Article

Fighting Deepfakes Using Body Language Analysis

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

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

Abstract: Recent improvements in deepfake creation made deepfake videos more realistic. Open-source software has also made deepfake creation more accessible, which reduces the barrier to entry for deepfake creation. This could pose a threat to the public privacy. It 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 the countries and the world. Research into automated detection for deepfaked media is therefore essential for public safety. We propose in this work the use of upper body language analysis for deepfake detection. Specifically, a many-to-one LSTM network was designed and trained as a classification model is trained for deepfake detection. Different models trained using various hyper-parameters to build a final model with benchmark accuracy. We achieve 94.39% accuracy on a test deepfake set. The experimental results shows that upper body language can effectively provide identification and deepfake detection.

Keywords: Imaging; Machine learning; Deepfakes; Human pose estimation; Upper body languages; World leader; Computer vision; Deep learning; Computer vision; 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 was also used to develop applications that can pose threats to the public's privacy, like deepfake. Traditionally, digital images used to be manipulated through vision-based tools like Adobe Photoshop. Manually processed images can be easily distinguished. However, synthetic images are becoming more convincing because of deep learning method's speedy development that triggered deepfakes popularity. Deepfake methods are getting better, currently able to manipulate media in a way that other people's face replaces the original face, while the original facial expressions and actions are retained. The problem of deepfakes arises along with advances in deep learning. Recent deepfakes are getting more and more realistic. It is getting 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 large, some deepfakes are conspicuous fake due to the lack of data. Therefore, it is really easy to target celebrities and world leaders who have plenty of online images and videos; that leads them to be main targets of almost real looking deepfakes. Disinformation could be misleading or even damaging due to the rapid spread of information on the large-scale internet platform. According to Chesney et al. [6] deepfake usually misleading for the general public. However, it is a severe problem for national and social security if it is used for political purposes. For instance, deepfakes of world leaders like Barack Obama

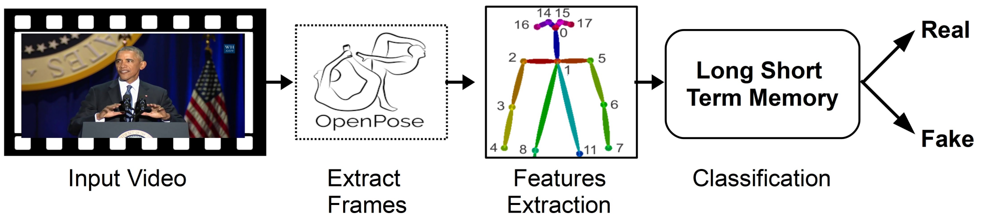

Figure 1. Deep Fake detection: In this deepfake detection process, the video frames will be extracted and trained using a machine learning algorithm. Later inference system will classify if the given video is deepfake or not.

and Donald Trump have sparked heated debate. Deepfakes created a worldwide heated debate when President Trump shared a deepfake GIF of Joe Biden [7]. Later that Tweet was deleted, though it spread a worldwide concern; deepfake can be politically dangerous in the coming future.

In recent years, due to the extensive evolution of machine learning APPs (like Zao and FaceApp) backed with facial manipulation engines, it has become really easy to create deepfakes by anyone [8]. Tiktok is also introducing filters that can create very realistic deepfakes. There are number of social media platforms where people are posting extensively realistic-looking celerity deepfakes. Recently, Tom Cruise appeared on the Tiktok platform and gained 486.2K followers with 1.4 Millon likes in just a couple of days. Later, it is revealed that he never had any account on Tiktok, and it was an artist who created the account "deeptomcruise" and created very realistic Tom Cruise deepfakes [9]. This proves that deepfake could damage someone credibility or help to spread disinformation. Therefore, it is essential to spot fake media for audiences 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 forensic approaches, including signal level cues, physical level evidence, or semantic level consistencies are not sufficiently useful or efficient for deepfake detection [11]. Additionally, these techniques are not robust against changing conditions caused by resize and compression operations [12]. Thus, deepfake detection methods exist based on deep learning or other machine learning technologies, but these approaches also tend to be short-lived because deepfake creation methods are continually improving. This research aims to create a novel deepfake detection method that can deal with emerging deepfake threats and cope with possible improvements in deepfake creation methods.

Considering the magnitude of the problems caused by fake videos, especially for world leaders, this research focuses on protecting world leaders from deepfake videos automatically using deep learning techniques. The speeches of world leaders can have a significant impact on the country or the world. Most world leaders speak from behind the lectern, only exposing the upper half of the body. We hypothesize that upper body movements are distinct for different individuals, and deep learning networks can utilize upper body languages to identify corresponding people and expose deepfakes. The upper body pose consists of the keypoints of the 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 gesture. The key contributions in this work can be described as follows.

- To provide a general literature review of technologies of deep learning, 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. It could be served 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.

We have discussed the problem of deepfake, the necessity of creating deepfake forensics, and possible outcomes in the introductory section. 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. The proposed methodology of data collection and pre-processing, network implementation, and model training/testing in Section 3. Then the following Section 4 provides a discussion of the results.

