Received: 8 June 2022 | Revised: 26 July 2022 | Accepted: 31 July 2022

DOI: 10.1049/bme2.12093

IET Biometrics

The Institution of Engineering and Technology WILEY

ORIGINAL RESEARCH

Protection of gait data set for preserving its privacy in deep learning pipeline

Anubha Parashar | **Rajveer Singh Shekhawat**

Computer Science and Engineering, Manipal University Jaipur, Jaipur, India

Correspondence

Anubha Parashar, Computer Science and Engineering, Manipal University Jaipur, Jaipur, India.
Email: anubhaparashar1025@gmail.com

Abstract

Human gait is a biometric that is being used in security systems because it is unique for each individual and helps recognise one from a distance without any intervention. To develop such a system, one needs a comprehensive data set specific to the application. If this data set somehow falls in the hands of rogue elements, they can easily access the secured system developed based on the data set. Thus, the protection of the gait data set becomes essential. It has been learnt that systems using deep learning are easily prone to hacking. Hence, maintaining the privacy of gait data sets in the deep learning pipeline becomes more difficult due to adversarial attacks or unauthorised access to the data set. One of the popular techniques for stopping access to the data set is using anonymisation. A reversible gait anonymisation pipeline that modifies gait geometry by morphing the images, that is, texture modifications, is proposed. Such modified data prevent hackers from making use of the data set for adversarial attacks. Nine layers were proposedto effect geometrical modifications, and a fixed gait texture template is used for morphing. Both these modify the gait data set so that any authentic person cannot be identified while maintaining the naturalness of the gait. The proposed method is evaluated using the similarity index as well as the recognition rate. The impact of various geometrical and texture modifications on silhouettes have been investigated to identify the modifications. The crowdsourcing and machine learning experiments were performed on the silhouette for this purpose. The obtained results in both types of experiments showed that texture modification has a stronger impact on the level of privacy protection than geometry shape modifications. In these experiments, the similarity index achieved is above 99%. These findings open new research directions regarding the adversarial attacks and privacy protection related to gait recognition data sets.

KEYWORDS

de-identification, gait anonymization, gait biometric, privacy, reversible deep learning pipeline

1 | INTRODUCTION

One of the main biometric identities used for human identification is gait. Gait-based identification is used in various application scenarios, such as surveillance systems of sensitive public and semi-public places. One needs a comprehensive data set specific to the application to develop such systems. If this data set somehow falls in the hands of rogue elements, they can easily access the secured system developed based on the data set. Thus, the protection of the gait data set becomes essential. It has been learnt that systems using deep learning are easily prone to hacking. Hence, maintaining the privacy of the gait data set in the deep learning pipeline becomes more difficult due to adversarial attacks or unauthorised access to the data set. One of the popular techniques for stopping access to the data set is using anonymisation. For privacy protection, there is a need for gait anonymisation [1], which is performed by a gait pipeline that may consist of gait detection, localisation, gait feature

points localisation, and gait anonymisation by masking, gait swapping, wrapping, morphing, scaling, rotation, translation or by scrambling [2]. Gait detection is a prerequisite technique for efficient gait anonymisation. To achieve the naturalness of de-identified gaits (silhouette-based features), the common approach is to detect the silhouette's feature points, which would be the starting points for de-identification based on morphing, geometrical gait changes, and/or their combination. In general, gait de-identification methods can be classified as model-free and model-based.

The model-free methods are very fast are simple to implement. One requires only an approximate bounding box of the object. The utility and naturalness of de-identified data are high. The methods are reversible, and privacy protection is high [3]. False-positive gait detections do not degrade the naturalness of the de-identified data because the original texture is not replaced with the modified one. There is a slight modification done to de-identified data. The model-based methods are slow, and implementation complexity is high. Further, one requires that many gait components (pose features, i.e. hand angle, wrist angle, pelvis, and ankle points) are precisely localised and have many degrees of freedom. But their utility is very high, being hardly irreversible (due to the wrong detection of pose feature points every time). The privacy obtained by such methods cannot be easily compromised. False-positive gait detections also degrade naturalness because a texture is modified [4], and the geometry of gait components is changed [5].

Computer vision has dumbed it down to capture and share large amounts of video data with a vast percentage of respondents. Today, many cameras monitor public and semi-public areas. These raise concerns about the unintentional and unwarranted invasion of privacy of those caught in the videos. We are using a data set of silhouettes and removingconcealing personal identifiers by scaling the silhouettes or merging the silhouette with the average silhouettes of all the subjects in the data set. Gait modification of gait geometry and morphing enables both anonymisation and preserving the naturalness of the gait derived from the silhouette. One of the main goals of this preliminary research is to demonstrate the impact of gait geometry and/or morphing modifications on the level of privacy protection when humans and/or machines perform recognition. The results of both types of experiments are given in Section 4.

Surveillance cameras are all over public and semi-public spaces today, thus encroaching on our privacy. There is every chance that rogue elements may use this information. It is assumed that the agencies collecting this information are reliable, but there is every chance that the data stored may be hacked and misused. To ensure data privacy, we developed:

1. An anonymisation (or de-identification) model that encrypts the data to protect the identity of a person without losing the natural silhouettes.
2. A reversible gait deep learning pipeline that re-generates the original data sets from the anonymised one for verification.
3. A set of psychological experiments that evaluate the impact of morphing and/or gait geometry modifications on the level of privacy protection and
4. A comparison of obtained results (anonymisation and reverse deep learning pipeline) for both humans and robots for automatic gait recognition.

To achieve images of the silhouette that are de-identified as much as possible, we can contribute something in the way of achieving 100% de-identification without losing the naturalness of the silhouette.

Following the introduction, the remaining parts of the paper are organised as follows: In Section 2, we will discuss the work that is related. In Section 3, an explanation of our methodology is specified. An anonymised pipeline along with the algorithms is given. The results of the experiments are presented in Section 4. Section 5 summarises our findings and draws a conclusion about this paper. Protection of Gait Data set for Preserving its Privacy in Deep Learning Pipeline.

2 | LITERATURE REVIEW

2.1 | Anonymisation and adversarial methods

Gait is a biometric technique that may be used to identify a person, and a human subject can be utilised to impersonate the identity of another subject to fool the system. In Ref. [6], the authors change gait silhouettes so that a person's gait is identifiable, but the naturalness of the gait stays the same. The approach of gait anonymisation they offer may prevent illegal gait identification. When high-quality silhouette gaits are employed, the approach devised achieves anonymisation while maintaining gait naturalness. When low-quality ones are utilised, though, the results seem strange. Imposters used early approaches to fool the gait recognition system by imitating walking and clothing patterns. This study employs spoofing attacks by substituting the current data set with manufactured image samples. The authors of Ref. [4] developed a technique for creating images of a topic based on a Generative Adversarial Network. Their strategy successfully tricked the GaitSet network. The similarity score of samples in the gallery and the attacking video were calculated. If the similarity score is more than the threshold, the assault is judged successful; the attack is not considered if the score is less than the threshold. As a result, it is dependent on the technique used to generate the similarity score.

