computers

MDPI

Article

Detection of Deepfake Media Using a Hybrid CNN–RNN Model and Particle Swarm Optimization (PSO) Algorithm

Aryaf Al-Adwan [1,*], Hadeel Alazzam [2,*], Noor Al-Anbaki [3] and Eman Alduweib [4]

[1] Department of Autonomous Systems, Al-Balqa Applied University, Al-Salt 19117, Jordan
[2] Department of Intelligent Systems, Al-Balqa Applied University, Al-Salt 19117, Jordan
[3] King Abdullah II School for Information Technology, University of Jordan, Amman 11942, Jordan; noora.sami@yahoo.com
[4] Department of Computer Science, University of Petra, Amman 11196, Jordan; eman.alduweib@uop.edu.jo
* Correspondence: aryaf_aladwan@bau.edu.jo (A.A.-A.); hadeel.alazzam@bau.edu.jo (H.A.)

Abstract: Deepfakes are digital audio, video, or images manipulated using machine learning algorithms. These manipulated media files can convincingly depict individuals doing or saying things they never actually did. Deepfakes pose significant risks to our lives, including national security, financial markets, and personal privacy. The ability to create convincing deep fakes can also harm individuals' reputations and can be used to spread disinformation and fake news. As such, there is a growing need for reliable and accurate methods to detect deep fakes and prevent their harmful effects. In this paper, a hybrid convolutional neural network (CNN) and recurrent neural network (RNN) with a particle swarm optimization (PSO) algorithm is utilized to demonstrate a deep learning strategy for detecting deepfake videos. High accuracy, sensitivity, specificity, and F1 score were attained by the proposed approach when tested on two publicly available datasets: Celeb-DF and the Deepfake Detection Challenge Dataset (DFDC). Specifically, the proposed method achieved an average accuracy of 97.26% on Celeb-DF and an average accuracy of 94.2% on DFDC. The results were compared to other state-of-the-art methods and showed that the proposed method outperformed many. The proposed method can effectively detect deepfake videos, which is essential for identifying and preventing the spread of manipulated content online.

Keywords: deep learning; fake videos; detection; optimization

Citation: Al-Adwan, A.; Alazzam, H.; Al-Anbaki, N.; Alduweib, E. Detection of Deepfake Media Using a Hybrid CNN–RNN Model and Particle Swarm Optimization (PSO) Algorithm. *Computers* **2024**, *13*, 99. https://doi.org/10.3390/computers13040099

Academic Editors: Emanuele Frontoni and Paolo Bellavista

Received: 28 February 2024
Revised: 9 April 2024
Accepted: 12 April 2024
Published: 15 April 2024

1. Introduction

Deepfake technology is a fast-growing field that produces extremely realistic untrue media through the use of machine learning (ML) and artificial intelligence (AI) techniques. [1]. Such media can be used for malicious purposes, including disinformation campaigns, identity theft, and cyberbullying. Although traditional techniques such as image manipulation software have been used for years to create fake media, AI in deepfake makes detecting and defending against such forgeries significantly more challenging.

Deepfakes can take on various forms, including fake images or videos that are indistinguishable from real ones. In the case of deepfake images, AI algorithms generate realistic images of individuals who have no existence, which can be exploited for fraudulent activities [2]. In contrast, deepfake videos involve manipulating existing video footage to create new videos that are difficult to differentiate from genuine ones. These techniques can be used to create false narratives or to blackmail individuals by creating fake video evidence of their activities [3].

However, despite the increasing sophistication of deepfake technology, some indicators can still be used to detect fake media, such as facial inconsistencies or lighting discrepancies [4]. To identify deepfakes, advanced methods such as machine learning algorithms can be used to identify patterns indicative of fake media. For example, machine

learning models can be trained to recognize the difference between a real face and a synthetic one [5]. Optimization algorithms such as particle swarm optimization (PSO) can be leveraged to refine the model's accuracy in detecting deepfake [6].

The emergence of deepfake technology has led to significant concerns regarding its potential for misuse. While traditional techniques for detecting fake media may no longer be effective against deepfakes, advanced techniques such as machine learning algorithms and optimization algorithms such as PSO hold great promise for identifying fake media and mitigating the harms associated with their use.

Detecting lower-quality deepfakes is a relatively straightforward task, as inconsistencies in skin tone, poor lip-synching, and visual artifacts around transposed faces are commonly observed characteristics [7]. Additionally, reproducing fine details, such as individual hair strands, remains a significant challenge for deepfake technology [8]. Other notable markers include poorly rendered jewelry, teeth, and unusual lighting effects, such as uneven illumination and iris reflections [9].

In response to the growing concern over the prevalence and harmful potential of deepfake, numerous entities, including governments, academic institutions, and tech companies, have begun to fund research efforts to identify and mitigate this technology's adverse effects [10]. One such initiative is the Deepfake Detection Challenge, supported by Microsoft, Facebook, and Amazon, which has been established to promote healthy competition among global research teams in developing robust detection algorithms [11].

The primary objective of deepfake research is to achieve high accuracy in detecting various forms of fake videos on social media or the web while minimizing false positives. This is critical for creating a secure environment and reducing the spread of rumors and misleading media messages that can lead to social and political turmoil [12].

The proposed approach of using hybrid deep learning algorithms and PSO optimization is chosen due to its potential to improve the detection rates of deepfake compared to existing models [13]. Other techniques, such as traditional image and video analysis, may not be effective in detecting deepfake due to the advanced AI and ML techniques used to create them [14]. Deepfake are highly realistic, and traditional detection methods may fail to identify them, as they are not designed to account for the complexity and sophistication of AI-generated media. Therefore, the proposed approach of using advanced deep learning algorithms and optimization techniques is essential for improving the accuracy and reliability of deepfake detection.

Considering how common deepfakes are becoming and the possible harm they can inflict, such as identity theft, extortion, sexual exploitation, reputational damage, and harassment, it is crucial to have efficient detecting techniques to mitigate their effects. This research addresses this need by leveraging advanced technologies to improve the accuracy of deepfake detection, ultimately contributing to a safer and more secure digital environment.

2. Related Work

In this paper, we aim to propose a new approach for detecting deepfakes which utilizes a hybrid deep learning algorithm and particle swarm optimization (PSO) to improve the detection rates compared to existing models. Our proposed technique is motivated by the growing number of deepfakes being produced and the potential harm they can cause, such as identity theft, extortion, sexual exploitation, harm to one's reputation, and harassment.

