In Ictu Oculi: Exposing AI Created Fake Videos by Detecting Eye Blinking

Yuezun Li, Ming-Ching Chang and Siwei Lyu
University at Albany, State University of New York, USA

Abstract

The new developments in deep generative networks have significantly improve the quality and efficiency in generating realistically-looking fake face videos. In this work, we describe a new method to expose fake face videos generated with deep neural network models. Our method is based on detection of eye blinking in the videos, which is a physiological signal that is not well presented in the synthesized fake videos. Our method is evaluated over benchmarks of eye-blinking detection datasets and shows promising performance on detecting videos generated with DNN based software `DeepFake`.

1. Introduction

The increasing sophistication of camera technology, the wide availability of cellphones and the ever-growing popularity of social networks (FaceBook, Twitter, WhatsApp, InstaGram, and SnapChat) and video sharing portals (YouTube and Vemeo) have made the creation, editing and propagation of digital videos more convenient than ever. This has also brought forth tampering of digital videos. Unlike digital images, editing videos has been a time-consuming and painstaking task due to the lack of sophisticated editing tools like `Adobe PhotoShop` and the large number of editing operations involved for a video – as a case in point, a 20 second video with 25 frames per second requires editing of 500 images. As such, highly realistic fake videos were rare, and most can be identified relatively easily based on some conspicuous visual artifacts.

However, the situation was changed dramatically with the recent development of generative deep neural networks, in particular, generative adversary networks (GANs) [11, 21], which has led the development of tools that can generate videos from large volume of images with minimum manual editing. The situation first caught the public attention in earlier 2018, when a software known as `DeepFake` was made publicly available. In particular, `DeepFake` uses GANs to replace faces of one individual in a video with synthesized faces of another individual (see Figure1). Subsequently, there have been a surge of fake videos generated using this tool and uploaded to `YouTube` for gross violations of privacy and identity, some with serious legal implications [1]. Detecting such fake videos becomes a pressing need for the research community of digital media forensics.

While traditional media forensic methods based on signal level cues (*e.g*, sensor noise, CFA interpolation and double JPEG compression), physical level evidence (*e.g*, lighting condition, shadow and reflection) or semantic level consistencies (*e.g*, consistency of meta-data) can be applied for this purpose, they are not sufficiently reliable or efficient for detecting `DeepFake` videos. This situation calls for novel detection techniques. In this work, we describe a method to expose `DeepFake` videos by detecting the lack of eye blinking of the synthesized faces.

Blinking refers to the rapid closing and opening movement of the eyelid. The spontaneous blink, which refers to blinking without external stimuli and internal effort, is controlled by the pre-motor brain stem and happens without conscious effort and serves an important biological function that moisturizes with tears and remove irritants from the surface of the cornea and conjunctiva. For a health adult human, generally, between each blink is an interval of 2-10 seconds but the actual rates vary by individual, and the length of a typical blink is 0.1-0.4 seconds/blink[2]. As such, we should expect to observe spontaneous eye blinking from a video of real humans with the aforementioned frequency and duration. However, this is not the case for many `DeepFake` videos, as the example in Figure 2 shows. This can be attributed to the fact that the core GAN model in `DeepFake` is trained based on large number of human face images. If we assume an average exposure time of 1/30 second, then the probability of capturing a photo with someone blinking is about 7.5%. Most photos of a person that can be obtained online will not show them with their eyes closed, so this likelihood is even smaller in practice. Therefore, the lack of eye blinking is thus a telltale sign of a `DeepFake` video.

[1] For example, see `https://www.lawfareblog.com/deep-fakes-looming-crisis-national-security-democracy-and-privacy`. As a result, `DeepFake` has been banned and excluded from the online community.

[2] `http://bionumbers.hms.harvard.edu/bionumber.aspx?id=100706\&ver=0`.

Our method uses a deep neural network model that combines CNN and recursive neural network, known as *long-term recurrent CNN* (LRCN) [7], to distinguish open and close eye states with the consideration of previous temporal knowledge. Our method is evaluated over benchmarks of eye-blinking detection datasets and also show promising performance on detecting videos generated with `DeepFake`.

