

Deepfake Detection Using Transfer Learning: An Approach to Distinguish AI-Generated and Camera Trap Images

Rakhi Kumari ^a, Pooja Meena ^a, Sadhana^a, Shweta Sharma^a, Sweeti Sah^a, Kamaldeep^b,
Manisha Malik^b, Sachi Nandan Mohanty^c

^aDepartment of Computer Science and Engineering, National Institute of Technology, Kurukshetra, Haryana, India

^bDepartment of Media Engineering, National Institute of Technical Teachers Training and Research, Chandigarh, India

^cSchool of Computer Science and Engineering (SCOPE), VIT-AP University, Amaravati, Andhra Pradesh, India

Email addresses: rakhiues04@gmail.com, poojamp450@gmail.com, sadhanabohariya@gmail.com, shweta.sharma@nitkkr.ac.in,
sweetisah3@nitkkr.ac.in, kamal.katyal@yahoo.com, manishamalik53@gmail.com, sachinandan09@gmail.com

Abstract: "Deepfake" technology, a recent development in artificial intelligence (AI), has become increasingly prevalent on social media and involves superimposing one person's face onto another's. This technology is often misused to generate explicit images and videos without consent, disproportionately targeting women and causing significant harm, including harassment, exploitation, reputational damage, and psychological distress. In response to these growing concerns, this research aims to distinguish between AI-generated deepfake images and authentic camera trap images. Our approach generates AI-generated datasets using an AI image generator for each stage of the process. Specifically, this work utilizes a dataset comprising 6,000 images, evenly split into 3,000 AI-generated and 3,000 camera-trapped images. The dataset is further divided into training, validation, and testing sets in an 8:1:1 ratio, with all images resized to a standard dimension. To enhance model generalization, data augmentation techniques are applied across all datasets. We employ several transfer learning models based on Convolutional Neural Network architectures, including Inception-v3, AlexNet, and MobileNet, to enhance the precision and effectiveness of image classification. By fine-tuning these models using supervised deep-learning techniques, we enable them to detect subtle differences in texture, color distribution, and high-level features unique to deepfakes. The experimental results show that MobileNet achieves superior performance with 95% accuracy, outperforming Inception-v3 and AlexNet. Furthermore, our approach surpasses existing methods in the literature, providing a more accurate and reliable solution for detecting deepfakes.

Keywords: DeepFake; Deep Learning; Transfer Learning; Real images; Data Augmentation; Classification

1. Introduction

With the rise of Artificial intelligence (AI), new software has emerged that can create highly realistic fake images. Criminals are exploiting this technology to their advantage as these tools allow anyone to generate fake images with minimal coding, and the results often look almost indistinguishable from real camera photographs. The increasing prevalence of AI-generated images and their challenges in various applications, including media, social networks, and digital security pose a serious security risk. As AI-generated images become more realistic, distinguishing them from camera-captured images becomes critical to maintaining trust and authenticity in visual media. Therefore, it is required to develop a detection system that is powerful, scalable, adaptable, and capable of precisely classifying images that are acquired by cameras and generated by AI through the application of deep learning methodologies.

Moreover, recent advances in architectures like Generative Adversarial Networks (GANs) [1][2] have made DeepFake generation considerably easier. Now, it just needs a source image and a set of desired distortions to produce convincingly altered images. For instance, in the year 2024 [3], a case was reported wherein AI-generated photos of American singer Taylor Swift circulated on social media, highlighting the potentially harmful implications of the use of AI technology. In the same year, a finance employee of a global organization was duped into giving \$25 million to scammers by utilizing deepfake technology to serve as the company's chief financial officer during a video conference call [4]. In the year 2022, scammers exploited AI-generated photos to benefit from the earthquake

that struck Turkey and Syria [5]. They published fictitious pictures of kids online to gain sympathy and compassion, then used this to steal money and the victims' credentials. Every Deepfake that was made public was sexual and mostly directed at people working in the entertainment business [6].

Therefore, this research proposes a novel method for detecting deepfake facial images using several transfer learning models, including MobileNet, AlexNet, and Inception-v3. We created a dataset of 2,413 AI-generated images using AI image generator tools and collected 2,426 authentic camera images. Data augmentation techniques were applied to enhance the dataset, expanding it to 48,260 AI-generated images and 48,520 camera images. Our approach utilizes these transfer learning models to analyze input images and predict whether each one is real or a deepfake.

