Appendix of Sparse Adversarial Video Attacks with Spatial Transformations

Mu, Ronghui[1]
ronghui.mu@lancaster.ac.uk
Ruan, Wenjie[2]
W.ruan@exeter.ac.uk
Soriano Marcolino,Leandro[1]
l.marcolino@lancaster.ac.uk
Ni,Qiang[1]
q.ni@lancaster.ac.uk

[1] Computing and Communications, Lancaster University Lancaster, UK
[2] Computer Science, University of Exeter, Exeter, UK

† This work was done while Ronghui Mu was visiting University of Exeter.
∗ Corresponding author.

A Calculating the Gradient of SSIM

The SSIM was first proposed in [1], and is detailed in [3]. Given x and $\hat{x}$ as the local pixels taken from the same location of the same frame in the clean video and adversarial video, respectively, the local similarity between them can be computed on three aspects: structures ($s(x,\hat{x})$), contrasts ($c(x,\hat{x})$), and brightness values ($b(x,\hat{x})$). The local SSIM is formed by these terms [3]:

$$S(x,\hat{x}) = s(x,\hat{x}) \cdot c(x,\hat{x}) \cdot b(x,\hat{x}) = \left(\frac{\sigma_{x\hat{x}} + D_1}{\sigma_x \sigma_{\hat{x}} + D_1}\right) \cdot \left(\frac{2\sigma_x \sigma_{\hat{x}} + D_2}{\sigma_x^2 + \sigma_{\hat{x}}^2 + D_2}\right) \cdot \left(\frac{2\mu_x \mu_{\hat{x}} + D_3}{\mu_x^2 + \mu_{\hat{x}}^2 + D_3}\right), \tag{1}$$

The structural similarity index (SSIM) measure in Equation (2) can be expressed as: [3]

$$\mathrm{SSIM}(\mathbf{x},\hat{\mathbf{x}}) = \frac{(2\mu_x \mu_{\hat{x}} + C_1)(2\sigma_{x\hat{x}} + C_2)}{(\mu_x^2 + \mu_{\hat{x}}^2 + C_1)(\sigma_x^2 + \sigma_{\hat{x}}^2 + C_2)} \tag{2}$$

The mean of x , the variance of x, and co-variance of x and $\hat{x}$ can be represented as μ_x, σ_x^2 and $\sigma_{x\hat{x}}$. They can be calculated respectively:

$$\begin{aligned} \mu_x &= \frac{1}{N_P}\left(\mathbf{1}^T \cdot \mathbf{x}\right) \\ \sigma_x^2 &= \frac{1}{N_P - 1}(\mathbf{x} - \mu_x)^T (\mathbf{x} - \mu_x) \\ \sigma_{x\hat{x}} &= \frac{1}{N_P - 1}(\mathbf{x} - \mu_x)^T (\hat{\mathbf{x}} - \mu_{\hat{x}}) \end{aligned} \tag{3}$$

Given x and $\hat{x}$ as the local pixels taken from the same location of the same frame in the clean video and adversarial video, respectively, the local similarity between them can be computed on three aspects: structures ($s(x,\hat{x})$), contrasts ($c(x,\hat{x})$), and brightness values ($b(x,\hat{x})$). The local SSIM is formed as [3]:

$$S(x,\hat{x}) = s(x,\hat{x}) \cdot c(x,\hat{x}) \cdot b(x,\hat{x}) = \left(\frac{\sigma_{x\hat{x}} + D_1}{\sigma_x \sigma_{\hat{x}} + D_1}\right) \cdot \left(\frac{2\sigma_x \sigma_{\hat{x}} + D_2}{\sigma_x^2 + \sigma_{\hat{x}}^2 + D_2}\right) \cdot \left(\frac{2\mu_x \mu_{\hat{x}} + D_3}{\mu_x^2 + \mu_{\hat{x}}^2 + D_3}\right), \tag{4}$$

where μ_x and $\mu_{\hat{x}}$ denote means, σ_x and $\sigma_{\hat{x}}$ are standard deviations of x and $\hat{x}$, respectively; $\sigma_{x\hat{x}}$ represents the cross correlation of x and $\hat{x}$ after deleting means; D1, D2, and D3 are weight parameters. For SSIM metric, a value of 1 means that the two images compared are the same. As the SSIM is calculated based on pixel level, it use a sliding window method, which moves pixel by pixel by the window across the whole image. As we use uniform pooling to combine the total SSIM for the whole videos, suppose we have N pixels in the total videos, the SSIM can be represented as:

$$\mathrm{SSIM}(\mathbf{X},\hat{\mathbf{X}}) = \frac{\sum_{i=1}^{N} \cdot \mathrm{SSIM}(\mathbf{x}_i,\hat{\mathbf{x}}_i)}{N} \tag{5}$$

where x_i and $\hat{x}_i$ are the i-th pixel of each frame in the video. To apply the gradient decent optimisation method described in Section 3, we have to compute the gradient of SSIM with respect to the adversarial video example $\hat{\mathbf{X}}$. As equation (9) shows, to compute $\vec{\nabla}_{\hat{\mathbf{X}}} \mathrm{SSIM}(\mathbf{X},\hat{\mathbf{X}})$,

Models	λ value	FR	ASP(SSIM)
CNN+LSTM	0.8	56.94%	0.0429
	1.0	56.94%	0.0412
	1.5	56.94%	0.0401
I3D	0.8	51.22%	0.0316
	1.0	48.78%	0.0268
	1.5	48.17%	0.0198
Inception-v3	0.8	66.05%	0.0534
	1.0	65.14%	0.0518
	1.5	64.22%	0.0454

Table 1: The results of DeepSAVA(without BO) on UCF101 dataset for different λ values.

we only need to calculate the gradient $\vec{\nabla}_{\hat{x}_i} \mathrm{SSIM}(x_i, \hat{x}_i)$. The process is represented as follows. [2] We define four parameters to deduce the derivative of local SSIM:

$$\begin{aligned} M_1 &= 2\mu_x\mu_{\hat{x}} + C_1, \quad M_2 = 2\sigma_{x\hat{x}} + C_2 \\ P_1 &= \mu_x^2 + \mu_{\hat{x}}^2 + C_1, \quad P_2 = \sigma_x^2 + \sigma_{\hat{x}}^2 + C_2 \end{aligned} \tag{6}$$

Therefore, the gradient can be expressed as:

$$\vec{\nabla}_{\hat{x}} \mathrm{SSIM}(x, \hat{x}) = \frac{2}{N_P P_1^2 P_2^2} \left[M_1 P_1 \left(M_2 x - P_2 \hat{x} \right) + P_1 P_2 \left(M_2 - M_1 \right) \mu_x + M_1 M_2 \left(P_1 - P_2 \right) \mu_{\hat{x}} \right] \tag{7}$$

B Effects of λ

To decide the value of λ, we applied the DeepSAVA without BO selection on 200 random selected videos of UCF101 dataset to evaluate the effect of λ. The average success perturbation (ASP) is the average of the SSIM score of perturbation for the adversarial examples that could mislead the model successfully:

$$ASP(SSIM)) = avg(SSIM(V_{adv} - V_{original})),$$

where V_{adv} denotes the generated adversarial video that could successfully mislead the classifier and $V_{original}$ is the original video. The results of applying $\lambda = 0.8, 1.0, 1.5$ on three models are presented in Table 1. We can see that the bigger the λ, the lower the FR while the lower the perturbation.While, for the CNN+LSTM model, the fooling rate remains the same across all tested λ values, but the perturbation level is the lowest at λ = 1.5. Thus, we choose $\lambda = 1.5$ for the CNN+LSTM model and $\lambda = 1.0$ for I3D and Inception-v3 model to trade off the performance in terms of the fooling rate and average success perturbation.

C Average Absolute Perturbation

Average Absolute Perturbation can be introduced to measure the perturbation level for each method. As mentioned in the paper, the sparse Flickering adds a small perturbation per frame, but cannot obtain comparable results to ours (Figure 4 in the paper). Thus, we will choose the pure sparse attack (Sparse) as the main baseline to show the average absolute perturbation. As the baseline is guided by $l_{1,2}$ norm and ours is based on SSIM loss, we will

Models	Attack Method	UCF101			
		FR	ANI	AAP($l_{1,2}$)	AAP(SSIM)
CNN+LSTM	Sparse	54.31%	15.31	0.054	0.043
	DeepSAVA(without BO)	56.94%	7.87	0.077	0.060
	DeepSAVA(BO)	57.11%	8.01	0.071	0.058
I3D	Sparse	11.22%	49	0.092	0.079
	DeepSAVA(without BO)	48.78%	11.34	0.0857	0.054
	DeepSAVA(BO)	99.89%	5.74	0.055	0.0233
Inception-v3	Sparse	41.84%	38.21	0.062	0.0512
	DeepSAVA(without BO)	65.14%	14.88	0.072	0.052
	DeepSAVA(BO)	77.49%	11.43	0.071	0.0508

Table 2: Comparison with Sparse baseline, DeepSAVA without BO and with BO on different models by only perturbing one frame. Gray cell shows the best results.

