Video Manipulation Detection via Recurrent Residual Feature Learning Networks

Matthew J. Howard
Computer Science and Engineering
UC Santa Cruz
Santa Cruz, CA, USA
matthoward@ucsc.edu

Alexander S. Williamson
Computer Science and Engineering
UC Santa Cruz
Santa Cruz, CA, USA
alswilli@ucsc.edu

Narges Norouzi
Computer Science and Engineering
UC Santa Cruz
Santa Cruz, CA, USA
nanorouz@ucsc.edu

***Abstract*—**
Over the past decade, the increased adoption of Internet-connected technology has led to a substantial growth in the amount of videos produced and digested by people around the world. Subsequently, manipulated content within videos has become far more common and difficult to detect. Material of this nature poses a significant problem as it provides a way to falsely affect viewers' beliefs. In this work, we first develop a pipeline to generate manipulated video segments from pre-existing videos and then develop a deep learning architecture to detect unique video manipulations on a frame-by-frame basis. Specifically, we develop a model which analyzes manipulated video segments represented as sequences of images via a joint Residual Network (ResNet) feature extractor and Long Short-Term Memory (LSTM) network to detect video frames that exhibit signs of manipulation and classify which type of manipulation was applied. We train our model on a modified subset of the UCF101 Action Recognition Dataset [1] which we alter with a set of four manipulation types: object insertion, compression, frame blackout, and blurring. We evaluate the classification accuracy of our model by creating manipulation "blocks", or sets of consecutive manipulated video frames, varying block length, distance between blocks, and manipulation distribution within blocks. Experimental results demonstrate that our model achieves high (>90%) classification accuracy in the presence of increased frame class transitions.

***Index Terms*—Video manipulation detection, recurrent neural networks, sequence processing**

I. Introduction

As Internet-connected technology becomes more embedded into the lives of millions, so increases the amount of digital media that is digested and disseminated around the world. In particular, the enormous growth of video-sharing and streaming services suggests that an increasing number of individuals gather information from digital video content. Unfortunately, the immense growth and popularity of video media has created an environment well-situated for exploitation. The prospect of "fake" or manipulated videos presents an enormous challenge to preventing misleading and dishonest information from spreading. Further, the implications of manipulated videos ranges from friendly alterations to major modifications that change the nature of the original video, such as manipulations used to hide information (e.g., obscuring objects or persons) or those used to alter the way information is presented (e.g., replacing objects or persons). Additionally, modern manipulation techniques allow for rapid, complex alteration of video content, prompting concern over the reputability of online videos. Considering these characteristics, methods for the detection of video manipulations have become quite desirable; however, work in this domain has thus far been limited. Recent advances in detecting manipulations have focused primarily on single images [4] and facial modifications in videos [2], while advances in video sequence processing have centered around action recognition [6] and object detection [5]. In this work, we seek to initiate development of a method to detect arbitrary video manipulations, and our contribution toward the advancement of these methods is three-fold: 1) an automated data generation pipeline to apply common manipulations to pre-existing video segments, 2) a Residual Network-based [7] recurrent deep learning model that accurately detects manipulations on a frame-by-frame basis, and 3) an experimental evaluation of manipulation detection accuracy across various patterns of manipulation. In the following sections, we first describe our dataset preparation, preprocessing, and modification and then we discuss our proposed model for the task of detecting video manipulations. Finally, we demonstrate performance of our model in classifying various types of manipulations within video segments across several experiments, highlighting advantages and disadvantages of our model.

II. Dataset

In this work, we develop a preprocessing framework for automatically generating manipulated videos from any provided video dataset in parallel with our proposed manipulation detection model. For this paper, we apply our manipulation framework on the UCF101 action recognition dataset for model development, training, and experimentation. In the following sections, we first describe the UCF101 dataset and then describe our preprocessing framework, including details on how we apply specific manipulations to the videos.

A. UCF101

UCF101 [1] is a dataset comprised of 13,320 videos sourced from YouTube depicting 101 types of action categories. We utilize UCF101 for model development and experimentation for several reasons, specifically the videos offer a high degree of variety in camera movement, lighting conditions, and background context, while also capturing realistic and dynamic human scenarios as opposed to staged or static actions.

