VideoFlip: Adversarial Bit Flips for Reducing Video Service Quality

Jung-Woo Chang Mojan Javaheripi Farinaz Koushanfar
University of California San Diego, La Jolla, CA, USA
{juc023, mojan, farinaz}@ucsd.edu

2023 60th ACM/IEEE Design Automation Conference (DAC) | 979-8-3503-2348-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/DAC56929.2023.10247863

***Abstract*—Video compression is a critical component in streaming applications that maximizes users' quality of experience (QoE) while satisfying the underlying bandwidth constraints. In this paper, we demonstrate the first hardware-based fault-injection attack on video compression, dubbed VideoFlip. The attack identifies and alters a few bits inside the video compression model to achieve one or more of the following adversarial goals: (1) increasing the bitrate and (2) degrading the quality of the compressed video. Extensive evaluations show VideoFlip irresistibly downgrades video compression performance for various models and video datasets while being resilient to contemporary defenses against hardware-based bit flips.**

I. Introduction

Video compression plays a significant role in minimizing redundancy for video content delivery. Several recent video compression frameworks have been built upon deep learning (DL) algorithms [1]–[3]. Due to their superior performance compared with traditional video codecs [4], [5], DL-based video compression has been explored by the Moving Picture Experts Group (MPEG) for adoption in the next-generation video delivery systems [6]. In addition, industry-led efforts [7] have developed custom accelerators for DL-based video compression on mobile processors which use low-bit quantization for further efficiency gains.

Although DL-based video compression is prevalent in many streaming services, its robustness against fault-injection attacks remains largely unknown. Several studies have shown that deep neural networks (DNNs) used for image classification are vulnerable to hardware-based faults [8]–[12], where the accuracy drops significantly after flipping a few weight bits stored in the DRAM via row hammering. In this work, we propose VideoFlip, the first hardware-based threat to video compression systems that can severely downgrade the end-users' quality of experience in video delivery systems. Our attack poses an important open problem in the reliability of DL-based video compression. As an example, VideoFlip can be used by phone hardware companies to favor certain third-party video delivery systems and cause denial-of-service for others.

VideoFlip progressively finds a set of vulnerable weight bits inside video compression modules that, when altered in the DRAM, hinder video delivery. Towards this goal, VideoFlip proposes an adaptive search to find attack target bits that simultaneously satisfy two criteria, namely, physical viability and attack success. Physical viability ensures that attacked bits can be flipped via row hammer attack to the DRAM. VideoFlip is equipped with a memory templating module to determine vulnerable bits from the hardware perspective. Attack success is achieved by finding bits that maximize a custom objective function, devised based on the utility of the video compression model and adversarial goals.

We formulate the adversarial objective function for three novel attacks that break the optimized balance between bitrate and video distortion as shown in Fig. 1. Our *distortion attack* (*DA*) considerably degrades the peak signal-to-noise ratio (PSNR) of the compressed video at any given bitrate. In contrast, our *bitrate attack* (*BA*) substantially increases the bitrate for any target distortion, thereby inducing video lags or rebuffering that cause denial-of-service in real-time streaming services, e.g., remote vehicle control. Finally, we propose a *bitrate and distortion attack* (*BDA*) which simultaneously degrades the PSNR and bitrate.

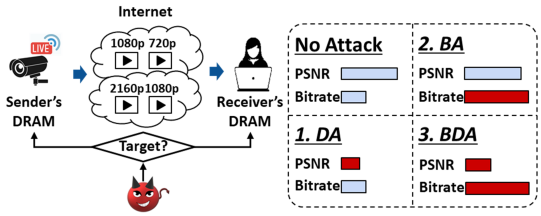


Fig. 1: VideoFlip attack scenarios on Internet video delivery systems. To show the adversarial effect of our attacks, we visualize the bitrate and peak signal-to-noise ratio (PSNR) as a measure for distortion.

In designing our attack objective functions, we face several challenges that are not present in the image domain. Firstly, contemporary video compression methods process frames in groups called Group of Pictures (GOP) which creates a high inter-frame dependency that renders attacks to a single frame unsuccessful. We incorporate the dependencies between adjacent frames in our attack objective formulation and find bits that transform the bitrate-distortion relationship of all frames in the GOP. Secondly, the underlying compression rate of video codecs can change abruptly depending on the available network bandwidth. We, therefore, formulate our (offline) attack to be universally applicable to different compression rates for real-time attack to video delivery systems.

In summary, our key contributions are as follows:

- Demonstrating the first nefarious hardware-based fault-injection attack on DL-based video compression models. Our attacks are devised based on algorithm-hardware codesign to ensure feasible degradation of video compression quality.
- Formulating the attack as a search over weight bits to maximize three well-defined objective functions that can selectively compromise the bitrate and/or video distortion depending on the adversary's goal.
- Capturing the relationship between weight bits and compression performance such that the attacked bits can universally downgrade performance for any incoming video stream, irrespective of the content or compression rate.
- Conducting proof-of-concept experiments to show the effect of VideoFlip attacks on state-of-the-art video compression models. We further show VideoFlip withstands contemporary defenses against bit flip attack on DNNs. Thus, designing an effective defense for such a clandestine attack remains an open problem.

