Universal 3-Dimensional Perturbations for Black-Box Attacks on Video Recognition Systems

Shangyu Xie[1], Han Wang[1], Yu Kong[2], Yuan Hong[1]
[1]Illinois Institute of Technology, Chicago, IL 60616
[2]Rochester Institute of Technology, Rochester, NY 14623
Email: {sxie14, hwang185}@hawk.iit.edu, yu.kong@rit.edu, yuan.hong@iit.edu

***Abstract*—Widely deployed deep neural network (DNN) models have been proven to be vulnerable to adversarial perturbations in many applications (e.g., image, audio and text classifications). To date, there are only a few adversarial perturbations proposed to deviate the DNN models in video recognition systems by simply injecting 2D perturbations into video frames. However, such attacks may overly perturb the videos without learning the spatio-temporal features (across temporal frames), which are commonly extracted by DNN models for video recognition. To our best knowledge, we propose the first black-box attack framework that generates universal 3-dimensional (U3D) perturbations to subvert a variety of video recognition systems. U3D has many advantages, such as (1) as the transfer-based attack, U3D can universally attack multiple DNN models for video recognition without accessing to the target DNN model; (2) the high transferability of U3D makes such universal black-box attack easy-to-launch, which can be further enhanced by integrating queries over the target model when necessary; (3) U3D ensures human-imperceptibility; (4) U3D can bypass the existing state-of-the-art defense schemes; (5) U3D can be efficiently generated with a few pre-learned parameters, and then immediately injected to attack *real-time* DNN-based video recognition systems. We have conducted extensive experiments to evaluate U3D on multiple DNN models and three large-scale video datasets. The experimental results demonstrate its superiority and practicality.**

I. Introduction

Deep neural network (DNN) models have been extensively studied to facilitate a wide variety of intelligent video recognition systems, such as face recognition [88], action recognition [23] and anomaly detection [69]. For instance, self-driving vehicles are equipped with many cameras to capture the visual information. Then, DNNs are adopted to accurately recognize road signs, detect and predict trajectories of pedestrians and vehicles, and thus make the driving decisions [54], [60]. Video anomaly detection systems [10], [69] integrate DNNs to monitor the activities under surveillance, and trigger alarms once anomalies (e.g., traffic accident, theft, and arson) are visually identified to advance the public safety.

However, DNNs have been revealed to be inherently vulnerable to adversarial attacks, where attackers can add well-crafted imperceptible perturbations to the inputs to deviate the learning results. Such attacks are initially identified in the image domain [17], [49]–[51], [73], and have also attracted significant interests in other contexts, e.g., text understanding [42], [70], and voice recognition [7], [11], [44]. Similarly, adversarial perturbations to the DNNs in video recognition systems could potentially cause severe physical and financial damages. For instance, they may misdirect the DNN models in autonomous vehicles to inaccurately recognize objects and make detrimental decisions towards accidents. Furthermore, DNN-based anomaly detection models in video surveillance or CCTV might be deviated via the perturbations to misclassify anomalous activities to routine ones, and vice-versa [69].

Although the adversarial attacks on images have been well-explored, there are very limited works on attacking DNN models for videos [32], [43], [82], [83], which need to address additional challenges, e.g., larger data sizes, a new set of DNN models for learning actions in the videos, different types of features extracted with additional temporal convolution, and different realizability. To our best knowledge, current video attacks [32], [43], [82], [83] adapt image perturbations in a frame-by-frame fashion to subvert DNNs for video classification, which have the following major limitations.

1) Frame-by-frame image perturbations may overly perturb the videos (human perceptible), and also lack the temporal consistency in the perturbations. These make the attacks not robust against the state-of-the-art detection schemes (e.g., AdvIT [85]). Adversarial examples crafted by [32], [43], [83] can be accurately detected by AdvIT (as evaluated in our experiments).
2) Frame-by-frame image perturbations may not be well aligned with the video frames (*boundary effect* by misaligning the perturbation and video frames) [43].
3) Crafting adversarial examples for videos frame-by-frame results in heavy computation overheads and lacks universality. It limits the application to attack large-scale videos or streaming videos (e.g., CCTV surveillance).

To address the above limitations, we propose a black-box attack framework that generates *universal 3-dimensional (U3D)* perturbations to subvert a wide variety of video recognition systems. U3D has the following major advantages: (1) as a transfer-based black-box attack [15], [46], [56], U3D can universally attack multiple DNN models for video recognition (each of which can be considered as the target model) without accessing to the target DNN models; (2) the high transferability of U3D makes such black-box attacks easy-to-launch, which can be further enhanced by integrating queries over the target model when necessary (validated); (3) U3D ensures

good human-imperceptibility (validated by human survey); (4) U3D can bypass the existing state-of-the-art defense schemes (extended towards defending against U3D), including universal adversarial training (UAT) [47], [63], detection schemes [85], [89], and certified schemes (e.g., PixelDP [41] and randomized smoothing [18]); (5) U3D perturbations can be generated on-the-fly with very low computation overheads (e.g., $\sim$0.015s per frame) to attack DNN models for streaming videos.

Specifically, in the attack design, we generate perturbations by maximally deviating features over the feature space representation of the DNNs while strictly bounding the maximum perturbations applied to the videos. We aim at generating more transferable adversarial examples (to be misclassified by multiple DNN models) by explicitly optimizing the attack performance w.r.t. layer-wise features of a video DNN model. Moreover, we integrate boundary effect mitigation and universality into the optimization for learning the U3D perturbations.

Different from traditional black-box attacks that may request intensive queries over the target DNN model, U3D perturbations can be efficiently derived independent of the target DNN model. Assuming that the adversary does not need to know the target DNN model under the black-box setting (and no need to query over the target model by default), our U3D attack computes the perturbation using a surrogate DNN model (any public DNN model, *which can have very different model structure and parameters from the target model*). Such black-box attacks are realized via high transferability across multiple DNN models on different datasets (as validated in Section V-C). We have also shown that our U3D attack can integrate queries over target model when necessary (turning into a hybrid black-box attack [71]).

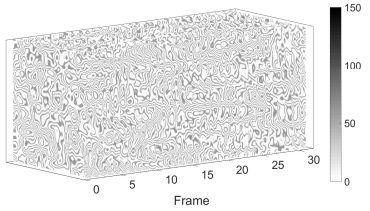


Fig. 1: Universal 3-dimensional (U3D) perturbation

Figure 1 demonstrates an example of the U3D perturbation, which is continuously generated. Compared to the state-of-the-art universal perturbations (see Section V), U3D achieves higher success rates with significantly less perturbations (mostly between [0,10] in grayscale [0,255]). It is also highly efficient for attacking multiple video recognition systems (e.g., classification and real-time anomaly detection). Therefore, we summarize our main contributions as below:

- To our best knowledge, we propose the first black-box attack that generates 3D perturbations to universally subvert multiple DNN-based video recognition systems.
- We construct two different types of novel U3D perturbations optimized in the feature space representation of DNNs, which can practically attack various DNN models and the related video recognition systems (e.g., classification and anomaly detection) with high transferability.
- We conduct extensive experiments to validate the U3D attack while benchmarking with the state-of-the-art attacks (e.g., C-DUP [43], V-BAD [32] and H-Opt [83]). Evaluations include success rate, transferability, universality, human-imperceptibility, performance against defenses, physical realization, and efficiency. The results have shown the superiority and practicality of U3D.
- In particular, we also evaluate the U3D against different types of state-of-the-art defense schemes. We have extensively adapted the defenses w.r.t. U3D, and studied the potential mitigation of the U3D. The high attack performance against defenses reveals the potential severity of the adversarial attack and the vulnerabilities in the DNN-based video recognition systems. Our novel U3D attack can facilitate the development of more robust and trustworthy DNN models for video recognition.

The remainder of this paper is organized as follows. We first briefly introduce the video recognition systems and define our threat model in Section II. Section III presents the U3D design goals and attack framework. Then, we give the detailed design of the U3D attack in Section IV. Section V demonstrates the experimental results on real datasets. Section VI discusses the mitigation of the U3D attack. Section VII reviews the related works. Finally, we draw conclusions in Section VIII.

II. Background

A. DNN-based Video Recognition Systems

DNNs have been widely adopted for accurate video recognition in numerous real-world applications, e.g., anomaly detection [69], self-driving vehicles [54] and smart security cameras [35]. There have been a series of works on designing video DNNs to improve model accuracy [20], [34], [65], [76]. For instance, Donahue et al. [20] proposed the long-term recurrent convolutional networks (LRCNs) for video recognition and description via combining convolutional layers and long-range temporal recursion. Moreover, two-stream network (TSN) [65] fusing static frames and optical flows was proposed for action recognition. Later, Tran et al. [76] proposed the C3D model to significantly improve classification accuracy by focusing on spatio-temporal feature learning with 3D convolutional neural network. Recently, more networks built on spatio-temporal convolutions (e.g., I3D [9]) have been exhibited high performance, which greatly promoted the video recognition systems. Two example applications are demonstrated in Appendix E.

B. Threat Model

Attack Scenarios. The U3D attack is applicable to the *offline* scenario, which is identical to the attack scenario of adversarial perturbations for other types of data, e.g., images [17], [49], [51], texts [42], and audio signals [7], [11], [44]. For instance, the adversary can craft adversarial examples by adding the pre-generated U3D perturbations to static videos. Then, the perturbed videos will be misclassified to wrong labels.

Furthermore, our U3D attack can work *online* to perturb the streaming video (e.g., real-time anomaly detection in CCTV surveillance). This is also feasible since our U3D perturbations are designed to universally perturb any video at any time (from any frame in the streaming video) without the boundary effect. Thus, the U3D perturbations can be generated offline and injected into the online videos in real-time applications.

Adversary's Capabilities. The adversary can either craft adversarial examples offline on static videos, or inject the U3D perturbations (pre-learned) into the streaming videos, similar to the attack setting in [43], [49]. Specifically, the adversary can manipulate the systems via malware, or perform man-in-the-middle (MITM) attack to intercept and perturb the streaming videos. Furthermore, the adversary could also slightly delay the streaming video when performing injections without affecting the overall quality of the streaming video.

Note that MITM adversary is unable to perform attacks by simply replacing streaming videos with pre-recorded videos or static frames while ensuring the stealthiness of the attack, since the adversary does not know what will happen in the future [43]. For instance, if the adversary wants to fool the video surveillance system in a parking lot, he/she may need to replace the video streams in long run (ideally all the time) to perform the attack. However, without prior knowledge on the future objects/events in the parking lot, it would be very hard to make the replaced video visually consistent with the real scenario (e.g., moving vehicles, humans, and weather). Then, the replaced video can be easily identified by the security personnel. Instead, U3D attack can be easier to be covertly realized (always human-imperceptible). The universal and boundary effect-free perturbation will be efficiently generated and continuously injected in real time (see our design goals in Section III-A). Thus, it can universally attack video streams in long run even if video streams may differ at different times.

We experimentally study the practicality of attack vectors (e.g., man-in-the-middle attack) in a video surveillance system [19], [33], [53] and implement the real-time attack based on U3D. The results show that U3D is efficient to attack real-time video recognition systems (as detailed in Section V-G).

Adversary's Knowledge (black-box). Similar to other black-box transfer-based attacks [15], [46], [56], the adversary does not necessarily know the structure and parameters of the target DNN model. U3D aims to generate universal perturbations that can successfully subvert a variety of DNN models, each of which can be the potential target DNN model. By default, the adversary does not need to query the learning results (e.g., classification score or label) from the target model either.

To successfully perform the attack, the adversary will leverage the *high transferability* of U3D to deviate the target DNN models. Specifically, we assume that the adversary can utilize any public DNN model as the surrogate (e.g., C3D, I3D, LRCN and TSN) and some labeled videos (e.g., from any public data such as the HMDB51 dataset [36]). Such data are not necessarily included the training data of the target DNN model. *The surrogate model can be very different from the target model.* Without querying the target model, the U3D attack is even easier to realize than the conventional query-based black-box attacks [5], [13], [31].

Indeed, the U3D attack can also integrate queries over the target DNN model when necessary (see such extended attack design and evaluations in Section V-D). Thus, the transfer-based back-box attack will turn into a hybrid black-box attack [71], which integrate both query-based and transfer-based attack strategies to improve the attack performance under the black-box setting. We have experimentally validated that integrating a number of queries over the target DNN model could slightly enhance the success rates.