Traditional cameras were unable to catch variations in intensity, whereas event-based cameras could do so. The visuals may be reconstructed using event cameras. Although several approaches have been developed to reconstruct intensity images from event streams, the resultant images still include unrealistic, noisy, and low-resolution information. The authors of Ref. [3] suggested an unsupervised upsampling adversarial learning reconstruction approach to avoid unrealistic, noisy, and low-resolution information. The image resolution determines a model's accuracy. The accuracy suffers when the images utilised for feature extraction are of poor quality. As a result, they rebuilt the images to improve image quality for

higher model accuracy, but they did not address the implications of event stack types on EventSR performance. Although gait detection systems are more accurate than other biometrics, there is still a need to investigate the resilience of adversarial systems. The authors of Ref. [7] presented a sparse adversarial temporal-based assault. In their model, attacks were successful in a significant percentage of cases. They have a limited capacity of the trained generator; therefore, some of the created adversarial samples are out of the typical gait silhouette distribution.

Gait recognition relies on temporal characteristics to recognise walking patterns; however, temporal features have a frame rate restriction, lowering performance. The authors of Ref. [4] developed a generative adversarial network (GAN) to capture the subtleties in time-serial data lost when the adjoining frame rate is captured. GAN generates 385 frames that may be used to create decent templates [8]. The suggested loss function, margin ratio loss, increases the performance of the gait classifier. They didn't test their method under covariate variables, such as clothing or carrying situations. They only worked with a tiny collection of data. To acquire the generalised findings, they must have assessed huge data sets like the OU-ISIR MVLP.

2.2 | Deep learning-based model-free gait methods

Deep learning models may identify a wide variety of features from low-level feature sets and create numerous levels of feature hierarchies. A deep learning model's layers are generic and automated with adjustable parameters. Several recent experiments on gait identification utilising deep learning algorithms have shown promising results. Convolution Neural Network (CNN) is a well-known deep learning architecture. CNN effectively extracts characteristics from images, but it takes a long time to train since it requires a huge data set and a lot of storage space. CNN's deep hierarchical structure generally has billions of parameters, necessitating additional time and processing resources to complete. In gait recognition systems, for real-time security and surveillance applications, it's critical to minimise the time it takes for a deep CNN model to analyse big data sets.

Rauf et al. [9] presented a novel architecture to speed up the process and reduce models based on CNN and subsequent gait recognition tests. They presented a completely linked network that is smaller and faster than the conventional CNN. However, since the procedure is more sophisticated and time-consuming, their suggested method has a high computational cost. McLaughlin et al. [10] proposed a CNN network with recurrent networks in the final layer that uses a convolutional neural network to collect spatial and temporal data. Their technique extracts the frame from gait sequence data easily and precisely; however, the computational cost of their suggested model is quite expensive. Sokolova et al. [11] suggested an optical flow image-based deep learning approach. Their suggested method effectively retrieves spatial information, and they employed the whole body profile rather than individual joints from various body segments. On the other hand, the authors' technique fails to capture temporal information, resulting in a loss of temporal aspects. They developed a multi-pipeline convolution neural network using optical flow images as an input. In Ref. [12], their CNN model effectively captures geographical characteristics. The downside of utilising this approach is that it does not account for temporal characteristics. Gait identification with low-resolution cameras is difficult, according to the work in Ref. [13]. They suggested a deep CNN network that extracts features, inputs Gait Energy Image (GEI), and is trained by a neural network. Their CNN model was good at capturing geographical variables but not so good at capturing temporal ones. Carley et al. [14] suggested an autocorrelation approach for learning characteristics from different perspectives. Their suggested architecture is both ideal for extracting posture characteristics and computationally inexpensive. Their system, however, has a poor level of accuracy.

Various processing stages, such as subject segmentation, feature learning, feature extraction, subject similarity check, and classification, must be included to recognise gait. Chunfeng et al. [15] introduced a CNN-based GaitNet that conducts subject segmentation, feature learning, feature extraction, subject similarity check, and classification. From raw gait images, their approach may extract different information. Their approach fails to capture temporal information, necessitating a fusion model to improve accuracy. Habiba et al. [16] suggested a fuzzy entropy-controlled skewness network and a fusion-based pre-trained Deep Neural Network (DNN) (VGG19 and AlexNet). Their model effectively captures spatial information. Although combining two suggested networks improves accuracy, their fusion model fails to extract temporal characteristics.

Zhaopeng et al. [17] presented a network-based capsule network to distinguish gait. Their network collects spatial data (like CNN does) but also preserves changes in characteristics such as clothing, perspective variance, and multi-walking scenarios. Because capsule networks can contain all forms of information created during pooling (overfitting issues), they are thought to be better and more efficient than basic CNN. When GEI is used as an input, the capsule network does not provide excellent accuracy. Makihara et al. [18] suggested a deformable spatial and temporal CNN. Their suggested technique performs well on numerous data sets and has high generalisation capabilities. Their strategy is less successful on smaller data sets, real-time data, and varied covariate circumstances. They did not, however, evaluate their approach under various covariate situations. Sun et al. [19] presented a quicker Recurrent Convolution Neural Network for extracting characteristics from a subject's clothes and locating persons in shadow images. The suggested architecture is scalable and can handle changes in covariates without losing accuracy. Because the areas formed outside the CNN employ a selective search technique, their suggested solution is sophisticated and time-consuming.

2.3 | Model-free GANs methods for gait recognition

Due to occlusion, the whole gait cycle is often absent, making gait detection problematic. M. Babaee et al. [20] suggested a technique for converting an incomplete gait cycle into a full gait cycle by producing an incomplete GEI and then reconstructing a full GEI using deep autoencoders. The results were improved by reconstructing the frames and producing partial GEI, but they did not consider clothing or other variables. Complete GEIs were rebuilt from a few frames of the gait cycle by the the authors in Ref. [21]. A DNN technique was developed to create a whole gait cycle for this. According to their research, the reconstruction of silhouettes does not function if the partial gait cycle is less than 20%. The authors suggested a partial gait cycle utilising Fully Connected Neural Network to detect people when an incomplete frame is available [22]. Because they are taking incomplete gait cycles and, in many cases, even conducting GEI with a single frame, the accuracy is substantially reduced. Li-min et al. [23] suggested a VGG-16, Long short-term memory, and GAN to detect occlusion and forecast frames to reconstruct a full gait cycle. They employed a deep learning algorithm to recreate the missing frames and improve the identification accuracy. They do not gather temporal characteristics in their method. Maryam et al. [24] suggested a nonstandard periodic GEI that they built without prior knowledge of the gait cycle. As a result, gait cycles are no longer required to generate GEI. Because no preprocessing is necessary, this approach is more efficient and easy.

GAN was utilised by Li et al. [25] to restore the GEI technique. When the generator is trained using the fusion of mean square error and adversarial loss, the outcome improves, but the computing cost of their technique is high. The tiny region of silhouette owing to occlusion significantly impacts recognition. Dhritimaan et al. [26] suggested a GAN-based occluded silhouette generation approach. The Wasserstein GAN is used to build sparse features to enhance gait-based categorisation. The authors did not compare the various percentages of the partial gait cycle in terms of the reconstruction error. Variations in current gait identification systems, such as carrying conditions, alter GEI characteristics. The authors of Ref. [27] developed a GAN-based solution for making transported items vanish to prevent occlusion produced by these things. Their approach is resistant to fluctuations in carrying capacity. A big data set is needed to train a 3D gait recognition algorithm; however, their network is basic (regression-based network–GEINet) and accurate for smaller data, making it unsuitable for real-time applications.