II. Background and Prior Work

A. Deep Learning-based Video Compression

DL-based video compression models take as input a video sequence $X = \{x_1, ..., x_T\} \in \mathbb{R}^{T \times W \times H \times C}$ with T frames and return reconstructed frames $Y = \{y_1, ..., y_T\} \in \mathbb{R}^{T \times W \times H \times C}$ by passing X through video encoder and decoder models. Here, x_t and y_t denote the input and output frames at time step t with H rows, W columns, and C color channels. Depending on the underlying temporal coding structure k used in the video compression model, each coded frame is mapped to a unique time step in the input video. Specifically, for a given k, the n-th coding order is mapped to the t-th display order by a deterministic function $m_k(n) = t$. Here, $1 \leq n \leq G$ where G is the number of frames in the GOP.

Contemporary DL-based video compression methods have different pipelines for the video sender and receiver. Specifically, the sender has both a video encoder and a decoder component to encode the current raw frame from previously decoded frames according to the temporal coding structure. The receiver, however, only has a video decoder module to restore the video sequences from the encoded video bitstream.

Video Encoder first estimates motion vectors between the current frame and the previously decoded frame. It then creates a reconstructed frame from the motion vectors and previously decoded frames. The encoder further creates a residual frame from the restored frame and the current frame. The residual frame and motion vectors are both compressed into a latent space using an auto-encoder network and converted into bitstreams via entropy coding to be transmitted to the receiver.

Let us denote by $\mathbf{\Theta} = \{\Theta_l\}_{l=1}^{L_e}$ the bit tensors containing (quantized) weights of the encoder. Here, Θ_l is the l-th layer weights and L_e is the total number of layers in the video encoder. We represent the video encoder as a function $\mathbb{E}_k(x_t, \mathcal{P}_t, \lambda; \mathbf{\Theta}) = z_t$ that compresses the current frame x_t into a bitstream z_t according to a compression rate λ and previously decoded frames $\mathcal{P}_t$. Here, $\mathcal{P}_t = \{y_{m_k(1)}, ..., y_{m_k(n-1)}\}$ where $y_{(\cdot)}$ is the output of the decoder at a certain time step. $\mathcal{P}_1 = \emptyset$ since the first frame, dubbed the I-frame, is coded independently using DL-based image compression [13].

Video Decoder recovers the latent representation via entropy decoding, then applies auto-encoder networks to obtain the reconstructed motion vectors and residual frame. Finally, the decoded frame is created by passing the reconstructed motion vectors and previously decoded frames through a motion compensation module and adding the reconstructed residual frame. We represent the bit tensors for decoder weights as $\mathbf{\Phi} = \{\Phi_l\}_{l=1}^{L_d}$ where L_d is the total number of decoder layers. The video decoder can be formulated as a function $\mathbb{D}_k(z_t, \mathcal{P}_t, \lambda; \mathbf{\Phi}) = y_t$ which restores the decoded frame y_t from the received bitstream z_t using the decoded frame buffer $\mathcal{P}_t$ and the compression rate λ.

B. Bit flip Attack

Algorithm. Recent work shows that changing a few carefully selected bits of DNN weights can lead to severe accuracy degradation on image classification tasks [10]. In this domain, gradient-based bit search [10] is a powerful algorithm to search for vulnerable bits. Bit flips are later used for Trojan attacks [11], [12] where the adversary injects a backdoor in a DNN. We note that prior work on bit flip attacks all target DNNs in image classification. To the best of our knowledge, VideoFlip is the first work that targets video compression frameworks by proposing new formulations of the attack.

Hardware Realization. Rowhammer is a hardware-based fault-injection attack that causes electrical distortion in DRAM cells [14] by frequently activating neighboring rows. Double-sided hammering has been widely used to induce DRAM cell disturbance, and consequently flip the stored bits. Recently, double-sided hammering is used to realize bit flip attacks on DNNs in image classification tasks [8], [9]. The process consists of several stages as follows: first, the adversary identifies the location of weak cells in the DRAM with row hammering. Based on the vulnerable templates, the adversary finds the least amount of bits that can be flipped to subvert the DNN. Advanced memory massaging methods [9] are then used to position the victim pages to vulnerable DRAM rows. Finally, the adversary uses row stripe patterns, e.g., 1-0-1 and 0-1-0, to trigger the desired bit flips.

III. Threat Model

A. Attack Scenarios

We assume the victim uses DL-based video compression for video delivery in a resource-sharing environment [15]. Following state-of-the-art practice [7], video compression modules are quantized to 8 bits to ensure low-cost execution. As shown in Fig. 1, the attacker targets the DRAM device of either the sender or receiver to adversarially alter bits inside the video compression model. We assume the victim models adopt a modulated scheme for adaptive bitrate control [16] to support variable bitrates like standard video codecs.

