Breaking Temporal Consistency:
Generating Video Universal Adversarial Perturbations Using Image Models

Hee-Seon Kim, Minji Son, Minbeom Kim, Myung-Joon Kwon, Changick Kim
Korea Advanced Institute of Science and Technology (KAIST)
{hskim98, ming0103, alsqja1754, kwon19, changick}@kaist.ac.kr

Abstract

As video analysis using deep learning models becomes more widespread, the vulnerability of such models to adversarial attacks is becoming a pressing concern. In particular, Universal Adversarial Perturbation (UAP) poses a significant threat, as a single perturbation can mislead deep learning models on entire datasets. We propose a novel video UAP using image data and image model. This enables us to take advantage of the rich image data and image model-based studies available for video applications. However, there is a challenge that image models are limited in their ability to analyze the temporal aspects of videos, which is crucial for a successful video attack. To address this challenge, we introduce the Breaking Temporal Consistency (BTC) method, which is the first attempt to incorporate temporal information into video attacks using image models. We aim to generate adversarial videos that have opposite patterns to the original. Specifically, BTC-UAP minimizes the feature similarity between neighboring frames in videos. Our approach is simple but effective at attacking unseen video models. Additionally, it is applicable to videos of varying lengths and invariant to temporal shifts. Our approach surpasses existing methods in terms of effectiveness on various datasets, including ImageNet, UCF-101, and Kinetics-400.

1. Introduction

Deep learning models have achieved remarkable performance in various computer vision tasks [5, 2, 33, 13, 1], including image and video recognition. However, there is growing concern about the robustness and reliability of these models, as they have been shown to be vulnerable to adversarial attacks [34, 6, 42]. Adversarial attacks use imperceptible perturbations to manipulate the inputs to produce inaccurate predictions. These attacks can have serious consequences in various applications of deep neural networks, such as autonomous vehicles and surveillance cameras [35] where false activity detection [19] can cause serious consequences. Despite these concerns, the problem of adversarial attacks on video models remains largely unsolved.

Adversarial attacks can be broadly categorized into white-box [37, 14, 29] and black-box [39, 40] attacks. White-box attacks exploit model information to generate adversarial examples, while black-box attacks are more challenging due to the lack of model access. In real-world scenarios, accessing the target model is often difficult or impossible, so black-box attacks are more practical. One way to launch black-box attacks is by leveraging the transferability of adversarial examples [39, 40, 42, 7, 24], applying adversarial examples crafted using accessible source models to the target models. Transfer-based attacks can also be cross-modal [40], which enables attackers to transfer adversarial examples between different modalities, such as image to video. For most cases, crafting adversarial examples still requires optimization for each individual adversarial example. On the other hand, Universal Adversarial Perturbations (UAPs) [26, 37, 14, 43, 23] poses a powerful threat as a sin-
Our study aims to extend the applicability of UAPs generated using image data and models, to the domain of video data and models. The overall scheme is illustrated in Fig. 1. This extension allows significant benefits as it allows us to leverage the wealth of image data [4] and image model-based studies [6, 42, 7, 24, 3] available for video applications. Furthermore, generating UAPs using image data requires relatively less computation compared to using video data. However, we face significant challenges due to the lack of access to video data [32, 17] and video models [9, 45, 36]. There are two main challenges in generating adversarial videos using image models only [12, 30, 15]. Firstly, image models have limited capability in effectively analyzing the passage of time, which is a crucial aspect for videos. Secondly, UAPs should be applicable to unseen videos of varying lengths. Despite the importance of temporal information, prior research has not been able to address these challenges.

As the first paper to consider temporal information in video attacks using image models and data, our study addresses this issue with the Breaking Temporal Consistency (BTC) method, as illustrated in Fig. 2. Our target UAP is a video consisting of N frames. Motivated by the high similarity pattern between neighboring frames in the original video, our UAP aims to generate adversarial videos that have opposite patterns to the original. To achieve this, we jointly optimize the adversarial and temporal aspects of the UAPs. First, to make the UAPs adversarial, we minimize the feature similarity between the original and adversarial images in the feature space using the Adversarial Loss. We treat the frames of the UAPs as images, and add them to the original to create corresponding adversarial images. To ensure length-agnosticity of adversarial Perturbation. To ensure length-agnosticity of adversarial images, we use an Adversarial Loss and prevent overfitting using the Feature Diversity method. Additionally, while treating the adversarial images as a pseudo video, we apply the Temporal Similarity Loss to the video frames and make each frame distinct from one another.