Approach	CNN+LSTM		Inception-v3		I3D		time (s)
	FR	average loss	FR	average loss	FR	average loss	
BO Selection	55.81%	0.21	72.22%	3.39	100%	1.35	16.1
brute force search	55.81%	0.27	72.22%	3.39	100%	1.35	70.4

Table 3: Fooling Rate, average selected maximum loss and average time spent for one video of BO Selection and brute force search.

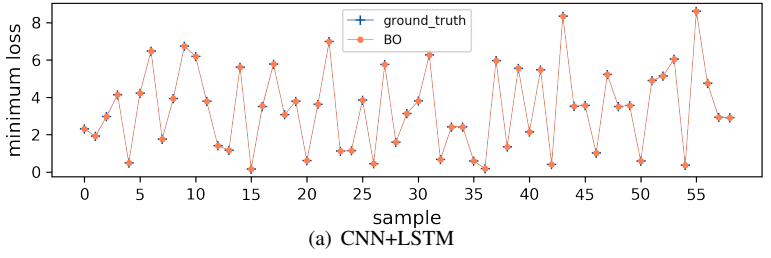
record the average perturbation of $l_{1,2}$ and SSIM separately. To achieve a fair comparison, we set the maximum $l_{1,2}$ norm ball constraint as 0.1 and maximum SSIM constraint as 0.92. Suppose the fooling rate is f, and distant matrix is D, which can be set to (1-SSIM) or $l_{1,2}$ norm, thus the average absolute perturbation(AAP) can be represented as:

$$AAP(D) = \frac{\sum_N D(V_{adv} - V_{original})}{N} * f + D_{max} * (1 - f),$$

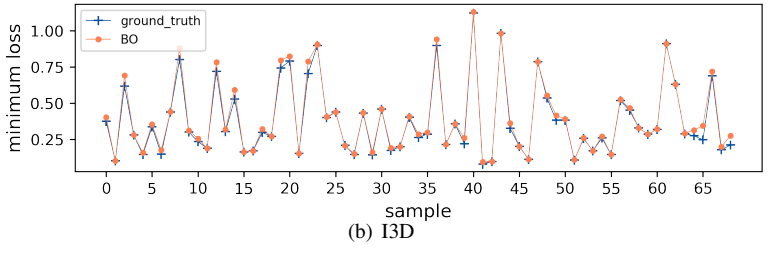
where V_{adv} denotes the generated adversarial video that could successfully mislead the classifier and D_{max} is the maximum constraint; N is the number of adversarial samples achieving successful attack. We run experiments on 200 random selected videos of UCF101 dataset and record the results of FR, ANI, AAP($l_{1,2}$) and AAP(SSIM) in Table 2.

D The accuracy of Bayesian Optimisation Selection

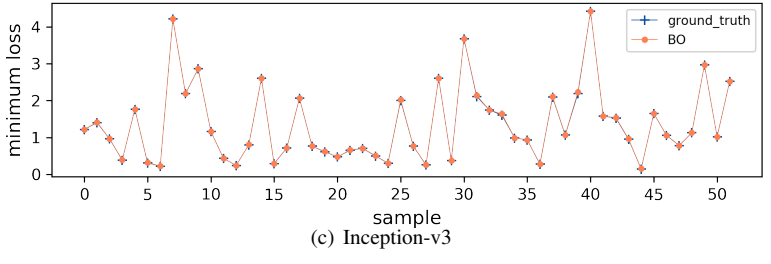
To justify whether the Bayesian Optimisation could select the most critical frames, we take the brute force search experiments to obtain the upper bound of the performance: when the selection frame is 1, we select the key frame manually one by one of the video, and then record the maximum loss found by the search. We randomly select 100 videos from UCF101 in different categories. The fooling rates, average maximum loss and average time spent for one video on three models are shown in the Table 3. We also compare the selected maximum loss by BO and brute force search along the video samples in Figure 1. We can see that we can obtain the same results as the brute force search, but spending much less time.