2. Literature Review

Human pose estimation refers to locating people's body parts in digital images or videos in computer vision. The human pose estimation problem includes the single-person pose estimation, 3D skeleton estimation, multi-person pose estimation, and video pose tracking [13]. MPII [14] and COCO [15] are two publicly available datasets for 2D pose estimation. Traditional methods for pose estimation are vision-based that use two kinds of contexts. The first one is the feature representation like the histogram of oriented gradients (HOG), and the secod one is the spatial context. In contrast, deep learning-based methods are mostly used because of it's rapid development capability. Jain et al. [16] used CNN to estimate the single-person skeleton. Convolutional Pose Machine (CPM) is one of the most important pose estimation methods proposed by Wei et al. [17]. A CPM consists of a sequence of CNNs, which can learn image features and spatial context simultaneously. OpenPose is an open-source real-time 2D pose detection system released by Cao et al. [18], which 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 that 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 performs detection and grouping simultaneously by applying associative embedding. Deepfake creation is the advanced stage of human pose estimation and manipulation. Deepfake creation is a combination of

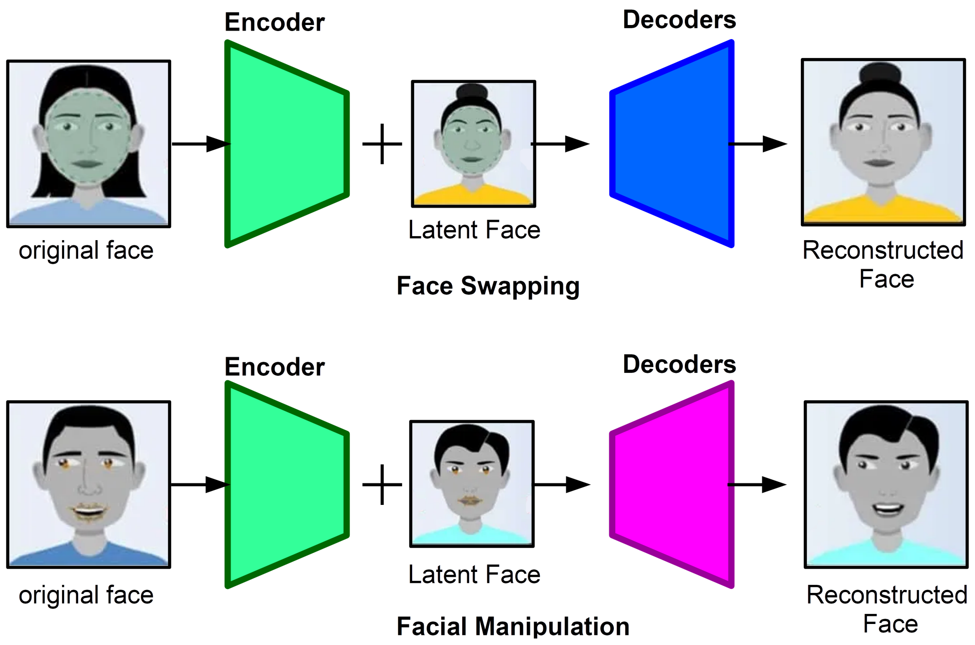

Figure 2. Process of Deepfake Creation: The original face is extracted and processed by the DNN Encoder. The latent face image with dormant features feed to DNN Decoder along with encoder input. The final reconstructed face image is produced from both images (original face and latent face).

computer vision algorithms and deep learning techniques. The unsupervised image-to-image translation framework developed by Liu et al. [20] is the basis of deepfake generation, which is established on the coupled Generative Adversarial Network (GAN). According to

Nguyen et al. [21], 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. 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 by encoding the face using the common encoder and decoding it using the target decoder. The process of creating a fake face is shown in Figure 2. Deepfake converting is one of the steps 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. Since deepfake creation methods are getting sophisticated, the detection and combating deepfakes is getting harder.

Several deepfakes detection methods emerged recently. These methods/algorithms 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 based on a wide range of different technologies. Besides, a deepfake detection tool using Support Vector Machine (SVM) has been developed specifically for world leaders [24]. However, it is a significant problem that these methods tend to be transient because new deepfakes can fix corresponding imperfections in a short time [25]. Thus, it is necessary to expose new imperfections of deepfakes and develop new detection algorithms or improve existing approaches.

Most works on fake video detection use deep learning to detect artifacts and inconsistencies of the video. Li et al. [11] used a CNN and a Long-term Recurrent Convolutional Network (LRCN) to detect unnatural blinking patterns in the video. Because of the lack of images with closed eyes for training model and generating deepfakes, the experimental results indicated that utilizing irregular blinking frequency in videos for deepfake detection performs well. 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 artifact detection method that uses CNNs to detect the resolution inconsistency between facial area and other areas 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 Gaussian blur to the facial area instead of spending a large amount of time implementing a network and training a model for deepfake generation like other approaches did. 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 frames. Their method uses CNN to extract frame-level features and use Long Short-term Memory(LSTM) to extract features in sequences of frames. Nevertheless, Agarwal et al. [24] argued that simple manipulations like additive noise, recompression, and resizing could easily eliminate inconsistencies mentioned in the above two methods.

In addition to deep learning methods, some methods rely on SVMs supervised machine learning models. For instance, Yang et al. [30] 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 experimental results achieved an Area Under the Receiver Operating Characteristic (AUROC) of around 0.9, which shows that deepfake generation can produce head pose errors, but using these features are insufficient. Moreover, Dang et al. [31] evaluated the inconsistent 3D head pose on Celeb-DF (a new deepfake dataset created by improved deepfake algorithm [32]) and only got 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 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 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 improve aggressively, deepfake detection methods are no longer useful or will be ineffective shortly. Additionally, deepfake detection faces 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. That 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 only used head movements instead of body movements. Deepfake techniques only temper with the facial area and manipulate inconsistencies, so detection methods generally concentrate on facial areas and head poses. Therefore, utilizing upper body languages for deepfake detection might be a new research direction.

3. Method

The proposed method is based on the hypothesis that Deep Neural Networks (DNNs) can learn the body language of an individual and expose deepfakes. Given a video of the person of interest as the input, the poses of that person in the video frames should be estimated to generate a sequence of poses. An RNN model needs to train to learn the target person's pose (body language) and spot fake videos. The proposed system designed in the PyTorch [33] deep learning framework. The standard framework of the proposed system demonstrated in Figure. 3.