B. Preprocessing

Several preprocessing steps are taken to ensure that our data is consistent and manageable. During preprocessing, we assume that we are provided a fixed selection of videos to choose from, and apply the following steps: *configuration*, *video parsing*, *frame assignment and manipulation*, and *data preparation*. In *configuration*, we set parameters to determine the number of video segments, video segment length, and manipulation patterns for blocks of manipulations including block length, block spacing, and manipulation distribution across blocks. Parameter configurations are designed to enable experimentation and testing across many manipulation patterns. Next, in *video parsing*, videos are parsed into sequences of images (frames) via OpenCV [9], with each video broken into segments according to the configuration of video segment length. Each frame is scaled to a fixed size of 224x224x3 pixels, and each pixel value is normalized from 0-255 to 0-1. Then, in *frame assignment and manipulation*, each parsed frame within each video segment is assigned a manipulation class from a set of available manipulations according to the configuration of manipulation patterns, then corresponding manipulations are applied individually to each frame. Manipulations are handled by applying individual image transformations on frames according to their assignment. Finally, in *data preparation*, each manipulated video segment is stored as a sequence of pixel-arrays alongside corresponding class labels which we store as one-hot encoded label vectors. Pairs of pixel-array sequences and label sequences are then split into training and validation sets using an 80%/20% separation.

C. Manipulation Details

We construct four unique video manipulations (Figure 1) to apply during data generation. The first manipulation, *black*, converts a video frame into all black pixels and represents a simple, yet common scenario in which an entire frame is occluded from view. The second manipulation, *compressed*, applies a random amount of compression to a frame and represents the scenario where a video loses quality, either intentionally, or by other means, such as a loss in network bandwidth. We implement this manipulation by applying lossy JPEG compression via PIL [10], with quality randomly selected between 1-10%. A range of low quality values are selected to simulate a variety of compression scenarios that noticeably affect the original video segment. The third manipulation, *insert*, inserts an image of an object with a transparent background over a given frame and represents the scenario where an object that was not originally present is added into the scene depicted by the video. We implement this manipulation by layering via PIL a single, transparent object PNG on top of a given frame, selecting randomly from a pool of 20 total PNGs. In order to generalize the scenario further, each PNG is inserted at a random location at a random size (between 30% - 50% of the frame size).

TABLE I
PROPOSED RNN ARCHITECTURE LAYERS.

Layer	Size (Units/%)	Output Shape
Input (Frame Sequences)	$n_{samples}$	$n_{samples} \times 20 \times 224 \times 224 \times 3$
ResNet50	(see [7])	$n_{samples} \times 20 \times 7 \times 7 \times 2048$
GlobalMaxPooling	–	$n_{samples} \times 20 \times 2048$
Dropout	50%	$n_{samples} \times 20 \times 2048$
LSTM	30	$n_{samples} \times 20 \times 30$
Softmax Classifier	$n_{classes}$	$n_{samples} \times 20 \times n_{classes}$

The final manipulation, *blurred*, applies a Gaussian blur to a random rectangular region within a frame and represents the scenario where a region of a video segment is occluded from view. We implement this manipulation by selecting a random rectangular region with 20-50% width and height to that of the full frame, then apply a Gaussian blur via PIL to all pixels within that region.

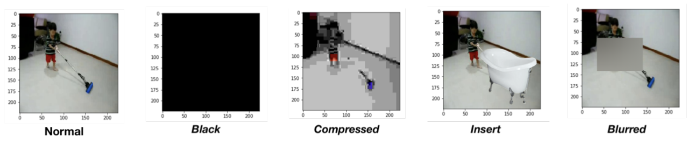

Fig. 1. Manipulated frame examples for *black*, *compressed*, *insert*, and *blurred* manipulations.

III. Models and Algorithms

In this section, describe the proposed DL architecture of our video manipulation detection model.

A. Proposed Network Architecture

In designing our model, we consider the classification power of our feature extraction layers to play a crucial role in the outcome of our predictions. Additionally, we also desire flexibility in our recurrent layers to adjust to the wide variety of video manipulation arrangements that we seek to capture. With these considerations, we propose a modified version of the Long-term Recurrent Convolutional Networks (LRCN) [3] architecture, by taking advantage of the state-of-the-art image classification model ResNet for feature extraction. More specifically, we first construct a feature extraction network that utilizes the architecture of ResNet-50 [7]. Then, we apply global max pooling and dropout regularization [8] on the extracted features to improve the content-agnostic generalizability of our model and reduce overfitting. Finally, we feed the regularized features into an LSTM network to decipher sequential feature dependencies before outputting a classification prediction. Table I summarizes the component layers of our full proposed architecture. For all non-recurrent layers, we utilize a time-distributed approach that shares one set of non-recurrent network weights across all frames within a provided sequence, such that the model is updated with respect to the full sequence of frames during training.

IV. Experimental Results

In this section, we first outline several experiments conducted to test the predictive performance of our video manipulation detection model and discuss our findings.

