Reinforcement Learning Based Sparse Black-box Adversarial Attack on Video Recognition Models

Zeyuan Wang , **Chaofeng Sha**∗ and **Su Yang**∗

Shanghai Key Laboratory of Intelligent Information Processing,
School of Computer Science, Fudan University
{18210440020, cfsha, suyang}@fudan.edu.cn

Abstract

We explore the black-box adversarial attack on video recognition models. Attacks are only performed on selected key regions and key frames to reduce the high computation cost of searching adversarial perturbations on a video due to its high dimensionality. To select key frames, one way is to use heuristic algorithms to evaluate the importance of each frame and choose the essential ones. However, it is time inefficient on sorting and searching. In order to speed up the attack process, we propose a reinforcement learning based frame selection strategy. Specifically, the agent explores the difference between the original class and the target class of videos to make selection decisions. It receives rewards from threat models which indicate the quality of the decisions. Besides, we also use saliency detection to select key regions and only estimate the sign of gradient instead of the gradient itself in zeroth order optimization to further boost the attack process. We can use the trained model directly in the untargeted attack or with little fine-tune in the targeted attack, which saves computation time. A range of empirical results on real datasets demonstrate the effectiveness and efficiency of the proposed method.

1 Introduction

In recent years, more and more machine learning models, especially deep learning models, are deployed in various vital fields with high security requirements, such as face recognition, speech recognition, video surveillance, etc. However, these models are vulnerable to adversarial examples [Goodfellow *et al.*, 2014], which are formed by applying slight but worst-case perturbations to examples from the data distribution, and might be misclassified by models with high confidence. This increases people's interest in adversarial attacks. Some researchers have investigated resisting adversarial examples [Buckman *et al.*, 2018] to improve model security, while others have explored how to attack networks [Cheng

et al., 2018; Cheng *et al.*, 2019; Wei *et al.*, 2019; Wei *et al.*, 2020] to assess their robustness.

Different from the white-box attack in which attackers have access to the model's parameters, we focus on the black-box attack where the model is unknown to attackers in this work. Jia *et al.* [2019] summarize that there are two types of black-box attack methods for video recognition models, dense attack and sparse attack. In dense attack [Li *et al.*, 2018], all the frames in a video are polluted. However, the dimensionality of video data is high since videos have both temporal and spatial dimensions. Based on the insight that different regions or frames in a video contribute differently to the video classification result, and there contains redundant information in a video when feeding to the model, Wei *et al.* [2019] introduce sparse attack, where only some key frame and some key regions in a video are polluted, which leads to the reduction of adversarial perturbations and the increasing efficiency of attacking.

The major challenge in sparse attack on video recognition models is how to select key regions and key frames. Wei *et al.* [2020] introduce a heuristic algorithm, in which they evaluate the importance of each salient region and frame, select regions and frames according to the descending order of importance scores, and finally perform attacks on the selected regions of selected frames. However, the key frames selection process in their methods is not time efficient. In some conditions such as a video example is close to its decision boundary, the performance of this method can even be worse than dense attack methods.

Reinforcement Learning (RL) has strong potential in sequence decision making. To further improve the efficiency of video attacking, we propose a RL based sparse black-box adversarial attack method on video recognition models. Specifically, we model the key frames selection process as a Markov Decision Process (MDP). The agent in RL explores the differences between the original class and the target class of videos, and determines the initialized direction of optimization to make key frames selection decisions. The agent receives rewards by querying the threat model which indicate the quality of the current selection decision. We use a Convolutional Neural Network (CNN) to extract features from the noise vectors and sample frames with a multinomial distribution. Proximal Policy Optimization (PPO) algorithm [Schulman *et al.*, 2017] is applied to optimize the policy. In particular, we di-

---

∗Corresponding Authors

vide the policy optimization into a two-stage approach to accelerate the key frames selection process. The agent can learn a general policy using video examples from any classes first, and then fine-tune its policy using the video examples from the specific class. Optimization-based attack method is finally performed to do the attack. Besides, we also use the fine grained saliency approach to construct saliency maps to select key regions on each frame and only estimate the sign of gradient instead of the gradient itself when performing attacks using zeroth order optimization to further boost the process. The figure in the supplementary material overviews our proposed method. Our major contributions can be summarized as follows:

- We model the key frames selection process in sparse video attack as an MDP and design a two-stage training approach for policy optimization.

- A query-efficient sparse black-box adversarial attack method on video recognition models is proposed based on saliency detection, key frames selection and Sign-OPT method.

- Extensive experiments on two benchmark datasets and two widely used video recognition models demonstrate the effectiveness and efficiency of the proposed method. It reduces more than 34% in query numbers and time compared to the current attack methods.

2 Related Work

2.1 Adversarial Attack on Image and Video Models

Past works on adversarial attack are mostly focused on image recognition models. Many attack algorithms for image attack have been proposed, such as transfer based attack [Papernot *et al.*, 2017] and query-based attack [Cheng *et al.*, 2018; Cheng *et al.*, 2019].

