Face Forensics on Visual Counterfeits using R-CNN

Mohammad Farukh Hashmi[1], B Kiran Kumar Ashish [2], Ashwik Sagi[3], Avinash G Keskar[4]

[1]National Institute of Technology-Warangal, India

[2]Tericsoft Technology, India

[3]Anurag Group of Institutions, India

[4]Visvesaraya National Institute of Technology-Nagpur, India

mdfarukh@nitw.ac.in ashish@tericsoft.com 16h61a0461@cvsr.ac.in agkeskarr@ece.vnit.ac.in

Abstract

In recent years with the advancements in the Deep Learning realm, it has been easy to create and generate synthetically the face swaps from GANs and other tools, which are very realistic, leaving few traces which are unclassifiable by human eyes. These are known as "DeepFakes" and most of them are anchored in video formats. Such realistic fake videos and images are used to create a ruckus and affect the quality of public discourse on sensitive issues, defaming one's profile, political distress, blackmailing and many more fake cyber terrorisms are envisioned. This work proposes a microscopic-typo comparison of video frames. This temporal-detection pipeline compares very minute visual traces on the faces of real and fake frames using Convolutional Neural Network (CNN) and stores the abnormal features for training. We extract facial landmarks of a total of 512 and compare each and every landmark with the real class. Parameters such as eye-blinking, lip-synch, eyebrows movement, and position, are few main deciding factors that classify into real or counterfeit visual data. The Recurrent Neural Network (RNN) pipeline learns based on these features-fed inputs and then evaluates the visual data. We have trained the network largely videos consisting of their real and fake, collected from multiple websites. Our proposed algorithm and designed network set a new benchmark for detecting the visual counterfeits and show how our system can achieve competitive results on any fake generated video or image.

1. Introduction

DeepFakes have become a new form of altering reality that are spreading very fast than expected [1]. Basically, DeepFakes is a machine learning technology that fabricates or manipulates video and audio recordings to show people doing and saying things that they never did or said. DeepFakes appear authentic and realistic, but they are not [1, 12, 13, 18, 26]. It could be superimposing a face on to a body, so it looks like they are doing something that they have never done [1].

Today, with recent advancements in deep learning, it has become very easy to build a model using the existing photos and videos of a person [18]. Here, people perform face detection on each of these images using various real time face detection algorithms.

To do this, one existing method is to use an auto encoder [27] which is a convolutional network that tries to reconstruct the input image and let's it learn a lower dimensional representation of the input image which will be later used to swap the faces. Specifically, DeepFakes uses one encoder and two decoders. During training it actually trains two networks, both the networks share a common encoder and different decoders. The encoder transforms the input image into a base vector which is a set of numbers that identify of the face. The decoder transforms that vector back into the image. There is an error function which measures how good that transformation works. The model then learns to lower that training error. During training both images are trained with the same encoder but two different decoders. After the network training is done, we can feed a video which is a collection of image frames one frame at a time. Finally, we can concatenate all the images together and watch the video results.

If we can use this kind of passive information, just photos and videos that are out there, that's the key to scaling to anyone as shown in Figure (1) below.

Figure 1. Example of AI-generated visual forgery, the original person's face is given on the top (left) and the target was swapped with the original face given on the bottom (left).

2. Related Work

DeepFakes. At the end of 2017 and the starting of 2018 DeepFakes came out. This is a deep learning approach to swap faces into videos. Comparing the progress of GANs, in four years we have gone from blurry to photorealistic faces. To create a DeepFake you just need a few thousand frames of training data both for the source face and the target face. After the training face you will have a trained swap face network. So, you start with the trained network and then the video you wish to modify, and it might take minutes to hours depending on the length of the video you wish to alter [1, 4, 5, 6, 13]

One of the popular DeepFakes approach uses a machine learning approach known as auto-encoders which has three parts [27].

