Detecting Video Speed Manipulation

Brian C. Hosler and Matthew C. Stamm
Drexel University
Philadelphia PA
{brian, mstamm}@drexel.edu

Abstract

Manipulated videos are frequently an important part of misinformation campaigns. While much attention has recently focused on sophisticated threats such as deepfakes, the majority of videos used in misinformation campaigns have been created using relatively simple manipulations that can be produced by commonly available editing software. One important manipulation that has been used in previous misinformation attempts is altering the speed of a video. In the previous 18 months, widely circulated videos have had their speed manipulated to make Speaker of the House Nancy Pelosi appear disoriented, and to make reporter Jim Acosta appear to act aggressively toward a White House staffer. Currently, however, there are no approaches to accurately detect video speed manipulation that can be deployed at scale. In this paper, we propose new algorithms to detect video speed manipulation and to estimate the rate by which a video's speed has been modified. To do this, we identify a new trace left by video speed manipulation and show how it can be extracted from a video. Our approaches to trace extraction, speed manipulation detection, and manipulation rate estimation are computationally efficient and can be run in a matter of milliseconds. We present experimental results that show that our proposed approach can detect manipulated videos with up to 99% accuracy.

1. Introduction

Videos have become a integral part of modern communication. Websites, such as YouTube and Facebook, as well as apps like Instagram and Twitter, allow users to instantly share videos with others across the world. However, is becoming increasingly easier to edit videos. Particularly, it is easy for some users to create maliciously edited videos. As a result, fake videos and misinformation are shared faster than they can be verified. This is causing the authenticity of many videos to be called into question [28, 4].

This work is supported by the National Science Foundation under Grant No. 1553610, as well as the Army Research Office and Defence Forensic Science Center under Cooperative Agreement Number W911NF-17-2-0087. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors, and do not reflect those of the National Science Foundation, the Army Research Office, or the Defence Forensic Science Center.

Recently, Deepfakes have emerged as a new threat, garnering attention from both researchers and the news media [17]. Through the use of deep learning techniques like Generative Adversarial Networks, an attacker can artificially create a visually realistic video of a target by swapping the face in one video, with another face [23, 3, 25]. In response, several methods have been developed to detect and combat these deepfake videos [30, 9, 18, 2]. Deepfakes are a very powerful and dangerous technology, however, their use is still limited. Creating fake videos usually requires a skilled attacker, and most deepfake algorithms also require a large amount of data, including images and videos, of the target.

While much research is targeted at these advanced techniques, longstanding, simpler techniques are left unchecked, with no means of detection. Less advanced video editing manipulations such as cropping, splicing, and speed adjustment can still result in effective attacks. These attacks can be performed by most video editing software.

One such attack, video speed adjustment, has been used in two separate instances over the last 2 years [11, 10]. A video of Speaker of the House Nancy Pelosi was shared which appeared to show he slurring her words and acting strangely [11]. It was not until later that this video was confirmed to be manipulated. The video and audio were artificially slowed, resulting in the speaker's apparent inebriation. Before this, a video of news reporter Jim Acosta appeared to show the man acting aggressively toward a staff member at a press conference [10]. In reality, parts of an authentic video of the exchange were sped-up and slowed down causing Acosta to appear as an aggressor. This video resulted in the revocation of Acosta's press pass, even after the video was declared fake by multiple sources.

The manipulation in both of these instances was the same; the speed of the events the video was changed, resulting in the altered perception when viewing. To our knowledge, there are no existing techniques to automatically detect speed manipulation in H.264 encoded video.

In this paper, we propose a method to quickly detect speed-adjusted videos encoded with the most common modern codec. First, we identify a trace that can be quickly extracted from an encoded video. We then propose a method to detect manipulated videos using this trace. We also propose a method of estimating the amount that a video's speed has changed. We perform experiments which demonstrate that our system can reliably detect manipulated videos with up to 99% accuracy. We also present a case study of a video which was used in a speed manipulation attack, and demonstrate that our system is capable to detecting this real-world threat.

Our contributions are as follows:

- We identify a trace that can be used to identify speed-adjusted videos without decoding. This trace can be extracted from a video in milliseconds, making it useful for processing videos at scale.
- We develop a model of this trace for both original and manipulated videos. This model offers intuition on the origin of the trace, so that it is accessible and creates a clear foundation for further work.
- We develop algorithms to both detect manipulated videos and estimate the parameter of their manipulation.

2. Related Work

To our knowledge, there is little existing work which aims to detect speed manipulation in modern video codecs. Previous work by Wang and Farid has shown that speed manipulation in interlaced and deinterlaced video can be detected via differences in the top and bottom fields within a frame [32]. Kobla *et al*. proposed a method for detecting slow motion replays in sports videos via the macroblock types and motion vectors [16]. Both of these methods however are designed for MPEG-1 and MPEG-2 encoded videos, which are not common today.

Some closely related work involves detecting the deletion of one or more consecutive frames of a video. Wang and Farid first proposed a method of detecting deleted frames in MPEG-1 and MPEG-2 encoded videos through analysis of the frame prediction residual [31]. This method was later improved upon by Stamm *et al*. [27]. Other techniques for detecting deleted frames involve tracking visual properties across multiple frames [26, 8, 33, 34]. Many visual-based methods for detecting frame deletion, also serve to detect frame insertion [8, 33, 34].