B. Adversary's Capability

Consistent with previous works [8], [9], we assume the adversary is co-located with the victim system and takes advantage of the memory allocation scheme to perform double-sided row hammer attack on the victim model. We further assume the architecture and weights of the video compression model are known to adversary. This is a realistic assumption as video compression models are standardized by various organizations, e.g., ISO/IEC and MPEG, and are open-source to the community. To reduce the cost of row hammer attack, the adversary aims to minimize the number of bit flips required to downgrade the performance. The attacker thus uses one GOP-sized video to learn the most destructive adversarial bit flips. Note that this video can be arbitrary and does not need to belong to the training dataset of the video compression model.

C. Adversary's Goal

Video compression models are trained to minimize video distortion *D* at a given bitrate *R*. By altering the weights in the pretrained video compression model, the adversary aims to break the balance between *R* and *D*. When the sender is attacked, the adversary can selectively increase the bitrate and/or distortion. When the receiver is targeted, the adversary can only deteriorate the video quality. Based on the adversarial goals, we formulate the following attacks:

Distortion Attack (*DA*) deteriorates the quality of the reconstructed video by increasing *D* while maintaining *R*. Due to the constant

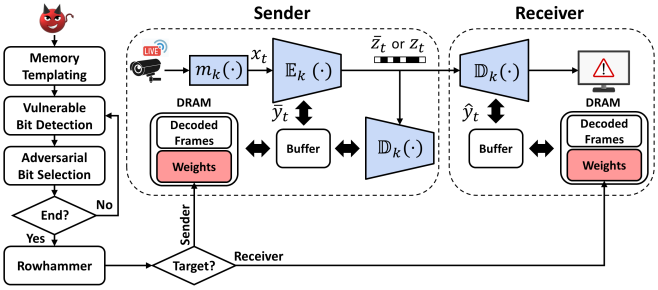


Fig. 2: Overview of the proposed VideoFlip attack.

bitrate, this attack remains stealthy even if the media administrator uses network traffic anomaly detection.

Bitrate Attack (*BA*) increases R for any given distortion level. Thus, the size of the video that needs to be downloaded by the receiver increases dramatically. *BA* causes video buffering or a decrease in video resolution at a fixed bitrate budget.

Bitrate and Distortion Attack (*BDA*) simultaneously increases bitrate and video distortion. Therefore, the receivers suffer from poor video quality or denial-of-service.

IV. VideoFlip Methodology

Fig. 2 shows a high-level flow of various steps performed in VideoFlip attacks. VideoFlip attack methodology is devised based on hardware-algorithm codesign to ensure applicability in real-world scenarios. Specifically, when determining vulnerable bits for the attack, VideoFlip simultaneously addresses two key design goals, i.e., attack success and physical viability. The first criterion, i.e., attack success, is tied to our algorithmic contributions which find a small number of vulnerable bits in the video compression model that, if altered, can lead to severe performance degradation. We devise two interlinked steps to identify target bit flips, namely, *vulnerable bit detection* and *adversarial bit selection*. The second criterion, i.e., physical viability, ensures that the located bit flips by the attack algorithm are in fact realizable inside the hardware via row hammering. This is performed via a *memory templating* step in VideoFlip framework.

As shown in Fig. 2, the attacker first performs memory templating to acquire information about viable bit flips. The attacker then iteratively repeats the vulnerable bit detection and adversarial bit selection steps to find a set of target bit offsets. This process continues until the performance of the video compression is degraded to the attacker's desired threshold. The adversary then performs memory massaging [8], [9] to place the target victim pages on the vulnerable templates. Finally, the attacker triggers the desired bit flips at the neighboring locations via double-sided row hammering. In what follows, we first introduce our attack notations. We then explain each step in our attack framework in detail.

A. Attack Notations

Depending on the adversary's target, we use the following notations for our attacks as illustrated in Fig. 2.

Attack on Sender alters a few bits in the sender's DRAM using row hammer to obtain adversarial weights $(\bar{\boldsymbol{\Theta}}, \bar{\boldsymbol{\Phi}})$. As a result, the sender creates an attacked bitstream $\bar{z}_t = \mathbb{E}_k(x_t, \bar{\mathcal{P}}_t, \lambda; \bar{\boldsymbol{\Theta}})$ for each input frame. Note that the decoded frame buffer $\bar{\mathcal{P}}_t$ on the sender side is also adversarial as it is filled with perturbed decoded frames $\bar{y}_t = \mathbb{D}_k(\bar{z}_t, \bar{\mathcal{P}}_t, \lambda; \bar{\boldsymbol{\Phi}})$. Upon receiving the attacked bitstream from the sender, the receiver reconstructs a perturbed frame $\hat{y}_t = \mathbb{D}_k(\bar{z}_t, \hat{\mathcal{P}}_t, \lambda; \boldsymbol{\Phi})$, where $\hat{\mathcal{P}}_t = \{\hat{y}_{m_k(1)}, \ldots, \hat{y}_{m_k(n-1)}\}$ is the attacked decoded frame buffer on the receiver side.