2. Related Works

2.1. AI Generation of Fake Videos

Previously, realistic images/videos were generated using detailed 3D computer graphics models. Recently, the development of new deep learning algorithms, especially those based on the generative adversarial networks (GANs). Goodfellow *et al*. [11] first proposed generative adversarial networks (GANs), which typically consist of two networks: the generator network and the discriminator network. The generator aims to produce an image that cannot be distinguished from training images, while the discriminator differentiates training images from images synthesized by the generator network. The two networks are trained in tandem with the generator and the discriminator competing with each other – the generator tries to create images that can confuse the discriminator, while the discriminator tries to classify the synthetic images from the real training images.

Subsequently, many general image synthesis or face synthesis works have been proposed based on the idea of GANs. Denton *et al*. [5] proposed a Laplacian pyramid GAN to generate images in a coarse-to-fine fashion. Radford *et al*. [24] proposed deep convolutional GANs (DCGAN) and showed the potential for unsupervised learning. Arjovsky *et al*. [1] used Wasserstein distance to make training stable. Isola *et al*. [15] investigated conditional adversarial networks to learn mapping from input image to output image and also the loss function to train the mapping. Taigman *et al*. [32] proposed the domain transfer network (DTN) to map a sample from one domain to an analogous sample in another domain and achieved favorable performance on small resolution face and digit images. Shrivastava *et al*. [25] reduced the gap between synthetic and real image distribution using a combination of the adversarial loss and the self-regularization loss. Liu *et al*. [21] proposed an unsupervised image to image translation framework based on coupled GANs, which aims to learn the joint representation of images in different domains. This algorithm is the basis for the `DeepFake` algorithm , the process of which is given in Figure 1.

2.2. Eye Blinking Detection

Detecting eye blinking has previously been studied in computer visions for applications in fatigue detection [14, 34, 8, 2, 22] and face spoof detection [4, 10, 20, 30, 19]. Pan *et al*. [23] constructed undirected conditional random field framework to infer eye closeness such that eye blinking is detected. Sukno *et al*. [31] employed active shape models with invariant optimal features to delineate the outline of eyes and computed the eye vertical distance to decide eye state. Torricelli *et al*. [33] utilized the difference between consecutive frames to analyze state of eyes. Divjak *et al*. [6] employed optical flow to obtain eye movement and extract the dominant vertical eye movement for blinking analysis. Yang *et al*. [35] modeled the shape of eyes based on a pair of parameterized parabolic curves, and fit the model in each frame to track eyelid. Drutarovsky *et al*. [9] analyzed the variance of the vertical motions of eye region which is detected by a Viola-Jones type algorithm. Then a flock of KLT trackers are used on the eye region. Each eye region is divided into 3x3 cells and an average motion in each cell is calculated. Soukupova *et al*. [28] proposed a scalar quantity that measures the aspect ratio of the rectangular bounding box of an eye (eye aspect ratio – EAR) corresponding to the eye openness degree in each frame. They then trained an SVM of EARs within a short time window to classify final eye state. Kim *et al*. [16] studied CNN-based classifiers to detect eye open and close state. To date, we are not aware of deep NN based eye blinking detection algorithms.

3. Method

In this section, we describe in detail our method to detect eye blinking in a video. We extend the work on CNN-based classifier to LRCN [7], which incorporates the temporal relationship between consecutive frames, as eye blinking is a temporal process which is from opening to closed, such that LRCN can memorize the long term dynamics to remedy the effect by artifacts introduced from single image. The general overview of our algorithm is provided in Figure 2.

3.1. Pre-processing

The first step in our method is to locate the face areas in each frame of the video using a face detector. Then facial landmarks, which are locations on the face carrying important structural information such as tip of the eyes, noses and mouths and contours of the cheek, are extracted from each detected face area.

The head movement and changes in face orientation in the video frames introduce distractions in facial analysis. As such, we first align the face regions to a unified coordinate space using landmark based face alignment algorithms. Specifically, given a set of face landmarks in original coordinate, 2D face alignment is to warp and transform the image to another coordinate space, where the transformed face is (1) in the center of image, (2) rotated to make eyes lie on a horizontal line and (3) scaled to a similar size.