(a) CNN+LSTM



(b) I3D



(c) Inception-v3

Figure 1: Minimum loss selected by BO and brute force search along videos

E Comparison experiments with lp-norm and SSIM constraints

E.1 I3D model

Constraint	$l_{2,1}$ budget = 0.1			SSIM budget = 0.94		
Methods	Sparse	DeepSAVA(no BO)	DeepSAVA	Sparse	DeepSAVA(no BO)	DeepSAVA
FR	60%	58.22%	**91.25%**	12.9%	66.2%	**97.46%**
Time (s)	24029.8	4109.78	**1483.96**	30803.2	8276.1	**1586.3**

Table 4: Comparison with Sparse baseline method

E.2 CNN+LSTM model

Constraint	$l_{2,1}$ budget = 0.08			$l_{2,1}$ budget = 0.1		
Methods	Sparse	DeepSAVA(no BO)	DeepSAVA	Sparse	DeepSAVA(no BO)	DeepSAVA
FR	**55.71**%	48.57%	50%	**58.57**%	**58.57**%	57.14%
Time (s)	22800.5	**13777.6**	15010	23336.8	**19774.4**	20866.4

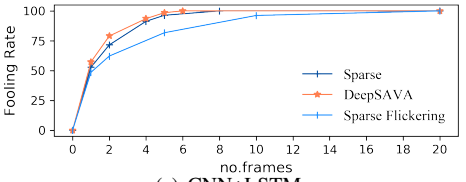
Table 5: Attack CNN+LSTM model on UCF101

Constraint	*SSIM* budget = 0.96			*SSIM* budget = 0.94		
Methods	Sparse	DeepSAVA(no BO)	DeepSAVA	Sparse	DeepSAVA(no BO)	DeepSAVA
FR	**50**%	47.14%	47.14%	**52.85**%	**52.85**%	**52.85%**
Time (s)	15120.21	**4039.24**	5131.24	15952.52	**6341.7**	7433.3

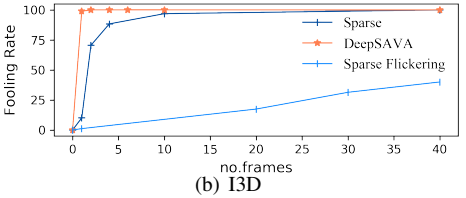
Table 6: Attack CNN+LSTM model on UCF101

For the CNN+LSTM model, we can see that although the Sparse baseline could obtain higher fooling rate, but it will spent much more time to generate the adversarial examples. Our method using combined perturbation will spent less time and obtain comparable fooling rate. Because here we select the first frame to attack, the BO cannot improve the performance as I3D model.

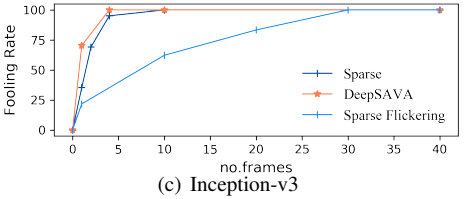
F Fooling Rate of attacking different number of frames across models.



(a) CNN+LSTM



(b) I3D



(c) Inception-v3

Figure 2: Fooling Rate of attacking different number of frames across different model.

G The effects of number of maximum allowed iteration

Here we show the relationship between the iteration, $l_{1,2}$, SSIM, and Fooling Rate for the I3D model with combine perturbation. It can be seen that set maximum iteration as 100 will not lead to a large distortion.

max iter	FR	max(lp)	max(ssim)	ave(lp)	ave(ssim)
30	0.5	0.11	0.094	0.052	0.069
50	0.5	0.135	0.094	0.059	0.099
80	0.529	0.131	0.0959	0.0595	0.081
100	0.529	0.131	0.095	0.052	0.067

H Transferability across RNN and CNN models

In Table 7, we studied the transferability from RNN models to CNN models. As the adversarial videos generated from the RNN models in rows are fed into the Inception-v3 and I3D models in columns to perform the black-box attack. In Table 8, we utilised the adversarial videos generated from Inception-v3 and I3D models (models in rows) to attack other models in columns. From the fooling rates shown in tables, we could see that the transferability from RNNs to CNNs is not as good as that from CNNs to RNNs. As in Table 8, the fooling rates to attack RNN models are higher than those to attack the I3D and Inception-v3 models. While that happens maybe because the I3D model has the highest training accuracy, it engages the lowest fooling rate when performing the black-box attack on it. As we can also conclude that due to the lowest training accuracy the CNN+Vanilla RNN model obtains, it achieves the highest fooling rate on attacking unseen Vanilla RNN model. Overall, compared with the Sparse baseline, our method could achieve better transferability.