A multitask GAN was suggested by Uddin et al. [28] for feature reconstruction. A new approach for imaging period energy is presented here to preserve temporal characteristics. They used GANs in conjunction with multitask learning to attain competitive outcomes. This procedure is useful for reducing mistakes brought on by shifts in perspective caused by gait, which occur often. Because of the great dimensionality of GANs, it is not feasible to construct a period energy image by achieving stable training. It's common for the subject's body parts to be obscured by pillars, other people walking, trees, automobiles, or beams in the frame.

3 | METHODOLOGY

The proposed gait anonymisation pipeline has two stages: gait preprocessing and reversible anonymised pipeline, that is, gait identification and de-identification pipeline. In gait preprocessing, we do gait detection [29], silhouette feature point localisation [30], and gait region decomposition [31]. Gait present in videos or still images is detected at the first stage. Then, silhouette feature points are obtained and decomposed with pixel parsing [32]. A reversible anonymised gait image is performed based on averaging the texture modification and scaling the gait geometry (Figure 1).

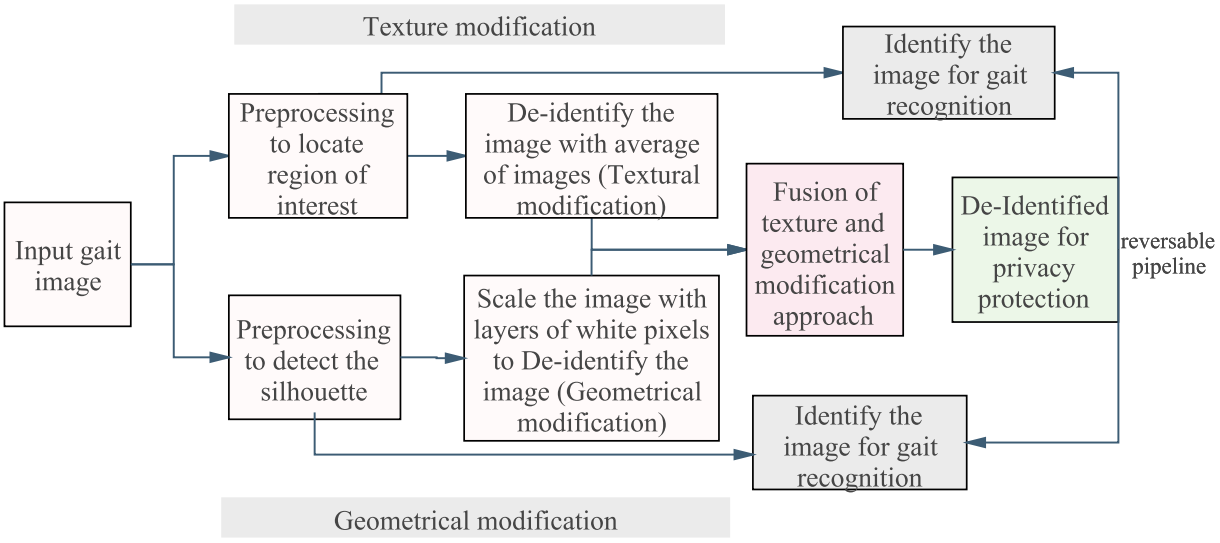


FIGURE 1 Methodology adopted to anonymize the gait silhouette by two techniques (textural modifications and geometrical modifications)

3.1 | Data set

The CASIA-B data set [33] is used, which contains images of 124 subjects walking at an angle of 0–180. The composition of the CASIA B data set is *"xxx-bg-01-yyy-zzz"*

- Xxx stands for the subject number here to achieve all the subjects from 000 to 124.
- Yyy stands for the angle considered while traversing the whole database.
- Zzz stands for the sequence or number of images to get the average out of them. Here, we can see we maintain the same sequence number and same angle but only change the subject number. There are different walking/views of the subject, which the 'bg-01' part can identify.

Different covariates that are present in the CASIA data set B are

- Nm:- Subject walking normally
- Bg:- Subject walking with a bag
- Cl:- Subject walking while wearing a coat

We are using the power of deep learning to build a well-defined model that would help us test the anonymised data set of silhouettes. The model would average all the silhouettes and place them on the input image to anonymise the data set. After this, scaling of silhouettes is done for geometric modifications. Then the anonymised data set will be compared with the original data set. Through experiments, we will find out which modifications anonymise the data better, geometric modifications or textural modifications. Then, we will try to fuse the two modifications to see the performance of the anonymised model.

3.2 | Gait de-identification pipeline

The entire gait silhouette is subject to modification, including pose features, wrist angle, pelvis, and ankle. For each gait silhouette, a set of modifications is defined; considering the morphological characteristics of a human gait, it is very difficult to define what gait components constitute the core concept that defines gait identity. Gait identity is subjective by nature for humans. Only certain combinations of gait component modifications can be performed if we want to preserve the naturalness of de-identified gaits. To preserve the naturalness of a gait, the first step is to determine the global morphological characteristics of a gait distance between features, head width and height, and toe tip position. Based on these global morphological characteristics, we introduced a range of allowable gait modification parameters expressed in the interval from −1 to 1; each gait modification parameter is (pseudo) random, where 0 defines original visual appearance and −1 and 1 are the maximum allowed change in opposite directions. The value selected from the interval was [−1,1]. For example, the hand size modification parameter value from the interval (−1,1) is linearly mapped into the value modified size from the hand size interval [min_hand_size, max_hand_size], which is calculated as follows:

$$[min\_size, max\_size] = \frac{distance\_between\_body}{K1} + \frac{head\_width}{K2} \quad (1)$$

Where the distance between hands and head width is determined for each gait subject of de-identification, and the parameters K1 and K2 are obtained based on the near frontal gait database analyses.

3.3 | Algorithm—encryption and decryption

To encrypt images to protect the visual identity of the subjects, we followed two methods. The two methods are (1) Average of all the sequences and (2) Encrypting using scaling methodology. Figure 2 depicts how these methods/algorithms will help us achieve encryption of images.

3.3.1 | Average—encryption

This is the first phase to de-identify the images. Algorithm 1 describes the method for encrypting the images and explaining its steps.

1. Traverse the data set so that we will be able to get all paths of all the images with the same sequence image and same walking style.
2. These paths are then added to a temporary list, which will contain a list of paths, what degree of the image we are using, and what sequence of images we are using. [degree, sequenceNumber, Path1, Path2, Path3]
3. After receiving the list in the function, first get the dimension of the image, which will help create a resultant array that will store the average image.
4. Open the image and convert the image into RGB (red, green, blue) format with dtype of float.
5. After the image is converted into RGB format, convert it into a NumPy array to make addition easier and less time-consuming.
6. Continue this process for all the available paths in the list; after executing the whole list, divide the array by the total number of paths available in the list.
7. After dividing, get the array containing the average image to the main/driver function.
8. After receiving the average image in an array form, the driver function converts and saves the image as a temporary image and runs a function that will help us superimpose the average image.
9. This function will again traverse the data set, read all the sequences of images, and get the average image from the main function.