III. U3D Attack Methodology

A. U3D Attack Design Goals

The goals in our U3D attack design include:

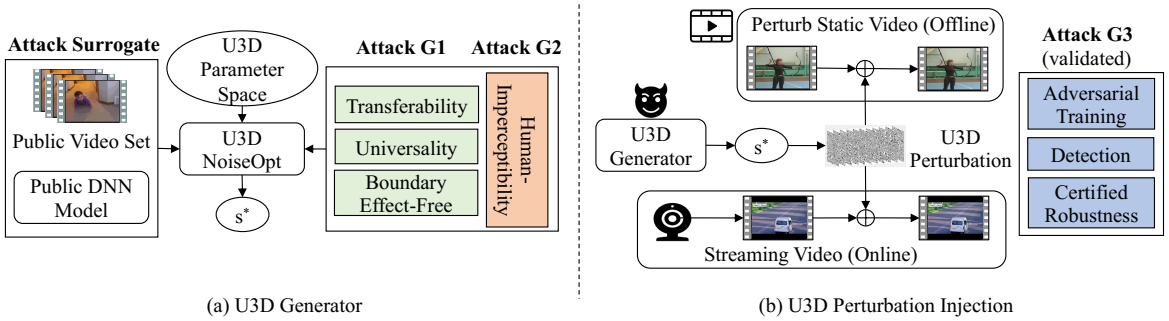
- G1: The attack should achieve high performance on the video recognition systems under the black-box setting.
- G2: The adversarial perturbations should be very small to obtain good human-imperceptibility.
- G3: The adversarial examples are robust against existing defense schemes (cannot be easily detected or mitigated).

G1: High Attack Performance. To launch the U3D attack, the following properties are desired: (1) transferable on a wide variety of DNN models for video recognition; (2) universal on a large number of videos; (3) boundary effect-free.

Different from increasing the magnitude of the perturbations for transferability [6], [11], [73], we formulate an optimization problem with a surrogate DNN model (which can be any public DNN model) in an interpretable fashion. The objective is to maximize the distance between the clean video and perturbed video in the feature space representation (Section IV-B1). First, the change of feature space representations via perturbations (especially the deep spatio-temporal feature space for videos) will non-trivially impact the classification results. This will increase the success rates of the attack. Second, the explicit attack in the feature space could craft more transferable adversarial examples since the intermediate layer features of DNNs have shown to be transferable [90]. Experimental results in Section V have demonstrated high cross-model transferability for feature space perturbations.

Moreover, the adversary does not have prior knowledge on the video (especially the streaming video), then the 3D perturbations should universally attack [50] a large number of videos (ideally, any video). We construct U3D perturbations from a relatively small set of videos to fool the target DNN model on arbitrary input videos with high success rates.

With temporal dimensions on multiple frames, the video perturbations should address the potential misalignment with the input video (boundary effect [43]), which can degrade the attack performance, especially in long run. While launching attacks, the perturbation should be effectively injected at any time in the video. To address the misalignment, we employ a transformation function to convert the perturbation temporally, and then optimize the attack on all temporal transformations



(a) U3D Generator

(b) U3D Perturbation Injection

Fig. 2: U3D attack framework (including three design goals: G1, G2 and G3). (a) `U3D_Generator` learns the near-optimal U3D parameters s^*. (b) U3D perturbations can be generated on-the-fly with s^* to perturb both static and streaming videos.

(see Section IV-B2), which enable the U3D perturbations to be injected at random times without the boundary effect.

G2: Human-Imperceptibility. We add a bound on the U3D perturbations with ℓ_∞-norm, which strictly restricts the pixel deviations. Later, we use MSE metrics to quantify the perturbations in the experiments. Moreover, we conduct surveys among humans to illustrate the imperceptibility of U3D.

G3: Robustness against Defenses. To show the robustness of our U3D attack, we implement attacks on the video recognition models equipped with defense schemes (*G3 is not directly designed but ensured with post-validation*). There are two rules of thumb for evaluating attacks: (1) we should carefully utilize current effective defenses to explicitly defend against the newly proposed attack, e.g., developing adaptive schemes which uncover the potential weaknesses of the attack; (2) the defenses should be in white-box setting, i.e., the defender should be aware of the attack, including the adversary's knowledge and strategy. The rules of thumb also work for evaluating newly proposed defenses vice versa [2], [8], [74].

Specifically, we adapt three types of major defense schemes: (1) adversarial training [47], [63]; (2) detection [85], [89]; (3) certified robustness [41], [52]. We redesign the defense schemes to defend against universal perturbations or U3D per the rules of thumb. For example, based on the adversarial training (AT) [47], [63], we design the U3D-AT, which utilizes the capability of AT to defend against the best U3D (iteratively updating U3D perturbations). See details in Section V-F.

B. U3D Attack Overview

We now overview the U3D attack in Figure 2. We first formulate the U3D perturbation generation (`U3D_Generator`) by synthesizing the procedural noise [39], [57] (which can be efficiently generated with low-frequency patterns) with the U3D parameters s (see Section IV-A). Meanwhile, the attack goals of U3D are optimized: transferability, universality, and boundary effect-free (see Section IV-B). Then, we apply the particle swarm optimization (PSO) to solve the problem to derive the near-optimal parameters s^* for generating U3D perturbations (see Section IV-C). Finally, U3D perturbations can be generated on-the-fly to be injected into the videos in the attack scenarios (either static or streaming videos).

C. U3D Attack Formulation

The DNN model can be modeled as a function $f(\cdot)$ that infers the video v with a label (e.g., the label with the top-1 probability). The attack crafts a video adversarial example v' by injecting the perturbation ξ into the original video v: $v' = v + \xi$, where the output label of v' by the DNN model $f(\cdot)$ would be $f(v') \neq f(v)$ (as a universal attack).

To pursue human-imperceptible perturbations, ℓ_∞-norm is adapted to bound the distance between the original and perturbed videos (w.r.t. U3D perturbation ξ) with a pre-specified small value ϵ: $||v' - v||_\infty = \max_i |\xi| \leq \epsilon$. Then, we formulate an optimization problem to generate U3D perturbations: $\arg\min_\xi : \Gamma(v + \xi), s.t.\ ||\xi||_\infty \leq \epsilon$, where Γ is a loss metric function, e.g., a distance or cross-entropy metric. In Section IV-B, we align the objective function with the attack goals in the optimization for the U3D design.

IV. Attack Design

A. U3D Perturbation Formalization

"Procedural noise" [39], [40], [57], [58] refers to the algorithmically generated noise with a predefined function, which can be added to enrich visual details (e.g., texture, and shading) in computer graphics. It can be directly computed with only a few parameters, and has no noticeable direction artifacts [17], [39], [57]. These properties make it *potentially fit for inexpensively computing adversarial perturbations*. While constructing U3D perturbations, we utilize two types of common procedural noises: (1) "Perlin noise" [57], [58] (a lattice gradient noise) due to its ease of use, popularity and simplicity; (2) "Gabor noise" [40] (a convolutional sparse noise) with good sparsity and accurate spectral control. We propose two types of U3D perturbations, "U3D$_p$" and "U3D$_g$", both of which universally perturb videos to subvert the DNN models.

We first formally define the U3D noise function. Denote $\mathcal{N}(x, y, t; S)$ as the U3D noise function, where (x, y, t) represents the 3D coordinates of each pixel in the video, and S is the parameter set for noise generation.

1) U3D$_p$ Noise: Perlin noise [57], [58] originally works as an image modeling primitive to produce natural-looking textures in realistic computer generated imagery.

Specifically, we denote every pixel in a video by its 3D coordinates (x, y, t) where (x, y) are the coordinates in frame

t, and denote the Perlin noise value of the pixel (x, y, t) as $p(x, y, t)$. To model the change of visual perturbations, we define three new parameters of wavelength λ_x, λ_y, λ_t to determine the octaves along the three dimensions x-axis, y-axis, and frame t, respectively, and define the number of octaves as Λ. The newly updated noise is computed as the sum of all the corresponding octaves for 3D coordinates:

$$\mathcal{N}(x, y, t) = \sum_{\ell=0}^{\Lambda} p(x \cdot \frac{2^\ell}{\lambda_x}, y \cdot \frac{2^\ell}{\lambda_y}, t \cdot \frac{2^\ell}{\lambda_t}) \tag{1}$$

Moreover, we compose the noise function with a color map function [72] to generate distinct visual perturbations in the video. Then, the noise of pixel (x, y, t) can be derived as:

$$\mathcal{N}_p(x, y, t) = cmap(\mathcal{N}(x, y, t), \phi) \tag{2}$$

where $cmap(p, \phi) = \sin(p \cdot 2\pi\phi)$ is a sine color map function, which ensures *the bound of noise value with the circular property*. ϕ indicates the period of the sine function, and the visualization of perturbations can be tuned with ϕ.

U3D$_p$ Parameters. Combining Equation 1 and 2, we denote the corresponding parameter set as S_p for U3D$_p$ noise:

$$S_p = \{\lambda_x, \lambda_y, \lambda_t, \Lambda, \phi\} \tag{3}$$

2) U3D$_g$ Noise: Gabor noise [39], [40] is a type of sparse convolution noise that obtains a better spectral control via a Gabor kernel, a multiplication of circular Gaussian envelope and a harmonic function [24]. We construct U3D$_g$ noise by first extending the 2D Gabor kernel to 3D Gabor kernel (adding the temporal dimension t):

$$g(x, y, t) = Ke^{-\pi\sigma^2(x^2+y^2+t^2)} \cos\left[2\pi F(x' + y' + t')\right] \tag{4}$$

where $x' = x \sin\theta\cos\omega, y' = y \sin\theta \sin\omega, t' = t\cos\theta$; K and σ are the magnitude and width of the Gaussian envelope; F and (θ, ω) are the magnitude and orientation angles of the frequency in the harmonic function. Then, we derive the noise $\mathcal{N}(x, y, t)$ with the sparse convolution and 3D Gabor kernel:

$$\mathcal{N}(x, y, t) = \sum_k g(x - x_k, y - y_k, t) \tag{5}$$

where the point set $\{\forall(x_k, y_k, t)\}$ are a set of sampled pixel points in the same frame t with Poisson distribution. Furthermore, to model the isotropy of the Gabor noise [39], we realize the two frequency orientations (θ, ω) as random variables (θ_i, ω_i) uniformly distributed in $[0, 2\pi]$. Then, the updated U3D$_g$ noise is given as below:

$$\mathcal{N}_g(x, y, t) = \sum_i \mathcal{N}(x, y, t; (\theta_i, \omega_i)) \tag{6}$$

U3D$_g$ Parameters. Similar to U3D$_p$, we denote the following parameter set as S_g for U3D$_g$ with Equation 4 and 6:

$$S_g = \{K, \sigma, F\} \tag{7}$$

We synthesize the procedural noise to construct the U3D perturbations, whose low-frequency patterns and low computational overhead can greatly advance the attacks. Formally, given the U3D noise function $\mathcal{N}$ and the parameters s, the generated U3D perturbation ξ of length T will be:

$$\xi = \{\mathcal{N}(t; s) | t \in [0, T-1]\} \tag{8}$$

If T is less than the video length, ξ will be circular. Note that T works as a pre-specified parameter. For simplification, we use $\xi = \mathcal{N}(T; s)$ to represent Equation 8. Next, we will present how to calibrate U3D perturbation to achieve the design goals.

B. Calibrating U3D Perturbations

1) Improving Transferability in Feature Space: U3D aims to deviate the intermediate layer's features, which could improve the transferability of the attacks. Large distance between the original and perturbed videos' features at intermediate layers of the DNN model can result in relatively high deviations in the final results. This will increase the probabilities on false learning by the unknown target DNN model and videos.

Specifically, we formally define $f_L(\cdot, d)$ as the truncated DNN model function, which outputs the intermediate feature of the input video at layer $L_d, d \in [1, M]$ of the DNN model $f(\cdot)$, M is the number of DNN layers. Then, $f_L(v, d), f_L(v', d)$ are denoted as the intermediate features of the original video v and perturbed video v', respectively. Thus, we have the ℓ_2-norm distance between the feature representations of the original video v and perturbed video $v' = v + \xi$ at layer d of the DNN as:

$$\mathcal{D}(v, v'; d) = ||P(f_L(v, d)) - P(f_L(v', d))||_2 \tag{9}$$

where $P(z) = sign(z) \odot |z|^\alpha$ is a power normalization function $\alpha \in [0, 1]$ and $\odot$ is the element-wise product [59].

Then, we maximize the distance $\mathcal{D}(v, v'; d)$ between the original and perturb videos over all the intermediate feature space as our attack objective function:

$$\max_{\xi} : \sum_{d \in [1, M]} \mathcal{D}(v, v + \xi; d) \tag{10}$$

2) Mitigating Boundary Effect: Recall that the boundary effect may potentially degrade the attack performance due to the misalignment between the adversarial perturbation and the input video. To tackle such issue, we introduce a temporal transformation function $\texttt{Trans}(\cdot)$ for the U3D perturbation with a shifting variable denoted as τ. Specifically, given a U3D perturbation ξ of length T, then $\texttt{Trans}(\xi; \tau)$ represents the U3D perturbation ξ temporally shifted by $\tau \in [0, T-1]$. Then, we maximize the expectation of the feature distances with all the T possible temporal shift transformation $\tau \in U[0, T-1]$ for U3D perturbation ξ (U denotes the uniform distribution):

$$\max_{\xi} : \mathop{\mathbb{E}}_{\tau \sim U[0, T-1]} [\sum_{d \in M} \mathcal{D}(v, v + \texttt{Trans}(\xi, \tau); d)] \tag{11}$$

To achieve such objective, we can consider all the possible transformed U3D perturbation (the size of transformation will be T) uniformly shifted with $\tau \in [0, T-1]$ (step 1 frame by frame in the video). τ will be sampled in the corresponding algorithm. Then, our U3D attack can learn a generic adversarial perturbation without the boundary effect, which can be injected into the streaming video anytime.