From the aligned face areas, we can extract a surrounding rectangular regions of the landmarks corresponding to

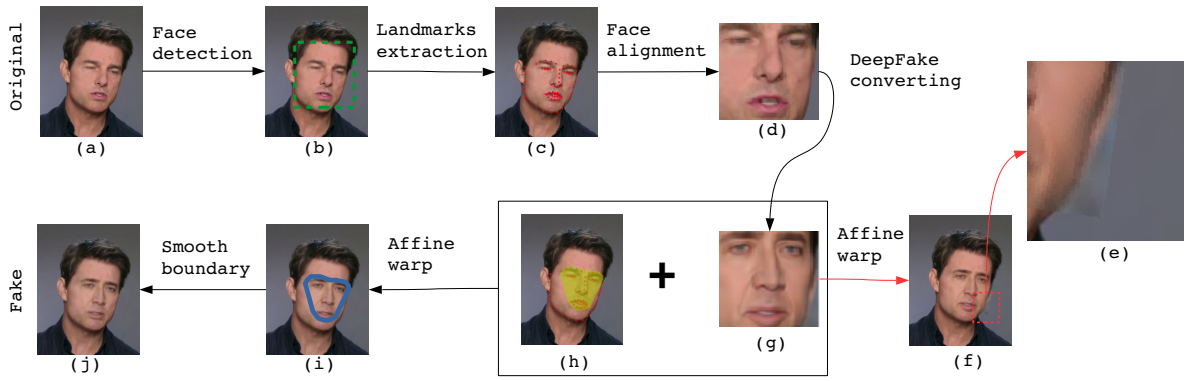

Figure 1. *Overview of fake face generation. (a) The original input image. The green dash box in (b) is the face area localized by face detector. (c) Detected face landmarks. (d) Face after alignment.* `DeepFake` *takes (d) as input and convert it to (g). The artifacts are introduced by directly affine warping generated face back to (a), as shown in (f, e). (h) The convex polygon mask that generated face inside is retained. (i) Smoothed boundary of mask. (j) The final fake image.*

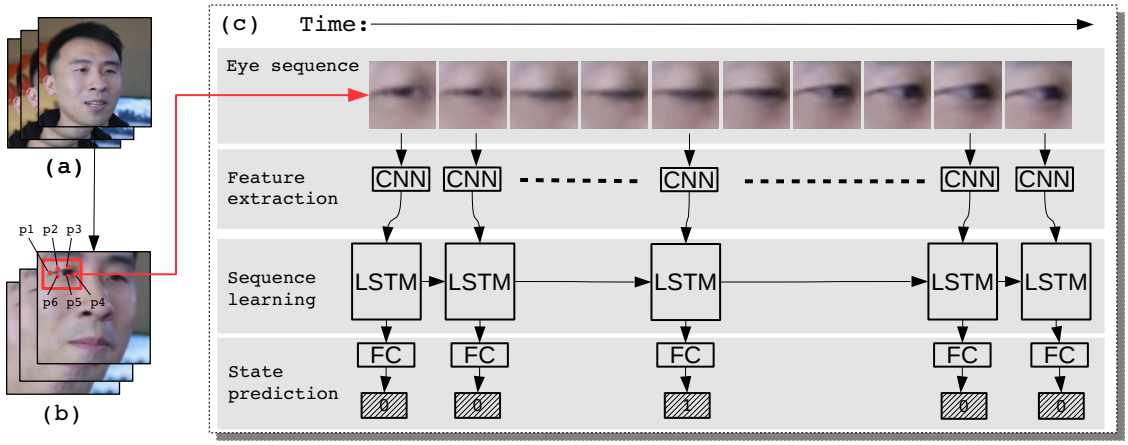

Figure 2. *Overview of our LRCN method. (a) is the original sequence. (b) is the sequence after face alignment. We crop out eye region of each frame based on eye landmarks $p_{1\sim6}$ in (b) and pass it to (c) LRCN, which consists of three parts: feature extraction, sequence learning and state prediction.*

the contours of the eyes into a new sequence of input frames, see Figure 2(b). Specifically, the rectangle region is generated by first extracting the bounding boxes of each eye's landmark points, then scaling the bounding box by 1.25 in the horizontal direction and 1.75 in the vertical direction, respectively, to ensure that the eye region is included in the cropped region. The cropped eye area sequences are passed into LRCN for dynamic state prediction.

