Three-Dimensional Convolutional Approaches for the Verification of Deepfake Videos: The Effect of Image Depth Size on Authentication Performance

Muhammad Salihin Saealal [1], Mohd Zamri Ibrahim [2,]*, Marlina Yakno [2], and Nurul Wahidah Arshad [2]

[1] Faculty of Electrical and Electronic Engineering Technology, Universiti Teknikal Malaysia Melaka, Melaka, Malaysia; Email: salihin@utem.edu.my (M.S.M.)
[2] Faculty of Electric and Electronics Engineering Technology, Universiti Malaysia Pahang, Pekan, Pahang, Malaysia; Email: marlinayakno@ump.edu.my (M.Y.), wahidah@ump.edu.my (N.W.A.)
*Correspondence: zamri@ump.edu.my (M.Z.I.)

***Abstract*—Deep learning has proven to be particularly effective in tasks such as data analysis, computer vision, and human control. However, as this method has become more advanced, it has also led to the creation of DeepFake video sequences and images in which alterations can be made without immediately appealing to the viewer. These technological advancements have introduced new security threats, including in the field of education. For example, in online exams and tests conducted through video conferencing, individuals may use Deepfake technology to impersonate another person, potentially allowing them to cheat by having someone else take the exam in their place. Several detection approaches have been proposed to address these issues, including systems that use both spatial and temporal features. However, existing approaches have limitations regarding detection accuracy and overall effectiveness. The paper proposes a technique for detecting Deepfakes that combines temporal analysis with convolutional neural networks. The study explores various 3-D Convolutional Neural Networks-based (CNN-based) model approaches and different sequence lengths of facial photos. The results indicate that using a 3-D CNN model with 16 sequential face images as input can detect Deepfakes with up to 97.3 percent accuracy on the FaceForensic dataset. Detecting Deepfakes is crucial as they pose a threat to the authenticity of visual media. The proposed technique offers a promising solution to this issue.**

***Keywords*—video forensic, deep learning, face forensic, 3-D convolution neural network, recurrent neural network, different sequence, online learning environment**

I. Introduction

The use of Deepfake technology in online exams and tests conducted through video conferencing can pose a significant security threat in the field of education. By using Deepfake technology, individuals can impersonate another person, potentially allowing them to cheat by having someone else take the exam in their place. This can undermine the integrity of the exam and compromise the validity of the results.

Deepfake technology is becoming increasingly sophisticated and widely accessible, making it easier for individuals to create and use fake videos for nefarious purposes. This technology can be used to manipulate audio and video recordings to make it appear as if someone else is speaking or performing a particular action. It is also possible to create a Deepfake video of an individual's face, where an impersonator can use the video to take an exam in the victim's place. This is a genuine concern and is becoming increasingly prevalent in the online learning environment.

It is clear that the use of Deepfake technology in online education poses significant challenges, and various measures and solutions must be implemented to mitigate this threat. Some solutions include using AI-based detection tools, facial recognition software, or proctoring software to detect and prevent cheating during online exams. In addition, institutions can also introduce measures such as multi-factor authentication, randomized test questions, and a zero-tolerance cheating policy.

Li *et al.* [1], Matern *et al.* [2], and Sabir *et al.* [3] proposed Deepfake detection methods to help address these issues. These techniques can be broadly classified into two categories [4]. First, detection based on visual artifacts within the video frame identifies abnormal features that can arise during Deepfake synthesis, such as teeth appearing as a single white blob instead of individual teeth. Second, rather than emphasizing temporal consistency, detection based on capturing temporal features across video frames is performed frame by frame [3].

There are numerous approaches to extracting this feature, the most commonly using a Convolution Neural Network (CNN) as the feature extractor [5]. CNN should

Manuscript received January 12, 2023; revised February 23, 2023; accepted March 20, 2023; published May 24, 2023.

doi: 10.12720/jait.14.3.488-494

become more familiar with delicate modification traces in order to perform image forensics operations. Guo *et al.* proposed learning manipulation traces on face photos using the Pre-processing module (AMTEN) [6]. They use of CNN and Fully Connected Network (FCN) for low-level feature extraction and categorization. Post-processing on the input photos, such as lossy compression, blurring, and scaling, can boost generalization ability. The method outperforms Bayar and Stamm's Constrained-Conv at residual extraction [7]. It can be used as a fundamental residual predictor for various face-forensic tasks.