Li *et al.* [2018] first extend adversarial attack to video recognition models with a generative dense attack method. Wei *et al.* [2019] propose that we don't need to pollute every frame and use an iterative optimization algorithm based on $l_{2,1}$-norm to sparsely attack the video classification models successfully. Zajac *et al.* [2019] introduce spatial sparsity to adversarial attack by only adding an adversarial framing on the border of each frame. However, these works are all white-box attacks. Jiang *et al.* [2019] propose the first black-box video attack framework which utilizes tentative perturbations and partition-based rectifications to obtain good adversarial gradient estimates with fewer queries. Sparse attack is introduced to black-box video attack in [Wei *et al.*, 2020], where they evaluate the importance of each salient region and each frame, heuristically search a subset of regions and frames, and only perform attacks on the selected subset. However, they consume too much queries sorting and searching, which offsets the advantages of sparsity. The recent arXiv paper [Yan *et al.*, 2020] first introduces RL to temporal sparse black-box video attack and they combine the key frames selection with the attacking step as a whole process. But they train agents for each model and use complex networks to extract features which is still not efficient enough.

Our method is also a sparse attack method. But unlike the algorithms mentioned above, the agent in our method only explores the difference between the original class and the target class of videos to make selection decisions and receives rewards from the threat model directly. It can be trained with a two-stage approach, and is much more query-efficient.

2.2 Deep Reinforcement Learning

Deep RL [Mnih *et al.*, 2013] is a category of artificial intelligence where agents can get rewards or punishments based on their actions, which is similar to the way humans learn from experience. It has received a lot of attention since AlphaGo [Silver *et al.*, 2016] beats humans, including those who work on adversarial attack problems. Lin *et al.* [2017] introduce tactics to attack deep RL agents, which is distinct from our work. Yang *et al.* [2020] use deep RL to generate adversarial image examples. However, except the recent arXiv paper [Yan *et al.*, 2020], there are no other examples that deep RL is applied to attack video recognition models.

In computer vision domain, Tang *et al.* [2018] use a deep progressive RL method to distil the most informative frames in sequences for recognizing actions. Zhou *et al.* [2018] formulate video summarization as a sequential decision-making process and develop a deep RL network to summarize videos. However, these works focus more on the inner features of a video and are different from ours.

3 Methodology

In this section, we introduce the proposed sparse adversarial attack method. For a soft-label black-box attack, we can only get the predicted top one class label and its probability from the video recognition model when given a clean video.

Denote the video recognition model as a function F. Specifically, it takes a clean video $\boldsymbol{x} \in \mathbb{R}^{T \times W \times H \times C}$ as an input and outputs the predicted top one class label y and its probability $p(y|\boldsymbol{x})$. Here T, W, H and C denote the length (i.e., the number of frames), width, height and the number of channels of the video, respectively. If the prediction is correct, then y equals the ground-truth label $\bar{y}$. Our goal is to generate an adversarial example $\boldsymbol{x}_{\text{adv}}$ close to $\boldsymbol{x}$ such that $F(\boldsymbol{x}_{\text{adv}}) \neq \bar{y}$ in the untargeted attack or $F(\boldsymbol{x}_{\text{adv}}) = y_{\text{adv}}$ in the targeted attack, where y_{adv} is the targeted adversarial class label. We introduce a mask $\boldsymbol{M} \in \{0, 1\}^{T \times W \times H \times C}$ to represent temporal and spatial sparsity. For each element in $\boldsymbol{M}$, only if its value is equal to 1, then the corresponding pixel in the video will be included in the perturbation process.

3.1 Saliency Detection for Spatial Sparsity

We apply spatial sparsity to each frame of the video by the saliency maps generation approach. This idea is from both the human visual system and some image and video classification models that focused locations contribute more to the recognition results.

We use the fine grained saliency approach [Sebastian and Alvaro, 2010] for saliency maps construction. It calculates saliency based on center-surround differences. We control the area ratio of the salient regions in one frame by introducing a parameter $\varphi \in (0, 1]$. Larger φ leads to smaller spatial sparsity. Denote the fine grained saliency detection as a function

S, it initializes $\boldsymbol{M}$ by setting the values of the salient regions for all video frames to 1. The initialized mask will be

$$\boldsymbol{M}_{\text{spatial}} = \begin{cases} S(\boldsymbol{x}, \varphi) & \text{(U)} \\ S(\boldsymbol{x}, \varphi) \cup S(\hat{\boldsymbol{x}}, \varphi) & \text{(T)}, \end{cases} \quad (1)$$

where (U) and (T) represent "in untargeted attack" and "in targeted attack", respectively. $\hat{\boldsymbol{x}}$ is a video sample of target class y_{adv} here in targeted attack and will be a video sample which does not belong to $\bar{y}$ and set as the "target direction" of attacking in untargeted attack.

3.2 Reinforcement Learning Based Key Frames Selection for Temporal Sparsity

For temporal sparsity, we use the "unimportant frames deletion" method to select a subset of frames that contribute the most to the success of an adversarial attack from the successive frames. We consider the key frames selection process as a multi-step MDP and use RL to learn a policy to select frames. In this subsection, we first formulate the MDP of key frames selection and then introduce the policy network and optimization algorithm.

Markov Decision Process for Key Frames Selection
The MDP consists of states, actions, transition function and reward function.

State. The state $\boldsymbol{s}_t$ at decision step t is set as the difference between the original class and the target class of videos masked by the current temporal and spatial sparsity mask

$$\boldsymbol{s}_t = (\hat{\boldsymbol{x}} - \boldsymbol{x}) \odot \boldsymbol{M}_t. \quad (2)$$

Here, $\odot$ represents Hadamard product. At the initial state $\boldsymbol{s}_0$, the mask is the spatial sparsity mask $\boldsymbol{M}_{\text{spatial}}$.

Action. An action a_t is a frame chosen from all T frames that the agent thinks it's not a key frame at decision step t.