3.2. Long-Term Recurrent CNNs

As human eye blinking shows strong temporal dependencies, we employ the long-term recurrent convolutional neural networks (LRCN) model [7] to capture such temporal dependencies. As shown in Figure 2(c), the LRCN model is composed by three parts, namely, *feature extraction*, *sequence learning* and *state prediction*. Feature extraction module converts the input eye region into discriminative features. It is implemented with a Convolutional Neural Network (CNN) based on the VGG16 framework [26] but without `fc7` and `fc8` layers[3]. VGG16 is composed by five blocks of consecutive convolutional layers `conv1` $\sim$ `5`, where max-pooling operation follows each block. Then three fully connected layers `fc6` $\sim$ `8` are appended on the last block. The output from the feature extraction is fed into sequence learning, which is implemented with a recursive neural network (RNN) with Long Short Term Memory (LSTM) cells [13]. The use of LSTM-RNN is to increase the memory capacity of the RNN model and avoid the gradient vanishing in the back-propagation-through-time (BPTT) algorithm in the training phase.

3.3. LSTM-RNN

LSTMs are memory units that control when and how to forget previous hidden states and when and how to update

[3]Other deep CNN architecture such as ResNet [12] can also be used but for simplicity we choose VGG16 in the current work.

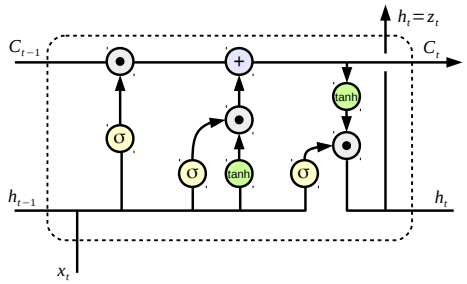

Figure 3. *A diagram of LSTM structure.*

hidden states [13]. We use LSTM as illustrated in Figure 3, where $\sigma(x) = \frac{1}{1+e^{-x}}$ is *sigmoid* function to push input into $[0, 1]$ range, $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ is *hyperbolic tangent* function which squash input into $[-1, 1]$ range, $\odot$ denotes inner product of two vectors. Given input C_{t-1}, h_{t-1}, x_t, the LSMT updates along with time t by

$$\begin{aligned} f_t &= \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f) \\ i_t &= \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i) \\ g_t &= tanh(W_{ch}h_{t-1} + W_{cx}x_t + b_c) \\ C_t &= f_t \odot C_{t-1} + i_t \odot g_t \\ o_t &= \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o) \\ h_t &= o_t \odot tanh(C_t) \end{aligned} \tag{1}$$

where f_t is forget gate to control what previous memories will be discard, i_t is input gate to selectively pass the current input, which is manipulated by g_t. o_t is output gate to control how much memory will be transferred into hidden state h_t. Memory cell C_t is combined by previous memory cell C_{t-1} controlled by f_t and manipulated input g_t controlled by i_t. We use 256 hidden units in LSTM cell, which is the dimension of LSTM output z_t.

For the final state prediction stage, the output of each RNN neuron is further sent to neural network consists of a fully connected layer, which takes the output of LSTM and generate the probability of eye open and close state, denoted by 0 and 1 respectively.

3.4. Model Training

The training of the LRCN model is performed in two steps. In the first step, we train the VGG based CNN model using a set of labeled training data consisting of eye regions corresponding to open and closed eyes. The model is trained using back-propagation implemented with stochastic gradient descent with dropout [29] in fully connected layers. In the second step, the LSTM-RNN and fully connected part of the network are trained jointly using the back-propagation-through-time (BPTT) algorithm. In both cases, the loss objective is cross entropy loss with binary classes (open or closed). Implementation details are given in the next section.

4. Experiments

We train the LRCN model based on image datasets of eye open states. We then test the algorithm detecting eye blinking on authentic and fake videos generated with the `DeepFake` algorithm.

