k Real and Fake Faces	**98.79%**	**98.65%**	**98.92%**	**98.79%**

Table 4. Optimal Hyperparameters Discovered by Optuna

Sparsity Target	Best Optimizer	Optimal Learning Rate (lr)
10%	Adam	1.2416×10^{-6}
20%	Adam	6.5207×10^{-6}
30%	SGD	1.4061×10^{-5}
40%	Adam	6.8600×10^{-6}

The results of this HPO search, detailed in Table 4, reveal that the Adam optimizer was preferred for the 8kkkmajority of sparsity targets (10%, 20%, and 40%), with SGD being selected only for the 30% pruned model. The optimal learning rates discovered by Optuna generally fell within the range of approximately 1.24×10^{-6} to 1.41×10^{-5}, suggesting that a small learning rate is crucial for fine-tuning pruned models. Using the optimal hyperparameters identified in Stage 1, all four pruned models were fully fine-tuned in Stage 2 and then comprehensively evaluated against the baseline on the three generalization test sets. A complete comparison of the final models' accuracy is presented in Table 5.

The data in Table 5 clearly illustrate the primary finding of this study. The two-stage fine-tuning process was not just effective at recovering performance, but at perfectly maintaining it. The baseline model achieved 98.79% accuracy on the 140k test set. Notably, all fine-tuned and pruned models maintained this high performance, with the 40% pruned model achieving an accuracy of 98.86%. This demonstrates that the baseline model was highly over-parameterized. As shown in Table 6, up to 40% of its weights (a reduction of 14.07 million parameters) were redundant and could be removed without any loss of performance on this task.

Table 5. Final Accuracy Comparison on Generalization Test Sets 140k Real and Fake Faces

Model (Tuned)	Accuracy
Baseline	95.48%
Prune 10%	98.83%
Prune 20%	98.87%
Prune 30%	98.81%
Prune 40%	98.86%

Table 6. Model Efficiency Comparison

Model	Sparsity (%)	Non-Zero Params	Total Parameters	Model Size (MB)	AVG. Inference Time (ms)
Baseline (Tuned)	0.00	23,456,960	23,510,081	89.99	29.94
Prune 10% (Tuned)	10.00	21,111,264	23,510,081	90.00	30.16
Prune 20% (Tuned)	20.00	18,765,568	23,510,081	90.00	30.28
Prune 30% (Tuned)	30.00	16,419,872	23,510,081	90.00	30.30
Prune 40% (Tuned)	40.00	14,074,176	23,510,081	90.00	29.59