1.1 Contributions

The main contributions of our proposed approach are discussed as follows:

- A. Multi-Stage Classification:** Our approach employs a multi-stage classification process encompassing image generation, augmentation, model training, cross-validation, and visualization. This structured methodology ensures thorough analysis and processing of camera and AI-generated images to capture intricate patterns indicative of various categories.
- B. Transfer Learning:** Our approach utilizes transfer learning models, including Inception-v3, AlexNet, and MobileNet, for deepfake image classification. The results demonstrate that MobileNet achieves the highest performance, with 95% accuracy in distinguishing deepfake images from camera-generated ones, surpassing the effectiveness of existing methods in the literature.
- C. Dataset Generation:** Our approach leverages the Microsoft Bing AI image generator [21] and Perchance [20] to create deepfake images that closely resemble camera-generated ones. The dataset is divided into three groups: training, testing, and validation to ensure consistent model performance. Additionally, all images are standardized in dimensions, making them suitable for use in the implemented models.
- D. Dataset Augmentation:** The augmentation techniques used to expand the dataset include rescaling, width and height shifts, flipping, zooming, filling, and rotation. These techniques are applied across various image types to enhance diversity and improve model performance, with linear regression employed to ensure accurate predictions.

The remainder of this paper is organized as follows: Section 2 reviews related work in deepfake detection. Section 3 details the dataset generated using AI image generator tools, along with data augmentation techniques employed to expand the dataset. Section 4 outlines the proposed approach for classifying deepfake and camera trap images. Section 5 presents the experimental setup and the results obtained using transfer learning models. Also, compared our proposed approach with the existing state-of-the-art. Finally, Section 6 provides the conclusion and discusses future work.

2. Related Work

Several researchers have reviewed and developed approaches for detecting deepfake images which are discussed as follows:

Jeon *et al.* [7] utilized a GANs-based dataset and a Deepfake-based dataset and then detected fake images using the attention module in a CNN-based pre-trained model. Several researchers [8], [9] reviewed deep learning models for detecting deepfake human face images and videos. [24] generated images from generative models such as Bing Image Creator, Midjourney, StableDiffusion, and DALL-E 2. They identified shortcomings in detecting deepfakes and highlighted areas for model improvement and developed strategies for detecting AI-generated images.

Adaptive Boosting and extreme Gradient Boosting approaches were merged by Dang *et al.* [10] to handle the imbalance dataset. They implemented the CNN model to identify manipulated faces. Moura *et al.* [11] performed face recognition with the Support Vector Machines. Hsu *et al.* [12] generated a pair of fake and real images with the Generative Adversarial Networks (GANs), and the DenseNet pre-trained model was employed to detect false pictures.

Lewis *et al.* [13] used the Facebook Deepfake Detection Challenge (DFDC) dataset. This dataset was generated by recording videos of actors with diverse backgrounds and applying various deepfake techniques to alter their faces and voices. They applied the VSNet model for deepfake video detection. Mitra *et al.* [14] used the FaceForensics++ and DFDC datasets and applied the Xception net model to detect deepfakes in social media. Xu *et al.* [15] gathered datasets from DeepFake-TIMIT, FaceForensics++, Celeb-df, and Preview. They detected DeepFakes by extracting texture features from video frames using techniques such as gradient analysis, standard deviation, co-occurrence matrices, and wavelet transforms. They applied Support Vector Machines (SVM) to identify deepfakes using texture features.

Other researchers in [16] and [17] generated datasets from Faceforensics++, DeeperForensics-1.0, and DFDC and used the CNN model for deepfake detection. Joshi & V [18] gathered the dataset from the deepfake_faces dataset from Kaggle and employed the Xception model to identify deep fake images and videos. Gura *et al.* [19] gathered the dataset from the DFDC and applied the CNN model for deepfake detection in images.

Table 1 provides a summary of related work on deepfake detection, detailing the dataset, model name, accuracy achieved, and the identified research gaps in each experimental work.

Table 1. Summary of related approaches for deepfake detection

Author	Year	Model	Dataset	Accuracy	Research Gaps
Lewis <i>et al.</i> [13]	2020	VSNet	DFDC	61.95%	Low performance
Mitra <i>et al.</i> [14]	2021	Xception net	FaceForensics++, DFDC	92.33%	Limited Focus on an individual model
Xu <i>et al.</i> [15]	2021	SVM	DeepFake-TIMIT, FaceForensics++, Celeb-df, Preview	94.4%	Dependency on Texture Features
Haseena <i>et al.</i> [17]	2023	CNN	DFDC	94.32%	Frame-level detection
Joshi & V [18]	2024	Xception net	Kaggle deepfake_faces	93.01%	Limited Focus on an individual model
Gura <i>et al.</i> [19]	2024	CNN	DFDC	91.47%	Imbalance dataset

The existing research on deepfake detection reveals several gaps that need to be addressed to improve the robustness and generalizability of detection models. For instance, Suratkar *et al.* [16] used small and limited datasets, with their model being trained on only 3,000 images. Similarly, other researchers in [14] and [18] primarily focused on using the Xception model for deepfake detection. While the Xception model shows promising results, exploring a wider range of models, such as other state-of-the-art architectures (e.g., EfficientNet, Vision Transformers), could provide a more comprehensive evaluation and potentially improve detection performance. Xu *et al.* [15] were

dependent on texture features only. Another gap identified is the reliance on frame-level detection, which limits the scope of current implementations [17]. Additionally, Gura et al. (2023) used imbalanced datasets, which can lead to biased detection results [19].