4.1. Datasets

To date, there are a few image datasets that can be used for evaluating algorithms that detect closed eyes, such as the CEW Dataset [27][4], which includes $1,193$ images of closed eyes and $1,232$ images of open eyes. However, no existing video dataset specially designed for the same purpose is available, which is important due to the temporal nature of eye blinking.

To be able to experimentally evaluate our algorithm, we downloaded 50 videos, where each represents one individual and lasts approximate 30 seconds with at least one blinking occurred, to form the eye blinking video (EBV) dataset. We annotate the left and right eye states of each frame of the videos using an annotation tool we developed[5]. In our experiments, we select 40 videos as our training set for the overall LRCN model and 10 videos as the testing set. We use the CEW dataset together with our training set to train the front-end CNN model.

Furthermore, we use `DeepFake` with post-processing to generate fake face videos, see Figure 1. Specifically, we first use `dlib` to detect face area in each image. Then face landmarks are extracted for face alignment as described in section 3. We then generate the corresponding fake faces using the `DeepFake` algorithm. If we directly affine warp this rectangle of fake face back to image using similarity transformation matrix, the boundary of rectangle is visible in most cases as the slight color difference of real and fake face area, as shown in Figure 1(e). To reduce such artifacts, we generate a specific mask which is a convex polygon determined by landmarks of left and right eyebrow, and the bottom mouth. As such, we only retain content inside this mask after affine warping fake face back to original image. To further smooth the transformation, we apply Gaussian blur to the boundary of mask. We collect interview and presentation episodes from web and generated 49 such fake videos in total.

4.1.1 Data preparation

Face detection, landmark extraction and face alignment are implemented based on library `dlib` [17], which integrates

[4]Downloaded from `http://parnec.nuaa.edu.cn/xtan/data/ClosedEyeDatabases.html`. The other dataset, the EEG Eye State Data Set `https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State`, is not available to download. Similarly, the ZJU Eyeblink Video Database [23] is not accessible.

[5]Our dataset and annotation tool will be made available to download after the reviewer period.

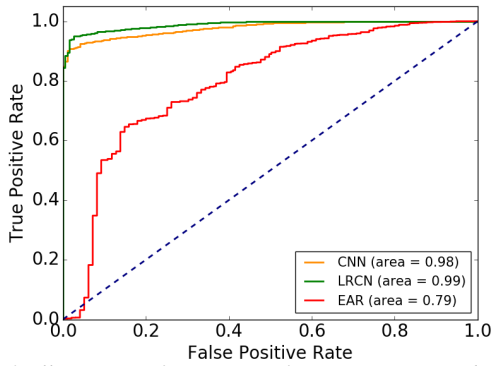

Figure 4. *Illustration of ROC curve for CNN, LRCN and EAR.*

the current state-of-the-art face analysis algorithms. We generate eye sequences by cropping out eye area of each frame of our video dataset.

We augment data to increase training robustness. The training of the front-end CNN model takes images as input, so we use each frame of generated eye sequences as training sample, with additional augmentation: horizontal flipping image, modifying image color contrast, brightness and color distortion. For LRCN joint training, eye sequences are required. In particular, the augmentation process for sequence should be consistent to avoid affect temporal relationship, such that the process for each frame in sequence should be same.

With combination of our cropped eye images and CEW dataset, we train VGG16 as a binary image classifier to distinguish eye state in image domain. The input size is fixed as 224x224 and the batch size is 16. The learning rate starts from 0.01 and decays by 0.9 each 2 epochs. We employ stochastic gradient descent optimizer and terminate training until it reaches the maximum epoch number 100. Then we remove `fc7`, `fc8` layers from trained VGG16 to be the feature extraction part of LRCN.

We randomly select a sequence which contains a variety of temporal consecutive eye images with at least one blinking occurred as LRCN input. Each sample has variable length between 10 to 20 images. We fix the parameters of CNN layers we obtain above and perform training on rest part: LSTM cells and `fc` layer. We set batch size as 4. The learning rate starts from 0.01 and decay by 0.9 each 2 epochs. We use the ADAM optimizer [18] and terminate training until 100 epochs.

