Stealthy Adversarial Perturbations Against Real-Time Video Classification Systems

Shasha Li*, Ajaya Neupane*, Sujoy Paul*, Chengyu Song*,
Srikanth V. Krishnamurthy*, Amit K. Roy Chowdhury* and Ananthram Swami†
*University of California Riverside, {sli057, ajaya, spaul003}@ucr.edu, {csong, krish}@cs.ucr.edu, amitrc@ece.ucr.edu
†United States Army Research Laboratory, ananthram.swami.civ@mail.mil

***Abstract*—Recent research has demonstrated the brittleness of machine learning systems to adversarial perturbations. However, the studies have been mostly limited to perturbations on images and more generally, classification tasks that do not deal with real-time stream inputs. In this paper we ask "Are adversarial perturbations that cause misclassification in real-time video classification systems possible, and if so what properties must they satisfy?" Real-time video classification systems find application in surveillance applications, smart vehicles, and smart elderly care and thus, misclassification could be particularly harmful (e.g., a mishap at an elderly care facility may be missed). Video classification systems take video clips as inputs and these clip boundaries are not deterministic. We show that perturbations that do not take "the indeterminism in the clip boundaries input to the video classifier" into account, do not achieve high attack success rates. We propose novel approaches for generating 3D adversarial perturbations (perturbation clips) that exploit recent advances in generative models to not only overcome this key challenge but also provide stealth. In particular, our most potent 3D adversarial perturbations cause targeted activities in video streams to be misclassified with rates over 80%. At the same time, they also ensure that the perturbations leave other (untargeted) activities largely unaffected making them extremely stealthy. Finally, we also derive a single-frame (2D) perturbation that can be applied to every frame in a video stream, and which in many cases, achieves extremely high misclassification rates.**

I. Introduction

Deep Neural Networks (DNN) based real-time video classification systems are being increasingly deployed in real world scenarios. Examples of applications include video surveillance [42], self driving cars [21], health-care [52], etc. To elaborate, video surveillance systems capable of automated detection of "targeted" human activities or behaviors (e.g., accident, violence), can trigger alarms (upon detection) and drastically reduce information workloads on human operators. Without the assistance of DNN-based classifiers, human operators will need to simultaneously monitor footage from a large number of video sensors. This can be a difficult and exhausting task, and comes with the risk of missing behaviors of interest and slowing down decision cycles. In self-driving cars, video classification has been used to understand pedestrian actions and make navigation decisions [21]. Similar applications can be envisaged in the Army Next Generation Combat Vehicle (NGCV). Real-time video classification systems have also been deployed for automatic "fall detection" in elderly care facilities [52], and detection of abnormal actions around automated teller machines [47]. All of these applications directly relate to the physical security or safety of people and property. Thus, stealthy attacks on such real-time video classification systems are likely to cause unnoticed pecuniary loss and compromise personal safety. Note that while objects can be detected or distinguished by examining the individual frames in a video (akin to object detection on images), many activities can only be recognized or distinguished by considering a sequence of frames holistically (i.e., a clip consisting of multiple frames).

Recent studies have shown that virtually all DNN-based systems are vulnerable to well-designed adversarial inputs [10], [29], [30], [39], [43], which are also referred to as *adversarial examples*. Szegedy *et al.* [43], showed that adversarial perturbations that are hardly perceptible to humans can cause misclassification in DNN-based image classifiers. Goodfellow *et al.* [11], analyzed the potency of realizing adversarial samples in the physical world. Moosavi *et al.* [29], and Mopuri *et al.* [32], introduced the concept of "image-agnostic" perturbations. Recent efforts by Hosseini et al. [15], and Wei et al. [54], explore adversarial perturbations on videos. However, they are limited in that their attack models do not work on real-time video classification systems (more details in § IX).

The high level question that we try to address in this paper is: *"Is it possible to launch stealthy attacks against DNN-based real-time video classification systems by adding adversarial perturbations on a video stream, and if so how?"* In contrast with the aforementined prior work, attacking a real-time video classifier poses new challenges that were not all previously identified or addressed. *First*, because video streams are collected in real-time, the corresponding perturbations also need to be generated on-the-fly with the same frame rate which can be extremely computationally intensive. *Second*, to make the attack stealthy, attackers would want to add perturbations on the video in such a way that they will only cause misclassification for the targeted (possibly malicious) activities, while keeping the classification of other activities unaffected. In a real-time video stream, since the activities change across time, it is hard to identify online and in one-shot [8], the target frames on which to add perturbations (and thereby ensure that the other untargeted activities are not affected). *Third*, real-time video classifiers use video clips (a set of frames) as inputs [8], [47] (i.e., as video is captured, it is broken up into clips and each clip is fed to the classifier). This

Network and Distributed Systems Security (NDSS) Symposium 2019
24-27 February 2019, San Diego, CA, USA
ISBN 1-891562-55-X
https://dx.doi.org/10.14722/ndss.2019.23202
www.ndss-symposium.org

introduces two additional hyper-parameters viz., the *length* of a clip and the *boundaries* (i.e., beginning and ending) of a clip. Even if attackers are aware of the length of each clip, it is hard to predict the boundaries of the clips as they are nondeterministic. This is problematic because when the attacker generated perturbations are applied to the wrong frame within a clip (i.e., perturbation for frame 1 of a clip being applied to frame 2 of that clip), the perturbations may not work as expected. (Please see Figure 4 and the associated discussion for more details).

In this paper, our first objective is to investigate how to generate adversarial perturbations against real-time video classification systems by overcoming the above challenges. We resolve the real-time challenge by using *universal perturbations* (UP) [29]. UPs are universal in the sense that a UP is not specific to one input example, but works on any input example from the same distribution as that of the training data. Universal perturbations affect the classification results by using just a (single) set of perturbations generated off-line. Because they work on unseen inputs, they preclude the need for intensive on-line computations to generate perturbations for every incoming video clip. To generate such universal perturbations, we leverage generative DNN models.

However, adding universal perturbations to all clips of the video can cause misclassification of all the activities in the video stream. This may expose the attack since the results may be abnormal (e.g., many people performing rare actions). It may even cause activities from other classes to be misclassified as the target class. To make the attack stealthy, we introduce the novel concept of *dual purpose universal perturbations*, which we define as universal perturbations which only cause misclassification of activities belonging to the target class, while minimizing, or ideally, having no effect on the classification results for activities belonging to the other classes.

Dual purpose perturbations by themselves do not provide high success rates because of the nondeterminism of the clip boundaries. To be more specific, let l be the length of a clip input to the classifier, and $p = \{p_1, p_2, \ldots, p_l\}$ be the perturbations for a clip of frames $x = \{f_1, f_2, \ldots, f_l\}$; then input $x' = \{f_1 \oplus p_1, f_2 \oplus p_2, \ldots, f_l \oplus p_l\}$, where $\oplus$ denotes pixel-wise addition, would yield misclassification but other combinations like $x'' = \{f_1 \oplus p_l, f_1 \oplus p_1, \ldots, f_l \oplus p_{l-1}\}$ (where p_l in the latter expression refers to the last frame in the previous clip) may not cause misclassification. To solve this problem, we introduce a new type of perturbation that we call the *Circular Universal Dual Purpose Perturbation (C-DUP)*. The C-DUP is a 3D perturbation which is effective on a video stream even in the presence of a temporal misalignment between the perturbation clips and the input video clips. Specifically, any cyclic permutations of a C-DUP perturbation clip are also still valid perturbations. For example, both $\{f_1 \oplus p_l, f_2 \oplus p_1, \ldots, f_l \oplus p_{l-1}\}$ and $\{f_1 \oplus p_{l-1}, f_2 \oplus p_l, \ldots, f_l \oplus p_{l-2}\}$ can cause expected misclassification. Because of this property, C-DUP works even if the sequential concatenation of two broken up parts of two consecutive perturbation clips, is added to an input video clip as a perturbation clip. To generate C-DUPs, we make significant changes to the baseline generative model used to generate universal perturbations. In particular, we add a new unit to generate circular perturbations, that is placed between the generator and the fixed discriminator (as discussed later). We demonstrate that the C-DUP is very stable and effective in achieving real-time stealthy attacks on video classification systems.

Finally, to better understand the effect of the temporal dimension, we also investigate the feasibility of attacking the classification systems using a simple and light 2D perturbation (frame level instead of clip level) which is applied across all the frames of a video. By tweaking our generative model, we are able to generate such perturbations which we name as *2D Dual Purpose Universal Perturbations (2D-DUP)*. These perturbations work well on a sub-set of videos, but not all. We will discuss the reasons for this when we describe these 2D attacks in § VI-D.

Our Contributions: In brief, our contributions are:

- We provide a comprehensive analysis of the challenges in crafting adversarial perturbations for real-time video classifiers. We empirically identify what we call the boundary effect phenomenon in generating adversarial perturbations against video streams (see § VI-B). In a nutshell, the boundary effect arises because of the nondeterminism of the boundaries of the clips input to the video classification system.
- We design and develop a generative framework to craft two types of stealthy adversarial perturbations against real-time video classifiers, viz., the circular dual purpose universal perturbation (C-DUP) and the 2D dual purpose universal perturbation (2D-DUP). These perturbations are agnostic to (a) the content the video streams capture (i.e., are universal) and (b) the clip boundaries within the streams.
- We demonstrate the potency of our adversarial perturbations using two different video datasets. In particular, the UCF101 dataset captures coarse-grained activities (human actions such as applying eye makeup, bowling, drumming) [41]. The Jester dataset captures fine-grained activities (hand gestures such as sliding hand left, sliding hand right, turning hand clockwise, turning hand counterclockwise) [7]. We are able to launch stealthy attacks on both datasets with over a 80 % misclassification rate, while ensuring that the other classes are correctly classified with relatively high accuracy.

II. Background

In this section, we provide the background relevant to our work. Specifically, we discuss how a real-time video classification system works and what standard algorithms are currently employed for action recognition.