Figure 2: Details of Breaking Temporal Consistency Method. Our goal is to create BTC-UAP for video attacks composed of N frames. We treat each frame of the UAP as an individual image, and add it to the original image to generate corresponding adversarial images. To ensure that these images are adversarial, we use an Adversarial Loss and prevent overfitting using the Feature Diversity method. Additionally, while treating the adversarial images as a pseudo video, we apply the Temporal Similarity Loss to the video frames and make each frame distinct from one another.
Figure 3: The feature similarity of frames within videos. This heatmap shows the average feature similarity between frames in the UCF-101 dataset, with brighter colors indicating lower levels of similarity.

- BTC-UAP is both temporal shift invariant and length-agnostic, making it a highly practical video attack method that can be applied to videos of varying lengths and datasets. We demonstrate the effectiveness of BTC-UAP through extensive experiments on various datasets, including ImageNet, UCF-101, and Kinetics-400, outperforming existing methods.

2. Related Work

2.1. Adversarial Attacks

Deep learning models are effective in computer vision tasks, but they can be easily fooled by adding imperceptible noise, which is known as adversarial perturbations. The adversarial perturbation is added to the original data to create an adversarial example, and using this adversarial example to attack a deep learning model is called an adversarial attack.

2.2. Image Classification Attacks

As studies on adversarial attacks began with tricking image classification models, various image classification attack methods have been developed [10, 20, 6, 42, 7, 24, 3, 31]. In the first stage, white-box image-specific adversarial attack methods were introduced. Fast Gradient Sign Method (FGSM) [10] creates adversarial examples by updating an input image with its gradient calculated to increase the classification loss. FGSM evolved into an iterative method called Iterative Fast Gradient Sign Method (I-FGSM) [20]. I-FGSM iteratively updates the input image with its gradients calculated in the same way as FGSM. Then, Momentum Iterative Fast Gradient Sign Method (MI-FGSM) [6] achieved better performance by integrating momentum during the iterative updates of I-FGSM.

Afterward, the transfer-based black-box attack methods have emerged. Diverse Input (DI) method [42] increases the transferability of adversarial examples by performing random resizing and random padding to input images at each iteration. Translation-Invariant (TI) method [7] uses multiple translated images to generate an adversarial perturbation, rather than using a single input image. They efficiently approximate this process by applying a convolutional operation with a kernel to the gradient obtained from a single input image without any translation. Scale-Invariant (SI) attack method [24] improves the transferability of adversarial examples by using a scaled copy of the input image to compute the gradient.

[26] showed the existence of a single adversarial perturbation that can fool image classifier models when added to any input images. This single perturbation is called a Universal Adversarial Perturbation (UAP). There are many studies on UAP designed for deep learning models that deal with images [26, 27, 11, 18, 28, 25, 46, 21, 47].

2.3. Video Classification Attacks

There are several methods to create UAPs for video classification models [37, 14, 43, 23]. [23] trains a Generative Adversarial Network (GAN) to generate UAPs, and [43] optimizes a noise generator to create a UAP. [37] and [14] are optimization-based white-box UAPs. In white-box settings, [37] introduced an optimization-based algorithm for generating adversarial perturbations on the whole video, specifically on LSTM-based models. They proposed to regularize to concentrate perturbations on key frames. Similarly, a one-frame attack [14] only adds adversarial noise to one selected video frame. The researchers choose a vulnerable frame and perturbed it using the I-FGSM attack method. Similar to [14], there are other key frame selection attack methods [38, 44, 8] for both white-box and black-box settings.