Many other video forensics works focus on different aspects of videos, such as identifying the device which captured a video [20, 29, 4], as well as its make and model [12, 13]. Some works have attempted to detect re-encoding as a form of manipulation [14, 31]. Content insertion is another way in which researchers identify manipulated videos, either by measuring the integrity of a frame's noise pattern [19, 15], or measuring the consistency of forensic traces within a frame [7, 21].

3. Background

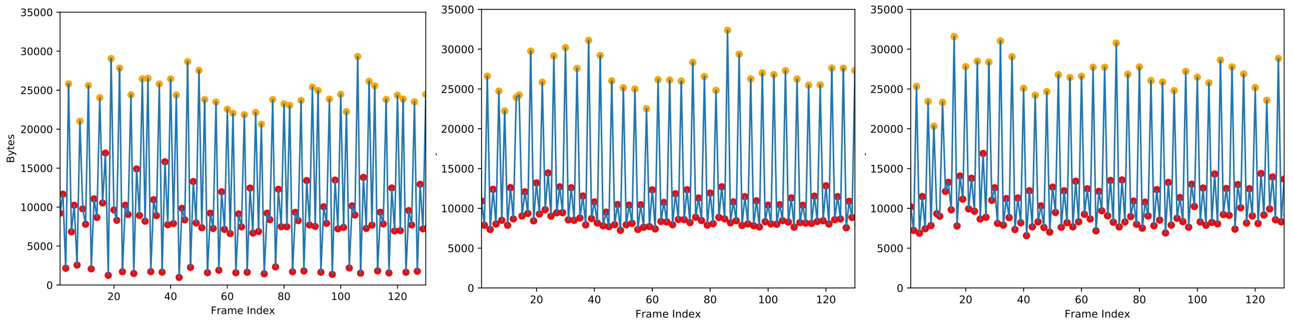
In order to justify our method, we first offer background on modern video coding techniques.

Frame Coding This work is based on the most popular modern video codec, known as AVC or H.264. This codec, and others like it, take advantage of the temporal and spatial redundancy in a video in order to reduce the number of bits to be stored or transmitted. A single frame or picture of a video is made up of one or more slices, which are then divided into macroblocks. Each slice will also have an associated slice type, which determines the way in which the macroblocks can be coded. In practice, it is common for one frame to correspond to a single slice; in this work we use frame and slice interchangeably, as well as frame type and slice type.

Frames are either I, P, or B frames. I frames are coded independently, so that they can be decoded without decoding any other frames. Macroblocks in P frames can be predicted, meaning they can be encoded using pixels from previous frames in the video. This greatly reduces the number of bytes needed to encode the frame, however these savings are dependent on how similar the frames are. In general, the further apart frames are in a sequence, the less similar they are. B frames, similar to P frames, can contain blocks which are predicted. Unlike P frames, the blocks in B frames can be encoded using pixels from both past and future frames. However, in order to use pixels from future frames, the frames must be encoded out of order.

Effects of Coding Order It is common coding practice to employ hierarchical B frame coding. To explain hierarchical coding, first consider five consecutive frames, $V(0)$, $V(1)$, $V(2)$, $V(3)$, and $V(4)$, where frame $V(0)$ has been previously been encoded. An encoder will next encode $V(4)$ as a P frame, using $V(0)$ (and earlier frames) as a source of prediction. Next, $V(2)$ will be encoded as a B frame, using $V(0)$ and $V(4)$ for prediction. Finally, frames $V(1)$ and $V(3)$ will be coded, also as B frames, having all other frames for prediction.

As a result of this process, $V(4)$ is likely to need the largest number of bytes to encode. Frame $V(2)$ will require fewer bytes, because it can be predicted from two frames, both only two time units away. Frames $V(1)$ and $V(3)$ will need the fewest number of bytes to encode, because they both have two adjacent frames from which to predict. One side effect of this, assuming approximately constant motion, is that $V(1)$ and $V(3)$ will be nearly the same size. If



(a) SMP = 0.8 (b) Original (c) SMP = 1.25

Figure 1: Size (number of bytes) of encoded frames of an original video, and two manipulated copies. Red points indicate B frames and orange points indicate P frames. I frames are not shown.

a frame is repeated, *i.e*. zero motion, the number of bytes needed to encode one of those frames will be near-zero.

4. Trace Origin

When the speed of a video is altered using video editing software, a new video sequence is created whose frames correspond to different times than the frames in the original video. While it is possible to interpolate between two consecutive frames to create the new sequence, this approach is extremely computationally expensive, and is not adopted in practice. Instead, editing software will periodically duplicate or delete frames from the original sequence, to achieve a result that is perceptually similar to a viewer but at a significantly decreased computational cost.

To systematically characterize this, we define the speed manipulation parameter (SMP) ρ as the ratio of an edited video's speed to that of its original counterpart. The SMP of a video is not a measure of its frame rate, it is a measure of the relative passage of time in the manipulated video, as compared to the original. For example, an item moving across the frame at one speed in an original video, will move at speed ρ times faster in a manipulated video. Videos with an SMP less than one are slowed down, while those with an SMP greater than one have had their speed increased.