3-D convolution kernels have been shown to learn spatial and temporal data simultaneously, with state-of-the-art results, thanks to recent advances in machine learning [8]. Nguyen et al. proposed a 3-D CNN-based technique for learning information in both spatial and temporal dimensions from straight faces obtained from a frame sequence, citing similar results but using a different approach [9]. To detect Deepfake videos, the authors used 3-D convolution kernels to build deep 3-D convolutional networks that extract spatiotemporal information from a short consecutive frame sequence.

Saealal et al. proposed a system for face detection that utilizes the eye-blinking state in temporal video frames to gather distinct feature information [10]. The system employs the VGG16 network to extract spatial features from the input images, using pre-trained weight on the ImageNet dataset. The authors then used the Long Short-term Memory (LSTM) network to extract temporal features from the input sequences, with a sampling rate of every 20 frames. Additionally, the authors incorporated the eye-blinking state as a prior probability to generate a new dataset for training a feed-forward network to classify the acquired data.

This study presented two distinct models utilized in the analysis: Both use 3-D convolution kernels to train a deep 3-D convolutional neural network to detect Deepfake videos by learning spatio-temporal properties in consecutive short frame sequences. The experiment was carried out on the most popular Deepfake video datasets [11, 12] and compared the efficiency of the suggested approaches for different depths of image sequences. The detailed analysis for each model is discussed in this manner. Section II describes the proposed method in full.

In conclusion, this paper contributes the following:

- Testing the efficacy of 3D convolution kernels for use in creating a deep 3-D CNN that can identify Deepfake videos by analyzing spatiotemporal properties extracted from relatively short sequences of consecutive frames;
- Evaluation of effective use of 16 image sequences compared to much lower image sequences for each model;
- A thorough examination of each proposed model's performance for three distinct depths of image sequence based on the accuracy and complexity of the models.

II. Methodology

This paper presents the sequence image features analysis method for efficient Deepfake detection. The low-level features of the cropped face image from each frame were extracted using a CNN-based model, together with its temporal features between 16, 8, and 4 sequence frames. The performance was then analyzed and tested on two different models. Below are the details of the approach used in this study.

A. Deepfake Model Detection

This study aimed to assess the efficacy of various machine learning models for detecting Deepfake videos. The study evaluated two models: a 3-D CNN baseline model (shown in Fig. 1) and a 3-D CNN extended model (shown in Fig. 2). The baseline model comprised 2 layers of 3-D CNN with a max pool before the fully connected network, which acted as the classifier. The extended model was based on the model proposed by Nguyen et al. and served as the core layer [9]. The FaceForensic++ dataset was used to train and test each model, and their performances were evaluated. Detailed descriptions of the experiment and a comprehensive analysis of the results are presented in the subsequent sections.

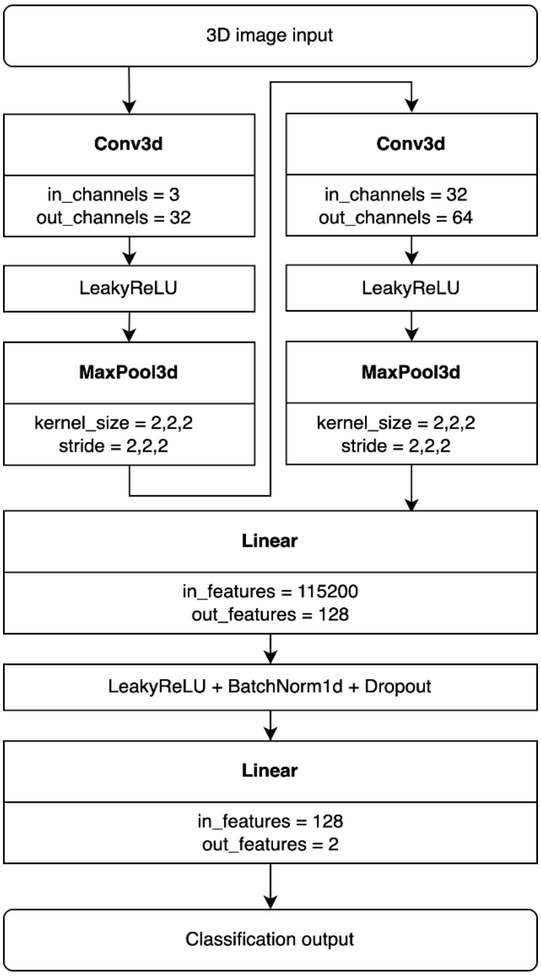


Figure 1. Model 1: 3-D CNN baseline model.