In black-box settings, there are query-based video classification attacks [16, 22, 48, 41] and transfer-based video classification attacks [39, 40], similar to image classification attacks. [39] introduced a method called TT (Temporal Translation) to enhance the transferability of video adversarial examples. They prevent overfitting the source model by optimizing over a set of video clips that have been translated in time for each video. I2V (Images to Videos) method [40] achieved better transferability without relying on video models. I2V minimizes the similarity between the features of the original video frames and the adversarial video frames obtained by the ImageNet pre-trained image model. These perturbations optimized with the image model applied to the videos to attack video models. Both previous works (TT and I2V) have significantly improved transferability, but they have the limitation of requiring optimization for each individual video, which is not the case for UAP.
3. Methodology

In this section, we describe the Breaking Temporal Consistency method for generating the BTC-UAP using an image classification model that takes images or video frames as input. This approach does not require any prior knowledge about the target video data or model and can fool the video model into producing an incorrect prediction.

3.1. Problem Definition

We consider a video $V \in \mathbb{R}^{T \times C \times H \times W}$ and aim to generate an adversarial video $V^{adv}$ by adding a BTC-UAP $\delta_N \in \mathbb{R}^{N \times C \times H \times W}$ to $V$. Here, $T$, $C$, $H$, $W$, and $N$ denote the frames of the video, channels, height, width, and frames of the UAP, respectively. To represent each frame of the $\delta_N$, we use $\delta_n \in \mathbb{R}^{C \times H \times W}$, where $n = 1, ..., N$ is the frame index. To ensure the imperceptibility of the perturbation, we constrain $\delta_N$ to have an $l_\infty$-norm, as in previous works [40, 14].

The value of $N$ is either less than or equal to $T$, and if $N < T$, we repeat the UAP in the frame dimension until it covers all $T$ frames of the video. We define the repeated UAPs as $\delta^T \in \mathbb{R}^{T \times C \times H \times W}$, where $\delta^T = \{\delta^1_T, ..., \delta^T_T\}$ is obtained by repeating the original UAP $\delta_N = \{\delta^1_N, ..., \delta^N_N\}$ in the frame dimension until it covers entire $T$ frames of the video. We can represent this operation as follows:

$$\delta^T = \text{Repeat}(\delta_N) = \{\delta^1_N, ..., \delta^N_N, \delta^1_N, ..., \delta^N_N, ..., \}.$$  

(1)

Let $g(\cdot)$ be a video recognition model, and $y$ be the true label of $V$. Our goal is to find a perturbation $\delta_N$ that misleads the video model’s prediction:

$$g(V + \delta_N) \neq y, \quad \text{s.t.} \quad \|\delta_N\|_\infty \leq \epsilon.$$  

(2)

To achieve this, we optimize $\delta_N$ with $f(\cdot)$, which represents a image classification model.

3.2. Feature Similarity Analysis of Video Frames

In this section, we measure the average similarity of features obtained between video frames in the dataset. Since feature maps represent characteristics of an image, we use them to compare the similarity between frames. Figure 3 represents the feature similarity between two frames of the videos. For example, the diagonal represents the similarity between identical frames, so it always has the value of 1. To obtain the value, we input each frame of the video to an image model and measured the similarity $Sim$ at a specific feature level using cosine similarity. The similarity between vectors $x_1$ and $x_2$ is expressed as follows:

$$Sim(x_1, x_2) = \frac{x_1 \cdot x_2}{\|x_1\| \|x_2\|}.$$  

(3)

To represent each frame of the video, we use $V^t \in \mathbb{R}^{C \times H \times W}$, where $t \in T$ is the frame index. We extract the feature map $F(\cdot)$ from a specific layer $l$ of an image classification model $f(V^t)$ and denote this feature map by $F_l(V^t)$. We visualize the similarity of frames within an original video $V$ in Figure 3-(a). We observe that the original videos tend to have high levels of similarity between consecutive frames.

Furthermore, we extend the non-UAP I2V method [40] to create an I2V-UAP. To make I2V universal, we optimize one perturbation for multiple videos within the dataset. To observe the effects of UAPs on the feature maps, we create an adversarial example $V^t_{adv}$ by adding the UAP $\delta^T_T$ to the original frame $V^t$ and extract the feature maps $F_l(V^t_{adv})$ in the same way as for the original frame. Applying the I2V-UAP shown in Figure 3-(b) results in a reduction in similarity across all frames.