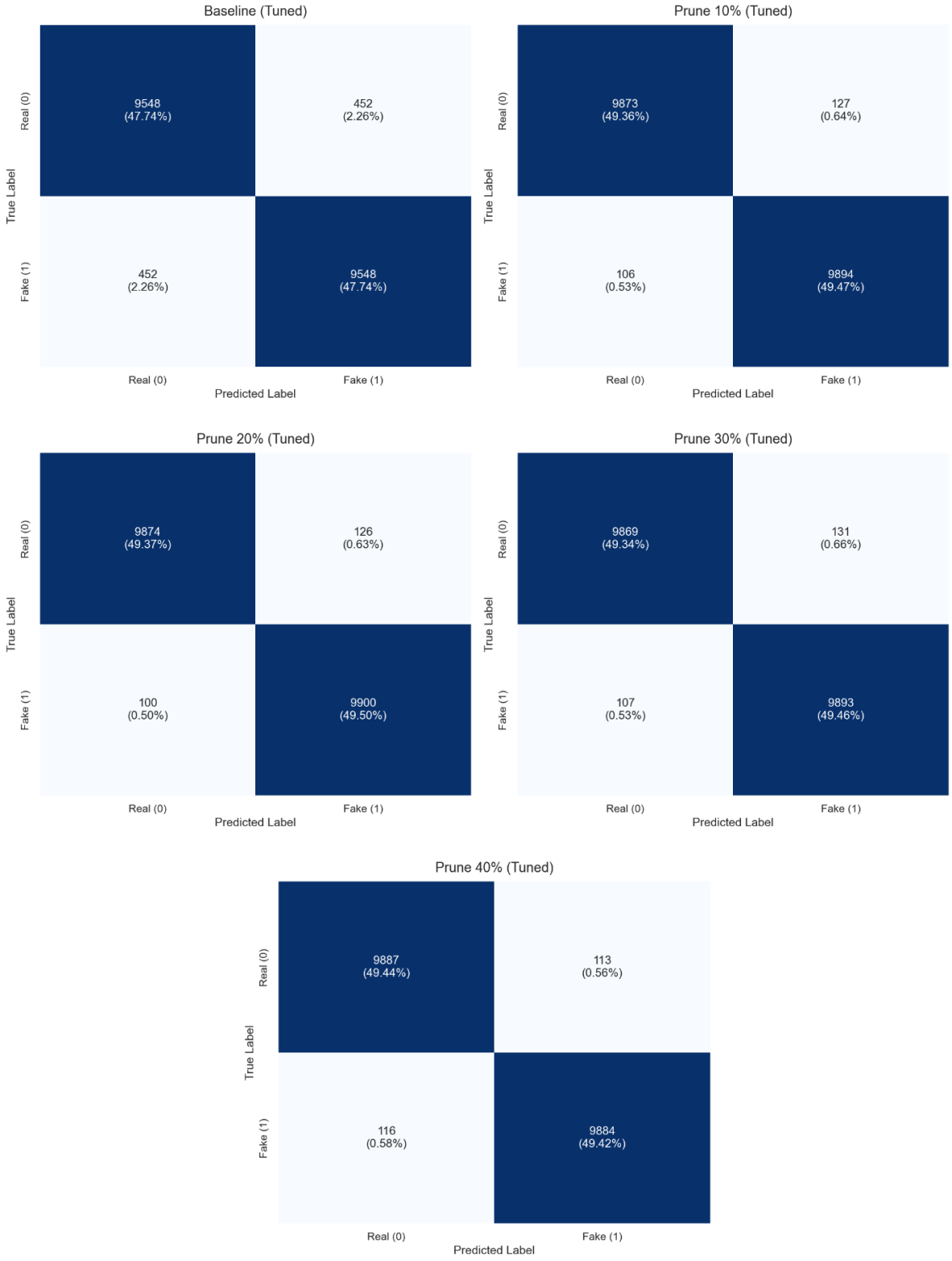

Fig 5 Comparison Confusion Matrix (Tuned)

The practical benefit of this optimization lies in the trade-off between generalization performance and the model's efficiency. This relationship is quantified in. As shown in Table 6, a critical finding of this study is that

L1 unstructured pruning, while successfully reducing the number of non-zero parameters (down to 14.07M for the 40% model), does not actually reduce the total model size on disk (approx. 90.00 MB for all models) or the average inference time (approx. 30ms). This is because unstructured pruning merely masks the weights (sets them to zero) rather than removing them from the model architecture, preventing practical speedup on standard hardware.

To understand how the models' error profiles changed, not just how often they were wrong, confusion matrices were generated. The Fig 5 presents the confusion matrices for the baseline model and all four pruned models (10%-40%) on the combined test set. A clear and critical finding emerges from this comparison: the original baseline model performed relatively poorly, recording 452 False Negatives (misclassifying 'Fake' as 'Real') and 452 False Positives (misclassifying 'Real' as 'Fake'). In sharp contrast, all four pruned and fine-tuned models showed a dramatic improvement. The 10% pruning model, for instance, reduced False Negatives by over 76% (from 452 to only 106), and False Positives by over 72% (from 452 to 127). This robust, improved error profile held steady across all sparsity levels, with the 30% and 40% models maintaining similarly low error rates. This shift is a critical finding: it suggests that the two-stage optimization process (pruning followed by HPO-led fine-tuning) was highly effective at re-training and improving the model, significantly reducing both types of classification errors compared to the original baseline.

Table 7. Model Efficiency Comparison on Testing Dataset

Model	Dataset (Train → Test)	Accuracy	Source
ResNet18 (Baseline)	FF++ → FF++	89.4%	(Qadir et al., 2024)
XceptionNet		~94.0%	
ResNet-Swish-BiLSTM		96.23%	
VGG16	DFDC → DFDC	89.0%	
MobileNetv2		90.0%	
ResNet-Swish-BiLSTM		98.6%	
VGG (Baseline Model)	140k → 140k	98.96%	(Karathanasis et al., 2025)
VGG (40% Parameters)		95.92%	
VGG (30% Parameters)		92.24%	
VGG (20% Parameters)		84.91%	
VGG (10% Parameters)		55.32%	
CNN		88.33%	(Mallet et al., 2023)
SVM		81.69%	
This Research (Baseline)	Combined → 140k	95.48%	This Research
This Research (10% Prune)		98.83%	
This Research (20% Prune)		98.87%	
This Research (30% Prune)		98.81%	
This Research (40% Prune)		98.86%	