The first is the encoder and second is the encoding or the bottleneck and the third is the decoder. It's the job of the encoder to take a high dimensional input image and compress it down into a semantically meaningful representation in the encoding and it is the job of the decoder to reconstruct that encoding to make an approximation for the input image. It gives us an unsupervised way to learn a semantically meaningful representation of the data and by comparing the differences between the input and the output data that provides us an error signal which can be used to update the weights in the network during training. Here the encoder is common to both the two faces. So, it shares weights and learns representations and there are two decoders which are trained individually on two people. At the production time, we take person A and we encode them and then we use the decoder for person B to reconstruct the input image that has the effect of swapping the faces across the two videos [19, 26, 27, 29].

Digital Media Forensics. It is the process of identifying, preserving, analysing and presenting digital evidence [1, 17]. Since the 1970's the field of digital forensics has evolved to keep up with the widespread adoption of technology. The use of computers for financial crimes in the 1980's helped shape digital forensics methods with what they are today. Due to the rise of large datasets such as ImageNet [15] which were basically created for further research in computer vision community such as object detection and image classification. It contains millions of manually labelled examples across thousands of classes and then in 2012 a breakthrough in performance in image classification AlexNet [8]. It used a deep learning technique for large scale visual recognition challenge that uses ImageNet. It provided such significant gain in performance that is essentially reshaped the computer vision community. Two years after AlexNet [8] was proposed, Ian Goodfellow [18] published a paper on generative adversarial networks and GANs are broadly applicable because they give new ways to train deep neural networks, they give us ways to generate new synthetic data for training. What's more important is the ability of GANs to generate synthetic faces. In 2016 an approach called face to face was published and this uses real time tracking to monitor both a source and a target face and to transfer expressions across them.
Limitation. The techniques used here are as simple as simple Image processing techniques. The accuracy levels are very low and an impossible task to identify/classify it into "real" or "fake".

Visual Classification. Availability of huge number of datasets on DeepFakes can now be used to train a simple classification model or even objection detection models of RPN or segmentation. Transfer Learning such as InceptionV3 and ResNet models can be used for advance trainings [1, 21]
Limitation. Datasets may be available in large number, but the visual forgery is in microscopic scale and specific parameters work precisely rather than a global feature scaling.

Video Classification. Most of the DeepFake datasets are video type, existing state-of-art methods are using LSTM to classify videos into "real" or "fake" [1, 14, 21, 22, 29]
Limitation. RNN would extract the features from the video frames as a whole where the features get diluted since the whole image frame is given for extraction. This would consume overloading of memory space and computational resources. The overall accuracy is somewhat satisfactory, i.e., *mAP~85%* accuracy.

3. Technical Approach

This paper gives an essence of combining two-novel architecture esteems, i.e., facial landmarks movement and convolutional long short-term memory (Conv-LSTM). Using convolutional drift neural network architecture, the features are extracted from the frames, without saving the frames for memory constraints that requires minimal training to achieve competitive performance on spatio-temporal tasks. LSTM alone as feature extractor and handling temporal data tend to be an expensive task to train. Hence, combining CNN for feature extraction and LSTM for storing the feature vector for frames would be efficient for training such humongous amount of data without any memory expense.

3.1 Facial Landmarks

Given input dataset, i.e., video frames, we run the face detector to locate the face region in the frame *k*. The face parametric model denoted by $S(0)$ represents the mean face shape in the frame.

The landmark vertices on $S(0)$ are projected onto the frame to determine the initial 2D landmark locations as well as their visibility in the frame. The alignment of 2D landmarks on the frame is done via 128 landmark vertices. The initial 2D locations of these landmarks are denoted by $X^{(0)} \in R^{136}$ [1]