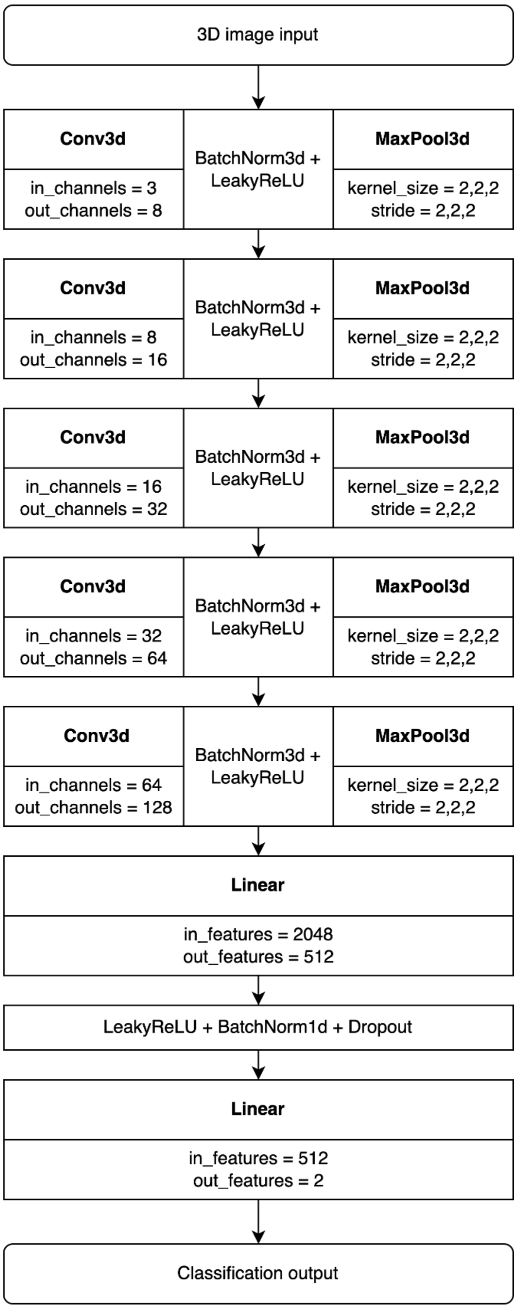


Figure 2. Model 2: 3-D CNN extended model.

B. *Video Pre-processing*

The videos from the FaceForensics++ dataset had to be extracted into individual frames. The face was then detected using the *dlib* library, and its landmark was determined. Based on this landmark, the face was cropped by forming a rectangle region around it. The cropped image was then resized to 128 by 128 pixels for consistency across the frames. The cropped images were then grouped and sorted into groups of 4, 8, and 16 and fed into the model as 3-D input images.

C. *Spatial-Temporal Model Training*

2-D and 3-D CNNs are artificial neural network types that process data with a grid-like topology, including images and videos. Each CNN type consists of layers that perform convolution operations on the input data and pooling and activation layers that introduce nonlinearity and downsampling. However, significant differences between 2-D and 3-D CNNs make them suited for different tasks.

2-D CNNs are the most commonly used type of CNN and are specifically designed to process 2-D data, such as images [7, 13]. They are composed of 2-D kernels that operate on 2-D inputs, sliding over the input data and performing a dot product with the kernel weights at each position (see Fig. 3). The output of this convolution operation is a 2-D feature map, which is then processed by subsequent layers in the network. 2-D CNNs are particularly effective at tasks such as image classification and object detection, where the spatial relationships between pixels in the input data are essential.

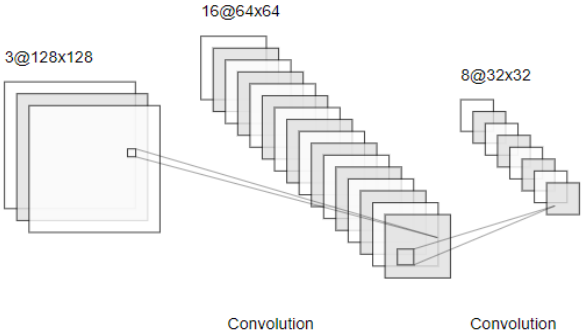


Figure 3. Example of 2-D CNN.

