Supplementary Material for the Paper: Over-the-Air Adversarial Flickering Attacks against Video Recognition Networks

Roi Pony[1]*, Itay Naeh[2]*, Shie Mannor[1,3]

[1]Department of Electrical Engineering, Technion Institute of Technology, Haifa, Israel
[2] Rafael - Advanced Defense Systems Ltd., Israel
[3] Nvidia Research
roipony@gmail.com, itay@naeh.us, shie@technion.ac.il

1. Modified Adversarial loss function

For achieving a more stable convergence, we used a loss mechanism similar to the loss presented by [1], with a small modification, which smoothly reaches the adversarial goal only to the desired extent, leaving space for other regularization terms. For untargeted attack:

$$\ell(y,t) = \max\left(0, \min\left(\frac{1}{m}\ell_m(y,t)^2, \ell_m(y,t)\right)\right) \quad (1)$$

$$\ell_m(y,t) = y_t - \max_{i \neq t}(y_i) + m. \quad (2)$$

$m > 0$ is the desired margin of the original class probability below the adversarial class probability. When loss values are within the desired margin, the quadratic loss term relaxes the relatively steep gradients and momentum of the optimizer, and the difference between the first and second class probabilities approach the desired margin m. When the loss starts rising, the quadratic term gently maintains the desired difference between these two classes, therefore preventing overshoot effects. In order to apply the suggested mechanism on targeted attack, the loss term changed to $\ell_m(y,t) = \max_{i \neq t}(y_i) - y_t + m$, while this time, t is the targeted adversarial class.

In some cases it would be beneficial to follow [1] and use the logits instead of the probabilities for calculating the loss. We suggest adapting this method partially by keeping the desired margin in probability space, normalized at each iteration accordingly, for margin defined in logit space may be less intuitive as a regularization term.

*Equal contribution

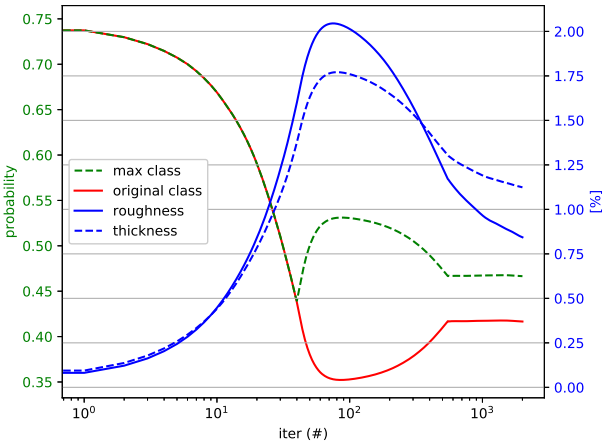


Figure 1: Learning process of the modified loss mechanism. Probabilities (green and red lines) corresponds to the left y-scale. Roughness and thickness (blue lines) are in percents from the full gray-level range of the image (right y-scale). *original class* is the probability of the actual class of the unperturbed video. *max class* is the probability of the most probable class as the classifier predicts.

2. Implementation Details

2.1. Experiments on I3D

Experiment codes are implemented in TensorFlow[1] and based on I3D source code[2]. The code is executed on a server with four Nvidia Titan-X GPUs, Intel i7 processor and 128GB RAM. For optimization we adopt the ADAM [2] optimizer with learning rate of 1e-3 and with batch size of 8 for the generalization section and 1 for a single video attack. Except where explicitly stated $\beta_1 = \beta_2 = 0.5$. For

[1]https://www.tensorflow.org/
[2]https://github.com/deepmind/kinetics-i3d

single video attack and for generalization sections $\lambda = 1$.

2.2. Experiments on MC3, R3D, R(2+1)D

Experiments code are implemented in PyTorch[3] and based on source code of computervision-recipes[4] and torchvision[5] package. Hardware, optimizer, batch size, β_1, β_2 and λ are the same as previously introduced for the I3D model.

3. Single Video Attack

3.1. Convergence Process

In order to demonstrate the convergence process we have attacked a single video. As can be seen, several trends regarding the trends can be observed (Figure 1). At first, the adversarial perturbation rises in thickness and roughness. At iteration 40 the top-probability class switches from the original to the adversarial class, which until now was not plotted, for this adversarial attack is untargeted. At that iteration, the adversarial loss is m. When the difference between the probability of the adversarial and original class is larger then m the adversarial loss is zero and the regularization starts to be prominent, causing the thickness and roughness to decay. This change of trend occurs slightly after the adversarial class change due to the momentum of the Adam optimizer and remaining intrinsic gradients. At iteration 600 the difference between the probability of the adversarial and original class is $m = 0.05$, the quadratic loss term maintaining the desired difference between these classes while diminishing the thickness and roughness. The binary loss changes at the interface between adversarial success and failure caused convergence issues, and the implementation of the quadratic term, as defined in Equation (1) handled this issue.