We can characterize the relationship between frames in an unaltered video sequence $V(n)$ and its manipulated counterpart $\tilde{V}(n)$ according to the equation

$$\tilde{V}(n) = V(\text{round}(\rho n)). \tag{1}$$

When $\rho < 1$, *i.e*. the video is slowed down, it can be shown that two consecutive frames in $\tilde{V}$ will correspond to a single frame in V. When this frame sequence is re-encoded, the second frame to be re-encoded will be very efficiently predicted, and therefore have a very small encoded frame size. Furthermore, this will happen periodically, with period determined by ρ. This will result in a very distinct trace that can be used to detect video speed manipulation.

To show this trace, we define the encoded frame size (EFS) sequence $s(n)$ as the number of bytes which are used to encode frame n of a given video. The EFS sequence can be extracted from a video by writing simple code to read these values from the video's bitstream.

When frames are repeated due to slowing down a video, the value of $s(n)$ will suddenly drop to near zero. This effect can be clearly seen in Figure 1. Figure 1b shows the EFS sequence of an unmanipulated video, while Figure 1a shows the EFS sequence of a video that has been slowed using an SMP of $\rho = 0.8$. In Figure 1a, we can distinctly see that the frame size periodically drops to near zero in the EFS sequence of the slowed down video. These periodic drops are video speed manipulation traces that occur when $\rho < 1$. We note that these are distinct from the natural variations in the size of B-frames due to coding order effects described in Section 3.

A similar, though less visually distinct effect can be observed when $\rho > 1$, *i.e*. the video has been sped up. Speeding up the video according to (1) will cause certain pairs of consecutive frames in $\tilde{V}$ to map to frames which are not consecutive in V, *i.e*. some frames will be skipped. When this occurs, the true time difference between these consecutive frames in $\tilde{V}$ will be greater than the true time difference between other consecutive frames in $\tilde{V}$. As a result, the later of the two frames will be less predictable and will require more bytes to encode. This will also occur in a periodic fashion, with the period determined by ρ.

When examining the EFS sequence $s(n)$ taken from a sped up video, the phenomenon described above will result in periodic increases in $s(n)$ that are distinct from those caused by the coding order effects described in Section 3. These periodic increases in $s(n)$ are video speed manipulation traces that occur when $\rho > 1$.

An example of video speed manipulation traces in a sped up video can be seen in Figure 1c, which shows the EFS sequence of a video that has been sped up using an SMP of $\rho = 1.25$. Here, we can observe that these speed manipula-

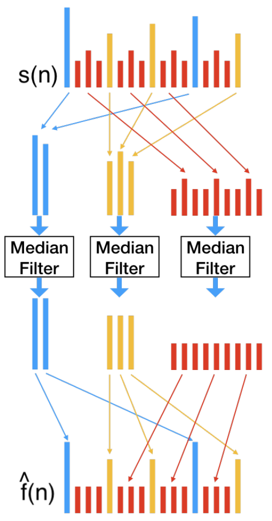


Figure 2: Estimation of trace-free EFS sequence, $\hat{f}(n)$ from the observed sequence, $s(n)$.

tion traces are much less visually distinct than those cause by slowing a video. While they are less visually detectable, we will show in subsequent sections that they can be accurately captured using well designed speed manipulation detection features.

5. Proposed Method

In this section, we provide a detailed discussion of our approach to detect video speed manipulation and to estimate the speed manipulation parameter ρ. To accomplish this, we first propose a set of features that can capture speed manipulation traces. We then discuss how these features are used by our speed manipulation detection and estimation systems.

5.1. Feature Extraction

In order to detect speed manipulation, or to estimate the speed manipulation parameter ρ, we need to extract a set of features from the EFS sequence $s(n)$ that encodes information about the presence and nature of any speed manipulation traces.

We can view the speed manipulation traces as a sequence $u(n)$ that modulates the EFS sequence of an unaltered video. These traces can be viewed as an impulse train that either periodically decreases the value of the EFS sequence to near zero in the case of slowed down videos, or periodically increases its value in the case of sped up videos.

Isolating speed manipulation traces can be challenging however, due to other variations that occur within $s(n)$. For example, the size of each frame depends on its frame type: I-frames require substantially more bytes to encode than P-frames, which themselves are larger than B-frames. As a result, the value of $s(n)$ varies substantially depending on the frame type of the n^{th}, as can be seen in Figure 1. Furthermore, the value of $s(n)$ varies with the motion of the video as well as the content of the scene being captured.

We wish to reduce the effects of these variations when capturing traces left by speed manipulation. To understand how we do this, let us first assume that $s(n)$ is the EFS sequence of a video whose speed has been manipulated. We can model the EFS sequence extracted from the video as

$$\begin{aligned} s(n) &= f(n)\big(1+u(n)\big)+\epsilon_m(n), \\ &= f(n)+u(n)f(n)+\epsilon_m(n), \end{aligned} \tag{2}$$

where $f(n)$ is the ideal frame size sequence that is free from any manipulation traces and $\epsilon_M(n)$ is model noise.

We note that we use this multiplicative model for the fingerprint because its effects are to scale $s(n)$ to near zero if the video is slowed, or to proportionally increase the value of $s(n)$ if the video is sped up. An additive fingerprint model would correspond to a constant increase or decrease in $s(n)$, which is not what we have observed when examining speed manipulation traces.

Next, we assume we can create some estimate $\hat{f}(n)$ of the sequence $f(n)$ such that

$$\hat{f}(n) = f(n) + \epsilon_e(n) \tag{3}$$

where $\epsilon_e(n)$ is noise due to estimation error.

