DUO: Stealthy Adversarial Example Attack on Video Retrieval Systems via Frame-Pixel Search

Xin Yao*, Yu Zhan*, Yimin Chen†§, Fengxiao Tang*, Ming Zhao*, Enlang Li*, Yanchao Zhang‡§
*School of Computer Science and Engineering, Central South University, Changsha, Hunan, 410082, China
†Miner School of Computer & Information Sciences, University of Massachusetts Lowell, Lowell, MA 01854, USA
‡School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287, USA
§Corresponding authors

2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS) | 979-8-3503-3986-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDCS57875.2023.00044

***Abstract*—Massive videos are released every day particularly through video-focused social media apps such as TikTok. This trend has fostered the quick emergence of video retrieval systems, which provide cloud-based services to retrieve similar videos using machine learning techniques. Adversarial example (AE) attacks have been shown to be effective on such systems by perturbing an unaltered video *subtly* to induce false retrieval results. Such AE attacks can be easily detected because the adversarial perturbations are all over pixels and frames. In this paper, we propose DUO, a stealthy targeted black-box AE attack which uses *DU*al search *O*ver frame-pixel to generate *sparse* perturbations and improve stealthiness. DUO is motivated by two observations: only "key frames" in a video decide model predictions, and different pixels and frames contribute far differently to AEs. We implement DUO into a sequential attack pipeline consisting of two components (i.e., SparseTransfer and SparseQuery) built upon such intuitions. In particular, DUO uses SparseTransfer to generate initial perturbations and then SparseQuery to further rectify them. Extensive evaluations on two popular datasets confirm the higher efficacy and stealthiness of DUO over existing AE attacks on video retrieval systems. In particular, we show that DUO achieves higher precision while significantly reducing adversarial perturbations by more than $\times 100$ than the state-of-the-art AE attack.**

***Index Terms*—sparse targeted adversarial example attack, video retrieval system, black-box attack, stealthiness**

I. Introduction

Videos have become one of the most important media with the quick rise of video-focused social networks apps like Instagram and TikTok. Popular apps covering entertainment, advertisement, and communications are generating a huge amount of video data daily. Major statistics show that about 500 hours of videos were uploaded to YouTube every minute in July 2021,[1] and the Zoom annual meeting minutes had exceeded 3.3 trillion by the third quarter of 2021.[2] Consequently, it is imperative to efficiently search and retrieve videos of interest efficiently from an ever-growing large database.

Recent years have witnessed the quick emergence of cloud-based video retrieval services exploring deep neural networks (DNN) [1]. As shown in Figure 1, upon receiving a query video from an end user, the DNN-based video retrieval system converts it into the feature space spanned by temporal and spatial features, locates videos in various distributed data nodes that are close to it in the feature space, and output them as the retrieval result. Popular models for video retrieval include reverse search [1], [2], video comparison retrieval [3], [4], face video retrieval [5], [6], etc. Unfortunately, recent studies have shown that these DNN-based systems are vulnerable to adversarial example (AE) attacks [7]–[12] where the attacker can fool the victim system to output wrong retrieval results by adding subtle perturbations to a query video. Existing defense mechanisms [13], [14] can detect certain query accounts with "adversarial behavior", but the adversary can easily evade such detection by using different query accounts which are fairly easy to create/purchase. AE attacks thus remain a severe threat to DNN-based video retrieval systems.

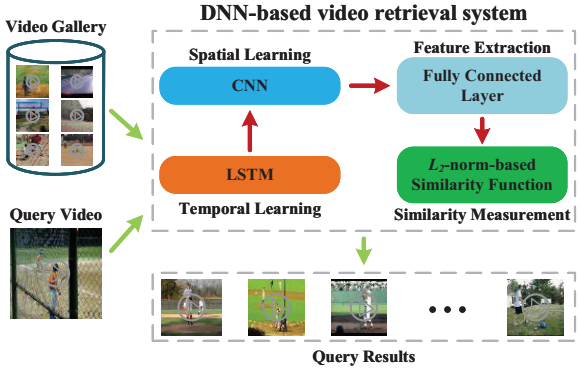


Fig. 1. Overview of a DNN-based video retrieval system.

AE attacks on DNN-based video retrieval systems can be classified into *untargeted attacks* in which retrieval results include arbitrary videos except for the correct ones and *targeted attacks* in which retrieval results are specific videos of the attacker's choice. In this paper, we focus on the more challenging targeted attacks, while our method can be easily extended to launch untargeted attacks as well. Apparently, incorrect retrieval results would severely harm the video retrieval systems in the long run. For instance, popular social media platforms supporting user-generated content may explore video retrieval services to automatically check if each submitted video is indeed original instead of modified from existing online videos. A malicious user can then launch the AE attack to submit and publish plagiarized videos for which no matching videos are returned by video retrieval systems

[1]https://www.omnicoreagency.com/youtube-statistics/
[2]https://backlinko.com/zoom-users

2575-8411/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDCS57875.2023.00044

to the querying social media platform. As a related example in copyright protection, a video owner may check whether her/his videos are protected by retrieving the top-k results from the database for each video and confirming whether there are very similar (even exactly the same) ones among the results. In this case, the adversary can bypass such copyright violation detection by publishing an adversarial example for a copyrighted video that is not included in the retrieval results to the video owner.

There are some existing attacks on DNN-based video retrieval systems such as transfer-based and query-based schemes [15]–[17]. These attacks assume a black-box attack scenario in which the attacker has no information about the structure and parameters of the target DNN model and suffer from some key drawbacks. Foremost, these attacks add *dense* perturbations to an original video, which make the synthesized AEs easily perceived and detected. In addition, it is challenging for transfer-based AE attacks [15] to achieve a high success rate because they completely rely on a surrogate model well approximates the victim model, which is very difficult to attain in practice. Moreover, query-based AE attacks [16], [17] exhibit low attack efficacy because the high dimensionality of video data involves too large space to search for an effective yet undetectable AE.

In this paper, we present **DUO**, a stealthy AE attack on DNN-based video retrieval systems based on *DU*al search *O*ver frames and pixels for generating sparse perturbations. The novelty of DUO lies in incorporating two observations into the formulation and generation of video AEs. First, we conjecture that only a few frames, i.e., "key frames", of a given video play the main role in video retrieval systems. Therefore, we consider the number of perturbed frames as a design constraint when formulating AE generation. Second, previous research confirms that different pixels and frames contribute differently to altering the prediction of a synthesized AE. Therefore, we further add the number of perturbed pixels as a design constraint. By doing so, we can apply sparsification to perturbations by conditioning the number of perturbed pixels and frames for synthesized AEs. We implement perturbation sparsification in a sequential attack pipeline, which consists of a transfer-based component and a query-based component, denoted by SparseTransfer and SparseQuery, respectively. Given an input video $\boldsymbol{v}$, SparseTransfer simply computes initial perturbations conditioned by sparsification, and SparseQuery further polishes the perturbations and maintains their sparsity level. We loop over the two and derive the optimal $\boldsymbol{v}_{adv}$.

Our contributions in this paper are summarized as follows.

- We propose DUO, a stealthy AE attack on popular DNN-based video retrieval systems. DUO achieves a higher level of stealthiness than existing attacks by sparsing adversarial perturbations with dual frame-pixel search. In addition, it is designed as a targeted AE attack under the black-box setting. All these factors render DUO a practical, severe threat to video retrieval services.
- We propose to incorporate perturbation sparsification into formulating AE generation and derive approximated solutions using the tooling in [10], [18].
- We extensively evaluate the performance of our DUO attack on two benchmark datasets (UCF101 [19] and HMDB51 [20]), over four video retrieval models (I3D [21], TPN [22], SlowFast [23], and Resnet34 [24]), and under different selections for system parameters. We also compare DUO with three state-of-the-art attacks, including a Vanilla attack, `TIMI` [25], and `HEU` [16]). Our evaluations confirm that DUO consistently achieves higher precision with much reduced adversarial perturbations than existing attacks. For example, when compared to `TIMI` [25], DUO achieves higher precision while reducing adversarial perturbations by more than $\times 100$. Finally, we confirm that DUO is more robust against two popular defenses (`feature squeezing` [26] and `Noise2Self` [27]).

II. Related Work

In this section, we introduce previous research relevant to our study.

Adversarial attacks on retrieval systems. Adversarial attacks on image retrieval systems can be divided into two categories: white-box and black-box attacks. White-box attackers have the full knowledge of the victim systems. Ref. [28] proposed an untargeted AE attack that simply maximized the distance between the original and adversarial images in the decision space while Refs. [29], [30] designed a targeted attack that minimized such a distance. Different from previous works, Ref. [31] proposed a generative adversarial network to directly synthesize AEs. In contrast, black-box attacks do not have full access to the model architecture and parameters of the victim systems and thus are much more challenging. Refs. [11], [15] were among the early works focusing on such attacks on image retrieval systems. So far, most research focused on image retrieval systems while Ref. [32] was the only one exploring adversarial attacks on video hash retrieval systems to the best of knowledge. Nonetheless their attack [32] was a white-box attack and perturbed every frame and every pixel of an original video, rendering the generated adversarial videos easy to detect in human eyes. In comparison, we believe that it is important to further explore stealthy attack under black-box setting due to its high impact.