Models	CNN+LSTM		CNN+Vanilla RNN		CNN+GRU	
	Sparse	DeepSAVA	Sparse	DeepSAVA	Sparse	DeepSAVA
Inception-v3	22.95%	24.36%	22.80%	26.72%	22.90%	31.03%
I3D	6.56%	10.08%	7.01%	9.48%	7.63%	8.62%

Table 7: Fooling Rate across CNN models on UCF101.

Models	Inception-v3		I3D	
	Sparse	DeepSAVA	Sparse	DeepSAVA
Inception-v3	100%	100%	33.61%	37.80%
I3D	13.04%	14.13%	100%	100%
CNN+LSTM	50.0%	52.17%	53.48%	54.62%
RNN	71.74%	82.40%	60.50%	64.02%
GRU	50.0%	51.08%	42.68%	49.58%

Table 8: Fooling Rate across CNN and RNN models on UCF101.

I Generated adversarial examples

The adversarial videos generated by DeepSAVA are demonstrated anonymously by https://www.youtube.com/channel/UCBDswZC2QhBhTOMUFNLchCg

I.1 target model: I3D

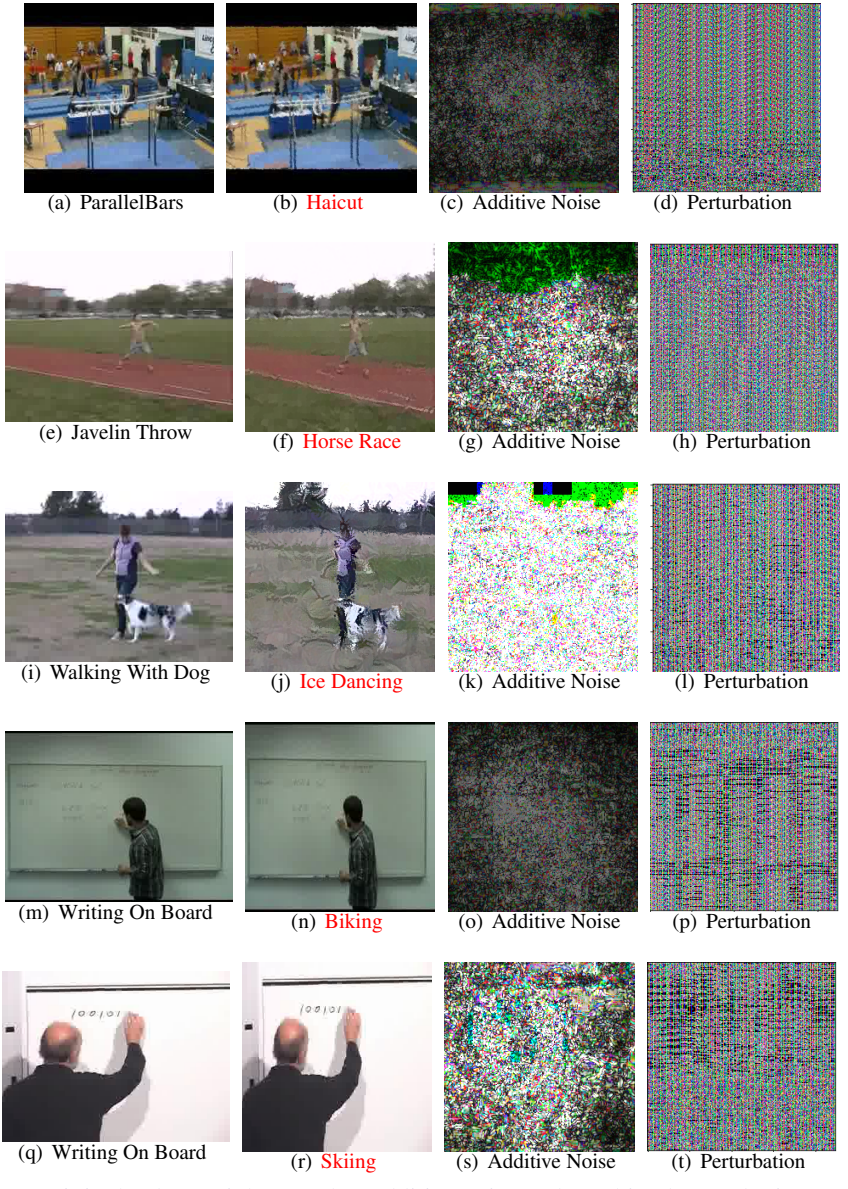


Figure 3: Original, adversarial examples, additive noise and combined perturbation (normalized into $[0, 1]$ for the visualization) when **only one frame** in the video is perturbed. The red labels are the wrong predictions based on videos.

I.2 target model : CNN+LSTM

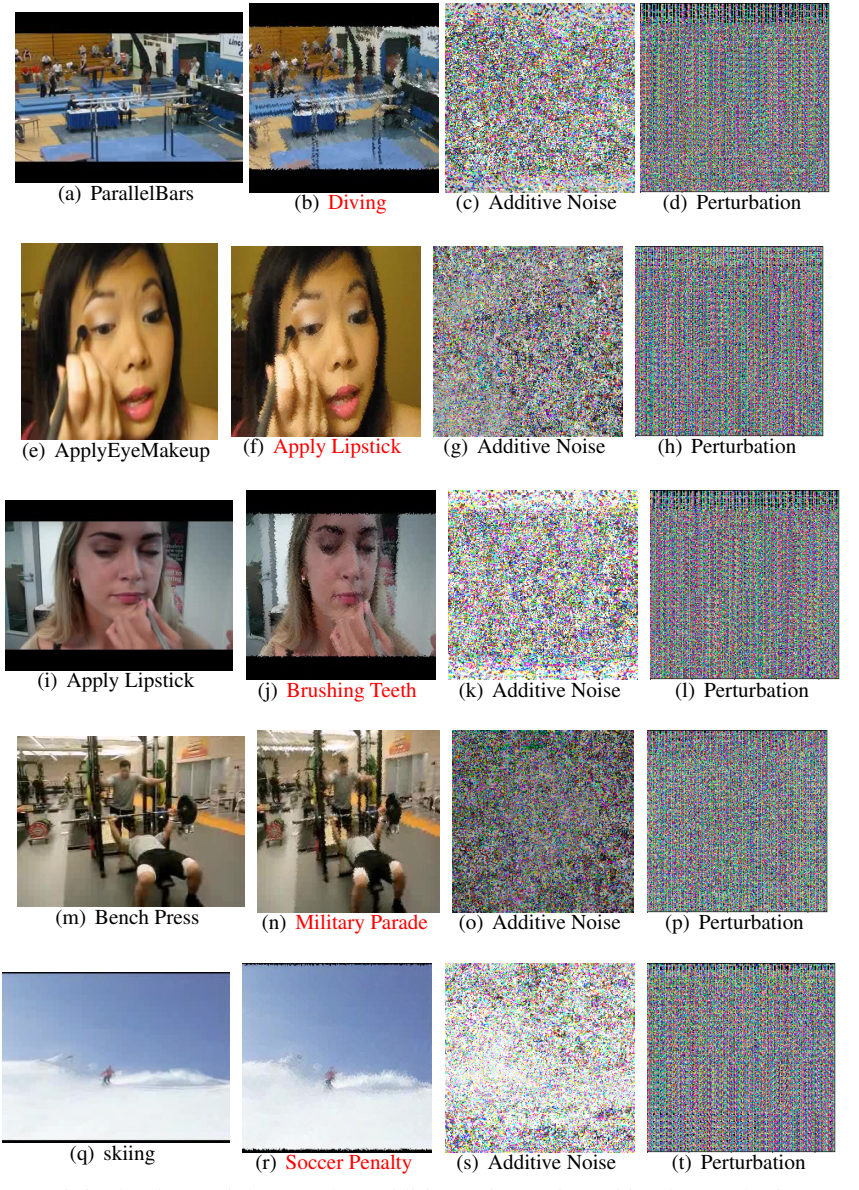


Figure 4: Original, adversarial examples, additive noise and combined perturbation (normalized into $[0, 1]$ for the visualization) when **only one frame** in the video is perturbed. The red labels are the wrong predictions based on videos.