3.2. Thickness Vs. Roughness

In order to visualize the trade-off between β_1 and β_2 we plotted three graphs in Figure 4. In top and bottom graphs we see the temporal amplitude of the adversarial perturbation of each frame and for each color channel, respectively. The extreme case (top) of minimizing only D_1 (given success of the untargeted adversarial attack) and leaving D_2 unconstrained ($\beta_1 = 1, \beta_2 = 0$). The signal of the RGB channels fluctuates strongly with a thickness value of 0.87% and a roughness of 1.24%. The other extreme case (bottom) is when D_2 is constrained and D_1 is not ($\beta_1 = 0, \beta_2 = 1$), leading to a thickness value of 1.66% and a roughness value of 0.6%. The central image displays all the gradual cases between the two extremities: β_1 goes from 1 to 0, and β_2 from 0 to 1 on the y-axis. The row denoted by $\beta_2 = 0$ corresponds to the upper graph and the row denoted by $\beta_2 = 1$ corresponds to the lower graph. Both D_1 and D_2 are very dominant in the received perturbation, as desired. Visualization of the path taken by our loss mechanisms at different β_1 and β_2 values can be found in the supplementary material.

Apart from the visualization experiments we showed, another experiment have been conducted in order to visualize the path taken by our loss mechanism at different β_1 and β_2. We have plotted a 3D representation in probability-thickness-roughness space for 10 different experiments (10 different single video attack on the same video) with gradual change of β_1 and β_2 parameters. Figure 5 shows the probability of the most probable class at 10 different scenarios as described in the legend. One can see that at the beginning the maximal probability (original class) drops from the initial probability (upper section of the graph) on the same path for all of the described cases, until the adversarial perturbation takes hold of the top class. From there, the β's parameters takes the lead. At this point, each different case is converging along a different path to a different location on the thickness-roughness plane. The user may choose the desired ratios for each specific application.

4. Additional models, baseline comparison and transferability

4.1. Baseline comparison

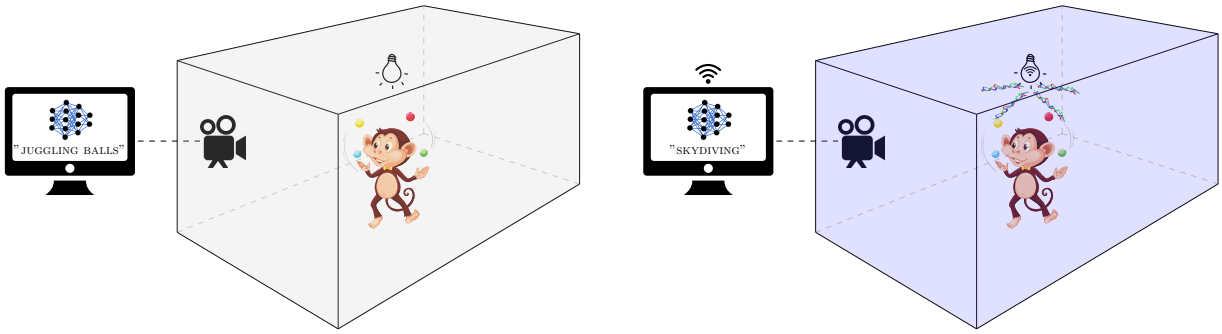
In addition to the table presented in the paper, we have analyzed our experiments from the attacked model perspective. Each Sub-figure in Figure 6 shows the average fooling ratio of the attacked model (out of four) with different perturbation as function of $\ell_\infty[\%]$. Each sub-figure combine three (two in I3D)[6] main graph types, the dashed graph represent the universal flickering perturbation developed upon the attacked model (δ^F), the dotted graphs represent the universal flickering attack developed upon other models (except for I3D) and the continues graphs represent the random generated flickering perturbation ($\delta^F_U, \delta^F_{MinMax}, \delta^F_{shuffle}$) where the shaded filled region is $\pm$ standard deviation around the average fooling ratio. Several consistent trends can be observed in each one of the sub-figure and thus for each attacked model. For each $\ell_\infty[\%]$ we can see that the fooling ratio order (high to low) is, first universal flickering attack, then the transferred universal flickering attack developed upon other models and finally, the random generated flickering perturbations.