3) Improving Universality with Public Videos: Another goal is to find a universal perturbation learned from a relatively small set of videos, which can effectively perturb the unseen videos for misclassification. Denoting a set of public videos as V, the optimization integrates the universality maximization on videos in V (and ℓ_∞-norm bound) as below:

$$\begin{aligned} \max_{\xi} : & \mathop{\mathbb{E}}_{v\sim V,\tau\sim U[0,T-1]}[\sum_{d\in M}\mathcal{D}(v, v+\texttt{Trans}(\xi,\tau); d)] \\ s.t.\ \ & \xi = \mathcal{N}(T; s),\ ||\xi||_\infty \le \epsilon \end{aligned} \tag{12}$$

C. Optimizing and Generating U3D Perturbations

Since the U3D perturbation ξ can be efficiently generated if the U3D parameter set $\mathcal{S}$ is pre-computed, Equation 12 will optimize the attack w.r.t. $\mathcal{S}$. To search the optimal U3D parameter set $\mathcal{S}$), we solve it with the Particle Swarm Optimization (PSO) method [21]. Specifically, the parameter values in $\mathcal{S}$ are viewed as the particles' positions, and the set of parameter ranges can be constructed as the search space. Then, the objective function (Equation 12) is the fitness function $\mathcal{A}(f, V, \mathcal{N}(T; \vec{s}))$, where $\vec{s}$ is the current position for U3D parameter set S in the iterations.

In the initialization phase, m points will be randomly selected from the searching space for $\mathcal{S}$ while satisfying ℓ_∞-norm bound. Then, in the iterations, every particle will iteratively update its position by evaluating the personal and group best location (determined by the output of the fitness function). Notice that, before fed into the fitness function, the algorithm validates if the U3D perturbation ξ generated by the parameter set $\vec{s_i}^{k+1}$ satisfies ℓ_∞-norm bound ϵ or not. Finally, we can get the near-optimal parameter set s^*. Then, we generate the U3D perturbation $\xi = \mathcal{N}(T; s^*)$. The computation of fitness function $\mathcal{A}(f, V, \mathcal{N}(T; \vec{s}))$ and PSO optimization process are detailed in Algorithm 1 and 2, respectively (Appendix B).

We have evaluated PSO by benchmarking with genetic algorithms [26], simulated annealing [80], and Tabu search [25]. PSO slightly outperforms them for U3D optimization (experimental setting and results are detailed in Appendix C).

V. Experiments

A. Experimental Setup

Datasets. We use three widely used datasets for video recognition to validate the proposed U3D attack.

- HMDB51 [36] dataset includes 6,766 video clips (30 fps) in 51 different actions, e.g., fencing, climb and golf.
- UCF101 [66] dataset includes 13,320 video clips (25 fps) in 101 different actions, e.g., archery, and punch.
- UCF Crime [69] dataset includes 1,900 long surveillance videos (30 fps) collected from Youtube and Liveleak, in 13 anomalies, e.g., accident, explosion, and shooting.

The HMDB51 and UCF101 datasets are used for video classification, and the UCF Crime dataset for anomaly detection.

Target DNN Models. We evaluate the U3D attack on two common DNN models for video recognition: (1) C3D model [76]; (2) I3D model [69]. We also implement two video recognition techniques based on both C3D and I3D: (1) video classification [76]; (2) video anomaly detection [69] identifying anomalies by scoring the video segments in sequence.

Note that we choose C3D and I3D as the main evaluation models to show the attack performance due to the popularity and practicality in the video recognition systems (as depicted in Section II-A). To fully evaluate the *transferability* of the U3D attack, we choose three more video classification models, including LRCN [20], DN [34] and TSN [65], and evaluate the U3D attack across five different DNN models. We summarize the differences of such five models in Appendix E1.

Benchmarks. We use the following baseline adversarial perturbations: (1) Gaussian Noise: $\xi_g \sim \mathcal{N}(0, \sigma^2)$ and σ=0.01; (2) Uniform Noise: uniformly sampled noise $\xi_u \sim [-\epsilon, \epsilon]$; (3) Random U3D: applying U3D without calibration by randomly choosing parameters. For the above three methods, we repeat each experiment 10 times, and return the average value; (4) The state-of-the-art video attacks, C-DUP [43] (as a *white-box* universal attack), V-BAD [32] and H-Opt [83] (both as *non-universal* black-box attacks).

Since V-BAD [32] and H-Opt [83] are non-universal, they might be incomparable with U3D and C-DUP on attacking a specific target (though their success rates are claimed to be high in such cases). It might also be unfair to compare U3D and C-DUP with V-BAD and H-Opt on transferability since the latter two are not designed towards that goal.

B. Attack Performance

We first evaluate U3D$_p$ and U3D$_g$ generated with a surrogate C3D model to attack unknown target models on different datasets. Specifically, we randomly select 500 videos from the HMDB51 dataset (retaining a similar distribution for classes as the full dataset) as the public video set (V), and consider the full UCF101 and UCF Crime datasets as the target set. We set $\epsilon = 8, T = 16$ and report the attack results on the target set. Note that the MSE of all the U3D perturbations are below 20 (very minor distortion out of 255^2 in the scale).

TABLE I. U3D vs. benchmarks (success rates; C3D/HMDB51 as surrogate; C3D/I3D and UCF101/UCF Crime as target).

Model / Noise	C3D		I3D	
	UCF101	UCF Crime	UCF101	UCF Crime
Gaussian	10.2%	15.3%	9.1%	12.6%
Uniform	5.3%	9.1%	1.7%	2.4%
Rnd. U3D	43.2%	52.6%	40.3%	51.8%
C-DUP [43]	80.2%	83.6%	54.4%	45.8%
U3D$_p$	82.6%	92.1%	80.4%	87.1%
U3D$_g$	85.4%	93.4%	82.9%	90.2%

Table I lists the results for applying U3D$_p$ and U3D$_g$ to attack unknown models and videos (in different datasets). The U3D perturbations are injected into both UCF101 (for *video classification*) and UCF Crime (for *anomaly detection*) datasets, which are then inferred by both C3D and I3D. Such black-box attacks are realized by the *transferability* and *universality* of U3D (which will be thoroughly evaluated in Section V-C). Table I also includes the attack performance of Gaussian, Uniform, Random, and C-DUP [43] (see the setting

in Section V-A). For both U3D and benchmarks, we apply the perturbations to full UCF101 and UCF Crime datasets.

Both U3D$_p$ and U3D$_g$ achieve high success rates on the two DNNs. For C3D, U3D$_p$ achieves 82.6% on the UCF101 dataset (video classification) and 92.1% on the UCF Crime (anomaly detection) while U3D$_g$ obtains a slightly higher success rate, i.e., 85.4% on the UCF101, and 93.4% on the UCF Crime. This can also show that our U3D perturbations effectively attack to other different DNN models on different datasets, e.g., HMDB51 and C3D $\rightarrow$ UCF Crime and I3D.

However, the benchmarks cannot achieve satisfactory attack performance. Injecting random noise (Gaussian and Uniform) to videos can only give 2.4%-15.3% success rates in all the experiments. Random U3D (random parameters without optimization) performs better but still not satisfactory (35.7%-52.6%). C-DUP [43] returns worse success rates on both C3D and I3D, even in the white-box setting. Since it is designed for attacking C3D, its performance on I3D is even worse (54.4% on UCF101 and 45.8% on UCF Crime, low transferability).

Finally, both U3D$_p$ and U3D$_g$ can perform very well (90%+) on anomaly detection regardless of the target models. We observe that anomaly detection is more vulnerable than video classification under the same perturbation, e.g., U3D$_p$ (92.1%>82.6%). The possible reason is that such DNN models have an extra computing model to output anomaly scores, which may make it more sensitive to perturbations.

C. Transferability and Universality

Transferability. *Transferability* refers to the perturbations designed for one classifier can also attack other classifiers (cross-model) [73]. To study the transferability, we first define the *transfer rate* (TR) as the percent of the adversarial examples which deviates one model f_{srg} (e.g., a public DNN model as the surrogate model) and also deviate the target model f_{tar} (black-box). We denote $f_{srg} \rightarrow f_{tar}$ as the transferability of the attack from surrogate model to target model.

TABLE II. Transferability: transfer rate (TR) on UCF101 from surrogate model f_{srg} to target model f_{tar}. See similar results on HMDB51 and UCF Crime in Appendix A.

Noise	f_{srg} \ f_{tar}	C3D	I3D	DN	LRCN	TSN
U3D$_p$	C3D	–	93.4%	92.7%	85.0%	87.2%
	I3D	89.7%	–	96.3%	88.7%	85.0%
	DN	84.0%	83.2%	–	85.5%	83.4%
	LRCN	85.8%	87.2%	92.4%	–	86.1%
	TSN	85.5%	82.5%	89.3%	87.5%	–
U3D$_g$	C3D	–	87.0%	93.2%	86.3%	85.3%
	I3D	88.2%	–	97.4%	85.2%	86.0%
	DN	82.6%	81.4%	–	83.7%	85.6%
	LRCN	81.2%	83.4%	88.6%	–	84.5%
	TSN	86.2%	83.6%	90.2%	86.4%	–

To evaluate the transferability, we choose C3D, I3D and other three more video classification models as surrogate/target models: DN [34], LRCN [20], and TSN [65], all of which are already fine-tuned on the UCF101 dataset. Then, we compute U3D$_p$ and U3D$_g$ ($\epsilon = 8$) with the surrogate model (as surrogate) and apply the U3D perturbations to craft adversarial examples on the UCF101 dataset, which are fed into the target models. We generate the U3D perturbations with 10% of the UCF101 dataset (randomly picked for each class), and select all the adversarial examples (crafted on the 90% of videos for target dataset) which can successfully fool the surrogate models. Then, we examine the percent of such adversarial examples that can fool the target model.

Table II presents the transfer rates of both U3D$_p$ and U3D$_g$. We can observe that all the attack can achieve high transfer rates (over 80%). This shows that our U3D perturbations achieve good transferability across these models. For example, U3D$_p$ can obtain 92.7% transfer rate for C3D→DN and 92.4% for LRCN→DN. We repeat the same set of experiments on the HMDB51 and UCF Crime datasets (results are given in Table XII and XIII in Appendix A). High cross-model transferability for U3D can also be observed from such experiments.

Universality. *Universality* refers to the perturbations learnt from one dataset can perturb many different datasets (cross-data) to fool the DNN models [50]. To evaluate the universality of U3D, we first randomly pick 500 videos as the surrogate video set (denoted as X) from each of the three datasets, HMDB51, UCF101 and UCF Crime, respectively (retaining a similar class distribution as the full datasets). Then, we compute the U3D perturbations ($\epsilon = 8$) on X with the C3D model and evaluate the attack success rates on all the three datasets (as the target dataset Y). For the same dataset case (intra-dataset attack), the target set Y excludes X to evaluate the universality. All the results are listed in Table III.

We can observe that the U3D achieve 80%+ success rates for all the cases ($X \rightarrow Y$ within the same dataset or across different datasets). The diagonal results are higher than other cases, which shows that the U3D can perform well among the unseen videos in the same dataset. Moreover, for the same U3D perturbation, e.g., U3D$_g$, the success rate of UCF Crime→UCF101 is lower than that of HMDB51→UCF101 (81.6%<85.4%). Similar results are also observed from UCF Crime→HMDB51 and UCF101→HMDB51 (82.2%<85.0%). Since HMDB51 and UCF101 consist of human action videos while UCF Crime includes surveillance videos for anomaly detection, U3D perturbations learned on UCF Crime will exhibit less universality to HMDB51 and UCF101. Thus, selecting different surrogate videos can slightly help tune the attack performance on different target models and videos.

TABLE III. Universality (success rate (SR); surrogate C3D). See similar results for surrogate I3D in Appendix A.

Noise	X \ Y	HMDB51	UCF101	UCF Crime
U3D$_p$	HMDB51	87.3%	82.6%	92.1%
	UCF101	84.2%	88.4%	91.5%
	UCF Crime	80.1%	82.4%	96.0%
U3D$_g$	HMDB51	88.7%	85.4%	93.4%
	UCF101	85.0%	86.2%	90.2%
	UCF Crime	82.2%	81.6%	95.3%

We repeat the same set of experiments for I3D as surrogate

(see similar high universality in Table XIV in Appendix A). Note that C-DUP [43] (as a *white-box* attack only on C3D) has low transferability (in Table I), and V-BAD [32] and H-Opt [83] (both as *non-universal* attacks) have low transferability.