3-D CNNs, on the other hand, are designed to process 3-D data such as videos or 3-D medical images. They are composed of 3-D kernels that operate on 3-D inputs, performing a 3-D convolution operation. The output of this operation is a 3-D feature map, which captures both spatial and temporal information in the input data [14]. 3-D CNNs are particularly useful for tasks such as video classification and action recognition, where the temporal relationships between frames in the input data are essential (see Fig. 4).

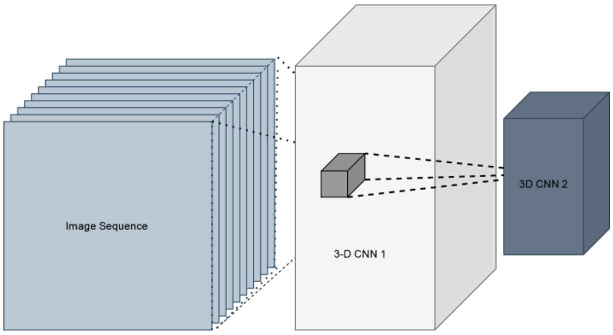


Figure 4. Example of 3-D CNN.

A CNN-based method extracted both spatial and temporal features in this study. The method employed a 3-D CNN model that extracted spatial features by applying two kernel dimensions on each frame. Furthermore, the model convolved the third dimension of the image sequence depth to extract temporal features.

3-D CNNs typically consist of multiple convolutional layers stacked on top of each other, followed by one or more fully connected layers. The convolutional layers apply a series of filters to the input data to extract features, while the fully connected layers use these features to make predictions or classifications.

The basic equation for a 3-D convolution operation is:

$$(f\times g)(x,y,z)=\sum_{a=-\infty}^{\infty}\sum_{b=-\infty}^{\infty}\sum_{c=-\infty}^{\infty}f(a,b,c)g(x-a,y-b,z-c) \qquad (1)$$

where f is the input volume, g is the convolutional kernel (also known as a filter), × represents the convolution operation, and (x,y,z) are the coordinates of the output volume [15].

This equation performs a summation over the elements of the input volume, *f*, and the kernel, *g*, using the formula for 3-D convolution. The result is a 3-D volume, $(f\times g)(x,y,z)$, which represents the output of the convolution operation.

The layer's output value relative to the size of the input $(N,C_{in},D,H,W)(N,C_{in},D,H,W)$ and output size $(N,C_{out},D_{out},H_{out},W_{out})(\ N,C_{out},D_{out},H_{out},W_{out})$ can be precisely defined as follows in the simplest possible scenario:

$$out(N_i,C_{outj})=\sum_{k=0}^{C_{in}-1}weight(C_{outj},k)\times input(N_i,k)+bias(C_{outj}) \qquad (2)$$

where $out(N_i,C_{outj})$ is the output at the position (N_i,C_{outj}) in the output volume, $weight(C_{outj},k)$ is the element at the position (C_{outj},k) in the convolutional kernel, $input(N_i,k)$ is the element at the position (N_i,k) in the input volume, $bias(C_{outj})$ is the element at position C_{outj} in the bias term, × represents the convolution operation, N_i is the batch size, k is the number of input channels, and C_{outj} is the number of output channels [16].

After each 3-D CNN layer, Batch Normalization (BN) is applied to improve the performance and stability of deep learning models. It was introduced by Ioffe and Szegedy in 2015 [17].

BN works by normalizing the activations of a layer for each minibatch. This stabilizes the learning process and allows higher learning rates to be used, leading to faster convergence. In addition, batch normalization has been shown to have a regularization effect that improves the generalization of the model to unseen data.

BN is usually implemented by inserting a BN layer after the linear transformation (e.g., convolution or fully connected) in a deep learning model. The BN layer computes the mean and standard deviation of the activations for each minibatch and uses these statistics to normalize the activations. The normalized activations are then scaled and shifted using learned parameters so that the model retains some of the information from the original activations.