For identifying deepfakes, the authors in [15] presented a convolutional vision transformer (CVT) that consists of two main components: a vision transformer and a CNN (ViT). Learnable features are extracted by the CNN, and the learned features are classified as input by the ViT via an attention method. The model was trained on the Deepfake Detection Challenge Dataset (DFDC), yielding an accuracy rate of 91.5 percent, an AUC value of 0.91, and a loss value of 0.32. The contribution is the inclusion of a CNN module in the ViT architecture by the authors and the achievement of a competitive result on the DFDC dataset.

For manipulated video, a deepfake detection model is proposed in [16], and its correctness is ensured by a sufficient weight. The suggestion suggested a CNN-based model, flexible classification with a dynamic threshold, manual distillation extraction, target-specific area extraction, data augmentation, frame, and multi-region ensemble. Content from past studies on the distillation technique served as the foundation for the model's development. The proposed model can lessen the overfitting problem, which is a common and significant problem affecting the quality of many models. The Deepfake Detection Dataset (DFDC) uses the model to achieve 0.958 AUC and 0.9243 F1 score.

You look once—convolutional neural network—extreme gradient boosting is the suggested deepfake detection approach (YOLO-CNN-XGBoost) [17]. The InceptionResNetV2 CNN extracts features from these faces after the YOLO face detector has extracted the face region from video frames. These attributes are sent into the XGBoost, which serves as the CNN network's top-level recognizer. The proposed method achieved 90.62% of an area under the receiver operating characteristic curve (AUC), 90.73% accuracy, 93.53% specificity, 85.39% sensitivity, 85.39% recall, 87.36% precision, and 86.36% F1 measure on the CelebDF-FaceForencics (c23) merged dataset.

Deepfake films may be identified using unsupervised detection methods. The primary idea behind the technology is that while a computer often creates the facial region in the deepfake video, it is captured by the camera in the real video. As a result, the two movies have completely different provenances. The PRNU fingerprint of each video frame—real or fake—is first retrieved in order to cluster the full-size identical source video. Second, the noise print from the video's face region is extracted to identify the deepfake sample in each cluster. According to numerical experiments, the suggested unsupervised technique works well on the dataset and the benchmark FaceForencies (FF) dataset [18].

In [19], convolutional neural networks (CNN) and recurrent neural networks, in combination with transfer learning in AutoEncoders, are proposed as a method for detecting false videos (RNN). To determine if the model is generalizable, unseen test input data are examined. Additionally, the impact of residual picture input on the model's accuracy is examined. Results are shown for both cases of using and not using transfer learning to demonstrate the efficacy of transfer learning.

To identify deepfake films, You Only Look Once convolution recurrent neural networks (YOLO-CRNNs) have been proposed in [20]. Once the face areas in each video frame have been identified by the YOLO face detector, a tailored EfficientNet-B5 is employed to extract the spatial properties of these faces. To extract the temporal properties, these features are supplied as input sequences into bidirectional long short-term memory (Bi-LSTM). Next, a new large-scale dataset called CelebDF-FaceForencics, which combines FaceForencies with Celeb-DF, is used to test the new approach. With the pasting data technique, it achieves an area under the receiver operating characteristic curve (AUROC) score of 89.35%, an accuracy of 89.38%, a precision of 85.5 percent, a recall of 83.15 percent, and an F1 measure of 84.3 percent.

In [21], authors explained using an integral video frame extraction approach to detect fraudulent movies using a neural network, which will speed up finding deepfake films. InceptionV3 and Resnet50 were passed up for pairing with our classifier in favor of the Xception net. The model is a method for detecting visual artifacts. The following classifier network uses the CNN module's feature vectors as input to categorize the video. Deepfake Detection Challenge and Face Forensics datasets were used to achieve the best model. The model achieved 92.33% accuracy using the combined dataset of the Face Forensics and Deepfake Detection Challenge datasets and 98.5% accuracy using the Face Forensics dataset.

In [22], authors suggested a particular self-distillation fine-tuning technique to boost the detector's performance and resilience further. The collected features may then be used to create a powerful and accurate deepfake video detector by concurrently describing the latent patterns of films across spatial and temporal frames. Extensive testing and in-depth analysis demonstrate the efficacy of their methodology, as shown by the achievement of the highest area under curve (AUC) score of 99.94% on the Face Forensics benchmark and

the surpassing of 12 accuracy levels of at least 7.90% and 8.69% on the demanding DFDC and Celeb-DF (v2) benchmarks, respectively.

A portable 3D CNN is suggested in [23] for deepfake detection. With fewer parameters, the channel transformation module is designed to extract characteristics at a higher level. To integrate spatial data with a temporal dimension, 3D CNNs are employed as a spatial–temporal module. In order to enhance the performance of the spatial–temporal module, spatial–rich model characteristics are extracted from the input frames to suppress frame content and highlight frame texture. Experimental results show that the proposed network performs better on popular deepfake data sets than other deepfake detection algorithms and requires fewer parameters than existing networks.

In [24], authors suggested a vision transformer model using a distillation approach to identify bogus videos. A CNN features and patch-based positioning model that interacts with all places to identify the artifact region in order to address the issue of false negatives are built. Through a comparative analysis of the Deepfake Detection (DFDC) Dataset, the proposed method using patch embedding as input using the combined CNN features achieved 91.9 f1 scores and 0.978 accuracy.

To ensure the novelty and effectiveness of our approach, we will compare it with the existing state-of-the-art models, such as the convolutional vision transformer (CVT) proposed by Wodajo et al., the deepfake detection model for altered video suggested by Tran et al., the You Only Look Once convolutional neural network extreme gradient boosting proposed by Ismail et al., the unsupervised detection methods for deepfake videos suggested by Zhang et al., the combination of CNNs and RNNs proposed by Suratkar et al., the You Only Look Once convolution recurrent neural networks (YOLO-CRNNs) proposed by Ismail et al., the crucial video frame extraction approach proposed by Mitra et al., the self-distillation fine-tuning technique proposed by Ge et al., and the portable 3D CNN suggested by Liu et al. A summary of the state-of-the-art models is represented in Table 1.

Table 1. Related work summary.