We can obtain a residual signal $r(n)$ that contains the speed manipulation fingerprint $u(n)$ while also largely suppressing components of $s(n)$ that may interfere with us capturing $u(n)$ such that

$$\begin{aligned} r(n) &= s(n) - \hat{f}(n), \\ &= f(n)+u(n)f(n)+\epsilon_m(n)-f(n)-\epsilon_e(n), \\ &= u(n)f(n)+\epsilon(n), \end{aligned} \tag{4}$$

where the separate noises $\epsilon_e(n)$ and $\epsilon_m(n)$ have been grouped together into a single noise term $\epsilon(n) = \epsilon_e(n) - \epsilon_m(n)$.

Alternatively, if the speed of the video has not been manipulated, the EFS can be modeled simply as

$$s(n) = f(n) + \epsilon_m(n). \tag{5}$$

As a result, the residual signal becomes

$$r(n) = \epsilon(n). \tag{6}$$

We can see from examining (4) and (6), that the residual signal is more suitable for searching for speed manipulation traces than directly examining the EFS sequence. If the video's speed has been altered, then $r(n)$ is a noisy version of the periodic speed manipulation trace. By contrast, if the video is unmanipulated then $r(n)$ is simply noise.

In order to obtain the residual, however, the estimated signal $\hat{f}(n)$ must be produced from $s(n)$. To accomplish this, we leverage the fact that speed manipulation traces are temporally isolated changes in the EFS sequence.

To produce $\hat{f}(n)$, we first split $s(n)$ into three subsequences corresponding to their frame type: $s_I(k)$ consisting only of I-frames, $s_P(k)$ consisting only of P-frames, and $s_B(k)$ consisting only of B-frames. We intentionally index these subsequences by a different variable k since $s_\ell(k)$ is the size of the k^{th} ℓ-frame, which is not the k^{th} entry of s. When creating these subsequences, we create three indexing functions $i_\ell(k)$ where $\ell \in \{I, P, B\}$ that relate the k^{th} entry of $s_\ell(k)$ to the n^{th} entry of $s(n)$, such that $s_\ell(k) = s(i_\ell(k))$.

As a result, each subsequence $s_\ell(k)$ of a manipulated video can be expressed as

$$\begin{aligned} s_\ell(k) &= f_\ell(k)\big(1 + u(i_\ell(k))\big) + \epsilon(i_\ell(k)) \\ &= f_\ell(k) + u'(k) f_\ell(k) + \epsilon'(k) \end{aligned} \tag{7}$$

where $u'(k) = u(i_\ell(k))$ and $\epsilon'(k) = \epsilon(i_\ell(k))$. Similarly, each subsequence in an unaltered video can be expressed as

$$s_\ell(k) = f_\ell(k) + \epsilon'(k). \tag{8}$$

We can then obtain an estimated version of each subsequence $f_\ell(k)$ by median filtering the subsequence $s_\ell(k)$

$$\hat{f}_\ell(k) = \text{med}_w(s_\ell(k)) \tag{9}$$

where w is the size of the median filter window and $\text{med}_w(s_\ell(k)) = \text{median}\{s_\ell(k - \frac{w-1}{2}), \ldots, s_\ell(k + \frac{w-1}{2})\}$. Because $u'(k)$ is a sparse signal and because $\epsilon(k)$ is noise, median filtering will remove these terms from (7) and (8).

After $\hat{f}_I$, $\hat{f}_P$, and $\hat{f}_B$ are individually estimated, these subsequences can be recombined to form the estimated sequence $\hat{f}$. We then calculate the residual signal $r(n)$ using (6). The formation of $\hat{f}$ from $s(n)$ is shown in Figure 2.

To create a low dimensional feature set that captures salient information about the presence and nature of speed manipulation traces, we build an N^{th} order auto-regressive (AR) model of the residual sequence, such that

$$r(n) = \sum_{i=1}^{N} a_i r(n - i) + \xi(n) \tag{10}$$

where $a_1, \ldots, a_N$ are the AR model coefficients and $\xi(n)$ is white noise.

We then group the AR model parameters to form the feature vector $\boldsymbol{\phi}$, defined as

$$\boldsymbol{\phi} = [a_1, \ldots, a_N]^\intercal, \tag{11}$$

which we use to perform speed manipulation detection and to estimate the parameter ρ. We note that we do not normalize the residual by $\hat{f}(n)$ before building the AR model to avoid amplifying components of $\epsilon(n)$ where $s(n)$ is relatively small.

5.2. Speed Manipulation Detection

To detect speed manipulated videos, we view this problem as a binary classification problem in which once class corresponds to unmanipulated videos and the second class corresponds to speed manipulated videos. We discriminate between these classes using a support vector machine trained to perform classification using the AR model parameter features $\boldsymbol{\phi}$ defined in (11). When performing classification, we utilize a radial basis function (RBF) kernel.

5.3. Speed Manipulation Parameter Estimation

Once speed manipulation has been detected in a video, an investigator may wish to estimate the speed manipulation parameter ρ. We estimate ρ by formulating this problem as a regression problem on the basis of the feature vector $\boldsymbol{\phi}$ defined in (11). Each manipulated video in a training set is given a label according to its SMP value (i.e. ρ) and each unmanipulated video is given the label '1'. Next, we train an SVM to perform support vector regression (SVR) on the basis of $\boldsymbol{\phi}$. Once trained, this SVR can be used to obtain an estimate of the SMP $\hat{\rho}$ of a video such that $\hat{\rho} = \text{SVR}(\boldsymbol{\phi})$.