Sparse AE attacks on image classifiers. There has been extensive research on sparse AE attacks under white-box setting in the image domain [33]–[37]. Unlike dense attacks [38], sparse AE attacks perturbed much less pixels (so-called 'sparsification') that have a higher impact on predictions in order to reduce visual perceptibility. Refs. [33]–[37] proposed novel schemes to select pixels and compute the corresponding perturbation magnitude. Nonetheless these attacks were designed for image classifiers and under white-box setting while we work on a more challenging task of designing black-box attacks on video retrieval systems.

Black-box AE attacks on video classifiers. Black-box setting urged attackers to opt for query-based schemes on video classifiers in which the attackers aimed to better the

generated AEs using prior query results. The key focus along this research path is to improve query efficiency (i.e., reduce the number of queries) by properly exploiting "prior knowledge" before launching an attack [16], [39]–[41]. Examples of such prior knowledge include pretrained model for initialization [39], motion correlation among video frames [40], and saliency map [16], [41]. Here we continue along this line while focusing on video retrieval models and improving attack stealthiness by designing proper prior knowledge in accordance to intuitions.

III. System and Adversary Model

A. Video Retrieval System

Let $\boldsymbol{v} \in \mathbb{R}^{N\times W\times H\times C}$ denote a video with N frames, where W, H, and C represent the width, height, and channel of each frame, respectively. In this paper, we target at the popular distributed video retrieval systems consisting of an end-to-end deep model. As shown below, our attack does not depend on the deep model itself and we adopt the widely used one from [42] illustrated in Figure 1. The deep model uses a long short-term memory and a stacked convolution neural network for temporal and spatial feature extraction, fully-connected layers for feature flattening, and a similarity function (e.g., ℓ_2-norm based) for computing a list of similar videos. Given a query video as the input, the deep model outputs a list of videos from a gallery that are similar to the query one. Denote the video retrieval model, the video gallery, and the query video by $\mathcal{R}(\cdot)$, $\mathbf{G}$, and $\boldsymbol{v}$, respectively. Then the output list can be denoted by

$$\mathcal{R}^m(\boldsymbol{v}) = \{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_i, \cdots, \boldsymbol{v}_m | \boldsymbol{v}_i \in \mathbf{G}\} ,$$

and m is the number of returned videos. Note that the output list is in descending order meaning that the smaller i the more similar $\boldsymbol{v}_i$ is to $\boldsymbol{v}$, i.e., the query video.

B. Adversary Model

As mentioned in Section I, we focus on black-box attacks as such attacks have high practicality and impact in real world scenarios. Hence first and foremost, we assume that the attacker knows only the public information of the victim model including the format requirements for input videos and output list (i.e., $\mathcal{R}^m(\cdot)$) but no other non-public information such as model architecture and parameters. Next we follow existing transfer-based attacks [10] to assume that the attacker is able to obtain sufficient training samples to train a surrogate model. In our evaluation, the maximum number of training samples the attacker obtains is 8,421 for both UCF101 [19] and HMDB51 [20]. Finally we follow existing query-based attacks [16], [17], [39]–[41] to assume that the attacker is able to obtain the prediction results, i.e., the retrieved video list, from the victim model.

C. Attacker's Goals

Denote the original video, the corresponding adversarial video, and the target video by $\boldsymbol{v}$, $\boldsymbol{v}_{adv}$, and $\boldsymbol{v}_t$, respectively. Our DUO attack generates $\boldsymbol{v}_{adv}$ from $\boldsymbol{v}$ by adding carefully-crafted perturbations $\boldsymbol{\phi} \in \mathbb{R}^{N\times W\times H\times C}$, i.e., $\boldsymbol{v}_{adv} = \boldsymbol{v} + \boldsymbol{\phi}$. DUO aims to achieve that the retrieval list of $\boldsymbol{v}_{adv}$ (i.e., $\mathcal{R}^m(\boldsymbol{v}_{adv})$) is as similar as possible to that of $\boldsymbol{v}_t$ (i.e., $\mathcal{R}^m(\boldsymbol{v}_t)$) while $\boldsymbol{v}_{adv}$ remains similar to $\boldsymbol{v}$ in human eyes. More specifically, DUO aims to be:

- **A targeted attack**: The retrieval lists of $\boldsymbol{v}_{adv}$ and $\boldsymbol{v}_t$ are similar, i.e., $\mathcal{R}^m(\boldsymbol{v}_{adv})$ is as close to $\mathcal{R}^m(\boldsymbol{v}_t)$ as possible.
- **A stealthy attack**: The added perturbation $\boldsymbol{\phi}$ should be imperceptible to human eyes. To start with, the exact perturbation for each pixel should not exceed a preset threshold, i.e., $\|\boldsymbol{\phi}\|_\infty = \max\{|\boldsymbol{\phi}_{i,w,h,c}|\} \leq \tau$, where τ is the threshold. More importantly, our attack aims to sparse the perturbed pixels and frames (i.e., reduce the number of perturbed pixels and frames) to improve its stealthiness. Let $\boldsymbol{\phi}_i$ denote the perturbation for the i-th frame, $\|\boldsymbol{\phi}_i\|_0 = \sum_j |\boldsymbol{\phi}_{i,j}|^0$ the number of non-zero elements in $\boldsymbol{\phi}_i$, and $\|\boldsymbol{\phi}\|_{2,0} = \sum_i \|\sum_j \boldsymbol{\phi}_{i,j}^2\|_0$ the number of non-zero rows in $\boldsymbol{\phi}$. Let B and N be the number of pixels in each frame and frames in the whole video, respectively. Then the number of perturbed pixels (or frames) in the i-th frame (or the video) should be as small as possible, i.e., $\|\boldsymbol{\phi}_i\|_0 \ll B$ and $\|\boldsymbol{\phi}\|_{2,0} \ll N$.

IV. Methodology

In this section, we first introduce the overview and then the detailed steps of our DUO attack.

A. Overview and a User Case

Figure 2 shows the overview of our DUO attack consisting of two components (i.e., SparseTransfer and SparseQuery) and a user case. As mentioned above, given an intact video and a target video (denoted by $\boldsymbol{v}$ and $\boldsymbol{v}_t$, respectively), DUO generates the corresponding AE (denoted by $\boldsymbol{v}_{adv}$). $\boldsymbol{v}$, $\boldsymbol{v}_t$, and $\boldsymbol{v}_{adv}$ are in the same format. Recall that we use W, H, and C to denote the width, height, and channel of video frame, respectively, while N the number of frames. In UCF101, $W = H = 112, C = 3, N = 16$. As shown in Figure 2, while $\boldsymbol{v}$ is labelled 'Run', DUO aims to perturb $\boldsymbol{v}$ to $\boldsymbol{v}_{adv}$ with subtle changes so that $\boldsymbol{v}_{adv}$ can retrieve videos labelled 'Clap', i.e., the same label as $\boldsymbol{v}_t$. The attack process can be described in the following. First, SparseTransfer in DUO launches frame-pixel search using a trained surrogate model and derives initial sparse perturbations for $\boldsymbol{v}$. In short, SparseTransfer outputs $\boldsymbol{\mathcal{I}}$, $\boldsymbol{\mathcal{F}}$, and $\boldsymbol{\theta}$ while the three adds up as the initial perturbation, i.e., $\boldsymbol{\phi} = \boldsymbol{\mathcal{I}} \odot \boldsymbol{\mathcal{F}} \odot \boldsymbol{\theta}$. Following that, SparseQuery finetunes $\boldsymbol{\phi}$ using queries to the victim model and derive the adjusted $\boldsymbol{\phi}$. In our implementation, we loop the two components to avoid being trapped at a local optimal. In the end, DUO synthesizes $\boldsymbol{v}_{adv}$ by $\boldsymbol{v}_{adv} = \boldsymbol{v} + \boldsymbol{\phi}$ and uses $\boldsymbol{v}_{adv}$ to retrieve similar videos to those of $\boldsymbol{v}_t$. For the above use case, $\boldsymbol{v}_{adv}$'s retrieved videos are labelled 'Clap' rather than 'Run', thus indicating that DUO achieves a targeted attack.

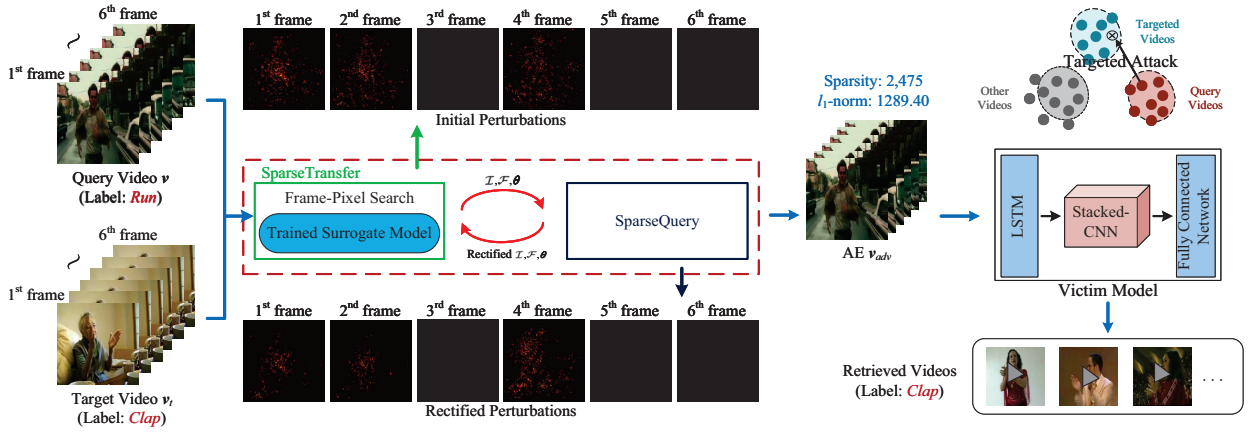


Fig. 2. Step-by-step illustration of DUO pipeline. Assume $\boldsymbol{v}$ has six frames. $\boldsymbol{\mathcal{I}}$, $\boldsymbol{\mathcal{F}}$, and $\boldsymbol{\theta}$: the selected pixels, the selected frames, and the perturbation magnitude.