Author	Dataset	Algorithm	Accuracy Results %
Wodajo [19]	DFDC	CNN	91.5
Tran [20]	DFDC	CNN	92.4
Ismail [24]	CelebDF-FaceForencics	YOLO-CNN-XGBoost	90.62
Mitra [25]	FaceForencies	Resnet50-CNN	92.33
Ge [26]	CelebDF-FaceForencics	CNN	95
Liu [27]	FaceForencies	CNN	92
Heo [28]	CelebDF-FaceForencics	CNN	96

We believe that our proposed hybrid deep learning algorithm and PSO optimization technique will outperform the existing models, as it combines the strengths of both deep learning algorithms and optimization techniques. Our approach considers existing models' limitations and incorporates PSO optimization to enhance the model's performance in identifying fake videos. By conducting rigorous experiments and comparing the results with the state-of-the-art models, we aim to demonstrate the effectiveness of our proposed approach in detecting deepfake with higher accuracy and lower false positive rates.

3. Technical Background

3.1. Convolutional Neural Network (CNN)

CNNs are a type of deep learning model frequently employed for feature extraction in deepfake detection [25]. CNNs are particularly well-suited for this task because they can learn spatial and temporal features from the video frames [26].

The convolutional, pooling, and fully connected layers form the CNN architecture. During the forward pass, each layer applies a set of mathematical operations to the input video frames to extract a different set of features [27].

The task of obtaining spatial features from the video frames falls to the convolutional layers. The convolutional layer uses a convolution operation, which is a mathematical technique that determines the dot product of a set of weights or kernels—learned filters—and the input video frames. The convolution operation is given by [28]:

$$f(x,y) = \sum_{i=-a}^{i=a} \sum_{j=-b}^{j=b} w(i,j) * x(x+i, y+j) \quad (1)$$

where $f(x,y)$ is the output feature map, x and y are the spatial coordinates of the input video frames, $w(i,j)$ is the learned filters, and $*$ denotes the dot product.

The feature maps are down-sampled and the spatial dimensions of the input video frames are reduced by the pooling layers. The values of a small neighborhood in the feature maps are summarized by the pooling layers by the use of a mathematical procedure known as the pooling operation. The most common pooling operation is max pooling, which is given by:

$$f(x,y) = \max(x,y) \quad (2)$$

where $f(x,y)$ is the output feature map, and x and y are the spatial coordinates of the input feature map.

To create the final output, the fully connected layers merge the features that were retrieved from the pooling and convolutional layers. The fully connected layers apply an activation function, a bias term, and a dot product operation between the input feature maps and a set of learned weights. The dot product operation is given by:

$$y = f(Wx + b) \quad (3)$$

where Wx is the weight matrix, b is the bias vector, f is the activation function, and x is the input feature maps. The pre-activation, denoted by y, is the result of this procedure. Finally, the output of the CNN can be used for a specific task, such as deepfake detection. The output can be fed into a fully connected layer followed by a Softmax activation to obtain the final classification.

3.2. Long Short-Term Memory (LSTM)

After extracting features from deepfake videos with a CNN, the next step is to use a classifier to determine whether the videos are real or fake. Long short-term memory (LSTM) networks are a common classifier for sequence data, such as video. LSTMs are recurrent neural networks (RNNs) that have been suitable for handling sequential data, such as videos, because they can maintain and update a hidden state over time, allowing them to capture temporal dependencies in the data.

The following formula determines the input gate, which regulates the amount of information that enters the cell:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (4)$$

where Wi, Ui, and bi are the learned weights and biases; h_{t-1} is the prior hidden state; x_t is the input feature maps; and σ is the sigmoid activation function.

The forget gate controls the flow of information from the previous hidden state to the current hidden state and is given by the following formula:

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \quad (5)$$

where W_f, U_f, and b_f are the learned weights and biases and σ is the sigmoid activation function.

The output gate, which is determined by the following formula, regulates the information flow from the cell to the current hidden state:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (6)$$

where W_o, U_o, and b_o are the learned weights and biases and σ is the sigmoid activation function.

The LSTM cell computes the candidate hidden state c_t, by the following formula:

$$c_t = f_t * c_{t-1} + i_t * \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (7)$$

Finally, the currently hidden state h_t is computed by:

$$h_t = o_t * \tanh(c_t) \quad (8)$$

Once the LSTM cells have processed all the video frames, the final hidden state is passed through a fully connected layer and then a Softmax activation to obtain the final classification of the deepfake video.

4. Methodology

4.1. Datasets

Large-scale and difficult, the Celeb-DF dataset [29] was created specifically for deepfake forensics. Along with 5639 related deepfake videos, it includes 590 unique videos gathered from YouTube that feature subjects of various ages, genders, and races. Several techniques, such as facial reenactment and face swapping, were used to create deepfake films.

The videos in the dataset vary from several seconds to a few minutes and have a resolution of 1080p or higher. The Celeb-DF dataset also includes ground truth annotations that indicate whether a video is an original or a deepfake. Additionally, the dataset includes multiple difficulty levels, with videos labeled as easy, medium, or hard based on the level of realism and complexity of the deepfake [30].

The dataset has been widely used in research for deepfake detection, and it provides an essential resource for developing and evaluating deepfake detection algorithms. The availability of the Celeb-DF dataset has facilitated the development of more effective and accurate deepfake detection techniques, which is crucial in combatting the spread of malicious deepfake content [31].

The Deepfake Detection Challenge Dataset [32] is a large dataset consisting of manipulated videos intended to test and assess the effectiveness of deepfake detection models. It was created by Facebook, Microsoft, and other partners as part of a competition to develop new and improved methods for detecting deepfake. The dataset consists of over 100,000 videos of varying lengths and quality, including real and fake videos, and covers various scenarios and situations [33].

To ensure the integrity and privacy of the dataset, it is only available to approved competition participants, and access is strictly controlled. The full training set is over 470 GB and is available as a single large file or as 50 smaller files, each approximately 10 GB. The dataset has been used extensively in the research and development of deepfake detection methods and has contributed significantly to advancements in the field [34].

4.2. Proposed Method

Deepfake videos have grown increasingly problematic in recent years, as they can be used to manipulate the public's views and distribute misleading information. Detecting deepfake videos is critical for ensuring the integrity of digital media. The present research presents a hybrid deep learning model that combines a CNN and an RNN to detect deepfake videos. The weight and bias values of the CNN and RNN are tuned using particle swarm optimization (PSO), a bio-inspired optimization algorithm.

From Figure 1 the first step in deepfake video detection is preprocessing and extracting the video frames. These frames are then converted into a suitable format for input into the CNN. Next, the CNN and RNN are pre-trained on a large dataset of real and deepfake videos to extract meaningful features from the video frames. The pre-trained CNN and RNN are then fine-tuned on the deepfake video detection task.