Attack on Receiver alters the tensors stored in the receiver's DRAM to create an adversarial decoder weight $\bar{\boldsymbol{\Phi}}$. Thus even though the bitstream received from the sender is benign in this scenario, the receiver still reconstructs a perturbed frame $\hat{y}_t = \mathbb{D}_k(z_t, \hat{\mathcal{P}}_t, \lambda; \bar{\boldsymbol{\Phi}})$, where $\hat{\mathcal{P}}_t = \{\hat{y}_{m_k(1)}, \ldots, \hat{y}_{m_k(n-1)}\}$ is the adversarial decoded frame buffer.

B. Memory Templating

During this step, the adversary scans the DRAM to find the vulnerable bit locations. In our system, video compression models are loaded into a DRAM that is divided into multiple pages with a size of 4KB. Each weight bit thus has an offset $\in [0, 32768]$ in the corresponding page. We use the well-known memory profiling tool in [17] to find vulnerable memory addresses in the DRAM. This tool generates a DRAM bit-flip profile that includes a list of vulnerable pages, bit offsets, and their corresponding flip direction $(0 \rightarrow 1,\ 1 \rightarrow 0)$. We leverage this information in the next steps of our attack pipeline to determine the final attacked bits.

C. Vulnerable Bit Detection

By definition, vulnerable weights bits in the model are those that cause the largest performance degradation when altered. Measuring the performance degradation for each bit flip requires running inference on the video compression model, which incurs a non-negligible overhead. It is infeasible to brute-force all possible bit flips in contemporary video compression models which contain hundreds of millions of parameters. We, therefore, propose a profiling step that can successfully locate the per-layer vulnerable bits for a given video compression model with only one round of inference.

We define a custom loss $\mathcal{L}$ for performance degradation which directly reflects the objectives in video compression. Recall from Section III-C that the adversary's goal is to break the balance between bitrate and video distortion. We therefore formulate $\mathcal{L}$ as:

$$\mathcal{L} = \begin{cases} \sum_{\lambda \in \Lambda} \Big[\sum_{t=m_k(1)}^{m_k(G)} \dfrac{R(\bar{z}_t) + \lambda \cdot D(x_t, \hat{y}_t)}{G} \Big] & \text{(Sender attack)} \\ \sum_{\lambda \in \Lambda} \Big[\sum_{t=m_k(1)}^{m_k(G)} \dfrac{R(z_t) + \lambda \cdot D(x_t, \hat{y}_t)}{G} \Big] & \text{(Receiver attack)} \end{cases} \tag{1}$$

where $R(\bar{z}_t)$ is the perturbed bitrate at time step t and $D(x_t, \hat{y}_t)$ is the distortion between the input frame and its perturbed reconstruction by the receiver decoder. Here $\bar{z}_t$ and $\hat{y}_t$ are conditioned on λ.

We highlight two key properties of our formulated loss in Eq. 1. Firstly, unlike the image domain, various frames in a compressed video have temporal dependencies, i.e., the encoding of the current frame is conditioned on previously decoded frames. Due to inter-frame dependencies, any perturbation on the t-th frame affects the compression performance for subsequent frames. To capture this effect in our loss, we perform a summation over all frames in a GOP according to their coding order: $\sum_{t=m_k(1)}^{m_k(G)}(\cdot)$.

Secondly, video players commonly use adaptive bitrate streaming [18] to dynamically control the compression rate according to the

channel condition. As such, the attacker is not aware of the particular compression rate λ used by the video delivery system. We, therefore, ensure that the selected bits by the attacker are globally applicable to various compression rates by performing a summation over commonly used encoding parameters $\Lambda \in \{256,512,1024,2048\}$.

With the performance loss formulation at hand, we can now identify the per-layer vulnerable bits at each iteration of the attack. Specifically, at each iteration i, we search for the most vulnerable bits on top of the prior $(i-1)$ flips such that the attack can successfully manipulate the rate-distortion model in Eq. 1. For each layer l, given the current state of its weight $\bar{W}_l^{i-1}$ at the beginning of the i-th iteration, the most vulnerable bits $\mathbf{b}_l^i$ are identified by solving:

$$\mathbf{b}_l^i = \underset{p}{\text{Top}}\Big(\nabla_{\bar{W}_l^{i-1}} \mathcal{L}\Big) \tag{2}$$

where $\{\text{Top}_p\}$ returns a set of bit offsets for the selected p vulnerable bits stored in the DRAM and $\bar{W}_l$ represents either the encoder ($\bar{\Theta}_l$) or decoder weights ($\bar{\Phi}_l$) depending on layer type. Solving Eq. 2 returns $N_b = p \times (L_e + L_d)$ and $N_b = p \times L_d$ candidate vulnerable bits for the sender and receiver attacks, respectively. Let us denote by $\boldsymbol{B}^i$ the concatenated list of all per-layer vulnerable bits at iteration i. The next step in our framework, i.e., adversarial bit selection, determines the most effective bit in $\boldsymbol{B}^i$ to carry out the attack.