B. SparseTransfer

In this component, we first train a surrogate model approximating the target victim model and use it for generating initial adversarial perturbations conditioned by frame-pixel sparsification.

1) Building a Surrogate Model: In SparseTransfer, a surrogate model serves in place of a white-box model approximating the black-box victim model. As a result, SparseTransfer can generate (initial) perturbations for a given $\boldsymbol{v}$. Denote the surrogate model and victim model by $\mathcal{S}(\cdot)$ and $\mathcal{R}(\cdot)$, respectively. The key to train $\mathcal{S}(\cdot)$ is to construct a proper training set while the exact backbone model is less important due to the transferability of AEs [15]. Below we briefly describe how we construct the training set (starting as an empty list denoted by $\boldsymbol{T}$). $\boldsymbol{v}_r$ denotes a random video. Each row of $\boldsymbol{T}$ corresponds to a training sample.

- **Step 1:** We upload $\boldsymbol{v}_r$ to $\mathcal{R}(\cdot)$, obtain $\mathcal{R}^m(\boldsymbol{v}_r)$, and append $\mathcal{R}^m(\boldsymbol{v}_r)$ to $\boldsymbol{T}$. That is, $\boldsymbol{T} = \{\langle \boldsymbol{v}_r, \boldsymbol{v}_i, \boldsymbol{v}_j \rangle | \boldsymbol{v}_i, \boldsymbol{v}_j \in \mathcal{R}^m(\boldsymbol{v}_r), i < j \in [1, m]\}$.
- **Step 2:** We uniformly select M videos from $\mathcal{R}^m(\boldsymbol{v}_r)$ and repeat **Step 1**.
- **Step 3:** We repeat **Step 1 and 2** for Z times and obtain the final $\boldsymbol{T}$ as the training set.

Finally, we use $\boldsymbol{T}$ as the training set, a typical video retrieval backbone from [43], and the following loss function [10], [44] to obtain $\mathcal{S}(\cdot)$:

$$\arg\max_{\boldsymbol{\rho}} \sum_{j>i} [D(\boldsymbol{v}, \boldsymbol{v}_j) - D(\boldsymbol{v}, \boldsymbol{v}_i) + \gamma]_+ \ .$$

Here $\boldsymbol{\rho}$ is the model parameters of $\mathcal{S}(\cdot)$, $D(\boldsymbol{v}, \boldsymbol{v}_i) = \|\texttt{Fea}_{\boldsymbol{\rho}}(\boldsymbol{v}) - \texttt{Fea}_{\boldsymbol{\rho}}(\boldsymbol{v}_i)\|_2^2$ is the feature distance between $\boldsymbol{v}$ and $\boldsymbol{v}_i$, $\texttt{Fea}_{\boldsymbol{\rho}}(\boldsymbol{v})$ is $\boldsymbol{v}$'s projected feature, and γ is a margin constant to make sure the loss value stays positive and set as 0.2 in this paper. More details about the stealing process can be found in the supplementary material at [45].

2) Computing Initial Perturbations with Sparsification: In this part, we use $\mathcal{S}(\cdot)$ to derive initial perturbations for $\boldsymbol{v}$ while following the intuitions to sparse perturbations on pixels and frames. We illustrate the process in Figure 2 and Algorithm 1 as well. The input are $\boldsymbol{v}$, $\boldsymbol{v}_t$, and $\mathcal{S}$ while the output are $\boldsymbol{\mathcal{I}}$, $\boldsymbol{\mathcal{F}}$, and $\boldsymbol{\theta}$, which correspond to the selected pixels, the selected frames, and the perturbation magnitude, respectively. $\boldsymbol{\mathcal{I}}$, $\boldsymbol{\mathcal{F}}$, and $\boldsymbol{\theta}$ are termed as "prior knowledge" for AE generation as well. As mentioned in Section I, we conjecture that

1) different pixels and frames contribute differently to the targeted attack goal, while
2) reducing the number of perturbed pixels and frames (thus frame-pixel sparsification) makes $\boldsymbol{v}_{adv}$ less perceptible, i.e., stealthier.

Consequently, we formulate AE generation on $\mathcal{S}(\cdot)$ as the following:

$$\begin{aligned} &\arg\min_{\boldsymbol{\theta}, \boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{F}}} \mathcal{L}(\texttt{Fea}_{\boldsymbol{\rho}}(\boldsymbol{v}_{adv}), \texttt{Fea}_{\boldsymbol{\rho}}(\boldsymbol{v}_t)) + \lambda \|\boldsymbol{\theta} \odot \boldsymbol{\mathcal{I}} \odot \boldsymbol{\mathcal{F}}\|_2^2 \\ &\quad s.t. \quad \mathbf{1}^\top \boldsymbol{\mathcal{I}} = k, \|\boldsymbol{\mathcal{F}}\|_{2,0} = n, \|\boldsymbol{\theta}\|_\infty \leq \tau \ . \end{aligned} \tag{1}$$

In the above formulation, λ is a constant for regularization, $\mathcal{L}(\cdot, \cdot)$ is an Euclidean distance function, τ is a threshold to limit the maximum allowable perturbation on arbitrary pixel (i.e., $\|\boldsymbol{\theta}\|_\infty \leq \tau$). We emphasize that *the constraints are where we implement the intuitions*. Specifically, $\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{F}}, \boldsymbol{\theta} \in \{0, 1\}^{N \times W \times H \times C}$, $\boldsymbol{\mathcal{I}}_{i,w,h,c} = 1$ indicates that the (w, h, c)-th pixel in the i-th frame is selected as a perturbed one ($\boldsymbol{\mathcal{F}}$ is defined similarly), $\boldsymbol{\phi} = \boldsymbol{\mathcal{I}} \odot \boldsymbol{\mathcal{F}} \odot \boldsymbol{\theta}$ is the output perturbation for $\boldsymbol{v}$. Hence $\mathbf{1}^\top \boldsymbol{\mathcal{I}} = k$ means that the number of perturbed pixels in the query video $\boldsymbol{v}$ is k and $\|\boldsymbol{\mathcal{F}}\|_{2,0} = n$ applies to the number of perturbed frames in $\boldsymbol{v}$. *By enforcing smaller k and n*, Equation (1) effectively sparses perturbations for $\boldsymbol{v}_{adv}$, thus making $\boldsymbol{v}_{adv}$ stealthier. More details about the optimization process can be found in the supplementary material at [45].

As Equation (1) reveals itself as a mixed integer programming problem [46], we follow the tooling in [18] and iterate to update $\{\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{F}}, \boldsymbol{\theta}\}$ until they converge. More details can be found in the supplementary material at [45] regarding how to formulate Equation (1) (including choosing $\mathcal{L}(\cdot, \cdot)$

Algorithm 1: SparseTransfer($\boldsymbol{v}$, $\boldsymbol{v}_t$, $\mathcal{S}$)

Input: $\boldsymbol{v}$, $\boldsymbol{v}_t$, $\mathcal{S}$;
Output: Pixel-mask $\boldsymbol{\mathcal{I}}$, Frame-mask $\boldsymbol{\mathcal{F}}$, perturbation magnitude $\boldsymbol{\theta}$;

Initialize $\boldsymbol{\mathcal{I}}$ and $\boldsymbol{\mathcal{F}}$ as $\mathbf{1}$ and $\boldsymbol{\theta}$ as $\mathbf{0}$;
while *not converge* **do**
  Given $\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{F}}$, update $\boldsymbol{\theta}$ via gradient descent under $\mathcal{S}$;
  Given $\boldsymbol{\theta}, \boldsymbol{\mathcal{F}}$, update $\boldsymbol{\mathcal{I}}$ with ADMM;
  Replace $\boldsymbol{\mathcal{F}}$ with a continuous variable $\boldsymbol{\mathcal{C}}$;
  Given $\boldsymbol{\mathcal{I}}, \boldsymbol{\theta}$, update $\boldsymbol{\mathcal{C}}$ following [47];
  Update $\boldsymbol{\mathcal{F}}$ based on $\|\boldsymbol{\mathcal{C}}_{\pi(1)}\|_2 \geq \cdots \geq \|\boldsymbol{\mathcal{C}}_{\pi(N)}\|_2$;
end
return $\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{F}}$, and $\boldsymbol{\theta}$;

Algorithm 2: SparseQuery($\boldsymbol{v}, \boldsymbol{v}_t, \boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{F}}, \boldsymbol{\theta}$)

Input: $\boldsymbol{v}$, $\boldsymbol{v}_t$, $\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{F}}$, $\boldsymbol{\theta}$;
Output: $\boldsymbol{v}_{adv}$;

Initialize $\boldsymbol{v}_{adv}^{(0)}$ as $\boldsymbol{v}$;
Calculate $\mathbb{T}^{(0)}$ using Equation (2);
Initialize ϵ from $\boldsymbol{\theta}$, $\mu = \texttt{iter\_numQ}$, $\kappa = 1$;
while $\kappa < \mu$ **do**
  Randomly sample $\boldsymbol{q}^{(\kappa)}$ from the Cartesian basis without replacement;
  for $\xi \in \{+\epsilon, -\epsilon\}$ **do**
    $\boldsymbol{v}' = \texttt{CLIP}(\boldsymbol{v}_{adv}^{(\kappa-1)} + \xi \boldsymbol{q}^{(\kappa)})$;
    Calculate $\mathbb{T}^{(\kappa)}$ using Equation (2);
    if $\mathbb{T}^{(\kappa)} < \mathbb{T}^{(\kappa-1)}$ **then**
      $\boldsymbol{v}_{adv}^{(\kappa)} = \boldsymbol{v}'$;
      break;
    end
    $\mathbb{T}^{(\kappa)} = \mathbb{T}^{(\kappa-1)}$
  end
  $\kappa + +$;
end
return $\boldsymbol{v}_{adv}^{(\kappa)}$;

and regularization term), relax conditioning, and design the approximation algorithm.