3. Dataset Description

We manually created our dataset using AI-image-generating tools, such as Perchance [20] and Microsoft Bing AI-image creator [21]. We gathered publicly available camera-trapped images from online sources like Pinterest [22] and Freepik [23]. The dataset contains approx. 6,000 images, evenly split between 3,000 AI-generated images and 3,000 camera-trapped images. These images are further divided into training, validation, and testing sets in an 8:1:1 ratio. The training set comprises 80% of the images from each class (AI-generated and camera-trapped), while the validation and testing set each contains 10% of the images from both classes, resulting in a total of 600 images for each.

After collecting and sorting all the images, we resize them to standardize their dimensions and properties, ensuring consistent evaluation of the models across images with originally varying sizes and characteristics. Examples of AI-generated images [20], [21] and camera-trapped images [22], [23] are shown in Figures 1 and 2, respectively.



Figure 1. AI-generated Images



Figure 2. Camera-trapped Images

3.1 Data Augmentation

We applied data augmentation techniques to expand the dataset and enhance model performance and generalization. Sample images before and after data augmentation are shown in Figures 3 and 4, respectively. After data augmentation, the dataset consists of 96,780 images for training, 600 images for validation, and 600 images for

testing. The details of the datasets used for training, validation, and testing, both before and after data augmentation, are presented in Table 2.



Figure 3. Sample images before data augmentation







Figure 4. Sample images after applying data augmentation

The following are some of the data augmentation techniques applied in this model:

- A. Rescaling:** This normalizes the image pixel values from a range of $[0, 255]$ to $[0, 1]$ by dividing each pixel value by 255, which helps in faster convergence during training.
- B. Rotation:** The `rotation_range=20` parameter randomly rotates images within a range of -20 to $+20$ degrees, allowing the model to become invariant to rotations.
- C. Width and Height Shifts:** The `width_shift_range=0.2` and `height_shift_range=0.2` parameters randomly translate images horizontally and vertically up to 20% of the total width or height, making the model robust to shifts.
- D. Shear Transformation:** The `shear_range=0.2` parameter applies a shear transformation, which slants the shape of the object in the image, adding more variation.
- E. Zoom:** The `zoom_range=0.2` parameter randomly zooms in on images up to 20%, which helps the model handle objects that appear at various distances.
- F. Horizontal Flip:** The `horizontal_flip=True` parameter randomly flips images horizontally, helping the model learn from mirror images.
- G. Fill Mode:** The `fill_mode='nearest'` parameter fills in any newly created pixels after transformations using the nearest pixel values, maintaining the image's appearance.

Table 2. Dataset Description before and after applying data augmentation

Before Augmentation		After Augmentation	
AI-image Sample	Camera Trapped Image Sample	Augmented AI-image Sample	Augmented Camera Trapped Image Sample
			
AI-images Dataset For Training: 2,413 images	Camera images dataset Training: 2,426 images	AI-images Dataset For Training: 48,260 images	Camera images dataset Training: 48,520 images
AI-images Dataset For Testing: 300 images	Camera images Dataset for Testing: 300 images	AI-images Dataset For Testing: 300 images	Camera images Dataset for Testing: 300 images
AI-images Dataset For Validation: 300 images	Camera images Dataset for Validation: 300 images	AI-images Dataset For Validation: 300 images	Camera images Dataset for Validation: 300 images

4. Proposed Approach for Classification of AI-generated and Camera Trap Images

In this experimental study, we present an approach to efficiently classify images as either AI-generated or real, achieving an accuracy of over 95%. The proposed methodology, illustrated in Figure 5, is based on transfer learning using a pre-trained "MobileNet" deep learning model. MobileNet demonstrates high precision and accuracy in distinguishing AI-generated images from camera trap images, outperforming other models such as Inception-v3 and AlexNet.