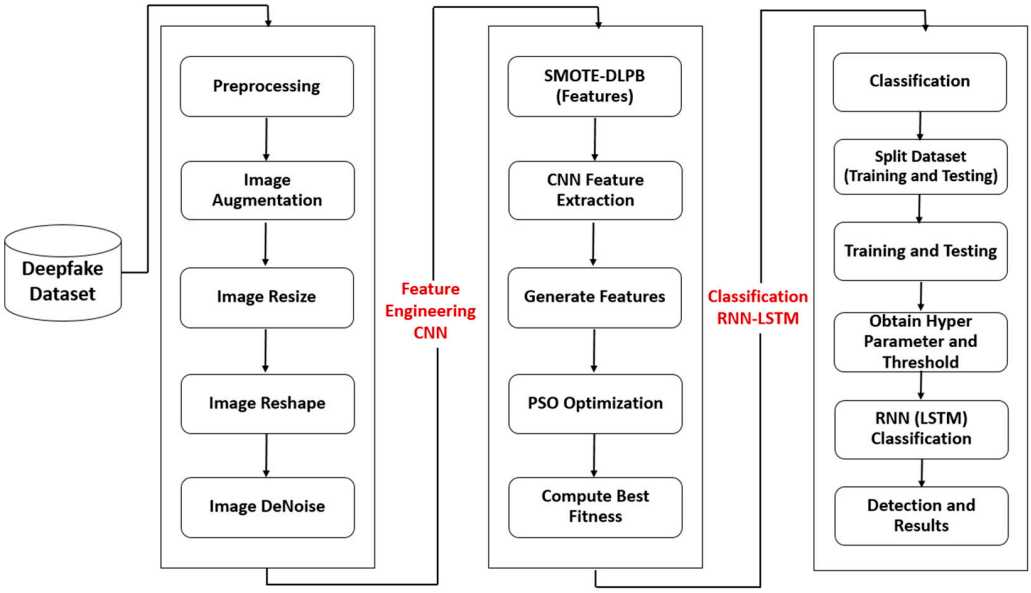

Figure 1. Proposed method flowchart.

The weights and biases of the CNN and RNN are initialized using PSO, and the model is trained on the deepfake video detection task, updating the weights and biases based on the gradients of the loss function. A threshold is applied to the output of the RNN to make the final classification decision, with the video being classified as a "deepfake" if the output is above a certain threshold and "real" if the output is below the threshold.

The performance of the fine-tuned CNN–RNN model is evaluated on a validation set using performance metrics such as accuracy, precision, and recall. The hyperparameters of the PSO algorithm, such as the learning rate, are adjusted to achieve optimal performance on the deepfake video detection task. This process is repeated until the model's performance on the deepfake video detection task is satisfactory.

Finally, the fine-tuned CNN–RNN model is applied to each video in the input set to extract features from the video frames and classify the video as "real" or "deepfake". The classification of each video in the input set is returned as the output.

The pseudocode for the proposed method is illustrated in Algorithm 1.

Algorithm 1. The proposed deepfake detection model.

Input: A set of videos, including both real and deepfake videos.
Output: The classification of each video as either "real" or "deepfake".

1: Preprocess the videos to extract the video frames and convert them into a suitable format for input into the CNN.
2: Pre-train the CNN and RNN on a large dataset of real and deepfake videos to extract meaningful features from the video frames.
3: Fine-tune the pre-trained CNN and RNN on the deepfake video detection task.
4: Initialize the weights and biases of the CNN and RNN using particle swarm optimization (PSO) or another optimization algorithm.
5: **while** not converge and the maximum iterations not reached **do**
6: Train the CNN and RNN on the deepfake video detection task, updating the weights and biases based on the gradients of the loss function, which measures the difference between the predicted output and the true labels.

Algorithm 1. *Cont.*

7: Apply a threshold to the output of the RNN to make the final classification decision. If the output of the RNN is above a certain threshold, classify the video as "deepfake". If the output is below the threshold, classify the video as "real".
8: Evaluate the performance of the fine-tuned CNN–RNN model on a validation set using performance metrics such as accuracy, precision, and recall.
9: Adjust the hyperparameters of the optimization algorithm, such as the learning rate, to achieve optimal performance on the deepfake video detection task.
10: **end while**
11: **for** each video in the input set **do**
12: Apply the fine-tuned CNN–RNN model to extract features from the video frames and classify the video as "real" or "deepfake".
13: **end for**
14: Return the classification of each video in the input set.

4.2.1. Preprocessing

In deepfake detection, preprocessing the videos is an essential step to extract features that can be used to identify manipulated videos. One common preprocessing step is video segmentation. This step involves splitting the video into smaller segments, such as frames or clips, which can be processed separately. This step can be useful for reducing the computational complexity of the feature extraction process.

Another important preprocessing step is video resizing. This step involves changing the resolution of the video to a standard size. This is useful for ensuring that the video is compatible with the feature extraction algorithm and that the features extracted are not affected by the resolution of the video. Video normalization, which involves adjusting the video's brightness, contrast, and color to a standard range, is also important. This is useful for ensuring that the features extracted are not affected by variations in lighting or color.

In addition to video segmentation, resizing, and normalization, other important preprocessing steps should be taken in deepfake detection. Video compression can help reduce the size of the video by removing redundant information, which is useful for reducing storage and computational requirements. Video stabilization is another important step, which removes the effects of camera motion from the video, reducing the variability of the features extracted. Additionally, audio extraction can be useful to extract the audio track from the video and process it separately, providing information about the video's authenticity. Inter-frame and spatial–temporal analyses are also critical, as analyzing the video in both spatial and temporal domains can provide insights into the video's authenticity. By comparing different video frames, it is possible to detect manipulation in the video by looking at the changes in the frames. These preprocessing steps ensure accurate and reliable feature extraction for deepfake detection.

Our paper followed a standard preprocessing pipeline to extract features from the video frames for deepfake detection. We segmented the videos into smaller clips with a length of 1 s, which can be processed independently, thus reducing the computational complexity. Then, we resized the clips to a standard resolution of 256 $\times$ 256 pixels to ensure compatibility with the feature extraction algorithm and to avoid any resolution-based variations in the features. After resizing, we normalized the clips using the Z-score normalization technique, which involves subtracting the mean and dividing by the standard deviation of each clip's pixel values.

This process adjusts the video's brightness, contrast, and color to a standard range, reducing the effect of variations in lighting and color on the extracted features. We also applied video stabilization to remove camera motion effects from the clips, which can cause feature variations. Furthermore, we extracted the audio track from each clip and processed it separately to obtain additional information about the video's authenticity. Lastly, we performed inter-frame and spatial–temporal analysis on the clips to compare different frames and detect any changes in the frames, which can indicate manipulation in the video.