C. SparseQuery

Apparently, $\boldsymbol{v}_{adv}$ relying only on $\boldsymbol{\mathcal{I}}$, $\boldsymbol{\mathcal{F}}$, and $\boldsymbol{\theta}$ ($\boldsymbol{v}_{adv} = \boldsymbol{v} + \boldsymbol{\mathcal{I}} \odot \boldsymbol{\mathcal{F}} \odot \boldsymbol{\theta}$) is guaranteed be good on $\mathcal{S}(\cdot)$ but can be less effective on $\mathcal{R}(\cdot)$. So we design a query-based scheme, i.e., SparseQuery, to further improve $\boldsymbol{v}_{adv}$ while following the same intuition, i.e., to make sure that perturbations on pixels and frames are sparse. We illustrate SparseQuery in Figure 2 and Algorithm 2 as well. The input are $\boldsymbol{\mathcal{I}}$, $\boldsymbol{\mathcal{F}}$, and $\boldsymbol{\theta}$ and the output is $\boldsymbol{v}_{adv}$. The high-level idea in query-based attacks is to iterate and move $\boldsymbol{v}_{adv}$ along the direction to decrease the objective function. To implement such an idea, we first formulate the objective function as follows so that when it decreases, $\mathcal{R}^m(\boldsymbol{v}_{adv})$ gets closer to $\mathcal{R}^m(\boldsymbol{v}_t)$.

$$\begin{aligned}\mathbb{T}(\boldsymbol{v}_{adv}, \boldsymbol{v}, \boldsymbol{v}_t) =& \mathbb{H}(\mathcal{R}^m(\boldsymbol{v}_{adv}), \mathcal{R}^m(\boldsymbol{v})) \\ &- \mathbb{H}(\mathcal{R}^m(\boldsymbol{v}_{adv}), \mathcal{R}^m(\boldsymbol{v}_t)) + \eta\,,\end{aligned} \tag{2}$$

where η is a margin constant and $\mathbb{H}(\mathcal{R}^m(\boldsymbol{v}), \mathcal{R}^m(\boldsymbol{v}'))$ is a probability-based similarity function derived from the NDCG-based function [10] to capture the co-occurrence probability that a returned video shows up in both $\mathcal{R}^m(\boldsymbol{v})$ and $\mathcal{R}^m(\boldsymbol{v}')$. More details about the loss function can be found in the supplementary material at [45]. Next, in each iteration, we modulate the corresponding perturbation term by $\boldsymbol{\mathcal{I}} \odot \boldsymbol{\mathcal{F}} \odot \boldsymbol{\theta}$ in order to *make sure perturbations at this stage maintain sparse*. Thus for the κ-th iteration, we update $\boldsymbol{v}_{adv}$ as follows:

$$\boldsymbol{v}_{adv}^{(\kappa)} = \begin{cases} \texttt{CLIP}(\boldsymbol{v}_{adv}^{(\kappa-1)} + \epsilon \boldsymbol{q}), & \mathbb{T}^{(\kappa)}(\cdot,\cdot,\cdot) \leq \mathbb{T}^{(\kappa-1)}(\cdot,\cdot,\cdot)\,; \\ \texttt{CLIP}(\boldsymbol{v}_{adv}^{(\kappa-1)} - \epsilon \boldsymbol{q}), & \mathbb{T}^{(\kappa)}(\cdot,\cdot,\cdot) \geq \mathbb{T}^{(\kappa-1)}(\cdot,\cdot,\cdot)\,, \end{cases} \tag{3}$$

where $\mathbb{T}^{(\kappa)}(\cdot,\cdot,\cdot) = \mathbb{T}^{(\kappa)}(\boldsymbol{v}_{adv}, \boldsymbol{v}, \boldsymbol{v}_t)$, $\mathbb{T}^{(\kappa-1)}(\cdot,\cdot,\cdot) = \mathbb{T}^{(\kappa-1)}(\boldsymbol{v}_{adv}, \boldsymbol{v}, \boldsymbol{v}_t)$, and $\boldsymbol{q}^{(\kappa)} \in \mathbb{R}^{N,W,H,C}$ is a random matrix modulated by $\boldsymbol{\mathcal{I}} \odot \boldsymbol{\mathcal{F}} \odot \boldsymbol{\theta}$, i.e.,

$$\boldsymbol{q}_{i,w,h,c}^{(\kappa)} = \begin{cases} \boldsymbol{q}_{i,w,h,c}^{(\kappa)}, & \boldsymbol{\mathcal{I}}_{i,w,h,c} \times \boldsymbol{\mathcal{F}}_{i,w,h,c} \times \boldsymbol{\theta}_{i,w,h,c} \neq 0\,; \\ 0, & \boldsymbol{\mathcal{I}}_{i,w,h,c} \times \boldsymbol{\mathcal{F}}_{i,w,h,c} \times \boldsymbol{\theta}_{i,w,h,c} = 0\,, \end{cases} \tag{4}$$

and ϵ is a step size. ϵ is derived from $\| \pm \epsilon \boldsymbol{q}^{(\kappa)}\|_\infty \leq \tau$ such that $\boldsymbol{v}_{adv}^{(\kappa)}$ complies with $\|\boldsymbol{\theta}\|_\infty \leq \tau$ in Equation (1). The above process stops until it converges or the number of iterations exceeds the preset maximum (i.e., $\kappa \geq \texttt{iter\_numQ}$) and SparseQuery outputs $\boldsymbol{v}_{adv}$. More details about $\mathbb{H}(\cdot,\cdot)$ and $\boldsymbol{q}$ can be found in the supplementary material at [45].

Summary. Here we briefly summarize our DUO attack. Given $\boldsymbol{v}$ and $\boldsymbol{v}_t$, DUO first launches SparseTransfer to obtain $\boldsymbol{\mathcal{I}}$, $\boldsymbol{\mathcal{F}}$, and $\boldsymbol{\theta}$. Following that, DUO launches SparseQuery to obtain $\boldsymbol{v}_{adv}$ based on $\boldsymbol{\mathcal{I}}$, $\boldsymbol{\mathcal{F}}$, and $\boldsymbol{\theta}$. To avoid being trapped at a local minimum [48], we loop SparseTransfer and SparseQuery together by using $\{\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{F}}, \boldsymbol{\theta}, \boldsymbol{v}_{adv}\}$ to initialize $\{\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{F}}, \boldsymbol{\theta}, \boldsymbol{v}\}$ for the next iteration until the process converges or the number of iteration exceeds a preset threshold, i.e., `iter_numH`. `iter_numH` is a small number in our implementation (less than 4).

V. Performance Evaluation

A. Datasets and Metrics

In this paper, we focus on UCF101 [19] and HMDB51 [20] for benchmarking our attack. For fair comparison against other attacks, we follow [1] to uniformly sample a 16-frame snippet from each video. Table I lists the details of our datasets. When evaluating attack performance, we randomly choose ten pairs of two videos from the training dataset: one as the original video and the other as the target video (i.e., $\boldsymbol{v}$ and $\boldsymbol{v}_t$). The experimental results in what follows are the average from all experiments on one of the ten pairs of $\boldsymbol{v}$ and $\boldsymbol{v}_t$.

