Article

# Fighting Deepfakes Using Body Language Analysis

**Robail Yasrab [1,*], Wanqi Jiang [2] and Adnan Riaz [3]**

[1] Department of Engineering Science, Institute of Biomedical Engineering, University of Oxford, Oxford OX3 7DQ, UK
[2] School of Computer Science, University of Nottingham, Nottingham NG8 1BB, UK; psywj5@nottingham.ac.uk
[3] School of Computer Science and Technology, Dalian University of Technology, Dalian 116024, China; adnanriaz107@mail.dlut.edu.cn
[*] Correspondence: robail.yasrab@eng.ox.ac.uk

**Abstract:** Recent improvements in deepfake creation have made deepfake videos more realistic. Moreover, open-source software has made deepfake creation more accessible, which reduces the barrier to entry for deepfake creation. This could pose a threat to the people's privacy. There is a potential danger if the deepfake creation techniques are used by people with an ulterior motive to produce deepfake videos of world leaders to disrupt the order of countries and the world. Therefore, research into the automatic detection of deepfaked media is essential for public security. In this work, we propose a deepfake detection method using upper body language analysis. Specifically, a many-to-one LSTM network was designed and trained as a classification model for deepfake detection. Different models were trained by varying the hyperparameters to build a final model with benchmark accuracy. We achieved 94.39% accuracy on the deepfake test set. The experimental results showed that upper body language can effectively detect deepfakes.

**Keywords:** imaging; machine learning; deepfake; human pose estimation; upper body language; computer vision; deep learning; Recurrent Neural Networks (RNNs); Long Short-Term Memory (LSTM)

## 1. Introduction

Deep learning has been effectively used in an extensive range of fields to solve complicated problems, like image segmentation and classification [1], fraud detection [2], medical image analysis [3], plant phenotyping [4,5], etc. However, it has also been used to develop applications that can pose a threat to people's privacy, like deepfakes. Traditionally, digital images were manipulated through vision-based tools like Adobe Photoshop. Manually processed images could be easily distinguished. However, synthetic images are becoming increasingly convincing due to the fast development of deep learning method, which has led to the popularity of deepfakes. Deepfake methods are gaining attention, as they are currently able to manipulate media in a way that another person's face can replace an original face, while the original facial expressions and actions are retained. The problems of deepfakes arise along with the advances in deep learning. Deepfakes are becoming more and more realistic. It is becoming difficult or even impossible for people to discern if the images and videos are real or fake.

Because the amount of data required for training a deepfake model is enormous, some deepfakes are conspicuous due to the lack of data. Therefore, it is very easy to target celebrities and world leaders who have plenty of images and videos on multiple media platforms, which leads them to being the main targets of seemingly real looking deepfakes. Disinformation could be misleading or even damaging, due to the rapid spread of information on the large-scale platform of the Internet. According to Chesney et al. [6], deepfakes usually mislead the general public. However, these are a severe problem for national and societal security if they are used for political purposes. For instance, deepfakes of world leaders, like Barack Obama and Donald Trump, have sparked heated debates. Deepfakes created a heated worldwide debate when President Trump shared a deepfake GIF of Joe Biden [7]. Later, that Tweet was deleted, though its spread was a worldwide concern; deepfakes could be politically dangerous in the near future.

In recent years, it has become very easy to create a deepfake of any object due to the extensive evolution of machine learning apps (like Zao and FaceApp) backed by facial manipulation engines [8]. TikTok has also been introducing filters that can create very realistic deepfakes. There are a number of social media platforms on which people extensively post realistic looking deepfakes of celebrities.

Recently, Tom Cruise appeared on the TikTok platform and gained 486.2 K followers with 1.4 million likes in a few days. Later, it was revealed that he never had an account on TikTok, and it was an artist who created the account "deeptomcruise" and created these very realistic Tom Cruise deepfakes [9]. This proved that deepfakes could damage someone's credibility or help to spread disinformation. Therefore, it is essential to help the audience identify fake media and protect people from deepfakes. Many companies and institutes have launched competitions, such as the Deepfake Detection Challenge (DFDC) [10], to explore innovative technologies for deepfake detection. Traditional forensics approaches, including signal-level cues, physical-level evidence, or semantic-level consistencies, have not been sufficiently useful or efficient in deepfake detection [11]. Additionally, these techniques are not robust against changing conditions that are caused by resizing and compression operations [12]. Thus, deepfake detection methods that are based on deep learning or other machine learning technologies exist, but these approaches also tend to be short lived because the deepfake creation methods are being continuously improved. This research aimed to create a novel deepfake detection method that can deal with emerging deepfake threats and cope with the possible improvements of the deepfake creation methods.

When considering the magnitude of problems caused by fake videos, especially for world leaders, this research focused on automatically protecting world leaders from deepfake videos by using deep learning techniques. The speeches of world leaders can have a significant impact on a country or the world. Most world leaders present debates behind a lectern, thereby only exposing the upper half of the body. We hypothesized that upper body movements are distinct for different individuals, and deep learning networks can utilize upper body language to identify the corresponding people and expose the deepfakes. The upper body pose consists of the keypoints of eyes, nose, neck, shoulders, elbows, and wrists. These keypoints can be used to train a deep neural network to learn someone's distinct posture and gestures. Figure 1 presents the proposed design. The key contributions in this work can be described, as follows.

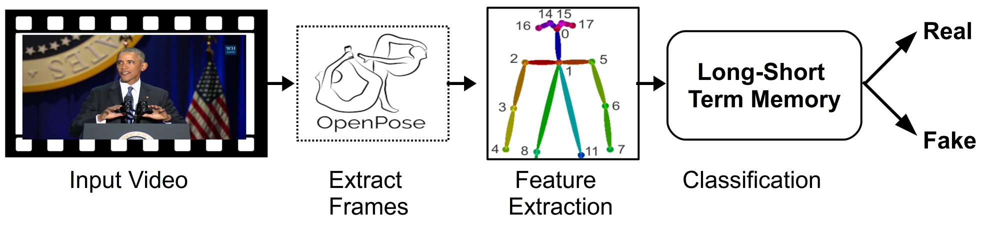

**Figure 1.** Deepfake detection: In this deepfake detection process, the video frames are extracted, and a machine learning algorithm is trained on them. Later, the inference system classifies the given video as a deepfake or not.

- To provide a general literature review of deep learning technologies, deepfake detection, and human pose estimation.
- To develop a human pose estimation program to detect upper body keypoints.
- To implement a deep learning network to learn the upper body languages of the target world leader.

- A comprehensive dataset of world leaders, which could serve as a pre-training arrangement for the smaller dataset (with less real/fake data samples).
- To examine the effectiveness of detecting deepfakes through upper-body language analysis.

The introductory section presents the problem of deepfake, the necessity of creating deepfake forensics, and possible outcomes. This research aims to detect deepfake videos of world leaders using upper body language analysis. The goal is achieved through providing a literature review of deepfake in Section 2. Section 3 presents the proposed methodology, data collection, and network implementation. The following Section 4 provides the results and discussion.

## 2. Literature Review

Human pose estimation refers to a mechanism that locates people's body parts in digital images or videos. The human pose estimation problem includes the single-person pose estimation problems, 3D skeleton estimation problem, multi-person pose estimation problem, and video pose tracking problem [13]. MPII [14] and COCO [15] are two public-available datasets for 2D pose estimation. The traditional methods for pose estimation are vision-based that are used in two contexts. The first is the feature representation like a histogram of oriented gradients (HOG) and the second is the spatial context. In contrast, deep learning-based methods are widely used these days because of the rapid development capability. Convolutional Pose Machine (CPM) is one of the essential pose estimation methods proposed by Wei et al. [16]. Jain et al. [17] used CNN to estimate the single-person skeleton. A CPM consists of a sequence of CNNs, which can learn image features and spatial context simultaneously. Some researchers have applied CPM in their design, for instance, OpenPose, an open-source real-time 2D pose detection system that was released by Cao et al. [18]. This has applied the CPM refinement and won first place in the COCO-2016 Keypoint Challenge. Cao et al. [18] created Part Affinity Fields (PAFs), a novel bottom-up representation of association scores, where the location and orientation of limbs are encoded as 2D vector fields. Newell et al. [19] also proposed a bottom-up method in the COCO 2016 Keypoint Challenge. This approach simultaneously performs detection and grouping by applying associative embedding. Based on CPM development, deepfake creation is considered to be an advanced human pose estimation and manipulation stage.