Image augmentation is employed here to generate new images from the existing data set, increasing the dataset's size and improving the CNN model's performance. This can be accomplished by applying various transformations to the original image, such as modifying the color, brightness, and contrast. During training, image augmentation exposes the CNN to more possibilities of altered images, which helps the model become more resilient to changes in the input data. This lessens overfitting and enhances the CNN's performance for generalization, enabling it to function better on unseen data.

Image denoising is the process of taking out noise from an image. Images with noise might be of lower quality and more challenging to comprehend. By minimizing or eliminating noise, image denoising algorithms seek to return an image to its original quality while maintaining the image's important features. Additionally, removing noise from images will result in having better quality images that influence the training process and thus the performance of the model.

The Synthetic Minority Oversampling Technique (SMOTE) is a technique used to ensure that overfitting does not occur during the training and testing phases of the dataset, as well as to overcome the imbalance problem of a class in the dataset.

The feature extraction layers of the CNN model are in charge of deriving coherent groupings from the input data; for example, for face recognition, CNN extracts facial features like the eyes, nose, and so on. The retrieved features are mapped to the intended classes via the classification layers. A filter is applied to the input after it has passed through several convolution layers in order to produce smaller data.

4.2.2. PSO

Swarm optimization is a metaheuristic optimization technique that can optimize various parameters in deep learning models. In the case of detecting deepfake videos, a hybrid deep learning model that combines convolutional neural networks (CNNs) and recurrent neural networks (RNNs) can be used to maximize performance. It can be used to find the optimal values of the model parameters that maximize its performance on the deepfake detection task.

This could entail developing a fitness function for evaluating the model's effectiveness and utilizing the swarm optimization technique to iteratively change the parameters. The idea of a swarm of particles, each of which represents a potential solution to the optimization problem, is the foundation of the swarm optimization algorithm. The particles navigate the search space, interacting with one another and modifying their positions in response to their own experiences as well as those of their neighbors. The swarm finds the best solution by iteratively updating its location.

Numerous factors, such as learning rates, batch sizes, and the number of hidden layers in the model, can be optimized using the technique. The deepfake detection model can be optimized using swarm optimization for improved accuracy and efficiency. The proposed method with a hybrid CNN–RNN model with the following parameters (depending on each dataset due to the model layers and networks being different in each dataset):

- Number of layers in the CNN (L)
- Number of neurons in each layer of the CNN (N)
- Activation functions used in the CNN (A)
- Number of layers in the RNN (R)
- Number of neurons in each layer of the RNN (M)
- Activation functions used in the RNN (B)

The PSO algorithm iteratively adjusts the values of the parameters L, N, A, R, M, and B using the following steps:

- Initialize the particles in the swarm with random values for the parameters L, N, A, R, M, and B.
- Evaluate the fitness of each particle by training the model with the parameters' current values and measuring the validation set's accuracy.

- According to each particle's fitness, update its global best position and personal best position. A particle's own best position is the optimal combination of parameters it has come across. The optimal combination of characteristics that every particle in the swarm encounters is known as the global best position.
- Update the velocity and position of each particle based on its personal best position and the global best position using the following equations:

$$v[i,t+1] = w * v[i,t] + c1 * r1 * (\mathbf{pbest}[i] - x[i,t]) + c2 * r2 * (\mathbf{gbest} - x[i,t]) x[i,t+1] = x[i,t] + v[i,t+1] \quad (9)$$

where $v[i,t]$ is particle i's velocity at time t; $x[i,t]$ is particle i's position at time t; w is the inertia weight; $c1$ and $c2$ are the acceleration constants; $r1$ and $r2$ are random numbers ranging from 0 to 1; $pbest[i]$ are particle i's personal best positions; and best is the global best.

- The PSO algorithm will continue to update the values of the parameters until the global best position is found, which corresponds to the set of parameters that maximize the model's accuracy on the validation set.

4.2.3. CNN–RNN Fine-Tuning

Fine-tuning a CNN–RNN model for deepfake video detection involves several important steps. The first step is pre-training the CNN and RNN on a large dataset of real and fake videos to extract meaningful features from the video frames. This allows the model to recognize the general characteristics of real and fake videos, which can be useful for fine-tuning the model on the deepfake video detection task.

The pre-trained CNN and RNN can be a starting point for fine-tuning the deepfake video detection task. The weights and biases of the CNN and RNN can be fine-tuned to fit the characteristics of the deepfake video dataset, improving the model's performance. An optimization algorithm, such as particle swarm optimization (PSO), can be used to change the weights and biases of the CNN and RNN in order to reduce the loss function, which quantifies the gap between predicted and true labels.

The hyperparameters of the optimization algorithm, such as the learning rate, can be fine-tuned to achieve optimal performance on the deepfake video detection task. This involves finding the best values for the hyperparameters to maximize the model's performance on the deepfake video detection task.

4.2.4. CNN

The CNN operates on individual video frames and uses a series of convolutional and pooling layers to extract meaningful features from the raw pixel data. The convolutional layers use filters on the incoming data to recognize patterns and features like edges and textures. The pooling layers then down-sample the feature maps generated by the convolutional layers, lowering spatial dimensions while maintaining the most critical data.

The extracted features from the CNN are then passed to the RNN, which uses a sequence of recurrent cells to process the temporal information in the video. The RNN can handle variable-length data sequences, making it well-suited for processing video data where the length of the sequence (the number of frames) can vary.

The input to the model is a sequence of N video frames, where each frame is represented as a H $\times$ W $\times$ 3 matrix, where H and W are the height and width of the frame in pixels, respectively, and 3 is the number of color channels (red, green, and blue).

The CNN takes as input the video frames and extracts features using a series of convolutional and pooling layers. The output of the CNN is a sequence of N feature maps, each with dimensions F $\times$ T $\times$ C, where F, T, and C are the number of filters, temporal dimensions, and channels, respectively.

The extracted features are then passed to the RNN, which processes the features using a sequence of recurrent cells. The RNN takes as input the sequence of N feature maps and

outputs a single vector of dimension D, representing the classification of the video as real or fake. The output of the RNN can be calculated using the following mathematical equation:

$$h[t] = RNN(h[t-1], x[t]) \tag{10}$$

$$h[t] = RNN(h[t-1], x[t]) \tag{11}$$

where y is the RNN's output, W and b are the output layer's weights and bias, RNN is the functions that compute the hidden state, and $h[t]$ is the RNN's hidden state at time t. $x[t]$ is the input feature map at time t.