We adopt four performance metrics to evaluate the performance of DUO. The first metric is mean average precision (denoted by `mAP`) and it measures the retrieval performance

TABLE I
DATASET DETAILS

Dataset	# of training videos	# of testing videos	# of categories
UCF101	9,324	3,996	101
HMDB51	4,900	2,100	51

of the victim system. Specifically, the mAP can be calculated as follow:

$$\texttt{mAP} = \frac{1}{N} \times \sum_{i=1}^{N} \frac{ctop(i)}{i} \ ,$$

where N and $ctop(i)$ indicate the total size of the correct retrieval list, and the number of correct items in top-i retrieval list, respectively. The higher mAP, the better retrieval performance. The average precision AP@m, as the second metric, measures the average precision between $\mathcal{R}^m(\boldsymbol{v}_{adv})$ and $\mathcal{R}^m(\boldsymbol{v}_t)$, i.e., the returned videos of $\boldsymbol{v}_{adv}$ and $\boldsymbol{v}_t$. The precision of the i-th retrieved video is defined as:

$$\texttt{prec}_i = \frac{|\mathcal{R}_i^m(\boldsymbol{v}_{adv}) \cap \mathcal{R}_i^m(\boldsymbol{v}_t)|}{i} \ ,$$

where $\mathcal{R}_i^m(\boldsymbol{v}_{adv})$ denotes the top-i items in the retrieval list. If $\texttt{prec}_i$ is equal to 1, it show that the i-th item of the retrieval list of $\boldsymbol{v}_{adv}$ and $\boldsymbol{v}$ is the same, otherwise 0. Hence, AP@m can be calculated as

$$\texttt{AP@}m = \sum_i \texttt{prec}_i / m \ .$$

Obviously, the higher AP@m, the higher similarity between $\mathcal{R}^m(\boldsymbol{v}_{adv})$ and $\mathcal{R}^m(\boldsymbol{v}_t)$. In addition, we use a sparsity metric to evaluate the sparsity of a synthesized AE, defined as $\texttt{Spa} = \sum_{i=1}^{N} \|\boldsymbol{\phi}_i\|_0$. $\boldsymbol{\phi}_i$ denotes the number of perturbed pixels in the i-th frame thus $\|\boldsymbol{\phi}_i\|_0$ denotes the number of non-zero elements in $\boldsymbol{\phi}_i$. N is the number of frames. Note that the smaller Spa, the better sparsity, thus the stealthier attack. Lastly, we adopt the perceptibility score [49] PScore to measure the imperceptibility of an AE defined as

$$\texttt{PScore} = \frac{1}{N \times B \times C} \sum_{i=1}^{N \times B \times C} |\boldsymbol{\phi}_i|,$$

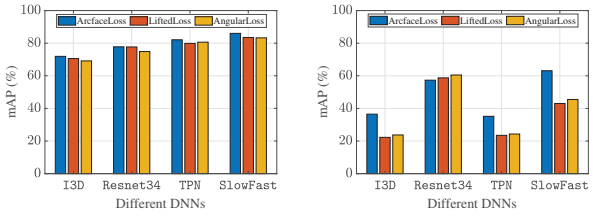
where B denotes the number of pixels in one frame. The smaller PScore, the less perturbations, thus the stealthier attack.

B. Implementation and Experimental Settings

Implementation. We implement DUO in Pytorch and run all the experiments on a server with a Intel(R) Xeon(R) Silver 4210 CPU@2.20GHz and eight NVIDIA Geforce RTX 3080. Our code can also be found in [45].

Victim model. We explore four DNN models for extracting spatial and temporal features from videos: I3D [21], TPN [22], SlowFast [23], and Resnet34 [24]. The features are flattened as a vector with a size of 768×1. We experiment three loss functions including ArcfaceLoss [50], LiftedLoss [51], and AngularLoss [52] and ℓ_2-norm for regularization in all cases.

Surrogate model. We experiment two DNN models (i.e., C3D [43] and Resnet18 [24]), four sizes of the surrogate dataset, and four output feature sizes (i.e., [256, 512, 768, 1,024]). On UCF101, the sizes of the surrogate dataset are [165, 1,111, 3,616, 8,421] while on HMDB51 [165, 1,111, 1,885, 2,995]. We use a 7:3 split ratio to divide them into a training dataset and a testing dataset.



(a) UCF101 (b) HMDB51

Fig. 3. mAPs on different (victim) video retrieval systems

System parameters. λ in Eq. (1) is set as e^{-5}. The maximum number of iterations in SparseQuery, i.e., iter_numQ, is 1,000. The step size for gradient descent there is initialized as 0.1 and decays for every 50 steps with a rate of 0.9.

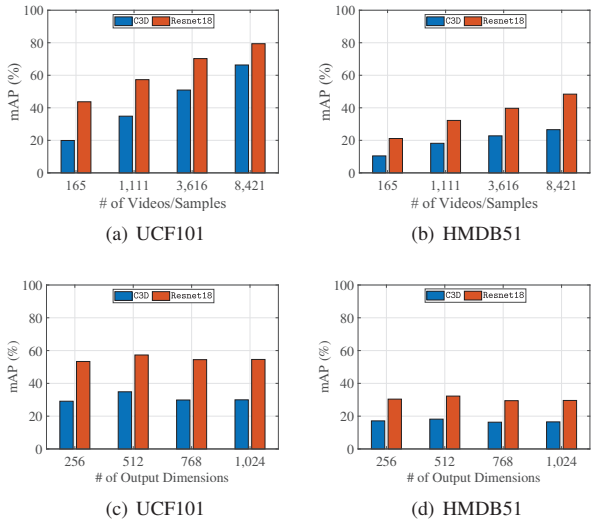
Baselines. We compare DUO to the following baselines. Note that DUO-C3D and DUO-Res18 correspond to DUO with C3D and Resnet18 as the surrogate model, respectively.

- Vanilla attack. Our Vanilla attack is very straightforward. It first randomly selects pixels for each frame given a fixed Spa. Then it uses a query-based attack [53] to generate $\boldsymbol{v}_{adv}$.
- TIMI attack [25]. As one of the popular transferable adversarial attacks, it combines the momentum iterative (MI) and translation-invariant (TI) methods to achieve a high success rate. Similarly, we also denote the transferable attack by TIMI-C3D and TIMI-Res18 corresponding to TIMI with C3D and Resnet18 as the surrogate model, respectively.
- HEU-Nes attack [16] and HEU-Sim attack that replaces the nature-estimated strategy in HEU-Nes with the random-selection strategy from Vanilla.

C. Experimental Results

mAPs of victim video retrieval systems. Figure 3 shows mAPs of different victim models on UCF101 and HMDB51. Note that UCF101 has a larger size than HMDB51. Our two observations are as follows. 1) On UCF101, the loss function has little impact on mAP while SlowFast leads to the highest mAP among the four feature extractors. 2) The situation on HMDB51 is quite different. In general, ArcfaceLoss leads to higher mAP among the three loss functions while Resnet34 tends to do so as well among the four feature extractors. Yet Slowfast plus ArcfaceLoss happens to achieve the highest mAP. To sum up, the best combination of feature extractor and loss function could still depend on the exact dataset.

mAPs on different setups for the surrogate model. We anticipate that mAP is under the impact of the size of the surrogate dataset and the size of the output feature vector. Our goal here is to select the surrogate model with the best mAP in hope of improving attack performance of DUO. We have



(a) UCF101 (b) HMDB51

(c) UCF101 (d) HMDB51

Fig. 4. mAPs of the surrogate model with different # of samples and output feature sizes.

the following three conclusions from the results in Figure 4. 1) As expected, the larger surrogate dataset, the higher mAP. The mAP on UCF101 increases from 19.91% to 50.92% as the size of the surrogate dataset increases from 165 to 3,616. 2) The output feature size has little impact on mAP.

Performance of different AE attacks. We conclude that a targeted AE attack succeeds if AP@m from $\mathcal{R}^m(\boldsymbol{v})$ and $\mathcal{R}^m(\boldsymbol{v}_t)$ (which we use to refer to the scenario "w/o attack", i.e., "without attack") is lower than that from $\mathcal{R}^m(\boldsymbol{v}_{adv})$ and $\mathcal{R}^m(\boldsymbol{v}_t)$. That is, the retrieved videos are more similar to $\boldsymbol{v}_t$ rather than $\boldsymbol{v}$. We summarize the results in Table II and draw the following conclusions. 1) All targeted AE attacks in our experiments succeed because the retrieved videos by $\boldsymbol{v}_{adv}$ in these attacks are more similar to $\boldsymbol{v}_t$ rather than $\boldsymbol{v}$. As a proof, if we look at the result when using I3D as the feature extractor on UCF101, AP@m from $\mathcal{R}^m(\boldsymbol{v})$ and $\mathcal{R}^m(\boldsymbol{v}_t)$ is only 48.67% while AP@ms from from $\mathcal{R}^m(\boldsymbol{v}_{adv})$ and $\mathcal{R}^m(\boldsymbol{v}_t)$ under all attacks are higher, ranging from 49.04% to 56.40%. 2) DUO tends to outperform other attacks consistently. Particularly, when the victim model is TPN on UCF101 and we fix the number of perturbed frames, DUO-C3D achieves a higher AP@m (79.29%) than Vanilla (72.54%). This indicates that DUO retrieves more similar videos, confirming that our dual frame-pixel search in SparseTransfer finds better perturbations than random selection in Vanilla. 3) PScores are approximately proportional to Spas, which is as expected by referring to Equation (3).

Impact of the size of surrogate dataset and the loss function of the victim model. Here we want to answer two questions. **Q1: does the size of the surrogate dataset impact the effectiveness (i.e., AP@m) and the stealthiness (i.e., Spa) of DUO?** If the answer is 'No', it is good news as the attacker no longer need to obtain many training samples for the surrogate model. The results in Table III tend to imply 'No' because for DUO-C3D on UCF101, when the surrogate dataset size increases significantly (from 165 to 3,616), Spas only decreases slightly (from 2,903 to 2,832, thus little improvement) while AP@m decreases (from 58.08% to 56.28% meaning a larger size could instead harm AP@m). *Therefore, we believe that DUO works even with only a handful of samples for the surrogate model and fix the size to be 1,111 and the output feature dimension 512 for the remaining part.* **Q2: which loss function is robust against DUO when used to train a victim model?** The one(s) in the answer should provide lower AP@m as well as higher Spa. In general, Table IV suggests that ArcFaceLoss is the most robust one (due to low AP@m and higher Spa) while LiftedLoss and AngularLoss tend to results at higher AP@m with smaller Spa.