BN has become a standard technique in Deep Learning and is used in many modern models. It is particularly useful in models with very deep architectures, where gradients can become unstable due to the vanishing gradient problem.

$$\mu_B=\frac{1}{m}\sum_{i=1}^{m}x_i \qquad (3)$$

$$\sigma_B^2=\frac{1}{m}\sum_{i=1}^{m}(x_i-\mu_B)^2 \qquad (4)$$

$$\hat{x}_i=\frac{x_i-\mu_B}{\sqrt{\sigma_B^2+\varepsilon}}\gamma+\beta \qquad (5)$$

where m is the number of examples in the mini-batch, x_i is the activation for the i-th example in the mini-batch, μ_B is the mean of the activations for the mini-batch, σ_B is the standard deviation of the activations for the mini-batch, ε is a small constant added to the variance to prevent division by zero, γ and β are learned parameters that scale and shift the normalized activations.

Eqs. (3) and (4) calculate the mean and variance of activations for a minibatch, which are used to normalize activations in the batch normalization Eq. (5) [17].

D. Classifier

At the classification stage, using a fully connected network as a classifier in a model can enable the model to distinguish between real and fake videos. The fully connected network takes the previous layer's output, which could be the final feature maps obtained from the CNN layers. It uses it to classify the input data into different categories. In the context of determining whether a video is real or fake, the fully connected network would receive the extracted features as input and use them to make a prediction about the authenticity of the video.

To make this prediction, the fully connected network uses weights and biases to combine the input features and produce a prediction. The weights and biases are learned during the training process. The model is presented with labeled examples of real and fake videos and adjusts the weights and biases to minimize the error between the predicted and actual labels. Once the model is trained, it can use the learned weights and biases to predict the authenticity of new, unseen videos (see Fig. 5).

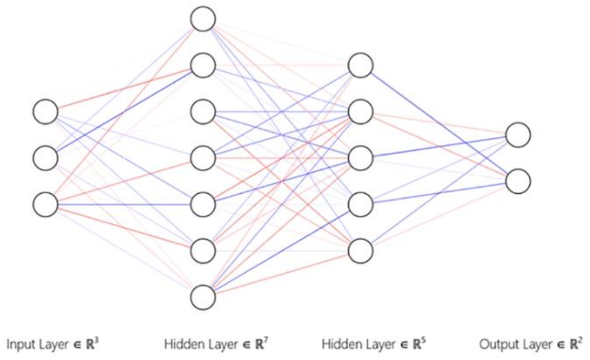

Input Layer ∈ $\mathbb{R}^3$
Hidden Layer ∈ $\mathbb{R}^7$
Hidden Layer ∈ $\mathbb{R}^5$
Output Layer ∈ $\mathbb{R}^2$

Figure 5. Example of a fully connected network.

The following operations are performed by the FCN:

$$n_0^{out} = I \tag{6}$$

$$n_0^{in} = n_0^{out} \times W_i + B_i \tag{7}$$

$$n_i^{out} = F_i(n_i^{in}) \tag{8}$$

where F_i is an activation function for the i layer. After performing a forward pass, the network's output can be computed at each layer using this method [18].

To update the neural network parameters during training, the backpropagation algorithm was employed to compute the gradient of the loss function with respect to the model weights and biases. Stochastic gradient descent was then used to update the parameters in the direction of the negative gradient, with the aim of minimizing the loss function. In addition, an adaptive learning rate algorithm called AdaGrad was utilized to adjust the learning rate during training. This algorithm enables the learning rate to be automatically scaled based on the historical gradient information, which can be useful for handling datasets with varying degrees of sparsity or uneven feature distributions [19]. The mathematical equation that describes the AdaGrad update process is shown below:

$$\theta_{t+1,i} = \theta_{t,1} - \frac{\eta}{\sqrt{\varepsilon + \sum_{\tau=1}^{t} (\nabla J(\theta_{\tau,i}))^2}} \nabla J(\theta_{\tau,i}) \tag{9}$$

where θ represents a parameter consisting of the weights, biases, and activations, η is the learning rate, ∇ denotes the gradient, and J is the objective function with its features and labels.

It is worth noting that the use of adaptive learning rate algorithms like AdaGrad can be beneficial for improving the convergence rate and robustness of neural network training. By adjusting the learning rate in a data-dependent manner, these algorithms can help prevent the model from getting stuck in local minima or overfitting to noisy or irrelevant features. However, it is important to carefully select the hyperparameters of the algorithm, such as the initial learning rate and decay rate, to ensure optimal performance on a given dataset.