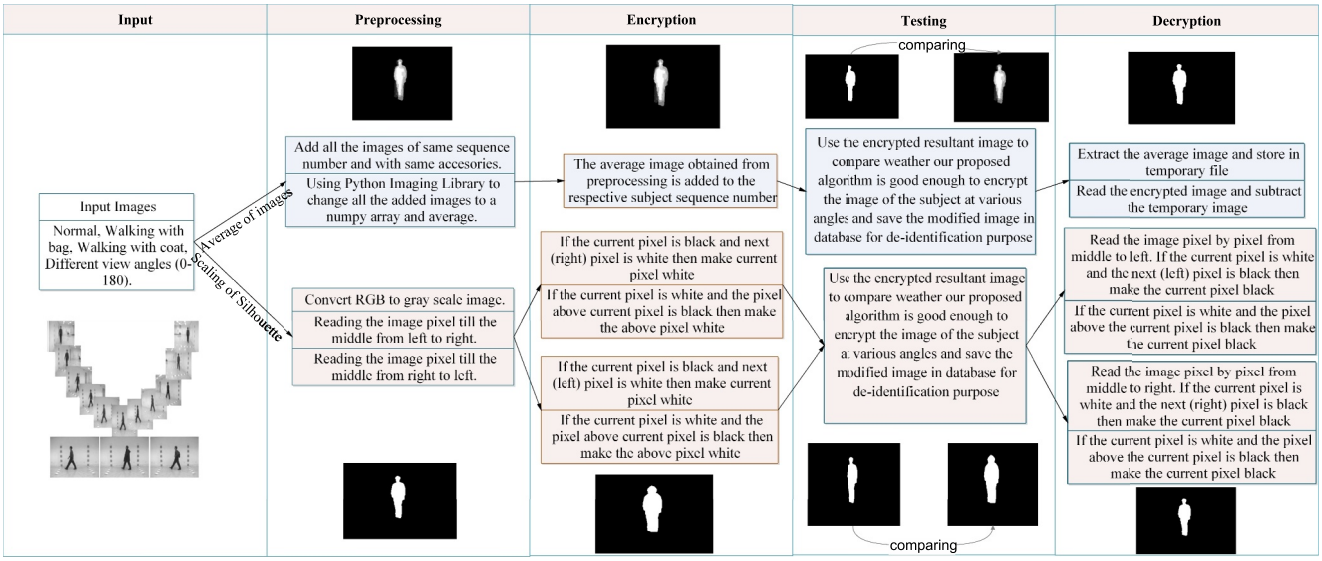


FIGURE 2 Detailed explanation of two anonymization techniques. First technique is textural modification where images are averaged and superimposed. Second technique is geometrical modification where scaling of silhouette is performed

10. Then use CV2. add (img1, img2) to add the image, thus encrypting the data set.

3.3.2 | Average—decryption

In this phase, the identification of images is done for recognition purposes. Algorithm 2 describes the method for decrypting the images.

Algorithm 1 Average—Encryption

```
Input: Image
Output: Encrypted Image
counts ← 1;
def avgImage(imlist):
w,h=Image.open(imlist[2]).size
for deg ← 0 to 181 do
    for seq ← 59 to 72 do
        if "−01"infilename then
            x ←("00" + str(seq))
        end
        if seq > 9 then
            x ←("0" + str(seq))
        end
        if deg > 0||deg < 100 then
            degStr ←("0" + str(deg))
        end
        if "nm−01"infilename then
            for a ← 0 to avergaeList do
                for b ← a do
                    data = Image.fromarray(avgList[a][2])
                    data.save('temp.png')
                end
            end
        end
    end
end
```

1) First, take the image provided by the user to decrypt and extract all the details from the image name.
2) The details will contain the walking style, the subject number, and the sequence number of the image.
3) After having all the details regarding the image, run the previous encryption algorithm to get the average image for that particular sequence.
4) The average will be in the form of a NumPy array that will be converted to a temporary image.
5) This temporary image and the provided image will then be read by the Cv2. imread (path of the file) function.
6) Using Cv2. subtract (img2-img1), we will be able to extract the original unencrypted image from the provided image.
7) After obtaining the original image, we can run a similarity algorithm that will come to 100%, as the image will now be decrypted.

3.3.3 | Scaling—encryption

This is the second phase to de-identify the images. Algorithm 3 describes the method for encrypting the images and explains its steps.

Traverse the image pixel by pixel and change the colour of the layer of pixels surrounding the silhouette to the same colour of the silhouette to scale the image. The steps are given below.

1. Traverse the data set subject by subject, encrypt all the different walking types, angles, and sequences by using the self-made scaling function and read the image using the function def open. What the scaling function does here is to make the layer of pixels surrounding the silhouette the same colour as the silhouette. Thus the image is scaled.

Algorithm 2 Average—Decryption

```
Input: Encrypted Image
Output: Decrypted Image
counts ← 1;
def avgImage(imlist):
for deg ← 0 to 181 do
    for seq ← 59 to 72 do
        if "−01"infilename then
            x ←("00" + str(seq))
        end
        if seq > 9 then
            x ←("0" + str(seq))
        end
        if deg > 0||deg < 100 then
            degStr ←("0" + str(deg))
        end
        if "nm − 01"or"bg − 01"or"cl − 01"infilename
         then
            returnedImage_nm = avgImage(tempNm)
            tempNm.append(returnedImage_xx)
            decryptImage(nm_sub)
            decryptImage(bg_sub)
            decryptImage(cl_sub)
        end
    end
end
```

2. As an image gets read by the function def open, first it is in greyscale. Convert greyscale images to RGB images by using the function def convert. Now calculate the image size by using the function size, which returns the requested size in pixels as a 2-tuple: (width, height), and we store it in variables w and h.
3. Traverse the image pixel by pixel to scale the image as the resize function in Python does not resize or scale any specific part of the image.
4. Traverse the image pixel by pixel from left to right and right to left, respectively, till the middle covers the whole image. For traversing, we are using 2 nested loops. The first loop traverses the width and the second loop traverses the height.
5. Use the function def getpixel to get the value of pixels, which denotes what colour the pixel is. It is in the format of a tuple (R, G, B). White is denoted by (255, 255, 255), and black is denoted by (0,0,0). These are the only colours present in the image. The value of the current pixel is stored in r1, g1, and b1, the value of the next pixel (right and left, respectively) to the current pixel is stored in r2, g2, and b2, and the value of the pixel above the current pixel is stored in r3, g3, and b3.
6. Following conditions are checked:
   If the current pixel is black (i.e. $r1 == 0$) and the next pixel is white (i.e. $r2 == 255$), make the current pixel white by using the function def putpixel. If the current pixel is white (i.e. $r1 == 255$) and the pixel above the current pixel is black (i.e. $r3 == 0$), make the above pixel white by using the function def putpixel.
7. Save the image using the function def save.