Impact of k and n on attack performance. Equation (1) conveys two suggestions for us. The first is that a larger k and/or n are/is very likely to result at higher AP@m and higher Spa simultaneously simply because the larger k and/or n allow(s) more perturbations. The second is that AP@m is likely to converge with the increase of k and/or n simply because DUO cannot find better perturbations even provided the resources. We are able to confirm the above two through the results in Table V. As proof, AP@m of DUO-C3D on UCF101 increases from 52.81% to 56.40% when k increases from $20K$ to $40K$ and converges after that. We can find proof to support conclusions on Spa in Table V as well.

Impact of the number of queries in SparseQuery. Here we want to confirm whether the queries in SparseQuery of DUO indeed rectify adversarial perturbations thus synthesizing better $\boldsymbol{v}_{adv}$ or not. We anticipate that if $\mathbb{T}$ in Equation (2) is able to drop with the increase of queries, the synthesized $\boldsymbol{v}_{adv}$ should result at higher AP@m. First, we show in Figure 5 the relationship between the number of queries and $\mathbb{T}$ from Equation (2) in SparseQuery. The results confirm that SparseQuery helps rectify the perturbations for $\boldsymbol{v}_{adv}$ because the retrieved videos become more similar to $\boldsymbol{v}_t$. Second, although a lower $\mathbb{T}$ does not guarantee a higher AP@m across different attacks, we do see that the lower $\mathbb{T}$s of our DUO-C3D and DUO-Res18 attack comparing to Vanilla in Figure 5(a) do result at higher AP@ms in Table II (AP@m of Vanilla, DUO-C3D and DUO-Res18 when the victim model is TPN on UCF101: 59.78%, 68.15% and 67.03%) under the same experimental settings. Therefore, we conjecture a lower $\mathbb{T}$ in SparseQuery would help achieve a higher AP@m.

Impact of τ in SparseTransfer on attack performance. We experiment four τ selection from $[15, 30, 40, 50]$ and summarize the resulting AP@m and Spa in Table VII. Note that τ from Equation (1) is the perturbation budget for a single pixel. 1) As expected, the results show that AP@m increases significantly as τ increases simply because a larger perturbation budget helps find larger perturbations and thus synthesizes more effective $\boldsymbol{v}_{adv}$. We can see that AP@m increases from 50.57% to 55.25% as τ increases from 15 to 40. 2) τ does not affect Spas significantly. This means that very likely the capability of adding perturbations with a higher magnitude does not impact the number of total perturbed pixels much.

TABLE II
ATTACK PERFORMANCE OF DIFFERENT AE ATTACKS

Target Model	TPN			SlowFast			I3D			Resnet34		
Dataset	UCF101											
	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore
w/o attack	67.84	–	–	40.06	–	–	48.67	–	–	52.12	–	–
TIMI-C3D ($n = 16$)	68.34	602,100	10.00	40.16	588,726	9.55	49.04	601,371	9.87	52.40	597,127	9.63
TIMI-Res ($n = 16$)	68.64	602,100	10.00	40.49	588,726	9.55	49.20	601,371	9.87	53.19	597,127	9.63
HEU-Nes ($n = 4$)	69.85	2,880	0.14	40.92	2,076	0.10	51.19	3,000	0.15	64.19	3,456	0.17
HEU-Sim ($n = 4$)	74.36	2,136	0.11	41.14	417	0.02	53.48	1,920	0.09	63.61	1,900	0.09
Vanilla ($n = 4$)	72.54	2,885	0.14	41.26	1,549	0.08	52.84	2,806	0.14	61.87	2,645	0.13
DUO-C3D ($n = 4$)	**79.29**	2,884	0.14	**48.34**	2,077	0.10	**56.40**	2,800	0.14	**67.40**	3,466	0.17
DUO-Res18 ($n = 4$)	**76.07**	2,138	0.11	**42.58**	873	0.04	**55.73**	2,404	0.12	**68.50**	2,797	0.14
Dataset	HMDB51											
	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore
w/o attack	28.90	–	–	38.47	–	–	24.72	–	–	36.64	–	–
TIMI-C3D ($n = 16$)	29.82	602,111	10.00	40.07	602,112	10.00	25.60	602,014	9.92	37.07	602,112	10.00
TIMI-Res ($n = 16$)	29.84	602,111	10.00	40.45	602,112	10.00	25.64	602,014	9.92	37.61	602,112	10.00
HEU-Nes ($n = 4$)	30.02	1,404	0.07	48.54	1,512	0.08	26.25	1,284	0.06	39.37	1,056	0.05
HEU-Sim ($n = 4$)	30.21	180	0.01	42.11	304	0.02	29.31	377	0.02	40.75	326	0.02
Vanilla ($n = 4$)	**33.89**	789	0.04	52.29	1,152	0.06	30.74	673	0.03	43.75	796	0.04
DUO-C3D ($n = 4$)	**44.22**	1,404	0.07	**58.83**	1,515	0.08	**34.61**	903	0.04	**53.41**	1,047	0.05
DUO-Res18 ($n = 4$)	32.70	662	0.03	**62.59**	1,153	0.06	**34.73**	656	0.03	**44.90**	716	0.04

TABLE III
ATTACK PERFORMANCE OF OUR ATTACKS WITH DIFFERENT SIZES OF THE SURROGATE DATASET

# of Videos/Samples	165			1,111			3,616			8,421		
Dataset	UCF101											
	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore
DUO-C3D	58.08	2,903	0.14	56.40	2,800	0.14	56.28	2,832	0.14	55.19	2,184	0.11
DUO-Res18	56.88	2,652	0.13	55.73	2,404	0.12	57.31	2,218	0.11	55.15	2,573	0.13
Dataset	HMDB51											
	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore
DUO-C3D	32.19	651	0.03	34.61	903	0.04	32.70	661	0.03	31.74	588	0.03
DUO-Res18	34.47	676	0.03	34.73	656	0.03	33.57	678	0.03	33.90	679	0.03

TABLE IV
ATTACK PERFORMANCE OF OUR ATTACKS ON VICTIM MODELS USING DIFFERENT LOSS FUNCTIONS

Loss Function	ArcFaceLoss			LiftedLoss			AngularLoss		
Dataset	UCF101								
	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore
DUO-C3D	56.40	2,800	0.14	67.87	1,620	0.08	63.88	2,536	0.12
DUO-Res18	55.73	2,404	0.12	66.46	1,103	0.05	67.14	2,128	0.10
Dataset	HMDB51								
	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore
DUO-C3D	34.61	903	0.04	51.67	537	0.03	67.15	1,034	0.05
DUO-Res18	34.73	656	0.03	51.97	565	0.03	66.96	1,052	0.05

We plan to look deeper into this in the future.

Impact of `iter_numH` on attack performance. We change `iter_numH` in DUO from 1 to 4 and list the results in Table VIII. The results show that 1) `AP@m` of DUO increases along with `iter_numH` which is as expected considering that more iterations should result at better $\boldsymbol{v}_{adv}$. For example, the `AP@m` increases from 53.04% to 56.94% as `iter_numH` increases from 1 to 3. 2) `Spa` and `PScore` also increase along with `iter_numH`. `Spa` increases from 1,712 to 2,942 when `iter_numH` from 1 to 3. Though such results are somehow expected since usually a higher `AP@m` is accompanied by a higher `Spa`/`PScore`, we plan to look deeper into this to see if we can maintain lower `Spa`/`PScore` even when `iter_numH` increases so that the effectiveness as well as the stealthiness of $\boldsymbol{v}_{adv}$ is enhanced.

Impact of AE Transferability. To evaluate the transferability of AE attacks, we run AE attacks to generate adversarial examples on the same surrogate models and evaluate the attack performance in various target models. Concretely, we adopt the l_2- and l_∞-norm constraints for Eq. (1), run SparseTransfer and SparseQuery on the same surrogate model, and finally output the `AP@m` from the target model. We show part of the results on the UCF101 dataset in Table IX given space limit. From Table IX, we can conclude that 1) the AEs we generated on the target model have a higher transferability on the surrogate model. It is reasonable as SparseQuery is expected to further fine-tune the results of SparseTransfer. 2) The AEs generated by our pipeline have better `AP@m`s and `Spa`s than `TIMI` on the more popular SlowFast model. For example, the `AP@m` under the `DUO-C3D` with the l_2 constraint is 44.94% while that under the `TIMI-C3D` is only 40.16%.