In order to start calculating the error gradients, a loss function must be defined to calculate the errors. In this work, the cross entropy loss is used because it is a loss function commonly used in this field [20]. The cross entropy loss (L) is given by:

$$L = -\sum_{j}^{C} y_j \ln \hat{y}_j \tag{10}$$

where, the FCN utilized a loss function that was based on the number of classes (C), the true labels (y), and the predicted labels ($\hat{y}$), as defined by a previous study [20].

During the backpropagation steps, the gradient of the loss function was updated using a selected learning rate, and the weights and biases of the FCN were updated at the end of each epoch based on a batch of input data. The performance of the FCN was evaluated using accuracy and loss metrics that were calculated at the end of the training process.

III. Result and Discussion

In order to determine the effectiveness of the proposed approach in real-world applications, the FaceForensics++ dataset was used to provide examples of Deepfake videos. The deep learning framework used in this work was *Pytorch v1.12,* and the *scikit-learn* library was used for data analysis.

The experimental procedure involved training and testing multiple models using a dataset consisting of 35,648 image frames that were extracted from 450 videos. Performance of the models was evaluated using a range of evaluation metrics, including accuracy, precision, recall, and F1-score. The experimental design and resulting outcomes are described in detail in this section.

A. Deepfake Detection Using Model 1

Results from Table I show the performance of Model 1 using the group input of 4, 8, and 16 depth sequences of images. It can be seen that the model performed best when the number of images in the sequence was 8 in terms of accuracy, recall, and F1 score, but not precision.

TABLE I. Performance Metrics for a Range of Depth Sequence Value for Model 1

Depth Sequence	Performance Metrics			
	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)
4	93.24	90.63	97.48	93.93
8	95.50	93.60	98.32	95.90
16	94.14	95.69	92.38	94.47

Accuracy is a metric that measures a model's ability to correctly classify or predict the output for a given set of input examples. It is computed by dividing the number of correct predictions made by the model by the total number of predictions. A model with an accuracy of 95.50% made correct predictions 95.50% of the time and had an error rate of only 4.50%. This indicates that the model performed well on the task it was trained for.

B. Deepfake Detection Using Model 2

Results from Table II show the performance of Model 2 using the same group input of 4, 8, and 16 depth sequences of images. The model's performance was best when the number of images in the sequence was 16, as indicated by its high accuracy, recall, and F1 score. The precision value was slightly lower under this condition.

The precision of 97.48% means that only 2.52% of Deepfake that the system identified as positives (Deepfake) were false positives (not Deepfake). Recall, or the measure of how many Deepfake the model identifies as true positives compared to the number of actual Deepfake in the dataset is around 97.48% at 97.48% precision detection.

TABLE II. PERFORMANCE METRICS FOR A RANGE OF DEPTH SEQUENCE VALUE FOR MODEL 2

Depth Sequence	Performance Metrics			
	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
4	96.40	97.44	95.80	96.61
8	97.30	98.29	96.64	97.46
16	97.30	97.48	97.48	97.48

Suppose the recall and precision values are of the same number. In that case, it means that the model was able to correctly identify all relevant examples in the dataset (high recall) and that all of the examples it identified as relevant were actually relevant (high precision). This would indicate that the model performed well on the task it was trained for and could classify or predict the output for the input examples accurately.

In general, a model should have both high recall and high precision, as this indicates that the model can identify all relevant examples in the dataset while also minimizing the number of false positives.

C. *Overall Comparison*

Fig. 6 compares the performances of all models across various image sequence sizes. Model 2 outperformed other models in most metrics, except for recall at image depths of 4 and 8. Additionally, the overall performance of both models improved as the image sequence size increased. However, Model 1 performance began to decline when the depth size reached 16, potentially due to the two layers of 3-D CNN struggling to process more complex information compared to Model 2 five layers of 3-D CNN.

In general, increasing the depth of a CNN by adding more layers can improve its performance on a given task, as deeper networks can learn more complex and abstract features from the input data. The complexity of a network increases with its depth, resulting in more parameters that the model needs to learn. This can make the training process more computationally intensive and may lead to overfitting if the model is not appropriately regularized.

Therefore, it is possible that one of the models performed better than the other when the number of layers was increased, either because it could learn more useful features from the data or because it could generalize better to unseen examples.