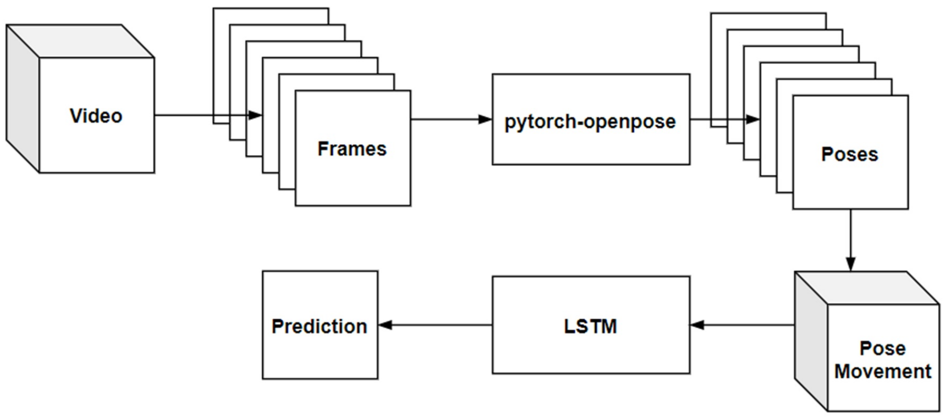

Figure 3. The proposed deepfake detection method: The proposed method will extract the frames (30fps) from the input video. Later OpenPose software used to extract key-point of each pose of the person in the image. The pose movement will be trained using the proposed LSTM architecture. The final inference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

3.1. Dataset and Preprocessing

This proposed idea is to detecting 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 we can train and test the established hypothesis are world leaders videos. These videos are widely available with a good quality pixel ratio, and we can also test these videos against deepfakes very easily. Therefore, we manually downloaded online videos to generate a customized dataset. These videos were annotated and labeled according to the proposed requirements. To test the proposed hypothesis, we need two kinds of videos, the original and the synthetic. For the original videos, to ensure the videos were not tampered with, so these videos downloaded from the official website of Miller Center [34], which has speeches by all presidents of the United States. Initially, we chose videos of the four most recent United states presidents George W. Bush, Barack Obama, Donald Trump and Joe Biden. The downloaded videos satisfy conditions (a) the file format is MP4 and the frame-per-second (fps) is 30, (b) a similar video frames size for the whole dataset, (c) the person of interest is in the middle of the frame, (d) there is no one else in the background, and (e) the camera is relatively stable. The Figure 4 shows dataset samples of real and fake videos for deepfake training.

The deepfake videos of these presidents were also downloaded from youtube. We downloaded deepfakes of all four presidents, where impersonators posed the same body language as our 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 is used to extract frames of videos, an open-source library for image processing and computer vision problems. To extract the upper body key-points in each video frame, we first tried the DNN of OpenCV for loading the deep learning model of OpenPose [35]. OpenPose performs well in human pose estimation; it trained three Caffe models that are BODY_25, COCO, and MPI models. The difference among these models is the output format, the BODY_25 format has 25 keypoints and the COCO format has 18 key-points. The indexes of the same keypoint are also different. We decided to use COCO-style keypoint, as it was more aligned to our proposed dataset and keypoints requirements.

The contributors of OpenPose recommend the BODY_25 model because of its higher accuracy and speed [35], however, we can not use this model due to the unavailability of lower body keypoint. We have used pytorch-openpose, which is a PyTorch implementation of the Caffe model of OpenPose. The output format is the COCO dataset format. An example pose is demonstrated in Figure 5. It shows that pytorch-openpose is more accurate than the original OpenPose. Additionally, it is about five times faster because CUDA supports PyTorch.