We further observe that adversarial videos disrupt the high similarity pattern of consecutive frames in the original videos. Based on this observation, we propose the BTC method to generate adversarial videos with opposite patterns to the original videos. Details of our method can be found in Section 3.3. Our proposed BTC-UAP, as shown in Figure 3-(c), generates a completely opposite pattern of...
similarity to the original video, with neighboring frames having low similarity. As we intentionally make the features of consecutive frames less similar to each other, the overall similarity between frames decreases when compared to the original video. These results indicate that neighboring frames are recognized as different images by image models. These effects are contrary to the original characteristics of the video, and our experiments in Section 4 demonstrates that BTC-UAP effectively confuses video models.

3.3. Breaking Temporal Consistency Method

In this section, we focus on Breaking Temporal Consistency method and discuss how to optimize the BTC-UAP using image data and models. Let \( x \in \mathbb{R}^{C \times H \times W} \) be an image, which can be a frame of a video \( V^t \in \mathbb{R}^{C \times H \times W} \). Our goal is to find a universal adversarial perturbation \( \delta_N \in \mathbb{R} \) using images. The overall optimization process is described in Algorithm 1.

**Adversarial Loss.** Feature maps represent characteristics and patterns of an image, which can be used to create adversarial examples. Therefore, decreasing the similarity between the feature representations \( F(x) \) of original images and adversarial examples \( x_{adv} = x + \delta_N \) will result in the UAP causing confusion in the information of the original image. To ensure that the BTC-UAP is effective against other data and prevent overfitting to the training dataset, we propose Feature Diversity method with a total of \( K \) random noises. This involves adding a random noise \( \eta_k \in [-\epsilon, \epsilon]^{C \times H \times W} \) to each \( x \) to increase diversity to avoid overfitting. This simple method is highly effective in improving the performance of the UAP framework. The adversarial loss can be expressed mathematically as follows:

\[
L_{adv}(x, n, \delta_N) = \sum_{k=1}^{K} \text{Sim}(F(x + \eta_k), F(x_{adv}^n)).
\]  

(4)

**Temporal Similarity Loss.** Our approach presents a novel solution to the issue that image models are unable to fully consider the temporal dimension, in contrast to video models. Our goal is to minimize the similarity between neighboring frames in videos using the optimized \( \delta_N \). To successfully deceive a video model, we introduce confusion in the temporal domain through the use of \( f(\cdot) \), by decreasing similarity between the neighboring frames. To achieve this, we generate the adversarial image \( x_{adv} = x + \delta_N \) and then treat the sequence of adversarial images as a pseudo video. Here, \( j \in J \) and \( J \) represents the set of temporal distances of neighbors, such as \( J = \{-2, -1, 1, 2\} \).

To reduce the similarity between \( x_{adv}^n \) and \( x_{adv}^{n+j} \), we extract feature of adversarial images \( F(x_{adv}) \) using the image model \( f(\cdot) \) and calculate the similarity between them, following Eq.3. The temporal similarity loss can effectively cause confusion in the temporal information when the perturbations \( \delta_N \) is added to video along the temporal axis. This temporal similarity loss can be expressed mathematically as follows:

\[
L_{temp}(x, n, \delta_N) = \sum_{j \in J} \text{Sim}(F((x_{adv}^n), F(x_{adv}^{n+j}))).
\]  

(5)

Compared to previous approaches, our method allows us to effectively minimize the temporal similarity between perturbed frames, enabling us to produce more robust adversarial examples.

By considering both adversarial and temporal aspects using image-based approaches, the proposed BTC-UAP can effectively perturb both types of information and successfully attack video models. To optimize \( \delta_N \), we utilized a Breaking Temporal Consistency Loss that is the sum of the adversarial loss and temporal similarity loss, mathematically represented as follows:

\[
L_{BTC}(x, n, \delta_N) = L_{adv} + L_{temp}.
\]  

(6)

Finally, we can get optimized BTC-UAP \( \delta_N^{opt} \) by minimizing BTC-Loss with \( I, J, K \) and \( f \):

\[
\delta_N^{opt} = \arg \min \delta_N L_{BTC}(x, n, \delta_N).
\]  

(7)

4. Experiment

4.1. Experiment Settings

We evaluate the Attack Success Rates (ASR) of UAPs in the following settings. The ASR indicates the rate at which the target model misclassifies the adversarial examples into the wrong label. A higher ASR indicates that the UAPs achieve higher transferability.