In addition, we use a dense set of landmark vertices to evaluate dense face feature descriptor, in order to capture more global and deep insight information of the face and produce more robust results in the face movements. We densely sample the face mesh to obtain 512 landmarks vertices, which includes the previous 128 alignments. The landmark locations are denoted by $U^{(0)} \in R^{320}$ and their visibility in frame is indicated using a binary vector $V^{(0)} \in R^{160}$. During the training, updating landmarks in each frame *k-th* iteration, we evaluate the dense face feature descriptors of the dense landmarks using their current locations $U^{(k)}$ and visibility on the frame $V^{(k)}$. The dense face features descriptors are concatenated into a feature vector $F^{(k)} \in R^{5120}$. For invisible landmarks, such as extreme side-angles, dark pixels, their corresponding alignments/components $F^{(k)}$ are set to zero. The target location defined by $\hat{X}^{(k+1)}$ which improves the accuracy of forgery detection are customised since those are the key parts in detecting the swapping from real ones. The camera parameters or the angle at which the target face is located is given by $w^{(k+1)}$. We adapt the state-of-art [1, 21] approach to determine $w^{(k+1)}$ and $\hat{X}^{(k+1)}$, by computing their displacements proposed as linear function $w^{(k+1)} - w^{(k)}$ and $\hat{X}^{(k+1)} - X^{(k)}$ of dense face feature descriptor $F^{(k)}$.

$$\hat{X}^{(k+1)} = X^{(k)} + R_X^{(k)} F^{(k)} + b_X^{(k)} \quad (1)$$

$$w^{(k+1)} = w^{(k)} + R_w^{(k)} F^{(k)} + b_w^{(k)} \quad (2)$$

where matrix $R_X^{(k)}$ and vector $b_X^{(k)}$ represents a generic descent direction that improves the accuracy of landmarks locations that overcomes the dark and extreme side-angles. The network learns from the training set, where the face is detected in each frame and landmarks are aligned in 2D space from the components, matrix $R_w^{(k)}$ and vector $b_w^{(k)}$.

The 512 landmarks define the movement of each part on the face. This is compared with the fake video, where the actual movement in different scenarios of real video is compared with movement detected of each part in the shammed one using landmarks.

3.2 CNN Feature Extractor

Generally, features are taken by the network and are trained. To reduce the spatial complexity, we are given the particular locations in the frame to the network, where the features are extracted and are trained. CNN feature extractors are build upon transfer learning approach using pre-trained model. This allows the network to leverage the power of pre-trained model combined with custom-trained model for video analysis. Our approach is designed in such a way, after identifying the layer in the source CNN to use as a feature extraction point, rest layers beyond that are removes, leaving behind a feature extractor network and then passing it on to LSTM network. The video data is extracted into frames but not saved for reducing memory complexities. These collection of frame images at frame level, extracts raw abstract face features with high dimensional visual information impeded as feature vector per frame, $u_n^{(v)}$,

$$u_n^{(v)} = (u_1, u_2, ..., u_m) \quad (3)$$

where *n* is the number of frames or frame sequence and *m* is the output dimension of the Deep-CNN feature extractor. Given *v* as unique video identifier for available number of total videos in the training set, features represented as the set of all $u_n^{(v)}$ frame feature vectors used for predicting the sequence $U^{(v)}$, given in set of:

$$U^{(v)} = \{u_1^{(v)}, ..., u_T^{(v)}\}$$

where *T* is the number of frames per video *v*.

3.3 LSTM Pipeline

In LSTM, there is a mean pooling layer to extract frame-level visual features without saving it by holding it temporarily in its memory unit and the output video-level feature vector [10, 16] are fed as an input to the LSTM cell inputs. The feature vectors are shot from the CNN feature extractor from the face embeddings. For each video frame *v*, LSTM memorizes the pattern in current time *t* and then the patterns are erased from the cell and successive patterns in frames in queue will be computed. The previous patterns and correlations are fed in hidden states h_t, internal memory cell state c_t and three gates i_t, o_t, f_t, g_t is a candidate memory cell state or bridge between current input and previous hidden/stored patterns. The computation mathematically is given by:

$$i_t = \sigma(W^i x_t \oplus w_t + U^i h_{t-1} + b_i) \quad (4)$$

$$o_t = \sigma(W^o x_t \oplus w_t + U^o h_{t-1} + b_o) \quad (5)$$

$$f_t = \sigma(W^f x_t \oplus w_t + U^f h_{t-1} + b_f) \quad (6)$$

$$g_t = \tanh(W^g x_t \oplus w_t + U^g h_{t-1} + b_g) \quad (7)$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes g_t \quad (8)$$

$$h_t = o_t \otimes \tanh c_t \quad (9)$$

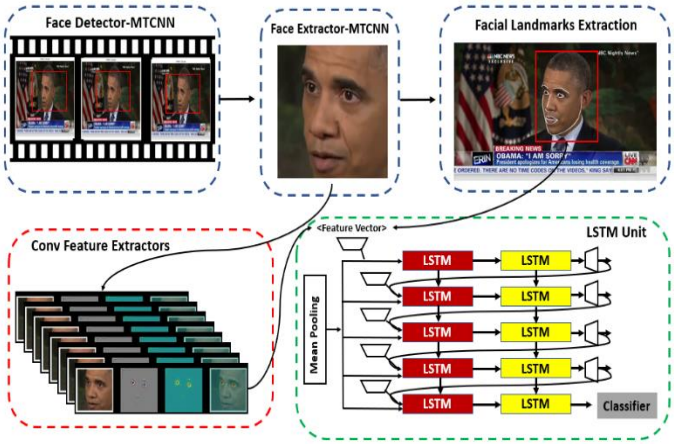

Figure 2: Proposed Architecture Conv-LSTM

where $\oplus$ is a vector concatenation operator, $\otimes$ is the element-wise multiplication between two vectors, W's are the weights of inputs to hidden states, U's are weights from hidden to hidden. All weight matrices and biases b's represent the value of each feature and needs to be trained. We divide LSTM pipeline as two parts:

i. the first part is the comprehensive feature that comes from the feature vector of CNN model that represents the whole frames of a given video *v*. We set the mean pooled frame feature as input of the model of each frame.

ii. the second part is the dividing each video into equal instances of frames and taking their feature vectors under target vector/label *T*. In our case we have two classes: real and fake, we divide the target label into one-hot encoding vectors (y_1, y_2). We finally add embedding layer to squeeze the high dimensional sparse vectors into the lower dimensional sparse vectors of weights $(w_1, ..., w_n)$ for each feature vector. Then the averaged frame feature *x* is duplicated and concatenated with weight vectors w_t, finally input to LSTM model at each time step, as $(x_1 \oplus w_1, ..., x_T \oplus w_T)$. The intermediate hidden layers $(h_1, ..., h_t)$ are outputs of LSTM cells in charge of memory cells of LSTM converging to the final goal state. The conditional probability for each video frame *v* would be:

$$P(y_t, ..., y_1 \mid x_t \oplus w_t, ..., x_1 \oplus w_1) = \prod_{1 \le t \le T} P(y_t \mid h_{t-1}) \quad (10)$$

4. Experiments

Our approach has a 3-phase architectural design, which is designed to reduce the memory constraints on physical GPU servers. AI-generated DeepFake videos are given to training and the reflection is given to testing.

4.1 Network Construction

Facial Landmarks Extraction: First step would be the facial landmarks extraction from each frame *v*. We are extracting *128D* and *512D* landmarks in 2D space and training both of them separately to check the accuracies of both the facial alignments. The initial step would be the face detection the frame $[v_1, v_2, v_3, ...v_n]$. Then the landmarks extractor is applied to extract landmarks and is saved under target label *T*. For detecting face swapping, we mainly consider the movements of ideal parts such as eyes, nose, eyebrows and mouth. Our third experiment is training with these specific parts of both *128D* and *512D* landmarks. The person's movement in "Real" and "Fake" class videos are noted in a *Nd-array* and are sent for training for each frame *v*.