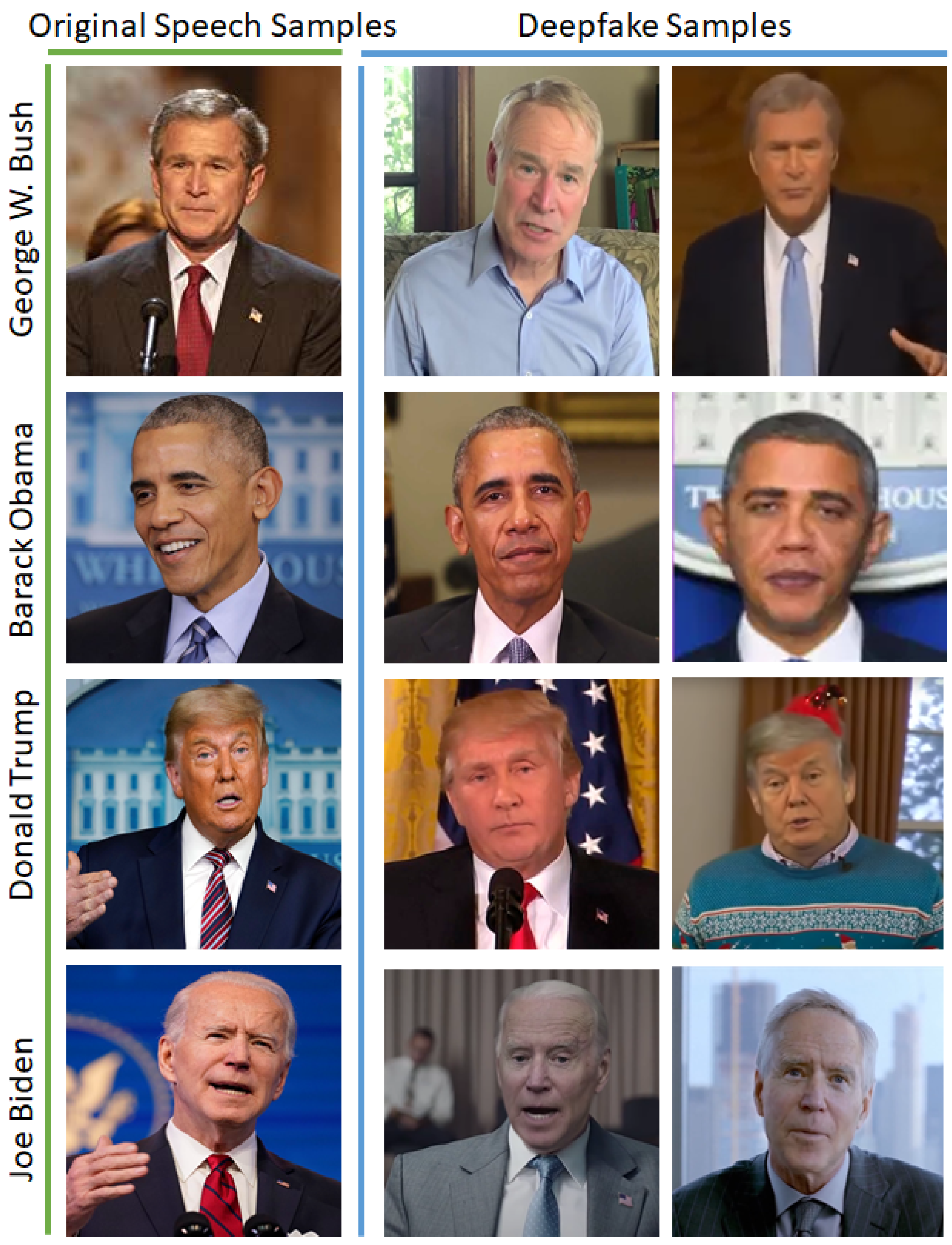

Figure 4. Dataset Sample: The dataset build for the proposed experiments is composed of four recent presidents of the United States. We did not extract videos of earlier presidents due to the unavailability (very few) deepfake samples.

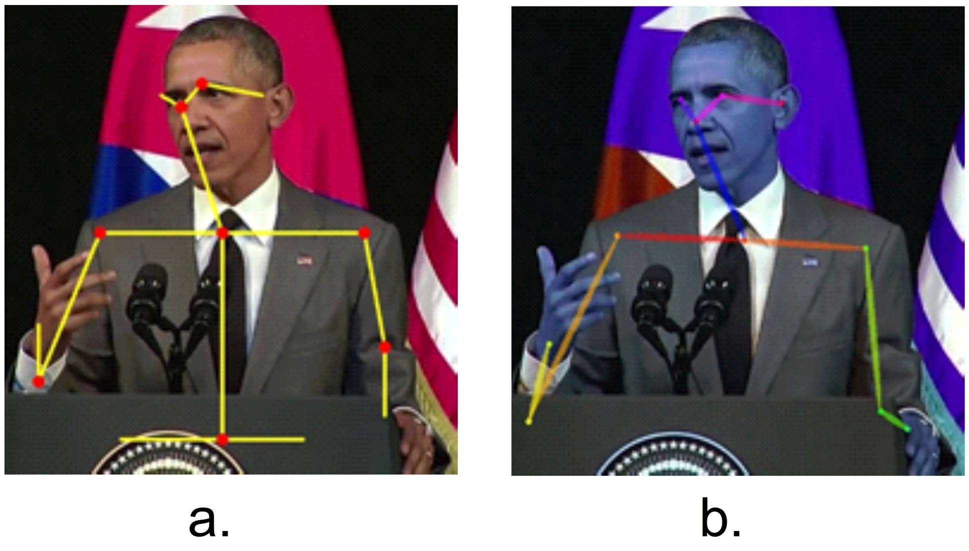

Figure 5. Sample Pose's Keypoints Extraction: a. Using the BODY_25 model of OpenPose and the DNN module of OpenCV. b. Using the PyTorch COCO model of pytorch-openpose.

OpenCV is used for 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 of the 18 key-points of the COCO format can be detected, listed in Table 1. A new output format was generated to satisfy proposed research's requirements, where coordinates represent each key-point. During the upper pose movement, some of the 12 key-points are also undetectable, where we replaced with undetected locations with Zero values. Therefore, the pose of a frame is a 1-D array that contains 24 features. An automated script is written to processes a list of videos frame by frame. It was considered that there should be enough frames to trace a pose movement. Besides, a set of frames must be continuous, that there is no camera view changing. We chose to use every segment of videos (that fulfills the aforementioned requirements) to analyze the upper pose movement. Because pytorch-openpose is for multi-person pose estimation, it can check if multiple people are in a frame to determine whether the camera view is changed. The dataset annotations 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 labeled 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 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 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 sampled from speeches of four recent presidents of the US. We used Miller Center authentic speech videos of US presidents to build the "Real" samples. For Deepfake samples, we crawl YouTube to extract related samples. These videos further segmented into useful samples for training and keypoint extraction 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 independent training, validation, and testing data accounting for 80%, 10%, and 10%, respectively. Then 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, μ and σ 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 μ and σ both are zeros. Therefore, the NaN values generated by Z-score normalisation are replaced with zero values.

3.2. RNN Architecture

To address the problem of detecting fake videos using upper body language analysis, the deep learning model trained to learn the spatial and temporal data and determine whether the input pose movement belongs to the target person's upper body languages. To implement the proposed method, the RNN is the best candidate to capture the dataset's spatial and temporal features. It can be applied to learn the dependencies of sequential data along with graphical features. Besides, LSTMs are advanced RNNs, which can learn long-term dependencies without the problem of gradient vanishing. A structural comparison between RNN and LSTM is shown in Figure 6. It analyzed that 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 at the last time step.

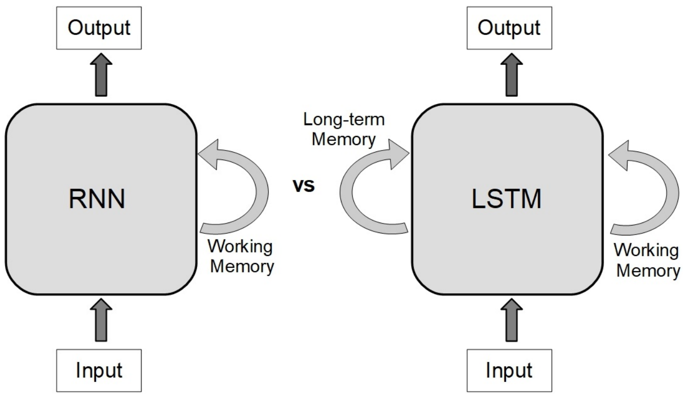

Figure 6. RNN v/s LSTM. a: RNNs use their internal state (memory) to process sequences of inputs, b: Long Short-Term Memory (LSTM) network is a variant of RNN, with additional long term memory to remember past data.