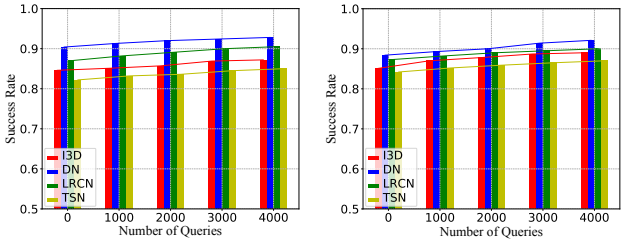
D. Hybrid Black-Box Attack with Queries over Target Model

U3D is designed to universally attack different target models, and it has shown high transferability. If the attack performance is not very satisfactory for a new target model (though not found in our extensive experiments), we can extend the U3D to a hybrid black-box attack [71] by integrating queries over the target model $g(\cdot)$. Note that this still maintains attack universality on different target models. Thus, given the surrogate model $f(\cdot)$ (including a small set of public videos) and the target model $g(\cdot)$ available for queries, we can update the optimization for U3D by integrating the queries using input videos $v_1, \ldots, v_n$ (perturbed by ξ before querying):

$$\begin{aligned} \max_{\xi} : & \mathop{\mathbb{E}}_{v\sim V, \tau\sim U[0,T-1]} [\sum_{d\in M} \mathcal{D}(v, v + \texttt{Trans}(\xi, \tau); d)] \\ & + \omega \cdot \mathcal{Q}(g, v_1, \ldots, v_n, \xi) \\ s.t. \;\; & \xi = \mathcal{N}(T; s), \;\; \|\xi\|_\infty \leq \epsilon \end{aligned} \tag{13}$$

where the query oracle $\mathcal{Q}(\cdot)$ first derives the final classification results of the perturbed videos $\{v_1 + \xi, \ldots\, v_n + \xi\}$ by the target model $g(\cdot)$, and then returns the success rate for such videos. ω is hyperparameter for weighing the transferability of the surrogate model and queries (success rate) over the target model. Note that the adversary only needs to query the final classification (limited information) instead of the specific probability scores or logits information.

This optimization will search the U3D perturbations which can successfully fool the target model $g(\cdot)$ by both transferability and queries (hybrid). Similarly, after revising the fitness function with the new objective (Equation 13), we can apply PSO to compute the optimal U3D parameters.



(a) U3D$_p$ (b) U3D$_g$

Fig. 3: Hybrid black-box attack performance (surrogate C3D).

To evaluate the hybrid attack, we choose the C3D as the surrogate model, and I3D, DN, LRCN, TSN as target models, respectively. Then, we follow the setting as the previous experiments on the UCF101 dataset (10% for learning U3D while 90% for target set) and U3D parameters. For the hybrid attack, we set the size of querying dataset as 50 (randomly chosen), $\epsilon = 8$ and $\omega = 10$, vary the number of queries as $\{0,\ 1000,\ 2000,\ 3000,\ 4000\}$ ("0" means the original U3D attack without queries). Then, we apply PSO to optimize U3D perturbations on Equation 13, and report the success rate for both U3D perturbations against the four target models. Figure 3 shows that the success rates of both U3D perturbations slightly increase as the number of queries increases. The hybrid attack with additional queries to the target model can improve the transfer-based attack to some extent [71].

E. Visual Impact and Human-Imperceptibility

Visual Impact. We arbitrarily select two videos, "shooting" and "fighting" to demonstrate the visual differences of the adversarial examples. Figure 5 in Appendix A presents a sequence of selected frames in two videos, and we can observe that the videos perturbed by U3D$_p$ and U3D$_g$ are much more human-imperceptible than C-DUP.

Human-Imperceptibility Study. We also conducted a human-imperceptibility study (with an IRB exempt) to validate if the perturbed videos could be visually discerned by humans compared with the original videos. We distributed the videos (original videos, adversarial examples based on U3D$_p$, U3D$_g$ and C-DUP) and an online survey to 87 anonymous students.

Specifically, we randomly pick 870 videos from the UCF101 and UCF Crime datasets. To avoid bias on the distribution of adversarial examples, we randomly choose 435 videos to inject perturbations (218 videos by U3D$_p$ and 217 by U3D$_g$; all the adversarial examples are selected from the previous groups of experiments to successfully fool the video classifiers). The students do not know the distribution of adversarial examples. Then, 435 pairs of videos (the adversarial examples and their clean videos) are mixed with another 435 pairs of videos (the remaining 435 clean videos and their duplicates).

Next, we randomly assign 10 pairs of videos to each student. Each of the 87 students is expected to watch and annotate 10 pairs of videos ("visual difference" or "no visual difference"). We also ask the participants to identify the region of difference if they choose "visual difference". Finally, we collected 740 valid annotations on pairs of videos (74 students have submitted their results over a 1-week time frame; among the assigned videos to these students, there exist 386 adversarial examples). We found that 96.4% (372) out of such 386 annotations are "no visual difference", while only 3.6% (14 annotations) of them are annotated with "visual difference". Among the remaining 354 pairs of original videos and their duplicates, there also exist 7 annotations submitted as "visual difference".

Meanwhile, we conduct the same group of studies for C-DUP by randomly picking another 870 different videos to inject perturbations (in order to avoid the bias on mutually linking the adversarial examples of U3D and C-DUP perturbed on the same videos).[1] As high as 85.4% (304 out of 356 valid adversarial examples of C-DUP) can be successfully identified by the same group of students. All the above results have indicated that our U3D perturbations achieve high human-imperceptibility (significantly better than C-DUP).

[1]If any visual difference is identified by the student from the C-DUP perturbed video (or vice-versa), this may give him/her prior knowledge to identify visual difference from U3D (or vice-versa) since both are perturbed.

F. U3D against Defense Schemes

To our best knowledge, there are very few defense schemes against the adversarial attacks on videos (mostly on images). We comprehensively evaluate the performance of U3D against three major categories of state-of-the-art defense schemes, which are adapted towards video/U3D defenses. They include: (1) adversarial training [47], [63]; (2) adversarial example detection [85], [89]; (3) certified robustness [18], [41].

Attack and Defense Setting. We use the U3D_p and U3D_g perturbations generated in Section V-B (surrogate C3D and HMDB51 dataset) to craft adversarial examples on a dataset (e.g., UCF101 or UCF Crime). The adversarial examples will be used to attack the target model (C3D or I3D), which will be adversarially trained, integrated into the detection schemes, or certified with randomization. In all the tables in this subsection, "Model" refers to the target model, and "Dataset" refers to the dataset used to craft adversarial examples.

Adversarial Training. Adversarial training [27], [47], [62], [63] refers to the model training by adding adversarial examples into the training dataset. It has been empirically validated to be effective on improving the robustness against adversarial examples and maintaining accuracy on clean data.

First, due to the universality of U3D, we evaluate U3D attack on a universal adversarial training (denoted as "UAT") [63] which defends against universal perturbations. Specifically, such scheme adopts PGD-based adversarial training [47] to formulate a min-max optimization problem as below:

$$\min_{\theta} \max_{\xi} : \frac{1}{|X|} \sum_{(x_i, y_i) \in X} L(\theta; x_i + \xi, y_i) \quad s.t. \quad ||\xi||_\infty \leq \epsilon \quad (14)$$

where θ denotes the model parameters, $X = \{(x_i, y_i), i \in [1, |X|]\}$ is the training sample set, $L(\cdot)$ is the loss function, and the ℓ_p-norm of universal perturbation ξ is bounded by ϵ. Different from the conventional PGD-based adversarial training (computing the perturbation for each instance), the inner optimization problem seeks a universal (more precisely, batch X-universal) perturbation ξ to maximize the adversarial loss w.r.t. the sample set X. It has been shown to be effective against universal perturbations compared to PGD-based adversarial training [47]. In addition, it is more efficient to compute one universal perturbation across all the training iterations, i.e., only updating perturbation ξ once for each step [63].

Second, besides "UAT", we tailor the universal adversarial training towards U3D (denoted as "U3D-AT"), and evaluate the U3D under a *stronger defense setting* (see the white-box defense of G3 in Section III-A). Specifically, the defender knows the U3D function $\mathcal{N}(\cdot)$ but does not know the specific values of the U3D parameters s. Recall that the U3D perturbation is computed by optimizing Equation 12, which is formulated as a attack fitness function $\mathcal{A}(f, V, s)$. Then, we can also adapt the UAT framework by replacing the inner optimization objective with the U3D function as $\mathcal{A}(f, X, s)$:

$$\min_{\theta} \max_{s} : \mathcal{A}(f, X, s) \quad s.t. \quad \xi = \mathcal{N}(T; s), \ ||\xi||_\infty \leq \epsilon \quad (15)$$

where the norm-bounded U3D perturbation ξ can be computed by the U3D function with the parameters s. Similar to UAT, we can iteratively update the best U3D perturbation among a batch of data (X) in the inner loop via `NoiseOpt`, which adapts PSO to find the optimal parameters.

For the experiments, we evaluate the defense performance of standard PGD-based adversarial training (denoted as "Normal"), universal adversarial training ("UAT") and our U3D-adaptive AT ("U3D-AT") against our U3D perturbations, respectively. We split the datasets (UCF101 and UCF Crime) into the training dataset (80%) and the test dataset (20%). We set the perturbation bound $\epsilon = 8$. For both UAT and U3D-AT, we set the batch size as 200. For UAT, we utilize FGSM to update ξ, and Momentum SGD optimizer to update model parameters as the original setting [63]. For the adversarially trained models, we evaluate the accuracy (Clean ACR) – the predication accuracy on the clean data, besides the attack success rate (SR) for misclassification. Note that we also report the accuracy and SR of the normal models.

Table IV summarizes the results of the adversarial training against U3D on the UCF101. The accuracy of both UAT and U3D-AT on the clean data declines since the training has included adversarial examples. Nevertheless, the success rates of both U3D_p and U3D_g have been reduced against both UAT and U3D-AT. The U3D-AT performs better than the UAT, e.g., the attack SR of U3D_p is 42.7%<67.4% on the C3D. This is because U3D-AT directly optimizes the defense on U3D (with the attack fitness function), which thus makes the model more robust against U3D. However, such U3D-AT is more like "white-box" defense in which the defender (model owner) already knows the adversary's strategy (e.g., U3D format and attack function). In practice, the defender usually cannot readily obtain such information.

TABLE IV. Adversarial training on UCF101. See similar results on UCF Crime in Table XV (Appendix A).

Model	Defense	Clean ACR	U3D_p (SR)	U3D_g (SR)
C3D	Normal	86.2%	83.7%	84.2%
	UAT	78.5%	67.4%	65.5%
	U3D-AT	77.2%	42.7%	45.3%
I3D	Normal	88.7%	82.1%	82.6%
	UAT	80.4%	70.2%	69.5%
	U3D-AT	78.6%	50.3%	47.4%

Adversarial Examples Detection. Most detection schemes [45], [48], [61], [85], [89] locally train a detector or utilize feature characteristics in adversarial examples to determine if the input is perturbed or not. For instance, a detector can be trained on both clean data and adversarial examples via adversarial training [89]. Although detection schemes have difficulties on mitigating adversarial attacks (e.g., Magnet [48] was broken by [8], and some recent defenses were broken by adaptive attacks [74]), we still evaluate our U3D against detection schemes (including that adapted to U3D). Note that the U3D attack can be both online and offline. Then, we evaluate both of them against the detection schemes (assuming that the

offline detection can be executed with arbitrary runtime).

First, for the online detection, we choose AdvIT [85] which is effective against the existing adversarial attacks on *real-time* video recognition. It finds the inconsistency among the temporally close frames with the optimal flow information, assuming that perturbations can destroy the frame consistency to some extent. Specifically, given one target frame (to be detected), AdvIT first estimates the optimal flow between the target frame and previous k frames, and then reconstructs pseudo frames by applying the optical flow to the beginning frame. Finally, it would compute the inconsistency score c between the target frame and pseudo frames, where high inconsistency score indicates that the target frame is adversarial. To defend against the adaptive attacks, AdvIT applies the Gaussian noise to fuzz the optical flow for generating the pseudo frames.

In the experiments, we randomly select 200 clean videos from the UCF101 and UCF Crime datasets (100 each), and apply both U3D$_p$ and U3D$_g$ perturbations to craft adversarial examples. We set the perturbation bound $\epsilon = 8$. For detection, we set $k = 3$ (which only slightly affects the detection rate) and utilize FlowNet [30] as the optical flow estimator in AdvIT. Then, we randomly select 5 frames in each video as the target frames, and average the inconsistency scores (reporting detection when ≥ 1) to derive the detection results.