A. Real-time video-based classification systems

DNN based video classification systems are being increasingly deployed in real-world scenarios. Examples include fall detection in elderly care [9], abnormal event detection on campuses [49], [50], security surveillance for smart cities [51], and self-driving cars [21], [22]. Given an input real-time video stream, which may contain one or more known actions, the goal of a video classification system is to correctly recognize the sequence of the performed actions. Real-time video classification systems commonly use a *sliding window*

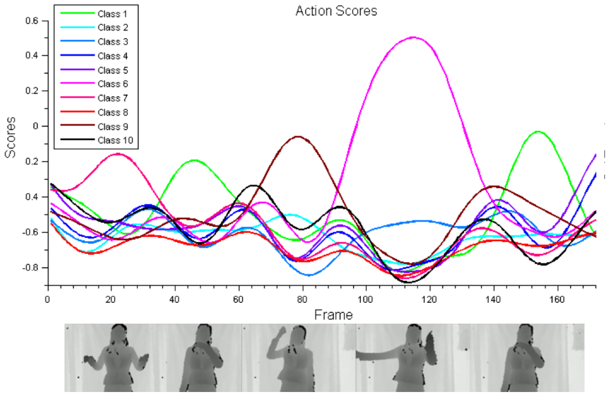


Fig. 1: This figure [8] illustrates the score curves computed by a video classifier with a sliding window for every class. Real-time video classification systems use these score curves to do online action recognition.

to extract video clips and use the clips as inputs to a classifier to analyze the video stream [8], [47]. The classifier computes an output score for each class in each sliding window. The sliding window moves with a stride. Moving in concert with the sliding window, one can generate "score curves" for each action class. Note that the scores for all the action classes evolve with time. The score curves are then smoothed (to remove noise) as shown in Figure 1. With the smoothed score curves, the on-going actions are predicted online. From the figure one can see that, the real-time video classification system is fooled if one can make the classifier output a low score for the true class in each sliding window; with this, the true actions will not be recognized.

B. *The C3D classifier*

Convolutional neural networks (CNNs) are being increasingly applied in video classification. Among these, spatio-temporal networks like C3D [46] and two-stream networks like I3D [6] outperform other network structures [13], [14]. However, two-stream networks require optical flow extraction as preprocessing. Without the requirement of non-trivial preprocessing on the video stream, spatio-temporal networks are more efficient and suitable for real-time applications; among these, C3D is the start-of-art model [13].

Given its desirable attributes and popularity, without loss of generality, we use the C3D model as our attack target in this paper. The C3D model is based on 3D ConvNet (a 3D convolutional neural network or CNN) [20], [46], [48], which is very effective in modeling temporal information (because it employs 3D convolution and 3D pooling operations). The architecture and hyperparamters of C3D are shown in Figure 2. The input to the C3D classifier is a clip consisting of 16 consecutive frames. This means that upon using C3D, the sliding window size is 16. Both the height and the width of each frame are 112 pixels and each frame has 3 (RGB) channels. The last layer of C3D is a softmax layer that provides a classification score with respect to each class.

III. Threat Model and Datasets

In this section, we describe our threat model. We also provide a brief overview of the datasets we chose for validating our attack models.

A. *Threat model*

We consider a white-box model for our attack, i.e., the adversary has access to the training datasets used to train the video classification system, and has knowledge of the deep neural network model used in the real-time classification system. We assume that the datasets are trusted. We also assume that the adversary is capable of injecting perturbations in the real-time video stream. In particular, we assume the adversary to be a man-in-the-middle that can intercept and add perturbations to streaming video [25], or that it could have previously installed a malware that is able to add perturbation prior to classification [34].

We assume that the adversaries seek to be stealthy i.e., they want the system to only misclassify malicious actions without affecting the recognition of the other actions. So, we consider two attack goals. *First*, given a target class, we want all the clips from this class to be misclassified by the real-time video classifier. *Second*, for all the clips from other (non-target) classes, we want the classifier to correctly classify them.

We point out here that a man-in-the-middle attacker will be unable to simply replace the streaming video with static frames or pre-recorded video and yet achieve the required stealthiness. This is because of two reasons. First, the attacker has no a priori knowledge about "when" a targeted action occurs. For example, an attacker with malicious intent may want to misclassify the action of an elderly person falling down at a smart elderly care center that is monitored by multiple cameras (e.g., [37]). Since the attacker does not know when and where exactly an elderly person will fall down, it has to replace the video streams from *all* the cameras with the pre-recorded video of elderlies doing something else (e.g., walking) *for extended periods or ideally all the time*. However, it is hard to guarantee that the replaced videos are visually similar to the real-time environment (e.g., people and their actions, weather) and replaying videos out of context may be noticeable. In addition, it is possible that the attacker may be capable of delaying the video by a short period to inject targeted perturbations against specific activities; however, while such an approach can eliminate universal and stealth requirements, it cannot overcome the boundary effect and cannot obviate the corresponding computation needed for online perturbation generation. Second, the attacker also has to replace the actions of multiple people involved at the facility captured with the multiple cameras. In other words, a large number of replacement videos capturing a large set of people at the facility will be necessary. If the replaced videos show the same person at different locations, or people who are not at the facility, this will be noticeable. Applying perturbations on the video will enable the attacker to stealthily misclassify only the specific activity relating to the falling an elderly, keeping all other actions unaffected. Furthermore, the imperceptibility of these perturbations will not cause any human operator to notice anything overtly wrong.

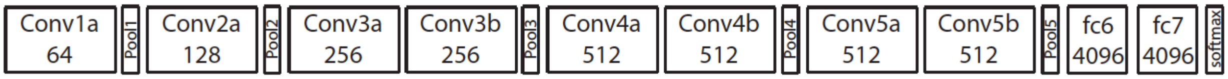


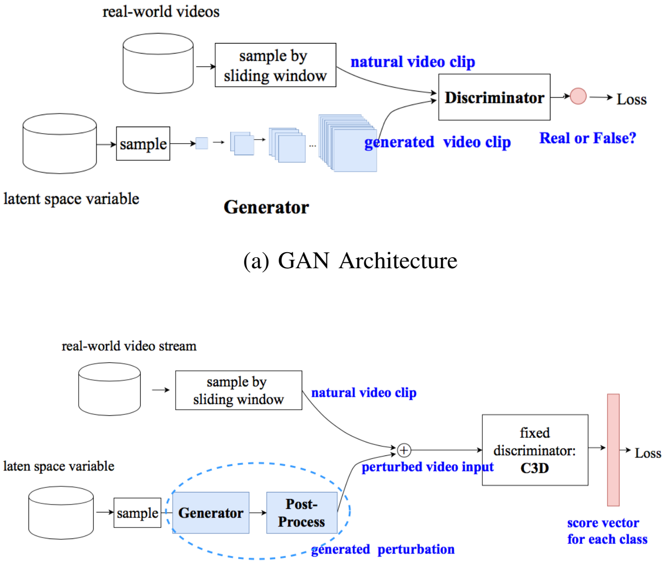
Fig. 2: The C3D architecture [46]. C3D net has 8 convolution, 5 max-pooling, and 2 fully connected layers, followed by a softmax output layer. All 3D convolution kernels are $3 \times 3 \times 3$ with a stride [46] of 1 in both spatial and temporal dimensions. The number of filters is shown in each box. The 3D pooling layers are represented as pool1 to pool5. All pooling kernels are $2 \times 2 \times 2$, except for pool1, which is $1 \times 2 \times 2$. Each fully connected layer has 4096 output units.

B. Our datasets

We use the human action recognition dataset UCF-101 [41] and the hand gesture recognition dataset 20BN-JESTER dataset (Jester) [7] to validate our attacks on video classification systems. We use these two datasets because they represent two kinds of classification, i.e., coarse-gained and fine-grained action classification.

The UCF 101 dataset: The UCF 101 dataset used in our experiments is the standard dataset collected from Youtube. It includes 13320 videos from 101 human action categories (e.g., applying lipstick, biking, blow drying hair, cutting in the kitchen etc.). The videos collected in this dataset have variations in camera motion, appearance, background, illumination conditions etc. Given the diversity it provides, we consider the dataset to validate the feasibility of our attack model on coarse-gained actions. There are three different (pre-existing) splits [41] in the dataset; we use split 1 for both training and testing, in our experiments. The training set includes 9,537 video clips and the testing set includes 3,783 video clips.

The Jester dataset: The 20BN-JESTER dataset (Jester) is a recently collected dataset with hand gesture videos. These videos are recorded by crowd-sourced workers performing 27 kinds of gestures (e.g., sliding hand left, sliding two fingers left, zooming in with full hand, zooming out with full hand etc.). We use this dataset to validate our attack with regard to fine-grained actions. Since this dataset does not currently provide labels for the testing set, we withhold a subset of the training set as our validation set and use the validation set for testing. The training set has 148,092 short video clips and our testing set has 14,787 short video clips.



(a) GAN Architecture

(b) Our Architecture

Fig. 3: We use a GAN-like architecture for the generative model. However, our architecture is different from GAN in the following aspects: 1) The discriminator is a pre-trained classifier we attack, whose goal is to classify videos, and not to distinguish between the natural and synthesized inputs; 2) The generator generates perturbations, and not direct inputs to the discriminator, and the perturbed training inputs are fed to discriminator; 3) The learning objective is to let the discriminator misclassify the perturbed inputs.

IV. Generating perturbations for real-time video streams

From the adversary's perspective, we first consider the challenge of attacking a real-time video stream. In brief, when attacking an image classification system, the attackers usually take the following approach. First, they obtain the *target* image that is to be attacked with its true label. Next, they formulate a optimization problem wherein they try to compute the ”minimum” noise that is to be added (towards imperceptibility) in order to cause a mis-classification of the target. The formulation takes into account the function of the classifier, the input image, and its true label. Backpropagation is commonly used to solve this optimization problem [11], [24], [27].

In the context of real-time video classification, the video is not available to the attackers a priori. Thus, they will need to create perturbations that can effectively perturb an incoming video stream, whenever a *target class* is present. Generation of online perturbations based on an incoming video stream would have an associated cost of $O(f \times b \times n)$ where, f is frame rate, b is cost of one backpropagation on the DNN, and n is the number of backpropagations needed to solve the optimization problem.

Our approach is to compute the perturbations offline and apply them online, and thus, the online computation cost is $O(1)$. Since we cannot predict what is captured in the video, we need perturbations which work with unseen inputs. A type of perturbation that satisfies this requirement is called the Universal Perturbation (UP), which has been studied in the context of generating adversarial samples against image classification systems [29], [32]. In particular, Mopuri *et al.*, have developed a generative model that learns the space of universal perturbations for images using a GAN-like architecture. Inspired by this work, we develop a similar architecture, but make modifications to suit our objective. Our goal is to generate adversarial perturbations that fool the discriminator instead of exploring the space for diverse UPs. In addition, we retrofit the architecture to handle video inputs. Our architecture is depicted in Figure 3b. It consists of three main components: 1) a 3D generator which generates universal perturbations (clips); 2) a post-processor, which for now does not do anything but is needed to solve other challenges described in subsequent sections; and 3) a pre-trained discriminator for video classification, viz., the C3D model described in § II-B.