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 are 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 [36]. 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, zero means no information can pass through, and one states that all information can pass through the gate. A normal 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, depending on the current cell state, the output gate determines the output information.

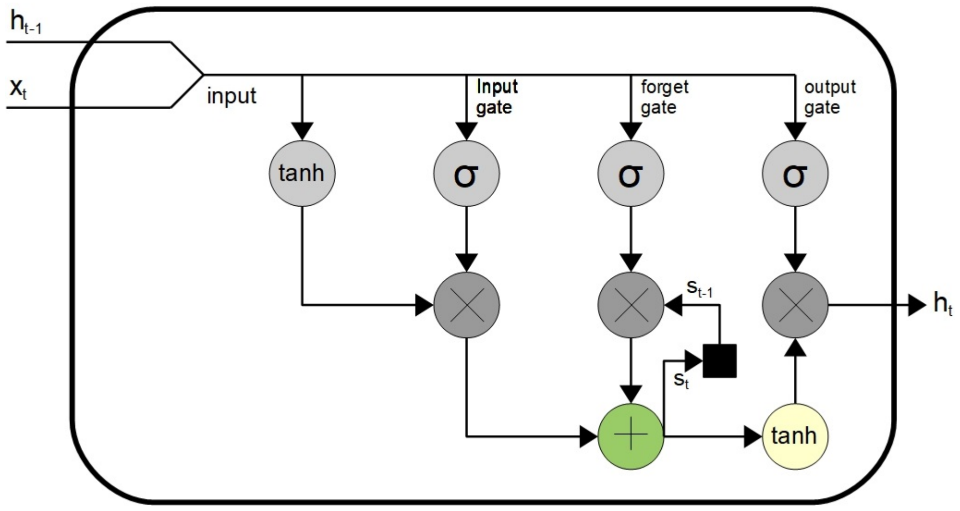

Figure 7. An unrolled LSTM. It takes input sequence x_t and h_{t-1}, process this input through different gates (input, forget, out) and produce the output that could serve as final state or input for next hidden state.

There are many LSTMs variants, but Greff et al. [37] indicate that the performance of various LSTMs are almost the same. The LSTM used in this project was defined using

PyTorch. PyTorch implements the normal LSTM as the memory cell. Each normal memory cell computes the following calculations:

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

where, $*_t$ means the time step; while, t, f_t, i_t, g_t, and o_t are the gates. σ 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 [38]. Therefore, LSTMs can learn short-term dependencies 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 proposed hypothesis that the upper body languages can identify and expose deepfakes, a many-to-one LSTM model designed using PyTorch to learn the pose languages. PyTorch is an open-source framework for machine learning, which provides flexible DNN implementation and GPU support. To set up the training and testing experiment, two kinds of objects are necessary: (a) the object for data loading and (b) the object for model creation. The custom dataset was divided into a 80% training set, a 10% validation set, and a 10% testing set. Moreover, values were normalized to the range of [-1, 1]. A specially designed (according to need of our custom dataset) DataLoader used for passing the datasets to the LSTM model for training. The data only contains the body languages with 24 keypoints, the model to be implemented with a binary classification model having an input layer with 24 features and 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. 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 the optimizer are two significant components for deep learning. We have used binary cross-entropy to be the loss function because this is a binary classification model. Then the cross-entropy measures the difference between the predicted and real distributions. For the optimizer to be applied, PyTorch implements various optimization algorithms to choose from. Optimization algorithms are grouped into adaptive methods like Adam [39], and non-adaptive approaches like stochastic gradient descent (SGD). Adam adapts learning rates for different intrinsic parameters of the network. Though some studies claim that adaptive methods do not perform well in deep learning, though Adam is still popular [40].

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 using the adaptive methods like Adam has adapted learning rates, Loshchilov and Hutter [41] provide evidence of using Adam with the learning rate decay 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 1e-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 sizebatch size)* determines how many times the weights are updated. Assuming 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 stability. It also reduce the generalization performance. Using a small batch size can avoid local optima, but the model may be hard to converge. Keskar et al. [42] 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 that causes NaN losses during the training. Applying gradient clipping is the 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 to use GPU and CUDA for computation and send the model to a GPU. Since the output layer outputs a score, the final prediction can be obtained by applying a Sigmoid function to the score and then rounding off the predicted probability. At each epoch during the training and validation experiment, the model weights were optimized by training using the training set. Then 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. Besides, gradient clipping was applied. Other hyper-parameters such as the hidden size was 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 useful LSTM model for deepfake detection.

3.4. Evaluation Metrics

Two metrics were used for the performance evaluation of the model, which are loss and accuracy. The loss function applied is the binary cross-entropy. The smaller the loss, the more similar the predicted distribution is to the target distribution. We plotted the loss and accuracy over time to check whether the model was underfitted or overfitted. Underfitting means the model fails to converge, which cannot simulate the real distribution. Overfitting refers to a model having poor generalization performance, that means the model learns the information of the training data but cannot generalize well to new instances. For example, underfitting happens when the loss does not decrease, or the accuracy does not increase. If the training loss is reducing or the training accuracy increases, while the validation loss is increasing or the validation accuracy is reducing, the overfitting problem occurs. However, sometimes having high loss can also lead to high accuracy. For instance, assuming the input label is 1, the predicted probabilities of two different models are 0.6 and 0.9, which means the second model has higher confidence. Rounding off two predictions gives one which equals the target value. The first model's loss is greater than that of the second model, but both models' accuracy is the same. This circumstance may due to inadequate training data. The generated dataset used for this project is small; high confidence may lead to overfitting. Therefore, we adjusted hyper-parameters according to both the losses and accuracy while choosing the model with the highest validation accuracy to be the best model. The accuracy also evaluated the performance of the final model.