Deepfake creation is a combination of computer vision algorithms and deep learning techniques. The unsupervised image-to-image translation framework, as developed by Liu et al. [20], is the basis of deepfake generation, which is established on the coupled Generative Adversarial Network (GAN). The networks for deepfake creation use two sets of encoders-decoders, consisting of a standard encoder using shared weights for two network pairs and two different decoders, according to Nguyen et al. [21]. The encoder aims to learn the similarities between two different faces. Given an image of an aligned face as the input, a fake face can be produced using a standard encoder and decoding. Figure 2 shows the process of creating a fake face. Deepfake conversion is an essential step in the process of deepfake creation. Li et al. [22] indicated that the deepfake generation process includes face detection, face landmarks extraction, face alignment, deepfake generation, affine warping, and post-processing, such as boundary smoothing. Meanwhile, advanced deepfake creation methods are also developed; for example, GAN-based deepfake creation adds the adversarial and perceptual losses to the original network [21]. Additionally, Mirsky et al. [23] indicate that current deepfake creation uses various combinations of Deep Neural Networks (DNNs), like Convolution Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and GANs. The detection and combating of deepfakes is getting harder because deepfake creation methods are getting sophisticated.

Several deepfake detection methods have recently emerged. These methods/algorithms are fundamentally divided into (a) fake image detection and (b) fake video detection [21]. AI-based video synthesis is a relatively new technology. Some forensic techniques have successfully addressed general deepfake detection problems that are based on a wide range of different technologies. Besides, a deepfake detection tool using Support Vector Machine (SVM) has been specifically developed for world leaders [24]. However, these methods tend to be transient, because new deepfakes can fix corresponding imperfections in a shorter time [25]. Thus, it is necessary to expose new imperfections of deepfakes and develop new detection algorithms or improve existing approaches.

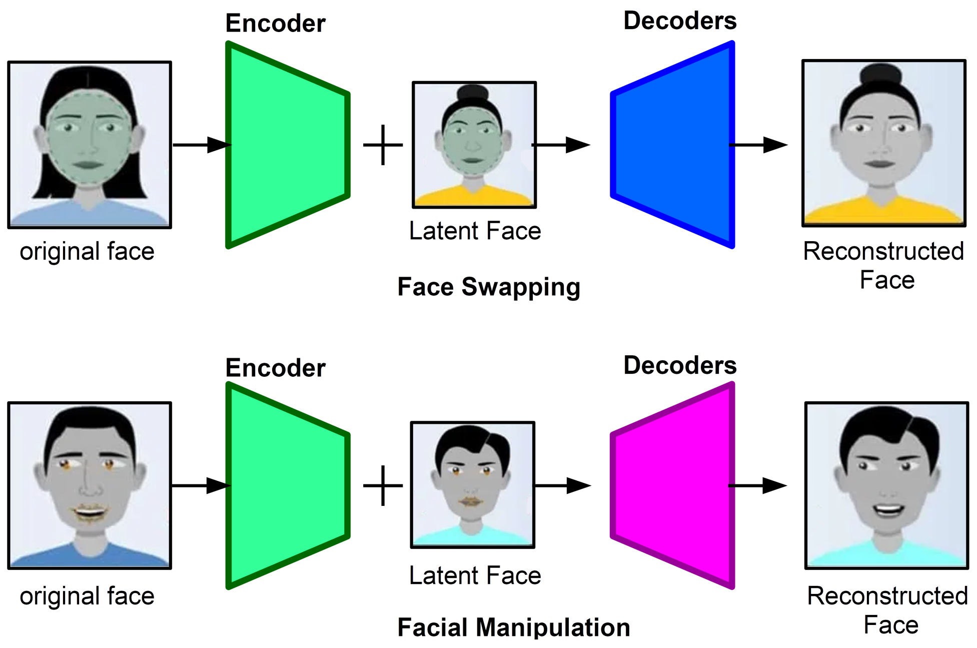

**Figure 2.** Process of Deepfake Creation: The original face is extracted and processed by the DNN Encoder. The latent face image with dormant features feed to DNN Decoder along with encoder input. The final reconstructed face image is produced from both images (original face and latent face).

Most research on fake video detection has used deep learning to detect artifacts and inconsistencies of the video. Li et al. [11] used a CNN and Long-term Recurrent Convolutional Network (LRCN) to detect unnatural blinking patterns in the video. The resultant videos contain irregular blinking frequency because of the lack of images (with closed eyes) for the training and deepfakes generation. However, Korshunov et al. [26] claimed that the effectiveness of eye blinking detection for exposing GAN-based deepfakes is unclear because the videos being used for training the model were collected from the web. In addition, eye blinking detection is not effective at present, because blinking was quickly incorporated into the new generation of deepfake technique [25]. Li et al. [11] proposed another detection method that uses CNNs to detect the resolution inconsistency between facial area and other areas that are caused by the affine wrapping process of deepfake creation. They trained four types of CNN models, including a VGG16 [27], and three different ResNets [28]. The results have higher accuracy than earlier benchmark approaches when tested with two public GAN-based deepfake datasets. This approach effectively generated negative data by applying the Gaussian blur to the facial area instead of spending a large amount of time implementing a network and training a model for deepfake generation as other approaches have. Likewise, Güera et al. [29] indicated intra-frame inconsistency due to the seamed fusion of face and the rest of the frame. They also demonstrated that multiple camera views and various lighting conditions could cause frame-level inconsistency between video segments. Their method uses CNN to extract frame-level features and use Long Short-term Memory (LSTM) to extract features in sequences of frames.

Zhang et al. [30] proposed a novel method that was based on the GAN simulator. This simulator approach frees developers from accessing the actual GAN model to generate forgery images when training classifiers. They reviewed the artefacts that were caused by GAN's up-sampling module. The proposed model achieved excellent performance in terms of binary classification of real and fake images.

Another method that was proposed by Jain et al. [31] detects GANs and retouches based digital alterations. The proposed architecture has three hierarchy levels to broadly classify and distinguish input images as original or altered, followed by subsequent levels in the architecture. The framework can detect retouching and GANs generated images with high accuracy, showing its applicability in real-life scenarios.

Guarnera et al. [32,33] proposed a similar approach that extracts deepfake fingerprints from images. Furthermore, the fingerprint extraction method is based on the expectation-maximization algorithm, emphasizing the extraction ability of convolution traces that are embedded by the generative process. In this paper, a random forest classifier is used to determine the pipeline of fakeness detection. The authors highlighted different CNN-based approaches for deepfake detection that are required highly computational power. The proposed technique obtained excellent classification results by only using CPU power. Nevertheless, Agarwal et al. [24] argued that simple manipulations, like additive noise, recompression, and resizing, could easily eliminate the inconsistencies mentioned in the above two methods.

In addition to deep learning methods, some methods rely on supervised machine learning models (SVMs). For instance, Yang et al. [34] introduced an SVM classifier to expose deepfakes by inconsistent 3D head poses, and argued that the difference between head poses is a useful feature for deepfake detection. The SVM classifier was evaluated on two datasets using five different features of estimated 3D head poses. However, most of the experimental results achieved an Area Under the Receiver Operating Characteristic (AUROC) of around 0.9, which shows that using these features is insufficient for deepfake detection.

Moreover, Dang et al. [35] evaluated the inconsistent 3D head pose on Celeb-DF (a new deepfake dataset that was created by the improved deepfake algorithm [36]) and only obtained an AUROC of 0.548. Almost all of the existing methods are general for various deepfake videos, but Agarwal et al. [24] created an SVM classifier that was specialized for world leaders, which uses biometric patterns, including facial action units (AU) and head movements. This method is based on the observation that each individual has unique facial expressions and behaviors, so that the manipulations cannot destroy it on videos, such as recompression, as mentioned above. These SVM models only need original videos for training, which saves time. They effectively trained different binary SVM models for five world leaders, and the average results achieved an AUROC of about 0.948. On the other hand, Agarwal et al. [24] also indicated that their method might fail for an individual speaking in different scenes. Furthermore, it is reported that it would not take a long time for deepfake techniques to overcome these deficiencies [25].