State transition. Once the agent chooses action a_t, we set the values of a_tth frame in $\boldsymbol{M}_t$ to 0. Denote this process as a function D, then we can update the state to

$$\boldsymbol{s}_{t+1} = (\hat{\boldsymbol{x}} - \boldsymbol{x}) \odot \boldsymbol{M}_{t+1} = (\hat{\boldsymbol{x}} - \boldsymbol{x}) \odot D(\boldsymbol{M}_t, a_t). \quad (3)$$

$\boldsymbol{s}_{t+1}$ will be the terminal state either if $F(\boldsymbol{s}_{t+1}) = \bar{y}$ in untargeted attack or $F(\boldsymbol{s}_{t+1}) \neq y_{adv}$ in targeted attack (i.e., the a_tth frame is a key frame), or $a_t \in \{a_1, a_2, ..., a_{t-1}\}$ (i.e., the a_tth frame has already been chosen in the past decision steps).

Reward. The reward reflects the quality of the chosen action. Our goal is that the agent learns to delete the least important frame and maximizes the long term expected reward. Based on this intention, we design the reward during pretraining as

$$r_t = \begin{cases} -1 & \text{if } a_t \in \{a_1, a_2, ..., a_{t-1}\} \\ 0 & \text{else if } F(\boldsymbol{s}_{t+1}) = \bar{y} \text{ (U) or } F(\boldsymbol{s}_{t+1}) \neq y_{adv} \text{ (T)} \\ 1 & \text{else if } F(\boldsymbol{s}_{t+1}) \neq \bar{y} \text{ (U) or } F(\boldsymbol{s}_{t+1}) = y_{adv} \text{ (T).} \end{cases} \quad (4)$$

We set different rewards for the two terminal states to distinguish between deleting unimportant frames and duplication. The former one might be a helpless action when no other unimportant frames can be found, while the latter one should not be done at any time. Note that in targeted attack,

---

Algorithm 1 Policy optimization for selecting key frames

Input: Clean video set $\boldsymbol{X}$, target class video set $\hat{\boldsymbol{X}}$, video recognition model F.
Parameter: Iteration number I, actor number N, timestep length T_{ts}, epoch number K, batch size $M \leq NT_{\text{ts}}$.
Output: Optimal policy parameter $\boldsymbol{\theta}_{\text{p}}^*$.
1: Initialize policy and value network parameter $\boldsymbol{\theta}_{\text{p}}, \boldsymbol{w}$;
2: **for** iteration $= 1, 2, \cdots, I$ **do**
3:     **for** actor $= 1, 2, \cdots, N$ **do**
4:         **while** timesteps $< T_{\text{ts}}$ **do**
5:             Get random $\boldsymbol{x}, \hat{\boldsymbol{x}}$ from $\boldsymbol{X}, \hat{\boldsymbol{X}}$;
6:             $\boldsymbol{s} \leftarrow (\hat{\boldsymbol{x}} - \boldsymbol{x}) \odot \boldsymbol{M}_{\text{spatial}}$;
7:             **while** $\boldsymbol{s}$ is not terminal state **do**
8:                 Run policy $\pi_{\boldsymbol{\theta}_{\text{old}}}$, update state and get reward;
9:             **end while**
10:         **end while**
11:         Compute advantage estimate A (Equation (9));
12:     **end for**
13:     Optimize $\boldsymbol{\theta}_{\text{p}}, \boldsymbol{w}$ for K epochs with batch size M;
14:     $\boldsymbol{\theta}_{\text{old}} \leftarrow \boldsymbol{\theta}_{\text{p}}$;
15: **end for**
16: **return** $\boldsymbol{\theta}_{\text{p}}$.

---

for certain $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$, the larger value of $p(y_{\text{adv}}|\boldsymbol{s}_{t+1})$ means the less importance of a_tth frame toward generating the adversarial example. Thus we define the reward during fine-tuning as

$$r_t = \begin{cases} -1 & \text{if } a_t \in \{a_1, a_2, ..., a_{t-1}\} \\ 0 & \text{else if } F(\boldsymbol{s}_{t+1}) \neq y_{\text{adv}} \\ p(y_{\text{adv}}|\boldsymbol{s}_{t+1}) & \text{else if } F(\boldsymbol{s}_{t+1}) = y_{\text{adv}}. \end{cases} \quad (5)$$

Policy Network
The agent behavior is defined by its policy. We use a policy network $\pi_{\boldsymbol{\theta}_{\text{p}}}(a_t|\boldsymbol{s}_t)$ to parameterize the policy.

Network Architecture. The input of the policy network at decision step t is the current state $\boldsymbol{s}_t$. The network consists of 5 convolutional layers and 3 fully connected (FC) layers. The kernel size and stride in the time dimension is set to 1 which makes the network only perform spatial convolutions on each frame. We use average pooling to sample the overall feature information and to reduce the parameter dimension. The FC layers compare time domain information of a frame with its surrounding frames and it ends with the softmax function which predicts a probability distribution $\boldsymbol{p}_t$ for each frame.