6. Experimental Results

To evaluate the performance of our proposed system, we conducted a series of experiments. First, we created a dataset of manipulated videos using different speed manipulation parameters. Next, we evaluated the baseline performance of our proposed speed manipulation detection algorithm and examined the effect of the AR model order on the algorithm's performance. After choosing an appropriate AR model order, we conducted more detailed experiments to assess the system's performance at different SMP values, as well as our algorithm's ability to accurately estimate the SMP parameter of a manipulated video. Finally, we demonstrated our system's ability to operate on organic videos – i.e. manipulated videos created using consumer oriented software that an attacker would likely use – and performed a case study on the manipulated video of Speaker Nancy Pelosi discussed in the introduction.

6.1. Dataset

We used 414 videos from the Deepfake Detection Challenge dataset [1] to create training and testing data for these experiments. To do this we first randomly selected one of the 50 partitions of the DFDC training data (partition 35) and used only the videos labeled 'REAL.' These videos are all 10 seconds in length and encoded according to the H.264/AVC standard, using the mp4 file format. Additionally, all videos have similar encoding parameters including, CABAC entropy coding, a keyframe interval of 250 frames, and a maximum of 3 B frames between P frames.

We then created manipulated copies of each of these video at several different speed manipulation parameters using FFmpeg [6]. We selected speed manipulation parameters between 0.6 and 1.4, at an interval of 0.05, and created a manipulated version of each video using each parameter. These parameters were chosen to span a reasonable range of manipulations; outside of this range the manipulation is likely to be detectable by the human eye. Manipulated videos were created using the FFmpeg command

```
ffmpeg -i $VIDEO -filter:v "setpts=$RATE*pts" $NEW_VIDEO
```

where $VIDEO and $NEW_VIDEO are the names of the real and manipulated videos, and $RATE is equal to one divided by the chosen speed manipulation parameter.

Our final dataset of 7038 videos is composed of 414 unmodified videos, and 6624 manipulated videos.

6.2. Manipulation Detection

Our system, as described in Section 5.1, first creates an AR model of the residual sequence. To do this, we wrote software to parse the mp4 file format and report, in display order, the frame type and the number of bytes used to encode each frame. Using this, we extracted EFS sequences from all videos in our dataset, and normalized them, as discussed in Section 5.1. We then used the *statsmodels* python library [24] to fit an n^{th} order AR model to each sequence. The model tap weights are then used as features to train an SVM classifier using the *sklearn* python library [22].

Initially, we used our dataset to train our system to identify if a video's speed has been manipulated, where all manipulated videos belong to a single class, and all original videos belong to another. We varied the number of coefficients in our system's AR model to study the effect that the model order has on classification accuracy. In this experiment, we considered AR models with between 4 and 30 coefficients. For each model order, we measured the detector's accuracy and AR model fitting time using 10-fold cross validation with a 90/10 train/test split. Because of the natural class imbalance, the accuracy was measured as the average of the true positive rate and the true negative rate. Figure 3 shows the detection accuracy and average AR model fit time for each model order.

As shown in Figure 3, our system achieves a maximum accuracy of 90% with a model order of 27. Beyond this, the accuracy of the system begins to decrease. Figure 3 also shows that the AR model fit time appears to grow linearly with the model order. This shows that our system is capable of rapidly detecting manipulated videos with high accuracy. Figure 3 shows that beyond a model order of 21, our system shows diminishing returns in its accuracy. For this reason, we use a 21st AR model for the test of our experiments.

We repeated this experiment to produce a Receiver Operator Characteristic (ROC) curve of our system using 21st order AR model coefficients as features, as shown in Figure 4. Figure 4 shows that our detector achieves a detection accuracy of 91.7% with a 9.04% probability of false alarm, and has an area-under-the-curve of over 0.96. This curve shows that our proposed system is a powerful tool for detecting manipulated videos.

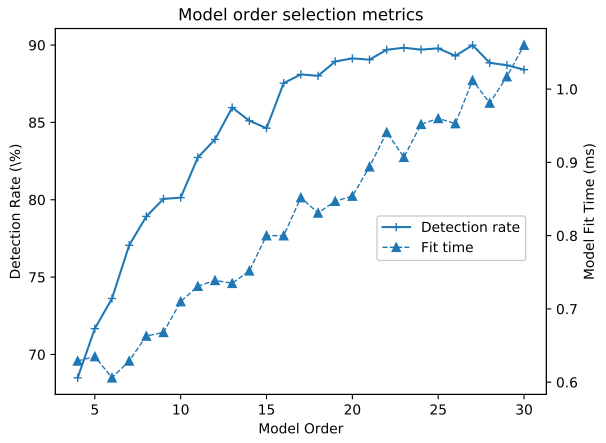


Figure 3: Detector accuracy and average AR model fit time for systems with varying model order.