As deepfake techniques improved aggressively, deepfake detection methods are no longer useful or will be ineffective in time. Additionally, deepfake poses detection challenges, in that there are more datasets released for deepfake forensics with higher quality. As a result, it becomes a cycle that these two opposing techniques continue to learn and reinforce each other. This requires more advanced forensic techniques to protect people from deepfakes. Agarwal et al. [24] mentioned that body movement could be used to identify the persons of interest, but they used head movements without taking body movements into account. Deepfake techniques only temper with the facial area and manipulate inconsistencies, so the detection methods generally concentrate on facial areas and head poses. Therefore, utilizing upper body languages for deepfake detection is a new and promising research direction.

## 3. Method

The proposed method is based on the hypothesis that Deep Neural Networks (DNNs) can learn an individual's body language and expose deepfakes. In a video, the poses of a specific person should be estimated using an automated approach. The proposed idea is to train an RNN model to learn the target person's pose (body language) and spot fake videos.

The proposed system was designed in PyTorch [37] deep learning framework. Figure 3 demonstrates the standard framework of the proposed system.

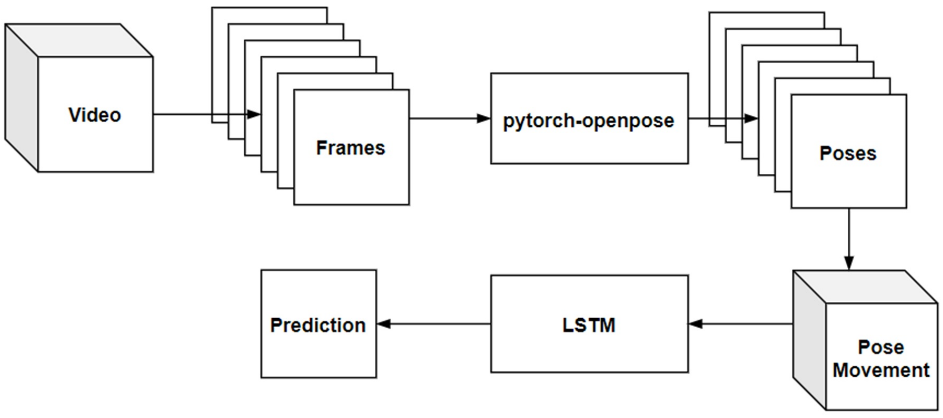

**Figure 3.** The proposed deepfake detection method: The proposed method will extract the frames (30 fps) from the input video. Later OpenPose software is used to extract key-point of each pose of the person. The proposed LSTM architecture trains the pose movement. The final inference system will classify that given a video of the US president is fake or real.

## *3.1. Dataset and Preprocessing*

The proposed idea is to detect deepfake videos using human-pose estimation. The publicly available datasets are not appropriate for this research as those do not fit the proposed face and body language landmark requirements. The only videos that we can train and test the established hypothesis are the videos of world leaders (i.e., president, vice president, etc.). These videos are widely available with a good quality pixel ratio, and we can also test these videos against deepfakes very quickly. Therefore, we manually downloaded online videos to generate a customized dataset. These videos were annotated and labelled according to the proposed requirements. We need two kinds of videos, the original and the synthetic, in order to test the proposed hypothesis. The original videos were downloaded from the official website of Miller Center [38] to ensure that the videos have not tampered. This website streams US presidential speeches. Initially, we chose videos of the four most recent United States (US) presidents George W. Bush, Barack Obama, Donald Trump, and Joe Biden. The videdownloadedos satisfy conditions (a) the file formats are MP4 with 30 frame-per-second (fps) quality, (b) a similar video frames size for the whole dataset, (c) the person of interest was in the middle of the frame, (d) there is no one else in the background, and (e) the cameras were relatively stable. Figure 4 shows the dataset samples of real and fake videos for deepfake training.

The deepfake videos of the US presidents were downloaded from YouTube. We downloaded deepfakes of the four (aforementioned) US presidents, where impersonators posed the same body language as Person-of-Interest (PoI). The information to be analyzed in this research is the upper body language. An upper body movement can be represented as a sequence of the upper body poses. For necessary image processing, the OpenCV was used to extract the frames of videos. OpenCV is an open-source library for image processing and computer vision problems. To extract the upper body keypoints in each video frame, we first tried the DNN of OpenCV for loading the deep learning model of OpenPose [39]. OpenPose performs well in human pose estimation; it trained three Caffe models, which are BODY_25, COCO, and MPI models. The output format is the difference among these models, the BODY_25 format has 25 and the COCO format has 18 keypoints. We decided to use COCO-style keypoint, as it was better aligned to our proposed dataset and keypoints requirements.

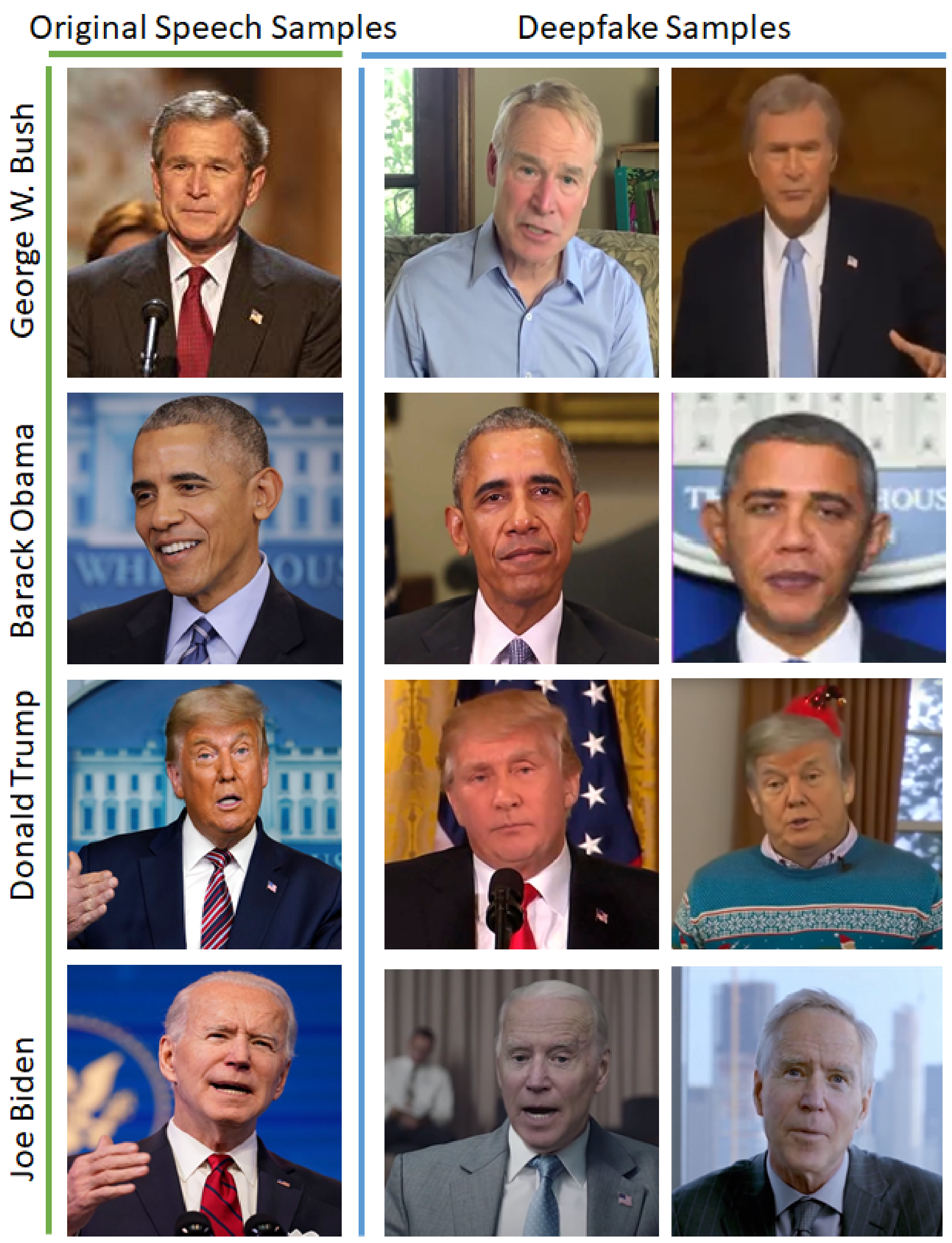

**Figure 4.** Dataset Sample: The dataset build for the proposed experiments is composed of four recent presidents of the United States. We did not extract videos of earlier presidents due to the unavailability (very few) deepfake samples.