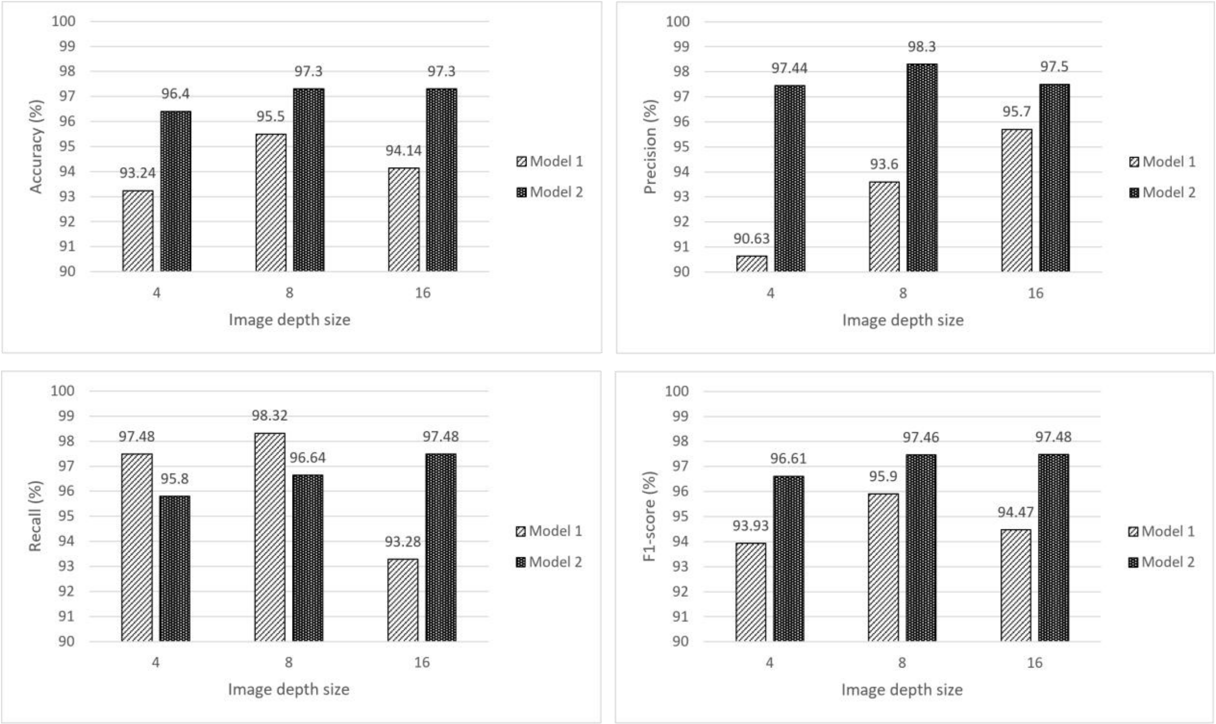
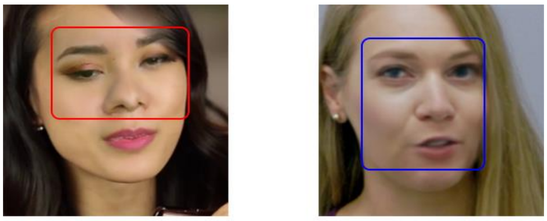


Figure 6. Comparison of the performance of Model 1 and Model 2 across various image sequence sizes.

Fig. 7 shows two examples of cropped faces extracted from videos. The proposed model successfully detected the first image, which had no visible artifacts created during the Deepfake generation process. However, the model was unable to detect the second image. The observations made in this study indicate that the model faced difficulty in detecting images with inadequate artifact creation and smooth transitions between frames. However, despite this limitation, the failure rate for detecting tampered videos remained relatively low.



(a) (b)

Figure 7. Comparison between tempered frames (a) Successfully detected (b) Unable to class as fake image.

IV. Conclusion and Future Recommendations

This study presents a method for learning features from clips in spatial and temporal dimensions by constructing image sequences using two CNN-based models. Testing with the FaceForensic++ dataset revealed that the proposed 3-D CNN model, which was adapted from the baseline 3-D CNN model with additional neural network parameters, outperformed it, achieving up to 97.30% accuracy.