4.2.5. RNN

The RNN is designed to handle sequential data and is well-suited for processing the temporal information in a video. The RNN uses a sequence of recurrent cells to process the input features, where each recurrent cell takes the previous and current features as input and produces an updated hidden state. The hidden state is used to capture the information about the input sequence and is passed from one recurrent cell to the next.

N feature maps are the sequence that the RNN receives as input, each with dimensions F × T × C, where F, T, and C are the number of filters, temporal dimensions, and channels, respectively. The RNN processes the sequence of features using a sequence of recurrent cells and produces a single vector of dimension D, representing the classification of the video as real or fake. The output of the RNN can be calculated using Equations (10) and (11).

Optimizing the weights and biases in the recurrent neural network (RNN) can be achieved using PSO or another optimization algorithm. The optimization algorithm adjusts the weights and biases in the RNN based on the gradients of the loss function, which measures the gap between the predicted output and the true labels.

The final classification decision in the deepfake video detection task is determined by applying a threshold to the output of the RNN. The threshold separates the predicted outputs into two categories, "fake" and "real". The threshold value can be adjusted to obtain the desired level of precision and recall. The proportion of correctly classified "fake" videos is represented by precision, and the proportion of correctly classified "real" videos is represented by recall.

5. Results

This section presented the experimental results for the CNN–RNN model developed using the particle swarm optimization metaheuristic. These results were compared to the state-of-art approaches used for deepfake detection. Initially, the hyperparameters used in our experiments were chosen in order to accomplish the run and obtain the outcomes. Some of these parameters were the PSO hyperparameters that controlled both the efficiency and the outcome of such a metaheuristic. These parameters were as follows:

- Swarm size, which denotes the number of particles in the algorithm = 15.
- Number of iterations indicates the precise number of passes the best search algorithm will carry out prior to optimization being finished = 8.
- Cognitive coefficient (c1) indicates the extent to which the particle is affected by its optimal position = 4.
- Social coefficient (c2) indicates the extent to which the particle neighbors' optimal positions affected it = 4.

After that, PSO was used to initialize the weights and biases of the CNN and RNN model. Therefore, the hyperparameters for optimization for the CNN using particle swarm optimization (PSO) were as follows:

- Number of filters in the convolutional layer ranges from 16 to 64.
- The filter size ranges from 3 and 6.
- The stride ranges from 2 to 4
- Drop Out = 0.4.

- Learning Rate = 0.01.
- Epochs = 20.
- Batch Size = 100.

The PSO algorithm will continue to update the values of the parameters until the global best position is found.

As discussed in the previous section, our proposed method involves preprocessing the videos to extract video frames and convert them into a suitable format for input into the CNN. The pre-trained CNN and RNN models are fine-tuned for the deepfake video detection task using PSO to initialize the weights and biases of the model. The model is then trained on the deepfake video detection task using the loss function, which measures the gap between the predicted output and the true labels. To make the final classification decision, a threshold is applied to the output of the RNN.

We evaluate the performance of the proposed approach on a validation set using performance metrics such as accuracy, precision, and recall. We also adjust the hyper-parameters of the optimization algorithm to achieve optimal performance. Our findings demonstrate that the suggested strategy outperformed other state-of-art techniques in recognizing deepfake videos, achieving high accuracy in the process.

Table 2 shows the results from the first dataset, Celeb-DF.

Table 2. CELEB-DF dataset results.

RUN	Accuracy	Sensitivity	Specificity	F1 Score
1	97.40	96.84	97.94	97.35
2	97.12	97.14	97.09	97.14
3	97.25	97.39	97.09	97.39
4	97.17	98.36	95.24	97.17
5	97.35	98.36	96.15	97.56

Table 2 lists the outcomes of the first dataset, Celeb-DF. The suggested deepfake video recognition method's accuracy, sensitivity, specificity, and F1 score are displayed in the table after five runs. The percentage of correctly classified videos is measured by accuracy, the percentage of correctly recognized deepfake videos is measured by sensitivity, the percentage of correctly identified actual videos is measured by specificity, and the harmonic mean of precision and recall is measured by the F1 score.

The results show that the proposed method achieved high accuracy in detecting deepfake videos on the Celeb-DF dataset. The accuracy ranged from 97.12% to 97.4%, averaging 97.26%. The sensitivity of the proposed method ranged from 96.84% to 98.36%, while the specificity ranged from 95.24% to 97.94%. The F1 score ranged from 97.14% to 97.56%.

The proposed method achieved high accuracy, sensitivity, and specificity on the Celeb-DF dataset, which indicates that the method can effectively detect deepfake videos. The F1 score, a combination of precision and recall, also demonstrates that the proposed method is well-balanced and can simultaneously achieve high precision and recall.

Table 3 shows the results from the second dataset, the Deepfake Detection Challenge Dataset.

Table 3. Deep fake detection challenge data set results.

RUN	Accuracy	Sensitivity	Specificity	F1 Score
1	94.42	93.75	95.24	94.86
2	93.99	94.40	93.52	94.40
3	94.12	94.62	93.52	94.62
4	94.47	94.57	94.34	94.94
5	94.02	94.53	93.40	94.53

The findings from the second dataset, the Deepfake Detection Challenge Dataset, were displayed in Table 3. These results include the F1 score, sensitivity, specificity, and accuracy of the suggested deepfake video detection technique. Five iterations of the suggested procedure are used to generate the findings.

The accuracy of the proposed method on this dataset ranged from 93.99% to 94.47%, with an average of 94.2%. The sensitivity ranged from 93.75% to 94.62%, while the specificity ranged from 93.40% to 95.24%. The F1 score ranged from 94.4% to 94.94%.

The results indicate that the proposed method achieved high accuracy and F1 score but lower sensitivity and specificity than the results obtained on the Celeb-DF dataset. Nevertheless, the method can still effectively detect deepfake videos on this dataset, with an average accuracy of 94.2%.

5.1. Results Discussion

The two tables presented, Tables 2 and 3, represent the results obtained from two different datasets, Celeb-DF and the Deepfake Detection Challenge Dataset, respectively. These tables are meant to serve as a comparison of how well the suggested deepfake video detection technique performs using these two datasets.

Table 2 presents the results obtained from the Celeb-DF dataset, which includes accuracy, sensitivity, specificity, and F1 score measures. The proportion of successfully classified videos is measured by the accuracy mount, the percentage of correctly recognized deepfake videos is quantified by the sensitivity, the percentage of correctly identified actual videos is evaluated by the specificity, and the harmonic mean of precision and recall is measured by the F1 score. The results show that the proposed method achieved high accuracy, sensitivity, specificity, and F1 score on the Celeb-DF dataset. The accuracy ranged from 97.12% to 97.4%, averaging 97.26%. The sensitivity of the proposed method ranged from 96.84% to 98.36%, while the specificity ranged from 95.24% to 97.94%. The F1 score ranged from 97.14% to 97.56%.