The 3D generator in our model is configured to use 3D deconvolution layers and provide 3D outputs as shown in Figure 8. Specifically, it generates a clip of perturbations, whose size is equal to the size of the video clips taken as input by the C3D classifier. To generate universal perturbations, the generator first takes a noise vector z from a latent space. Next, It maps z to a perturbation clip p, such that, $G(z) = p$. It then adds the perturbations on a training clip x (denote the set of inputs from the training class as X) to obtain the perturbed clip $x + p$. Let $c(x)$ be the true label of x. This perturbed clip is then input to the C3D model which outputs the score vector $Q(x + p)$ (for the perturbed clip). The classification should ensure that the highest score corresponds to the true class ($c(x)$ for input x) in the benign setting. Thus, the attacker seeks to generate a p such that the C3D classifier outputs a low score to the $c(x)$th element in the Q vector (denoted as $Q_{c(x)}$) for $x+p$. In other words, this means that after applying the perturbation, the probability of mapping x to class $c(x)$ is lower than the probability that it is mapped to a different class (i.e., the input activity is not correctly recognized).

We seek to make this perturbation clip p "a universal perturbation", i.e., adding p to any input clip belonging to the target class would cause misclassification. This means that we seek to minimize the sum of the cross-entropy loss over all the training data as per Equation 1. Note that the lower the cross-entropy loss, the higher the divergence of the predicted probability from the true label [17].

$$\underset{G}{\text{minimize}} \quad \sum_{x \in X} -\log[1 - Q_{c(x)}(x + G(z)] \tag{1}$$

When the generator is being trained, for each training sample, it obtains feedback from the discriminator and adjusts its parameters to cause the discriminator to misclassify that sample. It tries to find a perturbation that works for every sample from the distribution space known to the discriminator. At the end of this phase, the attacker will have a generator that outputs universal perturbations which can cause the misclassification on any incoming input sample from the same distribution (as that of the training set). However, as discussed next, just applying the universal perturbations alone will not be sufficient to carry out a successful attack. In particular, the attack can cause unintended clips to be misclassified as well, which could compromise our stealth requirement as discussed next in §V.

V. Making Perturbations Stealthy

Blindly adding universal perturbations will affect the classification of clips belonging to other non-targeted classes. This may raise alarms, especially if many of these misclassifications are mapped on to rare actions. Thus, while causing the target class to be misclassified, the impact on the other classes must be imperceptible. This problem can be easily solved when dealing with image recognition systems since images are self-contained entities, i.e., perturbations can be selectively added to target images only. However, video inputs change temporally and an action captured in a set of composite frames may differ from that in the subsequent frames. It is thus hard to a priori identify (choose) the frames relating to the target class, and add perturbations specifically to them. For example, consider a case with surveillance in a grocery store. If attackers seek to misclassify an action related to shoplifting and cause this action to go undetected, they are unlikely to have precise knowledge of the exact time when the action will occur and be captured by the video activity recognition system. Adding universal perturbations blindly in this case, could cause misclassifications of other actions (e.g., other benign customer actions may be mapped onto shoplifting actions thus triggering alarms). A similar example may be construed with respect to the elderly care system described in § III-A; here, the attacker has no way of knowing a priori when an elderly falls.

Since it is hard (or even impossible) to a priori identify the frame(s) that capture the intended actions and choose them for perturbation, the attackers need to add perturbations to each frame. However, to make these perturbations furtive, they need to ensure that the perturbations added only mis-classify the target class while causing other (non-targeted) classes to be classified correctly. We name this unique kind of universal perturbations as "Dual-Purpose Universal Perturbations" or DUP for short.

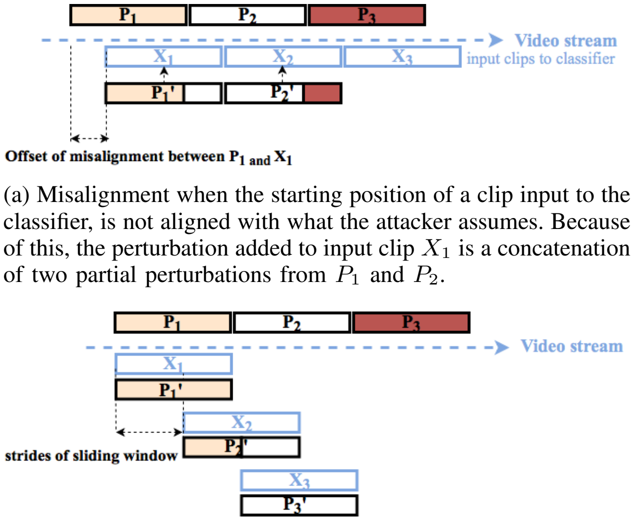
In order to realize DUPs, we have to guarantee that for the input clip x_t, if it belongs to the target class (denote the set of inputs from the target class as T), the C3D classifier returns a low score with respect to the correct class $c(x_t)$, i.e., $Q_{c(x_t)}$. For all input clips x_s that belong to other (non-target) classes (denote the set of inputs from non-target classes as S, thus, $S = X - T$), the model returns high scores with regard to their correct mappings ($Q_{c(x_s)}$). To cause the generator to output DUPs, we refine the optimization problem in Equation 1 as shown in Equation 2:

$$\begin{aligned} \underset{G}{\text{minimize}} \quad & \lambda \times \sum_{x_t \in T} -\log[1 - Q_{c(x_t)}(x_t + G(z))] \\ & + \sum_{x_s \in S} -\log[Q_{c(x_s)}(x_s + G(z))] \end{aligned} \tag{2}$$

The first term in the equation again relates to minimizing the cross-entropy of the target class, while the second term maximizes the cross-entropy relating to each of the other classes. The parameter λ is the weight applied with regard to the misclassification of the target class. For attacks where stealth is more important, we may use a smaller λ to guarantee that the emphasis on the misclassification probability of the target class is reduced while the classification of the non-target classes are affected to the least extent possible.

VI. Impact of Nondeterministic Clip Boundaries

In this section, we first discuss why directly applying existing methods to generate perturbations against video streams do


P1
P2
P3
Video stream
X1
X2
X3
input clips to classifier
P1'
P2'
Offset of misalignment between P1 and X1
P1
P2
P3
Video stream
X1
P1'
X2
strides of sliding window
P2'
X3
P3'

(a) Misalignment when the starting position of a clip input to the classifier, is not aligned with what the attacker assumes. Because of this, the perturbation added to input clip X_1 is a concatenation of two partial perturbations from P_1 and P_2.

(b) Misalignment can occur even if the starting position is aligned when a small stride is used. Here, the stride of the sliding window is half the clip size. This causes a misalignment because of which, the perturbation added to input clip X_2 is a concatenation of two partial perturbations from P_1 and P_2.

Fig. 4: Two cases that can potentially cause misalignment between perturbation clips and the input clips to the classifier. The first parts of both figures represent the temporal sequence of generated perturbation clips. The lower parts of both figures capture the temporal sequence of input clips tested by the video classifier and the perturbation clips added to them.

not work. Subsequently, we propose a new set of perturbations that do work (and are very effective) on video streams.

A. Misalignment due to Nondeterministic Clip Boundaries

The input to the video classifier is a clip composed of a sequence of frames. Given any input clip, the previously described attack methods (UP and DUP) can generate a perturbation clip that can be added to that input clip. As discussed in § II-A, an input clip is controlled by a sliding window which in turn is defined by three hyper-parameters: the *window size* l, the *sliding stride* o, and the *starting position* f_{start}. Because f_{start} is non-deterministic, the clip boundaries of an input to the classifier in a real-time video classification system are also *nondeterministic*. As a result, even for white-box attackers, they cannot know a priori the clip boundaries (the consecutive frames in a video stream belonging to an input clip) used by the video classifier.

The nondeterminism in the clip boundaries is likely to cause *a misalignment* between the perturbation clips generated by the attacker and the input clips used by the classifier. Figure 4 depicts two cases where misalignment happens even with the attacker-friendly white-box scenario. The first row shows three perturbation clips P_1, P_2 and P_3 generated by the attacker [1]. The second row shows three input clips X_1, X_2 and X_3 used by the classifier. The clips in the two sequences are not aligned because the starting point of the sliding window is different from that of the perturbation clip. Consequently, the perturbation applied to input clip X_1 is actually a concatenation of the latter part of P_1 and the first part of P_2 (a perturbation P_1').

In a second case, as shown in Figure 4b, the perturbation clip P_2 and the input clip X_2 are not aligned because the stride of the sliding window is smaller than the window size. This smaller stride is commonplace in video classification systems as discussed in [6], [8], [46], [47].

B. The Boundary Effect

Because C3D utilizes a 3D CNN, we find via empirical experiments that when there is a misalignment between the perturbation clip and the input clip, it can cause significant degradations in the attack success rates, even for universal perturbations. For example, considering Figure 4a, the DUP P_1 should work on any input clip; however, the actual applied perturbation clip P_1' (which is the concatenation of two partial broken up perturbations) is less likely to work. We refer to this phenomenon as the boundary effect.

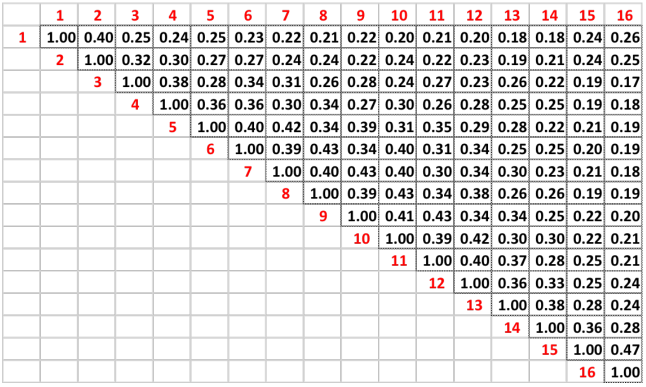
To formalize the boundary effect problem, let us consider a video stream represented by $\{\dots, f_{i-2}, f_{i-1}, f_i, f_{i+1}, f_{i+2}, \dots\}$ where, f_i represents the ith frame. The perturbation genererated by G to cause a misclassification of the clip $\{f_i, f_{i+1}, \dots, f_{i+l-1}\}$ (say $\{p_1, p_2, \dots, p_l\}$) will be different from the one generated for a temporally staggered clip $\{f_{i-1}, f_i, \dots, f_{i+l-2}\}$ (true for previously designed perturbations including UP and DUP). In other words, the perturbed clip $\{f_{i-1} \oplus p_1, f_1 \oplus p_2, \dots, f_{i+l-2} \oplus p_l\}$ is unlikely to be effective in achieving misclassification.