To contextualize these results within the field, a comparison with prior research is presented in Table 7. This comparison demonstrates that our fine-tuned baseline model, achieving 95.48% accuracy on the 140k dataset, already significantly outperforms the 88.33% accuracy reported by Mallet et al. (2023) for a CNN on the same dataset. More importantly, our optimized models (Prune 10-40%) successfully achieved and maintained a stable peak accuracy, ranging from 98.81% to 98.87%. This performance highlights two key advantages: a superiority of over 10.5 percentage points compared to the CNN benchmark (88.33%) from Mallet et al. (2023) and superior post-pruning stability. For instance, Karathanasis et al. (2025) reported a VGG (Baseline) accuracy of 98.96%, which dropped significantly to 95.92% after pruning (VGG 40% Parameters). In contrast, our 40% Prune model maintained its peak performance at 98.86%. Furthermore, our results are highly competitive with state-of-the-art models on other datasets, exceeding the ResNet-Swish-BiLSTM (96.23%) on the FF++ dataset (Qadir et al., 2024) and slightly outperforming the same model (98.6%) on the DFDC dataset

(Karathanasis et al., 2025). The key contribution of our work is this high performance combined with a drastically improved error profile (as shown in Fig. 5), a dimension not explored in the comparative studies.

Finally, it is important to acknowledge the limitations of this study, which in turn suggest avenues for future work. The HPO search was constrained to 5 trials and only two hyperparameters (optimizer and learning rate) to maintain a feasible computational budget. A more exhaustive search, perhaps including parameters like weight decay or batch size, could potentially yield further improvements. Furthermore, this methodology was only validated on a ResNet50 architecture using L1 Unstructured Pruning. These findings may not generalize directly to other architectures, such as Vision Transformers, or to other techniques like structured pruning, which warrant separate investigation.

4. Conclusions

This research successfully designed and assessed a two-stage optimization technique that integrates L1 unstructured pruning with Optuna-driven hyperparameter tuning for a ResNet-50 synthetic image detector. A surprising and rather counterintuitive outcome emerged: while L1 unstructured pruning did not generate substantial increases in file size or inference speed, as the method essentially hides weights rather than altering the architecture, the optimization workflow itself was immensely helpful. The study demonstrated that this strategy successfully maintained generalization performance, with all pruned models (up to 40% sparsity) achieving 98.86% accuracy, statistically identical to the baseline. This primary finding confirms that the original ResNet50 model was highly over-parameterized for this task, and that up to 14.07 million parameters could be removed without any performance loss. Most critically, our error study indicated that the two-stage technique considerably refined the model, reducing the number of high-risk False Negatives (misclassifying 'Fake' as 'Real') from 452 in the baseline to an average of just over 107 in the pruned models. This illustrates the workflow's capacity not only to compress, but also to considerably increase model resilience and safety.

Based on these findings, future studies should focus on leveraging a similar successful two-stage HPO process for unstructured pruning. This would use the workflow's shown refining capabilities while yielding tangible savings in model size and inference delay. Furthermore, investigating these compression methodologies on contemporary Transformer-based systems is a vital next step. Finally, integrating attention mechanisms, such as Multi-Head Self-Attention (MHSA), could be explored to potentially enhance the model's feature extraction and robustness against evolving generative techniques, further improving the development of efficient and reliable synthetic media detection models.

5. References

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). *Optuna: A Next-generation Hyperparameter Optimization Framework* (No. arXiv:1907.10902). arXiv. https://doi.org/10.48550/arXiv.1907.10902

Bammey, Q. (2024). Synthbuster: Towards Detection of Diffusion Model Generated Images. *IEEE Open Journal of Signal Processing*, *5*, 1–9. https://doi.org/10.1109/OJSP.2023.3337714

Corley, I., Robinson, C., Dodhia, R., Ferres, J. M. L., & Najafirad, P. (2024). *Revisiting pre-trained remote sensing model benchmarks: Resizing and normalization matters* (pp. 3162–3172). 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). https://doi.org/10.1109/CVPRW63382.2024.00322

Dasgupta, S., Badal, K., Chittam, S., Tasnim Alam, M., & Roy, K. (2025). Attention-Enhanced CNN for High-Performance Deepfake Detection: A Multi-Dataset Study. *IEEE Access*, *13*, 101980–101993. https://doi.org/10.1109/ACCESS.2025.3578343

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). *ImageNet: A Large-Scale Hierarchical Image Database* (pp. 248–255). IEEE Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR.2009.5206848