Action Selection. In the training process, the action $\boldsymbol{a}_t$ is sampled from a multinomial distribution as

$$a_t = \text{Multinomial}(\boldsymbol{p}_t). \quad (6)$$

When making decisions, the agent selects the frame with max probability according to

$$a_t = \arg\max_a(\boldsymbol{p}_t). \quad (7)$$

Policy Optimization
We train the policy network $\pi_{\boldsymbol{\theta}_{\text{p}}}(a_t|\boldsymbol{s}_t)$ to optimize the expected sum of rewards $L(\boldsymbol{\theta}_{\text{p}}) = \mathbb{E}_t[R(\boldsymbol{s}_{1:t})|\boldsymbol{s}_0, \boldsymbol{\theta}_{\text{p}}] =$

$\sum_{\boldsymbol{s}_i} \sum_{a_i} \pi_{\boldsymbol{\theta}_\mathrm{p}}(a_i|\boldsymbol{s}_i)r(\boldsymbol{s}_i, a_i)$. Here, the expectation $\mathbb{E}$ indicates the empirical average over a finite batch of samples.

We apply PPO algorithm with clipped surrogate objective [Schulman *et al.*, 2017], which we found to be more stable and sample efficient than REINFORCE [Williams, 1992]. We use a value network $V(\boldsymbol{s}_t)$ to approximate the expected future rewards for being in state $\boldsymbol{s}_t$. The value network has a similiar architecture as the policy network but the top FC layer only outputs a single value. Denote $\boldsymbol{\theta}_\mathrm{old}$ as the policy parameters before the update, and $v_t(\boldsymbol{\theta}_\mathrm{p})$ as the probability ratio $\frac{\pi_{\boldsymbol{\theta}_\mathrm{p}}(a_t|\boldsymbol{s}_t)}{\pi_{\boldsymbol{\theta}_\mathrm{old}}(a_t|\boldsymbol{s}_t)}$. The surrogate objective function that PPO maximizes can be written as

$$L(\boldsymbol{\theta}_\mathrm{p}) = \mathbb{E}_t[\min(v_t(\boldsymbol{\theta}_\mathrm{p})A_t, \mathrm{clip}(v_t(\boldsymbol{\theta}_\mathrm{p}), 1 - \epsilon_\mathrm{p}, 1 + \epsilon_\mathrm{p})A_t)]. \quad (8)$$

The first term inside min, $v_t(\boldsymbol{\theta}_\mathrm{p})A_t$, is the the objective used in Conservative Policy Iteration (CPI) algorithm [Kakade and Langford, 2002] which finds an approximately optimal policy. The second term $\mathrm{clip}(v_t(\boldsymbol{\theta}_\mathrm{p}), 1 - \epsilon_\mathrm{p}, 1 + \epsilon_\mathrm{p})A_t$ modifies the objective by clipping $v_t(\boldsymbol{\theta}_\mathrm{p})$ into the interval $[1 - \epsilon_\mathrm{p}, 1 + \epsilon_\mathrm{p}]$. We take the minimum of the unclipped and clipped objective. This scheme penalizes changes to the policy that move $v_t(\boldsymbol{\theta}_\mathrm{p})$ away from 1. We use the advantage estimator

$$A_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{(G-t+1)}\delta_{G-1}, \quad (9)$$

where G is the total number of timesteps when encountering the next terminal state and $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$. In our experiments, we set the hyperparameters $\epsilon_\mathrm{p} = 0.2$, $\gamma = 0.99$ and $\lambda = 0.95$.

During optimization, in each iteration, each of N parallel actors $\pi_{\boldsymbol{\theta}_\mathrm{old}}$ collect T_ts timesteps of data and compute advantage estimates $A_1, A_2, \cdots, A_{T_\mathrm{ts}}$. Then we construct the surrogate function on these NT_ts timesteps of data, and optimize it with Adam for K epochs. Algorithm 1 summarizes the whole procedure of policy ptimization.

3.3 Video Attacking

There are several algorithms that have been proposed for the black-box attack setting for generating adversarial images. However, most of them require huge amount of queries for attacking one sample. Cheng *et al.* [2018] suggest that this problem can be formulated as an optimization problem, where the objective function can be evaluated by binary search with additional model queries. Thereby a zeroth order optimization algorithm can be applied to solve the formulation. They [2019] later improved it by directly estimate the sign of gradient at any direction instead of the gradient itself, which enjoys the benefit of single query. Our attack algorithm is built based on their Sign-OPT algorithm.

For simplicity, we only focus on untargeted attack here, while the same procedure can be applied to targeted attack as well. In our formulation, $\boldsymbol{\theta}_\mathrm{d}$ represents the search direction and $g(\boldsymbol{\theta}_\mathrm{d})$ is the distance from the given example $\boldsymbol{x}$ to the nearest adversarial example along the direction $\boldsymbol{\theta}_\mathrm{d}$. The objective function can be written as

$$\min_{\boldsymbol{\theta}_\mathrm{d}} \ g(\boldsymbol{\theta}_\mathrm{d}) \text{ where } g(\boldsymbol{\theta}_\mathrm{d}) = \arg\min_{\lambda>0}(F(\boldsymbol{x} + \lambda\frac{\boldsymbol{\theta}_\mathrm{d}}{\|\boldsymbol{\theta}_\mathrm{d}\|}) \neq \bar{y}). \quad (10)$$

It can be evaluated by a binary search procedure locally. At each binary search step, we query the model $F(\boldsymbol{x} + \lambda\frac{\boldsymbol{\theta}_\mathrm{d}}{\|\boldsymbol{\theta}_\mathrm{d}\|})$

Algorithm 2 Reinforcement learning based sparse targeted video attack

Input: Video recognition model F, clean video $\boldsymbol{x}$, target class label y_adv, target class video set $\hat{\boldsymbol{X}}$, pretrained policy parameter $\boldsymbol{\theta}_\mathrm{p}$, pretrained value network parameter $\boldsymbol{w}$.
Parameter: Sampled examples number n, salient region ration φ, attack iteration number T_ai, gradient sampling number Q, smoothing parameter ϵ_d and step size η.
Output: Adversarial example x_adv.