I.3 target model:Inception-v3

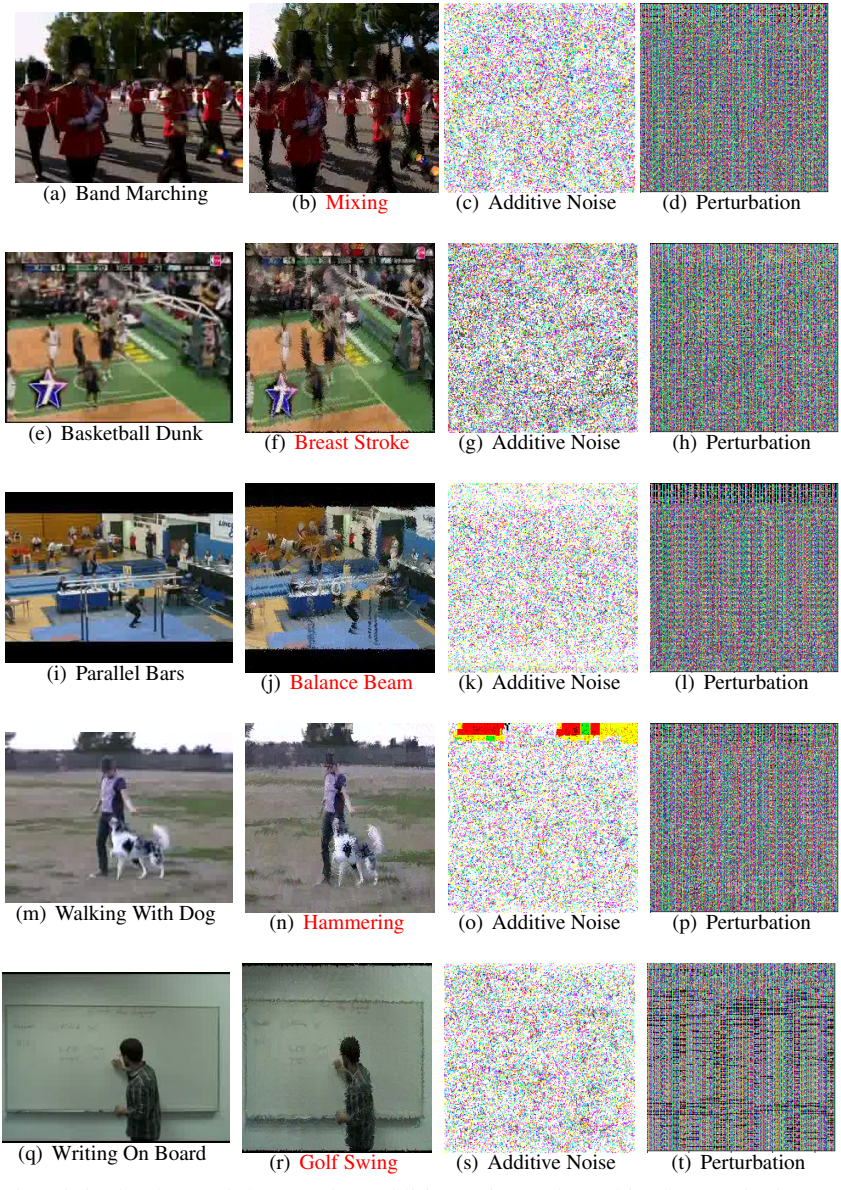


Figure 5: Original, adversarial examples, additive noise and combined perturbation (normalized into $[0, 1]$ for the visualization) when **only one frame** in the video is perturbed. The red labels are the wrong predictions based on videos.

References

[1] Zhou Wang and Alan C Bovik. A universal image quality index. *IEEE signal processing letters*, 9 (3):81–84, 2002.

[2] Zhou Wang and Eero P Simoncelli. Stimulus synthesis for efficient evaluation and refinement of perceptual image quality metrics. In *Human Vision and Electronic Imaging IX*, volume 5292, pages 99–108. International Society for Optics and Photonics, 2004.

[3] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.