Table 3: **Network Architecture and hyper-parameters:** Several experiments 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		

4. Results

The following chapter presents the results of model training and testing. After training a final version of LSTM model obtained bt the hyper-parameter adjustment. The proposed deepfake detection method is tested through an inference procedure.

The first problem is to determine the optimizer. Two optimizers, including the SGD and Adam, were tested. We first tried the SGD, and the learning rate was 1e-3. The results are shown in the subfigures (a) and (b) of Figure 8. Over time, the training and validation losses shown that the learning rate was too high so that the model failed to converge. Then we reduced the learning rate to 1e-4, and the results are presented in the subfigures (c) and (d) of Figure 8. The lower learning rate made the loss decrease slowly, but the model with an accuracy of about 70% was still underfitted. The trend towards loss change means that the new learning rate was too low. Then we used Adam with a learning rate of 1e-3. The subfigures (e) and (f) of Figure 8 present the results. 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, and Adam is more suitable for this model. Thus, we decided to use Adam to be the final optimizer.

The next step was to fix the overfitting problem. Dropout is an effective method to address this problem. The dropout of an LSTM having a single hidden layer is zero. Therefore, we changed the number of hidden layers to 2 and set the dropout probability as 0.5. The modified model was a stacked LSTM model. The results are shown in Figure 9. The overfitting problem still existed, but the validation accuracy was greater 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. To solve overfitting and to use appropriate learning rates, we applied the learning rate decay that the initial learning rate is 1e-2 and 0.5 times every 30 epochs decreased it. Besides, the weight decay with a λ coefficient of 1e-4 was used to prevent overfitting. The results 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 the 165^{th} epoch.

Table 3 provides the final hyper-parameters. The model was then tested using the testing data. The training, validation, and testing accuracy of the model are 99.06%, 96.23% and 94.39%, respectively, listed in Table .5. The model trained using the final hyper-parameters and a single TitanX GPU for computation.

To assess the most appropriate LSTM arrangement for experiments, we trained LSTMs with different neuron depths (hidden states). 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, it realized that LSTMs with 512 neuron depth offered us 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

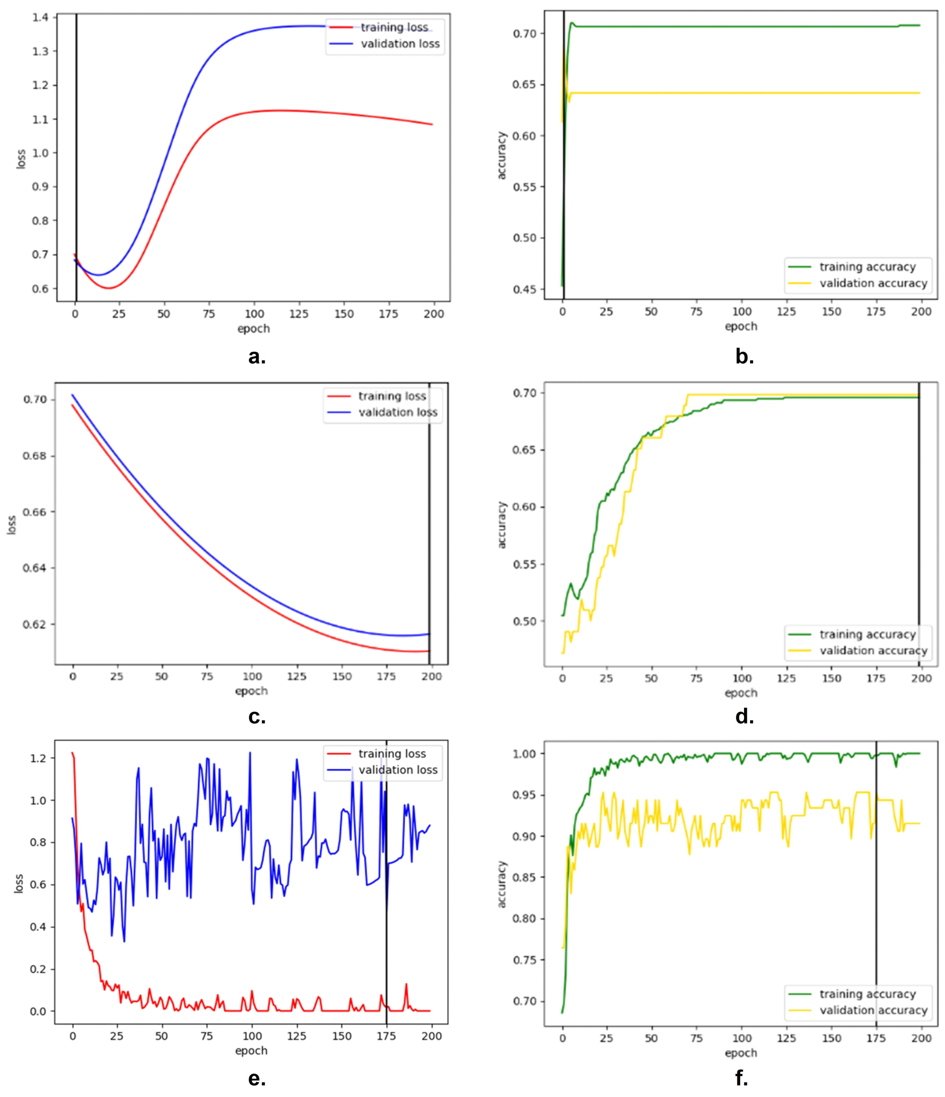

Figure 8. Loss and accuracy of using two different optimizers. (a) and (b) are results of using SGD and lr=1e-3, (c) and (d) are results of using SGD and lr=1e-4, (e) and (f) are results of using Adam and lr=1e-3, where lr is the learning rate. The black vertical line in each subfigure is the epoch having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

trained with the given dataset through a single end-to-end trainable design for 200 epochs (500 for initial experiments).