Algorithm 3 Scaling—Encryption

```
Input: Image
Output: Encrypted Image
im ← Image.open(new_path);
rgb_im = im.convert('RGB') ;        /* Converting
 Grayscale to RGB */
w,h ← im.size;
d ← int((w − 1)/2);
for i ← 0 to 3 do
    ;       /* change range according to the
     layers of encryption to be done */
    for x ← 0 to d do
        ;       /* TRAVERSING LEFT TO RIGHT */
        for y ← 0 to (h − 1) do
            r1,g1,b1 = rgb_im.getpixel((x,y))
            r2,g2,b2 = rgb_im.getpixel((x + 1,y))
            r3,g3,b3 = rgb_im.getpixel((x,y − 1))
            if r1 == 0&&r2 == 255 : then
                ;  /* If current pixel is black
                 and next(right) pixel is
                 white, make current pixel
                 white */
                rgb_im.putpixel((x,y),(255,255,255))
            end
            if r1 == 255&&r3 == 0 then
                ;  /* If current pixel is white
                 and above pixel is black,
                 make above pixel white */
                rgb_im.putpixel((x,y − 1),(255,255,255)))
            end
        end
    end
    for x ← 0 to w − 1,d − 1,−1 do
        ;       /* TRAVERSING RIGHT TO LEFT */
        for y ← 0 to (h − 1) do
            r1,g1,b1 = rgb_im.getpixel((x,y))
            r2,g2,b2 = rgb_im.getpixel((x − 1,y))
            r3,g3,b3 = rgb_im.getpixel((x,y − 1))
            if r1 == 0&&r2 == 255 : then
                ;  /* If current pixel is black
                 and next(left) pixel is
                 white, make current pixel
                 white */
                rgb_im.putpixel((x,y),(255,255,255))
            end
            if r1 == 255&&r3 == 0 then
                ;  /* If current pixel is white
                 and above pixel is black,
                 make above pixel white */
                rgb_im.putpixel((x,y − 1),(255,255,255)))
            end
        end
    end
    rgb_im.save('encrpted')
end
;                       /* Saving the image */
```

3.3.4 | Scaling—decryption

In this phase, identification of images is done for recognition purposes. Algorithm 4 describes the method for decrypting the images and explains its steps.

1. Traverse the data set subject by subject, encrypt all the different walking types, angles, and sequences by using the

self-made decryption function and read the image using the function def open.

What the decryption function does here is to make the encrypted layer of pixels surrounding the silhouette the same colour as the background. Thus the image is scaled down.

2. As an image gets read by the function def open, first it is in greyscale. Convert greyscale images to RGB images by using the function def convert.

   Calculate the size of the image by using the function size, which returns the requested size in pixels as a 2-tuple: (width, height), and we store it in variables w and h.
3. Traverse the image pixel by pixel to scale down the image to the original, as the resize function in Python does not resize or scale any specific part of the image.
4. Traverse the image pixel by pixel from the middle to the left and from the middle to the right, respectively, to cover the whole image. To traverse, we use 2 nested loops. The first loop traverses the width and the second loop traverses the height.
5. Use the function def get the pixel to get the value of pixels, which denotes what colour the pixel is. It is in the format of a tuple (R, G, B). White is denoted by (255, 255, 255), and black is denoted by (0, 0, 0). These are the only colours present in the image.

   The value of the current pixel is stored in r1, g1, and b1, the value of the next (left and right, respectively) pixel to the current pixel is stored in r2, g2, and b2, and the value of the pixel above the current pixel is stored in r3, g3, and b3.
6. Check the following conditions: If the current pixel is black (i.e. $r1$ == 255) and the next pixel is white (i.e. $r2$ == 0), make the current pixel black by using the function def putpixel.

   If the current pixel is white (i.e. $r1$ == 255) and the pixel above the current pixel is black (i.e. $r3$ == 0), make the current pixel black by using the function def putpixel.
7. Save the image using the function def save.

A geometric transformation can be done using rotation [34] and a few more ways [35–39]. We tried those, but they did not work well in our research, like rotating the parts of the silhouette by some degrees. We will only make slight changes to encrypt the silhouette while maintaining the humanoid figure of the silhouette. This method did not work as when we tried to rotate an image using the rotate or transpose function using the pillow library in Python, the whole image got rotated, not a specific part of the image. Even after approaching the problem with the new solution, that is, traversing the image pixel by pixel, the pixels could not be rotated individually; thus, this part of the encryption (i.e. rotation) was scrapped.

Scaling: This method involves scaling up or scaling down the silhouette slightly to encrypt the silhouette while maintaining the humanoid figure of the silhouette. The first approach was to scale up specific body parts [40] (i.e. head, arm, and feet) of the silhouette by different percentages to encrypt the image. However, this approach failed because when we tried to scale the image using the resize function present in the cv2 library of Python, the whole image got scaled, not any specific body part, as the resize function takes the whole image as a parameter. This method did not give the desired encryption results and thus was not used.

Algorithm 4 Scaling—Decryption

```
Input: Encrypted Image
Output: Decrypted Image
im ← Image.open(new_path);
rgb_im = im.convert('RGB') ;        /* Converting
 Grayscale to RGB */
w,h ← im.size;
d ← int((w − 1)/2);
for i ← 0 to 3 do
    ;       /* change range according to the
     layers of decryption to be done */
    for x ← 0 to d,1,−1 do
        ;   /* TRAVERSING FROM MIDDLE TO LEFT
         */
        for y ← 0 to (h − 1,1,−1) do
            r1,g1,b1 = im1.getpixel((x,y))
             r2,g2,b2 = im1.getpixel((x − 1,y))
             r3,g3,b3 = im1.getpixel((x,y − 1))
            if r1 == 255&&r2 == 0 : then
                ;  /* If current pixel is white
                 and next(left) pixel is
                 black, make current pixel
                 black */
                im1.putpixel((x,y),(0,0,0))
            end
            if r1 == 255&&r3 == 0 then
                ;  /* If current pixel is white
                 and above pixel is black,
                 make current pixel black */
                im1.putpixel((x,y),(0,0,0))
            end
        end
    end
    for x ← 0 to d,w − 1,1 do
        ;    /* TRAVERSING MIDDLE TO RIGHT */
        for y ← 0 to (h − 1,1,−1) do
            r1,g1,b1 = im1.getpixel((x,y))
             r2,g2,b2 = im1.getpixel((x + 1,y))
             r3,g3,b3 = im1.getpixel((x,y − 1))
            if r1 == 225&&r2 == 0 : then
                ;  /* If current pixel is white
                 and next(right) pixel is
                 black, make current pixel
                 black */
                im1.putpixel((x,y),(0,0,0))
            end
            if r1 == 255&&r3 == 0 then
                ;  /* If current pixel is white
                 and above pixel is black,
                 make current pixel black */
                im1.putpixel((x,y),(0,0,0)))
            end
        end
    end
    im1.save('decrypted')
end
;                          /* Saving the image */
```

We even tried face detection algorithms [41] to select the head of the body and then resize the image, but again the resize function present in the cv2 library could not give the desired results. After this approach, we tried to detect the pixel positions at the edge of the silhouette [42] by using the Canny Edge Detection algorithm [43]. Even though the resultant image was showing the edges of the silhouette, it could not give us the position of the pixels at the edges; thus, we could not make the layer of pixels surrounding the edges of the silhouette the same colour as the silhouette (i.e. from black to white) to scale up the silhouette [32]. After this, we ended up on our main and final approach, which was rather successful comparatively.

4 | EXPERIMENT SETUP AND RESULTS

We performed two types of experiments. We used a crowdsourcing approach in the first type of the experiment, and the second was automatic gait recognition based on ResNet [44].

4.1 | Crowdsourcing experiment

We compiled 30 gait images (7 females and 23 males, ranging from 30 to 75) from the CASIA B data set. Images of de-identified gaits are presented to the test subjects with a request to try to recognise them with the original images. Obtained answers are recorded for later matching with ground truth answers. The main aim of the performed testing is to evaluate the impact of geometrical and texture modifications on the human ability to recognise gaits. The evaluation is performed by means of crowdsourcing performed by 150 test subjects (20 females and 130 males). The test subjects were informed that gaits in the tests are de-identified gaits. For example, some are shown in (Figures 3, 4, and 5). Upon asking, the images look completely similar to the silhouette of a human, that is, the naturalness of the silhouette is preserved.