The contributors of OpenPose recommend the BODY_25 model because of its higher accuracy and speed [39]. Although, we cannot use this model due to the unavailability of lower body keypoint. Therefore, we decided to use the COCO format. We have used PyTorch-OpenPose, which is a PyTorch implementation of the Caffe model of OpenPose. The output format is the COCO dataset format. Figure 5 demonstrates an example pose. It shows that the PyTorch-OpenPose is more accurate than the original OpenPose. Additionally, it is approximately five times faster because CUDA supports PyTorch.

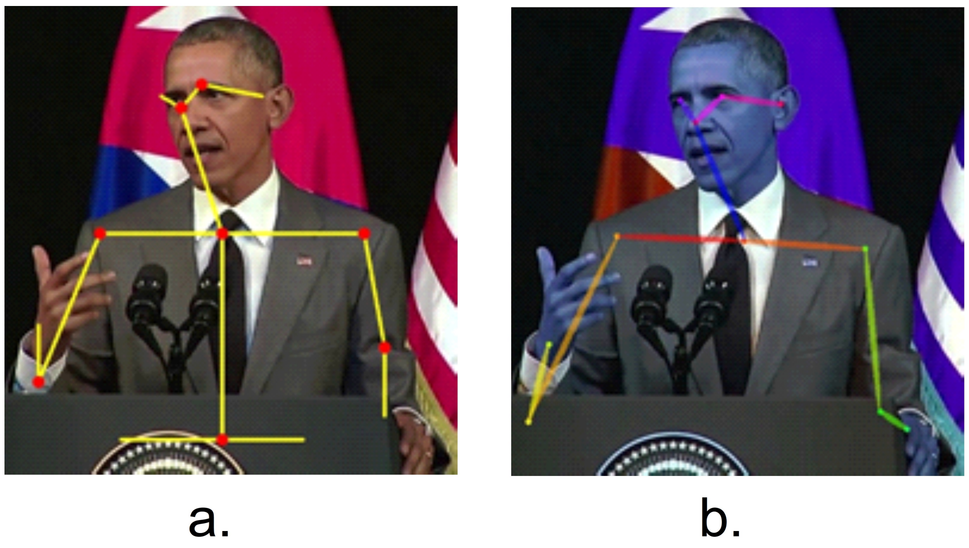

**Figure 5.** Sample Pose's Keypoint Extraction: (**a.**) Using the BODY_25 model of OpenPose and the DNN module of OpenCV. (**b.**) Using the PyTorch COCO model of PyTorch-OpenPose.

OpenCV is used for the frame reading and PyTorch-OpenPose for pose estimation. There is always a lectern in front of the PoI while giving the speech. Thus, only 12 keypoint could be detected out of 18 standard COCO formats. Table 1 lists these 12 extracted keypoints. A new output format was generated to satisfy the proposed research's requirements, where the coordinates represent each keypoint. During the upper pose movement, some of the keypoints are undetectable, where we replaced them with undetected locations (zero values). Therefore, the pose of a frame is a 1-D array that contains 24 features. An automated script is written to process a list of videos (frame by frame). It was considered that there should be enough frames to trace a pose movement. We chose to use every segment of videos (that fulfills the aforementioned requirements) to analyze the upper pose movement. The dataset annotations are saved as a CSV file, where each row is the pose of a frame and every 150 rows is a continuous pose movement. Each pose movement is labelled with 0 or 1, and the corresponding labels were saved in another CSV file. Table 2 provides a summary of the final dataset.

**Table 1.** Keypoint Sampling: The proposed experiments use input video samples, where PoI's upper body language is used to train the machine learning algorithm. Therefore, we selected a COCO-style keypoint dataset sampling method. This pose sampling method extracts 18 human body keypoint. Though for our proposed method, we only extracted 12 samples because the rest of the PoI body was hidden behind the lectern.

| Keypoint | Body Part | Keypoint | Body Part | Keypoint | Body Part |
|---|---|---|---|---|---|
| 0 | Nose | 1 | Neck | 2 | Right_Shoulder |
| 3 | Right_Elbow | 4 | Right_Wrist | 5 | Left_Shoulder |
| 6 | Left_Elbow | 7 | Left_Wrist | - | - |
| 14 | Right_Eye | 15 | Left_Eye | 16 | Right_Ear |
| 17 | Left_Ear | | | | |

**Table 2.** Summary of the dataset: A comprehensive dataset was sampled from speeches of four recent US presidents. The Miller Center website is used to extract authentic video samples. For Deepfake samples, we trawl YouTube to extract related samples. These samples were further segmented into functional segments for keypoint extraction and training purposes.

| Person of Interest (POI) | Total Videos (Count) | Video Duration (Hours) | Segments (Count) |
|---|---|---|---|
| **Real** | | | |
| George W. Bush (2001–2009) | 35 | 11.56 | 560 |
| Barack Obama (2009–2016) | 46 | 15.33 | 689 |
| Donald Trump (2017–2020) | 38 | 12.89 | 542 |
| Joe Biden (2021) | 4 | 1.9 | 32 |
| **Fake** | | | |
| George W. Bush | 6 | 0.23 | 12 |
| Barack Obama | 23 | 0.36 | 54 |
| Donald Trump | 35 | 0.56 | 66 |
| Joe Biden | 4 | 0.10 | 8 |

The proposed dataset is divided into training, validation, and testing divisions, accounting for 80%, 10%, and 10%, respectively. The values of the features are scaled to $[-1, 1]$ using Z-score normalization:

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

where $z$ is the normalised data, $x$, and $\mu$ and $\sigma$ are the original value, mean value, and standard deviation, respectively. Z-score normalisation may produce "NaN" values because of dividing by zero. For instance, the left wrist keypoint could not be estimated in a 150-pose sequence, so that consecutive zero values exist. The $\mu$ and $\sigma$ both are zeros. Therefore, the NaN values that are generated by Z-score normalization are replaced with zero values.

### *3.2. RNN Architecture*

The deep learning model was trained to learn the spatial and temporal data in order to address fake videos using body language analysis. The proposed method will be an automated system that can determine whether the input pose movement belongs to the target person or not. To implement the proposed method, the RNN is the best candidate to capture the dataset's spatial and temporal features. Furthermore, it can be applied to learn the dependencies of sequential data along with graphical features. LSTMs are advanced RNNs, which can learn long-term dependencies without gradient vanishing. Figure 6 shows a structural comparison between RNN and LSTM. LSTM is very effective for learning spatial and temporal features. Thus, the proposed model is a many-to-one LSTM that requires continuous poses as the input, and it predicts a fixed size output.

The original LSTM comprises an input layer, hidden layers (with recurrent edges), and an output layer. Multiple hidden layers could be stacked according to the complexity of the problem. The unique architecture of the hidden units of the LSTMs is the key to the improvement of the standard RNNs. The hidden nodes of RNNs use simple nonlinear functions, like the Sigmoid and Tanh, while LSTMs replace the hidden nodes with memory cells [40]. The memory cells use three types of gates to manage the cell state, including the input, output and forget gates. Each gate is composed of an element-wise product and a sigmoidal node with an output range between [0, 1]. Thus, multiplying the sigmoidal unit's input and output can limit the information flowing through the gate. For example, 0 means that no information can pass through, and 1 denotes that all information can pass through the gate. A standard LSTM memory cell is demonstrated in Figure 7; the architecture is composed of forget, input, and output gates, respectively. The forget gate can drop unnecessary information retained in the previous cell state, then the new information currently to be retained is determined by the input gate. Finally, the output gate determines the output information, depending on the current cell state.

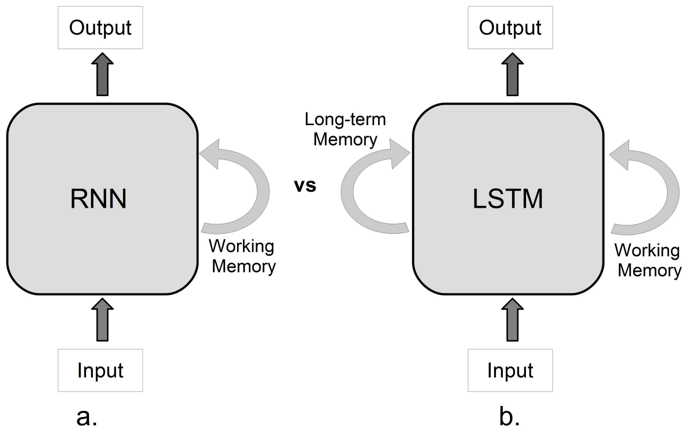

**Figure 6.** RNN v/s LSTM. (**a**): RNNs use their internal state (memory) to process sequences of inputs, (**b**): LSTM network is a variant of RNN, with additional long-term memory to remember past data.