The model is trained and validated using labeled datasets, employing the ImageDataGenerator for data augmentation techniques, including rotation, shear, zoom, width/height shift, rescaling, and horizontal flipping. This process enhances the model's robustness and accuracy by generating diverse variations of the original images. To further improve performance, we leverage transfer learning, fine-tuning, and a large, enriched dataset. K-Fold cross-validation is employed to ensure reliable and generalized performance across different data subsets. The model is trained on augmented data, evaluated on the validation set, and assessed based on key metrics such as F1 score, accuracy, precision, and recall.

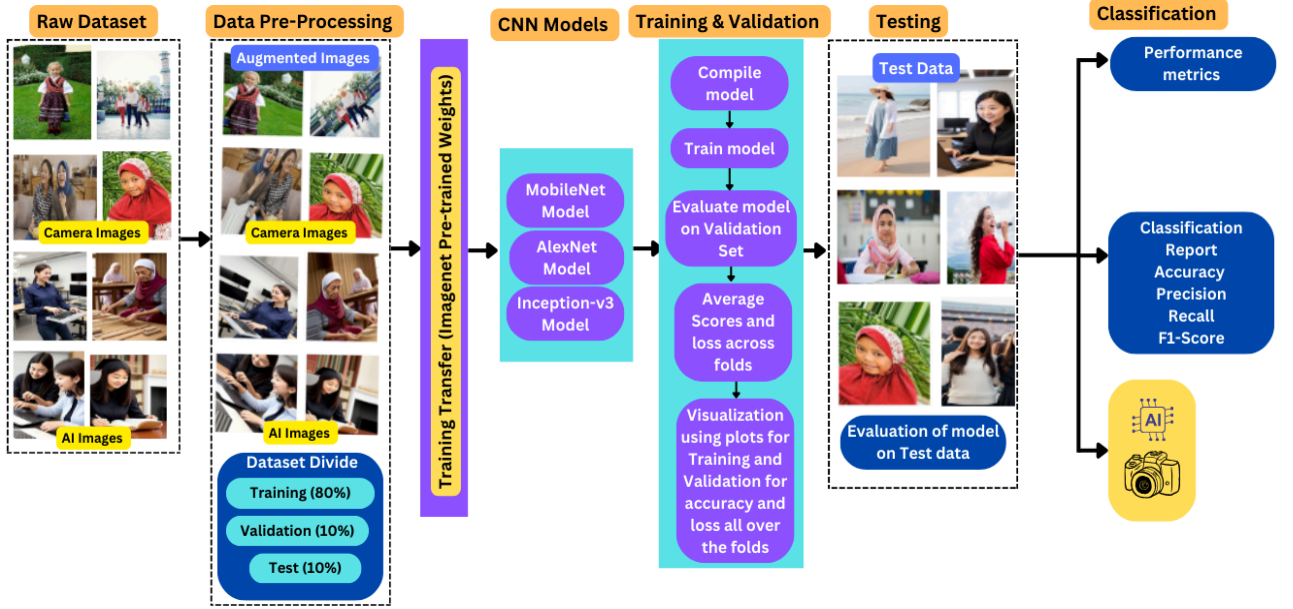


Figure 5. Proposed Approach for AI-generated and Camera Trap Image Classification

The framework depicted in Figure 5 outlines the flow and interaction of the main components involved in the image classification task using k -fold cross-validation and transfer learning with pre-trained models. The process begins with loading raw datasets, which are then augmented and prepared. After resizing and standardizing the images, data augmentation is applied where the training dataset undergoes further augmentation, including transformations such as width and height shifts, flipping, and zooming which expands 4,839 images into 96,780 unique variations, and enhances the dataset for robust model training.

The MobileNet, AlexNet, and Inception-v3 models are then implemented, assembled, and trained using the training datasets, with k -fold cross-validation to store and calculate the accuracy and loss across all folds. After training and cross-validation, the models are evaluated on the test datasets, resulting in a confusion matrix and a classification report detailing the F1-score, recall, precision, and accuracy.

Algorithm 1 shows preprocessing of the dataset by resizing images, applying data augmentation for training, and organizing files and labels into structured arrays, ensuring optimal input for model training and validation. Algorithm 2 outlines a pre-trained model, MobileNet, as a base, enhanced with custom dense and dropout layers for effective feature extraction and classification. It employs k -fold cross-validation to ensure robust model evaluation and tuning, minimizing overfitting and optimizing performance across different data splits.

Algorithm 3 conducts a thorough training and evaluation process by implementing k -fold cross-validation, creating diverse training and validation sets, and generating data from current folders. This approach calculates average accuracy and loss across all folds, ensuring consistent model performance and preventing overfitting. Algorithm 4

provides a comprehensive evaluation of the trained model on a test dataset, accurately measuring its performance using a confusion matrix and classification report.