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

5. Discussion

The experiments described earlier successfully trained a practical model with an accuracy of 94.39%. The results showed that pose languages could be used to distinguish different people and expose deepfakes using the model of pytorch-openpose for pose estimation and the LSTM model for deepfake detection. Given a target person's video assessed using a trained detection model, the pytorch-openpose first estimates the upper body poses and outputs the upper body pose movement. The LSTM model can then recognize whether the pose movement belongs to that target person's upper body language.

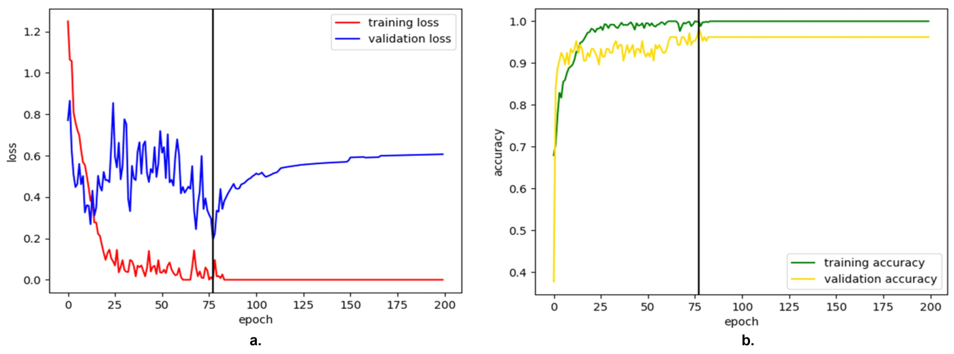

Figure 9. Results of using 2 hidden layers with a dropout probability of 0.5.

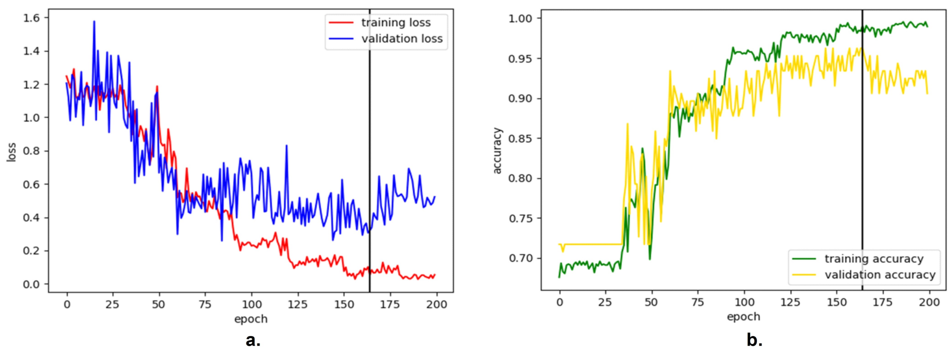

Figure 10. Results of applying learning rate decay and weight decay.

For example, the aforementioned experiments learned the pose languages of the target person and trained a model. Videos always have no less than 150 frames, which means there is (always) more than one pose movement. Therefore, the demo extracts all pose movements to be the input to the LSTM model. The video marked as real, only if the number of real pose movements is greater than that of fake pose movements.

This research has successfully addressed the problem of fighting deepfakes for world leaders. The testing results are presented in Table 5. The proposed method works well in terms of detection of deepfake though 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 drop in training accuracy and the validation accuracy more significant than the testing accuracy. The fake instances were extracted from deepfake videos on the internet (youtube), which may not be using the latest deepfake creation techniques. Thus, it is unknown whether this method can to discern deepfakes produced by advanced creation methods. Another limitation is that the number of input sequences is fixed. An RNN model that allows input to vary in sequence length 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. Conclusion and Future work

Misusing of deepfake techniques have a significant problem. In particular, deepfake videos of world leaders are a threat to the public. This research created a novel deepfake detection method that can spot fake videos for world leaders by analyzing the upper body languages to address the problem. To prove the hypothesis that "the upper body languages are distinct for different individuals and can be used to expose deepfake videos by RNN," a dataset was first generated using pytorch-openpose. The data pre-processing method extracts 12 body keypoints, including 24 coordinate values representing an upper body

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	2 Hidden Layer	3 Hidden Layer
512 Hidden Units	256 Hidden Units	128 Hidden Units
94.39%	**91.24%**	**86.62%**

Table 5: **Training, Validation and Testing accuracies of the final model:** The model trained and tested against several methods. We have trained the proposed dataset with the VGG16 [43], ResNet [28] and multimodel CNN+LSTM architectures. Though the proposed LSTM with given hyper-parameters are setting performed most effective invalidation and test accuracy.

Model	**Training accuracy**	**Validation accuracy**	**Testing accuracy**
VGG16 [43]	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%

pose and 150 pose movement. Then a many-to-one LSTM model was defined and trained to analyze the upper body language. During the experiments, the model's validated the proposed hypothesis effectively. With the fast development of deepfake creation techniques and the in-depth research of deepfake detection, deepfakes are becoming more and more realistic. It is getting more and more challenging to spot these imperfections. This research has created a new research direction that is not limited to using facial and body language. Over time, the deepfake creation methods are getting smarter, as 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. Several aspects can be employed to improve and enhance the proposed research to detect deepfakes. We aim to use and employ the following methods to improve the underlying research for our future work.

Collect more real videos to produce positive instances. Instead of downloading deepfakes from the internet, the development of synthetic instances of videos using advanced deepfake creation approaches. In particular, the target person's swap faces, and the impersonator may have similar pose languages to the target person. Moreover, continue to train the new models using transfer learning. Training using a larger dataset might enhance the generalization performance.