A. Experiment Design

We conduct five separate experiments that aim to study the impacts of manipulation uniformity, manipulation length, and quantity of manipulations on the temporal predictive classification accuracy of our proposed model. In each experiment, our goal is to measure the predictive classification accuracy of our model on all manipulated frames within a video. Further, in each experiment, we construct a unique arrangement of manipulation block characteristics, as pictured in Figure 2 to examine how block characteristics affect performance. Start position of each initial block is selected at random according to the experimental parameters. In comparison with existing models, we analyze performance of our proposed model against an LRCN baseline for Experiments 1 and 2. The tested LRCN baseline consists of a standard VGG-16 [11] convolutional feature network followed by 50% dropout regularization and a 256-unit LSTM network. For each experiment we randomly select 1000 20-frame video segments to train our model from the UCF101 dataset and apply manipulation modifications as described in Section II, and we employ k-folds cross validation with $k = 5$ to compute average predictive values for all evaluations.

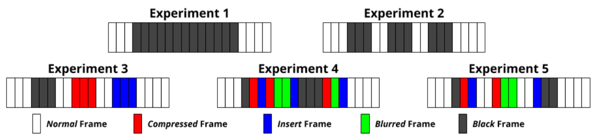

Fig. 2. Example manipulation patterns for experimental configurations.

B. Experiment 1: Uniform Single-Block Single-Manipulation

The first experiment, *uniform single-block single-manipulation*, applies a single, random manipulation to a single block of consecutive frames within each provided sample segment. We vary the length of blocks, and measure the individual accuracy for predicting each type of manipulation as well as the total accuracy across all manipulations. From the results (Table II), we observe a clear decrease in video manipulation classification accuracy for our proposed model as the size of the manipulated block is increased, except in the case of the *compress* manipulation. The results for the *black* manipulation class emphatically demonstrate this trend, starting at 91.7% classification accuracy with a single frame block, dropping to 56.9% accuracy at 40% of the video length, and finally to a meager 19.4% accuracy at 65% of the video length. An explanation for this lies in the amount of non-manipulated context that is available in a video segment. That is, as a larger fraction of the video segment is manipulated, less non-manipulated content is available for the model to decipher between, leading to the model interpreting that the majority of what it sees is in fact the non-manipulated class. In comparison, the tested LRCN baseline exhibits an entirely opposite performance pattern to our proposed model, capturing the *black* manipulation with high degree of accuracy across block sizes, yet failing to classify all other manipulations with any degree of success, though improving with block size overall.

TABLE II
CLASSIFICATION ACCURACIES FOR EXPERIMENT 1 (*Uniform Single-Block Single-Manipulation*) – PROPOSED MODEL (TOP) AND LRCN (BOTTOM).

	Block Size			
Manipulation	1 (5%)	4 (20%)	8 (40%)	13 (65%)
Compressed	0.749	0.960	0.997	0.958
Insert	0.498	0.867	0.667	0.375
Blurred	0.333	0.845	0.517	0.517
Black	0.917	0.600	0.569	0.194
All Manipulated	0.651	0.852	0.731	0.504

	Block Size			
Manipulation	1 (5%)	4 (20%)	8 (40%)	13 (65%)
Compressed	0.000	0.000	0.000	0.159
Insert	0.111	0.000	0.000	0.279
Blurred	0.000	0.000	0.000	0.502
Black	1.000	1.000	1.000	0.942
All Manipulated	0.281	0.252	0.268	0.417

C. Experiment 2: Uniform Multi-Block Single-Manipulation

The second experiment, *uniform multi-block single-manipulation*, applies a single manipulation (selected at random) to multiple fixed-length blocks (with fixed-length spacing) within each provided video segment. We consider configurations on the number of blocks, the distance between blocks, and the size of blocks. From the results (Table III), for our proposed model, we again note a decrease in video manipulation classification accuracy as the size of blocks is increased, especially when block spacing is low. Since all blocks are applied the same manipulation, these trends align with our observations for Experiment 1. Interestingly, the results demonstrate that classification accuracy tends to increase as block spacing is increased, though this trend becomes weaker as the block sizes increase, especially when the total number of manipulated frames (block size $\times$ n_{blocks}) exceeds 50%. In alignment with our discussion for Experiment 1, increasing block spacing introduces more original video context and provides a larger transition buffer between non-manipulated and manipulated frames. In the best-performing case, our model achieves 90.6% classification accuracy for 3 blocks of size 1 and 30% block spacing. In the worst-performing case, our model achieves 32.3% classification accuracy for 5 blocks of size 3 and 5% block spacing. In general, as the video segment becomes less uniform (i.e., more diversity in classes between frames), the accuracy of our model improves while LRCN improves as the segment becomes more uniform.