[3] https://pytorch.org/

[4] https://github.com/microsoft/computervision-recipes

[5] https://github.com/pytorch/vision

[6] The transferabilty between I3D to the other models (and vice versa) were not evaluated because the input of the models is not compatible.



(a) Without over-the-air attack, the action recognition network classify the action correctly as "juggling balls".

(b) With over-the-air attack, the action recognition network classify the action incorrectly as "skydiving".

Figure 2: Room sketched of our over-the-air attack setup.

5. Over-the-Air Real world demonstration

Our goal is to produce an adversarial universal flickering attack, which will be implemented in the real world by an RGB led light bulb in a room, causing miss-classification. The desired scenario for the demonstration of the attack includes a video camera streaming a video filmed in a room with a Wifi-controlled RGB led light bulb. A computer sends over Wifi the adversarial RGB pattern to the bulb. A figure performs actions in front of the camera. The hardware specifications are as follows:

- **Video camera**: We used 1.3 MPixel RGB camera streaming at 25 frames per second.
- **RGB led light bulb**: In order to applying the digitally developed (univrsel or scene based) perturbation to the scene, we use a RGB led light bulb[7], controlled over Wifi via Python api[8], allowing to set RGB value at relatively high speed.
- **Computer**: We use a computer to run the I3D action classifier on the streaming video input. The model input for prediction at time t are all consecutive frames between $t - 90$ to t (as described in I3D experiments section). The model prediction frequency is set to 2Hz (hardware performance limit). In addition, we use the computer in order to control the smart led bulb.
- **Acting figure**: Performs the actions we would like to classify and attack.

Figure 2 demonstrate our over-the-air attack setup, combining the hardware mentioned above. Figure 2a demonstrate the state when the attack is off (no adversarial pattern is transmitted) and the video action recognition network correctly classify the action, while Figure 2b demonstrate the state when the attack is on (adversarial pattern is transmitted) and the video action recognition network incorrectly classify the action.

5.1. Over-the-Air Scene-based Flickering Attack

As described in the paper, in the scene-based approach we assume a prior knowledge of the scene and the action. In this approach we record a video without any adversarial perturbation of the scene we would like to attack. Then we develop a time-invariant digital attack for this recording as described in the paper. Once we have the digital attack, we transmit it to a "similar" scene in order to apply the attack in the real world as can be found here[9]. For illustrating the meaning of "similar" scene, we show in Figure 3 two frames, where Figure 3a is a frame example from the video (scene) which the attack was trained upon and Figure 3b is a frame example from the scene on which the developed attack was applied on. The relevant videos shows that even though the positioning is different and the clothing are not the same, the attack is still very effective even with a small perturbation.

References

[1] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017. 1

[2] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014. 1

Acknowledgements

Cartoon in Figure 2 designed by "brgfx / Freepik".

[7] https://www.mi.com/global/mi-led-smart-bulb-essential/specs/

[8] https://yeelight.readthedocs.io/en/latest/

[9] https://bit.ly/Over_the_Air_scene_based_videos

(a) Frame example from *"ironing"* video used for training over-the-air scene based attack.

(b) Frame example from *"ironing"* scene used for testing over-the-air scene based attack.

Figure 3: Two frames from "similar" scenes.