1: Fine-tune the policy network $\pi_{\boldsymbol{\theta}_\mathrm{p}}$ using Algorithm 1 with $\boldsymbol{x}$ and random samples from $\hat{\boldsymbol{X}}$;
2: **for** $i = 1, 2, \cdots, n$ **do**
3:     Get random $\hat{\boldsymbol{x}}$ from $\hat{\boldsymbol{X}}$;
4:     $\boldsymbol{M}_\mathrm{spatial} \leftarrow S(\boldsymbol{x}, \varphi) \cup S(\hat{\boldsymbol{x}}, \varphi)$;
5:     Run policy $\pi_{\boldsymbol{\theta}_\mathrm{p}}$ to get temporal sparse mask $\boldsymbol{M}$;
6:     $\hat{y} \leftarrow F(\boldsymbol{x} + (\hat{\boldsymbol{x}} - \boldsymbol{x}) \odot \boldsymbol{M})$;
7:     **if** $\hat{y} = y_\mathrm{adv}$ **then**
8:        $\boldsymbol{\theta} \leftarrow \frac{(\hat{\boldsymbol{x}} - \boldsymbol{x}) \odot \boldsymbol{M}}{\|(\hat{\boldsymbol{x}} - \boldsymbol{x}) \odot \boldsymbol{M}\|}$;
9:        Compute $g(\boldsymbol{\theta})$ using the binary search algorithm;
10:        **if** $MAP(g(\boldsymbol{\theta}) \odot \boldsymbol{\theta}) < MAP(g(\boldsymbol{\theta}_\mathrm{d}) \odot \boldsymbol{\theta}_\mathrm{d})$ **then**
11:          $\boldsymbol{M}^*, \boldsymbol{\theta}_\mathrm{d} \leftarrow \boldsymbol{M}, \boldsymbol{\theta}$;
12:        **end if**
13:     **end if**
14: **end for**
15: **for** $t = 1, 2, \cdots, T_\mathrm{ai}$ **do**
16:     Sample random $\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_Q$ from Gaussian distribution $\mathcal{N}(0, 1)$;
17:     $\hat{\boldsymbol{g}} \leftarrow \frac{1}{Q_\mathrm{d}}\sum_{q=1}^{Q_\mathrm{d}} \mathrm{sign}(g(\boldsymbol{\theta}_\mathrm{d} + \epsilon_\mathrm{d}\boldsymbol{u}_q) - g(\boldsymbol{\theta}_\mathrm{d}))\boldsymbol{u}_q$;
18:     $\boldsymbol{\theta}_\mathrm{d} \leftarrow \boldsymbol{\theta}_\mathrm{d} - \eta\hat{\boldsymbol{g}}$;
19:     Evaluate $g(\boldsymbol{\theta}_\mathrm{d})$ using the binary search algorithm;
20: **end for**
21: $\boldsymbol{x}_\mathrm{adv} = \boldsymbol{x} + g(\boldsymbol{\theta}_\mathrm{d}) \cdot \boldsymbol{\theta}_\mathrm{d}$;
22: **return** $\boldsymbol{x}_\mathrm{adv}$.

---

and determine whether the distance to decision boundary in the direction $\boldsymbol{\theta}_\mathrm{d}$ is greater or smaller than λ based on the prediction. Usually, its directional derivative can be estimated by finite differences $\hat{\nabla}g(\boldsymbol{\theta}_\mathrm{d}; \boldsymbol{u}) = \frac{g(\boldsymbol{\theta}_\mathrm{d} + \epsilon_\mathrm{d}\boldsymbol{u}) - g(\boldsymbol{\theta}_\mathrm{d})}{\epsilon_\mathrm{d}}\boldsymbol{u}$, where $\boldsymbol{u}$ is a random Gaussian vector and ϵ_d is a very small positive smoothing parameter. However, this approach consumes a lot of queries when computing $g(\boldsymbol{\theta}_\mathrm{d} + \epsilon_\mathrm{d}\boldsymbol{u}) - g(\boldsymbol{\theta}_\mathrm{d})$. To improve query complexity, we do not need very accurate values of directional derivative and an imperfect but informative estimation of directional derivative of g which can be computed by a single query is enough for us. Thus, we can compute the sign of this value by

$$\mathrm{sign}(g(\boldsymbol{\theta}_\mathrm{d} + \epsilon_\mathrm{d}\boldsymbol{u}) - g(\boldsymbol{\theta}_\mathrm{d})) =$$

$$\begin{cases} 1 & \text{if } F(\boldsymbol{x} + g(\boldsymbol{\theta}_\mathrm{d})\frac{\boldsymbol{\theta}_\mathrm{d} + \epsilon_\mathrm{d}\boldsymbol{u}}{\|\boldsymbol{\theta}_\mathrm{d} + \epsilon_\mathrm{d}\boldsymbol{u}\|}) \neq \bar{y} \\ -1 & \text{Otherwise,} \end{cases} \quad (11)$$

and estimate the imperfect gradient by sampling Q_d random Gaussian vectors

$$\hat{\nabla}g(\boldsymbol{\theta}_\mathrm{d}) \approx \hat{\boldsymbol{g}} = \frac{1}{Q_\mathrm{d}}\sum_{q=1}^{Q_\mathrm{d}} \mathrm{sign}(g(\boldsymbol{\theta}_\mathrm{d} + \epsilon_\mathrm{d}\boldsymbol{u}_q) - g(\boldsymbol{\theta}_\mathrm{d}))\boldsymbol{u}_q. \quad (12)$$