4.2 | Scaling algorithm—geometical modification

After running the scaling algorithm and encrypting the images (Figure 3.), we observed that as we increasingly change the layer of pixels surrounding the silhouette from black to white, the image gets encrypted more and more, and the similarity drops.

4.3 | Averaging algorithm—textural modification

1) The original image of the first subject of the 059 sequence number is carrying a bag (Figure 4a).
2) The image after taking the average of all the subjects with the sequence number 59 and with a viewing angle of 11 degrees (Figure 4b).
3) The resultant image when we add the average image to the stock/original image (Figure 4c).

As we can see, the resultant image does not look similar to the original image to the naked eye and also preserves the naturalness of the silhouette.

4.4 | Automatic gait recognition based on ResNet

ResNet network with 29 convolution layers is used. The network was trained on a data set of about 3 million gaits. Identical test samples from one to five experiments described above are used to evaluate a level of privacy protection when the machine recognition approach is used. We are interested in comparing results obtained by humans and machines. Table 1 depicts the results. From the results, we can conclude that the machine is much better than humans in the task of recognising de-identified gaits (i.e. with altered visual appearances). We have performed five experiments.

4.4.1 | First experiment

The geometry of a gait is left unchanged, and the texture of a gait is replaced with the average texture obtained from all the gaits of the subject. All de-identified gaits thus have the same texture and the original geometry (Figure 4). The following results are obtained in the experiment: From the total of 4500 de-identified gaits, only 60 de-identified gaits looked similar (i.e. 133% fail rate of de-identification).

4.4.2 | Second experiment

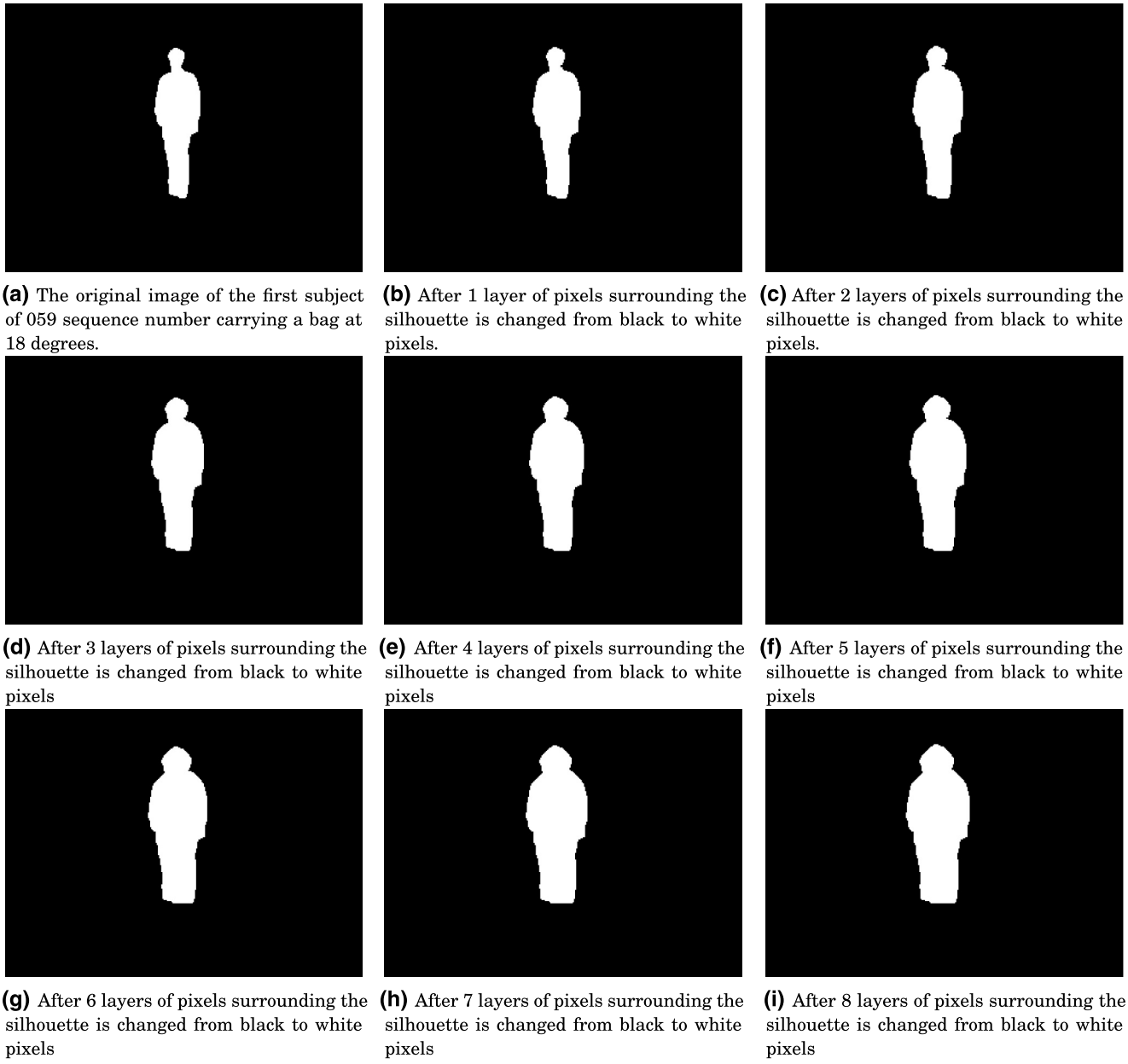
The geometry of a gait is changed as described in Section 3. A texture of a gait is obtained by scaling 50% of an original texture and 50% of an average gait texture (Figure 5). The following results are obtained: From the total of 4500 de-identified gaits, 260 de-identified gaits looked similar (i.e. 5.78% fail rate of de-identification).

4.4.3 | Third experiment

The geometry of a gait is changed as described in Section 3. An original gait texture was left unchanged (Figure 3b, c). The following results are obtained: From the total of 4500 de-identified gaits, original identities of 1011 de-identified gaits were revealed (i.e. 22.46% fail rate of de-identification).

4.4.4 | Fourth experiment

The geometry of a gait is changed to an average geometry obtained from 30 gaits (i.e. average positions of 6, 7, and 8



(a) The original image of the first subject of 059 sequence number carrying a bag at 18 degrees.

(b) After 1 layer of pixels surrounding the silhouette is changed from black to white pixels.

(c) After 2 layers of pixels surrounding the silhouette is changed from black to white pixels.

(d) After 3 layers of pixels surrounding the silhouette is changed from black to white pixels

(e) After 4 layers of pixels surrounding the silhouette is changed from black to white pixels

(f) After 5 layers of pixels surrounding the silhouette is changed from black to white pixels

(g) After 6 layers of pixels surrounding the silhouette is changed from black to white pixels

(h) After 7 layers of pixels surrounding the silhouette is changed from black to white pixels

(i) After 8 layers of pixels surrounding the silhouette is changed from black to white pixels

FIGURE 3 Encrypted picture obtained after 8 layers of pixels surrounding the silhouette are changed from black to white pixels

layers of gait feature points). At the same time, the texture is left unchanged (Figure 3, second row). The following results are obtained: From the total of 4500 de-identified gaits, the original identities of 480 de-identified gaits were revealed (i.e. 10.66% fail rate of de-identification).