D. Adversarial Bit Selection

Once the vulnerable bits are identified, we perform a search over them to find the most effective bit in achieving the attacker's goal as outlined in Section III-C. At each iteration i, our search algorithm receives the current state of the weights, including any applied flips in past iterations, along with the list of vulnerable bits $\boldsymbol{B}^i$ from Section IV-C. It then flips each bit $b_i \in \boldsymbol{B}^i$ and measures the average bitrate ($\mathcal{R}$) and distortion ($\mathcal{D}$) using Eq. 3. By summing over reconstructed frames with varying λ values, VideoFlip can find bits that are agnostic to the compression ratio. Note that the flipped bit is restored back after each measurement.

$$\mathcal{R}(b_i) = \sum_{\lambda \in \Lambda} \Big[\sum_{t=m_k(1)}^{m_k(G)} \frac{R(\bar{z}_t)}{G} \Big] , \mathcal{D}(b_i) = \sum_{\lambda \in \Lambda} \Big[\sum_{t=m_k(1)}^{m_k(G)} \frac{D(x_t, \hat{y}_t)}{G} \Big] \tag{3}$$

Once the attacker learns the individual effect of vulnerable bits on the video compression bitrate and distortion, they can select the adversarial bit that satisfies their goal as formulated below. To ensure the physical viability of our attack, we check whether a selected bit is flippable or not according to the information obtained from the memory templating step in Section IV-B. If a bit does not meet this condition, the next suitable bit is selected.

DA. The target bit offset b_i^t at iteration i is determined by solving the following optimization problem:

$$\underset{b_i \in \boldsymbol{B}^i}{\text{argmax}}\ \mathcal{D}(b_i) \quad \text{s.t.}\ \frac{|\mathcal{R}(b_i) - \mathcal{R}_o|}{\mathcal{R}_o} < \epsilon_0 \quad \text{and } b_i^t \text{ is flippable.} \tag{4}$$

where $\mathcal{R}_o$ is the bitrate of the original video compression model and ϵ_0 is the tolerance on bitrate change.

BA. The attacker aims to find a target bit offset b_i^t that maximizes bitrate while maintaining the distortion as follows:

$$\underset{b_i \in \boldsymbol{B}^i}{\text{argmax}}\ \mathcal{R}(b_i) \quad \text{s.t.}\ \frac{|\mathcal{D}(b_i) - \mathcal{D}_o|}{\mathcal{D}_o} < \epsilon_1 \quad \text{and } b_i^t \text{ is flippable.} \tag{5}$$

Here, $\mathcal{D}_o$ is the distortion of the original model and ϵ_1 is the upper bound on the allowed distortion change.

BDA. The attacker finds b_i^t by solving the following optimization problem that degrades both bitrate and distortion simultaneously:

$$\underset{b_i \in \boldsymbol{B}^i}{\text{argmax}}\ \mathcal{R}(b_i) + \lambda_{avg} \cdot \mathcal{D}(b_i) \quad \text{and } b_i^t \text{ is flippable.} \tag{6}$$

where λ_{avg} is the average value of $\lambda \in \Lambda$.

V. VideoFlip Evaluation

A. Evaluation Setup

Victim Models. We evaluate our attacks on three state-of-the-art video compression models, i.e., DVC [3], HLVC [1], and FVC [2], quantized to 8 bits. The first video frame is always encoded using the DL-based image compression in [13].

Dataset. We use the Vimeo-90K dataset for training each victim model. We set the GOP size to 10 following prior works [1]–[3]. Our vulnerable bit detection and adversarial bit selection steps are performed using one randomly sampled GOP-sized video from the UCF-101 dataset [19] which has a different distribution compared to the training dataset. Finally, we evaluate the attacked video compression performance on the HEVC video sequences [5] from class B and C with 1920×1080 and 834×480 resolution, respectively.

B. Experimental Results

In this section, we provide the experimental results to demonstrate the effectiveness of our attacks. Fig. 3 shows the distortion versus bitrate for the video compression models on various datasets, before and after applying each VideoFlip attack. Here, video distortion is measured in PSNR and bitrate is measured in Bpp (bit per pixel). For all benchmarks, we set the total number of bit flips, i.e., number of VideoFlip attack iterations, to 10. We compare VideoFlip with a baseline attack that randomly flips 100 bits and show that without our vulnerable bit detection and adversarial bit search, the attack cannot succeed even with $10\times$ larger number of bit flips.

VideoFlip Attack Performance. As shown in Fig. 3-left, *DA* dramatically degrades the PSNR by an average of 22.6dB while maintaining the same bitrate as the benign model. Our evaluations confirm that *DA* on the receiver's decoder results in a similar drop in the PSNR compared to the *DA* on the sender modules. We can further observe that the attack causes a larger drop in the PSNR for smaller values of λ. This is because the compression rate decreases as λ grows, i.e., the video compression model encodes less information on the current frame, thus inherently reducing the effect of the video compression model on reconstruction quality.