Algorithm 1: Data Preprocessing

1. **Input:** Dataset $D = \{\text{img}, \text{lbl}\}$; where img are images and lbl are labels ($\text{lbl} \in \{\text{AI}, \text{camera}\}$)
 2. **Output:** Labeled dataset for model training and validation.
 3. **for** each img in D **do**
 $\leftarrow \text{resize}(224, 224, 3)$
 // Augment for training; only rescale for validation and test datasets.
 $\leftarrow \text{augment}(\text{img})$
 4. **function** get_files_and_labels(directory):
 a. **for** idx, className **in** listdir(directory) **do**
 classDir $\leftarrow \text{join}(\text{directory}, \text{className})$
 b. **for** fileName **in** listdir(classDir) **do**
 fileList $\leftarrow \text{append}(\text{classDir}, \text{fileName})$
 labels $\leftarrow \text{append}(\text{idx})$
 c. **return** fileList, labels
 5. *// Retrieve file lists and labels and convert into arrays*
 fileList, labels = get_files_and_labels(trainDir)
 fileList $\leftarrow \text{as_array}(\text{fileList})$
 labels $\leftarrow \text{as_array}(\text{labels})$
-

Algorithm 2. Model Definition and cross validation Setup

1. baseModel $\leftarrow \text{MobileNet}(\text{weights}=\text{imagenet}, \text{include_top}=\text{False}, \text{input_shape}=\text{IpShape})$
 2. $x \leftarrow \text{add_GlobalAveragePooling2D}()(\text{baseModel.output})$
 3. $x \leftarrow \text{add_Dense}(a, \text{activation}=\text{'relu'})(x)$
 4. $x \leftarrow \text{add_Dropout}(d)(x)$
 5. $\text{output} \leftarrow \text{add_Dense}(1, \text{activation}=\text{'sigmoid'})(x)$
 6. model $\leftarrow \text{Model}(\text{inputs}=\text{baseModel.input}, \text{outputs}=\text{output})$
 7. *// Fine-tune base layers*
 for layer **in** baseModel.layers[] **do**
 layer.trainable = False
 8. *// K-fold cross-validation setup*
 kf = k-Fold($n_splits=\text{numFolds}$, shuffle=True, random_state=r)
-

Algorithm 3. Training and cross-validation of the model

1. **Input:** Training and validation Dataset = $\{\text{img}, \text{lbl}\}$ where img are images and lbl are labels ($\text{lbl} \in \{\text{AI}, \text{camera}\}$)
2. **Output:** Average accuracy across all folds on validation and training dataset
3. **for** trainIdx, valIdx **in** kf.split(fileList) **do** *// Repeat for all n folds*
4. *// Create training and validation sets from the image dataset*
 trainFiles, valFiles $\leftarrow \text{fileList}[\text{trainIdx}], \text{fileList}[\text{valIdx}]$
 trainLabel, valLabel $\leftarrow \text{labels}[\text{trainIdx}], \text{labels}[\text{valIdx}]$

5. *// Generate data for current fold*
trainGen \leftarrow create_generator(trainFiles, trainLabel, 'train')
valGen \leftarrow create_generator(valFiles, valLabel, 'validation')
6. *// Compile the model*
model.compile(optimizer=Adam(lr), loss='binary_crossentropy', metrics=['accuracy'])
7. *// Train the model on training dataset*
history \leftarrow model.fit(trainGen, validation_data, epochs)
8. *// Analyze the model using the validation dataset.*
scores \leftarrow model.evaluate(valGen)
accuracies \leftarrow append(scores[1] * 100)
losses \leftarrow append(scores[0])
9. Calculate and print aggregate accuracy and loss across all folds
10. *// Plot training and validation metrics (Accuracy and Loss)*
plot_metrics(history)

Algorithm 4. Model Evaluation on the Test Dataset

1. **Input:** Test Dataset = {img, lbl} where img are images and lbl are labels ($lbl \in \{AI, camera\}$)
 2. **Output:** Confusion matrix, classification report and labeled result for the test image dataset
 3. *// Generate data for testing*
testGen \leftarrow create_generator(testDir)
 4. *// Evaluation of model on the test dataset*
testLoss, testAcc = model.evaluate(testGen)
 5. *// Generate classification report*
y_pred = model.predict(testGen)
y_true = testGen.classes
conf_matrix = confusion_matrix(y_true, y_pred)
report = classification_report(y_true, y_pred, target_names=['AI-Generated', 'camera'],
output_dict=True)
-

5. Experimental Setup

This section outlines the experimental setup and presents the results achieved by implementing transfer learning with MobileNet, Inception-v3, and AlexNet for deepfake detection.