4.2. Evaluations

We evaluate our LRCN method with comparison to other methods: the eye aspect ratio (EAR) based method [28] and a CNN-based method. The EAR-based method replies on eye landmarks to analyze eye state, in terms of the ratio between the distance of upper and lower eyelid, and the distance between left and right corner point, which is defined as $EAR = \frac{\|p_2-p_6\|+\|p_3-p_5\|}{2\|p_1-p_4\|}$, where $p_i(i = 1, \cdots, 6)$ correspond to the landmark points of an eye (see Figure 2(b)). The EAR-based method runs fast as the computation of EAR is simple. However, the main drawback of EAR method is that it fully depends on eye landmarks, which cannot be reliably detected in many practical videos. CNN image classifier is trained on image domain to distinguish different classes. We employ VGG16 as our CNN model to distinguish eye state. The problem with the CNN based method is that it cannot take into consideration of the temporal consistency during eye blinking.

Table 1. *Performance of our method on collected original videos and corresponding fake videos.*

Video	Average video length	FPS	Rate of blinks
Origin	10 seconds	30	**34.1** / min
Fake	10 seconds	30	**3.4** / min

We evaluate these three methods on the test videos with 32 blinking events. Figure 4 illustrate the ROC curve of three methods. Note that LRCN show the best performance 0.99 compared to CNN 0.98 and EAR 0.79 in terms of the area under ROC (AUC). CNN-based method shows a good performance to distinguish the eye state on image domain, but it can be further improved with LRCN, which considers long term dynamics to effectively predict eye state. A comparison is shown in Figure 5. When the actual eye area is small, CNN-based model using only image input is ineffective, while LRCN using temporal correlation can correctly predict.

We set 0.5 as threshold to distinguish eye open and close state. We define a blink as a peak above threshold 0.5 with duration less than 7 frames. Table 1 shows the performance of our method on 49 collected videos and corresponding 49 fake videos generated using `DeepFake`, as in Figure 1. Rate of blinks is the number of detected blinks per 60 seconds. We can detect 34.1/min blinks in original videos [6], whereas only 3.4/min blinks is detected in fake videos. If we set the average blinking rate of a normal human being to 10/min [3], then all `DeepFake` generated videos are below this standard. One visual example is shown in Figure 6, with more examples provided in the supplementary materials.

5. Conclusion

The new developments in deep generative networks have significantly improve the quality and efficiency in generating realistically-looking fake face videos. In this work, we describe a new method to expose fake face videos generated with deep neural networks. Our method is based on detection of eye blinking in the videos, which is a physiological signal that is not well presented in the synthesized fake videos. Our method is tested over benchmarks of eye-blinking detection datasets and also show promising perfor-

[6] As the collected videos are mainly interview/presentation episodes, the rate of blink is a bit higher than normal case.