To exemplify this problem, we perform extensive evaluations of existing established methods with regard to attacking the C3D model. In particular, we use the APIs from the CleverHans repository [35] to generate video perturbations. We experiment with several methods from CleverHans, including the most recent ones (e.g., CarliniWagnerL2 and DeepFool). The results presented in the paper are based on the basic iteration method [24] with default parameters and all the videos in the UCF-101 testing set. We point out here that results based on all the other methods in the repository are very similar. We consider different boundaries for the clips in the videos (temporally staggered versions of the clips) and generate perturbations for each staggered version. Note that the sliding window size for C3D is 16 and thus, there are 16 staggered versions. We choose a candidate frame, and compute the correlations between the perturbations added in the different staggered versions. Specifically, the perturbations are tensors and the normalized correlation between two perturbations is the inner product of the unit-normalized tensors representing the perturbations.

We represent the average normalized correlations in the perturbations (computed across all frames in the testing set) for two locations in the matrix shown in Figure 5. The row index and the column index represent the location of the frames in the two staggered clips. For example, the entry corresponding to $\{7, 7\}$ represents the case where the frame considered was the 7^{th} frame in the two clips, (actually, here it is the same

[1] For UP and DUP, $P_1 = P_2 = P_3$.

clip). In this case, clearly the correlation is 1.00. However, we see that the correlations are much lower if the positions of the same frame in the two clips (two staggered versions) are different. As an example, consider the entry $\{5, 9\}$ which corresponds to the case where a frame is the fifth position in clip 1, and the same frame is at the ninth position in clip 2: the average normalized correlation between the two added perturbations is 0.39, which indicates that the perturbations that CleverHans adds in the two cases are quite different.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1.00	0.40	0.25	0.24	0.25	0.23	0.22	0.21	0.22	0.20	0.21	0.20	0.18	0.18	0.24	0.26
	2	1.00	0.32	0.30	0.27	0.27	0.24	0.24	0.22	0.24	0.22	0.23	0.19	0.21	0.24	0.25
		3	1.00	0.38	0.28	0.34	0.31	0.26	0.28	0.24	0.27	0.23	0.26	0.22	0.19	0.17
			4	1.00	0.36	0.36	0.30	0.34	0.27	0.30	0.26	0.28	0.25	0.25	0.19	0.18
				5	1.00	0.40	0.42	0.34	0.39	0.31	0.35	0.29	0.28	0.22	0.21	0.19
					6	1.00	0.39	0.43	0.34	0.40	0.31	0.34	0.25	0.25	0.20	0.19
						7	1.00	0.40	0.43	0.40	0.30	0.34	0.30	0.23	0.21	0.18
							8	1.00	0.39	0.43	0.34	0.38	0.26	0.26	0.19	0.19
								9	1.00	0.41	0.43	0.34	0.34	0.25	0.22	0.20
									10	1.00	0.39	0.42	0.30	0.30	0.22	0.21
										11	1.00	0.40	0.37	0.28	0.25	0.21
											12	1.00	0.36	0.33	0.25	0.24
												13	1.00	0.38	0.28	0.24
													14	1.00	0.36	0.28
														15	1.00	0.47
															16	1.00

Fig. 5: The average normalized correlation matrix computed with perturbations generated using the basic iteration API from CleverHans. The rows and columns represent the location of a frame in the two clips. The value represents the correlation between perturbations on the same frame but generated when that frame located in different positions (indicated by the row and column indices) in the two temporally staggered clips.

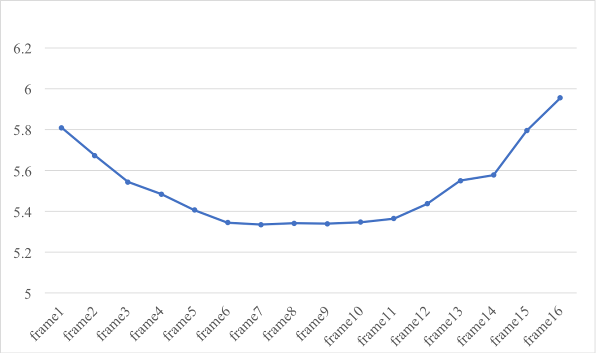


Fig. 6: Magnitude of perturbation on each frame: The abscissa is the frame position, and the ordinate is the magnitude of average perturbation on the frame. (The attack seeks to misclassify a given video clip from UCF 101 dataset.)

In Figure 6, we show the average magnitude of perturbations added (over all frames and all videos), when the target frame is at different locations within a clip. The abscissa depicts the frame position, and the ordinate represents the magnitude of the average perturbation. While the difference in the magnitude of perturbations added to two frames that are close to each other in terms of position (e.g., adjacent frames) within the clip, is small (this is because such frames are similar), the magnitude of perturbations added to frames that are distant in terms of location could potentially be quite different (because such frames could be quite dissimilar).

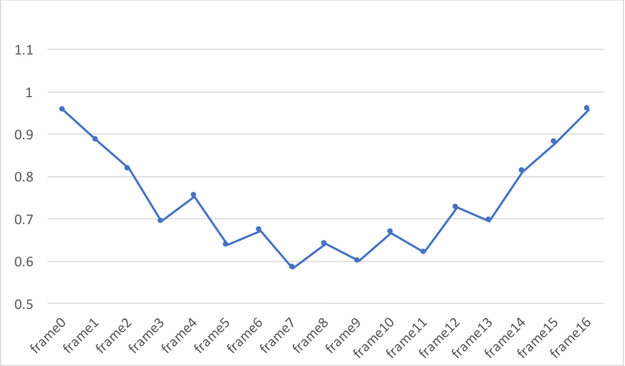


Fig. 7: Attack success rate when there is mismatch. The abscissa is the offset between the clip generating perturbation and the clip tested. The ordinate is the attack success rate. (Attack aims to misclassify a given video clip from UCF 101 dataset.)

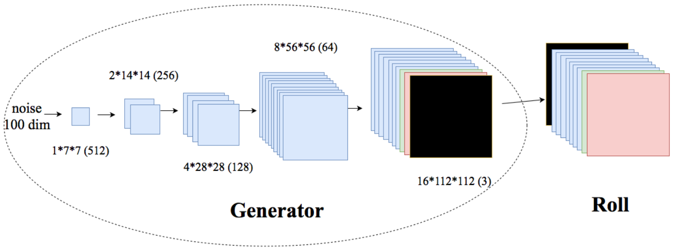


Fig. 8: This figure illustrates the Generator and Roll for generating C-DUP. 1) The generator takes a noise vector as input, and outputs a perturbation clip with 16 frames. Note that the number of temporal dimensions with the C3D model is 16. The output size for each layer is shown as temporal dimension × horizontal spatial dimension × vertical spatial dimension × number of channels. 2) The roll part shifts the perturbation clip by some offset. The figure shows one example where we roll the front black frame to the back.

We further showcase the impact of the boundary effect by measuring the degradation in attack efficacy due to mismatches between the anticipated start point when the perturbation is generated and the actual start point when classifying the clip (as shown in Figure 4a). Figure 7 depicts the results. The abscissa is the offset between the generated (intended) perturbation clip and the input clip used in classification. We can see that as the distance between the two start points increases, the attack success rate initially degrades but increases again as the the tested perturbation clip (a concatenation clip) is closer or more similar (has a better overlap) to the intended perturbation clip. For example, if the offset is 15, the perturbation clip added (concatenation clip) is offset by a single frame compared to the original (intended) perturbation clip.

C. Circular Dual-Purpose Universal Perturbation

To cope with the boundary effect, we develop a novel extension to the generative DNN model to significantly modify the DUPs proposed in § V to compose what we call "Circular Dual-Purpose Universal Perturbations (C-DUP)."

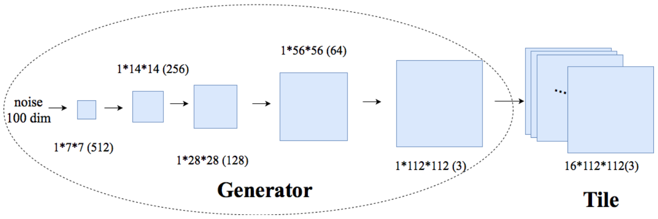


Fig. 9: This figure illustrates the Generator and Tile for generating 2D-DUP. 1) The generator takes a noise vector as input, and outputs a single-frame perturbation. 2) The tile part constructs a perturbation clip by repeating the single-frame perturbation generated 16 times.

Let us suppose that the size of the sliding window is 16. Then, the DUP clip P includes 16 frames (of perturbation), denoted by $\{p_1, p_2, \ldots, p_{16}\}$. Since P is a clip of universal perturbations, we launch the attack by repeatedly adding perturbations on each consecutive clip consisting of 16 frames, in the video stream. One can visualize that we are generating a perturbation stream which can be represented as $\{p_1, p_2, \ldots, p_{15}, p_{16}, p_1, p_2, \ldots\}$. Now, our goal is to guarantee that the perturbation stream works regardless of the clip boundaries chosen by the classifier. Towards this, we need to ensure that any sequential concatenation of partial perturbation clips (the last part of the first clip and the first part of the second clip) results in a valid perturbation. It is easy to see that for this to hold true, we need any *cyclic or circular shift* of the DUP clip to be a valid DUP perturbation too. In other words, we require the perturbation clips $\{p_{16}, p_1, \ldots, p_{15}\}$, $\{p_{15}, p_{16}, \ldots, p_{14}\}$, …, all to be valid perturbations. We emphasize here that UP and DUP do not have the cyclic property and thus, a sequential concatenation of parts of two consecutive UP or DUP clips will "not" be a valid perturbation.

To formalize, we define a permutation function $Roll(p, o)$ which yields a cyclic shift of the original DUP perturbation by an offset o. In other words, when using $\{p_1, p_2, \ldots, p_{16}\}$ as input to $Roll(p, o)$, the output is $\{p_{16-o}, p_{16-o+1}, \ldots, p_{16}, p1, \ldots, p_{16-o-1}\}$. Now, for all values of $o \in \{0, 15\}$, we need $p_o = Roll(p, o)$ to be a valid perturbation clip as well. Towards achieving this requirement, we use a post-processor unit which applies the roll function between the generator and the discriminator. This post processor is captured in the complete architecture as shown in Figure 3b.