5.1 Implementation Details

The experiments were conducted on a system running Windows 11, equipped with a 13th Gen Intel(R) Core (TM) i7-13620H processor (2.40 GHz), 16 GB of RAM, and a 64-bit operating system. For developing deep learning models, Python Integrated Development and Learning Environment (IDLE) version 3.8.80 was selected due to its user-friendly interface and simplicity. Additionally, the Scientific Python Development Environment (Spyder) was employed for deep learning tasks. The hyperparameters used for the implementation of the proposed approach using MobileNet are detailed in Table 3.

Table 3. Hyperparameters for the implementation of the proposed approach using MobileNet

S.No.	Hyperparameters	Value
-------	-----------------	-------

1. Batch Size	32
2. Epochs	20
3. Number of Folds (k)	5
4. random_state	42
5. rotation_range	20
6. Width_shift_range	0.2
7. height_shift_range	0.2
8. zoom_range	0.2
9. rescale	1./255
10. input_shape	(224,224,3)
11. Total number of hidden Layers	29 (28 layers from base model and 1 additional layer)
12. Total additional layers	4
13. Activation for additional hidden layer	'relu'
14. Dropout	0.5
15. learning rate	0.0001
16. Optimizer	Adam
17. loss	'binary_crossentropy'
18. Activation for the output layer	'sigmoid'

5.2 Experimental Results

The confusion matrix is used to evaluate the performance of pre-trained models in detecting deepfakes, offering a detailed analysis of the model's ability to distinguish between different classes (e.g., camera images vs. AI-generated images). The confusion matrix comprises four key components: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). TP represent cases where the model correctly identifies an AI-generated image as a deepfake, while TN denote instances where the model accurately classifies a camera image as not being a deepfake. FP occur when the model mistakenly labels a camera image as a deepfake, while FN refer to situations where the model incorrectly classifies an AI-generated image as a real camera image.

Equation (i), (ii), (iii), and (iv) illustrate the calculations for Precision, Recall, Accuracy, and F1-score based on the confusion matrix, as shown below:

$$Precision = TP / (TP + FP) \quad (i)$$

Precision is defined as the ratio of true positive predictions to all positive predictions made by the model.

$$Recall = TP / (TP + FN) \quad (ii)$$

The ratio of genuine positive predictions to all real positives in the dataset is known as recall (or sensitivity).

$$Accuracy = (TP + TN) / (TP + TN + FP + FN) \quad (iii)$$

The ratio of accurate predictions to total predictions is known as accuracy.

$$F1-score = 2 * Precision * Recall / (Precision + Recall) \quad (iv)$$

The F1-score provides a single measure that combines precision and recall by calculating their harmonic mean. The results for the MobileNet model are presented in Table 4, the findings for the InceptionV3 model are shown in Table 5, and the results for the AlexNet model are provided in Table 6.

Table 4. Confusion matrix for the MobileNet Model

	Predicted Positive	Predicted Negative
Actual Positive	397	19
Actual Negative	31	553

Table 5. Confusion matrix for the Inception-v3 Model

	Predicted Positive	Predicted Negative
Actual Positive	250	32
Actual Negative	46	272

Table 6. Confusion matrix for the AlexNet Model

	Predicted Positive	Predicted Negative
Actual Positive	437	51
Actual Negative	59	453

The experimental results, including accuracy, precision, recall, and F1-score for each model, are presented in Table 7. The MobileNet model outperforms both Inception-v3 and AlexNet, achieving an accuracy of 95%, a precision of 92.66%, a recall of 95.22%, and an F1-score of 94.90%. These results indicate that MobileNet is the most effective model for detecting AI images, while InceptionV3 and AlexNet demonstrate moderate performance with accuracies of 87.0% and 89.0%, respectively.

Table 7. Experimental results for classification of AI-generated and Camera Trap Images

Model	Accuracy	Precision	Recall	F1-score
-------	----------	-----------	--------	----------

MobileNet	95.00%	92.66%	95.52%	94.90%
Inception-v3	87.00%	84.46%	88.65%	86.65%
AlexNet	89.00%	88.05%	89.60%	89.07%

5.3 Comparison with the state-of-the-art

Figure 6 compares the proposed approach with the state-of-the-art for the automated classification of AI-generated and camera images, demonstrating that our approach achieves the highest accuracy (95%) among all.