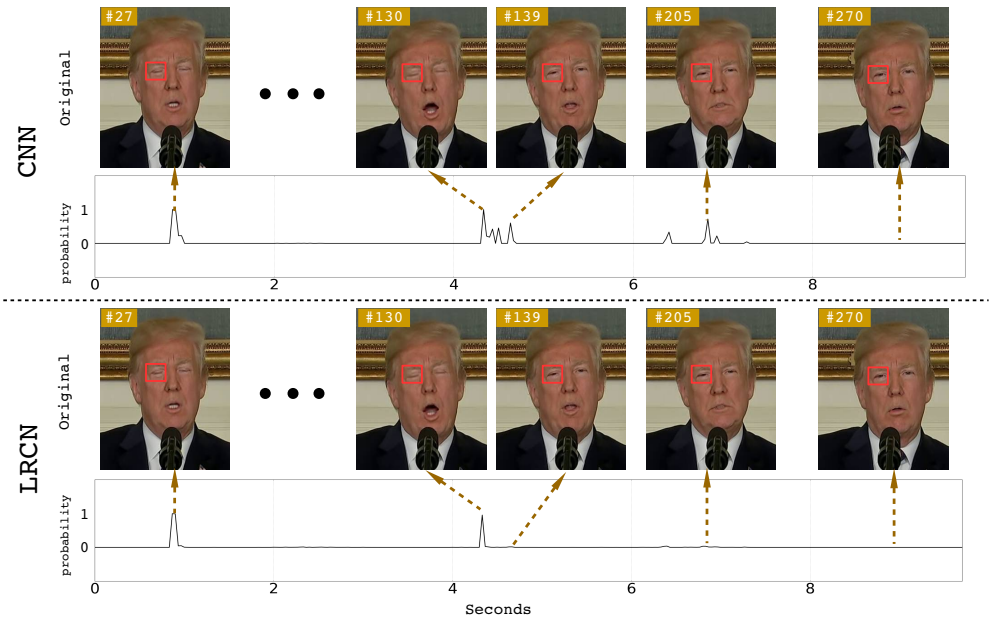

Figure 5. *Illustration of comparing CNN and LRCN on left eye of Trump video. LRCN exhibits more smooth and accurate results than CNN,* e.g, *if blinking has just occurred, the eyes in next couple frames are likely to be open (frame #139). If there is no trend of eye closing before, the eye state of next frame is likely to be open (frame #205)*

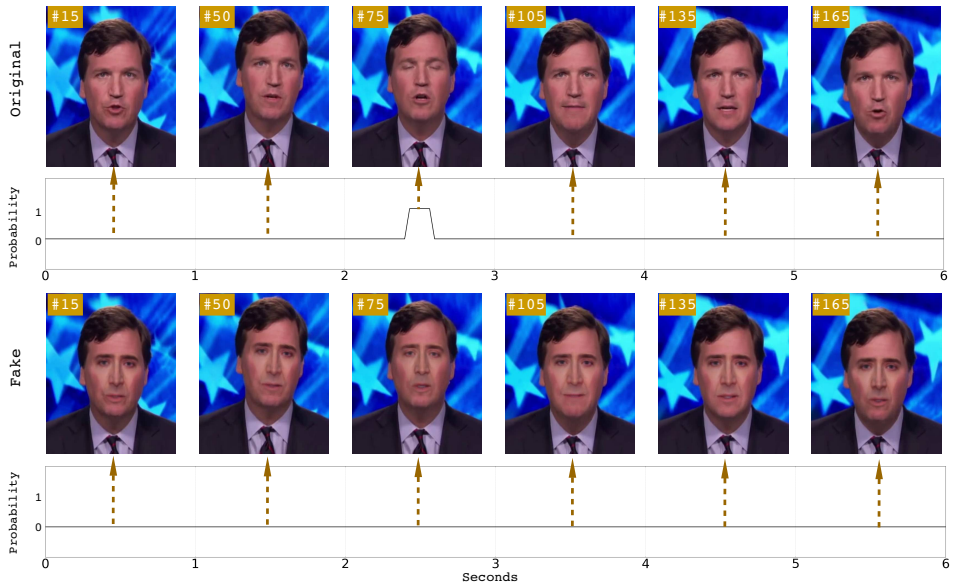

Figure 6. *Example of eye blinking detection on an original video (***top***) and a* `DeepFake` *generated fake video (***bottom***). Note that in the former, an eye blinking can be detected within 6 seconds, while none is detected in the latter.*

mance on detecting videos generated with the DNN-based software `DeepFake`.

There are several directions that we would like to further improve the current work. First, we will explore other deep neural network architectures for more effective methods to detect closed eyes. Second, our current method only uses the lack of blinking as a cue for detection. However, the dynamic pattern of blinking should also be considered – too frequent blinking that is deemed physiologically unlikely could also be a sign of tampering. Finally, eye blinking is a relatively easy cue in detecting fake face videos, and sophisticated forgers can still create realistic blinking effects with post-processing and advanced models trained with more data. Therefore, we will continue explore other types of physiological signals that are intrinsic to a live human but ignored in the AI synthesis methods.

Acknowledgement. This material is based upon work supported by the United States Air Force Research Laboratory (AFRL) and the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-16-C-0166.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[2] T. Azim, M. A. Jaffar, and A. M. Mirza. Fully automated real time fatigue detection of drivers through fuzzy expert systems. *Applied Soft Computing*, 18:25–38, 2014.