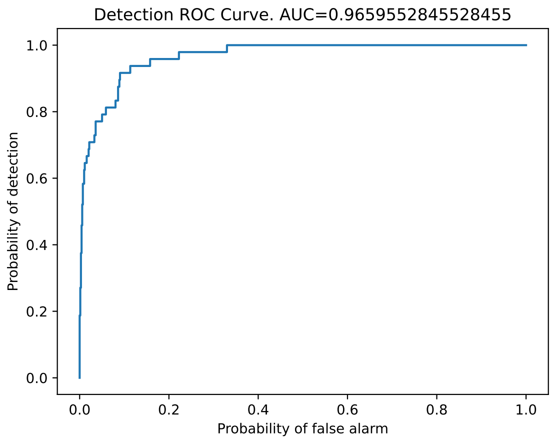


Figure 4: ROC curve of detector using 21st order AR model weights as features.

6.3. Effective Range of Operation

In our next experiment, we studied the efficacy of our system when confronted with different ranges of speed manipulation. For this experiment, we defined a reasonable range of speed manipulation parameters between 0.6 and 1.4. For each SMP in this range, we trained our system to discriminate between an original video, and a video modified using the given SMP. We did this by isolating the 414 original videos and the 414 manipulated videos of the SMP, creating a training set and a testing set, and training our proposed system. We calculated the accuracy of our sys-

Table 1: Classification accuracy of our system when considering single manipulation parameters.

Speed Manipulation Parameter (Speed Decrease)							
0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
98.8%	98.1%	99.1%	96.4%	97.0%	97.9%	97.9%	94.1%

Speed Manipulation Parameter (Speed Increase)							
1.05	1.1	1.15	1.2	1.25	1.3	1.35	1.4
83.1%	93.5%	94.6%	92.4%	90.4%	94.1%	91.6%	92.5%

tem at each SMP via 10-fold cross validation with a 90/10 train/test split. Figure 1 shows the performance of our system at each speed manipulation parameter.

Table 1 shows that we can reliably detect manipulated videos, achieving a maximum accuracy of 99.1% at a speed manipulation parameter of 0.7. We also see that our system performs better when detecting slowed videos (SMP ≤ 1), than when detecting sped-up videos, (SMP ≥ 1). This behavior is expected, and can be predicted when observing Figure 1. The effects of slowing a video, as shown in Figure 1a, are much more apparent than the effects of speeding-up the same video, as in Figure 1c.

We also note decreased performance at manipulation parameters close to 1, such as 0.95 and 1.05. At these rates, the fingerprint signal is very weak, appearing much less often than in the case of other parameters we considered. In light of this weak fingerprint signal, we expect these manipulated videos to be difficult to detect.

6.4. Parameter Estimation

Often, an investigator will want to know, not only if a video has been manipulated, but by how much. To do this, we labeled all 7038 video in our dataset with the SMP used to create it, then trained our system to estimate the SMP of a given video. We calculated the bias and variance of our estimator, and again verified our results using 10-fold cross validation.

The mean error of the estimated parameter was -0.0025, with a variance of 0.0259. Our proposed system does not appear to exhibit a strong bias. The reported variance suggests that the expected difference between the estimated parameter and the true parameter is 0.16. Figure 5 shows the distribution of estimated parameters for each SMP we consider in our dataset.

In Figure 5, we note that our proposed system's estimation appears to be multimodal when estimating some parameters in our dataset, as if two different manipulation parameters resulted in a similar fingerprint signal.

We note that slowed videos (SMP < 1) exhibit a different fingerprint signal than sped-up videos (SMP > 1) do. In light of this, we separated our dataset into two groups, 3726 videos with a SMP greater than or equal to 1, and 3726 videos with a SMP less than or equal to 1. Videos with a SMP equal to 1 are unmodified videos, and appear in both groups. For each group we again trained our system to estimate the SMP of videos within that group. The bias and error variance of each estimator is listed in Table 2.

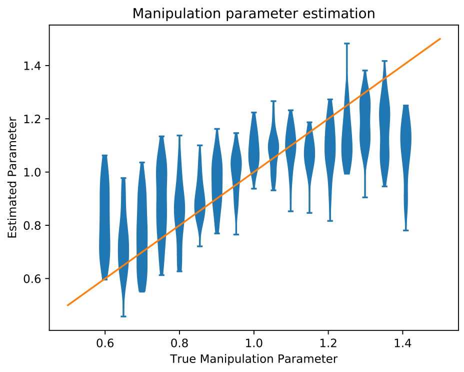


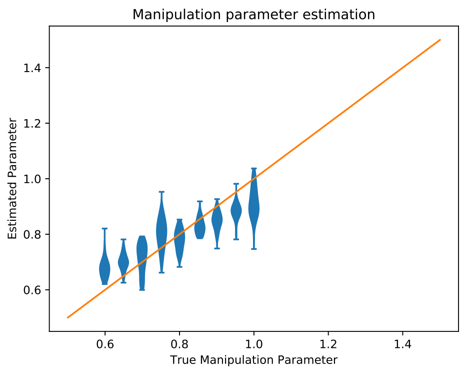
Figure 5: Distribution of speed manipulation parameter estimations for each SMP, with maximum and minimum values marked.

Table 2: Performance of SMP estimators.