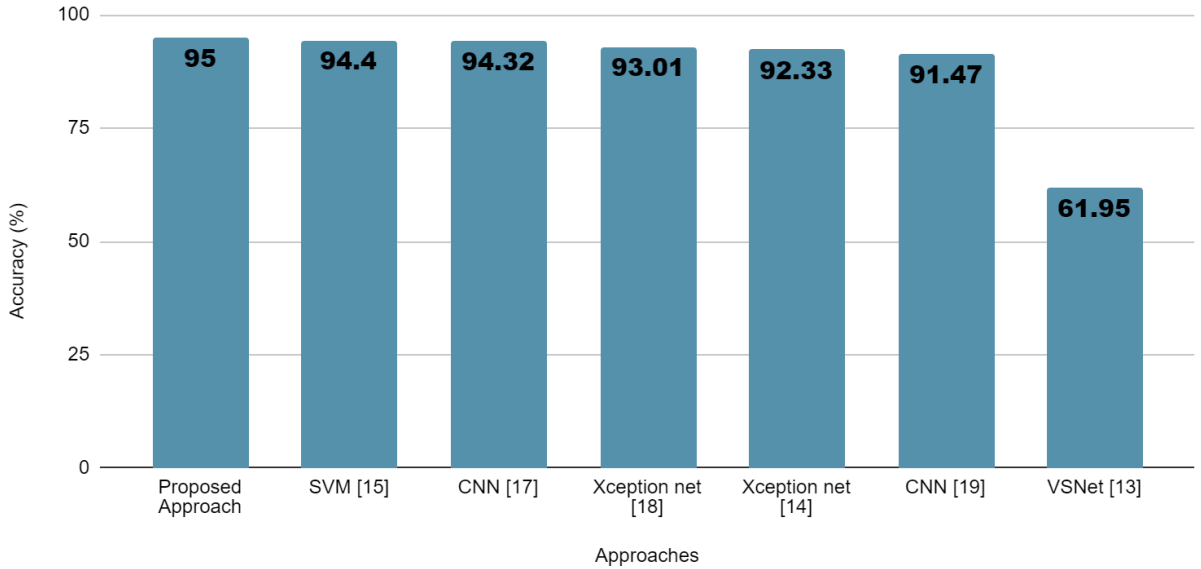


Figure 6. Accuracy comparison of the proposed framework and existing methods for classifying AI-generated and camera images

6. Conclusion

Motivated by the increasing digital manipulation of images, particularly of females, which leads to financial and reputational damage, this work introduces a novel approach to the automated classification of AI-generated and camera images. The proposed approach utilizes pre-trained model-based supervised learning techniques, including transfer learning and data augmentation, to achieve highly accurate results. The approach involves training and validating large datasets labeled as AI-generated or camera images, with cross-validation to ensure high accuracy. The proposed approach with the MobileNet model achieved the highest accuracy of 95% and an F1-score of 94.90%. The proposed approach demonstrates superior accuracy compared to existing techniques in the literature for automated real and deepfake image classification. In future work, this approach can be extended by applying it to datasets specifically created for male and child images.

REFERENCES

1. Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." *Advances in neural information processing systems* 27 (2014). doi: <https://doi.org/10.48550/arXiv.1406.2661>, Available Online: <https://arxiv.org/abs/1406.2661>, Accessed on: June 10, 2014.
2. Korshunov, Pavel, and Sébastien Marcel. "Deepfakes: a new threat to face recognition? assessment and detection." *arXiv preprint arXiv:1812.08685* (2018). Available Online: <https://arxiv.org/abs/1812.08685> Accessed on: December 20, 2018.
3. Samantha Murphy Kelly, Explicit, AI-generated Taylor Swift images spread quickly on social media, CNN, Available Online: <https://edition.cnn.com/2024/01/25/tech/taylor-swift-ai-generated-images/index.html>. Accessed on: January 25, 2024.
4. Heather Chen et al., Finance worker pays out \$25 million after video call with deepfake ‘chief financial officer’, Available Online: <https://edition.cnn.com/2024/01/25/tech/taylor-swift-ai-generated-images/index.html>. Accessed on: February 4, 2024.
5. Turkey Earthquake Charity Scams Alert, Available Online: <https://news.trendmicro.com/2023/02/20/turkey-earthquake-charity-scams/>. Accessed on: February 20, 2023.
6. Avril Ronan, Trend Micro, Fooled: Top Twelve AI Scams and Pranks, Available Online: <https://news.trendmicro.com/2024/04/01/april-fool-ai-scams-pranks/>. Accessed on: Mar 31, 2024.
7. Jeon, Hyeonseong, Youngoh Bang, and Simon S. Woo. "Fdftnet: Facing off fake images using fake detection fine-tuning network." In *IFIP international conference on ICT systems security and privacy protection*, pp. 416-430. Cham: Springer International Publishing, 2020. doi: https://doi.org/10.1007/978-3-030-58201-2_28, Available Online: https://link.springer.com/chapter/10.1007/978-3-030-58201-2_28, Accessed on: September 14, 2020.
8. Malik, Asad, Minoru Kuribayashi, Sani M. Abdullahi, and Ahmad Neyaz Khan. "DeepFake detection for human face images and videos: A survey." *Ieee Access* 10 (2022): 18757-18775. doi: 10.1109/ACCESS.2022.3151186. Available Online: <https://ieeexplore.ieee.org/abstract/document/9712265>, Accessed on: February 11, 2022.
9. Mirsky, Yisroel, and Wenke Lee. "The creation and detection of deepfakes: A survey." *ACM computing surveys (CSUR)* 54, no. 1 (2021): 1-41. doi: <https://doi.org/10.1145/342578>.
10. Dang, L. Minh, Syed Ibrahim Hassan, Suhyeon Im, and Hyeonjoon Moon. "Face image manipulation detection based on a convolutional neural network." *Expert Systems with Applications* 129 (2019): 156-168. doi: <https://doi.org/10.1016/j.eswa.2019.04.005>.
11. Moura, Augusto FS, Silas SL Pereira, Mário WL Moreira, and Joel JPC Rodrigues. "Video monitoring system using facial recognition: A facenet-based approach." In *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pp. 1-6. IEEE, 2020. doi: 10.1109/GLOBECOM42002.2020.9348216.
12. Hsu, Chih-Chung, Yi-Xiu Zhuang, and Chia-Yen Lee. "Deep fake image detection based on pairwise learning." *Applied Sciences* 10, no. 1 (2020): 370. doi: <https://doi.org/10.3390/app10010370>.
13. Lewis, John K., Imad Eddine Toubal, Helen Chen, Vishal Sandesera, Michael Lomnitz, Zigfried Hampel-Arias, Callyam Prasad, and Kannappan Palaniappan. "Deepfake video detection based on spatial,