The details of how the generator and the roll unit operate in conjunction are depicted in Figure 8. As before, the 3D generator (G) takes a noise vector as input and outputs a sequence of perturbations (as a perturbation clip). Note that the final layer is followed by a $tanh$ non-linearity which constrains the perturbation generated to the range [-1,1]. The output is then scaled by ξ. Doing so restricts the perturbation's range to $[-\xi, \xi]$. Following the work in [29], [32], the value of ξ is chosen to be 10 towards making the perturbation quasi-imperceptible. The roll unit then "rolls" (cyclically shifts) the perturbation p by an offset in $\{0, 1, 2, \ldots 15\}$. Figure 8 depicts the process with an offset equal to 1; the black frame is rolled to the end of the clip. By adding the rolled perturbation clip to the training input, we get the perturbed input. As discussed earlier, the C3D classifier takes the perturbed input and outputs a classification score vector. As before, we want the true class scores to be (a) low for the targeted inputs and (b) high for other (non-targeted) inputs. We now modify our optimization function to incorporate the roll function as follows.

$$\begin{aligned}&\underset{G}{\text{minimize}}\\ &\quad \sum_{o=1,2\cdots w}\{\lambda \times \sum_{x_t \in T} -\log[1 - Q_{c(x_t)}(x_t + Roll(G(z), o))]\\ &+ \sum_{x_s \in S} -\log[Q_{c(x_s)}(x_s + Roll(G(z), o))]\}\end{aligned} \tag{3}$$

The equation is essentially the same as Equation 2, but we consider all possible cyclic shifts of the perturbation output by the generator.

D. 2D Dual-Purpose Universal Perturbation

We also consider a special case of C-DUP, wherein we impose an additional constraint which is that "the perturbations added to all frames are the same." In other words, we seek to add a single-frame 2D perturbation on each frame which can be seen as a special case of C-DUP with $p_1 = p_2 = \cdots = p_{16}$. We call this kind of C-DUP as 2D-DUP. 2D-DUP allows us to examine the effect of the temporal dimension in generating adversarial perturbations on video inputs. 2D-DUP is light-weight compared to C-DUP in terms of both transmission and storage costs. In addition, 2D-DUP allows other attack possibilities besides the man-in-the-middle case, an example being physically adding transparent foil (to add perturbation) onto the camera lens.

The generator in this case will output a single-frame perturbation instead of a sequence of perturbation frames as shown in Figure 9. This is a stronger constraint than the circular constraint, which may cause the attack success rate to decrease (note that the cyclic property still holds).

We denote the above 2D perturbation as p_{2d}. The perturbation clip is then generated by simply creating copies of the perturbation and *tiling* them to compose a clip. The 2D-DUP clip is now $p_{tile} = \{p_{2d}, p_{2d}, \ldots, p_{2d}\}$ (Figure 9). Thus, given that the attack objective is the same as before, we simply replace the $Roll(p, o)$ function with a $Tile$ function and our problem formulation now becomes:

$$\begin{aligned}\underset{G_{2D}}{\text{minimize}} \quad & \lambda \times \sum_{x_t \in T} -log[1 - Q_{c(x_t)}(x_t + Tile(G_{2D}(z)))]\\ & + \sum_{x_s \in S} -log[Q_{c(x_s)}(x_s + Tile(G_{2D}(z)))]\end{aligned} \tag{4}$$

VII. Evaluations

In this section, we showcase the efficacy of the perturbations generated by our proposed approaches on both the UCF-101 and Jester datasets.

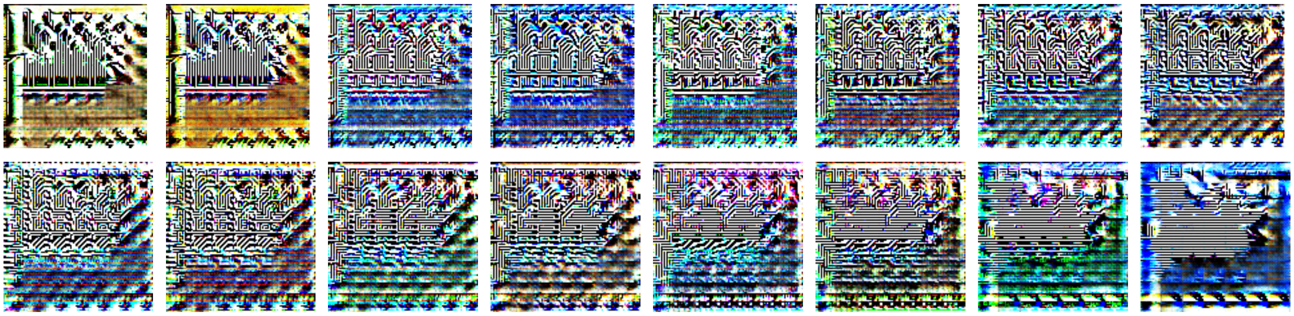


Fig. 10: DUP on UCF-101

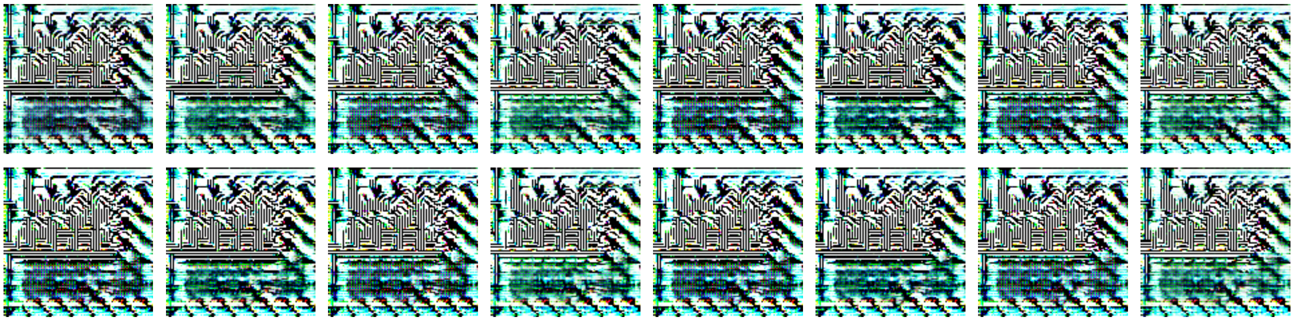


Fig. 11: C-DUP on UCF-101

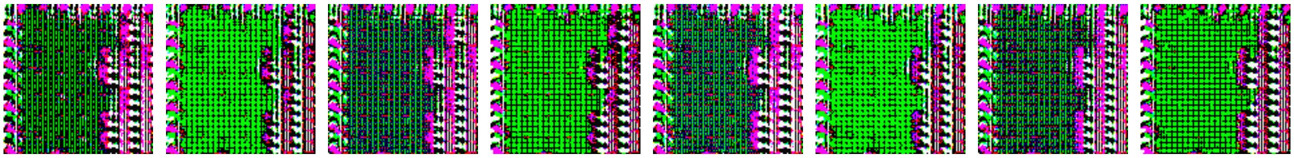


Fig. 12: C-DUP on Jester for $T_1 = \{\text{slding hand right}\}$

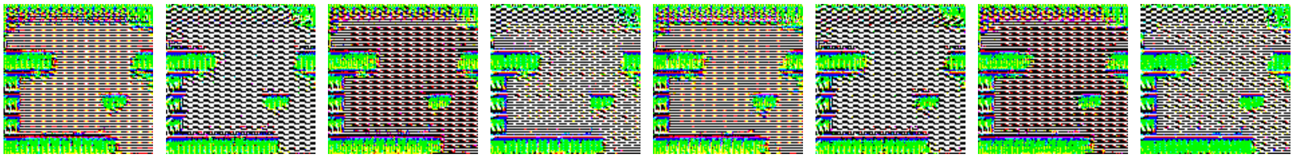


Fig. 13: C-DUP on Jester for $T_2 = \{\text{shaking hand}\}$

A. Experimental Setup

Discriminator set-up for our experiments: We used the C3D classifier as our discriminator. The discriminator is then used to train our generator. For our experiments on the UCF101 dataset, we use the C3D model available in the Github repository [44]. This pre-trained C3D model achieves an average clip classification accuracy of 91.8% on the UCF101 dataset in benign settings (i.e., no adversarial inputs). For the experiments on the Jester dataset, we fine-tune the C3D model from the Github repository [44]. First, we change the output size of the last fully connected layer to 27, since there are 27 gesture classes in Jester. We use a learning rate with exponential decay [57] to train the model. The starting learning rate for the last fully connected layer is set to be 10^{-3} and 10^{-4} for all the other layers. The decay step is set to 600 and the decay rate is 0.9. The fine-tuning phase is completed in 3 epochs and we achieve a clip classification accuracy of 90.03% in benign settings.

Generator set-up for our experiments: For building our generators, we refer to the generative model used by Vondrik *et al.* [53], which has 3D deconvolution layers.

For generators for both C-DUP and 2D-DUP, we use five 3D de-convolution layers [4]. The first four layers are followed by a batch normalization [18] and a $ReLU$ [33] activation function. The last layer is followed by a $tanh$ [19] layer. The kernel size for all 3D de-convolutions is set to be $3 \times 3 \times 3$. To generate 3D perturbations (i.e., sequence of perturbation frames), we set the kernel stride in the C-DUP generator to 1 in both the spatial and temporal dimensions for the first layer, and 2 in both the spatial and temporal dimensions for the following 4 layers. To generate a single-frame 2D perturbation, the kernel stride in the temporal dimension is set to 1 (i.e., 2D deconvolution) for all layers in the 2D-DUP generator, and the spatial dimension stride is 1 for the first layer and 2 for the following layers. The numbers of filters are shown in brackets in Figure 8 and Figure 9. The input noise vector for both generators are sampled from a uniform distribution $U[-1, 1]$ and the dimension of the noise vector is set to be 100. For training both generators, we use a learning rate with exponential decay. The starting learning rate is 0.002. The decay step is 2000 and the decay rate is 0.95. Unless otherwise specified, the weight balancing the two objectives, i.e., λ, is set to 1 to reflect equal importance between misclassifying the target class and retaining the correct classification for all the other (non-target) classes.

Technical Implementation: All the models are implemented in TensorFlow [1] with the Adam optimizer [23]. Training was performed on 16 Tesla K80 GPU cards with the batch size set to 32. The code is available at https://github.com/sli057/Video-Perturbation.git.