**Datasets.** We refer to the data used to generate UAPs as the source data, and the data where the UAP is added to create adversarial examples as the target data. We conducted experiments using various datasets. ImageNet [4] is a large image dataset with 1,000 classes. We used the ImageNet train set as source data, selecting 10 images per one class. UCF-101 [32] and Kinetics-400 [17] are video classification datasets that label human action categories. UCF-101 has 13,320 videos with 101 action classes, and Kinetics-400 has 650,000 videos with 400 classes. We used the UCF-101

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SF-101</th>
<th>SF-50</th>
<th>TPN-50</th>
<th>TPN-101</th>
<th>NL-50</th>
<th>NL-101</th>
<th>AVG.</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCF-101</td>
<td>90.2</td>
<td>91.7</td>
<td>91.7</td>
<td>93.6</td>
<td>86.9</td>
<td>88.4</td>
<td>90.4</td>
</tr>
<tr>
<td>Kinetics</td>
<td>69.8</td>
<td>71.0</td>
<td>73.9</td>
<td>75.0</td>
<td>69.5</td>
<td>65.9</td>
<td>71.4</td>
</tr>
</tbody>
</table>

Table 1: Clean Accuracy
<table>
<thead>
<tr>
<th>UCF-101</th>
<th>SF-101</th>
<th>All-UAP</th>
<th>76.12</th>
<th>98.59</th>
<th>49.76</th>
<th>45.90</th>
<th>48.41</th>
<th>48.76</th>
<th>61.26</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCF-101</td>
<td>TT-UAP</td>
<td>75.92</td>
<td>98.74</td>
<td>66.59</td>
<td>58.97</td>
<td>62.55</td>
<td>62.55</td>
<td>70.88</td>
<td></td>
</tr>
<tr>
<td>UCF-101</td>
<td>TPN-101</td>
<td>All-UAP</td>
<td>58.81</td>
<td>58.29</td>
<td>62.91</td>
<td>79.95</td>
<td>51.29</td>
<td>48.49</td>
<td>59.95</td>
</tr>
<tr>
<td>UCF-101</td>
<td>TT-UAP</td>
<td>53.99</td>
<td>53.16</td>
<td>48.85</td>
<td>44.68</td>
<td>43.86</td>
<td>41.62</td>
<td>47.69</td>
<td></td>
</tr>
<tr>
<td>UCF-101</td>
<td>NL-101</td>
<td>All-UAP</td>
<td>59.69</td>
<td>60.21</td>
<td>60.60</td>
<td>56.20</td>
<td>67.41</td>
<td>80.52</td>
<td>64.10</td>
</tr>
<tr>
<td>UCF-101</td>
<td>TT-UAP</td>
<td>56.69</td>
<td>56.89</td>
<td>53.13</td>
<td>49.75</td>
<td>50.21</td>
<td>47.69</td>
<td>55.30</td>
<td></td>
</tr>
<tr>
<td>ImageNet</td>
<td>Res-101</td>
<td>BTC-UAP</td>
<td>82.49</td>
<td>77.30</td>
<td>93.35</td>
<td>86.62</td>
<td>67.40</td>
<td>67.83</td>
<td>79.16</td>
</tr>
<tr>
<td>ImageNet</td>
<td>BTC-UAP</td>
<td>83.26</td>
<td>77.63</td>
<td>93.49</td>
<td>86.23</td>
<td>68.27</td>
<td>66.97</td>
<td>79.31</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Comparison with UAPs generated on video models. UAPs are optimized on the source datasets UCF-101 and ImageNet, respectively. The generated UAPs are repeated and added to Kinetics-400 videos until they cover the entire video. The bold numbers indicate the highest attack success rates (%) in each column. The gray color represents the white-box setting, where the source and target models are identical.