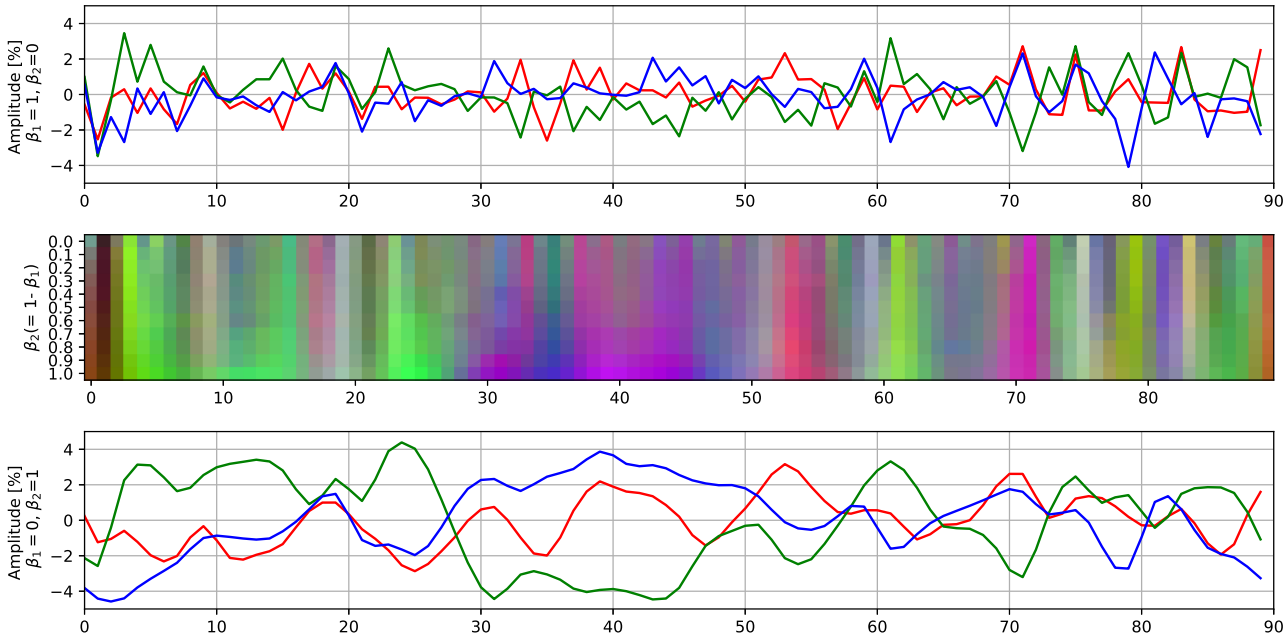


Figure 4: Top: The adversarial perturbation of the RGB channels (color represents relevant channel) as a function of the frame number at the case that $\beta_1 = 1$ and $\beta_2 = 0$ (D_1 minimization is preferred). Bottom: The adversarial perturbation of the RGB channels as a function of the frame number at the case that $\beta_1 = 0$ and $\beta_2 = 1$ (D_2 minimization is preferred). Top and bottom graphs are presented in percents from the full scale of the image. Middle: The gradual change of the adversarial pattern between the two extreme cases where $\beta_1 = 0$ corresponds to the top graph and $\beta_1 = 1$ corresponds to the bottom graph. Color (stretched for visualization purposes) represents the RGB parameters of the adversarial pattern of each frame.