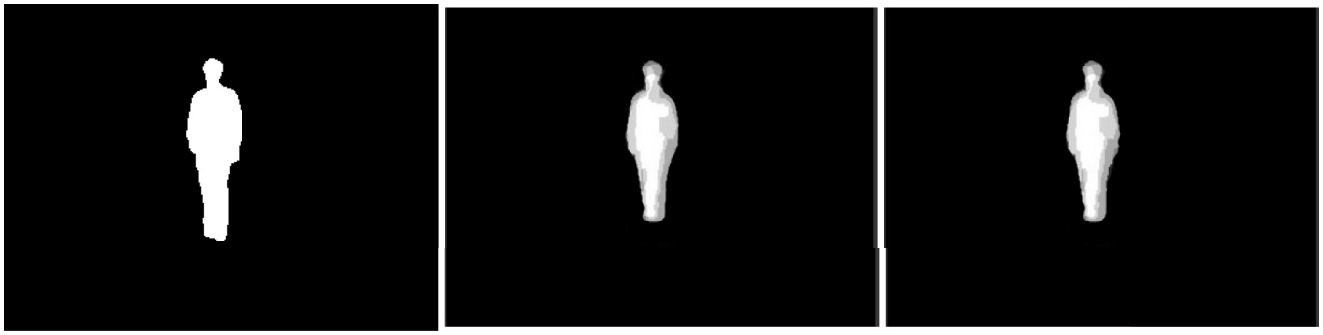
4.4.5 | Fifth experiment

Original images of subjects are used (Figure 3a). The following results are obtained: From the total of 4500 de-identified gaits, 4420 true identities were known (i.e. 98.22%). This also shows that the model's actual accuracy is 98.22.

The results of the above first four experiments have shown that texture is even more important than gait geometry.

4.5 | Similarity and naturalness

After we have applied the algorithm methods discussed above, we could see that there are some features of the humanoid/silhouette that are not visible to the naked eye, thus encrypting the image a little bit (Figure 3g, h, i). Running a similar image algorithm, we can see that images are encrypting a little bit but not to the extent that will hinder any algorithm from detecting whether it is the same person. While passing through a deep



(a) The original image of the first subject of the 059 sequence number carrying a bag.

(b) Image after taking the average of all the subjects with the sequence 59 (carrying a bag).

(c) Resultant image after adding the average image to the stock/original image.

FIGURE 4 Encrypted picture obtained after averaging all the silhouettes

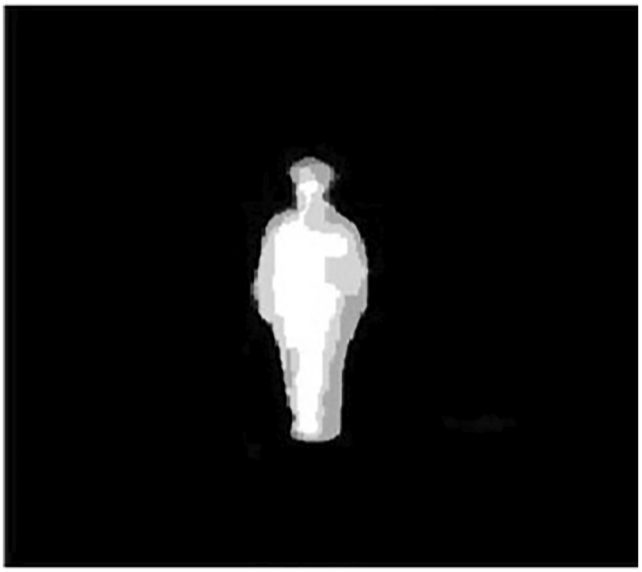


FIGURE 5 A texture of a gait is obtained by scaling 50% of an original texture and 50% of an average gait texture

TABLE 1 Failed rate of de-identification (recognition rate) for humans and machine

Experiment	Human/Croudsourcing	Machine/Resnet
1	0.1333	0.0133
2	0.2333	0.0578
3	0.4667	0.2246
4	0.3306	0.1066
5	1	0.9882

learning pipeline/image similarity detector, the image is 5%–23% detected, thus making our approach quite strong and safe if we apply it in real-world scenarios for de-identification (Table 1).

Executing both of our logic, now is the time to check how many similarities our modified image gets when we compare it against the original stock image. So for testing our code, we used the Triplet Sampling algorithm [45], through which we were able to identify how much similarity is present between the original and our modified image. We introduce a non-metric similarity function s (x,x') to compare two silhouette x and x'. This is a symmetric function whose value is large when x and x' are somehow "similar." For example, when the angle between two vectors is a meaningful measure of their similarity, then the normalised inner product

$$[s(x, x')] = \frac{x^t x'}{\|x\|\|x'\|} \tag{2}$$

It may be an appropriate similarity function. This measure, the cosine of the angle between x and x', is invariant to rotation and dilation, though it is not invariant to translation and general linear transformations. When the features are binary-valued (0 or 1), the similarity function of Equation (2) has a simple non-geometrical interpretation in terms of shared features or shared attributes. In our x possesses the i^th attribute if $x_i = L$, then $x^t x'$ is merely the number of attributes possessed by both x and x', and $\|x\|\|x'\| = (x^t x x'^t x')^1/2)$ is the geometric mean of the number of attributes possessed by x and the number possessed by x'. Thus, s (x,x') measures the relative possession of common attributes.

These results should lead to new research avenues in gait recognition privacy protection. Our study relied on silhouette sequences and video frames as a data source. It also involves the application of this technology to the anonymisation of object images.

4.5.1 | Comparison with state-of-the-art

The success rate is calculated as the proportion of times the gait recognition algorithm failed to recognise the gait. In our studies, the average success rate in all covariate situations is 100%, while in Ref. [6], the average success rate varied from 48.57% to 86.25%, depending on the view angle.

5 | CONCLUSION

In this paper, we proposed a hybrid reversible gait de-identification pipeline that combines the good qualities of naive and complex gait de-identification methods. The texture of a gait can be adaptively modified based on an original texture as in naive approaches. Consequently, we have removed the need for using a tedious texture model of gaits. The geometry of a gait can be completely modified as in complex methods. These geometrical modifications performed by scaling transformation are pseudo reversible, making them compliant with security requirements. Only eight geometrical modification layers are used, making them suitable for steganographic encoding into the de-identified image. For texture modification, a fixed texture template is used. We have investigated the impact of various geometrical and texture modifications of gait components on the ability of humans and machines to recognise gaits. The crowdsourcing and machine gait recognition experiments have shown that modification of a gait texture has a stronger impact on the level of privacy protection than gait geometry (shape) modifications. We build a model that will take a large amount of the data set, will encrypt the data set, and check for similarity.

CONFLICT OF INTEREST

The author declares that there is no conflict of interest that could be perceived as prejudicing the impartiality of the research reported.

DATA AVAILABILITY STATEMENT

Data subject to third party restrictions.