Create datasets for other world leaders and train models for different world leaders to test whether the proposed method is still useful. Define 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. In addition, using a upper-body pose, use a hand pose to train the model. The pytorch-openpose also provides a model for hand pose estimation. Adding a hand pose is similar to upper body languages that different people might have distinct hand gestures.

7. References

1. Goodfellow, I.; Bengio, Y.; Courville, A.; Bengio, Y. *Deep learning*; Vol. 1, MIT press Cambridge, 2016.
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. *Information Sciences* **2019**, *479*, 448–455.
3. Shen, D.; Wu, G.; Suk, H.I. Deep learning in medical image analysis. *Annual review of biomedical engineering* **2017**, *19*, 221–248.
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.
5. Yasrab, R.; Zhang, J.; Smyth, P.; Pound, M.P. Predicting Plant Growth from Time-Series Data Using Deep Learning. *Remote Sensing* **2021**, *13*, 331. doi:10.3390/rs13030331.
6. Chesney, B.; Citron, D. Deep fakes: a looming challenge for privacy, democracy, and national security. *Calif. L. Rev.* **2019**, *107*, 1753.
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.
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. Proceedings of the 28th ACM International Conference on Multimedia, 2020, pp. 4318–4327.
9. Hern, A. I don't want to upset people': Tom Cruise deepfake creator speaks out. *The Guardian* **2021**.
10. Dolhansky, B.; Howes, R.; Pflaum, B.; Baram, N.; Ferrer, C.C. The deepfake detection challenge (dfdc) preview dataset. *arXiv preprint arXiv:1910.08854* **2019**.
11. Li, Y.; Chang, M.C.; Lyu, S. In ictu oculi: Exposing ai created fake videos by detecting eye blinking. 2018 IEEE International Workshop on Information Forensics and Security (WIFS). IEEE, 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 preprint arXiv:2001.00179* **2020**.
13. Chen, Y.; Tian, Y.; He, M. Monocular human pose estimation: A survey of deep learning-based methods. *Computer Vision and Image Understanding* **2020**, *192*, 102897.
14. Andriluka, M.; Pishchulin, L.; Gehler, P.; Schiele, B. 2d human pose estimation: New benchmark and state of the art analysis. Proceedings of the IEEE Conference on computer Vision and Pattern Recognition, 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. European conference on computer vision. Springer, 2014, pp. 740–755.
16. Jain, A.; Tompson, J.; Andriluka, M.; Taylor, G.W.; Bregler, C. Learning human pose estimation features with convolutional networks. *arXiv preprint arXiv:1312.7302* **2013**.
17. Wei, S.E.; Ramakrishna, V.; Kanade, T.; Sheikh, Y. Convolutional pose machines. Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2016, pp. 4724–4732.
18. Cao, Z.; Hidalgo, G.; Simon, T.; Wei, S.E.; Sheikh, Y. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. *arXiv preprint arXiv:1812.08008* **2018**.
19. Newell, A.; Huang, Z.; Deng, J. Associative embedding: End-to-end learning for joint detection and grouping. Advances in neural information processing systems, 2017, pp. 2277–2287.
20. Liu, M.Y.; Breuel, T.; Kautz, J. Unsupervised image-to-image translation networks. Advances in neural information processing systems, 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 preprint arXiv:1909.11573* **2019**, *1*.
22. Li, Y.; Lyu, S. Exposing deepfake videos by detecting face warping artifacts. *arXiv preprint arXiv:1811.00656* **2018**.
23. Mirsky, Y.; Lee, W. The creation and detection of deepfakes: A survey. *ACM Computing Surveys (CSUR)* **2021**, *54*, 1–41.
24. Agarwal, S.; Farid, H.; Gu, Y.; He, M.; Nagano, K.; Li, H. Protecting World Leaders Against Deep Fakes. CVPR Workshops, 2019, pp. 38–45.
25. Vincent, J. Deepfake detection algorithms will never be enough. *The Verge* **2019**, *27*.

26. Korshunov, P.; Marcel, S. Deepfakes: a new threat to face recognition? assessment and detection. *arXiv preprint arXiv:1812.08685* **2018**.
27. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 3431–3440.
28. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
29. Güera, D.; Delp, E.J. Deepfake video detection using recurrent neural networks. 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). IEEE, 2018, pp. 1–6.
30. Yang, X.; Li, Y.; Lyu, S. Exposing deep fakes using inconsistent head poses. ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019, pp. 8261–8265.
31. Dang, H.; Liu, F.; Stehouwer, J.; Liu, X.; Jain, A.K. On the detection of digital face manipulation. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 5781–5790.
32. Li, Y.; Yang, X.; Sun, P.; Qi, H.; Lyu, S. Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 3207–3216.
33. Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic differentiation in pytorch **2017**.
34. Center, M. Miller Center Foundation website.
35. Cao, Z.; Hidalgo, G.; Simon, T.; Wei, S.E.; Sheikh, Y. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. *IEEE transactions on pattern analysis and machine intelligence* **2019**, *43*, 172–186.
36. Lipton, Z.C.; Berkowitz, J.; Elkan, C. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019* **2015**.
37. Greff, K.; Srivastava, R.K.; Koutník, J.; Steunebrink, B.R.; Schmidhuber, J. LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems* **2016**, *28*, 2222–2232.
38. Jozefowicz, R.; Zaremba, W.; Sutskever, I. An empirical exploration of recurrent network architectures. International conference on machine learning, 2015, pp. 2342–2350.
39. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* **2014**.
40. Wilson, A.C.; Roelofs, R.; Stern, M.; Srebro, N.; Recht, B. The marginal value of adaptive gradient methods in machine learning. *arXiv preprint arXiv:1705.08292* **2017**.
41. Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* **2017**.
42. 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 preprint arXiv:1609.04836* **2016**.
43. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* **2014**.