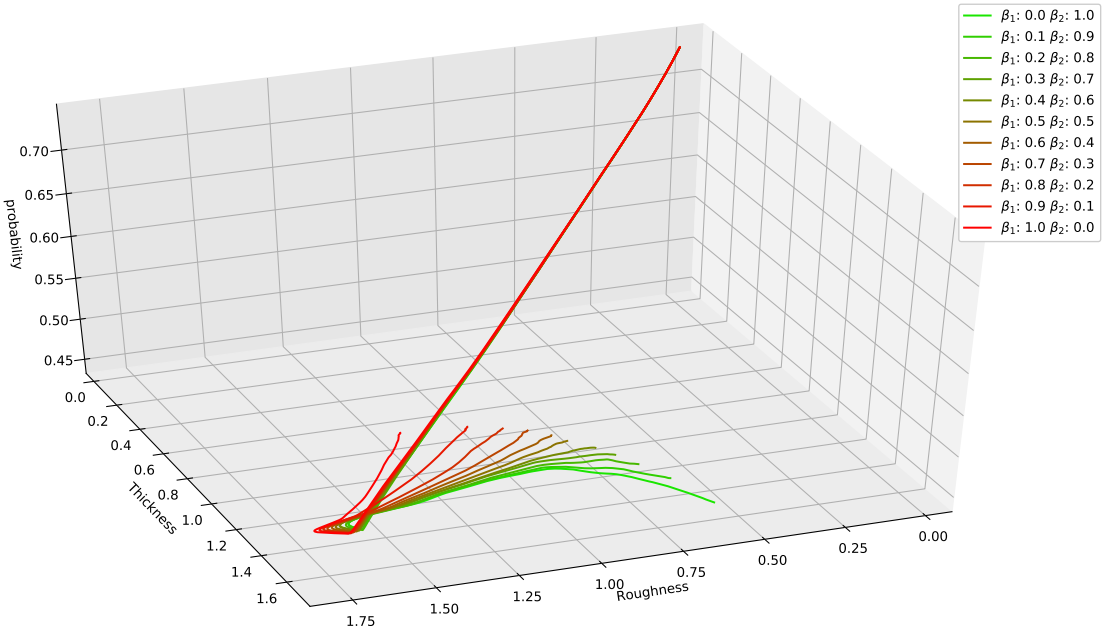
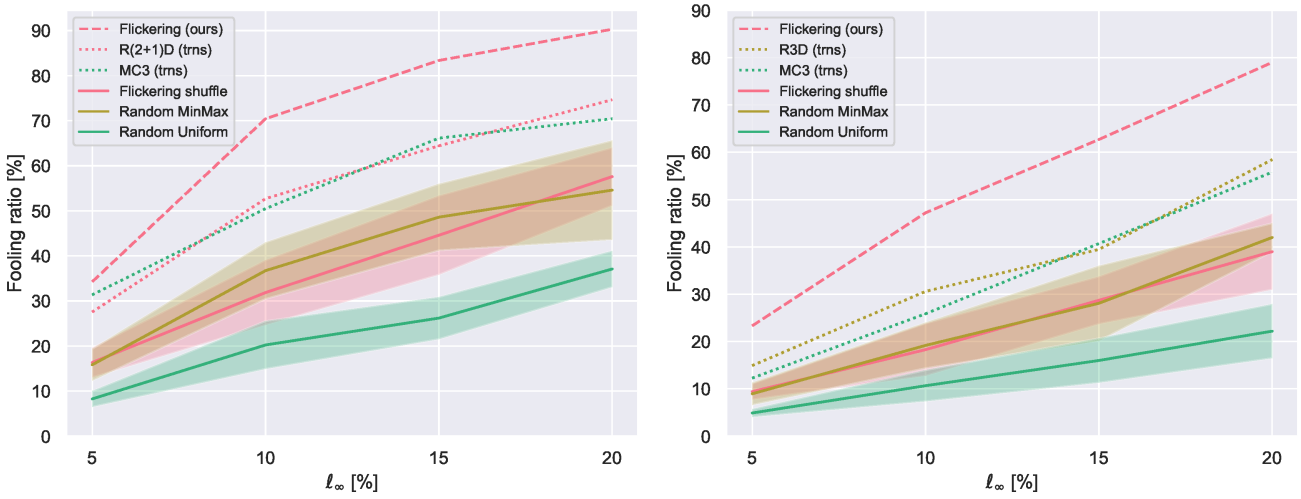
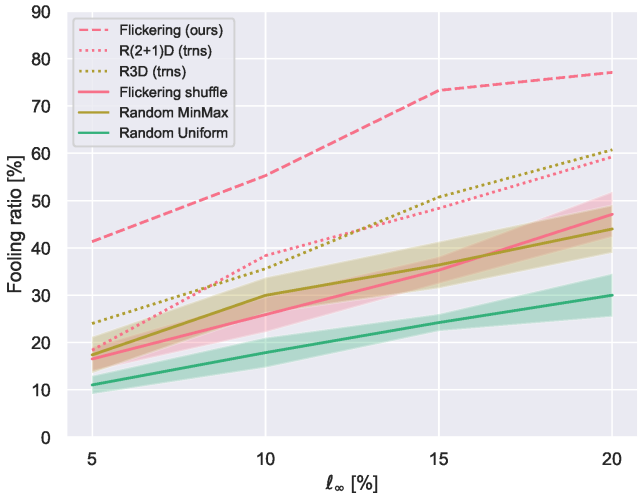


Figure 5: Convergence curve in probability-thickness-roughness space of an untargeted adversarial attack with different β_1 and β_2 parameters.

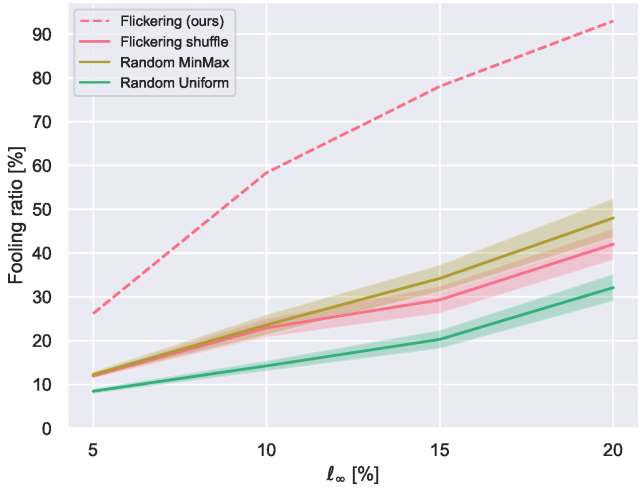


(a) R3D

(b) R(2+1)D



(c) MC3



(d) I3D

Figure 6: Each one of the sub-figures shows the average fooling ratio of the attacked model (described in caption) with different perturbations as a function of $\ell_\infty[\%]$. Each sub-figure combine three (two in I3D) main graph types, the dashed graph represent the Universal flickering perturbation developed on the attacked model (δ^F), the dotted graphs represent the universal flickering attack developed upon other models (except for I3D) and the continues graphs represent the random generated flickering perturbations ($\delta^F_U, \delta^F_{MinMax}, \delta^F_{shuffle}$) where the shaded filled region is $\pm$ standard deviation around the average Fooling ratio.