The current work can be improved in several ways. One possible avenue for improvement is to examine the topologies of various deep neural networks to determine if there are more efficient methods for training face sequence pattern recognition. Additionally, the current method only utilizes face pixel information as input. However, it is crucial also to consider the continuity between frames, as unexpected changes and anomalies during transitions may indicate manipulation.

Future research will aim to enhance the results by exploring different deep neural network architectures and incorporating additional input factors, such as frame continuity and anomalies indicative of tampering. The study will also look into methods for leveraging more information, such as temporal continuity between frames using the LSTM network, to further improve the models' performance.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

All the authors conducted the research together. Particularly, M. S. Saealal, and M. Z. Ibrahim generated the problems, analyzed the data, and wrote the paper. M. Yakno, and N. W. Arshad supervised the method implementation.; all authors had approved the final version.

Acknowledgment

This research is financially supported by the Fundamental Research Grant Scheme (FRGS/1/2021/ICT07/UMP/02/1) with the RDU number RDU210136 which is awarded by the Ministry of Higher Education (MOHE) and Postgraduate Research Grants Scheme (PGRS) PGRS210338 via the Research and Innovation Department, Universiti Malaysia Pahang (UMP) Malaysia.

References

[1] Y. Li, M.-C. Chang, and S. Lyu, "In ictu oculi: Exposing AI created fake videos by detecting eye blinking," in *Proc. IEEE International Workshop on Information Forensics and Security (WIFS)*, 2019.

[2] F. Matern, C. Riess, and M. Stamminger, "Exploiting visual artifacts to expose deepfakes and face manipulations," in *Proc. IEEE Winter Applications and Computer Vision Workshops (WACVW)*, 2019, pp. 83–92.

[3] E. Sabir, J. Cheng, A. Jaiswal, W. AbdAlmageed, I. Masi, and P. Natarajan, "Recurrent convolutional strategies for face manipulation detection in videos," in *Proc. CVPR Workshops*, 2019.

[4] T. T. Nguyen, Q. V. H. Nguyen, D. T. Nguyen, D. T. Nguyen, T. Huynh-The, S. Nahavandi, T. T. Nguyen, Q.-V. Pham, and C. M. Nguyen, "Deep learning for deepfakes creation and detection: A survey," *Computer Vision and Image Understanding*, vol. 223, 2022.

[5] P. Kumar, M. Vatsa, and R. Singh, "Detecting face2face facial reenactment in videos," in *Proc. IEEE Workshop on Applications of Computer Vision (WACV)*, 2020, pp. 2578–2586.

[6] Z. Guo, G. Yang, J. Chen, and X. Sun, "Fake face detection via adaptive manipulation traces extraction network," *Computer Vision and Image Understanding*, vol. 204, p. 103170, 2021.

[7] B. Bayar and M. C. Stamm, "Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection," *IEEE Transactions on Information Forensics and Security*, vol. 13, pp. 2691–2706, 2018.

[8] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proc. 2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 4489–4497.

[9] X. H. Nguyen, T. S. Tran, V. T. Le, K. D. Nguyen, and D. T. Truong, "Learning spatio-temporal features to detect manipulated facial videos created by the deepfake techniques," *Forensic Science International: Digital Investigation*, vol. 36, 301108, 2021.

[10] M. S. Saealal, M. Z. Ibrahim, D. J. Mulvaney, M. I. Shapiai, and N. Fadilah, "Using cascade cnn-lstm-fcns to identify ai-altered video based on eye state sequence," *Plos One*, vol. 17, no. 12, p. e0278989, 2022.

[11] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Niessner, "Faceforensics++: Learning to detect manipulated facial images," in *Proc. 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1–11.

[12] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "Faceforensics: A large-scale video dataset for forgery detection in human faces," arXiv Preprint, arXiv:1803.09179, 2018.

[13] N. Bonettini, L. Bondi, E. D. Cannas, P. Bestagini, S. Mandelli, and S. Tubaro, "Video face manipulation detection through ensemble of CNNs," in *Proc. 2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 5012–5019, doi: 10.1109/icpr48806.2021.9412711

[14] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, 2013.

[15] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," arXiv preprint, arXiv:1603.07285, 2016.

[16] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," arXiv preprint. arXiv:1603.07285, 2016.

[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. the 32nd International Conference on International Conference on Machine Learning*, 2015, pp. 448–456.

[18] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015, doi: 10.1016/j.neunet.2014.09.003

[19] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Jul. 2011.

[20] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.