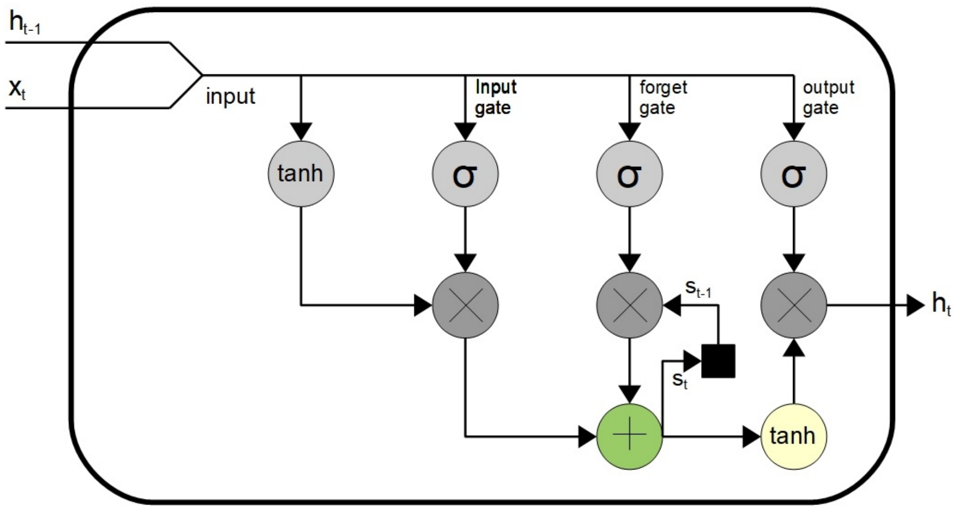

**Figure 7.** An unrolled LSTM. It takes input sequence $x_t$ and $h_{t-1}$, process this input through different gates (input, forget, out) and produces the output that could serve as the final state or input for the next hidden state.

There are many LSTMs variants Greff et al. [41], whereas we have used the Many-to-One LSTM model. This variant was designed using PyTorch. PyTorch implements the normal LSTM as the memory cell. Each normal memory cell computes the following calculations:

$$\begin{aligned} f_i &= \sigma(W_{if}x_t + W_{hf}h_{t-1} + b_f) \\ i_t &= \sigma(W_{ii}x_t + W_{hi}h_{t-1} + b_i) \\ g_t &= \tanh(W_{ig}x_t + W_{hg}h_{t-1} + b_g) \\ o_t &= \sigma(W_{io}x_t + W_{ho}h_{t-1} + b_o) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \tanh c_t \end{aligned} \quad (2)$$

where, $*_t$ means the time step; while, $t, f_t, i_t, g_t$, and $o_t$ are the gates. $\sigma$ denotes the Sigmoid, $W_*$ variables are recurrent weight matrices, $x_t$ represents the input, $h_t$ and $h_{t-1}$ are the hidden states at the $t^{th}$ and $t-1^{th}$ time steps, $b_*$ variables are the biases, $c_t$ and $c_{t-1}$ are the cell states at the $t^{th}$ and $(t-1)^{th}$ time steps, and $\odot$ is the element-wise product. LSTMs have two different hidden states, which are $h_t$ and $c_t$ states [42]. Therefore, LSTMs can learn short-term dependencies that are based on the state $h_t$, and the problems of gradient vanishing and long-term dependencies are addressed by the use of the state $c_t$.

## 3.3. Experimental Setup

To test the proposed hypothesis that body languages can identify and expose deepfakes, a many-to-one LSTM model was designed using PyTorch. PyTorch provides flexible DNN implementation and GPU support. Two kinds of objects are necessary for setting up the training and testing experiment: (a) the object for data loading and (b) the object for model creation. The custom dataset was divided into an 80% training set, a 10% validation set, and a 10% testing set. Moreover, the values were normalized to the range of $[-1, 1]$. A specially designed (according to needs of our custom dataset) DataLoader that was used for passing the datasets to the LSTM model for training. The data only contain the body languages with 24 keypoints, the model to be implemented with a binary classification model having an input layer with 24 features. It also contains a fully connected output layer with a single output unit. The many-to-one LSTM model outputs the prediction at the last time step. It is observed that every 150 poses are a continuous movement; therefore, the time step of the model is 150. The hidden size and the number of hidden layers should be decided based on the experiment. The proper depth and width of the network can prevent underfitting and overfitting problems. The initial setup was based on the original LSTM architecture, which only has a single hidden layer. A single hidden layer cannot be followed by a dropout layer, so that the initial dropout was zero. A stacked LSTM can be produced by setting more than one hidden layer, so we have initialized with the hidden size to 128–512. The loss function and optimizer are two significant components for deep learning. We have used binary cross-entropy as the loss function because this is a binary classification model. Subsequently, the cross-entropy measures the difference between the predicted and real distributions. We have used the Adam optimizer for training purposes. Optimization algorithms are grouped into adaptive methods, like Adam [43], and non-adaptive approaches, like stochastic gradient descent (SGD). Adam adapts learning rates for different intrinsic parameters of the network [44]. The learning rate is a hyper-parameter that substantially impacts the convergence and generalization performance of the model. A high learning rate makes the model learn very fast at the beginning of the training, but the loss may decrease later. On the other hand, the model may fail to converge due to the low learning rate. There are various approaches to adjust the learning rate, such as the adaptive gradient methods and learning rate decay. Though adaptive methods (Adam) have adapted learning rates, Loshchilov and Hutter [45] provide evidence that using rate decay with Adam can improve the performance of Adam. In addition to the learning rate, optimizers in PyTorch have an optional parameter, called 'weight_decay'. The weight decay parameter is a regularisation coefficient that can add an L2 penalty to the loss function to prevent overfitting. Thus, we also experimented with the learning rate decay and weight decay. The initial learning rate was fixed to $1^{-3}$ without the learning rate decay. Batch size and epoch are two significant hyper-parameters to determine the number of iterations, affecting the model's generalization performance. The number of iterations *epoch* $\times$ *(data size* $\div$ *batch size)* determines how many times the weights are updated. Assuming that the epoch is fixed, the iterations will decrease with the increment of the batch size. Using large batch sizes can reduce the training time and enhance the stability. It also reduces the generalization performance. Using a small batch size can avoid local optima, but the model may be hard to converge. Keskar et al. [46] indicate that small batch sizes make a better generalization. We first used 200 epochs with a batch size of one. The vanishing and exploding gradient problems are also crucial to RNNs. LSTMs have successfully fixed the vanishing gradient problem, but the exploding gradient problem still exists, which causes NaN losses during the training. Applying gradient clipping is a solution to the exploding gradient problem. Therefore, we set the maximum norm of the gradients to 0.25. We transferred PyTorch tensors to CUDA tensors for computation and sent the model to a GPU. Since the output layer outputs a score, the final prediction can be obtained by applying a Sigmoid function and then rounding off the predicted probability. The model weights were optimized using the training set at each epoch during the training and validation experiment. Subsequently, the model's performance was evaluated by testing on the validation data. After running the specified epochs, the best-performing model was re-evaluated using the testing data to check whether the model can generalize well to new data. Overall, the experiment started with the basic LSTM model that has a single hidden layer. Fixed hyper-parameters included 24 input nodes, one output node, and 150 time-steps. The losses were calculated using binary cross-entropy. Moreover, gradient clipping was also applied. Other hyper-parameters, such as the hidden size, were first set to 128, the dropout probability was 0, the batch size was initialized to 1, and the epoch was 200. The experiment adjusted hyper-parameters, including different optimizers, learning rates, batch size, and tried decay methods, to train a practicable LSTM model for deepfake detection.

### *3.4. Evaluation Metrics*

Loss and Accuracy metrics were used to evaluate the model's performance. The binary cross-entropy loss function was applied for experiments. A lower testing loss value projects a more similar predicted distribution to the target distribution. We plotted the loss and accuracy over time to check whether the model was underfitted or overfitted. Whereas, underfitting means that the model fails to converge and overfitting refers to a model having poor generalization performance. This circumstance may happen due to inadequate training data. The generated dataset used for this project is small; high confidence may lead to overfitting. Therefore, we adjusted the hyper-parameters according to both losses and accuracy, while choosing the model with the highest validation accuracy to be the best model.