Dataset setup for our experiments: On the UCF-101 dataset (denoted UCF-101 for short), different sets of target class T are tested. We use $T = \{\text{apply lipstick}\}$ for presenting the results in the paper. Experiments using other target sets also yield similar results. UCF-101 has 101 classes of human actions in total. The target set T contains only one class while the "non-target" set $S = X - T$ contains 100 classes. The number of training inputs from the non-target classes is approximately 100 times the number of training inputs from the target class. Directly training with UCF-101 may cause a problem due to the imbalance in the datasets containing the target and non-target classes [26]. Therefore, we under-sample the non-target classes by a factor of 10. Further, when loading a batch of inputs for training, we fetch half the batch of inputs from the target set and the other half from the non-target set in order to balance the inputs.

For the Jester dataset, we also choose different sets of target classes. We use two target sets $T_1 = \{\text{sliding hand right}\}$ and $T_2 = \{\text{shaking hands}\}$ as our representative examples because they are exemplars of two different scenarios. Since we seek to showcase an attack on a video classification system, we care about how the perturbations affect both the appearance information and temporal flow information, especially the latter. For instance, the 'sliding hand right' class has a temporally similar class 'sliding two fingers right;' as a consequence, it may be easier for attackers to cause clips in the former class to be misclassified as the later class (because the temporal information does not need to be perturbed much). On the other hand, 'shaking hands' is not temporally similar to any other class. Comparing the results of these two target sets could provide some empirical evidence on the impact of the temporal flow on our perturbations. Similar to UCF-101, the number of inputs from the non-target classes is around 26 times the number of inputs from the target class (since there are 27 classes in total and we only have one target class in each experiment). So we under-sample the non-target inputs by a factor of 4. We also set up the environment to load half of the inputs from the target set and the other half from the non-target set, in every batch during training.

Metrics of interest: For measuring the efficacy of our perturbations, we consider two metrics. *First,* the perturbations added to the videos should be quasi-imperceptible. *Second*, the attack success rate for the target and the non-target classes should be high. We define attack success rates as follows:

- The attack success rate for the target class is the misclassification rate.
- The attack success rate for the other classes is the correct classification rate.

B. Stealth with DUP

Recalling the discussion in §V, one can expect that UP would cause inputs from the target class to be misclassified, but also significantly affect the correct classification of the other non-target inputs. On the other hand, one would expect that DUP would achieve a stealthy attack, which would not cause much effect on the classification of non-target classes.

By testing on UCF-101 with "apply lipstick" as the target class, we observe that with UP, "archery" is misclassified as "swing," "baby crawling" is misclassified as "cutting in kitchen," "biking" is misclassified as "golf swing," and so on. We find that only 45.2% of the video clips from non-target classes are classified correctly, i.e., the attack success rate for non-target inputs is only 45.2%. This violates the stealthiness needed to successfully launch an attack. However, DUP does not affect the classification of non-target inputs much; the non-target attack success rate is 88.03%. At the same time, both UP and DUP work well on target inputs, which means the perturbed target clips are misclassified at high rate. DUP achieves a attack success rate of 84.49 % for target inputs and UP achieves 84.01%. These results are obtained under the assumption that clip boundaries are exactly known while performing the attack. Given the inferior performance of UP on non-target inputs (i.e., in preserving stealth), we do not consider it any further in our evaluations.

C. Showcasing C-DUP

In this subsection, we discuss the results of the C-DUP perturbation attack. We use DUPs as our baselines.

1) Experimental Results on UCF101:

Visualizing the perturbations: The perturbation clip generated by the DUP model is shown in Figure 10 and the perturbation clip generated by C-DUP model is shown in Figure 11. The visualizations of all perturbations are scaled from [0,10] to [0,255]. We observe that the perturbation from DUP manifests an obvious disturbance among the frames. With C-DUP, the perturbation frames look similar, which implies that C-DUP does not perturb the temporal information by much, in UCF101.

Impact of misalignment and C-DUP performance: Based on the discussion in §VI, we expect that DUP would work well only when the perturbation clip is well-aligned with the start point of each input clip to the classifier; and the attack success rate would degrade as the misalignment increases. We expect C-DUP would overcome the misalignment effect and provide a better overall attack performance (even with temporal misalignment).

Case study: We perform a case study to showcase the impact of the misalignment. We consider one "apply lipstick" video clip for our case study. When DUP and C-DUP are added to this clip without any offset (no misalignment) i.e., the clip is in the form $[f_1, f2, \cdots, f_{16}]$, both perturbed clips are misclassified to "apply eye makeup". When there is an offset of 8, meaning that DUP and C-DUP are added to the clip in the form $[f_9, f_{10}, \cdots, f_{16}, f_1, \cdots, f_8]$, DUP fails to misclassify the clip while C-DUP still successfully misclassifies it. In fact, we observe that C-DUP works for all offsets from 0 to 15 while DUP only works when the offset = 0, 1, 2, 15, on this input clip.

Aggregate results: The attack success rates with DUP and C-DUP, on the UCF-101 test set, are shown in Figure 14a and Figure 14b. The x axis is the misalignment between the perturbation clip and the input clip to the classifier. Figure 14a depicts the average attack success rate for inputs from the target class. We observe that when there is no misalignment, the attack success rate with DUP is 84.49%, which is in fact slightly higher than C-DUP. However, the attack success rate with C-DUP is significantly higher when there is misalignment. Furthermore, the average attack success rate across all alignments for the target class with C-DUP is 84%, while with DUP it is only 68.26%. This demonstrates that C-DUP is more robust against misalignment.

Figure 14b shows that, with regard to the classification of inputs from the non-target classes, C-DUP also achieves a performance slightly better than DUP when there is mismatch. The average attack success rate (across all alignments) with C-DUP is 87.52% here, while with DUP it is 84.19%.

2) Experimental Results on Jester:

Visualizing the perturbations: Visual representations of the C-DUP perturbations for the two target sets, $T_1 = \{\text{sliding hand right}\}$ and $T_2 = \{\text{shaking hands}\}$ are shown in Figure 12 and Figure 13. The perturbation clip has 16 frames, and we present a visual representation of the first 8 frames for compactness. We notice that compared to the perturbation generated on UCF-101 (see Figure 11). there is a more pronounced evolution with respect to Jester. We conjecture that this is because UCF-101 is a coarse-grained action dataset in which the spatial (appearance) information is dominant. As a consequence, the C3D model does not extract/need much temporal information to perform well. However, Jester is a fine-grained action dataset where temporal information plays a more important role. Therefore, in line with expectations, we find that in order to attack the C3D model trained on the Jester dataset, more significant evolutions of the perturbations on the frames in a clip are required (i.e., more changes in the temporal dimension).

Attack success rate: To showcase a comparison of the misclassification rates with respect to the target class between the two schemes (DUP and C-DUP), we adjust the weighting factor λ such that the classification accuracy with respect to non-target classes are similar. By choosing λ = 1.5 for DUP and 1 for C-DUP, we are able to achieve this. The attack success rates for the above two target sets are shown in Figure 14c and Figure 14d, *and* Figure 14e and Figure 14f, respectively. We see that with respect to $T_1 = \{\text{sliding hand right}\}$, the results are similar to what we observe with UCF101. The attack success rates for C-DUP are a little lower than those for DUP when the offset is 0. This is to be expected since DUP is tailored for this specific offset. However, C-DUP outperforms DUP when there is a misalignment. The average success rate for C-DUP is 85.14% for the target class and 81.03% for the other (non-target) classes. The average success rate for DUP is 52.42% for the target class and 82.36% for the other (non-target) classes.

Next we consider the case with $T_2 = \{\text{shaking hands}\}$. In general, we find that both DUP and C-DUP achieve relatively lower success rates especially with regard to the other (non-target) classes. As discussed in §VII-A, unlike in the previous case where 'sliding two fingers right' is temporally similar to 'sliding hand right', no other class is temporally similar to 'shaking hand'. Therefore it is harder to achieve misclassification. The attack success rates with the two approaches for the target class are shown in Figure 14e. We see that C-DUP significantly outperforms DUP in terms of attack efficacy because of its robustness to temporal misalignment (i.e., the boundary effect). The average attack success rate for the target class with C-DUP is 79.03% while for DUP it is only 57.78%. Overall, our C-DUP outperforms DUP in being able to achieve a better attack success rate for the target class. We believe that although stealth is affected to some extent, it is still reasonably high.