D. Experiment 3: Uniform Multi-Block Multi-Manipulation

The third experiment, *uniform multi-block multi-manipulation*, applies a single random manipulation to each block. We vary the number of blocks from 2 to 4, the size of blocks from 5% to 35%, and the number of manipulations per video from 2 to 4, fixing spacing at 10%. From the results (Table IV, top left), we report general success in correctly classifying manipulated video frames, achieving above 91% classification accuracy in all

TABLE III
CLASSIFICATION ACCURACIES FOR EXPERIMENT 2 (*Uniform Multi-Block Single-Manipulation*) – PROPOSED MODEL (TOP) AND LRCN (BOTTOM).

	Block Spacing				
Block Size	1 (5%)	3 (15%)	6 (30%)	10 (50%)	15 (75%)
1 (5%)					
$n_{blocks} = 2$	0.823	0.800	0.768	0.842	0.810
$n_{blocks} = 3$	0.793	0.779	0.906	–	–
$n_{blocks} = 4$	0.836	0.877	–	–	–
$n_{blocks} = 5$	0.677	0.771	–	–	–
3 (15%)					
$n_{blocks} = 2$	0.815	0.789	0.824	0.709	–
$n_{blocks} = 3$	0.626	0.747	–	–	–
$n_{blocks} = 4$	0.598	–	–	–	–
$n_{blocks} = 5$	0.323	–	–	–	–
5 (25%)					
$n_{blocks} = 2$	0.681	0.645	0.633	0.653	–
$n_{blocks} = 3$	0.495	–	–	–	–
7 (35%)					
$n_{blocks} = 2$	0.525	0.577	0.562	–	–

	Block Spacing				
Block Size	1 (5%)	3 (15%)	6 (30%)	10 (50%)	15 (75%)
1 (5%)					
$n_{blocks} = 2$	0.211	0.158	0.193	0.175	0.281
$n_{blocks} = 3$	0.214	0.176	0.228	–	–
$n_{blocks} = 4$	0.214	0.229	–	–	–
$n_{blocks} = 5$	0.250	0.232	–	–	–
3 (15%)					
$n_{blocks} = 2$	0.449	0.231	0.278	0.286	–
$n_{blocks} = 3$	0.325	0.420	–	–	–
$n_{blocks} = 4$	0.503	–	–	–	–
$n_{blocks} = 5$	0.658	–	–	–	–
5 (25%)					
$n_{blocks} = 2$	0.468	0.555	0.302	0.277	–
$n_{blocks} = 3$	0.568	–	–	–	–
7 (35%)					
$n_{blocks} = 2$	0.584	0.599	0.508	–	–

experimental configurations. In addition, we notice a mild trend in improving classification accuracy as the number of unique manipulations is increased. Again, this trend can be attributed to the uniformity of the manipulated video segment, as more unique blocks allow our model to discern that the true, non-manipulated video content lies between segments of manipulations. For example, with block size 15%, we achieve 94.4% classification accuracy for 2 blocks of unique manipulations, while achieving 96.9% and 96.2% accuracy for 3 and 4 blocks of unique manipulations.

TABLE IV
TOTAL MANIPULATION CLASSIFICATION ACCURACIES FOR EXPERIMENTS 3 (LEFT, TOP), 4 (LEFT, BOTTOM), AND 5 (RIGHT).

	Number of Manipulations/Blocks		
Block Size	2	3	4
1 (5%)	0.947	0.912	0.952
3 (15%)	0.944	0.969	0.962
5 (25%)	0.956	0.960	–
7 (35%)	0.946	–	–

	Number of Manipulations		
Block Size	2	3	4
4 (20%)	0.991	0.982	0.978
8 (40%)	0.941	0.965	0.978
13 (65%)	0.963	0.970	0.973

	Number of Manipulations		
Block Size	2	3	4
3 (15%)			
$n_{blocks} = 2$	0.822	0.991	0.982
$n_{blocks} = 3$	0.977	0.963	0.972
$n_{blocks} = 4$	0.990	0.978	0.992
$n_{blocks} = 5$	0.952	0.952	–
5 (25%)			
$n_{blocks} = 2$	0.996	0.975	0.967
$n_{blocks} = 3$	0.945	0.982	0.986
7 (35%)			
$n_{blocks} = 2$	0.969	–	–

E. Experiment 4: Random Single-Block Multi-Manipulation

The fourth experiment, *random single-block multi-manipulation*, applies random manipulations to all frames within a single fixed-length block for all video segments. We vary the size of manipulated blocks and the number of manipulations selected from. From the results (Table IV, bottom left), we notice a stark improvement in the single-block setting over the results reported for Experiment 1. Specifically, we achieve greater than 94% classification accuracy in detecting manipulated video frames in all tested scenarios. Although this scenario is unrealistic in practical settings, it again highlights the strength of our model in detecting video manipulations when there is high variability between frames. The model excels at classifying manipulated frames even when a large number of frames are manipulated, such as in the case of 95% block size where the uniformity argument again holds, as accuracy increases from 96.6% when 2 manipulations are assigned to the block to 98.7% and 98.9% when 3 and 4 manipulations are assigned, respectively.