CNN Feature Extractor

The further advance step is the feature extraction from the facial landmarks and their locations. The in-depth insights are extracted using CNN feature extractors alongside with the facial landmark movements for better and precise accuracies, in order to avoid false positives and false negatives. The CNN layers are trained as we will train the features with LSTM as a video itself. CNN layers are used to extract features and give the feature vector to the LSTM cell as input. The image size is set as *(224,224)*. We load the face detection model through *Multi-Cascaded Convolutional Network (MTCNN)* architecture. We initialise the *bottleneck feature extraction* method in the CNN layers with *Global Average Pooling* of the features extracted from the video frame *v*. The features are stored as *Nd-array* and are combined with frame-level feature vector for each frame *v*.

The CNN network starts with *Sequential* model, first and second layers consists of *Conv2D* filters of random number with size *3x3* followed by *ReLu* activation function, with first layer having *Padding* of *'1'*. Based on the number of frames *v*, the *filters* are taken in the hidden layer from here, i.e., *Conv2D* filters based on frames with size *3x3* and *padding '1'* followed by *ReLu* activation

Figure 3: Face detection and cropping for testing model, taken from DeepFake dataset, Google AI

Figure 4: Visualisation of facial landmark extractions of 512 alignments and tracking their movements in each frame

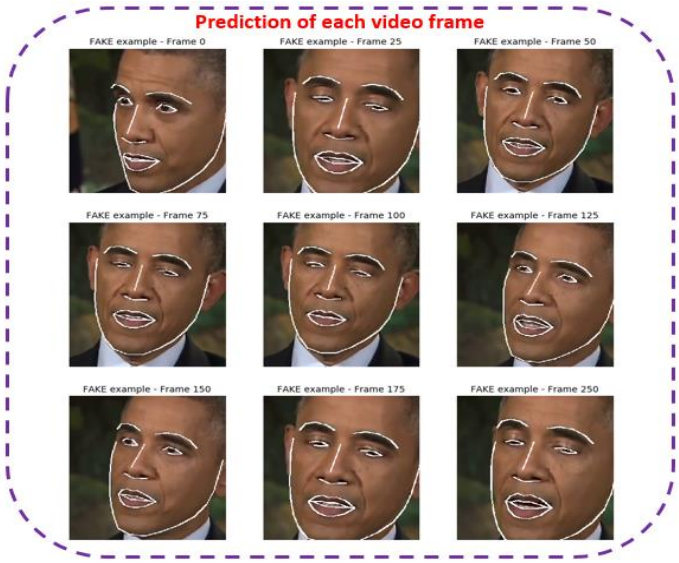

Figure 5: Prediction of each video frame that is extracted in the LSTM network. Prediction is based upon movements of facial parts

function. *MaxPooling2D* is then applied with *pool size* of *2x2*. *Dropout* function is applied to remove unnecessary features for eliminating the memory constraints. The final output layer consists of *Flatten* the features extracted with is fed to *Dense* node of size *256* and *ReLu* activation function. A final *Dropout* is given to clean the features for a final time. This output is fed to the LSTM cell unit.

LSTM Unit

Due to heavy and larger dataset, extracting frames from the videos and given them to the network consumes loads of memory and requires very high computational power. LSTM network works best for video classification and it allows to store the patterns or features without saving the frame in the physical server. It extracts the frames from the videos, extracts the facial landmarks and features but never saves the frame physically. The pattern recorded by the LST memory cell stores the pattern and learns the patterns until next pattern arrives and then the current gets erased from the cell [10, 11, 29]. These patterns are given weights and are stored in the LSTM memory units. The loop continues throughout the video frames *v*.