[3] A. R. Bentivoglio, S. B. Bressman, E. Cassetta, D. Carretta, P. Tonali, and A. Albanese. Analysis of blink rate patterns in normal subjects. *Movement Disorders*, 12(6):1028–1034, 1997.

[4] Z. Boulkenafet, J. Komulainen, and A. Hadid. Face anti-spoofing based on color texture analysis. In *ICIP*, pages 2636–2640, 2015.

[5] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[6] M. Divjak and H. Bischof. Eye blink based fatigue detection for prevention of computer vision syndrome. In *MVA*, pages 350–353, 2009.

[7] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, pages 2625–2634, 2015.

[8] W. Dong and X. Wu. Fatigue detection based on the distance of eyelid. In *International Workshop on VLSI Design and Video Technology*, pages 365–368, 2005.

[9] T. Drutarovsky and A. Fogelton. Eye blink detection using variance of motion vectors. In *ECCV*, pages 436–448, 2014.

[10] J. Galbally and S. Marcel. Face anti-spoofing based on general image quality assessment. In *ICPR*, pages 1173–1178, 2014.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[13] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[14] W.-B. Horng, C.-Y. Chen, Y. Chang, and C.-H. Fan. Driver fatigue detection based on eye tracking and dynamk, template matching. In *International Conference on Networking, Sensing and Control*, volume 1, pages 7–12, 2004.

[15] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

[16] K. W. Kim, H. G. Hong, G. P. Nam, and K. R. Park. A study of deep cnn-based classification of open and closed eyes using a visible light camera sensor. *Sensors*, 17(7):1534, 2017.

[17] D. E. King. Dlib-ml: A machine learning toolkit. *JMLR*, 10:1755–1758, 2009.

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] H. Li, P. He, S. Wang, A. Rocha, X. Jiang, and A. C. Kot. Learning generalized deep feature representation for face anti-spoofing. *TIFS*, 13(10):2639–2652, 2018.

[20] L. Li, X. Feng, X. Jiang, Z. Xia, and A. Hadid. Face anti-spoofing via deep local binary patterns. In *ICIP*, pages 101–105, 2017.

[21] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *NIPS*, pages 700–708, 2017.

[22] B. Mandal, L. Li, G. S. Wang, and J. Lin. Towards detection of bus driver fatigue based on robust visual analysis of eye state. *IEEE Transactions on Intelligent Transportation Systems*, 18(3):545–557, 2017.

[23] G. Pan, L. Sun, Z. Wu, and S. Lao. Eyeblink-based anti-spoofing in face recognition from a generic webcamera. In *ICCV*, pages 1–8, 2007.

[24] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[25] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, volume 3, page 6, 2017.

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[27] F. Song, X. Tan, X. Liu, and S. Chen. Eyes closeness detection from still images with multi-scale histograms of principal oriented gradients. *Pattern Recognition*, 47(9):2825–2838, 2014.

[28] T. Soukupova and J. Cech. Real-time eye blink detection using facial landmarks. In *21st Computer Vision Winter Workshop*, pages 1–8, 2016.

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[30] H. Steiner, A. Kolb, and N. Jung. Reliable face anti-spoofing using multispectral swir imaging. In *ICB*, pages 1–8, 2016.

[31] F. M. Sukno, S.-K. Pavani, C. Butakoff, and A. F. Frangi. Automatic assessment of eye blinking patterns through statistical shape models. In *International Conference on Computer Vision Systems*, pages 33–42, 2009.

[32] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016.

[33] D. Torricelli, M. Goffredo, S. Conforto, and M. Schmid. An adaptive blink detector to initialize and update a view-basedremote eye gaze tracking system in a natural scenario. *Pattern Recognition Letters*, 30(12):1144–1150, 2009.

[34] Q. Wang, J. Yang, M. Ren, and Y. Zheng. Driver fatigue detection: a survey. In *The Sixth World Congress on Intelligent Control and Automation*, volume 2, pages 8587–8591, 2006.

[35] F. Yang, X. Yu, J. Huang, P. Yang, and D. Metaxas. Robust eyelid tracking for fatigue detection. In *ICIP*, pages 1829–1832, 2012.