## 4. Results

The following chapter will present the results of model training and testing. The final version of the LSTM model was trained through extensive hyper-parameters adjustments. The proposed deepfake detection method was tested through an inference procedure.

Step one was to determine the appropriate optimizer. For this process, two optimizers, including the SGD and Adam, were tested. We first tried the SGD with a learning rate of $1e^{-3}$. The results are shown in the subfigures (a) and (b) of Figure 8. Over time, the training and validation losses implied that the learning rate was too high, so that the model failed to converge. Therefore, we reduced the learning rate to $1e^{-4}$. The subsequent results are presented in the subfigures (c) and (d) of Figure 8. The lower learning rate made the loss convergence slow, but the model with an accuracy of about 70% was still underfitted. Subsequently, we used Adam with a learning rate of $1e^{-3}$; the results are shown in the subfigures (e) and (f) of Figure 8. The training loss drastically decreased and approached zero, but the validation loss was unstable and getting higher. Thus, the model was overfitted. The above results show that it is hard to adjust the learning rate for the SGD, so Adam is more suitable for this model. Thus, we decided to use Adam to be the final optimizer.

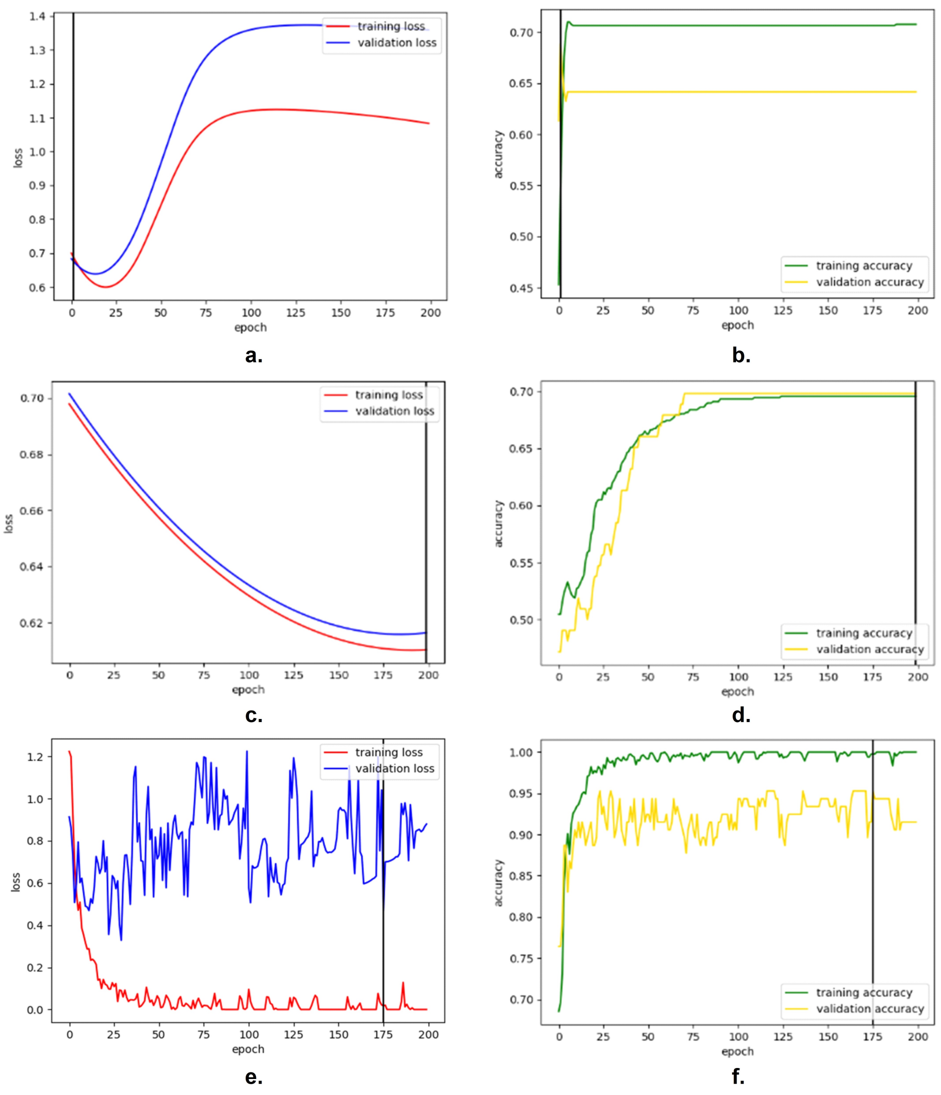

**Figure 8.** Loss and accuracy results for two different optimizers. (**a**,**b**) are results of using SGD and lr = $1e^{-3}$, (**c**,**d**) are results of using SGD and lr = $1e^{-4}$, (**e**,**f**) are results of using Adam and lr = $1e^{-3}$, where lr is the learning rate. The black vertical line in each subfigure is the epoch having the highest validation accuracy.

The next step was to solve the overfitting problem. It is assessed that dropout is an effective method for addressing this problem. The dropout for an LSTM model with a single hidden layer is zero. Therefore, we changed the number of hidden layers to 2 and set the dropout probability as 0.5. A stacked LSTM model was the modified model. Figure 9 shows the results. After the modification, the overfitting problem still existed, but the results revealed that the validation accuracy was more significant than using a single hidden layer. Furthermore, the learning rate also needed to be tuned. A reasonable learning rate should make the loss decrease gradually.

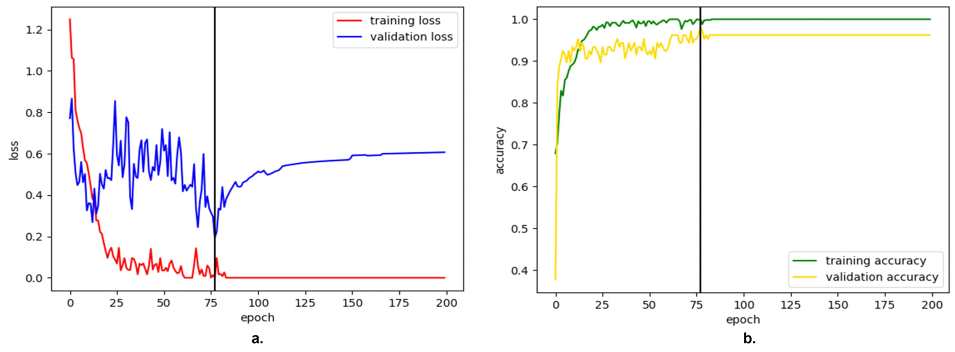

**Figure 9.** The results of using two hidden layers with a dropout probability of 0.5. (**a**) Training and validation for model training. (**b**) Training and validation for accuracy training.

The next step was to solve overfitting and adopt an appropriate learning rate. For this purpose, we applied learning rate decay (0.5 for every 30 epochs) with the initial learning rate of $1e^{-2}$. Besides, the weight decay with a $\lambda$ coefficient of $1e^{-4}$ was used to prevent overfitting. The results that are presented in Figure 10 show that both the losses and accuracy remained virtually unchanged during the first 30 epochs, but the values gradually decreased after the first learning rate decay. The best model with the highest validation accuracy was at 165th epoch. Table 3 provides the final hyper-parameters.

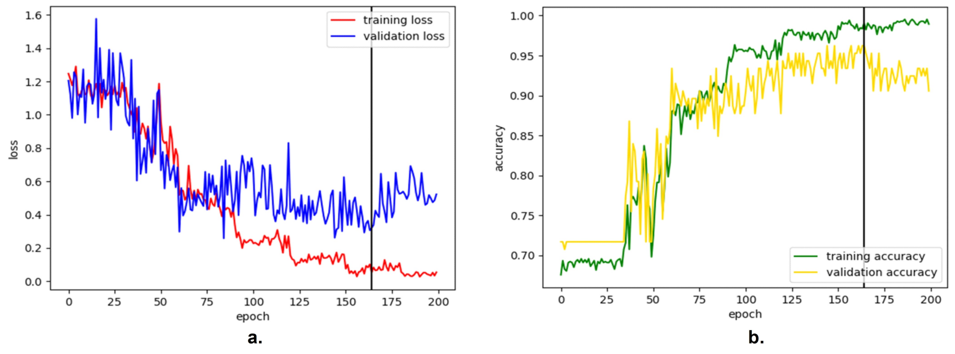

**Figure 10.** Results of applying learning rate decay and weight decay. (**a**) Training and validation for model training. (**b**) Training and validation for accuracy training.