<table>
<thead>
<tr>
<th>UCF-101</th>
<th>SF-101</th>
<th>All-UAP</th>
<th>76.12</th>
<th>98.59</th>
<th>49.76</th>
<th>45.90</th>
<th>48.41</th>
<th>48.76</th>
<th>61.26</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCF-101</td>
<td>TT-UAP</td>
<td>75.92</td>
<td>98.74</td>
<td>66.59</td>
<td>58.97</td>
<td>62.55</td>
<td>62.55</td>
<td>70.88</td>
<td></td>
</tr>
<tr>
<td>UCF-101</td>
<td>TPN-101</td>
<td>All-UAP</td>
<td>58.81</td>
<td>58.29</td>
<td>62.91</td>
<td>79.95</td>
<td>51.29</td>
<td>48.49</td>
<td>59.95</td>
</tr>
<tr>
<td>UCF-101</td>
<td>TT-UAP</td>
<td>53.99</td>
<td>53.16</td>
<td>48.85</td>
<td>44.68</td>
<td>43.86</td>
<td>41.62</td>
<td>47.69</td>
<td></td>
</tr>
<tr>
<td>UCF-101</td>
<td>NL-101</td>
<td>All-UAP</td>
<td>59.69</td>
<td>60.21</td>
<td>60.60</td>
<td>56.20</td>
<td>67.41</td>
<td>80.52</td>
<td>64.10</td>
</tr>
<tr>
<td>UCF-101</td>
<td>TT-UAP</td>
<td>56.69</td>
<td>56.89</td>
<td>53.13</td>
<td>49.75</td>
<td>50.21</td>
<td>47.69</td>
<td>55.30</td>
<td></td>
</tr>
<tr>
<td>ImageNet</td>
<td>Res-101</td>
<td>BTC-UAP</td>
<td>82.49</td>
<td>77.30</td>
<td>93.35</td>
<td>86.62</td>
<td>67.40</td>
<td>67.83</td>
<td>79.16</td>
</tr>
<tr>
<td>ImageNet</td>
<td>BTC-UAP</td>
<td>83.26</td>
<td>77.63</td>
<td>93.49</td>
<td>86.23</td>
<td>68.27</td>
<td>66.97</td>
<td>79.31</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Comparison with UAPs generated on video models. UAPs are optimized on the source datasets UCF-101 and ImageNet, respectively. Adversarial videos are generated by adding UAPs to UCF-101 videos. The bold numbers indicate the highest attack success rates (%) in each column for the UCF-101 dataset. The gray color represents the white-box setting, where the source model and target model are identical.

The perturbation budget $\epsilon$ was set to 16/255, and the step size $\alpha$ was set to 0.004. We used the number of feature layers $l$ to optimize BTC-UAP and I2V-UAP as follows: BTC-UAP optimization, and set the number of UAP frames $N$ to 32. The number of random noise $K$ was set to 4 and the set of temporal distance $J$ to $\{−2, −1, +1, +2\}$.

In Section 4.4, we show how we selected the hyper-parameters for BTC-UAP. The implementation details for
Table 4: Attack success rates (%) of UAPs generated on image models using image data. UAPs are optimized on ImageNet and adversarial videos are generated by adding UAPs to Kinetics-400 videos. The generated UAPs are repeated and added to Kinetics-400 videos until they cover the entire video. The bold numbers indicate the highest attack success rate among attack methods.