ORCID

Anubha Parashar https://orcid.org/0000-0002-8474-3623

REFERENCES

1. Ribaric, S., et al.: De-identification for privacy protection in multimedia content: a survey. Signal Processing Image Communication 47, 131–151 (2016). ISSN 0923–5965. https://doi.org/10.1016/j.image.2016.05.020
2. Rida, I., et al.: A comprehensive overview of feature representation for biometric recognition. Multimed. Tool. Appl. 79(7), 4867–4890 (2020). https://doi.org/10.1007/s11042-018-6808-5
3. Wang, L.: EventSR: from Asynchronous events to image reconstruction, restoration, and super-resolution via end-to-end adversarial learning. Arxiv (2020)
4. Jia, M.: Attacking gait recognition systems via silhouette guided GANs. Int. Conf. on Multimedia (2019)
5. Rida, I., Almaadeed, N., Almaadeed, S.: Robust gait recognition: a comprehensive survey. IET Biom. 8(1), 14–28 (2019). https://doi.org/10.1049/iet-bmt.2018.5063
6. Tieu, N.-D.T.: An approach for gait anonymization using deep learning. In WIFS, 1–6 (2017)
7. He, Z.: Temporal sparse adversarial attack on gait recognition. Arxiv (2020)
8. W, X., et al.: Frame-GAN: Increasing the frame rate of gait videos with generative adversarial networks. Neurocomputing 380, 95–104 (2020). https://doi.org/10.1016/j.neucom.2019.11.015
9. Rauf, M., et al.: Knowledge transfer between networks and its application on gait recognition. In: 2016 IEEE International Conference on Digital Signal Processing (DSP), pp. 492–496 (2016)
10. McLaughlin, N., Martinez del Rincon, J., Miller, P.: Recurrent convolutional network for video-based person Re-identification. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1325–1334 (2016)
11. Sokolova, A., Konushin, A.: Gait recognition based on convolutional neural networks. In: International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences, 42 (2017)
12. Marín-Jiménez, M.J., et al.: Deep multi-task learning for gait-based biometrics. In: 2017 IEEE International Conference on Image Processing (ICIP), pp. 106–110 (2017)
13. Nithyakani, P., Shanthini, A., Ponsam, G.: Human gait recognition using deep convolutional neural network. In: 2019 3rd International Conference on Computing and Communications Technologies (ICCCT), pp. 208–211 (2019)
14. Carley, C., Ristani, E., Tomasi, C.: Person Re-identification from gait using an Autocorrelation network. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 2345–2353 (2019)
15. Song, C., et al.: GaitNet: an end-to-end network for gait based human identification. Pattern Recogn. 96, 106988 (2019). https://doi.org/10.1016/j.patcog.2019.106988
16. Arshad, H., et al.: A multilevel paradigm for deep convolutional neural network features selection with an application to human gait recognition. Expet Syst. 39(7) (2020). https://doi.org/10.1111/exsy.12541
17. Xu, Z., et al.: Gait recognition based on capsule network. J. Vis. Commun. Image Represent. 59, 159–167 (2019). https://doi.org/10.1016/j.jvcir.2019.01.023
18. Makihara, Y., et al.: Gait recognition by deformable registration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 561–571 (2018)
19. Sun, Y., Liu, Q.: Attribute recognition from clothing using a Faster R-CNN based multitask network. Int. J. Wavelets, Multiresolut. Inf. Process., 16 (2018)
20. Babaee, M., Li, L., Rigoll, G.: Gait recognition from incomplete gait cycle. In: 2018 25th IEEE International Conference on Image Processing (ICIP), pp. 768–772 (2018)
21. Babaee, M., Li, L., Rigoll, G.: Gait energy image reconstruction from degraded gait cycle using deep learning. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops (2018)
22. Babaee, M., et al.: Gait energy image restoration using generative adversarial networks. In: 2019 IEEE International Conference on Image Processing (ICIP), pp. 2596–2600. IEEE (2019)
23. Xia, L.M., Wang, H., Guo, W.T.: Gait recognition based on wasserstein generating adversarial image inpainting network. J. Central South University, 26, 2759–2770 (2019)
24. Babaee, M., Li, L., Rigoll, G.: Person identification from partial gait cycle using fully convolutional neural networks. Neurocomputing 338, 116–125 (2019). https://doi.org/10.1016/j.neucom.2019.01.091
25. Li, X., et al.: Make the bag Disappear: carrying status-invariant gait-based human Age estimation using Parallel generative adversarial networks. In: 2019 IEEE 10th International Conference on Biometrics Theory, Applications and Systems (BTAS), pp. 1–9 (2019)
26. Das, D., et al.: RGait-NET: an effective network for recovering missing information from occluded gait cycles (2019)
27. He, Y., et al.: Multi-task GANs for view-specific feature learning in gait recognition. IEEE Trans. Inf. Forensics Secur. 14(1), 102–113 (2019). https://doi.org/10.1109/tifs.2018.2844819
28. Uddin, M.Z., et al.: Spatio-temporal silhouette sequence reconstruction for gait recognition against occlusion. IPSJ Transactions on Computer Vision and Applications 11(1), 1–18 (2019). https://doi.org/10.1186/s41074-019-0061-3
29. Pappas, I.P.I., et al.: A reliable gait phase detection system. IEEE Trans. Neural Syst. Rehabil. Eng. 9(2), 113–125 (2001). https://doi.org/10.1109/7333.928571
30. Jin, Q., et al.: Multi-feature hybrid gait recognition algorithm based on iterative closest point algorithm (2015)
31. Gao, S., et al.: Gait-D: skeleton-based gait feature decomposition for gait recognition. IET Comput. Vis. 16(2), 111–125 (2022). https://doi.org/10.1049/cvi2.12070

32. Rida, I., Almaadeed, S., Bouridane, A.: Gait recognition based on modified phase-only correlation. Signal. Image and Video Processing 10(3), 463–470 (2016). https://doi.org/10.1007/s11760-015-0766-4
33. Zheng, S., et al.: Robust view transformation model for gait recognition In: International Conference on Image Processing (ICIP). Belgium, Brussels (2011)
34. Gidaris, S., Singh, P., Komodakis, N.: Unsupervised representation learning by predicting image rotations. arXiv preprint arXiv:1803.07728. (2018)
35. Rida, I., et al.: Improved human gait recognition. In: International Conference on Image Analysis and Processing, pp. 119–129. Springer, Cham (2015)
36. Rida, I., et al.: Robust model-free gait recognition by statistical dependency feature selection and globality-locality preserving projections. In: 2016 39th International Conference on Telecommunications and Signal Processing (TSP), pp. 652–655. IEEE (2016)
37. Rida, I., et al.: Improved model-free gait recognition based on human body part. In: Biometric Security and Privacy, pp. 141–161. Springer, Cham (2017)
38. Rida, I.: Towards human body-part learning for model-free gait recognition. arXiv preprint arXiv:1904.01620. (2019)
39. Rida, I.: Temporal signals classification. (Classification de signaux temporels). Doctoral dissertation, Normandy University, France (2017)
40. Isack, H., et al.: Repose: learning deep kinematic priors for fast human pose estimation. arXiv preprint arXiv:2002.03933 (2020)
41. Dang, K., Sharma, S.: Review and comparison of face detection algorithms. In: 2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence, pp. 629–633. IEEE (2017)
42. Rida, I.: Feature extraction for temporal signal recognition: an overview. arXiv preprint arXiv:1812.01780 (2018)
43. Rong, W., et al.: An improved CANNY edge detection algorithm. In: 2014 IEEE International Conference on Mechatronics and Automation, pp. 577–582. IEEE (2014)
44. He, K., et al.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
45. Wang, L., Rajan, D.: Deepu Rajan, an image similarity descriptor for classification tasks. J. Vis. Commun. Image Represent. 71, 102847 (2020). ISSN 1047-3203. https://doi.org/10.1016/j.jvcir.2020.102847

How to cite this article: Parashar, A., Shekhawat, R.S.: Protection of gait dataset for preserving its privacy in deep learning pipeline. IET Biome. 1–13 (2022). https://doi.org/10.1049/bme2.12093