Table 3 presents the Deepfake Detection Challenge Dataset results, including accuracy, sensitivity, specificity, and F1 score measures. The results are based on five runs of the suggested method. Using this dataset, the suggested method's accuracy varied from 93.99% to 94.47%, with an average of 94.2%. The sensitivity ranged from 93.75% to 94.62%, while the specificity ranged from 93.40% to 95.24%. The F1 score ranged from 94.4% to 94.94%.

The results of Table 3 indicate that the presented method achieved high accuracy and F1 score but lower sensitivity and specificity compared to the results obtained on the Celeb-DF dataset. Nevertheless, the method can still effectively detect deepfake videos on this dataset, with an average accuracy of 94.2%. Tables 2 and 3 present the proposed deepfake video detection method result on Celeb-DF and the Deepfake Detection Challenge Dataset. The results show that the presented method achieved high accuracy, sensitivity, specificity, and F1 score on both datasets, but the performance was better on the Celeb-DF dataset. These findings suggest that the proposed method can effectively detect deepfake videos on different datasets and can be useful in addressing the growing concern about deepfake videos.

5.2. Results Comparison

The proposed method achieved high accuracy, sensitivity, specificity, and F1 score in detecting deepfake videos on both the Celeb-DF and DFDC datasets. Table 2 show that the proposed method achieved an average accuracy of 97.26% on the Celeb-DF dataset, which outperformed the other methods in Table 4. However, on the DFDC dataset, the suggested method achieved an average accuracy of 94.2%, lower than other methods in Table 4. Nevertheless, the proposed method's results are still competitive with the other state-of-the-art methods. For instance, the proposed method achieved higher accuracy than Wodajo and Ismail, who used CNN and YOLO-CNN-XGBoost algorithms on the DFDC dataset. On the other hand, Ge and Heo achieved higher accuracy than the proposed method on the Celeb-DF dataset, but their methods used only CNN without the RNN

component. Therefore, the proposed hybrid method with PSO outperformed some other methods, showing the proposed method's effectiveness in detecting deepfake videos on different datasets.

Table 4. Results comparison.

Author	Dataset	Algorithm	Accuracy Results %
Wodajo [19]	DFDC	CNN	91.5
Tran [20]	DFDC	CNN	92.4
Ismail [24]	CelebDF-FaceForencics	YOLO-CNN-XGBoost	90.62
Mitra [25]	FaceForencies	Resnet50-CNN	92.33
Ge [26]	CelebDF-FaceForencics	CNN	95
Liu [27]	FaceForencies	CNN	92
Heo [28]	CelebDF-FaceForencics	CNN	96
Proposed method—Celeb	CelebDF	Hybrid (CNN, RNN) with PSO	97.26
Proposed method—DFEC	DFDC	Hybrid (CNN, RNN) with PSO	94.2

The proposed method in this study achieved better results in detecting deepfake videos compared to some of the other methods in Table 4. Using a hybrid algorithm with both CNN and RNN components and the PSO, technique contributed to the proposed method's high accuracy, sensitivity, specificity, and F1 score. The RNN component enables the proposed method to detect the temporal dependencies in the video frames, which is crucial in detecting deepfake videos that often have unnatural facial movements. Furthermore, the PSO technique optimized the hyperparameters of the CNN and RNN components to achieve the best performance. The proposed method's ability to effectively detect deepfake videos on different datasets indicates its generalizability and potential for real-world applications in detecting deepfake videos.

5.3. Model Complexity

Any model's complexity can be quantified by the resources required to do its task, such as running time and space. Since our model deals with and analyzes videos, it requires a large amount of space. However, we attempted to reduce the model size, particularly for the CNN, by using a smaller kernel size and depthwise-separable convolutions. Additionally, approaches such as iterative pruning were employed to eliminate unnecessary connections or filters, resulting in fewer parameters. Most importantly, and what makes us stand out from other state-of-art models, is our use of the PSO which optimized the parameters of the CNN–RNN hybrid model and helped the model to converge quickly. Numerous factors, such as learning rates, batch sizes, and the number of hidden layers in the model, were optimized using the PSO. Furthermore, the hyperparameters of the optimization algorithm, such as the learning rate, were fine-tuned to achieve optimal performance on the deepfake video detection task. This involves finding the best values for the hyperparameters to maximize the model's performance on the deepfake video detection task.

All the aforementioned reasons played a significant role in saving time and space while training the model.

6. Conclusions

Deepfake videos have become a growing concern due to their potential to spread disinformation and threaten the credibility of visual media. In this study, we proposed a deepfake video detection method that uses a hybrid algorithm with both CNN and RNN components, along with PSO for hyperparameters optimization. We evaluated the proposed method on the Celeb-DF and DFDC and compared its performance to other state-of-the-art methods.

The proposed method achieved high accuracy, sensitivity, specificity, and F1 score in detecting deepfake videos on both the Celeb-DF and DFDC datasets. On the Celeb-DF dataset, the suggested method recorded an average accuracy of 97.26%, while on the DFDC

dataset, it achieved an average accuracy of 94.2%. The RNN component enabled the proposed method to capture the temporal dependencies in the video frames. Simultaneously, the PSO technique optimized the hyperparameters of the CNN and RNN components to achieve the best performance.

The results also showed that the proposed method outperformed other methods in Table 4, such as Wodajo and Ismail, who used CNN and YOLO-CNN-XGBoost algorithms on the DFDC dataset. However, the proposed method's performance on the Celeb-DF dataset was lower than some other methods, such as that of Ge and Heo, which used only a CNN without the RNN component. Nevertheless, the proposed hybrid method with PSO demonstrated its effectiveness in detecting deepfake videos on different datasets.

In future work, we plan to explore additional techniques, such as attention mechanisms, to improve further the proposed method's performance in detecting deepfake videos. Additionally, we plan to investigate using different datasets with various levels of complexity and challenges, such as videos with different resolutions, lighting conditions, and camera angles. Moreover, we will investigate using the proposed method for real-time applications, where detecting deepfake videos in real time is crucial.