<table>
<thead>
<tr>
<th>Source Models</th>
<th>Attack</th>
<th>SF-50 (Kinetics)</th>
<th>SF-101 (Kinetics)</th>
<th>Target Models</th>
<th>TPN-50 (Kinetics)</th>
<th>TPN-101 (Kinetics)</th>
<th>NL-50 (Kinetics)</th>
<th>NL-101 (Kinetics)</th>
<th>AVG.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res-101 (ImageNet)</td>
<td>MI-UAP</td>
<td>64.42</td>
<td>57.46</td>
<td>58.02</td>
<td>54.43</td>
<td>49.36</td>
<td>47.37</td>
<td>54.36</td>
<td></td>
</tr>
<tr>
<td>MI-UAP</td>
<td>60.55</td>
<td>56.63</td>
<td>59.26</td>
<td>58.66</td>
<td>51.64</td>
<td>48.91</td>
<td>55.94</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TI-UAP</td>
<td>60.56</td>
<td>56.63</td>
<td>59.26</td>
<td>58.64</td>
<td>56.57</td>
<td>54.41</td>
<td>57.68</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SI-UAP</td>
<td>65.98</td>
<td>64.61</td>
<td>71.26</td>
<td>70.13</td>
<td>55.72</td>
<td>55.30</td>
<td>63.83</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2V-UAP</td>
<td>69.31</td>
<td>64.37</td>
<td>83.85</td>
<td>73.64</td>
<td>53.26</td>
<td>50.97</td>
<td>65.90</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BTC-UAP</td>
<td>83.26</td>
<td>77.63</td>
<td>93.49</td>
<td>86.23</td>
<td>68.27</td>
<td>66.97</td>
<td>79.31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VGG16 (ImageNet)</td>
<td>MI-UAP</td>
<td>54.42</td>
<td>52.56</td>
<td>49.53</td>
<td>45.43</td>
<td>45.64</td>
<td>43.44</td>
<td>49.11</td>
<td></td>
</tr>
<tr>
<td>MI-UAP</td>
<td>57.96</td>
<td>55.47</td>
<td>51.97</td>
<td>48.22</td>
<td>49.25</td>
<td>45.96</td>
<td>51.47</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TI-UAP</td>
<td>57.95</td>
<td>55.45</td>
<td>51.95</td>
<td>48.21</td>
<td>55.51</td>
<td>52.63</td>
<td>53.61</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SI-UAP</td>
<td>59.63</td>
<td>58.29</td>
<td>56.22</td>
<td>51.25</td>
<td>50.19</td>
<td>48.67</td>
<td>54.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2V-UAP</td>
<td>57.82</td>
<td>52.73</td>
<td>60.40</td>
<td>54.90</td>
<td>44.62</td>
<td>41.92</td>
<td>52.06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BTC-UAP</td>
<td>74.25</td>
<td>75.05</td>
<td>82.97</td>
<td>75.68</td>
<td>68.59</td>
<td>63.93</td>
<td>73.41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Squeeze (ImageNet)</td>
<td>MI-UAP</td>
<td>53.91</td>
<td>52.92</td>
<td>52.22</td>
<td>49.55</td>
<td>45.64</td>
<td>43.44</td>
<td>49.61</td>
<td></td>
</tr>
<tr>
<td>MI-UAP</td>
<td>54.61</td>
<td>53.42</td>
<td>52.22</td>
<td>49.24</td>
<td>45.24</td>
<td>43.61</td>
<td>49.72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TI-UAP</td>
<td>66.25</td>
<td>66.05</td>
<td>59.55</td>
<td>54.79</td>
<td>57.52</td>
<td>55.06</td>
<td>59.87</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SI-UAP</td>
<td>58.36</td>
<td>58.40</td>
<td>54.51</td>
<td>49.86</td>
<td>47.70</td>
<td>45.70</td>
<td>52.42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2V-UAP</td>
<td>66.20</td>
<td>63.40</td>
<td>66.50</td>
<td>58.94</td>
<td>54.30</td>
<td>50.60</td>
<td>59.99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BTC-UAP</td>
<td>70.71</td>
<td>68.14</td>
<td>71.29</td>
<td>65.25</td>
<td>58.88</td>
<td>54.72</td>
<td>64.83</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.2. Experimental Results

4.2.1 Comparison with Video-based attack method

We evaluated the transferability of UAPs optimized on UCF-101 in Tables 2 and 3. Table 2 shows the performance of UAPs on each target model trained on Kinetics-400, evaluated by adding the UAPs to Kinetics-400 videos. In Table 3, we divided the UCF-101 dataset into two groups and evaluated methods on the unseen group. The gray color in the tables represents the white-box setting, where the target model is used as the source model during UAP generation. Please note that our method aims to transfer the UAPs generated from image models to video models for cross-modal attacks, which cannot be conducted under the white-box setting. Excluding the white-box evaluation, BTC-UAP achieves the highest transferability in most cases. For example, in Table 2, BTC-UAP (Res-101, ImageNet) achieved the highest average ASR of 79.31%, compared to All-UAP(SF-101, UCF-101) with 61.26% and TT-UAP(SF-101, UCF-101) with 70.88%.

When compared to UAPs optimized using videos as the source data, the performance of BTC-UAP generated on image data is comparable or even better. This demonstrates that our Breaking Temporal Consistency method can effectively consider temporal information, even without video models or data, and achieve superior performance compared to I2V-UAP. Furthermore, in Table 2, the generated UAP was optimized for 32 frames, while the evaluation on Kinetics-400 was conducted on 64 frames. Therefore, we repeated the UAP without optimizing it for 64 frames to generate universal adversarial perturbations for Kinetics-400, following Eq.1. Despite the challenging condition of evaluating on 64 frames while the generated UAP was optimized for 32 frames, BTC-UAP still achieves high transferability in attacking video classification models. This demonstrates that our method is effective even in complete black-box situations, such as evaluating on an unseen video model with a different number of video frames.