Dataset	Threat Model	Attack Methods	Untargeted Attacks				Targeted Attacks			
			Q	MAP	S (%)	t (min)	Q	MAP	S (%)	t (min)
UCF-101	C3D	OPT	54313.7	3.8807	0.00	54.01	124768.0	12.6996	0.00	64.14
		Heuristic	49648.8	3.8358	19.00	52.70	198685.7	10.0651	29.50	104.31
		Sign-OPT	13305.7	2.9076	0.00	14.08	31639.8	8.9471	0.00	24.71
		Our method	**4172.9**	**2.4333**	**69.63**	**5.72**	**12990.5**	**8.8138**	47.85	**8.49**
	LRCN	OPT	12011.7	2.8482	0.00	12.27	214718.7	18.2769	0.00	122.53
		Heuristic	12585.4	2.6900	15.05	13.24	265144.3	15.1814	24.46	144.36
		Sign-OPT	4945.1	2.4269	0.00	5.03	32485.1	8.9249	0.00	24.92
		Our method	**2201.5**	**2.2892**	**54.78**	**2.90**	**18472.2**	**8.8039**	45.53	**11.65**
HMDB-51	C3D	OPT	21328.3	3.4674	0.00	24.79	43975.0	12.1758	0.00	28.51
		Heuristic	29071.3	3.1053	18.13	33.55	103050.8	10.1357	**45.13**	53.81
		Sign-OPT	4273.7	2.8127	0.00	4.62	24370.0	**8.7348**	0.00	17.79
		Our method	**2636.6**	**2.3883**	**63.63**	**3.85**	**16066.9**	8.7568	41.86	**9.98**
	LRCN	OPT	24857.0	2.8578	0.00	29.54	40072.9	10.9374	0.00	22.17
		Heuristic	29340.4	**2.5014**	20.25	31.62	88080.9	8.9667	30.09	47.42
		Sign-OPT	4457.9	2.5747	0.00	4.82	24063.9	8.7354	0.00	17.43
		Our method	**2703.6**	2.5697	**67.69**	**3.81**	**15742.2**	**8.7305**	45.42	**9.78**

Table 1: Untargeted and targeted attack average results on C3D and LRCN models. For all attack models, FR is 100%. Q and t used for pretraining are not counted. In each task, the best performance is emphasized with the bold number.

When attacking, we first initialize $\boldsymbol{\theta}_d$ using

$$\boldsymbol{\theta}_d = \frac{(\hat{\boldsymbol{x}} - \boldsymbol{x}) \odot \boldsymbol{M}}{\|(\hat{\boldsymbol{x}} - \boldsymbol{x}) \odot \boldsymbol{M}\|}. \qquad (13)$$

We sample n examples of target class video $\hat{\boldsymbol{x}}$ randomly, and choose the one with smallest $g(\boldsymbol{\theta}_d)$. This helps us to find a good initialization direction and thus get a smaller distortion in the end. Then we update $\boldsymbol{\theta}_d$ with the zeroth order optimization and get the optimal direction $\boldsymbol{\theta}_d^*$. Finally, we can get the adversarial example $\boldsymbol{x}_{adv}$ using

$$\boldsymbol{x}_{adv} = \boldsymbol{x} + g(\boldsymbol{\theta}_d^*) \cdot \boldsymbol{\theta}_d^*. \qquad (14)$$

We set $n = 100$ in all of the following experiments.

The whole process of our method for targeted attack is described in Algorithm 2. The pretrained parameter $\boldsymbol{\theta}_p$ and $\boldsymbol{w}$ is optimized by Algorithm 1. Function MAP computes the mean absolute perturbation which we will define in Equation (15). For untargeted attack, we skip the fine-tuning step since the pretrained policy network is good enough.

4 Experiments

4.1 Experimental Setting

Datasets. Similar to [Wei *et al.*, 2020], we use UCF-101 [Soomro *et al.*, 2012] and HMDB-51 [Kuehne *et al.*, 2011] in our experiments. UCF-101 contains 13,320 videos collected on YouTube with 101 action categories. HMDB-51 contains 7000 clips downloaded from various cinemas distributed in 51 action classes. We split 70% of the videos in both datasets as training set and the remaining 30% as test set. We use 16-frame snippets that are uniformly sampled from each video as input samples of threat models and only consider videos that are correctly predicted by the threat model. During pretraining, we use the training set as both clean video set $\boldsymbol{X}$ and target class video set $\hat{\boldsymbol{X}}$. When fine-tuning, we choose a video from test set as clean video $\boldsymbol{x}$ and the corresponding training set as target class video set $\hat{\boldsymbol{X}}$.

Threat Models. In our experiments, two video recognition models, Convolutional 3D ResNeXt (C3D) [Hara *et al.*, 2018] and Long-term Recurrent Convolutional Networks (LRCN) [Donahue *et al.*, 2015] are used as threat models.