Table V summarizes the detection accuracy (DR) and false positive rate (FPR) of AdvIT. It shows that U3D can bypass the detection of the state-of-the-art detection scheme, even though AdvIT achieves low false positive rates. For instance, AdvIT only obtains 12% accuracy to detect U3D$_p$-based adversarial examples for the C3D. The results show that U3D is immune to the temporal consistency detection by AdvIT, since the U3D perturbations are constructed on continuous 3-dimensional noise, which can still retain the consistency in temporal space.

TABLE V. Detection and false positive rates of AdvIT [85]

Model	Dataset	U3D$_p$		U3D$_g$	
		DR	FPR	DR	FPR
C3D	UCF101	12%	2%	18%	2%
	UCF Crime	12%	5%	19%	3%
I3D	UCF101	10%	3%	17%	3%
	UCF Crime	12%	5%	22%	3%

TABLE VI. Detection AUC of AdvIT [85] against U3D, C-DUP, V-BAD, and H-Opt. C3D:1st/3rd row. I3D:2nd/4th row

Dataset	U3D$_p$	U3D$_g$	C-DUP	V-BAD	H-Opt
UCF101	54.2%	56.7%	97.2%	98.4%	99.2%
	56.4%	55.3%	98.7%	97.3%	98.6%
UCF Crime	61.2%	64.8%	97.6%	99.5%	98.3%
	55.6%	58.1%	97.4%	99.7%	99.8%

Furthermore, we have evaluated the Area Under Curve (AUC) values of the Receiver Operation Characteristic Curve (ROC) of AdvIT for U3D perturbations and other three benchmarks: C-DUP [43], V-BAD [32] and H-Opt [83]. The AUC metric represents the probability that the detector assigns a higher score to a random positive sample (adversarial example) than to a random negative sample (clean data) [85]. It can better measure the detection performance than the DR/FPR. Table VI summarizes the results. From the table, we can observe that the AUC values of U3D are close to random guess, e.g., 54.2% (U3D$_p$) and 56.7% (U3D$_g$) on C3D and UCF101 while all the benchmarks can be almost fully detected by AdvIT (*all the AUC values are very close to 1*). This occurs since the temporal consistency cannot hold in the adversarial examples by C-DUP, V-BAD and H-Opt (perturbations are generated specific to the frames as frame-by-frame perturbations).

Second, for the offline detection, we evaluate the U3D against another recent work [89] based on the adversarial training [47]. If the universal adversarial training (UAT) can defend against U3D to some extent, we can also extend it to train a universal perturbation detector against U3D. Specifically, the *asymmetrical adversarial training* (AAT) [89] trains K detectors (for a K-class classification model) to detect adversarial examples. Given an input x, each detector $h_k, k \in [1, K]$ will output a *logit* score corresponding to the class label, which can determine if data is perturbed or not (see details in [89]). To defend against the U3D, we revise the K detectors $h_k, k \in [1, K]$ with the UAT by changing the training objective as below (denoted as "U3D-AAT"):

$$\min_{\theta}: \ [\mathop{\mathbb{E}}_{x\sim D'_k} \max_{s} L(h_k(x+\xi),1) + \mathop{\mathbb{E}}_{x\sim D_k} L(h_k(x),0)]$$
$$s.t. \ \ \xi = \mathcal{N}(T;s), \ ||\xi||_\infty \leq \epsilon \tag{16}$$

The objective includes two parts: (1) the maximum loss of adversarial examples (by U3D perturbation ξ) on the out-of-class data samples D'_k; (2) the loss of intra-class natural data samples D_k. $L(\cdot)$ is a loss function, e.g., binary cross-entropy loss. For the inner optimization of the first part, we adopt similar procedures as U3D-AT to update the U3D perturbations (as depicted earlier).

TABLE VII. Detection and false positive rates of U3D-AAT.

Model	Dataset	U3D$_p$		U3D$_g$	
		DR	FPR	DR	FPR
C3D	UCF101	56.2%	6.3%	53.4%	5.9%
	HMDB51	44.5%	8.2%	47.4%	7.1%
I3D	UCF101	55.4%	4.2%	56.5%	5.1%
	HMDB51	52.6%	5.7%	54.3%	5.9%

To evaluate the performance of the detectors, we choose the action classification on the UCF101 and HMDB51 as K-Class problem ($K = 101$ and 51). Specifically, we split the training/testing datasets by 80%/20% for each category. We set the perturbation bound $\epsilon = 8$, and apply the two U3D perturbations to craft the adversarial examples, which are mixed up with the clean videos for detection (for instance, in UCF101 dataset, there are 2664 clean videos, 2664 videos perturbed by U3D$_p$, and 2664 videos perturbed by U3D$_g$). We adopt the *integrated classifier* which computes the estimated class label $c = f(x)$ with the original classifier f and computes a logit vector $h_c(x)$ using the corresponding detector h_c [89]. We report the detection accuracy (DR) and false positive rate (FPR). The results in Table VII have shown that such universal adversarial detector can detect the U3D perturbations to some extent: the universal AAT detector can

achieve about 50% detection rate while maintaining a low FPR (less than 7%). Such FPR is reasonable considering there could still exist overlapped adversarial subspaces, i.e., U3D-AAT may not be trained to be perfect to learn U3D perturbations and thus separate the perturbed video and clean ones. However, training such AAT detectors should know the U3D attack (*white-box defense*), and it is only limited to defend against offline attacks due to the computational costs.

Certified Robustness. Recently, certified schemes [4], [18], [37], [41], [84] have been proposed to defend against norm-bounded adversarial perturbations with theoretical guarantees. We evaluate the U3D attack against two representative certified schemes: PixelDP [41] and randomized smoothing [18].

First, PixelDP [41] adopts the Gaussian mechanism of differential privacy to slightly randomize the image pixels [81]. After injecting Gaussian noise, the small change of image pixels (adversarial perturbation) will not affect the classification results with some probabilistic bound (thus provide robustness guarantee for DNN models). It will be extended from protecting image DNN models to video DNN models.

To evaluate the U3D attack against PixelDP, we modify the video DNN models by placing the noise layer in the first convolutional layer under the same Gaussian mechanism setting [41] w.r.t. an ℓ_2 attack bound $L = 0.1$ (such setting ensures a high accuracy in [41]). We split training/test as 80%/20% for retraining the model. Note that PixelDP admits that the certified effectiveness against ℓ_∞ attacks is substantially weaker via empirical evaluations (which conforms to the performance of other certified schemes such as randomized smoothing). Then, we generate U3D perturbations bounded by ℓ_2 norm value of 0.5 (which indeed generates very minor perturbations in case of very high video dimensions).

TABLE VIII. Accuracy (ACR) and success rate (SR) of PixelDP [41] (UCF101 and UCF Crime).

Model	Dataset	Clean ACR	U3D$_p$ (SR)	U3D$_g$ (SR)
C3D	UCF101	63.2%	83.4%	85.3%
	UCF Crime	65.9%	86.2%	89.7%
I3D	UCF101	65.8%	82.3%	79.4%
	UCF Crime	67.4%	84.7%	85.2%

We report the classification accuracy of PixelDP on clean videos, and the success rates of the U3D attack in Table VIII. The accuracy of PixelDP drastically declines after injecting Gaussian noises (vs. the baseline models), e.g., 86.2%→63.2% on C3D. Meanwhile, the U3D attack can still achieve high success rates in all the cases. This shows that PixelDP cannot defend against U3D since PixelDP only ensures a weak bound with the Gaussian mechanism of differential privacy.

Second, we also evaluate the certified robustness via randomized smoothing [18]. It provides a tight guarantee (based on the Neyman-Pearson Lemma) for any random classifier by smoothing inputs with an additive isotropic Gaussian noise. However, it only certifies ℓ_2 radius of the perturbation bound. The certified schemes via smoothing against ℓ_∞ have been shown to be ineffective as the input dimensionality d increases. The certified radius is bounded by $O(\frac{1}{\sqrt{d}})$ as $p > 2$ [4], [37].

To evaluate our U3D attack against such certified scheme, we first generate the optimal U3D perturbations ξ by changing perturbation bound ℓ_∞ in `NoiseOpt` to ℓ_2.[2] Specifically, we evaluate the accuracy of the smoothing classifier on the perturbed videos against U3D, which is the percentage of the perturbed videos to be correctly classified (we also evaluate the accuracy on the clean videos as benchmarks). Furthermore, we also derive certificated radius R for the videos, which indicates that the classification results can be certified against any perturbation with ℓ_2-norm no greater than R (see [41]).

Next, we set the number of Monte Carlo samples as $n = 100/1000$ and failure rate $\alpha = 0.001$. The failure rate indicates that the robust classifier can have 1-α confidence to return the classification result. We set the Gaussian variance $\sigma = 0.25$ (same as [41]), and the radius bound for U3D perturbations as $\epsilon = 0.5$ (which generates minor perturbations in case of high video dimensions). We report the accuracy and average certified radius in Table IX. The results show that the randomized smoothing cannot defend against the U3D attack under ℓ_2-norm perturbations (can only certify very small radius), and the robust classifier only achieves less than 70% accuracy on the clean video samples.

TABLE IX. Accuracy and radius of rand. smoothing [18] on UCF101. See similar results in Table XVI (Appendix A).

Model	n	Clean ACR	U3D$_p$		U3D$_g$	
			ACR	Radius	ACR	Radius
C3D	100	67.4%	14.5%	0.23	15.2%	0.19
	1000	68.2%	16.2%	0.24	18.4%	0.25
I3D	100	71.5%	21.7%	0.32	19.8%	0.28
	1000	72.2%	25.2%	0.37	21.2%	0.26

G. Practicality for the U3D Attack

We now discuss the possible attack vectors, and evaluate the U3D attack on a real system. Prior real-time video attack scenarios can also be used for U3D (e.g., manipulating the system via a pre-installed malware in C-DUP [43]). Besides them, we design extra ways for the real-time attack. Other adversarial attacks on videos (including future attacks) can also use our physical scenarios to inject real-time perturbations.

First, the network topology of the recent intelligent video surveillance system (VSS) [19], [53], include: (1) camera; (2) communication network; (3) server. Then, the adversary needs to inject the U3D perturbations in two cases: data-at-rest and data-in-motion [49]. The data-at-rest locally stores videos in the camera or the server. The data-in-motion transfers videos across the network or loads them into the volatile memory. Per the potential threats to VSS [33], [53], we consider the local infiltration to the systems in two scenarios: (1) malware; (2) man-in-the-middle (MITM) attack. First, malware can be locally installed via a malicious firmware update over the USB port. Moreover, the surveillance cameras could be sold through

[2]Since randomized smoothing cannot certify defense against ℓ_∞ bounded attack for high dimensional inputs (e.g., videos) [4], [37], the U3D perturbations using ℓ_2 bound instead of ℓ_∞ are still effective against such scheme.

legitimate sales channels with the pre-installed malware [67]. Second, for the MITM attack, the adversary could access to the local network (e.g., by penetration) which connects to the camera and server, and behave like a normal user. Here, we take the MITM attack as an example.

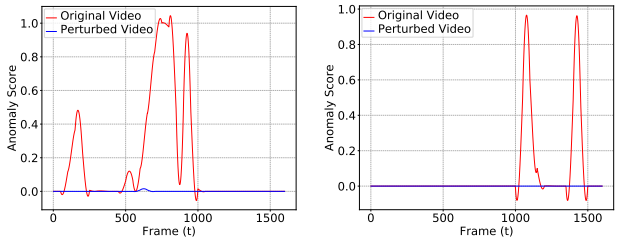
Specifically, we setup a local camera-server network, where one PC works as the surveillance camera to continuously send video streams to another PC (as a server) using the real-time streaming protocol (RTSP). Then, we use the third PC as the adversary running `Ettercap` (https://www.ettercap-project.org/) with ARP poisoning to implement the man-in-the-middle attack (sniffing the network traffic). All three PCs use Ubuntu 18.04 OS, connected on a LAN. According to the recent survey on the security of IP-based video surveillance systems [33], [68], a large number of unencrypted cameras (4.6 millions) are exposed to the network, e.g., using HTTP instead of HTTPS. Although the percentage of such unencrypted cameras is not disclosed, the unencrypted RTSP has been a major security vulnerability in video surveillance [33], [68]. Thus, in our attack setting, we assume that the camera network is open with unencrypted RTSP. By exploiting the vulnerabilities, the adversary will target the camera-server communication without decryption and temporarily intercept the communication session by injecting the TEARDOWN request to the server. When the server tries to send a new request for the new communication session with the camera, the adversary will capture it, modify the delegated client port and forward the request to the camera. Finally, the adversary can receive video streams from the camera in real time.