4.2.2 Comparison with Image-based attack method

We conducted experiments to evaluate the transferability of UAPs generated using image data and models. Table 4 shows the ASR of adversarial videos, where UAP is optimized on ImageNet using each rightmost image model. In this experiment, the 32-frame UAPs are repeatedly added to Kinetics-400 videos to create adversarial videos, following Eq.1. Compared to other methods, BTC-UAP achieved the highest average ASR and demonstrated good transferability. For instance, in Table 4, the I2V-UAP has a total average ASR 60.31% on all cases, while BTC-UAP shows superior performance with 70.79%. This result demonstrates that our proposed method effectively considers temporal information, resulting in the highest performance among image-based methods.

other baselines are in the supplementary material.
4.3. Discussion

In this section, we conducted experiments to demonstrate the effectiveness of our proposed Breaking Temporal Consistency Loss for creating adversarial examples in video-based attacks. We applied the BTC-UAP generated using Res-101 and ImageNet data, to 32-frame UCF-101 videos and analyzed its effectiveness on six different models.

To analyze the performance of our proposed method, we compared the cosine similarity between the original and adversarial videos in the video model. Figure 4-(a) shows cosine similarity scores between the original and adversarial videos for each attack method under black-box settings. The bottom graph shows the same comparison under white-box settings. When all attack settings are black-box, our proposed method achieved the lowest similarity score among all the attack methods. In the context of confusing video models, we found that BTC(I), which is generated using image data, is more effective than BTC(V), which is generated using video data. The results show that BTC(I) had a greater impact on the UAPs generated with video models despite being generated using image models, highlighting its superior robustness.

To demonstrate the effectiveness of the proposed method with a small number of BTC-UAP frames, we applied the UAP iteratively with a small value of N, repeating a subset of N frames within the total of T = 32 frames in the adversarial video. We compared the results for N=2,4,8,12,16 and 32. Figure 4-(b) compares the performance of BTC-UAP with different numbers of N. Even when N=2, our proposed method exhibits comparable performance, demonstrating its effectiveness even with a small number of UAP frames. We further demonstrated the shifting invariance of our proposed BTC-UAP by conducting experiments in which we shifted the UAP along the temporal axis from 1 to 8 frames. Figure 4-(c) demonstrates the shifting invariance of BTC-UAP by displaying attack success rates for different temporal shifts of the UAP frames. It showed that the attack success rate was consistent regardless of the temporal shifting. These results demonstrated the BTC-UAP is robustness against temporal shifts and the effectiveness even with a small number of optimized frames.

4.4. Ablation study

In this section, we explore the effects of the most critical parameters, K and J, in our BTC-method. Specifically, we investigate the impact of the number of random noise K employed in the adversarial loss and the temporal distance of neighbors set J utilized in the temporal similarity loss. Figure 4-(d) shows that K = 4 provided the best performance in terms of the adversarial loss. We then conducted experiments with different symmetric sets of J while keeping K fixed at 4. In the graph, please note that we represented the highest value among the set of J on the y-axis for convenience. Our results showed that when the max(J) = 2, the use of a set J = {−2, −1, 1, 2} achieved the highest performance.

Importantly, we observed that although the computations required for K = 6 and K = 4 with J = {−1, 1} were the same, the latter yielded significantly better performance. This demonstrates that reducing the similarity between frames was a more effective approach to improving performance than simply increasing computational resources.

5. Conclusion

In this paper, we proposed the Breaking Temporal Consistency Method, which was the first to attack videos using only image models while considering temporal information. Our method was designed to minimize the similarity between neighboring frames, by jointly optimizing adversarial and temporal similarity losses. Specifically, by using adversarial loss, we reduced the similarity between original and adversarial examples, and by using temporal similarity loss, we reduced the similarity between UAPs. BTC-UAP was both temporal shift invariant and length-agnostic. Our extensive experiments on various datasets demonstrated the effectiveness of our proposed BTC-UAP.
References