**Table 3.** Network Architecture and Hyper-parameters: Several experiments were performed to finalize this architecture and hyper-parameters settings. These settings offered the most relevant outcomes.

| **Input Size** | **Hidden Size** | **Hidden Layers** | **Dropout** | **Output Size** | **Optimizer** | **Learning Rate Decay** |
|---|---|---|---|---|---|---|
| 24 | 512 | 2 | 0.5 | 1 | ADAM | lr = 1e−2 stepsize = 30 gamma = 0.5 |
| Time Step | Weight decay | Gradient clipping | Batch size | Epoch | | |
| 150 | 1e−4 | 0.25 | 1 | 200 | | |

**Table 4.** Quantitative Comparison of different LSTM Models: this table shows a quantitative comparison of test accuracy different LSTM models trained to learn human pose. It proved that deep hidden state LSTM performs exceptionally well as compared to shallow models.

| Model A LSTM | Model B LSTM | Model C LSTM |
|---|---|---|
| 2 Hidden Layer<br>512 Hidden Units | 2 Hidden Layer<br>256 Hidden Units | 3 Hidden Layer<br>128 Hidden Units |
| 94.39% | 91.24% | 86.62% |

We trained LSTMs with different neuron depths (hidden states) to assess the most appropriate LSTM arrangement for experiments. It started with 128 neuron depth (number of neurons for LSTM's hidden state) and kept moving-up till 512 neuron depth. After evaluating the final results, this research identified that LSTMs with 512 neuron depth offered people much better learning capabilities in comparison to less deep neurons LSTMs. Therefore, 512 hidden neuron layers finalized for deepfake detection architecture. Each of these models trained with the given dataset through a single end-to-end trainable design for 200 epochs (500 for initial experiments). Table. 4 presents the quantitative results.

The model was then tested using the testing data. The training, validation, and testing accuracy of the model were 99.06%, 96.23% and 94.39%, respectively, as listed in Table 5. The model was trained using the final hyper-parameters and a single TitanX GPU for computation.

A detailed benchmark analysis was performed to assess the effectiveness of the proposed method. In this scenario, the proposed LTSM was compared to the well-known deep learning architectures. Table 5 shows the final results of the overall assessment. The proposed method performed best in terms of high detection accuracy.

**Table 5.** Training, Validation, and Testing accuracies of the final model: the model was trained and tested against several methods. We have trained the proposed dataset with the VGG16 [47], ResNet [28], and multimodel CNN+LSTM architectures. The proposed LSTM with given hyper-parameters offered the highest validation and testing accuracy.

| Model | Training Accuracy | Validation Accuracy | Testing Accuracy |
|---|---|---|---|
| VGG16 [47] | 82.1% | 76.34% | 74.03% |
| ResNet50 [28] | 91.29% | 87.64% | 84.49% |
| ResNet101 [28] | 93.00% | 89.90% | 88.00% |
| ResNet152 [28] | 96.00% | 92.90% | 92.01% |
| CNN (VGG16) + LSTM | 98.11% | 94.00% | 93.78% |
| Proposed LSTM | 99.06% | 96.23% | 94.39% |

Li et al. [1] used CNNs to detect artifacts, such as face regions and surrounding areas. The use of facial landmarks is the most appropriate way to analyze the proposed method. The proposed dataset also has upper body landmarks, which include facial landmarks. Therefore, we only trained facial landmarks using the CNN architecture, as suggested in the proposed research [1]. The results in Table 6 show that our methods offer much better performance when compared to the facial landmarks detection method.

**Table 6.** Testing Accuracies: testing performance of the proposed method with the state-of-the-art method by Li et al. [22] on the custom dataset. Li et al. [22] used VGG16, ResNet50, ResNet101, and ResNet152 CNN models for analysis. We trained the ResNet152 network using the facial landmarks and compared it with the proposed best-performing LSMT model.

| Model | Testing Accuracy |
|---|---|
| Li et al. [22] ResNet152 | 93.20% |
| Proposed LSTM | 94.39% |

## 5. Discussion

The experiments that were described earlier have successfully trained a practical model with an accuracy of 94.39%. The results have shown that pose languages could distinguish different people and expose deepfakes. This method used the PyTorch-OpenPose model for pose estimation and the LSTM model for deepfake detection. A given person's (PoI) video was assessed using a trained detection model. The PyTorch-OpenPose first estimates the upper body poses and outputs the upper body pose movement in this process. The LSTM model can then recognize whether the pose movement belongs to a specific person's body language.

In the experiments (aforementioned), the system learned the pose languages of the target person using the trained CNN model. Videos always have no less than 150 frames, which means that there is (always) more than one pose movement. The demo extracts all possible pose movements to feed the LSTM model. The video will only be marked as authentic if the number of actual pose movements is greater than that of fake pose movements.

This research has successfully addressed the problem of fighting deepfakes for world leaders. Table 5 shows the testing results. The proposed method works well in terms of detection of deepfake, although there are some limitations. The first problem is that the dataset generated in this research is small, which does not contain enough instances of the target person's pose languages (lower-body). The lack of data results leads to a more significant drop in training and validation accuracies than the testing accuracy. Fake data quality is the next challenge. The fake instances were extracted from deepfake videos (from YouTube), which may not be using the latest deepfake creation techniques. Thus, it is unknown whether this method can discern deepfakes produced by advanced creation methods. Another limitation is that the number of input sequences is fixed. An RNN model that allows for varying input sequences would be better and more flexible. Thus, this research has indicated the feasibility of the proposed method, but further research is required to address these limitations.

## 6. Conclusions and Future Work

The misuse of deepfake techniques has caused a significant problem (especially deepfake videos of world leaders being threatening to the public). Research on this issue created a novel deepfake detection method that can spot fake videos for world leaders by analyzing the upper body languages. A dataset was first generated using PyTorch-OpenPose to test the hypothesis that "the body language is distinct for different individuals and can be used to expose deepfakes by RNN" . The data pre-processing method extracted 12 body key-points that represent the upper body pose. Subsequently, a many-to-one LSTM model was designed and trained to analyze the upper body language. In the experiments, the LSTM model has proved the proposed hypothesis with high accuracy. With the fast development of deepfake creation and detection, this technology is becoming increasingly realistic. Hence, it is increasingly challenging to spot imperfections. This research has created a new research direction that is not limited to using face or body language. Over time, the deepfake creation methods are getting smarter, as are the detection methods. It is an evolutionary process that continues to evolve.

Deepfakes are continually evolving and mutating against the currently available detection methods. It is a kind of war between good and evil. Additionally to this paper, several aspects can be employed to improve and enhance the proposed research to detect deepfakes in the forthcoming works. Therefore, we now suggest the following methods to improve the underlying research for our future work:

- Collecting more real videos to produce positive instances. Instead of downloading deepfakes from the internet, developing synthetic instances of videos using advanced deepfake creation approaches.
- Training using a larger dataset might enhance the generalization performance, so the researcher can continue training the new models using transfer learning. The future researcher can create datasets for multiple world leaders and train models for different world leaders, in order to test whether the proposed method is still useful.
- Defining a new RNN network and train a model that allows input sequences with different lengths. Use the 3D head pose to replace head keypoints. The 3D pose might be more informative, because it contains spatial position instead of planimetric position.
- Using an upper-body pose and a hand pose to train the model. The PyTorch-OpenPose also provided a model for hand pose estimation. Adding a hand pose is similar to upper body languages, as different people might have distinct hand gestures.