Note that we utilize the `FFmpeg` compiled with the video encoder `libx264` to execute the codec (decode and encode process) on the video from RTSP streams. Since our attack is performed through unencrypted video streams, there is no extra cost for decrypting video packets. To evaluate the computational overheads for the codec on the video streams, we set the following encoding parameters: (1) PRESET (encoding speed): "medium" by default; (2) bit rate: same as the streaming video bit rate ($\sim$ 350kbps) [1]. The average cost for the codec on the video is $\sim 0.3 < 1$ second, which will not affect the streaming video quality. It can also be accelerated by hardware, e.g., GPU. Overall, we have experimentally shown that the delay of our U3D attack (including both codec and injection) are negligible. Finally, the adversary can forward the perturbed video streams to the server for misclassifications.

TABLE X. Amortized runtime (each frame) for attacking the streaming video on UCF101 (in Seconds). See similar results on UCF Crime in Table XVII (Appendix A).

Video Name	Codec	Inject	Runtime
Bowling	0.010	0.004	0.014
BoxingPunchingBag	0.012	0.005	0.017
CliffDiving	0.010	0.005	0.015
CuttingInKitchen	0.009	0.005	0.014
HorseRace	0.010	0.004	0.014

Evaluation. We randomly pick 10 videos (5 videos from each of UCF101 and UCF Crime) to evaluate the attack against the video classification and anomaly detection. For each video, we repeat 10 times while injecting 10 different U3D perturbations (pre-generated with the HMDB51 dataset and C3D) in the video streams. The attack success rate for classification is 88% (44/50) and for anomaly detection is 98% (49/50). Table X presents the amortized online time for processing each frame. All the runtimes are less than 1/30 (the frame rate is between 24fps and 30fps in experimental videos). Thus, U3D can efficiently attack streaming videos with negligible latency.



(a) Road Accident (U3D$_p$) (b) Explosion (U3D$_g$)

Fig. 4: Real-time attack on anomaly detection

Moreover, Figure 4 presents the real-time anomaly scores of two example videos (i.e., "Road Accident" and "Explosion"), where each streaming video (sent from the camera to the server) is perturbed by a U3D perturbation in real-time (w.l.o.g., U3D$_p$ for "Road Accident" and U3D$_p$ for "Explosion"). In Figure 4(a) ("Road Accident"), we can observe that there are three wave peaks in the original video, e.g., around frame 750, which will trigger the anomaly alarm (reporting "Road Accidents" if the score is greater than a pre-set threshold). While our U3D attack perturbs the streaming video, the anomaly scores of the perturbed video are reduced to almost zero in all the frames. The "Explosion" example (Figure 4(b)) also shows similar results. This illustrates that our U3D can perfectly compromise the video anomaly detection systems.

TABLE XI. Success rates of U3D perturbations (boundary effect-free), injected at 10 different times for each video.

Model / Noise	C3D		I3D	
	UCF101	UCF Crime	UCF101	UCF Crime
U3D$_p$	81.2%	90.3%	80.2%	85.3%
U3D$_g$	84.5%	93.0%	82.6%	89.4%

Finally, to validate the boundary effect-free property of the U3D attack on streaming videos, we conduct another group of experiments on the UCF101 and UCF Crime datasets. Note that the lengths of videos are at least 15 seconds. Then, for each input video in two datasets, we insert the U3D perturbation at 10 different times (from 0 to 5s with a step of 0.5s), and the classification and anomaly detection will start from the first perturbed frame to the end of the video. Table XI summarizes the results for success rates in two applications. We can observe that our U3D perturbations can still achieve high success rates while the misalignment may occur, e.g., U3D$_p$ still achieves 81.2% on UCF101 against C3D, and U3D$_g$ achieves 93.0% on UCF Crime against C3D. This shows that U3D can mitigate the boundary effect well.

VI. Mitigation of U3D Perturbations

The experiments show that the adversarial training (AT) is still the state-of-the-art on improving the model robustness regardless of overheads. The universal AT and AT adapted to U3D (U3D-AT) have shown some effectiveness on reducing the attack success rates (though the accuracy on clean videos has been reduced). To further improve the performance of AT against U3D, we can enhance the search in a larger U3D perturbation space. Also, we can integrate the adaptive inference method, e.g., applying stochastic interpolation to reduce the effect of U3D [55], and certified robustness [18].

For detection methods, the properties of the procedural noise (e.g., low frequency texture structure) can be utilized. For instance, since the background scenes in most surveillance videos captured by static cameras do not change, the defender can extract the static frame of the background and compare it with the perturbed video frame(s) to check the possible perturbation. An alternative way is to check the moving objects or humans in the videos. Since the U3D perturbations applied to the same object in different frames are likely to be different due to the changed coordinates, the deviations between the perturbed object in different frames might be identified. This needs other object detection/tracking algorithms, which may only be suitable for offline analysis due to high overheads.

Furthermore, although certified robustness cannot defend against the U3D, it is promising since it provides theoretical guarantee against norm-bound perturbations. One potential method to improve the robustness is the integration of randomized smoothing with UAT, which could potentially make the trained model robust against more unknown perturbations and thus improve the robust accuracy. However, this also poses challenges on expensive training (not model-agnostic either). We should also address the high dimensionality of videos since the certified guarantee can be jeopardized drastically on high dimensional data under ℓ_∞ bound. Thus, we can execute transformation to reduce the dimension of input data (e.g., by autoencoder) while certifying the robustness after transformation. We will explore these in the future.

Last but not least, we can also mitigate the online U3D attacks by enhancing the security of video recognition systems, e.g., upgrade to encrypted communication channels or add watermarking to the video streams (to detect injections).

VII. Related Work

Security in machine learning, especially the vulnerabilities of AI systems to the adversarial inputs, has been intensively studied in both security and machine learning communities. Since adversarial examples were introduced [3], [27], [73], there have been numerous works on attacking image classifiers. For instance, FGSM [27], PGD [47], UAP [50] and many others [6], [38], [51], work well in the white-box setting. For black-box attacks, researchers have proposed two main types of methods: transfer-based [15], [46], [56] and query-based attacks [5], [13], [14], [31]. Recently, a hybrid attack [71] combines both of them to improve the attack performance. Moreover, adversarial attacks emerge in voice recognition [7], [11], malware classification [28], text understanding [42], etc.

Recent research has extended adversarial attacks from attacking DNNs on 2-D images to 3-D videos [32], [43], [82], [83]. Wei et al. [83] proposed a heuristic algorithm based on the query-based optimization attack [14] to search the saliency region in the video frames for perturbation. V-BAD [32] utilizes natural evaluation strategy (NES) [31] to query the target model for estimating gradient, and then craft adversarial examples via PGD. Both methods compute the perturbation for each frame, which requires heavy computational overheads. They cannot attack real-time videos (due to lack of universality either). C-DUP [43] applies GAN to generate universal perturbations offline and attack real-time video classification. However, it is a white-box attack, which is also limited to only the C3D model. More importantly, due to lack of consistency in the perturbations across frames, all these three attacks can be directly mitigated by AdvIT [85] with high accuracy.

To defend the model against adversarial attacks, a wide range of defense schemes [18], [47], [52], [63], [85], [89] have been proposed, which aim to either improve the robustness of model or detect adversarial examples. To our best knowledge, existing defense schemes (e.g., [41], [47]) mainly work on images, and have not empirically studied videos. Instead, we have thoroughly evaluated our U3D attack by redesigning current defense schemes in Section V-F, which show some effectiveness against the U3D attack on videos. We anticipate that our U3D can motivate to build more robust defense schemes for DNN-based video recognition.

VIII. Conclusion

In this paper, we have successfully constructed two novel U3D perturbations to universally attack multiple DNN-based video recognition systems in the black-box setting. The proposed U3D$_p$ and U3D$_g$ can be efficiently generated on-the-fly while ensuring transferability, universality and human-imperceptibility. Also, U3D$_p$ and U3D$_g$ can be applied to attack video streams in real-time applications. Furthermore, we have conducted extensive experiments on three large-scale video datasets to validate the performance of U3D perturbations by benchmarking with the state-of-the-art attacks. The experimental results demonstrate that our U3D attack greatly outperforms other attacks (e.g., C-DUP) on attack success rate, transferability, and human-imperceptibility. We also perform experiments to evaluate U3D attack against three different types of defense schemes, adapted to universal perturbation or U3D on videos. It is more difficult to defend against U3D compared to C-DUP, V-BAD, and H-Opt (e.g., by AdvIT).

Acknowledgements

This work is partially supported by the National Science Foundation (NSF) under the Grants No. CNS-1745894 and CNS-2046335. We would like to thank Zhaorui Liu and Junwen Chen for their help on some preliminary results and figures. We are also grateful to the anonymous reviewers and the PC point of contact for their constructive comments.

REFERENCES

[1] "Ffmpeg guide," 2020. Available: https://ffmpeg.org/ffmpeg.html

[2] A. Athalye and N. Carlini, "On the robustness of the cvpr 2018 white-box adversarial example defenses," *arXiv:1804.03286*, 2018.

[3] 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 *ECML-KDD*. Springer, 2013, pp. 387–402.

[4] A. Blum, T. Dick, N. Manoj, and H. Zhang, "Random smoothing might be unable to certify ℓ_∞ robustness for high-dimensional images," *JMLR*, vol. 21, no. 211, pp. 1–21, 2020.

[5] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," in *ICLR*, 2018.

[6] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE S&P*, 2017, pp. 39–57.

[7] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, "Hidden voice commands," in *USENIX Security 16*, 2016, pp. 513–530.

[8] N. Carlini and D. Wagner, "Magnet and "efficient defenses against adversarial attacks" are not robust to adversarial examples," 2017.

[9] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the kinetics dataset," in *CVPR*, 2017, pp. 4724–4733.

[10] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[11] G. Chen, S. Chen, L. Fan, X. Du, Z. Zhao, F. Song, and Y. Liu, "Who is real bob? adversarial attacks on speaker recognition systems," in *IEEE S&P*, 2021.

[12] J. Chen, M. I. Jordan, and M. J. Wainwright, "Hopskipjumpattack: A query-efficient decision-based attack," in *S&P*, 2020, pp. 1277–1294.

[13] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 15–26.

[14] M. Cheng, T. Le, P. Chen, H. Zhang, J. Yi, and C. Hsieh, "Query-efficient hard-label black-box attack: An optimization-based approach," in *ICLR*, 2019.

[15] S. Cheng, Y. Dong, T. Pang, H. Su, and J. Zhu, "Improving black-box adversarial attacks with a transfer-based prior," in *NeurIPS*, 2019.

[16] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[17] K. T. Co, L. Muñoz-González, S. de Maupeou, and E. C. Lupu, "Procedural noise adversarial examples for black-box attacks on deep convolutional networks," in *CCS*, 2019, pp. 275–289.

[18] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *ICML*, 2019.

[19] A. Costin, "Security of cctv and video surveillance systems: Threats, vulnerabilities, attacks, and mitigations," in *Proceedings of the 6th international workshop on trustworthy embedded devices*, 2016.

[20] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *CVPR*, 2015.

[21] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95*, 1995, pp. 39–43.

[22] H. J. Escalante, M. Montes, and L. E. Sucar, "Particle swarm model selection," *JMLR*, vol. 10, no. Feb, pp. 405–440, 2009.

[23] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *CVPR*, 2016.

[24] D. Gabor, "Theory of communication. part 1: The analysis of information," *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering*, vol. 93, no. 26, pp. 429–441, 1946.

[25] F. Glover, "Tabu search—part i," *ORSA Journal on computing*, vol. 1, no. 3, pp. 190–206, 1989.

[26] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," 1988.

[27] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *ICLR*, 2015.

[28] 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.

[29] R. Hassan, B. Cohanim, O. De Weck, and G. Venter, "A comparison of particle swarm optimization and the genetic algorithm," in *46th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference*, 2005, p. 1897.

[30] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *CVPR*, 2017.

[31] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in *ICML*, 2018, pp. 2137–2146.

[32] L. Jiang, X. Ma, S. Chen, J. Bailey, and Y.-G. Jiang, "Black-box adversarial attacks on video recognition models," in *MM*, 2019, pp. 864–872.

[33] N. Kalbo, Y. Mirsky, A. Shabtai, and Y. Elovici, "The security of ip-based video surveillance systems," *Sensors*, vol. 20, no. 17, 2020.

[34] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. Li, "Large-scale video classification with convolutional neural networks," in *CVPR*, 2014, pp. 1725–1732.

[35] M. S. Khandare and A. Mahajan, "Mobile monitoring system for smart home," in *2010 3rd International Conference on Emerging Trends in Engineering and Technology*. IEEE, 2010, pp. 848–852.

[36] H. Kuehne, H. Jhuang, E. Garrote, T. A. Poggio, and T. Serre, "HMDB: A large video database for human motion recognition," in *ICCV*, 2011, pp. 2556–2563.