Fig. 3-middle shows the performance of our *BA*. As seen, *BA* increases the bitrate for every reference PSNR, causing large network traffic and delays or reduced video quality for a similar bitrate as the benign model. Specifically, our attack increases the Bpp by up to $3.4\times$ on both HEVC class B and C datasets for a similar PSNR ($\pm 2\%$) as the benign video compression model. Note that *BA* can only be performed on the sender side since the receiver cannot control the communication bitrate.

Fig. 3-right demonstrates that our *BDA* significantly compromises both Bpp and PSNR of video compression. Compared to the PSNR drop caused by *DA*, our *BDA* results in 1.4dB higher

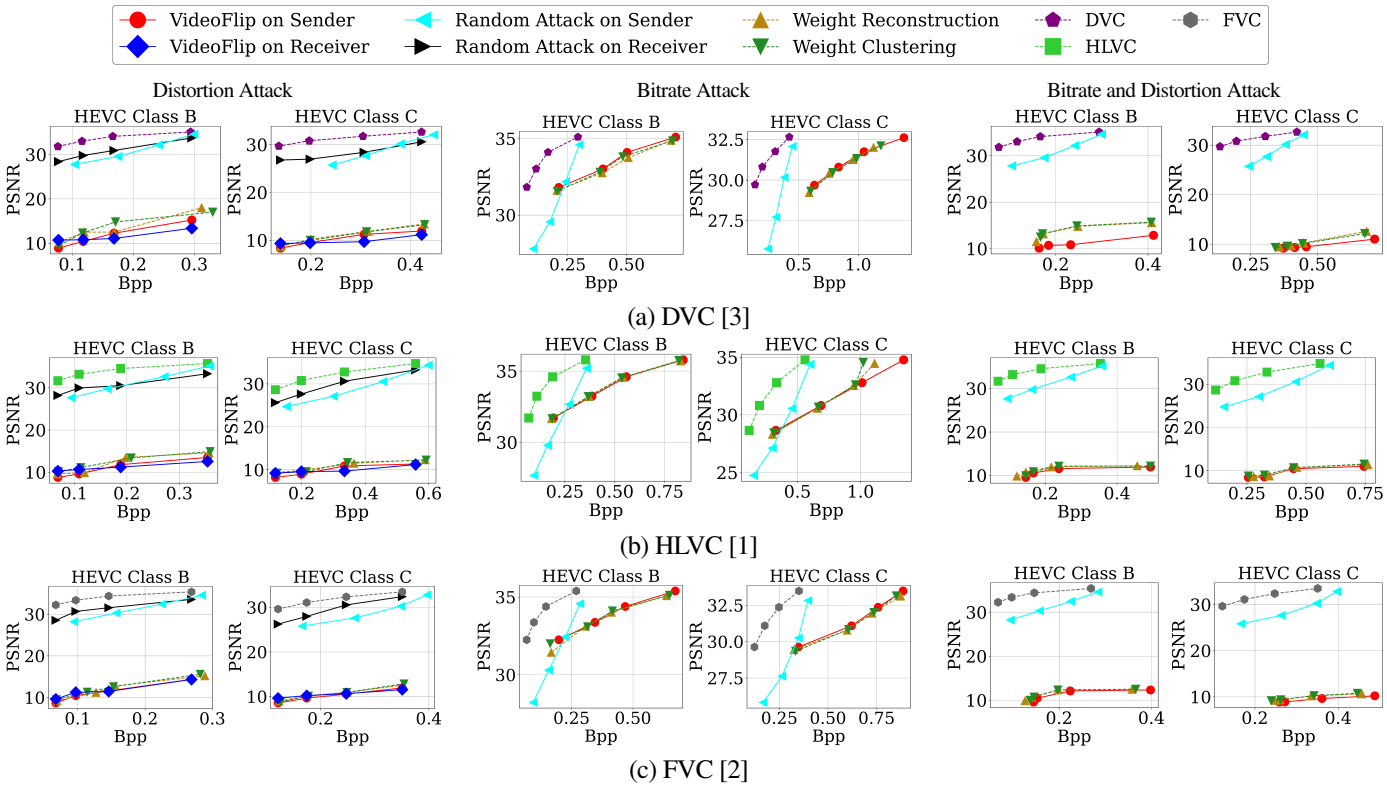
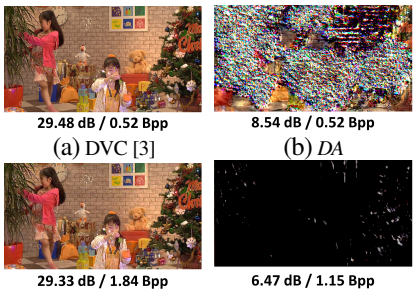


Fig. 3: VideoFlip attacks on DVC [3], HLVC [1], and FVC [2] video compression frameworks. Each scatter curve contains four points representing different compression ratios $\lambda \in \{256,512,1024,2048\}$.

distortion, on average. However, the BPP increase is on average $1.8\times$ lower compared with *BA*. This is intuitive since there exists a rate-distortion trade-off when determining the vulnerable bit for *BDA* in Eq. 6. The attacker can use a smaller value of λ_{avg} to put more emphasis on the bitrate, leading to similar Bpps as our *BA*.