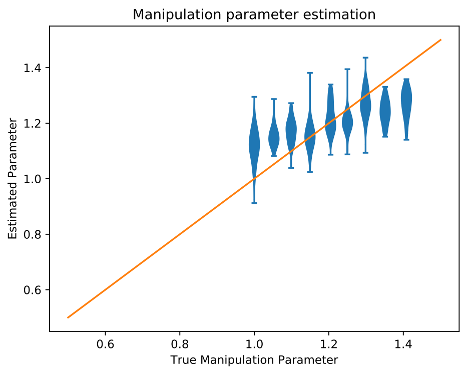
	Mean Bias	Error variance
All	-0.0025	0.02590
SMP ≥ 1	0.0011	0.01006
SMP ≤ 1	-0.0013	0.00543

In Table 2, we note that the error variance of our system when trained and tested only on slowed videos (SMP ≤ 1) is half that of the system trained and tested on sped-up videos (SMP ≥ 1). This result is consistent with our earlier findings which showed that our system was better able to discriminate slowed videos, as compared to sped-up videos. From Figure 1 we can see that speed manipulation traces are more pronounced in slowed videos. This reinforces the result in Table 1, which indicated that slowed videos were easier to detect. We attribute this difference to the bidirectional nature of B frames, which allows them many more frames from which to predict, and lessens the effect of any particular frame being deleted.

Figure 6 shows the distribution of estimations for each video when models are predicated on increased or decreased speed. Comparing Figure 6 to Figure 5, it can be seen that restricting the system to either sped-up or slowed down video significantly reduces our system's error variance. In Figure 6 we note that at SMP's close to 1, our estimator is biased away from 1. We also observe that at SMP's near 0.6, our estimator is biased away form 0.6, and likewise at SMP's near 1.4. These extreme areas, 0.6 and 1.4, are areas where the manipulation is likely to be perceptible to the human eye. Videos with SMP near 1 are not manipulated very strongly; we assume these manipulations are less likely.



(a) SMP ≤ 1



(b) SMP ≥ 1

Figure 6: Distribution of speed manipulation parameter estimations for each SMP, predicated on SMP magnitude. Ticks show maximum and minimum values.

6.5. Organic Video

Finally, we evaluated our system's performance when detecting videos manipulated with widely available consumer video editing software [5]. Because these manipulations mimic those that would naturally be used when creating a speed altered video, we refer to these as *organic* videos. First, we trained our manipulation detection system using all 7038 videos in our dataset. Next, to create organic video, we used consumer software (without using the FFmpeg encoder) to alter the speed of all 414 original videos using a rate distortion parameter of 0.75. We then classified each video using our trained system. As a result, only 3 of the 414 organic videos were misclassified as unaltered, resulting in a detection accuracy of 99.3%.

We calculated the average time needed to read a video, extract features, and perform classification based on those features. On average, this process took 0.042 seconds per video. However, when we subtract the time needed to load the video into memory and read its EFS, the average feature extraction and classification time was measured at 0.001 seconds. From these measurements, we conclude that our system provides a reasonable large-scale test, for use on video sharing and social media platforms.

Case Study The last organic video we tested was a copy of the manipulated video posted to Facebook, which showed Speaker of the House Pelosi, in an interview. This video was downloaded from the original Facebook post. To ensure a reasonable sample size, we extracted both the EFS and frame type sequences from the video, then split the sequences into 27 sub-sequences of 200 frames each. These sequences were treated as individual sequences, and features were extracted from each sequence independently. Using the same model trained in the previous experiment, we classified each clip. All 27 clips were found by our system to be manipulated, for a 100% detection accuracy.

Figure 7 shows a portion of the EFS sequence associated with the video. In this Figure, we note the line of red poins along the bottom edge, indicating the inserted frames. We also note that the average encoded B frame size is between 0.7KB and 1.3KB, an order of magnitude less than in Figure 1a. Despite this, our system is still able to correctly identify every part of this video as modified.

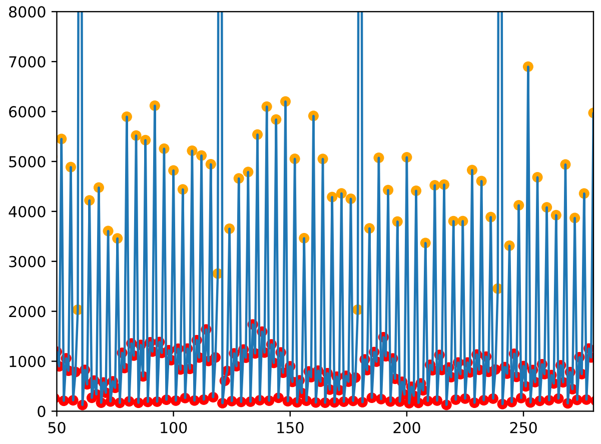


Figure 7: Encoded frame size sequence of manipulated video of Speaker Pelosi.

7. Concluding Remarks

In this work, we presented a novel trace for the detection of manipulated videos. Our trace uses the number of bytes used to encode each frame as a method of detecting inserted and deleted frames, which occur when a video's speed has been manipulated. This trace can be extracted from a video faster than the video can be decoded, which allows a large number of videos to be processed in a relatively small amount of time. We also propose a detection algorithm based on this trace, and show that our algorithm performs with over 99% accuracy on organic videos.