[37] A. Kumar, A. Levine, T. Goldstein, and S. Feizi, "Curse of dimensionality on randomized smoothing for certifiable robustness," in *ICML*, 2020, pp. 5458–5467.

[38] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[39] A. Lagae, S. Lefebvre, R. Cook, T. DeRose, G. Drettakis, D. S. Ebert, J. P. Lewis, K. Perlin, and M. Zwicker, "A survey of procedural noise functions," in *Computer Graphics Forum*, 2010, pp. 2579–2600.

[40] A. Lagae, S. Lefebvre, G. Drettakis, and P. Dutré, "Procedural noise using sparse gabor convolution," *ACM Trans. Graph.*, vol. 28, no. 3, p. 54, 2009.

[41] M. Lécuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified robustness to adversarial examples with differential privacy," in *IEEE S&P*, 2019, pp. 656–672.

[42] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "Textbugger: Generating adversarial text against real-world applications," in *NDSS*, 2019.

[43] S. Li, A. Neupane, S. Paul, C. Song, S. V. Krishnamurthy, A. K. Roy-Chowdhury, and A. Swami, "Stealthy adversarial perturbations against real-time video classification systems," in *NDSS*, 2019.

[44] Z. Li, Y. Wu, J. Liu, Y. Chen, and B. Yuan, "Advpulse: Universal, synchronization-free, and targeted audio adversarial attacks via subsecond perturbations," in *CCS*, 2020, p. 1121–1134.

[45] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in *CVPR*, 2018, pp. 1778–1787.

[46] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *ICLR*, 2017.

[47] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *ICLR*, 2018.

[48] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *CCS*, 2017, pp. 135–147.

[49] Y. Mirsky, T. Mahler, I. Shelef, and Y. Elovici, "Ct-gan: Malicious tampering of 3d medical imagery using deep learning," in *USENIX Security*, 2019, pp. 461–478.

[50] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *CVPR*, 2017, pp. 86–94.

[51] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *CVPR*, 2016, pp. 2574–2582.

[52] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun, "Off-road obstacle avoidance through end-to-end learning," in *NeurIPS*, 2006, pp. 739–746.

[53] J. Obermaier and M. Hutle, "Analyzing the security and privacy of cloud-based video surveillance systems," in *Proceedings of the 2nd ACM international workshop on IoT privacy, trust, and security*, 2016.

[54] T. Onishi, T. Motoyoshi, Y. Suga, H. Mori, and T. Ogata, "End-to-end learning method for self-driving cars with trajectory recovery using a path-following function," in *IJCNN*, 2019, pp. 1–8.

[55] T. Pang, K. Xu, and J. Zhu, "Mixup inference: Better exploiting mixup to defend adversarial attacks," in *ICLR*, 2020.

[56] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *ASIA CCS*, 2017, pp. 506–519.
[57] K. Perlin, "An image synthesizer," *SIGGRAPH Comput. Graph.*, vol. 19, no. 3, p. 287–296, Jul. 1985.
[58] K. Perlin, "Improving noise," *ACM Trans. Graph.*, 2002, p. 681–682.
[59] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *ECCV*, 2010, pp. 143–156.
[60] V. Rausch, A. Hansen, E. Solowjow, C. Liu, E. Kreuzer, and J. K. Hedrick, "Learning a deep neural net policy for end-to-end control of autonomous vehicles," in *ACC*, 2017, pp. 4914–4919.
[61] K. Roth, Y. Kilcher, and T. Hofmann, "The odds are odd: A statistical test for detecting adversarial examples," in *ICML*, 2019, pp. 5498–5507.
[62] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" in *NeurIPS*, 2019.
[63] A. Shafahi, M. Najibi, Z. Xu, J. Dickerson, L. S. Davis, and T. Goldstein, "Universal adversarial training," in *AAAI*, 2020, pp. 5636–5643.
[64] M. Shen, Z. Liao, L. Zhu, K. Xu, and X. Du, "Vla: A practical visible light-based attack on face recognition systems in physical world," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 3, no. 3, 2019.
[65] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *NeurIPS*, 2014, pp. 568–576.
[66] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," *CoRR*, vol. abs/1212.0402, 2012.
[67] S. Srivastava, "amazon-surveillance-cameras-infected-with-malware," 2016. [Online]. Available: https://www.zdnet.com/article/amazon-surveillance-cameras-infected-with-malware/
[68] "Shodan report," 2016, https://www.shodan.io/report/UMAja2tN
[69] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," in *CVPR*, 2018, pp. 6479–6488.
[70] J. Sun, B. Liu, and Y. Hong, "LogBug: Generating Adversarial System Logs in Real Time," in *CIKM*, 2020, pp. 2229–2232.
[71] F. Suya, J. Chi, D. Evans, and Y. Tian, "Hybrid batch attacks: Finding black-box adversarial examples with limited queries," in *USENIX Security*, 2020, pp. 1327–1344.
[72] D. A. Szafir, "Modeling color difference for visualization design," *IEEE TVCG*, vol. 24, no. 1, pp. 392–401, 2017.
[73] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *ICLR*, 2014.
[74] F. Tramer, N. Carlini, W. Brendel, and A. Madry, "On adaptive attacks to adversarial example defenses," in *NeurIPS*, 2020.
[75] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *ICLR*, 2018.
[76] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *ICCV*, 2015, pp. 4489–4497.
[77] D. Tran, H. Wang, L. Torresani, and M. Feiszli, "Video classification with channel-separated convolutional networks," *CoRR*, vol. abs/1904.02811, 2019.
[78] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *CVPR*, 2018, pp. 6450–6459.
[79] F. Van Den Bergh *et al.*, "An analysis of particle swarm optimizers," Ph.D. dissertation, University of Pretoria, 2007.
[80] P. J. Van Laarhoven and E. H. Aarts, "Simulated annealing," in *Simulated annealing: Theory and applications*. Springer, 1987, pp. 7–15.
[81] H. Wang, S. Xie, and Y. Hong, "VideoDP: A Flexible Platform for Video Analytics with Differential Privacy," in *PETS*, 2020, pp. 277–296.
[82] X. Wei, J. Zhu, S. Yuan, and H. Su, "Sparse adversarial perturbations for videos," in *AAAI*, vol. 33, 2019, pp. 8973–8980.
[83] Z. Wei, J. Chen, X. Wei, L. Jiang, T.-S. Chua, F. Zhou, and Y.-G. Jiang, "Heuristic black-box adversarial attacks on video recognition models," in *AAAI*, 2020.
[84] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *ICML*, 2018.
[85] C. Xiao, R. Deng, B. Li, T. Lee, B. Edwards, J. Yi, D. Song, M. Liu, and I. Molloy, "Advit: Adversarial frames identifier based on temporal consistency in videos," in *ICCV*, 2019, pp. 3968–3977.
[86] C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, and A. L. Yuille, "Improving transferability of adversarial examples with input diversity," in *ICCV*, 2019, pp. 2730–2739.
[87] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *NDSS*, 2018.
[88] J. Yang, P. Ren, D. Zhang, D. Chen, F. Wen, H. Li, and G. Hua, "Neural aggregation network for video face recognition," in *CVPR*, 2017, pp. 5216–5225.
[89] X. Yin, S. Kolouri, and G. K. Rohde, "Adversarial example detection and classification with asymmetrical adversarial training," in *ICLR*, 2020.
[90] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *NeurIPS*, 2014, pp. 3320–3328.
[91] Z. Zhou, D. Tang, X. Wang, W. Han, X. Liu, and K. Zhang, "Invisible mask: Practical attacks on face recognition with infrared," 2018.

APPENDIX

A. Additional Experimental Results

TABLE XII. Transferability: transfer rate (TR) on HMDB51 from surrogate model f_{srg} to target model f_{tar}

Noise	f_{srg} \ f_{tar}	C3D	I3D	DN	LRCN	TSN
$U3D_p$	C3D	–	92.0%	89.5%	83.2%	84.6%
	I3D	87.6%	–	91.2%	82.5%	81.4%
	DN	82.3%	81.6%	–	84.5%	80.5%
	LRCN	85.0%	85.4%	95.3%	–	85.6%
	TSN	82.5%	85.7%	88.0%	84.2%	–
$U3D_g$	C3D	–	86.6%	96.4%	81.4%	83.1%
	I3D	92.2%	–	96.0%	88.3%	84.5%
	DN	82.0%	84.8%	–	87.5%	82.3%
	LRCN	85.6%	83.4%	88.2%	–	82.9%
	TSN	84.6%	81.2%	86.1%	84.2%	–

TABLE XIII. Transferability: transfer rate (TR) on UCF Crime from surrogate model f_{srg} to target model f_{tar}

Noise	f_{srg} \ f_{tar}	C3D	I3D	DN	LRCN	TSN
$U3D_p$	C3D	–	91.5%	94.1%	92.3%	89.0%
	I3D	90.7%	–	94.5%	90.2%	89.1%
	DN	87.2%	87.4%	–	88.2%	90.7%
	LRCN	92.8%	87.2%	92.4%	–	86.1%
	TSN	91.7%	90.2%	93.4%	91.7%	–
$U3D_g$	C3D	–	87.0%	93.2%	86.3%	85.3%
	I3D	91.3%	–	93.4%	90.2%	89.0%
	DN	89.3%	88.4%	–	93.4%	89.5%
	LRCN	93.2%	90.8%	91.2%	–	90.4%
	TSN	89.4%	88.6%	92.4%	89.5%	–

TABLE XIV. Universality (success rate (SR); I3D surrogate).

Noise	X \ Y	HMDB51	UCF101	UCF Crime
$U3D_p$	HMDB51	89.6%	80.4%	87.1%
	UCF101	83.5%	86.3%	93.4%
	UCF Crime	80.1%	82.4%	96.0%
$U3D_g$	HMDB51	86.3%	82.9%	90.5%
	UCF101	82.5%	86.7%	91.7%
	UCF Crime	80.1%	84.3%	98.5%

First, Table XII and XIII show additional results for cross-model transferability tested on the HMDB51 and UCF Crime datasets. Table XIV show the results for cross-data universality

by considering I3D as the surrogate DNN to learn the U3D. All the results have confirmed the high transferability and universality of U3D (similar to the results in Section V-C).

TABLE XV. Adversarial training on UCF Crime

Model	Defense	Clean ACR	U3D$_p$ (SR)	U3D$_g$ (SR)
C3D	Normal	92.5%	91.6%	90.7%
	UAT	84.5%	74.7%	75.3%
	U3D-AT	82.4%	62.7%	65.3%
I3D	Normal	95.3%	88.4%	91.2%
	UAT	89.4%	76.2%	80.5%
	U3D-AT	86.2%	58.6%	59.5%

TABLE XVI. Randomized smoothing on UCF Crime

Model	n	Clean ACR	U3D$_p$		U3D$_g$	
			ACR	Radius	ACR	Radius
C3D	100	70.6%	26.3%	0.26	25.1%	0.22
	1000	71.2%	30.2%	0.21	32.4%	0.23
I3D	100	74.6%	28.3%	0.23	25.6%	0.20
	1000	75.1%	29.2%	0.25	26.4%	0.18

TABLE XVII. Amortized runtime (each frame) for attacking the streaming video on UCF Crime (in Seconds)

Video Name	Codec	Inject	Runtime
Arson	0.010	0.004	0.014
Assault	0.010	0.005	0.015
Explosion	0.009	0.007	0.016
Fighting	0.009	0.006	0.015
Shooting	0.010	0.004	0.014

Second, Table XV and XVI present the additional results for U3D against defense schemes. All the results have confirmed the high attack performance even against defense schemes (similar to the results in Section V-F). Table XVII presents the amortized runtime (each frame) for attacking the streaming video (i.e., five videos in the UCF Crime).

B. Particle Swarm Optimization for U3D Parameters

PSO was first developed for social behavior simulation [16], [21], [29], [79] and then commonly applied to optimization, e.g., model selection [22]. Moreover, PSO can be approximated for the optimization with a high-dimensional intense search space and numerous local optimal.