F. Experiment 5: Random Multi-Block Multi-Manipulation

Finally, the fifth experiment, *random multi-block multi-manipulation*, applies a random manipulation to each frame within a sequence of fixed-length repeating blocks that are separated by a fixed distance. We vary the quantity of manipulated blocks from 2 to 5, block size from 15% (3 frames) to 35% (7 frames), and the number of unique manipulations from 2 to 4, fixing spacing at 10%. From the results (Table IV, right), we observe high-performing results similar in nature to those in Experiment 4, achieving accuracies above 94.5% in all tested scenarios, except for an 82.2% accuracy result for 2 blocks of size 15% and 2 manipulations. The lowest accuracy is likely a result of uniform blocks generated by the randomization process leading to a increase in uniformity, as we see highly accurate predictions when 2 manipulations are used at higher block sizes that are less likely to result in uniform manipulation segments. Once more, the negative trend in accuracy seen in experiment 2 for our proposed model becomes non-existent as manipulations are varied within blocks themselves.

V. CONCLUSION

In conclusion, we have successfully developed a framework that modifies existing video datasets with select manipulations applied to the original videos for use in training DL-based models to detect video manipulations. Additionally, we have prototyped a model combining the state-of-the-art ResNet classification network with recurrent LSTM networks that performs accurately at classifying manipulations on a frame-to-frame basis. In comparison with a VGGNet-based LRCN baseline model, our model vastly outperforms the baseline in correctly classifying manipulated frames, achieving 65.1%, 85.2%, 73.1%, and 50.4% accuracy for block sizes of 5%, 20%, 40%, and 65%, respectively, in Experiment 1, as opposed to 28.1%, 25.2%, 26.8%, and 41.7% accuracy for the baseline. Furthermore, for Experiments 3, 4, and 5, our proposed model achieves an average prediction accuracy of 96.3%, with the highest overall accuracy attained at 99.6%. Our experimental pattern evaluation provides considerations for designing manipulation detectors for more complex alterations.

References

[1] Soomro, Zamir, and Shah. “UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild.” https://arxiv.org/abs/1212.0402. https://www.crcv.ucf.edu/data/UCF101.php. University of Central Florida. Accessed March 2019.

[2] Guera and Delp. “Deepfake Video Detection Using Recurrent Neural Networks.” https://engineering.purdue.edu/ dgueraco/content/deepfake.pdf. Purdue University. Accessed June 2019.

[3] Donahue, Hendricks, Rohrbach, Venugopalan, Guadarrama, Saenko, and Darrell. “Long-term Recurrent Convolutional Networks for Visual Recognition and Description.” https://arxiv.org/pdf/1411.4389.pdf. UC Berkeley UT Austin UMass Lowell. Accessed March 2019.

[4] Zhou, Han, Morariu, and Davis. ”Learning Rich Features for Image Manipulation Detection.” https://theblog.adobe.com/spotting-image-manipulation-ai/. University of Maryland, College Park Adobe Research. Accessed March 2019.

[5] Ren, He, Girshick, and Sun. ”Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” https://arxiv.org/pdf/1506.01497.pdf. University of Science and Technology of China Microsoft Research Facebook AI. Accessed May 2019.

[6] Simonyan, Karen, and Andrew Zisserman. ”Two-stream convolutional networks for action recognition in videos.” Advances in neural information processing systems. 2014.

[7] He, Kaiming, et al. “Deep residual learning for image recognition.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[8] Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting.” https://www.cs.toronto.edu/ hinton/absps/JMLRdropout.pdf. University of Toronto. Acessed May 2019.

[9] Bradski, G. (2000). The OpenCV Library. Dr. Dobbs Journal of Software Tools.

[10] Clark, A., et.al, Pillow: Python Imaging Library (Fork), https://pillow.readthedocs.io. Accessed May 2019.

[11] Simonyan, Karen, and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition.” arXiv preprint arXiv:1409.1556 (2014).