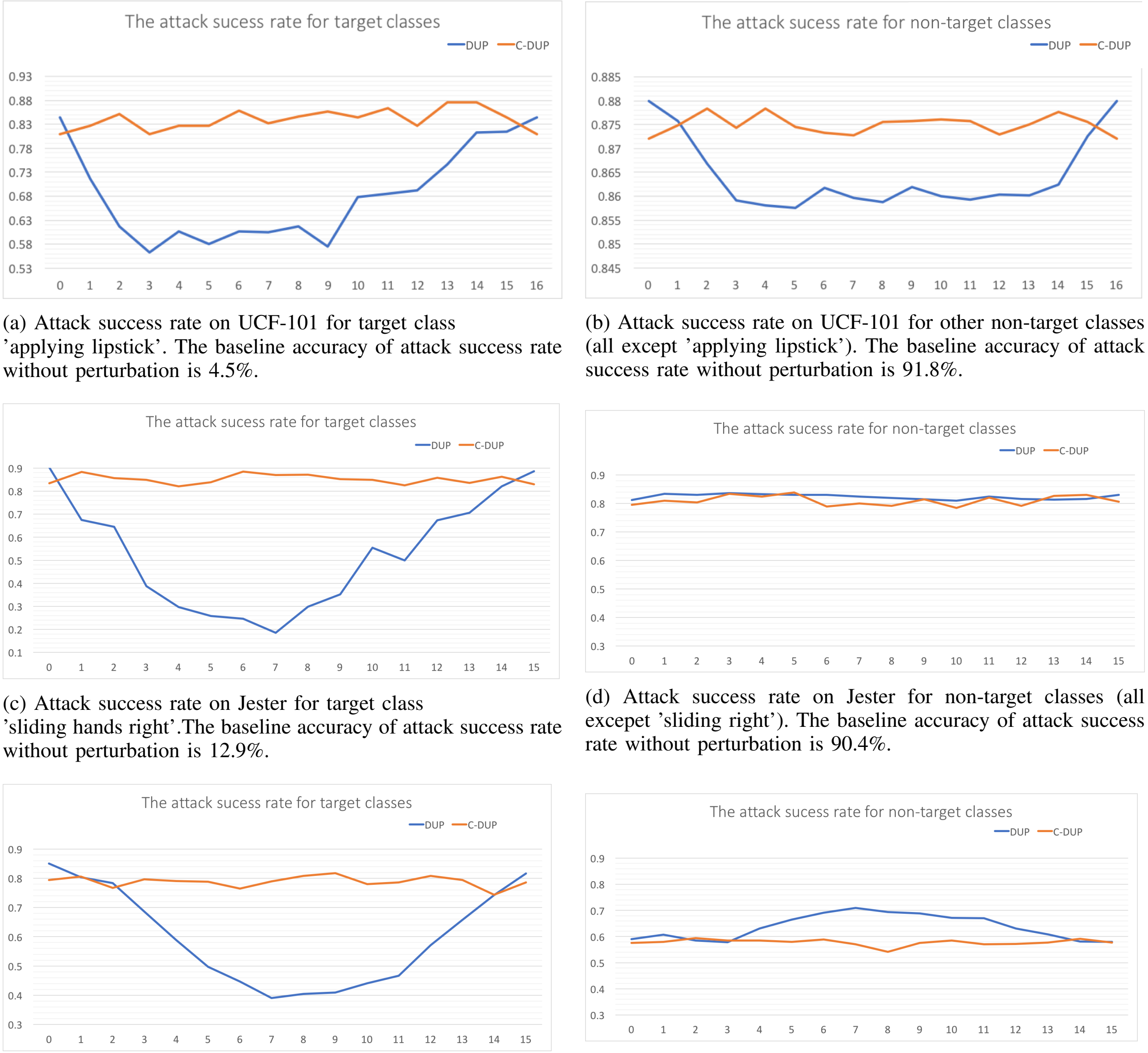
D. Effectiveness of 2D-DUP

The visual representations of the perturbations with C-DUP show that perturbations on all the frames are visually similar. Thus, we ask if it is possible to add "the same perturbation" on every frame and still achieve a successful attack. In other words, will the 2D-DUP perturbation attack yield performance similar to the C-DUP attack ?

1) Experimental Results on the UCF101 Dataset:

Visual impact of the perturbation: We present a sequence of original frames and its corresponding perturbed frames in Figure 15. Original frames are displayed in the first row and perturbed frames are displayed in the second row. We observe that the perturbation added to the frames is quasi-imperceptible to human eyes (similar results are seen with C-DUP but are omitted in the interest of compactness).

Attack success rate: By adding 2D-DUP on the video clip, we achieve an attack success rate of 87.58% with respect to the target class and an attack success rate of 83.37% for the non-target classes. Recall that the average attack success rates with C-DUP were 87.52% and 84.00%, respectively. Thus, the performance of 2D-DUP seems to be on par with that of C-DUP on the UCF101 dataset. This demonstrates that C3D is



(a) Attack success rate on UCF-101 for target class 'applying lipstick'. The baseline accuracy of attack success rate without perturbation is 4.5%.

(b) Attack success rate on UCF-101 for other non-target classes (all except 'applying lipstick'). The baseline accuracy of attack success rate without perturbation is 91.8%.

(c) Attack success rate on Jester for target class 'sliding hands right'.The baseline accuracy of attack success rate without perturbation is 12.9%.

(d) Attack success rate on Jester for non-target classes (all excepet 'sliding right'). The baseline accuracy of attack success rate without perturbation is 90.4%.

(e) Attack success rate on Jester for target class 'shaking hand'. The baseline accuracy of attack success rate without perturbation is 6.3%.

(f) Attack success rate on Jester for non-target classes (all except 'shaking hand'). The baseline accuracy of attack success rate without perturbation is 89.9%.

Fig. 14: Attack success rates for DUP and C-DUP along with the offset of mismatch

vulnerable even if the same 2D perturbation generated by our approach is added to every frame.

2) *Experimental Results on Jester Dataset:*

Attack success rate: For $T_1 = \{\text{sliding hand right}\}$, the attack success rate for the target class is 84.64% and the attack success rate for the non-target classes is 80.04%. This shows that 2D-DUP is also successful on some target classes in the fine-grained, Jester action dataset.

For the target set T_2, the success rate for the target class drops to 70.92%, while the success rate for non-target class is 54.83%. This is slightly degraded compared to the success rates achieved with C-DUP (79.03% and 57.78% respectively), but is still reasonable. This degradation is due to more significant temporal changes in this case (unlike in the case of T_1) and a single 2D perturbation is less effective in manipulating these changes. In contrast, because the perturbations within C-DUP evolve, they are much more effective in achieving the misclassification of the target class.

VIII. Discussion

Black box attacks: In this work we assumed that the adversary is fully aware of the DNN being deployed (i.e., white box attacks). We argue that this is reasonable given that this is one

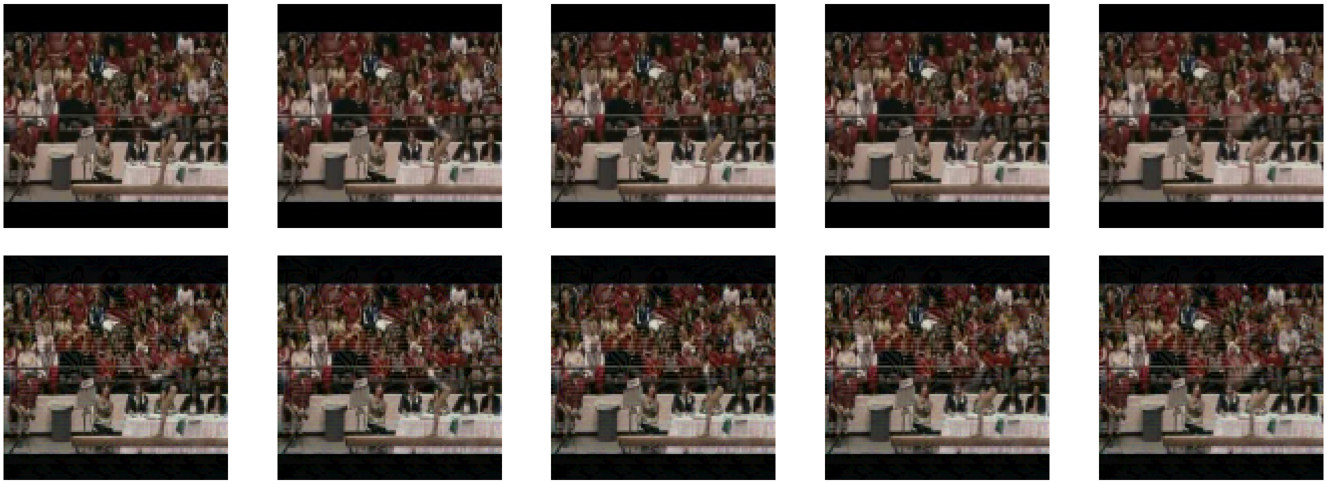


Fig. 15: Visualizing images after adding 2D dual purpose universal perturbation: Original frames are displayed in the first row and perturbed frames are displayed in the second row. The perturbation added to the frames in the second row is mostly imperceptible to the human eye.

of the first efforts on generating adversarial perturbations on real-time video classification systems. However, in practice the adversary may need to determine the type of DNN being used in the video classification system, and so a black box approach may be needed. Given recent studies on the transferability of adversarial inputs [36], we believe black box attacks are also feasible. We will explore this in our future work.

Context dependency: Second, the approach that we developed does not account for contextual information, i.e., consistency between the misclassified result and the context. While in some cases with a limited set of classes (e.g., actions possible at an elderly care facility), this may be not matter, in some other cases a loss in context may cause a human operator to notice discrepancies. For example, if the context relates to a baseball game, a human overseeing the system may notice an inconsistency when the action of hitting a ball is misclassified into applying makeup. Similarly, because of context, if there is a series of actions that we want to misclassify, inconsistency in the misclassification results (e.g., different actions across the clips) may also raise an alarm. For example, let us consider a case where the actions include running, kicking a ball, and applying make up. While the first two actions can be considered to be *reasonable* with regard to appearing together in a video, the latter two are unlikely. Generating perturbations that are consistent with the context of the video is a line of future work that we will explore and is likely to require new techniques. In fact, looking for consistency in context may be a potential defense, and we will also examine this in depth in the future.

Data Augmentation: We point out here that for both UPs and DUPs, the training set included all possible strides (data augmentation). Unfortunately, the issues relating to the boundary effect cannot be solved by data augmentation. In particular, recall that the misalignment due to the nondeterminism in clip boundaries input to the classifier cause the perturbation clips added by the attacker to be broken up. While UPs are effective on any video clip, concatenations of broken up UPs are no longer UPs and thus, are not effective.

Defenses: In order to defend against the attacks against video classification systems, one can try some existing defense methods in image area, such as feature squeezing [55], [56] and ensemble adversarial training [45] (although their effectiveness is yet unknown). Considering the properties of video that were discussed, we envision some exclusive defense methods for protecting video classification systems below, which we will explore in future work.

One approach is to examine the consistency between the classification of consecutive frames (considered as images) within a clip, and between consecutive clips in a stream. A sudden change in the classification results could raise an alarm. However, while this defense will work well in cases where the temporal flow is not pronounced (e.g., the UCF101 dataset), it may not work well in cases with pronounced temporal flows. For example, with respect to the Jester dataset, with just an image it may be hard to determine whether the hand is being moved right or left.

The second line of defense may be to identify an object that is present in the video, e.g., a soccer ball in a video clip that depicts a kicking action. We can use an additional classifier to identify such objects in the individual frames that compose the video. Then, we can look for consistency with regard to the action and the object, e.g., a kicking action can be associated with a soccer ball, but cannot be associated with a make up kit. Towards realizing this line of defense, we could use existing image classifiers in conjunction with the video classification system. We will explore this in future work.

IX. Related Work

There is quite a bit of work [2], [3], [16] on investigating the vulnerability of machine learning systems to adversarial inputs. Researchers have shown that generally, small magnitude perturbations added to input samples, change the predictions made by machine learning models. Most efforts, however, do not consider real-time temporally varying inputs such as video.

Unlike these efforts, our study is focused on the generation of adversarial perturbations to fool DNN based real-time video action recognition systems.

The threat of adversarial samples to deep-learning systems has also received considerable attention recently. There are several papers in the literature (e.g., [10], [11], [29], [30], [39]) that have shown that the state-of-the-art DNN based learning systems are also vulnerable to well-designed adversarial perturbations [43]. Szegedy *et al.*show that the addition of hardly perceptible perturbation on an image, can cause a neural network to misclassify the image. Goodfellow *et al.* [11] analyze the potency of adversarial samples available in the physical world, in terms of fooling neural networks. Moosavi-Dezfooli *et al.* [29]–[31] make a significant contribution by generating image-agnostic perturbations, which they call universal adversarial perturbations. These perturbations can cause all natural images belonging to target classes to be misclassified with high probability.