To utilize PSO to optimize U3D perturbations, we first define the fitness function $\mathcal{A}(f, V, \mathcal{N}(T; \vec{s_i}))$, detailed in Algorithm 1. Then we aim to find the optimal particle position (i.e., U3D parameters value) for such fitness function. Algorithm 2 demonstrates the detailed process of U3D optimization. At the beginning, a swarm of m particles denoted as $S = \{\vec{x_1}^k, \vec{x_2}^k, \ldots, \vec{x_m}^k\}$ will be initialized. For each iteration k, each particle i holds a position $\vec{x_i}^k = [x^k_{i,1}, x^k_{i,2}, \ldots, x^k_{i,d}]$, where d is the dimension of the searching parameter space and $x_{i,j}, j \in [1, d]$ indicates the parameter value of jth dimension. To update its position $\vec{x_i}^{k+1} = \vec{x_i}^k + \vec{v_i}^k$, each particle i compute with its current velocity $\vec{v_i}^k = [v^k_{i,1}, v^k_{i,2}, \ldots, v^k_{i,d}]$ as the following equations:

$$v^{k+1}_{i,j} = W * v^k_{i,j} + c_1 * r_1(s_{i,j} - x^k_{i,j}) + c_2 * r_2(s_{g,j} - x^k_{i,j}) \quad (17)$$

$$x^{k+1}_{i,j} = v^k_{i,j} + x^k_{i,j} \quad (18)$$

where (1) $s_{i,j}$ is the value for kth dimension of the best solution searched via particle i so far; $S_i = [s_{i,j}], j \in [1, d]$ is called personal best; (2) $s_{g,j}$ is the value for kth dimension of the best solution in the Swarm S so far; $S_g = [s_{g,j}], j \in [1, d]$ is called leader. Note that every particle can use the S_i (local information) and S_g (social information) to iteratively update its velocity and position. $c_1, c_2 \in \mathbf{R}$ are weights for quantifying the impacts of the personal and social best solution correspondingly; r_1, r_2 is uniformly distributed values of range $[0, 1]$ which represents of randomness in the search. $W = \{W_s, W_f, W_e\}$ is called inertia weight, which can control the impacts of the previous velocity on the current iteration, and then influence searching ability. W will be decreased with every iteration via the following equation: $W = W - \frac{W_s - W_e}{k * W_f}$, where W is initialized as W_s and ended as W_e.

Algorithm 1: Attack Objective $\mathcal{A}(f, V, \mathcal{N}(T; \vec{s}))$

Input: public DNN model f, public video dataset V, current U3D noise parameters $\vec{s}$, sample times I
Output: Output value r of fitness function

Initialize $r \leftarrow 0$
$\xi \leftarrow \mathcal{N}_p$
for $v_i \in V$ **do**
  $t \leftarrow 0$
  `// Sample I times`
  for $i \in I$ **do**
    $\tau \leftarrow U[0, T-1]$
    $t \leftarrow t + \mathcal{D}(v_i, v_i + \texttt{Trans}(\xi, \tau); d)$
  $r \leftarrow r + \frac{t}{I}$
$r \leftarrow \frac{r}{|V|}$
return r

Algorithm 2: `NoiseOpt`(f, V)

Input: U3D function $\mathcal{N}(\cdot)$, DNN model f, video dataset V, ℓ_∞-norm bound ϵ, search space $\mathcal{X}$ for U3D perturbation parameter $\mathcal{S}$; PSO model: inertia weight $W = \{W_s, W_f, W_e\}$, individual/social weights c_1, c_2, swarm size m, maximum iteration number h
Output: optimal parameter set $\mathcal{S}^*$
`// each node has` $||(\mathcal{N}(T; \vec{s_i}))||_\infty \leq \epsilon$

$\mathcal{X}_{sample} \leftarrow$ randomly sample m points from $\mathcal{X}$
for *each* $\vec{s_i} \in \mathcal{X}_{sample}$ **do**
  Call Algorithm 1: $\mathcal{A}(f, V, \mathcal{N}(T; \vec{s_i}))$
  Set personal best of each particle $S_i \leftarrow \vec{s_i}$
Find the leader S_{gb}
Initialize $\vec{s_i}^k \leftarrow \vec{s_i}, i \in [1, m]$
while $k = 1 \leq h$ **do**
  for $i \in [1, m]$ **do**
    Update velocity and position per Equation 17, 18
    Repeat Line 2-4
  Update S_{gb} if leader changes
  Update inertia weight W
return S_{gb} as S^*

(a) U3D$_p$

(b) U3D$_g$

(c) C-DUP [43]

Fig. 5: Some selected frames in videos corresponding to "shooting" and "fighting": (a) perturbed by U3D$_p$, (b) perturbed by U3D$_g$ and (c) perturbed by C-DUP [43]. Both U3D$_p$ and U3D$_g$ show good human-imperceptibility compared with C-DUP.

C. Comparison of PSO with Other Meta-Heuristic Algorithms

We use the C3D model as the public DNN model and randomly sample 500 videos from the HMDB51 dataset as the public dataset. The ϵ is set as a small bound 8. The parameter of the normalization α is set to 0.5. Table XVIII shows the specified value ranges of the parameters for U3D$_p$ and U3D$_g$, respectively. As for PSO, we set up the parameters as follows: (1) swarm size $m = 20$; (2) individual and social weight $c_1 = c_2 = 2$; (3) inertia weight $W = \{1.2, 0.5, 0.4\}$; (4) maximum iteration times $h = 40$.

TABLE XVIII. U3D parameters setting

U3D$_p$			U3D$_g$		
$\lambda_x, \lambda_y, \lambda_t$	Λ	ϕ	K	σ	F
$[2, 180]$	$[1, 5]$	$[1, 60]$	$[1, 5]$	$[1, 20]$	$[0.25, 20]$

Then, we compare PSO with genetic algorithm (GA) [29], simulated annealing (SA) [80], and Tabu search (TS) [25] on tuning the U3D parameters. For GA, we set the number of chromosomes to be 20 (same as the number of PSO's particles) with the combination of tournament selection with a 50% uniform crossover probability [29] and mutation rate 0.5%. For SA, we set the initial temperature is 5000, and cooling factor 0.99. For TS, the tabu list size is set to 4. We implement the four methods for both U3D$_p$ and U3D$_g$ on the 500 videos, which are repeated 5 times and averaged for the final results. Table XIX illustrates their experimental results. We can observe that the PSO-based method is efficient, and also slightly outperforms GA, SA and TS on attack performance, e.g., PSO improves 1.7% over GA and 3.3% over SA for U3D$_p$. Besides, the MSE of both U3D perturbations are below 20 (very minor distortion out of 255^2 in the scale).

TABLE XIX. PSO vs. GA, SA and TS (learning U3D parameters offline) for U3D$_p$ and U3D$_g$ (success rate "SR").

Method	U3D$_p$			U3D$_g$		
	Time (s)	SR	MSE	Time (s)	SR	MSE
PSO	847	88.7%	15.3	789	89.6%	16.0
GA	1,481	87.0%	14.6	1,164	89.4%	17.8
SA	1,976	85.4%	16.9	2,267	87.7%	13.7
TS	1,039	86.5%	17.5	822	88.1%	15.8

D. Discussions

U3D Stealthiness. Stealthiness of adversarial attacks can be reflected with two properties: (1) human-imperceptibility with minor perturbations; (2) misclassified outputs are not always rare. U3D can ensure much better human-imperceptibility (as validated in Section V), compared to the white-box attack C-DUP [43]. For the latter one, as a black-box attack, adversaries of U3D do not know the class labels and their distributions (different from the white-box attack [43]), and inject the U3D perturbations to misclassify the perturbed videos to "random class labels". If the adversaries prefer to further improve stealthiness by "supervising" such misclassification to "common class labels" in the attack, they can query the target DNN model (similar to the traditional black-box attack) to retrieve some class labels and their distribution, and then adjust the objective function/constraint(s) in the U3D parameters learning to tune the U3D perturbation towards such goal. This also applies to other DNN-based video recognition systems, such as anomaly detection. We will explore these in the future.

Potential Advanced Attacks. Our U3D obtains good transferability to attack the video recognition systems in the black-box setting. To further improve the attack performance, we can utilize the query-based black-box attack to determine the attack gradient direction with a limited number of queries [12], [13]. We can also improve the U3D attack via learning the U3D perturbations on an ensemble of diverse models [75] or diverse inputs [86]. In summary, the adversarial attack can be more adaptive in the arms race.

Attacking New and Future DNNs. To date, besides the

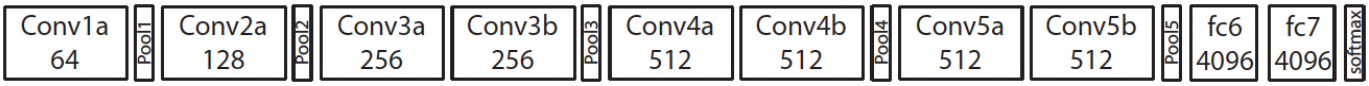


Fig. 6: The C3D architecture [76] consists of 8 convolution, 5 max-pooling, and 2 fully connected layers, followed by a softmax output layer. All 3D convolution kernels are 3 ×3 ×3 with a stride of 1 in both spatial and temporal dimensions. The number of filters is denoted in each box [76]. The 3D pooling layers are represented as pool1 to pool5. All pooling kernels are 2 × 2 × 2, except for pool1, which is 1 × 2 × 2. Each fully connected layer has 4,096 output units.

commonly-used C3D [76] and I3D [9], new DNN models built on the 3D spatio-temporal convolutions are proposed all the time, such as R(2+1)D [78], and CSN [77]. For example, the recently proposed Channel Separated Convolutional Network (CSN) [77] factorizes 3D convolution via separating spatio-temporal and channel interactions, which improves both the accuracy and computation efficiency. Considering that our U3D perturbations work directly on spatio-temporal features across consecutive frames, we anticipate that those new spatio-temporal features may also be vulnerable to U3D perturbations. We plan to experimentally validate such vulnerabilities (in case of U3D perturbations) in the newly proposed and future DNN models as soon as they have been deployed.

Attacking Image Classifiers. It is straightforward to downgrade our U3D attack to DNN models on images. Since the image can be considered as a 1-frame video. We only need to fix t to 1 to generate the 2-D (frame) perturbation.

Physical U3D Attack. The U3D attack can be extended to the physical adversarial attack in real world [38]. For example, our U3D perturbations can be integrated with the visual light technology (e.g., smart LED) [64], [91], with which the U3D perturbations can be projected (realizing the U3D perturbations with more programmable light building block) on the scenes inconspicuously. In addition, with computer graphic primitives, our U3D perturbations look visually like natural textures, then the manipulated light will not easily be discerned by humans. We will explore this in the future.

E. Video Recognition Systems and Models

1) Summary of Difference among Video DNN Models: The five DNN models (C3D, I3D, DN, LRCN, and TSN) in the transferability evaluations are very different.

First, C3D and I3D are using 3D convolution (3DConv) kernels, while DN, LRCN, and TSN are using 2D convolution kernels. Therefore, C3D and I3D can naturally capture temporal information in their convolution operations, while DN, LRCN, and TSN require additional components to capture the temporal information. Second, C3D is a shallow network with only 8 convolution layers, and its convolution kernel sizes are predefined. Although I3D is also based on the 3D (spatio-temporal) feature extraction, it requires another extra stream of optical flow information, which is also very different from the C3D in the model architecture. Furthermore, other three models (DN, LRCN, and TSN) have totally different architectures. DN is purely based on 2D convolution networks for images. It uses 2D convolution to extract features from each video frame, and then fuses features from multiple frames. LRCN also uses a 2D convolution network, but it uses a long short-term memory (LSTM) to fuse features from multiple frames. TSN uses two convolution networks, one for video frames, and the other one for optical information computed from frames.

2) C3D Network: Figure 6 shows the detailed structure and the characteristics of the C3D network.

3) DNN-based Anomaly Detection: Figure 7 illustrates a video anomaly detection system with the I3D model (by integrating additional optical flow information into the spatio-temporal features). Figure 8 illustrates the curve of anomaly scores inferred by the I3D model in time series. At first, the score is close to 0 for the normal scenery. Then, it increases as the explosion occurs to report such anomalous event (with a threshold).

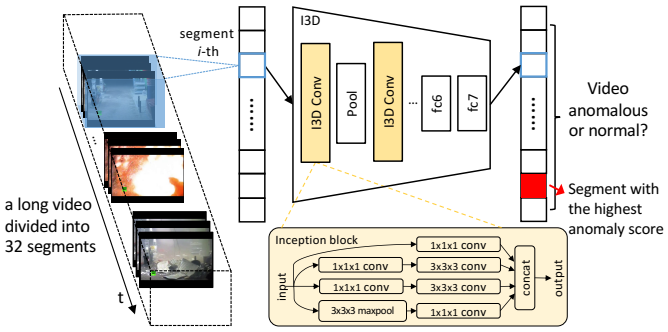


Fig. 7: Anomaly detection [69] with I3D.

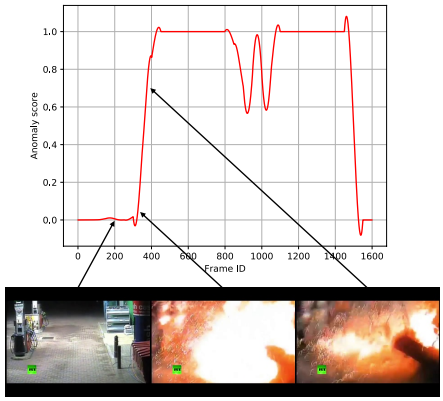


Fig. 8: Video anomaly detection system for detecting an "Explosion" event. The score increases from 0 (normal scenery) to high (explosion).