**Author Contributions:** Conceptualization, R.Y. and W.J.; methodology, R.Y. and W.J.; software R.Y. and W.J.; validation, R.Y. and W.J.; formal analysis, R.Y. and W.J.; investigation, R.Y. and W.J.; resources, W.J.; data curation, W.J.; writing—original draft preparation, R.Y. and W.J.; writing—review and editing, R.Y., A.R. and W.J.; visualization, R.Y. and W.J.; supervision, R.Y.; project administration, R.Y.; funding acquisition, N/A. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

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

## References

1. Goodfellow, I.; Bengio, Y.; Courville, A.; Bengio, Y. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; Volume 1.
2. Fiore, U.; De Santis, A.; Perla, F.; Zanetti, P.; Palmieri, F. Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. *Inf. Sci.* **2019**, *479*, 448–455. [CrossRef]
3. Shen, D.; Wu, G.; Suk, H.I. Deep learning in medical image analysis. *Annu. Rev. Biomed. Eng.* **2017**, *19*, 221–248. [CrossRef] [PubMed]
4. Yasrab, R.; Atkinson, J.A.; Wells, D.M.; French, A.P.; Pridmore, T.P.; Pound, M.P. RootNav 2.0: Deep learning for automatic navigation of complex plant root architectures. *GigaScience* **2019**, *8*, giz123. [CrossRef] [PubMed]
5. Yasrab, R.; Zhang, J.; Smyth, P.; Pound, M.P. Predicting Plant Growth from Time-Series Data Using Deep Learning. *Remote Sens.* **2021**, *13*, 331. [CrossRef]
6. Chesney, B.; Citron, D. Deep fakes: A looming challenge for privacy, democracy, and national security. *Calif. Law Rev.* **2019**, *107*, 1753. [CrossRef]
7. Dyer, C. Trump Shares 'Deep Fake' GIF of Joe Biden Sticking His Tongue Out in Series of Late-Night Twitter Posts after His Briefing was Cut Short-Even Retweeting HIMSELF Three Times. Available online: https://www.dailymail.co.uk/news/article-8260455/Trump-shares-deep-fake-GIF-Joe-Biden-sticking-tongue-series-late-night-posts.html (accessed on 1 March 2021).
8. Qi, H.; Guo, Q.; Juefei-Xu, F.; Xie, X.; Ma, L.; Feng, W.; Liu, Y.; Zhao, J. DeepRhythm: Exposing deepfakes with attentional visual heartbeat rhythms. In Proceedings of the 28th ACM International Conference on Multimedia, Seattle, WA, USA, 12–16 October 2020; pp. 4318–4327.
9. Hern, A. I Don't Want to Upset People': Tom Cruise Deepfake Creator Speaks Out. *The Guardian*, 5 March 2021.
10. Dolhansky, B.; Howes, R.; Pflaum, B.; Baram, N.; Ferrer, C.C. The deepfake detection challenge (dfdc) preview dataset. *arXiv* **2019**, arXiv:1910.08854.
11. Li, Y.; Chang, M.C.; Lyu, S. In ictu oculi: Exposing ai created fake videos by detecting eye blinking. In Proceedings of the 2018 IEEE International Workshop on Information Forensics and Security (WIFS), Hong Kong, China, 11–13 December 2018; pp. 1–7.

12. Tolosana, R.; Vera-Rodriguez, R.; Fierrez, J.; Morales, A.; Ortega-Garcia, J. Deepfakes and beyond: A survey of face manipulation and fake detection. *arXiv* **2020**, arXiv:2001.00179.
13. Chen, Y.; Tian, Y.; He, M. Monocular human pose estimation: A survey of deep learning-based methods. *Comput. Vis. Image Underst.* **2020**, *192*, 102897. [CrossRef]
14. Andriluka, M.; Pishchulin, L.; Gehler, P.; Schiele, B. 2d human pose estimation: New benchmark and state of the art analysis. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 3686–3693.
15. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*; Springer: Berlin, Germany, 2014; pp. 740–755.
16. Wei, S.E.; Ramakrishna, V.; Kanade, T.; Sheikh, Y. Convolutional pose machines. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 4724–4732.
17. Jain, A.; Tompson, J.; Andriluka, M.; Taylor, G.W.; Bregler, C. Learning human pose estimation features with convolutional networks. *arXiv* **2013**, arXiv:1312.7302.
18. Cao, Z.; Hidalgo, G.; Simon, T.; Wei, S.E.; Sheikh, Y. OpenPose: Realtime multi-person 2D pose estimation using Part Affinity Fields. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4724–4732.
19. Newell, A.; Huang, Z.; Deng, J. Associative embedding: End-to-end learning for joint detection and grouping. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2017; pp. 2277–2287.
20. Liu, M.Y.; Breuel, T.; Kautz, J. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2017; pp. 700–708.
21. Nguyen, T.T.; Nguyen, C.M.; Nguyen, D.T.; Nguyen, D.T.; Nahavandi, S. Deep learning for deepfakes creation and detection. *arXiv* **2019**, arXiv:1909.11573.
22. Li, Y.; Lyu, S. Exposing deepfake videos by detecting face warping artifacts. *arXiv* **2018**, arXiv:1811.00656.
23. Mirsky, Y.; Lee, W. The creation and detection of deepfakes: A survey. *ACM Comput. Surv.* **2021**, *54*, 1–41. [CrossRef]
24. Agarwal, S.; Farid, H.; Gu, Y.; He, M.; Nagano, K.; Li, H. Protecting World Leaders Against Deep Fakes. In Proceedings of the CVPR Workshops, Long Beach, CA, USA, 16–20 June, 2019; pp. 38–45.
25. Vincent, J. Deepfake Detection Algorithms Will Never Be Enough. *The Verge*, 27 June 2019, Volume 27.
26. Korshunov, P.; Marcel, S. Deepfakes: A new threat to face recognition? assessment and detection. *arXiv* **2018**, arXiv:1812.08685.
27. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
28. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
29. Güera, D.; Delp, E.J. Deepfake video detection using recurrent neural networks. In Proceedings of the 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Auckland, New Zealand, 27–30 November 2018; pp. 1–6.
30. Zhang, X.; Karaman, S.; Chang, S.F. Detecting and simulating artifacts in gan fake images. In Proceedings of the 2019 IEEE International Workshop on Information Forensics and Security (WIFS), Delft, The Netherlands, 9–12 December 2019; pp. 1–6.
31. Jain, A.; Majumdar, P.; Singh, R.; Vatsa, M. Detecting GANs and retouching based digital alterations via DAD-HCNN. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 14–19 June 2020; pp. 672–673.
32. Guarnera, L.; Giudice, O.; Battiato, S. Fighting Deepfake by Exposing the Convolutional Traces on Images. *IEEE Access* **2020**, *8*, 165085–165098. [CrossRef]
33. Guarnera, L.; Giudice, O.; Battiato, S. Deepfake detection by analyzing convolutional traces. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 14–19 June 2020, pp. 666–667.
34. Yang, X.; Li, Y.; Lyu, S. Exposing deep fakes using inconsistent head poses. In Proceedings of the ICASSP 2019–2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; pp. 8261–8265.
35. Dang, H.; Liu, F.; Stehouwer, J.; Liu, X.; Jain, A.K. On the detection of digital face manipulation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 5781–5790.
36. Li, Y.; Yang, X.; Sun, P.; Qi, H.; Lyu, S. Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 3207–3216.
37. Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic Differentiation in Pytorch. Available online: https://openreview.net/forum?id=BJJsrmfCZ (accessed on 1 September 2020).
38. Center, M. Miller Center Foundation Website. Available online: https://millercenter.org/ (accessed on 1 September 2020).
39. Cao, Z.; Hidalgo, G.; Simon, T.; Wei, S.E.; Sheikh, Y. OpenPose: Realtime multi-person 2D pose estimation using Part Affinity Fields. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *43*, 172–186. [CrossRef] [PubMed]
40. Lipton, Z.C.; Berkowitz, J.; Elkan, C. A critical review of recurrent neural networks for sequence learning. *arXiv* **2015**, arXiv:1506.00019.
41. Greff, K.; Srivastava, R.K.; Koutník, J.; Steunebrink, B.R.; Schmidhuber, J. LSTM: A search space odyssey. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *28*, 2222–2232. [CrossRef] [PubMed]

42. Jozefowicz, R.; Zaremba, W.; Sutskever, I. An empirical exploration of recurrent network architectures. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 2342–2350.
43. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
44. Wilson, A.C.; Roelofs, R.; Stern, M.; Srebro, N.; Recht, B. The marginal value of adaptive gradient methods in machine learning. *arXiv* **2017**, arXiv:1705.08292.
45. Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. *arXiv* **2017**, arXiv:1711.05101.
46. Keskar, N.S.; Mudigere, D.; Nocedal, J.; Smelyanskiy, M.; Tang, P.T.P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv* **2016**, arXiv:1609.04836.
47. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.