The LSTM network is built using recurrent neural network functions. Under the *Sequential* model, GRU nodes are initiated of random choice based on the input frames. The *first GRU* node with *return sequences* and *Recurrent Dropout* of *'0.2'* is added, followed by *Dropout* function. This loop is continued throughout the total number of layers in the network. Following the loop of initial layers is the *GRU* node with *Recurrent Dropout* of *'0.2'*, followed by *Dropout* function. This is given as input to *Dense* layers of total *256* with activation function *ReLu*. The final layers consist of *Dense* layers with number of classes, for our case it is *'2'* and with activation function *Sigmoid* since it is binary class. Applying more complex convolutions for CNN and LSTM pipeline for prediction, we now concatenate the features coming down from CNN feature extractor and LSTM for training. *1D-Conv* layers with *ReLu* activation function is initialised for embedded sequence followed by *MaxPooling* x *5* for *1D-Conv*. These are appended and are concatenated with the *1D-Conv* of filter size *'5'* and *ReLu* activation function. A *Dropout* + *MaxPooling* for size *'5'* is applied and appended with the master vector. Now, a *1D-Conv* layer with filter size *'5'* and *ReLu* activation function followed with a *Dropout* and *MaxPooling* x *30* which is flattened into *1024* vector. Now, *Dense* layer with *ReLu* activation function is used followed by *Dropout* which gives *512* vectors. The above layers are appended and concatenated with each other in a loop for each frame v. Finally, *Dense* layer with *ReLu* activation function and *Dropout* with *Sigmoid* of *[0,1]* which gives the prediction.

4.2 Training on DFDC Dataset [2]

The dataset is collected from *Kaggle DeepFake Detection Challenge*. The total dataset consists of *470 GB* of videos which are AI-generated dataset of multiple persons. The whole dataset is given for training to the Conv-LSTM network into 2 classes: *"Real"* and *"Fake"*.

4.3 Results on Google AI DeepFake Dataset [3]

Post training, *Google AI DeepFake* dataset is tested on our trained network. The training almost lasted for 7 days on the humongous data. The Google's dataset are videos too, here we saved each frame to check the framewise results. As we have trained different approaches, our experiment created a new benchmark in detecting visual forgery on videos and set a new state-of-art methods.

Model	Training acc (%)	Validation acc (%)	Test acc (%)
68-landmarks	67.7	69.3	75.1
128-landmarks	98.5	95.8	**97.5**
212-landmarks	91.2	89.0	90.0
512-landmarks	**99.3**	**96.5**	96.3

Table 1: Classification results on our approach of different approaches.

Model	Train. acc (%)	Valid. acc (%)	Test acc (%)
Eye Blinking	37.6	21.4	26.8
Eye Blinking + Eyebrows Movement	44.5	40.3	36.7
Eyes+Eyebrows+Head Movement	61.1	60.4	54.1
Eyes+Eyebrows+ Head+Mouth Movement	**98.5**	**97.6**	**94.3**
Full Face Movements	51.2	45.6	40.2

Table 2: Classification results on different parts of face

5. Conclusion

In this paper, we have designed a benchmark setting architecture *Conv-LSTM* which uses facial landmarks and convolutional features to automatically detect the visual forgery in videos and images. Visual forgery on videos and images can be automatically detected with precise accuracies from our approach. As our architecture was built and trained on humongous amount of data, any visual forgery can be easily detected using our approach which gives the best accuracy when compared to other art-of-states. Our approach is simple and best in terms of usage although the black-box training inside neural nets is much complex, memory constraints are addressed as we are not saving the frames from the videos during training process.

This research has addressed on detecting the visual forgery through convolutional methods, the further scope of this research includes the reverse engineering of AI-generated visual forgery by saving the patterns and features while generating visual forgery and directly training them under forgery class. This would be much complex than expected but opens gate for light-weight model approaches.