References

[1] The deepfake detection challenge (dfdc), 2019. 5

[2] Darius Afchar, Vincent Nozick, Junichi Yamagishi, and Isao Echizen. Mesonet: a compact facial video forgery detection network. In *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–7, 2018. 1

[3] Aayush Bansal, Shugao Ma, Deva Ramanan, and Yaser Sheikh. Recycle-gan: Unsupervised video retargeting. In *The European Conference on Computer Vision (ECCV)*, September 2018. 1

[4] Paolo Bestagini, Marco Fontani, Simone Milani, Mauro Barni, Alessandro Piva, Marco Tagliasacchi, and Stefano Tubaro. An overview on video forensics. In *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 1229–1233, 2012. 1, 2

[5] Avidemux Developers. Avidemux. `http://www.avidemux.org`, 2020. 8

[6] FFmpeg Developers. Ffmpeg. `http://ffmpeg.org/`, 2020. 6

[7] Luca D'Amiano, Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. A patchmatch-based dense-field algorithm for video copy–move detection and localization. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(3):669–682, 2019. 2

[8] A. Gironi, Marco Fontani, Tiziano Bianchi, Alessandro Piva, and Mauro Barni. A video forensic technique for detecting frame deletion and insertion. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6226–6230, 2014. 2

[9] David Güera and Edward J. Delp. Deepfake video detection using recurrent neural networks. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, 2018. 1

[10] Drew Harwell. White house shares doctored video to support punishment of journalist jim acosta, Nov 2018. 1

[11] Drew Harwell. Faked pelosi videos, slowed to make her appear drunk, spread across social media, May 2019. 1

[12] Brian C. Hosler, Owen Mayer, Belhassen Bayar, Zhao Xinwei, Chen Chen, James A. Shackleford, and Matthew C. Stamm. A video camera model identification system using deep learning and fusion. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8271–8275, 2019. 2

[13] Brian C. Hosler, Xinwei Zhao, Owen Mayer, Chen Chen, James A. Shackleford, and Matthew C. Stamm. The video authentication and camera identification database: A new database for video forensics. *IEEE Access*, 7:76937–76948, 2019. 2

[14] Xinghao Jiang, Wan Wang, Tanfeng Sun, Yun Q. Shi, and Shilin Wang. Detection of double compression in mpeg-4 videos based on markov statistics. *IEEE Signal Processing Letters*, 20(5):447–450, 2013. 2

[15] Pedro R. M. Júnior, Luca Bondi, Paolo Bestagini, Anderson Rocha, and Stefano Tubaro. A prnu-based method to expose video device compositions in open-set setups. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 96–100, 2019. 2

[16] Vikrant Kobla, Daniel DeMenthon, and David Doermann. Detection of slow-motion replay sequences for identifying sports videos. In *IEEE Workshop on Multimedia Signal Processing*, pages 135 – 140, 02 1999. 2

[17] Pavel Korshunov and Sebastien Marcel. Deepfakes: a new threat to face recognition? assessment and detection, 2018. 1

[18] Yuezun Li, Ming-Ching Chang, and Siwei Lyu. In ictu oculi: Exposing ai created fake videos by detecting eye blinking. In *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–7, 2018. 1

[19] Sara Mandelli, Paolo Bestagini, Stefano Tubaro, Davide Cozzolino, and Luisa Verdoliva. Blind detection and localization of video temporal splicing exploiting sensor-based footprints. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 1362–1366, 2018. 2

[20] Sara Mandelli, Paolo Bestagini, Luisa Verdoliva, and Stafano Tubaro. Facing device attribution problem for stabilized video sequences. *IEEE Transactions on Information Forensics and Security*, 15:14–27, 2020. 2

[21] Owen Mayer, Brian C. Hosler, and Matthew C. Stamm. Open set video camera model verification. In *ICASSP 2020 - 202 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020. 2

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 6

[23] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics: A large-scale video dataset for forgery detection in human faces, 2018. 1

[24] Skipper Seabold and Josef Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010. 6

[25] Tianxiang Shen. " deep fakes " using generative adversarial networks (gan). 2018. 1

[26] Raahat Singh and Naveen Aggarwal. Detection of re-compression, transcoding and frame-deletion for digital video authentication. In *2015 2nd International Conference on Recent Advances in Engineering Computational Sciences (RAECS)*, pages 1–6, 2015. 2

[27] Matthew C. Stamm, Sabrina Lin, and K.J. Ray Liu. Temporal forensics and anti-forensics for motion compensated video. *IEEE Transactions on Information Forensics and Security*, 7(4):1315–1329, 2012. 2

[28] Matthew C. Stamm, Min Wu, and K.J. Ray Liu. Information forensics: An overview of the first decade. *IEEE Access*, 1:167–200, 2013. 1

[29] Samet Taspinar, Manoranjan Mohanty, and Nasir Memon. Source camera attribution using stabilized video. In *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2016. 2

[30] Ruben Tolosana, Ruben Vera-Rodriguez, Julian Fierrez, Aythami Morales, and Javier Ortega-Garcia. Deepfakes and

beyond: A survey of face manipulation and fake detection, 2020. 1

[31] Weihong Wang and Hany Farid. Exposing digital forgeries in video by detecting double mpeg compression. In *Proceedings of the 8th workshop on Multimedia and security*, pages 37–47. ACM, 2006. 2

[32] Weihong Wang and Hany Farid. Exposing digital forgeries in interlaced and deinterlaced video. *IEEE Transactions on Information Forensics and Security*, 2(3):438–449, 2007. 2

[33] Weihong Wang and Hany Farid. Exposing digital forgeries in video by detecting duplication. In *Proceedings of the 9th Workshop on Multimedia & Security*, page 35–42, 2007. 2

[34] Yuxing Wu, Xinghao Jiang, Tanfeng Sun, and Wan Wang. Exposing video inter-frame forgery based on velocity field consistency. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2674–2678, 2014. 2