There are very few recent studies [15], [54] which explore the feasibility of adversarial perturbation on videos. Hosseini et al. [15] attack the Google Cloud Video Intelligence API, which makes decisions only based on the first frame of every second of the video, by inserting images/perturbing frames at the rate of one frame per second. This attack method cannot be generalized to the common case where video classification systems use sequences of consecutive frames to perform activity recognition. In addition, the authors assume that the starting frame used by the API is known to the attacker, which in real-time applications is not deterministic (and thus, is unknown). Wei et al. [54] attack the video recognition system by adding perturbations only on the first few consecutive frames in a video clip. However, unlike our attack, these attacks do not work on practical real-time video classification systems when the boundaries of video clips are not known.

GANs or generative adversarial networks have been employed by Goodfellow *et al.* [10] and Radford *et al.* [38] in generating natural images. Mopuri *et al.* [32] extend a GAN architecture to train a generator to model universal perturbations for images. Their objective was to explore the space of the distribution of universal adversarial perturbations in the image space. We significantly extend the generative framework introduced by Mopuri *et al.* [32]. In addition, unlike their work which focused on generating adversarial perturbations for images, our study focuses on the generation of effective perturbations to attack videos.

The feasibility of adversarial attacks against other types of learning systems including face-recognition systems [28], [39], [40], voice recognition systems [5] and malware classification systems [12], has been studied. However, these studies do not account for the unique input characteristics that are present in real-time video activity recognition systems.

X. Conclusions

In this paper, we investigate the problem of generating adversarial samples for attacking video classification systems. We identify three key challenges that will need to be addressed in order to generate such samples namely, generating perturbations in real-time, making the perturbations stealthy and dealing with the intedeterminism of video clip boundaries that are input to a real-time video classifier. We exploit recent advances in generative models, extending them significantly to solve these challenges and generate very potent adversarial samples against video classification systems. We perform extensive experiments on two different datasets one of which captures coarse-grained actions (e.g., applying make up) while the other captures fine-grained actions (hand gestures). We demonstrate that our approaches are extremely potent, achieving around 80 % attack success rates in both cases. We also discuss possible defenses that we propose to investigate in future work.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable feedback on this paper. This work was partially supported by the U.S. Army Research Laboratory Cyber Security Collaborative Research Alliance under Cooperative Agreement Number W911NF-13-2-0045. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to re-produce and distribute reprints for Government purposes, notwithstanding any copyright notation hereon.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[2] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2013, pp. 387–402.

[3] B. Biggio, G. Fumera, and F. Roli, "Pattern recognition systems under attack: Design issues and research challenges," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 28, no. 07, p. 1460002, 2014.

[4] D. S. Biggs, "3d deconvolution microscopy," *Current Protocols in Cytometry*, pp. 12–19, 2010.

[5] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, "Hidden voice commands." in *USENIX Security Symposium*, 2016, pp. 513–530.

[6] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 4724–4733.

[7] J. Dataset, "Humans performing pre-defined hand actions," https://20bn.com/datasets/jester, 2016, [Online; accessed 30-April-2018].

[8] S. R. Fanello, I. Gori, G. Metta, and F. Odone, "One-shot learning for real-time action recognition," in *Iberian Conference on Pattern Recognition and Image Analysis*. Springer, 2013, pp. 31–40.

[9] H. Foroughi, B. S. Aski, and H. Pourreza, "Intelligent video surveillance for monitoring fall detection of elderly in home environments," in *Computer and Information Technology, 2008. ICCIT 2008. 11th International Conference on*. IEEE, 2008, pp. 219–224.

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

[11] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples (2014)," *arXiv preprint arXiv:1412.6572*.

[12] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial perturbations against deep neural networks for malware classification," *arXiv preprint arXiv:1606.04435*, 2016.

[13] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognition*, 2017.

[14] S. Herath, M. Harandi, and F. Porikli, “Going deeper into action recognition: A survey,” *Image and vision computing*, vol. 60, pp. 4–21, 2017.

[15] H. Hosseini, B. Xiao, A. Clark, and R. Poovendran, “Attacking automatic video analysis algorithms: A case study of google cloud video intelligence api,” in *Proceedings of the 2017 on Multimedia Privacy and Security*. ACM, 2017, pp. 21–32.

[16] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar, “Adversarial machine learning,” in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*. ACM, 2011, pp. 43–58.

[17] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie, “Stacked generative adversarial networks,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2017, p. 4.

[18] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.

[19] B. L. Kalman and S. C. Kwasny, “Why tanh: choosing a sigmoidal function,” in *Neural Networks, 1992. IJCNN., International Joint Conference on*, vol. 4. IEEE, 1992, pp. 578–581.

[20] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale video classification with convolutional neural networks,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[21] H. Kataoka, Y. Satoh, Y. Aoki, S. Oikawa, and Y. Matsui, “Temporal and fine-grained pedestrian action recognition on driving recorder database,” *Sensors*, vol. 18, no. 2, p. 627, 2018.

[22] H. Kataoka, T. Suzuki, S. Oikawa, Y. Matsui, and Y. Satoh, “Drive video analysis for the detection of traffic near-miss incidents,” *arXiv preprint arXiv:1804.02555*, 2018.

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

[24] A. Kurakin, I. Goodfellow, and S. Bengio, “Adversarial machine learning at scale,” *arXiv preprint arXiv:1611.01236*, 2016.

[25] K. Lab, “Man-in-the-middle attack on video surveillance systems,” https://securelist.com/does-cctv-put-the-public-at-risk-of-cyberattack/70008/, Defcon,2014, [Online; accessed 30-April-2018].

[26] R. Longadge and S. Dongre, “Class imbalance problem in data mining review,” *arXiv preprint arXiv:1305.1707*, 2013.

[27] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” *arXiv preprint arXiv:1706.06083*, 2017.

[28] M. McCoyd and D. Wagner, “Spoofing 2d face detection: Machines see people who aren’t there,” *arXiv preprint arXiv:1608.02128*, 2016.

[29] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, “Universal adversarial perturbations,” in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 86–94.

[30] S. M. Moosavi Dezfooli, A. Fawzi, and P. Frossard, “Deepfool: a simple and accurate method to fool deep neural networks,” in *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, no. EPFL-CONF-218057, 2016.

[31] K. R. Mopuri, U. Garg, and R. V. Babu, “Fast feature fool: A data independent approach to universal adversarial perturbations,” *arXiv preprint arXiv:1707.05572*, 2017.

[32] K. R. Mopuri, U. Ojha, U. Garg, and R. V. Babu, “Nag: Network for adversary generation,” *arXiv preprint arXiv:1712.03390*, 2017.

[33] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[34] Z. Net, “Surveillance cameras sold on Amazon infected with malware,” https://www.zdnet.com/article/amazon-surveillance-cameras-infected-with-malware/, ZD Net,2016, [Online; accessed 30-April-2018].

[35] N. Papernot, N. Carlini, I. Goodfellow, R. Feinman, F. Faghri, A. Matyasko, K. Hambardzumyan, Y.-L. Juang, A. Kurakin, R. Sheatsley *et al.*, “cleverhans v2. 0.0: an adversarial machine learning library,” *arXiv preprint arXiv:1610.00768*, 2016.

[36] N. Papernot, P. McDaniel, and I. Goodfellow, “Transferability in machine learning: from phenomena to black-box attacks using adversarial samples,” *arXiv preprint arXiv:1605.07277*, 2016.

[37] R. Planinc, A. Chaaraoui, M. Kampel, and F. Flrez-Revuelta, “Computer vision for active and assisted living,” pp. 57–79, 01 2016.

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

[39] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, “Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1528–1540.

[40] ——, “Adversarial generative nets: Neural network attacks on state-of-the-art face recognition,” *arXiv preprint arXiv:1801.00349*, 2017.

[41] K. Soomro, A. R. Zamir, and M. Shah, “Ucf101: A dataset of 101 human actions classes from videos in the wild,” *arXiv preprint arXiv:1212.0402*, 2012.

[42] W. Sultani, C. Chen, and M. Shah, “Real-world anomaly detection in surveillance videos,” *arXiv preprint arXiv:1801.04264*, 2018.

[43] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” *arXiv preprint arXiv:1312.6199*, 2013.

[44] C. Tensorflow, “C3D Implementation,” https://github.com/hx173149/C3D-tensorflow.git, 2016, [Online; accessed 30-April-2018].

[45] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel, “Ensemble adversarial training: Attacks and defenses,” *arXiv preprint arXiv:1705.07204*, 2017.

[46] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3d convolutional networks,” in *Computer Vision (ICCV), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4489–4497.

[47] V. Tripathi, A. Mittal, D. Gangodkar, and V. Kanth, “Real time security framework for detecting abnormal events at atm installations,” *Journal of Real-Time Image Processing*, pp. 1–11, 2016.

[48] G. Varol, I. Laptev, and C. Schmid, “Long-term temporal convolutions for action recognition,” *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[49] U. C. Vision, “Case Study: Elementary Scholl in Taiwai,” https://news.umbocv.com/case-study-taiwan-elementary-school-13fa14cdb167.

[50] ——, “Umbo Customer Case Study NCHU,” https://news.umbocv.com/umbo-customer-case-study-nchu-687356292f43.

[51] ——, “Umbo’s Smart City Featured on CBS Sacramento,” https://news.umbocv.com/umbos-smart-city-featured-on-cbs-sacramento-26f839415c51.

[52] ——, “Case Studies,” https://news.umbocv.com/case-studies/home, 2016, [Online; accessed 30-April-2018].

[53] C. Vondrick, H. Pirsiavash, and A. Torralba, “Generating videos with scene dynamics,” in *Advances In Neural Information Processing Systems*, 2016, pp. 613–621.

[54] X. Wei, J. Zhu, and H. Su, “Sparse adversarial perturbations for videos,” *arXiv preprint arXiv:1803.02536*, 2018.

[55] W. Xu, D. Evans, and Y. Qi, “Feature squeezing: Detecting adversarial examples in deep neural networks,” *arXiv preprint arXiv:1704.01155*, 2017.

[56] ——, “Feature squeezing mitigates and detects carlini/wagner adversarial examples,” *arXiv preprint arXiv:1705.10686*, 2017.

[57] M. D. Zeiler, “Adadelta: an adaptive learning rate method,” *arXiv preprint arXiv:1212.5701*, 2012.