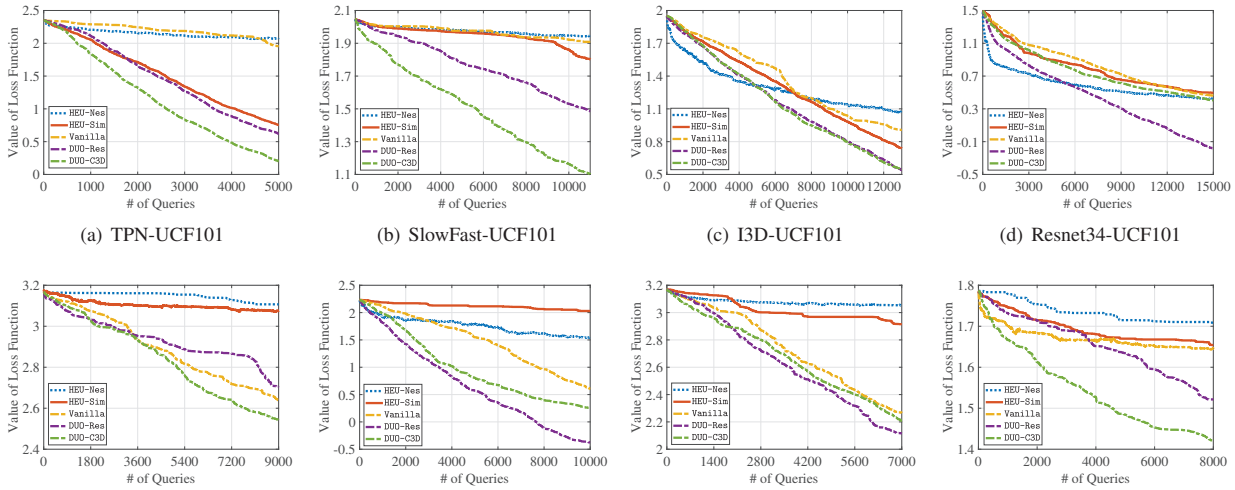
Summary. We summarize the key takeaway pieces from

TABLE V
ATTACK PERFORMANCE UNDER OUR DUO ATTACKS WITH $n = 4$ AND $k = \{20K, 30K, 40K, 50K\}$

k	$20K$			$30K$			$40K$			$50K$		
Dataset	UCF101											
	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore
DUO-C3D	52.81	2,508	0.12	54.97	2,683	0.13	56.40	2,800	0.14	56.93	2,844	0.13
DUO-Res18	51.31	1,956	0.10	52.98	2,142	0.11	55.73	2,404	0.12	58.81	3,364	0.16
Dataset	HMDB51											
	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore
DUO-C3D	30.87	725	0.04	32.95	897	0.04	34.61	903	0.04	35.79	1,020	0.05
DUO-Res18	32.35	438	0.02	32.81	556	0.03	34.73	656	0.03	36.45	795	0.04

TABLE VI
ATTACK PERFORMANCE UNDER OUR DUO ATTACKS WITH $k = 40K$ AND $n = \{2, 3, 4, 5\}$

n	2			3			4			5		
Dataset	UCF101											
	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore
DUO-C3D	53.35	1,832	0.09	54.18	2,620	0.13	56.40	2,800	0.14	56.45	2,955	0.14
DUO-Res18	52.63	1,487	0.07	54.33	2,167	0.11	55.73	2,404	0.12	54.94	2,555	0.13
Dataset	HMDB51											
	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore
DUO-C3D	26.20	380	0.02	33.80	630	0.03	34.61	903	0.04	32.42	970	0.05
DUO-Res18	27.00	438	0.02	34.56	663	0.03	34.73	656	0.03	33.41	968	0.05



(a) TPN-UCF101 (b) SlowFast-UCF101 (c) I3D-UCF101 (d) Resnet34-UCF101

(e) TPN-HMDB51 (f) SlowFast-HMDB51 (g) I3D-HMDB51 (h) Resnet34-HMDB51

Fig. 5. Impact of the number of queries on the loss function $\mathbb{T}$ in SparseQuery. A decreasing $\mathbb{T}$ implies a rectified $\boldsymbol{v}_{adv}$.

our evaluations. For a DUO attacker,

- a handful of samples and using `C3D` as the backbone could already make a good surrogate model.
- k and τ are better choices to optimize because they tend to improve `AP@m` without sacrificing `Spa`/`PScore`, particularly when compared to n and `iter_numH` (thus they require more careful selection).

For a potential victim model, try to use ArcFaceLoss rather than LiftedLoss and AngularLoss for model training if there are options and model robustness is preferred.

D. Attack Robustness against Popular Defenses

Existing defenses. In this part, we briefly explore the robustness of different attacks under two popular defenses: `feature squeezing` [26] and `Noise2Self` [27]. The high-level idea of `feature squeezing` is that AEs are not robust against linear transformation such as reducing the size of the feature vector while that of `Noise2Self` is simply to treat AEs as noise and apply de-noising to reduce the impact of AEs. We summarize only the robustness of attacks using `I3D` as the victim model in Table X due to page limit while we have similar observations in other settings as well. The results show that DUO is more resilient towards `feature squeezing` and `Noise2Self` when compared to `Vanilla`, confirming the better stealthiness of DUO. DUO also exhibits similar robustness when compared to `TIMI` and `HEU` while it achieves the lowest detection rate (8.25%) on UCF101. Therefore, we believe that perturbation sparsification

TABLE VII
ATTACK PERFORMANCE OF OUR DUO ATTACKS WITH DIFFERENT PERTURBATION BUDGET τ IN EQUATION (1)

τ	15			30			40			50		
Dataset	UCF101											
	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore
DUO-C3D	51.62	2,249	0.06	56.40	2,800	0.14	57.33	2,634	0.17	57.88	2,557	0.20
DUO-Res18	50.57	1,761	0.04	55.73	2,404	0.12	55.25	2,282	0.15	58.92	2,546	0.20
Dataset	HMDB51											
	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore
DUO-C3D	28.26	362	0.01	34.61	903	0.04	35.65	994	0.07	36.47	1,001	0.08
DUO-Res18	28.36	361	0.01	34.73	656	0.03	35.72	979	0.07	36.28	970	0.08

TABLE VIII
ATTACK PERFORMANCE OF OUR DUO ATTACKS WITH DIFFERENT ITER_NUMHS.

iter_numH	1			2			3			4		
Dataset	UCF101											
	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore
DUO-C3D	53.04	1,712	0.08	56.40	2,800	0.14	56.94	2,942	0.14	56.12	3,007	0.15
DUO-Res18	52.68	1,590	0.08	55.73	2,404	0.12	56.91	2,911	0.15	58.32	2,966	0.15
Dataset	HMDB51											
	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore
DUO-C3D	32.63	549	0.03	34.61	903	0.04	34.65	979	0.05	34.69	988	0.05
DUO-Res18	31.82	419	0.02	32.39	571	0.03	33.85	654	0.03	34.04	664	0.03

TABLE IX
ATTACK PERFORMANCE OF OUR SPARSETRANSFER WITH DIFFERENT TARGET MODEL UNDER UCF101 DATASET.

Target Model	TPN			SlowFast			I3D			Resnet34		
Dataset	UCF101											
	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore	AP@m	Spa	PScore
TIMI-C3D ($n = 16$)	68.34	602,100	10.00	40.16	588,726	9.55	49.04	601,371	9.87	52.40	597,127	9.63
TIMI-Res ($n = 16$)	68.64	602,100	10.00	40.49	588,726	9.55	49.20	601,371	9.87	53.19	597,127	9.63
DUO-C3D (l_2)	69.45	2,132	0.11	44.94	2,135	0.11	48.76	2,130	0.11	56.23	2,311	0.12
DUO-Res18 (l_2)	68.71	2,490	0.12	43.51	2,436	0.12	49.01	2,116	0.11	55.61	2,100	0.10
DUO-C3D (l_∞)	68.31	2,105	0.10	43.43	2,274	0.11	48.30	2,390	0.12	54.49	2,112	0.11
DUO-Res18 (l_∞)	68.33	2,450	0.12	43.42	2,364	0.12	48.31	2,195	0.11	54.08	2,105	0.10

TABLE X
ATTACK DETECTION RATE (%) OF TWO DEFENSES [26], [27]

Attacks	feature squeezing		Noise2Self	
	UCF101	HMDB51	UCF101	HMDB51
Vanilla	82.68	66.56	25.01	60.89
TIMI-C3D	24.31	26.86	3.94	23.53
TIMI-Res18	28.56	26.44	4.84	26.60
HEU-Nes	21.67	22.53	21.96	48.36
HEU-Simba	8.74	22.39	23.29	24.16
DUO-C3D	8.25	24.07	26.22	40.56
DUO-Res18	17.96	43.47	21.85	43.06

helps improve the stealthiness of DUO.

A potential defense against DUO. Provided the observations in our evaluations, we believe that the effectiveness and stealthiness of an arbitrary attack including DUO inevitably show (some) dependence on the backbone and loss function of the victim model. Hence ensemble models built from multiple backbones would be more robust against most AE attacks, DUO included. We plan to explore this direction as well.

VI. CONCLUSIONS

In this paper, we explore stealthy AE attacks on popular video retrieval systems focusing on AE stealthiness and effectiveness. Our DUO attack is a targeted attack under black-box setting, thus rendering itself practical in real world scenarios. DUO roots in that perturbation sparsification helps improve AE stealthiness without sacrificing much effectiveness. We implemented DUO as a novel sequential pipeline and evaluated its performance extensively, confirming its better performance over three existing attacks in terms of precision, stealthiness, and robustness against two popular defenses.

VII. ACKNOWLEDGMENTS

This work was partially supported by National Natural Science Foundation of China through grant 61902433, the Key Research and Development Program of Xinjiang Autonomous Region under Grant 2021B01002, the Major Special Project of Changsha Science and Technology Plan under Grant KH2103016, and the High Performance Computing Center of Central South University.

REFERENCES

[1] L. Yuan, T. Wang, X. Zhang, F. Tay, Z. Jie, W. Liu, and J. Feng, "Central similarity quantization for efficient image and video retrieval," in CVPR'20, Virtual, Jun. 2020.