- spectral, and temporal inconsistencies using multimodal deep learning." In 2020 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), pp. 1-9. IEEE, 2020. doi: 10.1109/AIPR50011.2020.9425167.
14. Mitra, Alakananda, Saraju P. Mohanty, Peter Corcoran, and Elias Kougianos. "A machine learning based approach for deepfake detection in social media through key video frame extraction." *SN Computer Science* 2, no. 2 (2021): 98. doi: <https://doi.org/10.1007/s42979-021-00495-x>.
 15. Xu, Bozhi, Jiarui Liu, Jifan Liang, Wei Lu, and Yue Zhang. "DeepFake Videos Detection Based on Texture Features." *Computers, Materials & Continua* 68, no. 1 (2021). doi: 10.32604/cmc.2021.016760
 16. Suratkar, S., Kazi, F. Deep Fake Video Detection Using Transfer Learning Approach. *Arab J Sci Eng* **48**, 9727–9737 (2023). <https://doi.org/10.1007/s13369-022-07321-3>
 17. Haseena, S., S. Saroja, and A. Nivetha. "TVN: Detect Deepfakes Images using Texture Variation Network." *Inteligencia artificial* 26, no. 72 (2023): 1-14. doi: <https://doi.org/10.4114/intartif.vol26iss72pp1-14>
 18. P. Joshi and N. V, "Deep Fake Image Detection using Xception Architecture," 2024 5th International Conference on Recent Trends in Computer Science and Technology (ICRTCST), Jamshedpur, India, 2024, pp. 533-537, doi: 10.1109/ICRTCST61793.2024.10578398. keywords: {Training;Deepfakes;Visualization;Accuracy;Transfer learning;Data preprocessing;Computer architecture;CNN;Deep Fake;Fine-tuned;Xception}, Accessed on: July 04, 2024.
 19. D. Gura, B. Dong, D. Mehیار, and N.A. Said "Customized Convolutional Neural Network for Accurate Detection of Deep Fake Images in Video Collections," *Comput. Mater. Contin.*, vol. 79, no. 2, pp. 1995-2014. 2024. <https://doi.org/10.32604/cmc.2024.048238> .
 20. Perchance.org, AI Image Generator, Available Online: <https://perchance.org/ai-text-to-image-generator>. Accessed on: June 2024.
 21. Microsoft AI image generator, Create any image you can dream up with Microsoft's AI image generator. Available Online: <https://designer.microsoft.com/image-creator> Accessed on: June 2024
 22. Pinterest. (n.d.). Pinterest. Available Online: <https://in.pinterest.com/>. Accessed on: June 2024
 23. Freepik, Create great designs, faster. (n.d.-b). Available Online: <https://freepik.com/>. Accessed on: June 2024
 24. Borji, Ali. "Qualitative failures of image generation models and their application in detecting deepfakes." *Image and Vision Computing* 137 (2023): 104771. doi: <https://doi.org/10.1016/j.imavis.2023.104771>