Metrics. We use five metrics to evaluate the performance of our method in various aspects: 1) Fooling ratio (FR) is defined as the percentage of adversarial examples that are successfully misclassified. 2) Query number (Q) counts the times of queries to finish an attack. 3) Mean absolute perturbation (MAP) [Wei *et al.*, 2019] denotes the perceptibility score of the adversarial perturbation:

$$MAP = \frac{1}{N_p} \sum_i |\mathbf{r}_i|, \qquad (15)$$

where N_p is the number of pixels and $\mathbf{r}_i$ is the intensity vector (3-dimensional in the RGB color space). 4) Sparsity (S) represents the proportion of pixels with no perturbations versus all the pixels in a specific video, which is defined as

$$S = 1 - \frac{1}{T} \sum_i \varphi_i, \qquad (16)$$

where φ_i is the salient region ratio of the corresponding important frame and T is the total number of frames in a video. 5) Time (t) counts the time used for finishing an attack. The computing infrastructures used for running experiments include 8 Nvidia GeForce RTX 2080 Ti GPUs, Intel Xeon E5-2680 v4 CPU, 320 GB memory and Ubuntu 16.04.1.

4.2 Parameter Setting

There are several parameters need to be set in our algorithms. We set most of them by the grid search method. Here we only take the parameter salient region ratio φ as an example. Since when $\varphi < 0.3$ most initial states will be terminal states, we fix other parameters and set φ as $\{0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. We use 20 randomly sampled videos from the original training set of UCF-101 as validation set to do the experiment. We find that larger
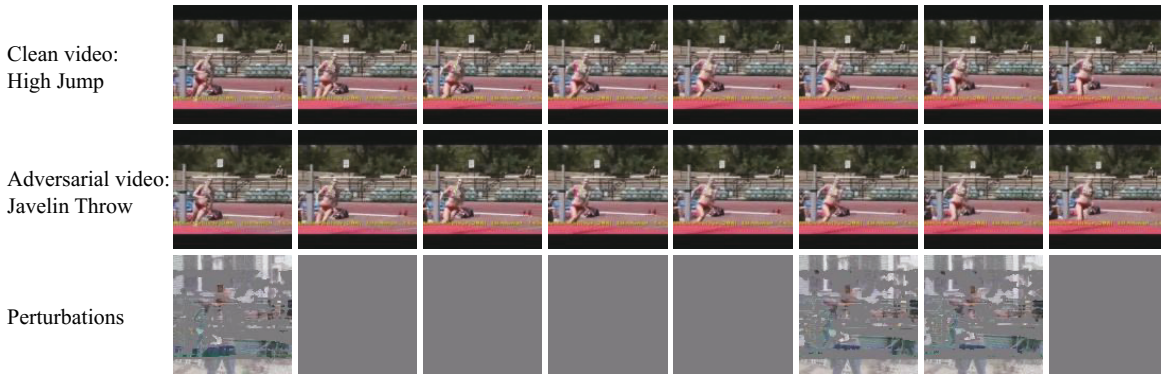
Figure 1: The first 8 frames of an example of adversarial video generated in the untargeted attack by our method. The frames of clean video are shown in the top row and the corresponding adversarial frames are shown in the middle row. The corresponding perturbations are re-scaled into the range of 0-255 and then visualized in the bottom row. The place in grey represents that there is no perturbation.

φ leads to larger MAP and Q, while smaller φ causes shorter decision trajectory and less temporal sparsity which also leads to larger MAP and Q. Finally we choose a median φ which gets relatively small MAP, Q and large S as $\varphi = 0.6$.

4.3 Performance Comparison

We compare our method with the OPT attack [Cheng *et al.*, 2018] and heuristic attack [Wei *et al.*, 2020], which we set as the baselines of black-box adversarial attack on video recognition models. We also compare it with the Sign-OPT attack [Cheng *et al.*, 2019] as an ablation study for the sparsity in our method. We are unable to compare our method with [Yan *et al.*, 2020] since the code of its model is not publicly available. The evaluations are performed with two video recognition models on two datasets mentioned above. For fair comparison, we set the bound of MAP as $MAP = 3$ in untargeted attack and $MAP = 9$ in targeted attack, according to the median of best performances in the above works. The iteration of attacking steps will terminate either if MAP reaches the bound or the iteration number exceeds the maximum iteration number $T_{ai} = 1000$.

Table 1 lists the performance comparisons regarding to both the untargeted attack and the targeted attack on two datasets. The average query numbers Q and time t (min) for pretraining are 17600.5, 11.72 for untargeted attack and 30052.2, 18.68 for targeted attack, respectively. The cost of pretraining can easily be offset by no more than 10 times of attack and can be ignored when doing numbers of attacks.

For untargeted attacks, compared to other methods, we have the following observations. First, our method significantly reduces the query numbers by more than 38%. Particularly, it verifies our sparsing strategy's efficiency when comparing to Sign-OPT method with no sparsity. Second, both our method and Sign-OPT method can reach the MAP bound within a limited iteration number, while OPT method and heuristic method fail on some examples when attacking C3D model. Third, our method gets better sparsity than heuristic method. It shows that our agent learns a better frames selection strategy than their method. Last, by looking into the detailed results, we found that only around 60%

of the queries are spent on attacking, while others are spent on inferring.

For targeted attacks, which usually need more query numbers and perturbations than untargeted attacks due to their difficulty, similar trends are observed as untargeted attacks. Our method reduces the query numbers by more than 34% and can always reach the MAP bound within a limited iteration number. It gets better sparsity than heuristic method on most examples. We also found that heuristic method does not preform as expectation since many of its generated sparse directions make the attack fall into local optimal solutions. For summary, our our proposed method achieves the best performance in the majority of test tasks. It demonstrates the effectiveness and the efficiency of our method.