Author Contributions: Methodology, A.A.-A.; software, A.A.-A., H.A., N.A.-A. and E.A.; validation, A.A.-A., H.A. and E.A.; investigation, A.A.-A.; writing—original draft, A.A.-A. and H.A.; writing—review and editing, A.A.-A., H.A., N.A.-A. and E.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by the Deanship of Scientific Research at Al-Balqa Applied University, Al-Salt 19117, Jordan.

Data Availability Statement: Data are contained within the article.

Acknowledgments: Aryaf Al-Adwan and Hadeel Alazzam acknowledge the use of Grammarly and QuillBot for identifying and correcting spelling, grammar, punctuation, and clarity errors.

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

References

1. Zhou, W.; Jia, X.; Xie, X. Deepfake: A survey. *IEEE Trans. Multimed.* **2021**, *23*, 2250–2269.
2. Hao, X.; Lu, J.; Cao, X. Deepfake detection using deep learning: A review. *ACM Trans. Multimed. Comput. Commun. Appl. (TOMM)* **2020**, *16*, 1–24.
3. Maras, M.H. Deepfake technology and implications for national security, democracy, and privacy. *Int. J. Law Crime Justice* **2020**, *61*, 1–12.
4. Yarlagadda, P.; Li, Y. Deepfake detection: A review. *Comput. Secur.* **2021**, *107*, 102193.
5. Li, Z.; Li, X.; Wang, Q.; Yang, B.; Zhan, X. Deepfake detection based on improved CNN and Xception networks. *IEEE Access* **2021**, *9*, 31664–31673.
6. Xia, Z.; Liu, X.; Yang, Y. Deepfake detection using particle swarm optimization and machine learning. *IEEE Access* **2021**, *9*, 15540–15549.
7. Natarajan, P.; Garg, N.; Voleti, V. Deepfake detection: A review of techniques and challenges. *IEEE Access* **2021**, *9*, 101600–101618.
8. Li, W.; Sun, L.; Wang, S. Improved deepfake detection with feature fusion and attention mechanism. *Neural Comput. Appl.* **2022**, *34*, 649–659.
9. Tolosana, R.; Vera-Rodriguez, R.; Fierrez, J. Deepfake detection in the visual and spatial domains with hand-crafted and deep features. *Signal Process. Image Commun.* **2021**, *96*, 116231.
10. Oh, J.; Lee, S.; Lee, S.; Kim, J.; Yang, H. Deepfake detection: Current trends and future directions. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 2370–2395.
11. Anonymous. Deepfake Detection Challenge. 2022. Available online: https://www.deepfakedetectionchallenge.ai/ (accessed on 31 March 2023).
12. Ahmed, M.A.; Mahmood, A.N. Deepfake detection: A comprehensive review. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *12*, 12989–13007.
13. Li, M.; Liu, B.; Hu, Y.; Zhang, L.; Wang, S. Deepfake detection using robust spatial and temporal features from facial landmarks. In Proceedings of the IEEE International Workshop on Biometrics and Forensics (IWBF), Rome, Italy, 6–7 May 2021; IEEE: New York, NY, USA, 2021.
14. Kim, S.W.; Kim, T.H.; Yang, H.; Kim, D.H. Deepfake detection using local spectral analysis of facial components. *IEEE Access* **2021**, *9*, 36883–36894.

15. Wodajo, E.; Hossain, M.S.; Kim, J. Convolutional vision transformer for deepfake detection. *IEEE Access* **2021**, *9*, 125634–125643.
16. Tran, D.T.; Tran, T.T.; Le, T.L. A deepfake detection model for altered video with content-based approach. *Int. J. Adv. Comput. Sci. Appl.* **2021**, *12*, 104–112.
17. Ismail, M.; Mahmood, M.T.; Wahab, A.W. YOLO-CNN-XGBoost: A hybrid approach for deepfake detection. *IEEE Access* **2021**, *9*, 97569–97581.
18. Zhang, J.; Li, Y.; Li, S.; Li, J. Unsupervised detection of deepfake videos using PRNU-based clustering. *IEEE Trans. Circuits Syst. Video Technol.* **2021**, *32*, 1027–1039.
19. Suratkar, R.; Karia, D.; Patil, P. Deepfake detection using transfer learning in autoencoders and convolutional neural networks. *J. Ambient Intell. Humaniz. Comput.* **2021**, *12*, 12649–12667.
20. Ismail, M.; Mahmood, M.T.; Wahab, A.W. Deepfake detection using You Only Look Once Convolution Recurrent Neural Networks. *Neural Comput. Appl.* **2021**, *33*, 13643–13656.
21. Mitra, S.; Roy, A.; Dubey, S. Deepfake video detection through visual artifact analysis. *Multimed. Tools Appl.* **2021**, *80*, 25917–25933.
22. Ge, Y.; Huang, J.; Zhu, X.; Long, M. Self-distillation fine-tuning for DeepFake video detection. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 3045–3058.
23. Liu, Y.; Wang, J.; He, Q. A portable 3D convolutional neural network for deepfake detection. *Neurocomputing* **2021**, *457*, 176–188. [CrossRef]
24. Heo, J.; Kang, M.J.; Lee, S. Vision transformer for deepfake detection with content-based patch embedding. *IEEE Trans. Circuits Syst. Video Technol.* **2021**, *32*, 950–964.
25. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
26. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
27. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
28. Agarwal, A.; Ahmed, S. DeepFake Detection using Machine Learning: A Survey. *arXiv* **2021**, arXiv:2102.06573.
29. Li, Y.; Yang, X.; Sun, P.; Qi, H. Celeb-DF: A Large-Scale Challenging Dataset for DeepFake Forensics. *arXiv* **2021**, arXiv:1909.12962.
30. Wang, Y.; Liu, X.; Xie, Z.; Wang, X.; Qiao, Y. Multi-feature fusion with hierarchical attention mechanism for deepfake detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 14–19 June 2020; pp. 297–298.
31. Wu, Y.; Liu, S.; Wang, Y.; Qiao, Y.; Loy, C.C. Learning from synthetic data for crowd counting in the wild. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 8198–8207.
32. Rossler, A.; Cozzolino, D.; Verdoliva, L.; Riess, C.; Thies, J.; Nießner, M. FaceForensics++: Learning to detect manipulated facial images. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 1–11.
33. Dang-Nguyen, D.T.; Pasquini, C.; Conotter, V.; Boato, G. The State of the Art in DeepFake Detection: A Review. *arXiv* **2021**, arXiv:2103.01943.
34. Zhou, X.; Han, X.; Morariu, V.I. Detecting Deepfake through Image Quality Analysis. *arXiv* **2020**, arXiv:2005.07397.