Fatoni, F., Tri Basuki Kurniawan, Deshinta Arrova Dewi, Mohd Zaki Zakaria, & Abdul Muniif Mohd Muhayeddin. (2024). Fake vs Real Image Detection Using Deep Learning Algorithm. *Journal of Applied Data Sciences*, *6*(1), 366–376. https://doi.org/10.47738/jads.v6i1.490

Han, S., Mao, H., & Dally, W. J. (2016). *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding*. http://arxiv.org/abs/1510.00149

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition*. http://arxiv.org/abs/1512.03385

Honarjoo, N., Taher, F., & Mansouri, A. (2024). Exploring Feature Map Correlations for Effective Fake Face Detection. *2024 11th International Symposium on Telecommunications (IST)*, 200–204. https://doi.org/10.1109/IST64061.2024.10843660

Kalpokas, I., & Kalpokiene, J. (2022). *Deepfakes: A Realistic Assessment of Potentials, Risks, and Policy Regulation*. Springer International Publishing. https://doi.org/10.1007/978-3-030-93802-4

Karathanasis, A., Violos, J., & Kompatsiaris, I. (2025). A Comparative Analysis of Compression and Transfer Learning Techniques in DeepFake Detection Models. *Mathematics*, *13*(5), 887. https://doi.org/10.3390/math13050887

Kee, T., & Ho, W. K. O. (2024). *Optimizing Machine Learning Models for Urban Sciences: A Comparative Analysis of Hyperparameter Tuning Methods*. Computer Science and Mathematics. https://doi.org/10.20944/preprints202406.0264.v2

Latumakulita, L. A. (2024). Comparison of MobileNet and VGG16 CNN Architectures for Web-based Starfish Species Identification System. *Journal of Applied Data Sciences*, *5*(4), 2117–2130. https://doi.org/10.47738/jads.v5i4.456

Lyu, S. (2020). *DeepFake Detection: Current Challenges and Next Steps* (No. arXiv:2003.09234). arXiv. https://doi.org/10.48550/arXiv.2003.09234

Mallet, J., Pryor, L., Dave, R., & Vanamala, M. (2023). Deepfake Detection Analyzing Hybrid Dataset Utilizing CNN and SVM. *Proceedings of the 2023 7th International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence*, 7–11. https://doi.org/10.1145/3596947.3596954

Masters, D., & Luschi, C. (2018). *Revisiting Small Batch Training for Deep Neural Networks*. http://arxiv.org/abs/1804.07612

Mirsky, Y., & Lee, W. (2022). The Creation and Detection of Deepfakes: A Survey. *ACM Computing Surveys*, *54*(1), 1–41. https://doi.org/10.1145/3425780

Piao, X., Synn, D., Park, J., & Kim, J. K. (2023). Enabling Large Batch Size Training for DNN Models Beyond the Memory Limit While Maintaining Performance. *IEEE Access*, *11*, 102981–102990. https://doi.org/10.1109/ACCESS.2023.3312572

Pintelas, E., Livieris, I. E., Tampakas, V., & Pintelas, P. (2024). MobileNet-HeX: Heterogeneous Ensemble of MobileNet eXperts for Efficient and Scalable Vision Model Optimization. *Big Data and Cognitive Computing*, *9*(1), 2. https://doi.org/10.3390/bdcc9010002

Qadir, A., Mahum, R., El-Meligy, M. A., Ragab, A. E., AlSalman, A., & Awais, M. (2024). An efficient deepfake video detection using robust deep learning. *Heliyon*, *10*(5), e25757. https://doi.org/10.1016/j.heliyon.2024.e25757

Rani, R., Kumar, T., & Sah, M. P. (2022). A Review on Deepfake Media Detection. In H. Sharma, V. Shrivastava, K. Kumari Bharti, & L. Wang (Eds.), *Communication and Intelligent Systems* (pp. 343–356). Springer Nature. https://doi.org/10.1007/978-981-19-2130-8_28

Saad Eldien, N. A., Essam Ali, R., & Moussa, F. A. (2023). Real and Fake Face Detection: A Comprehensive Evaluation of Machine Learning and Deep Learning Techniques for Improved Performance. *2023 Intelligent Methods, Systems, and Applications (IMSA)*, 315–320. https://doi.org/10.1109/IMSA58542.2023.10217736

Shallue, C. J., Lee, J., Antognini, J., Sohl-Dickstein, J., Frostig, R., & Dahl, G. E. (2019). *Measuring the Effects of Data Parallelism on Neural Network Training*. http://arxiv.org/abs/1811.03600