Figure 1 gives an visualized example of adversarial video generated by our method. The ground-truth label for the clean video is "High Jump". By adding the generated human-imperceptible adversarial perturbations, the model tends to predict a wrong label "Javelin Throw" at the top-1 place. The generated adversarial perturbations are quite sparse.

5 Conclusion

In this work, we proposed an RL based sparse black-box adversarial attack method on video recognition models. To speed up the attack process and to reduce query numbers, we explored the sparsity of adversarial perturbations. For temporal domain, we modeled the key frames selection process as an MDP. The agent in our model continuously explored the differences between videos of distinct classes and received rewards from the threat model. Combined with policy optimization method, it finally learned an optimized frame selection strategy. We also used saliency detection to select key regions and Sign-OPT method which only estimates the sign of gradient in zeroth order optimization to further speed up the attack process. Our algorithm is applicable to multiple threat models and video datasets. The experimental results demonstrated that our algorithm is more effective and more efficient to generate adversarial examples than previous methods.

The pertinent area of future work is to investigate the black-box attack on video recognition models using fewer queries and time.

References

[Buckman *et al.*, 2018] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *ICLR*, 2018.

[Cheng *et al.*, 2018] Minhao Cheng, Thong Le, Pin-Yu Chen, Huan Zhang, JinFeng Yi, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. In *ICLR*, 2018.

[Cheng *et al.*, 2019] Minhao Cheng, Simranjit Singh, Patrick H Chen, Pin-Yu Chen, Sijia Liu, and Cho-Jui Hsieh. Sign-OPT: A query-efficient hard-label adversarial attack. In *ICLR*, 2019.

[Donahue *et al.*, 2015] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, pages 2625–2634, 2015.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[Hara *et al.*, 2018] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and imagenet? In *CVPR*, pages 6546–6555, 2018.

[Jia *et al.*, 2019] Xiaojun Jia, Xingxing Wei, and Xiaochun Cao. Identifying and resisting adversarial videos using temporal consistency. *arXiv preprint arXiv:1909.04837*, 2019.

[Jiang *et al.*, 2019] Linxi Jiang, Xingjun Ma, Shaoxiang Chen, James Bailey, and Yu-Gang Jiang. Black-box adversarial attacks on video recognition models. In *ACM MM*, pages 864–872, 2019.

[Kakade and Langford, 2002] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pages 267–274, 2002.

[Kuehne *et al.*, 2011] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. HMDB: A large video database for human motion recognition. In *ICCV*, pages 2556–2563. IEEE, 2011.

[Li *et al.*, 2018] Shasha Li, Ajaya Neupane, Sujoy Paul, Chengyu Song, Srikanth V Krishnamurthy, Amit K Roy Chowdhury, and Ananthram Swami. Adversarial perturbations against real-time video classification systems. *arXiv preprint arXiv:1807.00458*, 2018.

[Lin *et al.*, 2017] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. In *IJCAI*, pages 3756–3762, 2017.

[Mnih *et al.*, 2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[Papernot *et al.*, 2017] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *ASIACCS*, pages 506–519, 2017.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Sebastian and Alvaro, 2010] Montabone Sebastian and Soto Alvaro. Human detection using a mobile platform and novel features derived from a visual saliency mechanism. *Image and Vision Computing*, 28(3):391–402, 2010.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[Soomro *et al.*, 2012] Khurram Soomro, Amir Roshan Zamir, and M Shah. A dataset of 101 human action classes from videos in the wild. *Center for Research in Computer Vision*, 2(11), 2012.

[Tang *et al.*, 2018] Yansong Tang, Yi Tian, Jiwen Lu, Peiyang Li, and Jie Zhou. Deep progressive reinforcement learning for skeleton-based action recognition. In *CVPR*, pages 5323–5332, 2018.

[Wei *et al.*, 2019] Xingxing Wei, Jun Zhu, Sha Yuan, and Hang Su. Sparse adversarial perturbations for videos. In *AAAI*, volume 33, pages 8973–8980, 2019.

[Wei *et al.*, 2020] Zhipeng Wei, Jingjing Chen, Xingxing Wei, Linxi Jiang, Tat-Seng Chua, Fengfeng Zhou, and Yu-Gang Jiang. Heuristic black-box adversarial attacks on video recognition models. In *AAAI*, pages 12338–12345, 2020.

[Williams, 1992] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[Yan *et al.*, 2020] Huanqian Yan, Xingxing Wei, and Bo Li. Sparse black-box video attack with reinforcement learning. *arXiv preprint arXiv:2001.03754*, 2020.

[Yang *et al.*, 2020] Chenglin Yang, Adam Kortylewski, Cihang Xie, Yinzhi Cao, and Alan Yuille. Patchattack: A black-box texture-based attack with reinforcement learning. In *European Conference on Computer Vision*, pages 681–698. Springer, 2020.

[Zajac *et al.*, 2019] Michał Zajac, Konrad Zołna, Negar Rostamzadeh, and Pedro O Pinheiro. Adversarial framing for image and video classification. In *AAAI*, volume 33, pages 10077–10078, 2019.

[Zhou *et al.*, 2018] Kaiyang Zhou, Yu Qiao, and Tao Xiang. Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward. In *AAAI*, volume 32, 2018.