Comparison with Baseline Attack. As shown in Fig. 3, flipping randomly selected bits performs very poorly compared to VideoFlip attacks. Specifically, the average PSNR drop and Bpp increase are only -2.5dB and $1.3\times$ for the random attack, compared to the -21.8dB and $2.9\times$ obtained by VideoFlip. Notably, the random attack is flipping $10\times$ more bits which leads to exceptionally high, if not infeasible, hardware realization costs for the attacker. The results here validate that DL-based video compression models are inherently robust to attacks and can only be compromised using our novel attack formulations that find vulnerable bits.

Resiliency to Defense. Recently proposed defenses, i.e. weight clustering [20] and weight reconstruction [21] add specific constraints in the training stage to improve the robustness of DL models against bit flip attacks in the image domain [10]. We enforce the same constraints when training our victim video compression model and remeasure the resiliency against VideoFlip attacks. As shown in Fig. 3, the integration of weights clustering and reconstruction comes at a cost of ($+1.2\%$Bpp, -1.4dB) and ($+1.4\%$Bpp, -1.6dB) change in the rate-distortion of the benign model even when the attack is not present. The defenses can recover only an average of 0.5dB when *DA* is carried out. Similarly, for our *BA*, the defenses cannot mitigate Bpp increase. The reason is that our attacks maximize error propagation in a GOP based on dependencies between



(a) DVC [3] (b) *DA* (c) *BA* (d) *BDA*

Fig. 4: Visualization of decoded frames when the sender's model weights are benign versus perturbed by VideoFlip.

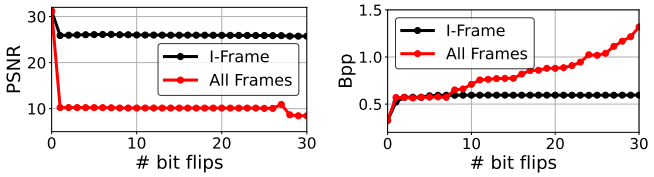
adjacent frames. In addition, bit flips cause error propagation between multiple modules. Therefore, rebuilding the model with training constraints is not effective as a defense against VideoFlip.

Attack Visualization. Fig. 4 demonstrates the effect of our sender attacks on the quality of the decoded videos on the receiver side. As seen, *DA* causes visibly large distortions in the decoded frame with little change in the Bpp. Similar distortions are observed when *DA* is applied to the receiver's weights. *BA* produces a similar video quality as the clean model but increases Bpp by $3.6\times$. *BDA* causes denial-of-service by hindering video reconstruction from the perturbed bitstream. In addition, *BDA* causes more distortion compared to *DA*, while requiring $2.2\times$ larger Bpp than the benign model.

C. Discussion and Analysis

Below we provide discussions on various design choices for implementing our attacks. Unless otherwise noted, evaluations are performed on the DVC compression model with HEVC class C dataset.

Attack to Image Compression. Recall from Section II-A that the first frame in the GOP, a.k.a, the I-frame, is compressed using DL-



(a) *DA* (b) *BA*

Fig. 5: Attack performance when flipping bits in the IC model (perturbing the I-frame) versus the VC model (perturbing all frames).

based image compression (IC). The subsequent frames are encoded by the video compression model (VC) while referring to the I-frame via the decoded frame buffer. Thus, the I-frame can greatly affect the overall coding performance. Building on this intuition, we evaluate a new attack where only the weights bits in the IC are changed. As shown in Fig. 5, attacking only the IC still degrades compression performance since the resulting perturbations created in the I-frame propagate to the next frames. However, the attack effectiveness is significantly less compared to VideoFlip method of flipping bits in the VC which adds strong perturbations to every frame in the GOP.