References

[1] https://arxiv.org/abs/1707.04045

[2] https://www.kaggle.com/c/deepfake-detection-challenge/data

[3] https://github.com/ondyari/FaceForensics/

[4] Faceapp. https://www.faceapp.com/. (Accessed on 05/29/2018). 1

[5] Fakeapp. https://www.fakeapp.org/. (Accessed on 05/29/2018). 1, 2

[6] The Outline: Experts fear face swapping tech could start an international showdown. https://theoutline.com/post/3179/deepfake-videos-are-freaking-experts-out?zd=1&zi=hbmf4svs. (Accessed on 05/29/2018). 1

[7] M. Abadi et al. Tensorflow: A system for large-scale machine learning. Proceedings of the USENIX Conference on Operating Systems Design and Implementation, 16:265–283, Nov. 2016. Savannah, GA. 1

[8] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *NIPS:2012*.

[9] H. Averbuch-Elor et al. Bringing portraits to life. ACM Transactions on Graphics, 36(6):196:1–196:13, Nov. 2017. 2

[10] P. Bestagini et al. Local tampering detection in video sequences. Proceedings of the IEEE International Workshop on Multimedia Signal Processing, pages 488–493, Sept. 2013. Pula, Italy. 2

[11] M. Brundage et al. The malicious use of artificial intelligence: Forecasting,

prevention, and mitigation. arXiv:1802.07228, Feb. 2018. 2

[12] F. Chollet et al. Keras. https://keras.io, 2015. 1

[13] V. Conotter, E. Bodnari, G. Boato, and H. Farid. Physiologically-based detection of computer generated faces in video. Proceedings of the IEEE International Conference on Image Processing, pages 248–252, Oct. 2014. Paris, France. 2

[14] K. Dale, K. Sunkavalli, M. K. Johnson, D. Vlasic, W. Matusik, and H. Pfister. Video face replacement. ACM Transactions on Graphics, 30(6):1–130, Dec. 2011. 2

[15] Deng. J, Dong. W, Socher. R, Li. Li. K and Fei-Fei. L. ImageNet: A Large-Scale Hierarchical Image Database. *CVPR:2009.*

[16] J. Donahue et al. Long-term recurrent convolutional networks for visual recognition and description. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(4):677–691, Apr. 2017. 2

[17] H. Farid. Photo Forensics. MIT Press Ltd, 2016. 2

[18] I. Goodfellow et al. Generative adversarial nets. Advances in Neural Information Processing Systems, pages 2672–2680, Dec. 2014. Montreal, Canada. ´ 1

[19] D. Guera, Y. Wang, L. Bondi, P. Bestagini, S. Tubaro, and ¨ E. J. Delp. A counter-forensic method for CNN-based camera model identification. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 1840–1847, July 2017. Honolulu, HI. 2

[20] D. Guera, S. K. Yarlagadda, P. Bestagini, F. Zhu, S. Tubaro, ¨ and E. J. Delp. Reliability map estimation for cnn-based camera model attribution. Proceedings of the IEEE Winter Conference on Applications of Computer Vision, Mar. 2018. Lake Tahoe, NV. 2

[21] Guera, David and Edward J. Delp. “Deepfake Video Detection Using Recurrent Neural Networks.” *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* (2018): 1-6.

[22] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Computation, 9(8):1735–1780, Nov. 1997. 2

[23] P. Isola, J. Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5967–5976, July 2017. Honolulu, HI. 1

[24] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. arXiv:1710.10196, Oct. 2017. 2

[25] G. Lample et al. Fader networks: Manipulating images by sliding attributes. Advances in Neural Information Processing Systems, pages 5967–5976, Dec. 2017. Long Beach, CA. 2

[26] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8, June 2008. Anchorage, AK. 4

[27] Y. Liao, Y. Wang, and Y. Liu. Graph regularized autoencoders for image representation. IEEE Transactions on Image Processing, 26(6):2839–2852, June 2017. 2

[28] Y. Lu, Y.-W. Tai, and C.-K. Tang. Conditional cyclegan for attribute guided face image generation. arXiv:1705.09966, May 2017. 2

[29] Z. Lu, Z. Li, J. Cao, R. He, and Z. Sun. Recent progress of face image synthesis. arXiv:1706.04717, June 2017. 2