[2] G. Wu, J. Han, Y. Guo, L. Liu, G. Ding, Q. Ni, and L. Shao, "Unsupervised deep video hashing via balanced code for large-scale video retrieval," IEEE Transactions on Image Processing, vol. 28, no. 4, pp. 1993–2007, 2018.

[3] K. Liao, H. Lei, Y. Zheng, G. Lin, C. Cao, M. Zhang, and J. Ding, "Ir feature embedded bof indexing method for near-duplicate video retrieval," IEEE Transactions on Circuits and Systems for Video Technology, vol. 29, no. 12, pp. 3743–3753, 2018.

[4] X. Nie, W. Jing, C. Cui, C. J. Zhang, L. Zhu, and Y. Yin, "Joint multi-view hashing for large-scale near-duplicate video retrieval," IEEE Transactions on Knowledge and Data Engineering, vol. 32, no. 10, pp. 1951 – 1965, Oct. 2019.

[5] C. Jing, Z. Dong, M. Pei, and Y. Jia, "Heterogeneous hashing network for face retrieval across image and video domains," IEEE Transactions on Multimedia, vol. 21, no. 3, pp. 782 – 794, Jul. 2018.

[6] S. Qiao, R. Wang, S. Shan, and X. Chen, "Deep video code for efficient face video retrieval," Pattern Recognition, vol. 113, no. 3, p. 107754, May. 2021.

[7] Douceur, John R, "The sybil attack," in IPTPS'02, Cambridge, MA, Jun. 2002.

[8] S. Li, A. Neupane, S. Paul, C. Song, S. Krishnamurthy, A. Roy-Chowdhury, and A. Swami, "Stealthy adversarial perturbations against real-time video classification systems," in NDSS'19, San Diego, CA, Feb. 2019.

[9] Y. Lin, H. Zhao, Y. Tu, S. Mao, and Z. Dou, "Threats of adversarial attacks in dnn-based modulation recognition," in INFOCOM'20, Toronto, Canada, Jul. 2020.

[10] X. Li, J. Li, Y. Chen, S. Ye, Y. He, S. Wang, H. Su, and H. Xue, "Qair: Practical query-efficient black-box attacks for image retrieval," in CVPR'21, Nashville, TN, June 2021.

[11] M. Chen, J. Lu, Y. Wang, J. Qin, and W. Wang, "Dair: A query-efficient decision-based attack on image retrieval systems," in SIGIR'21, Montreal, Canada, Jul. 2021.

[12] N. Wang, Y. Chen, Y. Hu, W. Lou, and Y. T. Hou, "Manda: On adversarial example detection for network intrusion detection system," in INFOCOM'22, British Columbia, Canada, May 2022.

[13] Chen. S, Carlini. N, Wagner .D.,"Stateful detection of black-box adversarial attacks," in Asia CCS'20, Taipei, Taiwan, Jun. 2020.

[14] Juuti. Mika and Szyller. Sebastian and Marchal. Samuel and Asokan N, "PRADA: protecting against DNN model stealing attacks," in EuroS&P'19, Stockholm, Sweden, Aug. 2019.

[15] Y. Xiao and C. Wang, "You see what i want you to see: Exploring targeted black-box transferability attack for hash-based image retrieval systems," in CVPR'21, Nashville, TN, Jun. 2021.

[16] Z. Wei, J. Chen, X. Wei, L. Jiang, T. Chua, F. Zhou, and Y. Jiang, "Heuristic Black-Box Adversarial Attacks on Video Recognition Models," in AAAI'20, New York, NY, Feb. 2020.

[17] S. Li, A. Aich, S. Zhu, S. Asif, C. Song, A. Roy-Chowdhury, and S. Krishnamurthy, "Adversarial Attacks on Black Box Video Classifiers: Leveraging the Power of Geometric Transformations," in NeurIPS'21, Dec. 2021.

[18] B. Wu and B. Ghanem, "ℓ_p-box ADMM: A Versatile Framework for Integer Programming," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 7, pp. 1695–1708, Jan. 2018.

[19] K. Soomro, R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," in ICCV'13, Sydney, Australia, Dec. 2013.

[20] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: a large video database for human motion recognition," in ICCV'11, Barcelona, Spain, Nov. 2011.

[21] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in CVPR'17, Hawaii, HI, Jul. 2017.

[22] C. Yang, Y. Xu, J. Shi, B. Dai, and B. Zhou, "Temporal pyramid network for action recognition," in CVPR'20, Virtual, Jun. 2020.

[23] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in ICCV'19, Seoul, Korea, Oct. 2019.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in CVPR'16, Las Vegas, NV, Aug. 2016.

[25] Y. Dong, T. Pang, H. Su, and J. Zhu, "Evading defenses to transferable adversarial examples by translation-invariant attacks," in CVPR'19, Long Beach, CA, Jun. 2019.

[26] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in NDSS'18, San Diego, CA, Feb. 2018.

[27] J. Batson and L. Royer, "Noise2self: Blind denoising by self-supervision," in ICML'19, Long Beach, CA, Jun. 2019.

[28] E. Yang, T. Liu, C. Deng, and D. Tao, "Adversarial examples for hamming space search," IEEE Transactions on Cybernetics, vol. 50, no. 4, pp. 1473–1484, Apr. 2018.

[29] G. Tolias, F. Radenovic, and O. Chum, "Targeted mismatch adversarial attack: Query with a flower to retrieve the tower," in ICCV'19, Seoul, South Korea, Oct. 2019.

[30] J. Bai, B. Chen, Y. Li, D. Wu, W. Guo, S. Xia, and E. Yang, "Targeted attack for deep hashing based retrieval," in ECCV'20, Glasgow, UK, Aug. 2020.

[31] X. Wang, Z. Zhang, B. Wu, F. Shen, and G. Lu, "Prototype-supervised adversarial network for targeted attack of deep hashing," in CVPR'21, Nashville, TN, Jun. 2021.

[32] J. Bai, B. Chen, D. Wu, C. Zhang, and S. Xia, "Universal adversarial head: Practical protection against video data leakage," in ICML'21, Virtual Only, Jul. 2021.

[33] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in Euro S&P'16, Saarbrücken, Germany, Mar. 2016.

[34] D. Karmon, D. Zoran, and Y. Goldberg, "Lavan: Localized and visible adversarial noise," in ICML'18, Stockholm, Sweden, Jul. 2018.

[35] K. Xu, S. Liu, P. Zhao, P. Chen, H. Zhang, Q. Fan, D. Erdogmus, Y. Wang, and X. Lin, "Structured adversarial attack: Towards general implementation and better interpretability," in ICLR'18, Vancouver, Canada, Nov. 2018.

[36] A. Modas, M. Dezfooli, and P. Frossard, "Sparsefool: a few pixels make a big difference," in CVPR'19, Virtual, Jun. 2019.

[37] X. Dong, D. Chen, J. Bao, C. Qin, L. Yuan, W. Zhang, N. Yu, and D. Chen, "Greedyfool: Distortion-aware sparse adversarial attack," in NeurIPS'20, Vancouver, Canada, Jan. 2020.

[38] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in S&P'17, San Jose, CA, Jun. 2017.

[39] L. Jiang, X. Ma, S. Chen, J. Bailey, and Y. Jiang, "Black-box adversarial attacks on video recognition models," in MM'19, New York, NY, Oct. 2019.

[40] H. Zhang, L. Zhu, Y. Zhu, and Y. Yang, "Motion-Excited Sampler: Video Adversarial Attack with Sparked Prior," in ECCV'20, Virtual, Aug. 2020.

[41] X. Wei, H. Yan, and B. Li, "Sparse black-box video attack with reinforcement learning," International Journal of Computer Vision, vol. 130, no. 6, pp. 1459 – 1473, Apr. 2022.

[42] Z. Cao, M. Long, J. Wang, and P. Yu, "Hashnet: Deep learning to hash by continuation," in ICCV'17, Venice, Italy, Oct. 2017.

[43] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in ICCV'15, Santiago, Chile, Dec. 2015.

[44] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in ICLR'15, San Diego, CA, May 2015.

[45] GitHub, "Supplementary material," https://github.com/xinyaocse/Supplementary_Material_for_DUO.

[46] B. Fan, Y. Wu, H. Li, Y. Zhang, Y. Li, F. Li, and J. Yang, "Sparse adversarial attack via perturbation factorization," in ECCV'20, Glasgow, UK, Aug. 2020.

[47] W. Gu, X. Gu, J. Gu, B. Li, Z. Xiong, and W. Wang, "Dependence guided unsupervised feature selection," in AAAI'18, New Orleans, LA, Feb. 2018.

[48] J. Yang, Y. Jiang, X. Huang, B. Ni, and C. Zhao, "Learning black-box attackers with transferable priors and query feedback," in NeurIPS'20, Virtual, Nov. 2020.

[49] X. Wei, J. Zhu, S. Yuan, and H. Su, "Sparse adversarial perturbations for videos," in AAAI'19, Honolulu, HI, Jan. 2019.

[50] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in CVPR'19, Long Beach, CA, Jun. 2019.

[51] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in CVPR'16, Las Vegas, NV, June. 2016.

[52] B. Yu and D. Tao, "Deep metric learning with tuplet margin loss," in CVPR'19, Long Beach, CA, Jun. 2019.

[53] C. Guo, J. Gardner, Y. You, A. Wilson, and K. Weinberger, "Simple black-box adversarial attacks," in ICLR'19, New Orleans, LA, May. 2019.