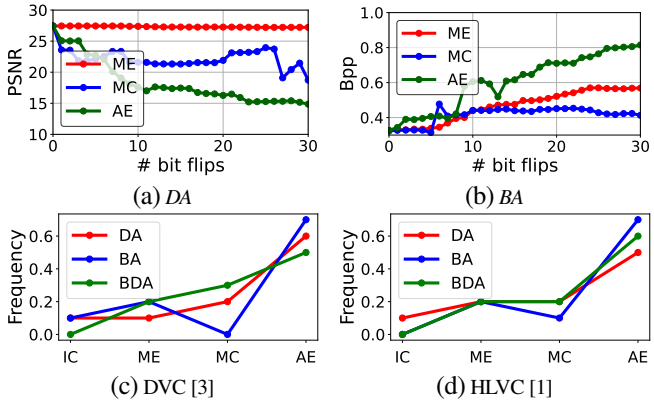
Attacks to Different Modules. Contemporary video compression models can be divided into different components for motion estimation (ME), motion compression (MC), and several auto-encoders (AE), each of which consists of multiple convolutional layers. We study the vulnerability of various components in Fig. 6 (a), (b) where bit flips are limited to only one component. As shown, the AE modules are the most vulnerable to VideoFlip attacks, resulting in the largest drop in PSNR and increase in Bpp. This is intuitive since AEs are the last component for generating the (perturbed) bitstream and the first component for reconstructing the frames from the bitstream. As such, the effect of AE bit flips on the intermediate model features easily propagates to all other components. As shown in Fig. 6 (c), (d), VideoFlip's vulnerable bit detection automatically learns which components to target to increase attack efficiency and effectiveness as most bit flips occur in AE layers.

VI. Conclusion

This paper presents VideoFlip, a novel hardware-based fault-injection attack on video compression models. Our attacks identify and alter a few vulnerable weight bits inside the video compression DL model to break the optimized balance between bitrate and distortion according to the adversary's goal. We formalize VideoFlip attacks such that the identified bits universally downgrade performance for any incoming video stream, irrespective of the content or the compression rate used. Our comprehensive evaluations show that our attack can successfully compromise state-of-the-art video compression models while maintaining robustness against various defense techniques.

Acknowledgement

The authors would like to thank Huili Chen and Seira Hidano for fruitful discussions. This work was supported in part by National Science Foundation (NSF) Trust-Hub under award number CNS-2016737 and the Intelligence Advanced Research Projects Agency (IARPA) under the contract W911NF20C0045. The content of this paper does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.



(a) *DA* (b) *BA*

(c) DVC [3] (d) HLVC [1]

Fig. 6: (a), (b): Vulnerability analysis of different components of a video compression pipeline to bit flip attacks. (c), (d): Normalized frequency of bit flips per component in VideoFlip attacks.

References

[1] R. Yang *et al.*, "Learning for video compression with hierarchical quality and recurrent enhancement," in *CVPR*, June 2020.

[2] Z. Hu *et al.*, "Fvc: A new framework towards deep video compression in feature space," in *CVPR*, June 2021, pp. 1502–1511.

[3] G. Lu *et al.*, "Dvc: An end-to-end deep video compression framework," in *CVPR*, June 2019.

[4] T. Wiegand *et al.*, "Overview of the h. 264/avc video coding standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003.

[5] G. J. Sullivan *et al.*, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[6] "Explorations: Neural network-based video compression. https://www.mpeg.org/standards/Explorations/36/."

[7] "How ai research is enabling next-gen codecs. https://www.qualcomm.com/news/onq/2021/07/14/how-ai-research-enabling-next-gen-codecs."

[8] S. Hong *et al.*, "Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks," in *USENIX Security Symposium*, 2019, pp. 497–514.

[9] F. Yao *et al.*, "{DeepHammer}: Depleting the intelligence of deep neural networks through targeted chain of bit flips," in *USENIX Security Symposium*, 2020, pp. 1463–1480.

[10] A. S. Rakin *et al.*, "Bit-flip attack: Crushing neural network with progressive bit search," in *CVPR*, 2019, pp. 1211–1220.

[11] H. Chen *et al.*, "Proflip: Targeted trojan attack with progressive bit flips," in *CVPR*, 2021, pp. 7718–7727.

[12] A. S. Rakin *et al.*, "Tbt: Targeted neural network attack with bit trojan," in *CVPR*, 2020, pp. 13 198–13 207.

[13] J. Liu *et al.*, "A unified end-to-end framework for efficient deep image compression," *arXiv preprint arXiv:2002.03370*, 2020.

[14] Y. Kim *et al.*, "Flipping bits in memory without accessing them: An experimental study of dram disturbance errors," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 3, pp. 361–372, 2014.

[15] M. Ribeiro *et al.*, "Mlaas: Machine learning as a service," in *International Conference on Machine Learning and Applications*. IEEE, 2015, pp. 896–902.

[16] J. Lin *et al.*, "A deeply modulated scheme for variable-rate video compression," in *International Conference on Image Processing*. IEEE, 2021, pp. 3722–3726.

[17] A. Tatar *et al.*, "Defeating software mitigations against rowhammer: a surgical precision hammer," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2018, pp. 47–66.

[18] I. Sodagar, "The mpeg-dash standard for multimedia streaming over the internet," *IEEE multimedia*, vol. 18, no. 4, pp. 62–67, 2011.

[19] K. Soomro *et al.*, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[20] Z. He *et al.*, "Defending and harnessing the bit-flip based adversarial weight attack," in *CVPR*, 2020, pp. 14 095–14 103.

[21] J. Li *et al.*, "Defending bit-flip attack through dnn weight reconstruction," in *ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.